Enriching Frame-based Structured Representations for Requirements using Case Frames
An Approach Towards Handling Incompleteness in Informal Requirements

Akanksha Mishra[1] and Richa Sharma[2]
[1]*Indraprastha Institute of Information Technology, Delhi, India*
[2]*BML Munjal University, Gurgaon, India*

Keywords: Requirements Engineering, Incompleteness, Quality Attributes, Requirements, Case Frames, FrameNet.

Abstract: Requirements gathered during early phase of requirements engineering are informal and vague. These informal requirements are analyzed with the goal of detecting three major problems in requirements – ambiguity, inconsistency, and incompleteness in order to arrive at correct and formal set of requirements. These problems are quite intertwined, with one problem leading to another. Incompleteness in requirements, however, is considered to be a principal reason for poor quality of requirements, and is the most difficult issue to address. There are multiple views around defining and detecting incompleteness in requirements. In this paper, we present an approach towards handling incompleteness in informal requirements considering individual requirements statement expressed in natural language as an atomic requirement. Our approach is based on enriching frame-based structured representation using FrameNet database that, in turn, can prove useful in identifying potential missing information from requirements. We also report our observations from the evaluation study conducted with a case study.

1 INTRODUCTION

Requirements Engineering (RE) is the most crucial and critical phase in software development as rest of the successive phases depend on the quality of requirements gathered and analysed during RE. Though there is no precise criterion for defining good quality of requirements but an abundant work in context of requirements quality (Saavedra et al., 2015; Firesmith, 2003; Zowghi and Gervasi, 2002; Fabbrini et al., 2001) identifies completeness, consistency, verifiability, non-ambiguity, and traceability as some of the important indicators of good quality of software requirements.

Completeness of requirements is relatively hard to address among other indicators of requirements quality. There are various differing propositions (Kuchta, J., 2016, Génova et al., 2013, ISO/IEC/IEEE International Standard, 2011; Pohl, 2013; Firesmith, 2005, Durán et al., 2001) on the definition and measurement of completeness of requirements. However, there is an agreement on two points: (a) we cannot achieve absolutely complete requirements (Carson and Shell, 2001); (b) completeness of requirements is related to other indicators of requirements quality (Saavedra et al., 2015). Ambiguity in requirements statements, for instance, could possibly be there because of incompleteness in the gathered requirements. Similar such interference of requirements completeness exists with consistency and correctness of requirements. This leads to concluding that addressing completeness can help addressing other quality indicators of requirements though absolute completeness cannot be achieved in requirements specifications.

The challenge in addressing the completeness concern with informal requirements gathered during early phases of RE lies in understanding what completeness of requirements mean, and how to ensure that completeness is achieved. These challenges are the motivating factors behind our work presented in this paper. We study completeness concern in requirements with respect to atomic (individual) requirements statement. Our approach makes use of frame-based structured representation (Bhatia et al., 2013, Sharma, 2016) of the requirements statement under study, and checks for related (possibly missing) information by

Mishra, A. and Sharma, R.
Enriching Frame-based Structured Representations for Requirements using Case Frames - An Approach Towards Handling Incompleteness in Informal Requirements.
DOI: 10.5220/0006379103140319
In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2017)*, pages 314-319
ISBN: 978-989-758-250-9

looking up for relevant frame for a key concept from the requirements statement in the FrameNet (Atkins et al., 1988; Fillmore et al., 2003) database.

The rest of the paper is organized as: section 2 briefly discusses related work towards handling incompleteness concern in requirements. In section 3, we present background of the concepts used in our approach followed by the proposed approach and case studies conducted to verify the feasibility of our proposed approach in section 4. Section 5 finally presents conclusion and future work.

2 RELATED WORK

As introduced in section 1, completeness in requirements is acknowledged as an important criterion for establishing high quality of requirements. However, completeness attribute of requirements quality has been interpreted differently by different authors (Kuchta, J., 2016, Génova et al., 2013, ISO/IEC/IEEE International Standard, 2011; Pohl, 2013; Firesmith, 2005, Durán et al., 2001, Boehm, 1984) in their work. Davis (1993) too has pointed out that it is difficult to precisely define *completeness* of requirements.

As per IEEE standard 29148:2011 (ISO/IEC/IEEE International Standard, 2011), a requirements specification document is said to be complete if: (i) the stated requirement or a set of requirements need no further amplification; (ii) the stated requirement is measurable, and it sufficiently describes the capability and characteristics to meet the stakeholder's need, and (iii) there is no TBx (To Be Defined/Specified/Resolved) item in the specification document. However, identifying whether a requirement statement or a set of requirements need further amplification remains a practical challenge. Boehm (1984) has earlier discussed the completeness is terms of: (i) internal completeness, and (ii) external completeness. Here, internal completeness emphasizes that no information in the document should be left unstated or to be determined. External completeness states that there should be no missing information. But, it is difficult to find ‘missing information’ without the knowledge or idea that something is ‘missing’ in the requirement statement or set of requirements.

Durán et al. (2001), in their work on XML-based approach for automated verification of software requirements, emphasize page numbering and the presence of referenced material as the defining criteria for completeness of requirements. Their viewpoint on requirements completeness is primarily driven by their solution approach for requirements verifiability. Génova et al. (2013) and Pohl (2013) share similar views on completeness concern that all relevant requirements must be specified. Firesmith (2005) defines *completeness* in terms of requirements models, namely – context models, data models, decision models, formal models, state models, and use case models. Kutcha (2016) has proposed metrics for Software Requirements Specifications (SRS) in terms of formal model of requirements, missing semantic elements, and missing references.

Most of the earlier works (Zowghi and Gervasi, 2002; Sutcliffe and Maiden, 2002) in the direction of addressing completeness concern in requirements recommend the domain knowledge as an assistive tool for uncovering ‘missing information’ from requirements. However, domain knowledge is often not available in the form of clear and well-structured documents that can be referred to. The absence of domain knowledge indicates the need for some other source of knowledge that can assist in detecting if there is some missing information from the requirements.

Our contribution lies in detecting the presence of ‘missing information’, i.e. external completeness as indicated by Boehm (1984), using the existing knowledge base of FrameNet. Our approach strives to find missing semantic elements as proposed by Kutcha (2016). The key concept to be searched for in FrameNet database is selected from the frame-based structured representation of the requirements statement under study. We present our proposed approach in detail in section 4. The background concepts used in our proposed approach are discussed in the following section.

3 BACKGROUND

In this section, we present background concepts that we have used in our proposed approach towards enriching requirements statements expressed in Natural Language (NL) using additional knowledge components/concepts (frame elements) from FrameNet lexical database. Our contribution lies in finding lexical units from the requirements statement that act as reference pointers for evoking frame(s) from FrameNet. The lexical units are extracted by converting NL requirements statement to its corresponding frame-based structured representation (Bhatia et al., 2013; Sharma, 2016). Following sub-section presents a brief overview of these structured representations.

3.1 Frame-based Structured Representation of Requirements

Frame-based structured representation (FBSR) of requirements, proposed by Bhatia et al. (2013) and refined in the work of Sharma (2016), is a structured representation of NL requirements statement in the form of frames (Minsky, 1981). These frames store information elements from the requirements statement as key-value pairs, where each key represents syntactic units present in the statement.

The authors have proposed seven different types of frame structures based on the Grammatical Knowledge Pattern (Marshman et al., 2002) present in the NL requirements statement, namely: (a) Active Vice frame, (b) Passive Voice frame, (c) Conjunction frame, (d) Preposition frame, (e) Precondition frame, (f) Marker frame, and (g) Relative Clause frame. These frame-based structured representations of NL requirements statement capture the semantics of the statement in the form of union of the above-mentioned frame structures. The advantage of using FBSR form of NL requirements is that the process of generating FBSR does take care of anaphora ambiguity and coordination ambiguity (Sharma, 2016).

Let us consider a sample requirements statement, RS1, to show how FBSR of NL requirements captures the semantic of NL requirements statement in the form of frame keys:

RS1: *If a person is not a member of library then the person cannot borrow the book.*

Table 1 below illustrates the FBSR of RS1, which is a union of three types of frames identified for RS1:

Table 1: Frame Structure – RS1.

FRAME KEY	VALUES
Active Voice	
Actor	person
Action	borrow
Neg Action	Not
Object	Book
Marker	
Marker	if
Actor	person
Actor Modifier	member
Neg Actor Mod	not
Action	-
Preposition	
Preposition	Of
Preposition Object	library
Governing Object	member

The frame-based structured representation of NL requirements statements can be used for automated reasoning, refining, and reusing the knowledge of requirements statement stored in its corresponding FBSR. We have used FBSR representations of NL requirements, in our study, for refining them after deriving additional related and relevant knowledge from FrameNet. We present a brief overview of FrameNet in the following sub-section.

3.2 FrameNet

FrameNet is a lexical database of words or phrases in NL to describe how words or phrases are used in NL statement through annotated examples (Fillmore et al., 2003). FrameNet is based on the theory of meaning - Frame Semantics, which is a conceptual structure describing the meaning of a word in an NL statement in terms of frame elements like entities participating in an event, type of event, location of event etc. Frame Elements (FE) are the keys representing semantic roles for words in an NL statement. A frame is composed of core FEs and non-core FEs. Another constituting element of FrameNet is a Lexical Unit (LU) that is responsible for evoking a frame. One frame can be evoked by multiple LUs. Let us consider a sample statement to understand how knowledge in that statement is organized as FEs and LUs in FrameNet:

S2: The chairman of the company only has the authority to approve a claim.

Authority frame from FrameNet, evoked by LU, 'authority' best describes the statement S2 in terms of core FEs of *authority* frame – agent and theme; and non-core FEs – descriptor, domain, and source as indicated below:

Agent	-	The chairman of the company
Domain	-	claim

FrameNet lexical database contains over 1200 semantic frames, 13,000 lexical units and 202,000 example statements. FrameNet organizes its frames in terms of relationships among frames. These relationships indicate inheritance relation, preceding frame, perspectivize_in frame, causative, uses, and inchoative associations between frames in FrameNet.

4 PROPOSED APPROACH

In this section, we present our proposed approach towards enriching NL requirements statements using additional knowledge components (frame elements) from FrameNet lexical database. The requirements statement can possibly be grammatically, syntactically, and semantically correct. But, it is still possible that this statement does not reflect complete view of the real-world knowledge around it. Our contribution lies in bringing forward additional concepts (not present in the requirements statement) related to the concepts present in the statement under study. Such identified additional concepts can assist analysts in finding potentially missing information in the requirement.

4.1 Enriching NL Requirements using Framenet

In this sub-section, we present our approach to evoke frame(s) from FrameNet database while referring to FBSR of an NL requirements statement under study. As discussed in section 3.1, FBSR is a union of two or more frames. Each of these frames is referred to while evoking frames from FrameNet. The FEs from the evoked frame, in turn, assist in enriching NL requirements

Figure 1 presents our algorithm for finding new related and relevant concepts for an NL requirements statement. We refer to FBSR generation algorithm (Sharma, 2016) to identify concepts or lexical units present in the input requirements statement. The algorithm presented in Figure 1 requires manual intervention in the step 3(d) when questions need to be articulated for the newly identified concepts. Requirements analysts can present the formulated questions to domain experts for enriching the requirements.

Let us consider RS1 as example to understand how algorithm presented in Figure 1 helps in refining RS1:

RS1: *If a person is not a member of library then the person cannot borrow the book.*

LUs present in corresponding FBSR of RS1 after dropping duplicates include - *person, borrow, book, member, library.* Referring to FrameNet library, we found that there are no corresponding frames for LUs – person, book, and library. However, for the concepts – borrow and member, following respective frames are evoked:

```
Input:    NL requirements statement, RS_in
Output:   New related concepts identified

   1. FBSR (RS_in)

   2. List of LU = value of these frame keys
      from each frame - 'actor', 'object',
      modifiers of 'actor', 'object', and
      'action'. Ignore 'neg' key and the
      frame-identifying key like marker,
      preposition etc.

   3. For each LU_i in the list of LU:

      (a) Search for LU_i in FrameNet
          database against lemmatised LU in
          FrameNet to evoke its
          corresponding frame.
      (b) If no frame exists in FrameNet for
          LU_i , report 'No Frame found',
      (c) If duplicate (LU_i), continue in the
          loop.
      (d) If a frame is found for LU_i, then
          extract core FEs and non-core FEs
          for that frame. Formulate
          questions around the extracted FEs
          to help analyst uncover any
          potentially missing information.
```

Figure 1: Algorithm for finding new concepts from FrameNet database.

1. Borrowing Frame – This frame has three core FEs, namely – borrower, lender and theme. There are six non-core FEs for this frame – duration, manner, means of transfer, place, purpose, and time. This frame inherits from 'Receiving' frame and has perspective on relationship with 'Temporary Transfer scenario' frame. The non-core FEs in this frame, thus, include relevant aspects around lending, transferring and receiving acts.

   Analysts need to check which FE is present in RS1 and which FE is missing with reference to FEs from membership frame. RS1 has all the three core FEs present – the borrower (person/member), lender (library), and theme (book). However, non-core FEs are potential candidates of missing information from RS1. Analyst can enrich RS1 by seeking information from domain expert around non-core FEs as:

(a) What is the duration for which member can borrow the book?
(b) What is the means of transfer, i.e. how to identify the notion of 'borrowing' with reference to theme – 'book'?
(c) What is the purpose of borrowing? Does it need to be stored?
(d) What is the time of borrowing? Does it need to be stored with the details of borrowing?

Responses to these questions will help analyst in refining the informal requirements gathered during early phases of RE, and enriching these requirements with newly acquired knowledge from domain experts. In the absence of domain knowledge or any other body of knowledge, it is difficult and challenging to find any missing information when there is no clue as to what should be asked to gather more information.

2. Membership Frame – This frame has two core FEs – group and member (person belonging to the group) and four non-core FEs – manner, place, standing and time. This frame inherits from 'Be subset of' frame, and is used by two frames – 'Exclude member' and 'Member of military'. Following similar approach as for *borrowing* frame, it is found that RS1 has core FEs information incorporated within its corresponding NL statement. Further, it can be enriched by getting information around these non-core FEs - standing and time.

The above example considered for RS1 indicates that even though FrameNet does not correspond to domain knowledge of library management but it is capable of providing useful pointers for adding more information to RS1. Encouraged by this observation, we have carried out our study on event-processing case study (Sharma, 2016) to check the applicability and viability of our solution approach towards handling incompleteness in requirements. We present observations from the case studies to evaluate our approach in the following section.

4.2 Case Study

The case study (Sharma, 2016) that we have used to carry out evaluation and viability study of our proposed approach for enriching the informal requirements is as stated below:

<u>Event-processing Scenario:</u> *An event, announced on a security, has an event type. The customer who holds the security can get benefits of the event. System should permit creating events online. System should be able to process file XXX received from ZZZ server to create events in batch. The event details should be displayed in a list. The number of events displayed on one page should be configurable. The customer can get benefits on the event as cash, stock or both. The GUI should allow customer to opt for one or more of these benefits. If the customer opts benefits as cash, then cash is distributed to the customer for the event announced on security held by the customer. If the client opts stock as benefit for the event announced on security held by the customer, then stock is distributed. If the client opts benefit as both for the event announced on security held by the customer, then both are distributed. However, base country of security and the country of the customer may influence the benefits distributed to the customer. The customer can view his entitlement after selecting an event and clicking on the entitlement button.*

For this scenario, 32 unique LUs are identified to which algorithm presented in Figure 1 is applied to evoke relevant frames in FrameNet. For these 32 LUs, only 8 corresponding frames were found and evoked, i,e. 25% matching LUs between the scenario and the FrameNet database. These eight LUs are: *event, create, get, hold, type, system, process,* and *receive.*

The fact that FrameNet database is meant for generally used concepts in news, discourses, and it does not extend to any business or technical domain can be attributed to fewer overlaps between LUs collected from scenario and the FrameNet's LUs. Nevertheless, the FEs present in the frames served as guiding pointers to further enrich event-processing scenario. For instance, '*getting*' frame corresponding to 'get' LU added value by assisting in collecting details for these FEs – means, result, and time.

Our study on sample statements from library management scenario and the case study on event-processing scenario indicate that though FrameNet might not be of help to find most of the missing issues but in the absence of domain knowledge or reference documentation, it can serve as a guiding tool to find a considerable number of missing elements in the gathered informal requirements. Our study currently processes FEs from the evoked frame only, and does not make reference to other related frames (inheritance, uses, used_by, perspective_on etc.). Secondly, we are not considering synonyms in our current implementation. We intend to work on these two lines in future and improve our solution approach.

5 CONCLUSIONS

In this paper, we have presented an approach to enrich and refine informal requirements gathered during early RE with the objective of addressing incompleteness concern in these requirements. The presented study is only a preliminary investigation of the proposed approach. There are challenges with the proposed approach as frames in FrameNet lexical database correspond to generic concepts whereas software requirements pertain to a specific business domain covering technical aspects. The preliminary study, however, reveals sufficiently encouraging observations to further refine the proposed approach to handle incompleteness problem in the informal requirements. In future, we plan to extend our algorithm to other related frames while invoking a frame for an LU. Secondly, we need to work with synonyms, and conduct more rigorous case-studies for validating our proposed approach. We believe that as FrameNet database is increasing, our approach will yield in better results though the same needs to be supported by a number of case-studies.

REFERENCES

Saavedra, R, Ballejos, L & Ale, M 2015, Quality Properties Evaluation for Software Requirements Specifications: An Exploratory Analysis. *Proceedings of WER'13, 16th edition of Workshop on Requirements Engineering*, Uruguay.

Firesmith, DG 2003, 'Specifying Good Requirements', *Journal of Object Technology*, vol 2, no. 4, July-August 2003, pp. 77-87.

Zowghi, D & Gervasi, V 2002. The Three Cs of Requirements: Consistency, Completeness, and Correctness. *Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality*, Germany.

Fabbrini, F, Fusani, M, Gnesi, S & Lami, G 2001. An Automatic Quality Evaluation for Natural Language Requirements. *Proceedings of 7th International Workshop on Requirements Engineering: Foundation for Software Quality*, Switzerland.

Kuchta, J 2016. Completeness and Consistency of the System Requirement Specification. *Proceedings of Federated Conference on Computer Science and Information Systems*, pp. 265-269, Poland.

Génova, G, Fuentes, JM, Llorens, J, Hurtado, O & Moreno, V 2013, 'A Framework to Measure and Improve the Quality of Textual Requirements', *Requirements Engineering*, vol. 18, no. 1, pp. 25-41.

Bhatia, J, Sharma, R, Biswas, KK & Ghaisas, S 2013, Using Grammatical Knowledge Patterns for structuring requirements specifications. *Proceedings of 3rd IEEE International Workshop on Requirements Patterns (RePa'2013)*, in conjunction with 21st IEEE International Requirements Engineering Conference (RE'13), pp. 31-34, July 2013.

Sharma, R 2016, 'A semi-automated approach to support logical formalism for Requirements Analysis and Evolution' PhD Thesis, School of Information Technology, IIT Delhi, India.

Atkins, BTS, Klavens, J & Levin, B 1988, 'Anatomy of a verb entry: from linguistic theory to lexicographic practice', *International Journal of Lexicography*, vol. 1, no. 2, pp.: 84–126.

Fillmore, CJ, Johnson CR & Petruck, MRL 2003, 'Background to FrameNet', *International Journal of Lexicography*, vol. 16, no. 3, pp. 235–250.

ISO/IEC/IEEE International Standard 2011, Systems and software engineering -- Life cycle processes -- Requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, doi: 10.1109/IEEESTD.2011.6146379.

Pohl, K 2010, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer-Verlag Berlin Heidelberg.

Firesmith, D 2005. 'Are Your Requirements Complete?', *Journal of Object Technology*, vol. 4, no. 1, pp. 27-43.

Durán, A, Bernárdez, B, Ruiz, A & Toro, M 2001. An XML–based Approach for the Automatic Verification of Software Requirements Specifications. *Proceedings of 4th Workshop on Requirements Engineering*, pp. 181-194.

Carson, RS & Shell, T 2001. Requirements completeness: Absolute or relative? comments on 'system function implementation and behavioural modelling[syst eng 4 (2001), 58-75]', *Systems Engineering*, vol. 4, no. 3, pp. 230–231.

Boehm, BW 1984. 'Verifying and validating software requirements and design specifications', *IEEE Software*, vol. 1, no. 1, pp. 75-88.

Davis, AM 1993. *Software Requirements: Analysis and Specification*. Prentice Hall, second edition.

Sutcliffe, A & Maiden, N 2002. 'The domain theory for requirements engineering', *IEEE Transactions on Software Engineering*, vol. 24, no. 3, pp. 174-196.

Minsky, M 1981, *A Framework for Representing Knowledge*, J. Haugeland, Ed., Mind Design, MIT Press.

Marshman, E, Morgan, T & Meyer, I 2002, 'French patterns for expressing concept relations', *Terminology*, vol. 8, no. 1.