Constructing Complete Requirement Specification from Natural Language Requirements Document

Navin Kumar Sehgal
Department of Computer Science,
Kalindi College, Delhi University
New Delhi, India

Krishan Kumar Rohilla
Department of Computer Science
Kalindi College, Delhi University
New Delhi, India

***Abstract*— SRS is an important document for SDLC (Software Development Life Cycle). It helps project manager in obtaining requirements for further phases. Extracting requirements from requirements document written NL can cause errors. In this paper we use derivation, verification and completion techniques to obtain a well defined and accurate requirements table.**

Keywords - Requirement specification , correctness, completeness

I. INTRODUCTION

Requirements document can be defined as a document specifying various types of requirements including product description, design and implementation constraints, user documentation, user interface, system features and other non functional requirements.

The requirements engineering team is responsible for elicitation, validation and negotiation of requirements. The requirements are presented in an SRS. There is high probability of requirements being error prone or incomplete. It can lead to defects being carried to further phases of SDLC (Software Development Life Cycle). The result is a defective software product.

SRS therefore cannot be considered as the final document. It contains a number of defects which cannot be corrected in the normal requirements engineering process. An additional technique is required to weed out the errors. Errors can be attributed to different types of specifications. An Incorrect specification refers to a specification which is not in accordance with either user or product specification. An ambiguous specification can lead to a number of different implementations. A superfluous specification is abstract in nature. An incomplete specification can leave out important details. A non implementable specification can not be covered using existing project resources. The SRS is a large document and therefore it becomes difficult to correct it in its existing format.

In this paper we introduce a CRS (Complete Requirement Specification). It is created by removing the errors of SRS.A CRS can be derived from SRS using techniques mentioned in this paper.

A CRS is in the form of a table. Therefore it shows the relevant requirements in an unambiguous manner. CRS is constructed using well defines techniques like filtering, error checking and completion checks. CRS thus obtained is complete and fully correct. A CRS contains requirements that are implementation friendly.

CRS relies on CROSSWORD method for requirements definition. In the game CROSSWORD the words have to be filled in the spaces to complete it. Similarly the SRS serves as the clue and keywords are obtained from it under different categories. The keywords are filed in the CRS table.

II. CRS TECHNIQUES

A. Filtering and Prioritization

A number of techniques are followed for producing a CRS. They include derivation of requirements, filtering the requirements, prioritizing them, checking them against parameters, completing the requirements.

In the first step requirements are derives from the requirements document written in NL. This involves extracting keywords. The keywords contain all the important aspects of the software project. The keywords are important therefore they should encapsulate the key project elements. These are put in a formative CRS table. The exercise is similar to a CROSSWORD puzzle. The keywords are put in different categories – functional requirements, no functional requirements, external functional requirements, use cases, database requirements and constraints.

Requirements	Keywords	Check	Priority
Functional	input-patient-data	C	4
	web-based-service	A	3
	user-validation	S	4
	historical-patient-information	I	3
	unique-patient-identification	C	3
	physician-wise-record	A	4
	date-time-display	C	3
	medication-info-search	D	3
	diagnostic-imagery-record	T	4
	diagnostic-imagery-report	A	4
	patient-diagnosis-validation	A	3

Table 1. Formative CRS table for functional requirements of hospital information system

The next step is prioritizing the requirements. Each requirement keyword is assigned a priority in the scale 1 – 4. The priority is decided if it stakeholder friendly and does not violate constraints.

B. Checking for correctness

The next technique involves error checking. The superfluous, ambiguous, inconsistent, incorrect and non implementable requirements are removed from the CRS table. A separate column is appended to CRS table and requirements are classified as correct (C), superfluous(S), ambiguous (A), not-agreed (D) inconsistent (T) incorrect (N) and non-implementable (I). A requirement is considered superfluous(S) if it is abstract in nature. It is ambiguous (A) if it relates to two or more implementations. It is not agreed if stakeholders have differing opinion on them. It is inconsistent (T) if it overrides another requirement. A requirement is Incorrect (I) if it is either not related with a project or wrongly defines a project element. A non-implementable (I) is one which is beyond the technical and financial considerations. Other than correct requirements all other undergo modification.

C. Checking for completeness

Next comes the completion techniques. Again a column is added to the table and each requirement is marked as complete (C) or omission (O). An omission requirement is one which needs further elaboration in order to be made implementable. Each such requirement needs adding additional requirement keywords. This technique is iterated till all the requirements are complete.

D. Iterative procedure

III CASE STUDY

A case study based on Hospital Management System is conducted for the purpose of examining the techniques mentioned in the paper.

The SRS of a hospital management system consists of description of the project, requirements of physician office system, requirements of the hospital system and requirements for patient monitoring system.

These are classified into functional requirements, non-functional requirements, database system, external interface requirements, use cases and constraints.

A formative CRS table is created as shown in table 1.

A. CRS Formulation

The procedure is of CRS generation is iterative. After the formative CRS table has been created, next step is to assign priorities and check for errors.

The procedure is iterative in nature. Each iteration consists of two phases. In the corrective phase each requirement is checked manually for errors. Each error is marked with a symbol S (superfluous), A (ambiguous), D (not agreeable), T (inconsistent), N (incorrect), I (non implementable). The errors are correct by modifying or adding new requirement keywords. Each new requirement keyword is assigned a new priority value and modified requirements cause change in the priority value.

The next phase is completeness phase. The requirement is marked either C (correct) or O (omission). All omitted requirements are appended to existing requirements and assigned priority numbers.

Iteration	Phase	Priority	Functional	Non Functional	Database	Use Cases	Constraints
	Initial	79	9	3	7	2	3
1	Correctness	94	11	4	7	2	3
	Completeness	114	13	4	7	3	3
	Total Change	35	4	1	0	1	0
	% change	44.30380	44.44444	33.33333	0.00000	50.00000	0.00000
2	Correctness	137	17	5	8	3	5
	Completeness	151	20	6	8	3	5
	Total Change	37	7	2	1	0	2
	% change	32.45614	53.84615	50.00000	14.28571	0.00000	66.66667
3	Correctness	156	20	6	9	4	5
	Completeness	160	21	6	9	4	5
	Total Change	9	1	0	1	1	0
	% change	5.96026	5.00000	0.00000	12.50000	33.33333	0.00000
4	Correctness	170	23	7	9	4	5
	Completeness	174	23	7	9	5	5
	Total Change	14	2	1	0	1	0
	% change	8.75000	9.52381	16.66667	0.00000	25.00000	0.00000

Table 2. Changes in priority and number of requirements for iterations 1 to 4

B. *Equations*

If set of requirements at correctness phase and completion phase are denoted by Tr and Cr respectively. The priority values for Tr and Cr are Pt and Pc respectively. If Pch is the change in priority .

$$|Tr_{i+1}| > |Tr_i| \quad , \quad |Cr_{i+1}| > |Cr_i| \tag{1}$$

$$Pt_{i+1} > Pt_i \quad , \quad Pc_{i+1} > Pc_i \tag{2}$$

.

$$Pch_i = a * Pt_i + b * Pc_i \tag{3}$$

Equation (1) relates to the changes in the requirements over various iterations. The change in the priority of the requirement is shown in equation (2). The third equation specifies the relationship between priority change and number of priorities at correction and completion step. Values of constants a and b lies between 0 and 1.

C. *Useful Obervations*

- As shown in table 2 it can be deduced that the no of requirements shows an increase in the first few iterations or is maximum for first few iterations.
- The number of requirements reaches a saturation level after there is decline in the change in requirements
- The number of functional requirements increases at a faster rate than number of non functional requirements.
- The correctness phase changes the priority of the requirements as compared to completeness phase that changes the number of requirement.
- Overall priority changes more during completeness phase than correctness phase.
- The number of changes in all other requirements is less than changes in functional requirements
- As the number of iterations increases the number of changes reduces gradually till it becomes zero.

IV USES OF THE STUDY

The study is relevant for SDLC as it reduces the product defects in the first phase itself. The CRS study can guide the remaining phases of the SDLC. Table formulation and change procedures can be followed in further stages also to reduce errors.

FUTURE WORK

The number of correctness and completeness checks can be increased based on the project domain in addition to current basic checks in order to improve the quality of the software product.

The current method involves manual checking that involves one or more members of the requirements team and stakeholders. The future tables can be generated by programs and the process can be made automatic.

REFERENCES

[1] Amira A. Alshazly, Ahmed M Elfatatry and Mohammed S Abougabal, “Detecting defetcts in software requirement specifications”, Alexendria Engineering Journal, vol. 2014,in press.

[2] Tejalal Choudhary and Anurag Goswami, “Investigating the effect of fault category on overall capture probability during requirements inspection”, International journa; of computer science and information technology, vol(5) 4,2014.

[3] Ninaus, Gerald, et al. "INTELLIREQ: Intelligent Techniques for Software Requirements Engineering." 21st European Conference on Artificial Intelligence/Prestigious Applications of Intelligent Systems (PAIS 2014), p. to appear, Prague, Czech Republic. 2014.

[4] Amit Kumar Jakhar and Kumar Rajnish, “A new cognitive approach to measure the complexity of software”, International journal of software engineering and applicatios, vol 8, No. 7(2014)

[5] Dorfman, Merlin. "System and software requirements engineering." IEEE Computer Society Press Tutorial. 1990..

[6] Hallerstede, Stefan, Michael Jastram, and Lukas Ladenberger. "A method and tool for tracing requirements into specifications." Science of Computer Programming 82 (2014): 2-21

[7] Ghanavati, Sepideh, Daniel Amyot, and André Rifaut. "Legal goal-oriented requirement language (legal GRL) for modeling regulations." Proceedings of the 6th International Workshop on Modeling in Software Engineering. ACM, 2014.

[8] Achimugu, Philip, et al. "A systematic literature review of software requirements prioritization research." Information and Software Technology 56.6 (2014): 568-585.