Estimating Software Size using Metaheuristic Approach: An Experimental Review

Manisha

Research Scholar, Maharshi Dayanand University, Rohtak, Haryana

Dr. Rahul Rishi

Director, U.I.E.T Maharshi Dayanand University, Rohtak, Haryana

Abstract

Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. Requirements Engineering Process consists of the following main activities:

Requirements of a). Elicitation b). Specification c). Verification and validation and d). Management

By combining all the requirements, we can design the project. These are the minimum steps which a developer or a user follow for a successful project.

Introduction

The project can be defined as a sequence of steps which are needed for the development. For the complete project development, the user has to follow the sequence of steps such as requirement Collecting, Analysis, Design, Coding, Testing and Maintaining. Each step mentioned detail as given below:

Requirements Elicitation

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards and other stakeholders of the project.

The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping. Some of these are discussed. Elicitation does not create formal models of the prerequisites comprehended. Instead, it augments the space information on the examiner and in this manner encourages in giving a contribution to the following stage.

Requirement's Specification

This activity is used to produce formal software requirement models. These models in totality specify all the requirements including the functional as well as the non-functional requirements and the constraints. During specification, more knowledge about the problem may be required, which can again trigger the elicitation process. [1]

The models used at this stage include ER diagrams, Data Flow Diagrams (DFDs), Function Decomposition Diagrams (FDDs), data dictionaries, etc.

Requirement's verification and validation

Verification

It alludes to the arrangement of assignments that guarantees that the product accurately executes a particular capacity.

Validation

It alludes to an alternate arrangement of errands that guarantees that the product that has constructed is detectable to client necessities.

If prerequisites are not approved, mistakes in the basic definitions will proliferate to the progressive stages bringing about many alterations and revise.

The principal ventures for this procedure include

→ The necessities ought to be steady with the various prerequisites i.e no two prerequisites should include strife with one another. [2]

→ The prerequisites ought to be finished in each sense.

→ The necessities ought to be basically attainable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements Management

Requirement management is the way toward examining, archiving, following, organizing and concurring on the necessity and controlling the correspondence to essential partners. This stage deals with the changing idea of necessities. It ought to be guaranteed that the SRS is as modifiable as could be expected under the circumstances in order to fuse changes in necessities determined by the end-clients at later stages as well. Having the option to adjust the product [3] according to prerequisites in a deliberate and controlled way is a critical piece of the necessities designing procedure.

Characteristics of SRS Document

1. **Correctness**

User review is used to guarantee the correctness [4] of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are expected from the system.

2. **Completeness**

Completeness of SRS indicates every sense of finishing including the numbering of the considerable number of pages, settling the to be resolved parts to however much degree as could reasonably be expected just as covering all the utilitarian and non-practical prerequisites appropriately.[5]

3. **Consistency**

Requirements in SRS are said to be steady if there are no contentions between any arrangement of necessities. Instances of contention remember contrasts for phrasings utilized at isolated places,[6] legitimate clashes like timeframe of reportage.

4. **Unambiguousness**

An SRS is said to be unambiguous if all the prerequisites expressed have just 1 understanding. A portion of the approaches to forestall unambiguousness [7] incorporates the utilization of displaying methods like ER outlines, appropriate audits and pal checks, and so on.

5. **Ranking for importance and stability**

There should form a basis to arrange the necessities as less or increasingly significant or all the more explicitly as alluring or essential [8]. An identifier imprint can be utilized with each necessity to show its position or strength.

6. **Modifiability**

SRS should be made as modifiable as could reasonably be expected and ought to be prepared to do effectively tolerating changes to the framework somewhat. Changes ought to be appropriately recorded and cross-referenced.

7. Verifiability

An SRS is verifiable if there exists a particular strategy to quantifiably quantify the degree to which the framework meets each requirement.[9] For instance, a necessity expressing that the framework must be easy to understand is not inevitable and posting such prerequisites ought to have stayed away from.

8. Traceability

One should be able to trace a requirement to a design component and then to a code segment in the program [10]. So also, one ought to have the option to follow a necessity to the comparing experiments.

9. Design Independence

There should be an option to choose from multiple design alternatives for the final system. More specifically, the SRS should not include any implementation details.

10. Testability

An SRS should be written in such a way that it is easy to generate test cases and test plans from the document.

11. Understandable by the customer

An end-user may be an expert in his/her specific domain but might not be an expert in computer science. Hence, the use of formal notations and symbols should be avoided too as much extent as possible. The language should be kept secure and transparent. [11]

12. The right level of abstraction

If the SRS is written for the requirements phase, the details should be explained explicitly. Whereas, for a feasibility study, fewer details can be used. Hence, the level of abstraction varies according to the purpose of the SRS.

Conclusion

For the practical consummation of the undertaking, the base things are required the product life cycle. The product life cycle relies upon the number of steps the product is required for the advancement of the venture and for the effective finishing of the undertaking the Software Requirement Specifications additionally need to follow for indicating the task. This paper in the global spotlight on the various sorts of software and the types of software.

References

[1] Liang Ping, A Quantitative Approach to Software Maintainability Prediction, International Forum on Information Technology and Applications (IFITA), 2010

[2] Sunday, D.A., Software maintainability-a new ility', Annual Reliability and Maintainability Symposium, 1989. Proceedings., Publication Year: 1989, Page(s): 50-51

[3] Perepletchikov, M. ; Ryan, C. ; Frampton, K., Cohesion Metrics for Predicting Maintainability of Service-Oriented Software, Seventh International Conference on Quality Software, 2007. QSIC '07. Publication Year: 2007, Page(s): 328-335

[4] Bowles, J.B., Code from requirements: new productivity tools improve the reliability and maintainability of software systems, Annual Symposium - RAMS Reliability and Maintainability, 2004, Publication Year: 2004, Page(s): 68-72

[5] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model, Version 3.1 Rev 1," Nat'l Inst. Of Standards and Technology CCMB-2006-09-001, Sept. 2006.

[6] Davis A.M, Bersoff E., Hm Comer E.R, A strategy for comparing alternative software development life cycle models, IEEE Transactions on Software Engineering, Volume: 14, Issue: 10 Publication Year: 1988, Page(s): 1453-1461

[7] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model, Version 3.1 Rev 1," Nat'l Inst. Of Standards and Technology CCMB-2006-09-001, Sept. 2006.

[8] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components,

Version 3.1 Rev 1," Nat'l Inst. Of Standards and Technology CCMB-2006-09-002, Sept. 2006.

[9] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1 Rev 1," Nat'l Inst. Standards and Technology CCMB-2006-09-003, Sept. 2006.

[10] IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, IEEE Std 830-1998.

[11] S Velmourougan, P Davachelvan, and R Baskaran, Software Reliability Qualification Model, International Journal of Performability. 8(2012) 437-446.