J. of Comp. and I.T. Vol.1(2), 123-126 (2010). ISSN 2229-3531 (Print)
www.compitjournal.org 2455-9997 (Online)

Improving Requirement Specification Task

VIVEK MISHRA

Lecturer, Department of M.C.A. I.E.T. Dr. R. M. L. Avadh University, Faizabad (INDIA)

(Acceptance Date 27th October, 2010)

Abstract

Improvement in traditional paper based R S Task is basically improvement in requirement specification tools by which the R S Tasks does its work. The improvement in R S Tools will not only enable better Requirement Management; but it also enable the automatic generation of consistent, current and audience specific Request Specification that for meet the demand of their individual looker-on.

Key words : BLOB, CaliberRM, DOORS, CASE, IDE.

Introduction

In traditional R S there are several problems. That paper based R S was not sufficient and effective to fulfill the needs of their looker-on. Because different audience have their different requirements. For example executive need executive level document, Managers use those documents to manage project scope and estimate endeavor schedules and requirement reason, Subject Matter Experts need requirement specification that concentrate on there area of subject matter expertise. Increasing size and complexity of application also demand better tool support for R S Tasks. Because over the last quarter century typical applications have consistently grown larger and more complex, monolithic client server application have been replaced, by client server application which in turn have been replaced by n-tire applications, simple websites being replaced with e-com and e-marketing websites so there is need to improve the Requirement Specification Task.

Based on the previously mentioned challenges to and trends affecting requirements engineering in general (and requirements specification in particular), what should we and industry professionals[1] do ? I would make the following recommendations designed to improve the requirements specifications produced by the requirements specification task.

Requirements Repository:

Store your requirements in a requirements repository instead of a paper document. Keep the granularity of the repository small so that individual requirements can be entered, iterated, approved, placed under configuration management, published, managed, and traced. Thus,

requirements should be considered to be individual objects and the last thing you want to do is to store a complete requirements specification as a binary large object (BLOB). Ensure that the requirements repository stores all kinds of requirements related information including individual requirements, requirements metadata (the attributes of requirements objects), and requirements models (aggregate objects) including diagrams and tables.

Automatic Specification Generation:

Do whatever it takes (within reason) to enable the automatic generation of requirements specifications from the requirements repository. This goes beyond the simple use of international (e.g., IEEE830-1998), industry (e.g. OPEN or RUP), or business-internal templates for one or more requirements specifications. It also includes the creation of arbitrary requirements reports for requirements management and other purposes. This should also include the generation of the entire publishable requirements specification and not just the generation of a part of the requirements specification that then requires significant amounts of manual labor to complete. This enables the requirements specifications to be current with the official requirements in the repository. It also enables the generation of electronic specifications and reports that save a huge amount of paper in an iterative development cycle in which the requirements change on essentially a daily basis. The first recommendation for a requirements repository as well as the current recommendation both recognize the separation of Model and View that has been so beneficial in the production of graphical user interfaces.

Different Specifications for Different Audiences:

Once you have your requirements stored in a well-organized requirements repository and the ability to automatically generate requirements specifications from that repository, then you have the ability to produce multiple audience-specific versions of the requirements specifications. This is nothing more than the recognition that there is often a need for multiple views (specifications, reports) of different parts of the same model (requirements). This includes specifications and reports that differ in the types of requirements viewed (e.g., functional requirements vs. interface requirements), the different levels of requirements details (executive overview specifications vs. detailed requirements specifications for testers), predefined specifications vs. ad hoc reports, and specifications based on metadata (specifications or reports based on requirements due date, status, owner, last modified date, etc.).

Requirements Tools:
User Interface:

The best way to enter requirements and their metadata is by using a user-friendly graphical user interface that allows one to easily enter and maintain individual requirements, their metadata, and requirements models including text, diagrams, and tables, etc. The user interface should understand the underlying requirements model including requirements types, their common and type specific metadata, the different types of models, the different types of diagrams, etc. The user interface should not only support requirements input and

maintenance but also requirements specification and report generation including template creation, querying, and actual production.

Requirements Engineering Support:

It should support multiple requirements analysis approaches so that multiple types of models (e.g., use cases, decision tables, state models, context models) can be specified. It should also support the entire requirements model including all types of requirements including functional requirements, data requirements, quality requirements (a large list), interface requirements, and constraints (also a large list). Requirements management tools should also include requirements traceability to architecture, design, implementation, and testing work products and back.

Support for Related Activities: This should include scope control, configuration management, and quality engineering. This includes interoperability with management, configuration management, quality engineering, modeling, and testing tools for forward and reverse engineering. Thus, a requirements tool should not be stand-alone, but a critical component of an integrated development environment.

Team Development: Requirements engineering is best performed by a cross functional requirements team that provides an adequate experience base to capture all of the requirements and to iterate them in a timely fashion. Although it is useful to have a technical writer versed in requirements modeling and the use of modeling tools to be the primary person to initially capture the requirement during joint requirements engineering sessions, all requirements team members should be able to work on the requirements simultaneously (another good reason for fine granularity in the requirements repository). Similarly, other members of the endeavor team need to have simultaneous access to the requirements for purposes of learning, evaluation, and approval.

Security: Requirements for many applications involve proprietary information, trade secrets, or even national secrets. Any requirements management tool should support the security of the requirements including the identification, authentication, and authorization to perform role-specific tasks of its users. It should also include privacy, integrity, and non-repudiation of requirements and and non-repudiation of requirements and their updates.

Other Quality Factors: A requirements tool is an application in many ways like any other. Thus, requirements tools to be used for the specification of requirements should also have the appropriate amounts of other quality factors[3] besides security and interoperability. Thus, when evaluating such tools, consider the typical quality factors such as completeness, internationalization, performance, scalability, usability, and user friendliness.

Distributed Development: A requirements tool should support requirements engineering including specification by distributed users.

Requirements Reuse: The requirements tools should enable the easy incorporation of existing requirements into its repository so that each endeavor need not start from scratch. Because these requirements were probably generated using traditional approaches, the requirements tool needs to be

able to parse the old requirements specifications, recognize potential requirements and incorporate them in a manner that will be easy for them to be reviewed and either accepted as is, accepted with modification, or rejected.

Not Just a CASE Tool: An adequate requirements tool is more than just a simple stand-alone modeling tool or requirements repository It should be part of an Integrated Development Environment (IDE), but it is not merely a simple Computer Aided Software Engineering (CASE) tool in the traditional sense.

Conclusion

This article has explained traditional R S Specification approach and recommended a requirements specification approach to solve these problems . Using this new requirement specification tools, you can automatically, easily, and inexpensively generate various types of high-quality requirements specifications that are tailored to meet the individual needs of their various audiences.

References

1. Professional liability and the use of Computers" in Proc. ASLE 1st Int. Convention of Computer in Civil Eng (1981).
2. The Open Process framework. Addison Weslem -Lofman (2001).
3. "Software Quality assurance", IEEE transaction of software eng Vol. Se. 10 No.1 Jan (1984).