Bob Builds a Lightweight Requirements Manager

David Gelperin
ClearSpecs Enterprises
2425 Zealand Ave. N.
Golden Valley, MN 55427 USA
Tel: +1 480-296-3559
Email: david@clearspecs.com

Abstract

Bob explores the feasibility of building a lightweight requirements management tool. He develops requirements to assist in evaluating potential platforms. Then, he selects a platform and builds a requirements manager for his current project.

1. Introduction

Bob's head falls to his desk as he dozes off after lunch. It strikes a deck of 3 x 5 index cards containing the requirements for his latest project, and the cards scatter to the floor. Bob awakes with a start, sees the cards, and thinks, "There must be a better way."

Bob knows about commercial and open source requirements managers (RMs) [1]. But these medium to heavyweight tools -- with generators for specification documents, simulations, and tests, as well as strong change control capabilities – provide much more power (and overhead) than he needs.

What Bob wants is a lightweight RM, but he doesn't know of any. He searches the web to see if one exists and discovers the phrase "lightweight RM" has at least four meanings. It refers to the use of:

1. word processors and spreadsheets
2. wiki templates and documents [2]
3. a virtual, wiki-based story wall [3]

Bob was not thinking of any of these. He imagines a versatile, easy to use, inexpensive tool that provides the core RM functions of medium to heavyweight tools, i.e., most of the functionality that does not involve controlling or generating anything.

He wonders if he can build one using an existing tool as a platform. Bob considers tools used to keep track of ideas and wonders if any of these might work. He realizes there are three types of tools he should explore: Outliners [4], Personal Information Managers [5], and Mind Mappers [6].

2. Preparation

To assess platform candidates, Bob develops the following list of assumptions about requirements information:

1. The goal of requirements activity is to help developers acquire a sufficiently deep understanding of customer and user needs so they can successfully develop.

2. Recorded requirements information is useful to the extent that it promotes this goal at a reasonable cost.

3. There are many different types of information related to requirements e.g., user roles, object models, use cases, acceptance tests, as well as the requirements themselves.

4. The need for recording specific requirements information (i.e. the content of a useful ontology) is context-dependent, depending on (a) type of solution system, (b) initial understanding of developers, and (c) ready access to subject matter experts.

These assumptions suggest that 'one size does not fit all', i.e., that project teams must be able to choose the type of information they record and change choices during the project.

Bob prefers to organize information in outlines. He has used the outlining feature in MS Word but found these outlines difficult to change. Since requirements can be volatile, ease of creation and change is important.

Bob develops an analogy to help understand his needs. He thinks about a grove of assorted fruit trees where each tree contains a distinct type of requirements information e.g., definitions and object models. Each piece of fruit represents an instance of a type e.g., a specific acceptance test or user role. A piece of fruit can be in several states e.g., unimplemented and superficially understood. A piece of fruit in one tree can be linked to pieces of fruit in other trees e.g., a use case could be linked to its derived acceptance tests.

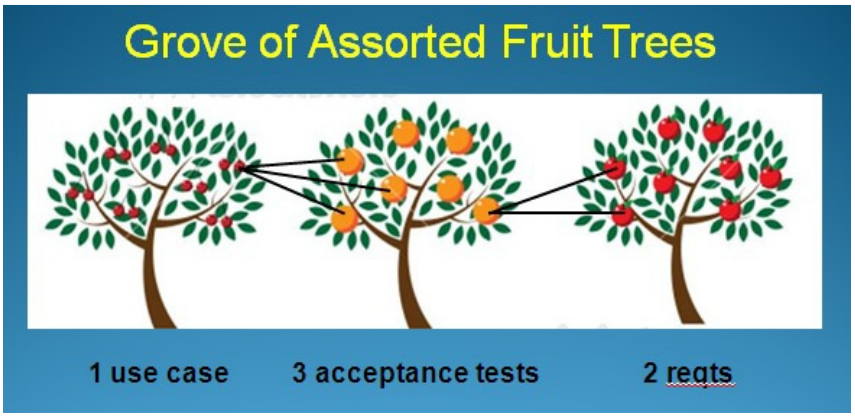

Figure 1 – A Grove

The RM (i.e., grove manager) needs to be able to answer many important questions. Examples include:

1. What are the details of each requirement?
2. Which requirements are essential?
3. Which requirements were provided by a specific stakeholder?
4. Which requirements are associated with a specific user role?
5. Which requirements are derived from this one?
6. Which requirements are dangerous?
7. Which requirements are poorly understood by the developers?
8. Which requirements have been implemented?
9. Which requirements and acceptance tests are associated with this use case?
10. How many high-priority requirements' issues are open?

Bob realizes his grove won't be worth much unless it is shared so that all stakeholders can track and review the quality of the requirements information. In addition, pictorial forms of requirements information such as diagrams and mockups do not need to be in the grove if they can be linked to it.

Using his grove analogy, he develops the following process for building a useful requirements information structure (i.e., customizing a grove):

1. Choose your trees: Build-your-own (organizational) framework or tailor a pre-built framework.
2. Add the fruit: Populate your framework with project-specific instances.
3. Annotate the fruit: Add details to each instance including attributes, states, and associations.
4. Cost justify elements of your grove: Perform an informal cost/benefit analysis on each instance and detail to justify its inclusion.
5. Modify (add, change/reorganize, delete) elements of your grove based on project needs.

Steps 2 and 3 can be combined by copying and customizing templates for requirements information types (i.e., fruit templates).

When Bob reviews idea management tools, he uses the following requirements for a lightweight RM tool. The tool must be actively supported by a reliable supplier and enable a user to:

1. choose types of requirements information to be recorded
2. change information choices during a project at little cost
3. create and manage outlines
4. create and manage tables
5. describe and manage states
6. manage links between content elements
7. manage links from content to external information
8. perform Boolean searches of its contents
9. export and print search results
10. export RTF or Word files [for import into a heavy-weight RM]
11. collaborate with geographically distributed groups

3. Development

These requirements help Bob review over 30 tools. He finds a few strong candidates, but as often happens when repurposing a system, bugs and limitations are uncovered and new functionality needs to be added. Thus, the cooperation of the platform developers is essential.

When the platform modifications are complete, Bob builds a template file (pre-built framework) to be tailored at the start of a project. The template has a transaction-processing (TP) bias, since that is what Bob's team is about to build. Other templates can be developed as needed for other types of systems [7].

Bob uses the TP template to build a lightweight RM for his current project – the development of the KidsIdeals website. KidsIdeals is to be a website that discounts kid-related products – similar to Groupon™ but with important differences.

4. Results

The following samples show the TP framework used for KidsIdeals.

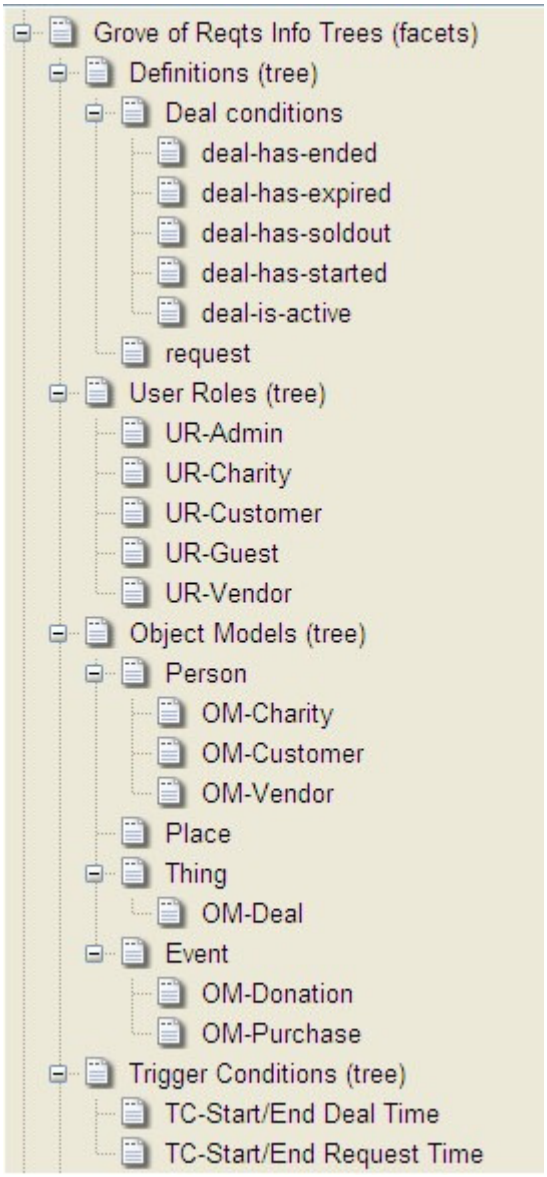

Figure 2 – Start of the Grove

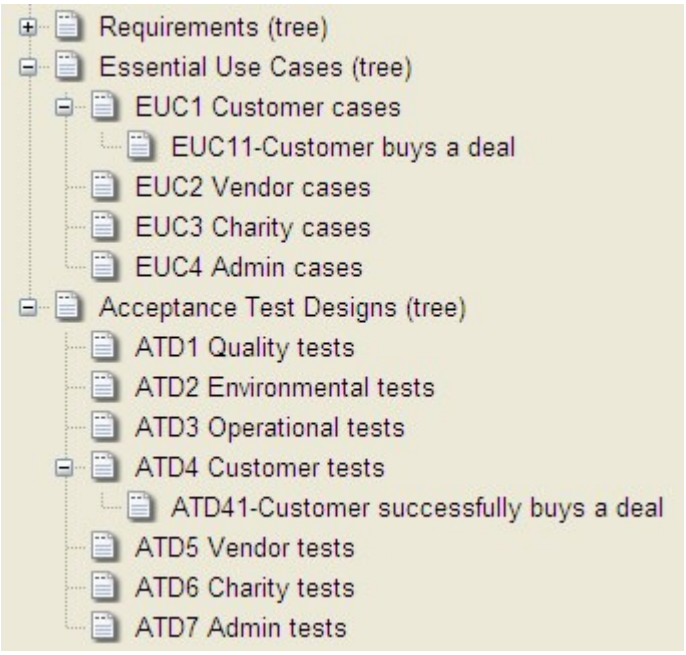

Figure 3 – End of the Grove

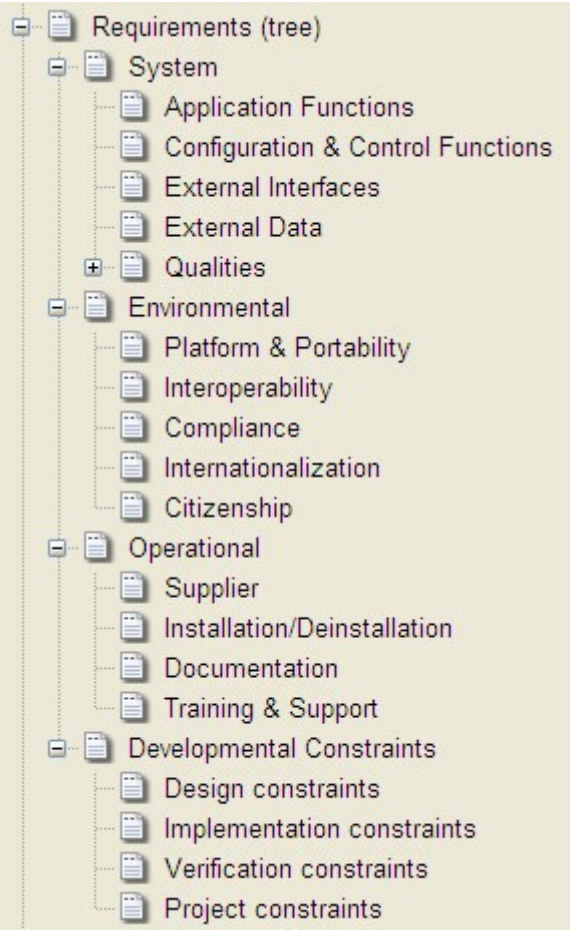

Figure 4 – Top of the Requirements Tree

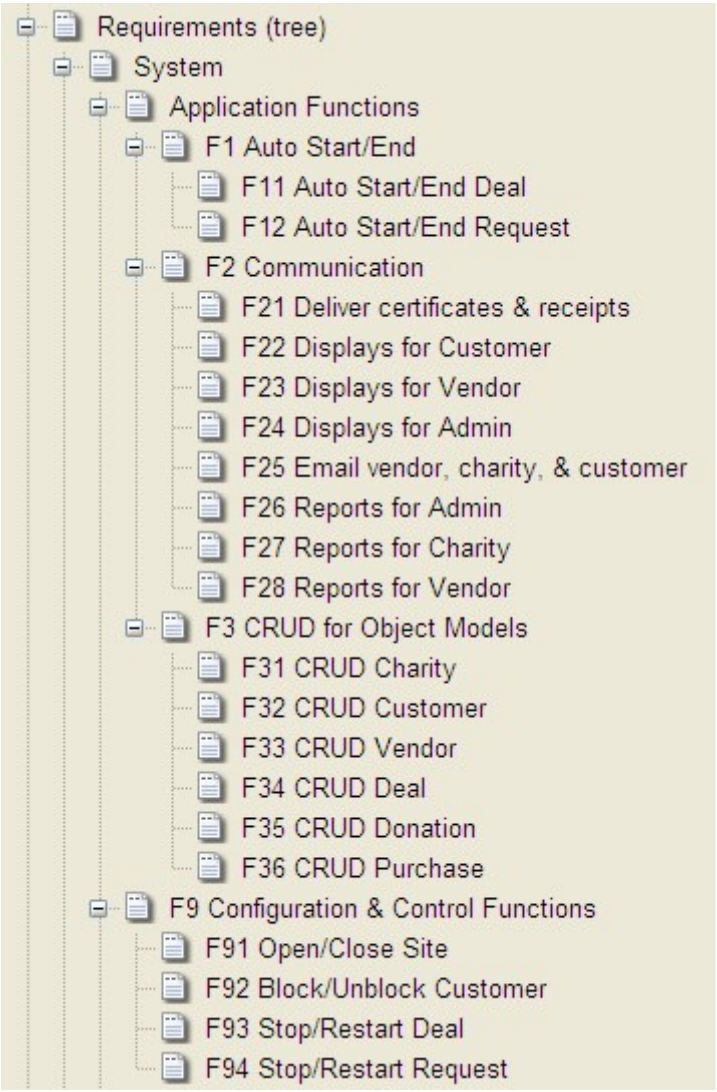

Figure 5 – Some Functional Requirements

All requirement attributes are user-defined allowing Bob to include just those he wants (Figure 6). He also includes just the states and links he wants to manage.

Details

Limits & Constraints (on both functionality and usage)

Assumptions/Rationale

Implications/Expectations

Risk Factors

a. Developer Understanding states = < Sup, Lim, Deep >

b. Likelihood of serious defects = [high, medium, low, NR]

c. Threat level = [high, medium, low] -- (threat of valid reqts to design, schedule, or budget)

Other Attributes

a. Sources/Parents:

b. Value (to org) = [essential, necessary, desirable]

c. Cost (to impl) = [high, medium, low]

d. Priority (to impl) = [high, medium, low]

e. Version =

f. Iteration:

g. Allocation:

Figure 6 – Some Attributes of a Function Requirement

Bob's RM satisfies his requirements and answers all his important questions. Building on an idea management platform not only provides development leverage but also provides the platform's base functionality for other activities. All this for an out-of-pocket cost of less than $100.

Bob's tool can be used to create both a traditional assortment of requirements information (Figures 2-6) and an automated deck of index cards with front and back sections. This means that a wide variety of information and organizations is possible. The tool can help visualize and manage developer understanding [8] as well as manage requirements.

In addition to his own project, Bob wonders who else in his organization might be interested in the tool. He considers teams using index cards, word processors, or spreadsheets as well as teams having, but not using, heavier weight RMs (i.e., having RM shelfware). He will have to see what other teams are using.

Just then, Bob awakens from his dream to find his requirements deck scattered on the floor. The dream seemed so real. [9]

5. References

1. **…** **INCOSE Requirements Management Tools Survey** Available at www.incose.org/productspubs/products/rmsurvey.aspx
2. Crosby, Kevin (2010) "**Lightweight Requirements Management with Confluence**" Available at nvisia.com/techs/?p=367
3. Delgadillo, Lorena and Gotel, Orlena (2007) "**Story-Wall: A Concept for Lightweight Requirements Management**" Proceeding of 15th IEEE International Requirements Engineering Conference
4. Murtland, Chris (2008) **"List of Outliners"** Available at www.outlinersoftware.com/topics/viewt/807/0/list-of-outliners
5. **…** **"List of Personal Information Managers"** Available at en.wikipedia.org/wiki/List_of_personal_information_managers
6. **…** **"List of Concept Mapping and Mind Mapping Software"** Available at en.wikipedia.org/wiki/List_of_concept_mapping_and_mind_mapping_software
7. Jackson, Michael (2000) **Problem Frames: Analyzing & Structuring Software Development Problems** Addison-Wesley Professional
8. Gelperin, David (2011) "**Visualize and Manage Developer Understanding**" Available at www.literm.com/manage-understanding.php
9. LiteRM realizes the dream. www.LiteRM.com