Requirements Management Made Easy

by
Alan M. Davis
Ed Yourdon
Ann S. Zweig

Omni-Vista, Inc.
4350 ArrowsWest Drive
Colorado Springs, CO 80907
www.omni-vista.com

adavis@omni-vista.com
ed@yourdon.com
azweig@omni-vista.com

Abstract

Requirements management has been discussed for at least fifteen years. As a discipline and as a practice, it has become more and more complex. We have lost sight of the fact that requirements management was created to *simplify* software development, to reduce its cost, and reduce the inherent risk associated with building software. Instead, requirements management has become yet one more chore, one more error-prone activity. The purpose of this paper is to distill the common wisdom of requirements management, and to return it to a simple method of ensuring that software development organizations build the right software.

Keywords

Requirements, Features, Elicitation, Triage, Specification, Requirements Specification, Requirements Management, Requirements Analysis, Analysis, Systems Analysis

1. Introduction

Requirements are those externally observable characteristics of a system that a user, buyer, customer, or other stakeholder desires to have present in the system. Requirements management is the set of activities encompassing the collection, control, analysis, filtering, and documentation of a system's requirements. Requirements management consists of three activities:

- *Requirements elicitation*. The art of understanding the needs of stakeholders, and collecting them in a repository for future analysis. The needs can be expressed quite abstractly and in terms of a problem, e.g., "I want to reduce my billing error rates by at least 35%." And the needs can be expressed quite specifically and in terms of a solution, e.g., "I want there to be a large red button on the operator's console." In all cases, these needs are called *features*.
- *Requirements Triage*. The art of deciding which features are the appropriate features to include in the product. Rarely is it possible to satisfy every requested feature gathered from every stakeholder during the elicitation activity. Disparate priorities, limited resources, time-to-market demands, and risk intolerance are but a few of the reasons for this. Triage takes into considerations all the painful realities of the marketplace and makes the decision of which features will we build now, which will be built in the next release, and which will be deferred until even later.
- *Requirements Specification*. The art of detailing the exact external behavior of a system that will address the features selected during the triage process. The level of detail of these requirements depends greatly on the situation. For example, if specifying a hand-held remote mouse, it might be sufficient to state "The system shall contain three programmable buttons, corresponding to the three buttons on a standard three-button mouse." However, if the device is to be mounted in a holster and controlled by robotic fingers instead of being hand-held, then the statement of requirements for the buttons would need to be considerably more detailed.

Seventy-one percent of all software development projects result in complete failure (i.e., premature cancellation or shelfware upon completion)[1]. Poor requirements management is generally considered one of the major causes for product failure[2,3]. After all, if we do a poor job of understanding our customers' needs, if we do a poor job of deciding the right features to build, and if we do a poor job of writing down what we think we want out of a system, how can we possibly expect a successful project? All the software development techniques and tools and whatever level of CMM maturity you have achieved will be of no use to you if you are not building the "right" product.

This paper is organized into three main sections, each addressing the techniques, tools, and common wisdom of each of the three aspects of requirements management.

2. Requirements Elicitation

Requirements elicitation is the art of determining the list of all possible features that you might want to include in your next product. This is the time to be broad and inclusive. The candidates may come from potential customers, existing customers, customers of customers, marketing, sales win/loss reports, change requests, trouble reports, features rejected from earlier releases, internal development personnel, manufacturing, customer support personnel, inventors/innovators within the company, management, and so on. In the following paragraphs, we will use the term *stakeholder* to identify any of these sources of candidate features.

The techniques of requirements elicitation vary considerably depending on the specific situation. For example, if you have identified a set of key stakeholders who are committed to mutual success and are willing to help you help

them, you might seriously consider facilitated group sessions (a.k.a. brainstorming). To conduct one of these sessions, assemble key stakeholders in a room with plenty of wall space. Announce the purpose of the meeting, i.e., to learn potential features for version *x* of product *y*. Advise the participants that the goal is to synthesize the largest possible list of candidate features. To do so, participants should encourage each other as much as possible, and not criticize any idea. In many cases the seemingly most outrageous idea can become the catalyst for the creation of the best ideas.

Another approach, useful especially when you are unable to isolate such stakeholders in one place, you might consider an analysis of needs, pains, and solutions as prescribed by Bosworth[4]. In this approach, the target customers are analyzed in detail. Their activities are modeled, their pains identified, the impacts of those pains explored, current solutions to those pains are delineated, and the features that could alleviate those pains most effectively are listed. Obviously, the more information you can gather from "real" potential customers the better. But in any case, the idea of deriving candidate features from a thorough understanding of customers and their pains is difficult to argue with.

Other elicitation include interviewing[5], ethnomethodologic studies[6], surveys, and requirements workshops[7].

Interviewing is ideal when you have identified a reasonable-sized set of representative stakeholders, and it is feasible to meet them each individually. It is also helpful when competitive, logistic, or political issues make it impractical to gather the stakeholders in one place for brainstorming. Ethnomethodological studies try to eliminate preconceived notions of problems and solutions to those problems. The objective third-party observer in these studies eliminates the bias inherent in interviewing and brainstorming. Surveys work especially well when the population of representative stakeholders is too large to assemble in one place or to large to be interviewed. However, they only work well when you have very specific questions that you want answered. They eliminate any of the spontaneity inherent in interviews or brainstorming. Requirements workshops are a special case of brainstorming.

Tools are available to assist in a variety of elicitation techniques. For example, Computer-Assisted Cooperative Work (CSCW)[8] tools can facilitate brainstorming sessions whether the stakeholders are co-located or are physically distributed. Forms (e.g., the pain sheet) of Bosworth[9] (while not automated to our knowledge) can be used easily to organize and analyze pains, impacts, competition, and remedies. On-line polling tools[10] can gather opinions of many stakeholders, and thus be useful aids to performing surveys.

Requirements elicitation is the most difficult of the three requirements management activities because you are creating something from nothing.

The result of elicitation is a list of candidate features, each annotated with relative technical risk, an estimate of effort required to address the feature, and a relative priority or measure of importance from the customers' perspective. Tools exist to record this result. These range from basic tools such as word processors, spreadsheets, and databases, to simple requirements-specific repositories such as Omni-Vista RM[11], to more complex requirements management tools such as Rational RequisitePro[12], QSS DOORS[13], and TBI Caliber[14].

3. Requirements Triage

Requirements triage is the process of deciding precisely which product is to be built.

From a purely development manager's perspective, triage is quite simple. Development managers estimate the effort and time required to satisfy the stated features. They compare these with the budgets and schedules given to them. If they are not compatible, the development manager simply removes features until the work involved in compatible with the given schedule and budget! Presto, triage is complete.

Just one problem however! Removing features may have a positive affect on development risk, but this logic completely ignores impacts[15] on market, price, revenue, and profit. Thus, requirements triage *must* consider marketing, financial, *and* development factors.

The goal of requirements triage is quite straightforward:

- a set of features,
- which can be implemented using available resources and with acceptable levels of risk,
- which can be sold at an acceptable price to a known market,
- in sufficient quantities to achieve
- acceptable levels of revenue and profit,
- and thus achieve a reasonable return on investment.

You will note that the naïve development manager's view of triage explained above considers only the first two bullets.

The variables at the disposal of the team performing triage are:

- Add, delete or change a feature
- Make the delivery date earlier or later
- Increase or decrease the resources applied to development
- Increase or decrease the price
- Increase or decrease costs of good sold
- Increase or decrease the resources devoted to marketing and sales.

Triage then is the process of altering assumptions about these six variables until the results are acceptable. Once you arrive at an acceptable set of values for the variables, most companies produce a product plan. A *product plan* (also called a "business case") details

- which features are to be included,
- what market is to be addressed (or for internal developments, what internal problem or opportunity is to be addressed),
- what percentage of that market is likely to be successfully sold (or for internal developments, what percentage of potential users will actually use it),
- how many resources (and over what time frame) are required for development,
- how much risk exists for development, sales, meeting the schedule, meeting the development budget,
- how many units will be sold and at what price (or for internal developments, how many times will the system be used),
- how much revenue will be generated (or for internal developments, how much savings or revenue or reduction in cost, or reduction in errors will occur), and over what time frame.

Tools are available to assist in some aspects of requirements triage. For example, Omni-Vista SP[16] allows users to witness the effects of features, resources, and cost of goods sold on revenue, risk, profit and return on investment. Primavera Monte Carlo 3.0[17] allows users to simulate over time the effects of making key product planning decisions. QSS TechPlan 2[18] allows users to compare the effects of decisions on technology, risk, and marketing.

4. Requirements Specification

Requirements specification is the task of documenting the precise external behavior of the system that is to be built. It takes the features selected during requirements triage and expands them into considerable detail. The primary purposes of doing this are to ensure that (1) customers and developers have the same understanding of what is to be built, (2) all developers have identical understanding of what is to be built, (3) testers are testing for the same qualities that developers are building, (4) management is applying resources to the same set of tasks that the developers are performing. Traditionally, these requirements have been documented in a software requirements specification. However, more recently there is a trend toward maintaining the requirements in a database or repository of discrete requirements, rather than a formal document.

Numerous textbooks have been written to teach how to properly document requirements[19]. Key to all these works is that every documented requirement should be at least[20]:

- Correct, i.e., it describes something the stakeholders want,
- Consistent, i.e., it does not conflict with other requirements,
- Unambiguous, i.e., it has only one possible interpretation, and
- Verifiable, i.e., there must exist some means by which we can check that the final system meets this requirement.

Numerous tools exist to record these annotated requirements including Omni-Vista RM[21], Rational RequisitePro[22], QSS DOORS[23], and TBI Caliber[24].

5. Summary

Requirements management has somehow grown over the years to be more and more complex. In actuality, it is a quite straightforward set of activities. Performing each of the activities is actually quite simple. The most complex aspect of requirements management is doing the background investigation (e.g., learning what effect on sales will occur when a feature is eliminated). However, these investigations *must* be done whether or not you are performing requirements management! Otherwise you are embarking on a highly risky business route. You are inviting your company to become a statistic; to become part of the 71% of projects that are never completed or become shelfware.

References

1 See www.standishgroup.com

2 Jones, C., *Patterns of Software Systems Failure and Success*, International Thomson Press, 1996.

3 McConnell, S., *Rapid Development*, Microsoft Press, 1996, p. 86.

4 Bosworth, M., *Solution Selling*, New York: McGraw Hill, 1995.

5 Gause, D., and G. Weinberg, *Exploring Requirements*, New York, NY: Dorset House, 1989.

6 Goguen, J., and M. Jirotka, *Requirements Engineering*, Boston, MA: Academic Press, 1994.

7 Leffingwell, D., and D. Widrig, *Managing Software Requirements*, New York: Addison Wesley, 2000.

8 Spurr, K., et al., *Computer Support for Co-operative Work*, New York: John Wiley, 1994.

9 Bosworth, M., *Solution Selling*, New York: McGraw Hill, 1995.

10 See for example http://apps3.vantagenet.com/zpolls/

11 See www.omni-vista.com/products

12 See www.rational.com

13 See www.qssinc.com/products/doors

14 See www.tbi.com/products/asq.html

15 Reinertsen, D., *Managing the Design Factory: A Product Developer's Toolkit,* The Free Press, 1997.

16 See www.omni-vista.com/products

17 See www.primavera.com/products/monte.html

18 See www.qssinc.com/products/visiontools/techplan.html

19 Davis, A., *Software Requirements*, Englewood Cliffs, NJ: Prentice Hall, 1993; and Leffingwell, D., and D. Widrig, *Managing Software Requirements*, New York: Addison Wesley, 2000; and Kotonya, G., and I. Sommerville, *Requirements Engineering: Processes and Techniques*, New York: John Wiley and Sons, 1998.

20 Davis, A., et al., "Identifying and Measuring Quality in Software Requirements Specification," *IEEE International Software Metrics Symposium,* May 1993; reprinted in Thayer, R., and M. Dorfman, *Software Requirements Engineering*, Los Alamitos, CA: IEEE Computer Society Press, 1997, pp. 164-175.

21 See www.omni-vista.com/products

22 See www.rational.com

23 See www.qssinc.com/products/doors

24 See www.tbi.com/products/asq.html