The Five Levels of Requirements Management Maturity

by **Jim Heumann**
Requirements Evangelist
Rational Software

Maturity: the quality of sound judgment associated with adult humans.
-- The Wordsmyth English Dictionary/Thesaurus

Being mature means being able to see the big picture and make good choices. In a business context, that means basing decisions on a clear understanding of the full range of both the costs and benefits of doing one thing over another.

This article looks at the decisions organizations make and what they do as they move up the scale in requirements management maturity (RMM). Just as hiking up a mountain has a cost (in energy and time), so does this climb upward. Therefore, as we look at the benefits of reaching higher levels of maturity, we will not ignore the investment required in terms of time, effort, and money. In addition, we will analyze how automated requirements management (RM) tools can help support organizations striving for greater RM maturity.

Those familiar with the CMM (Capability Maturity Model) from the Software Engineering Institute (SEI) will note some similarities to our parallel model, which has no direct relationship to the CMM save one: Achieving Level Five of the RMM will assuredly help an organization get to at least Level Three of the CMM. Of course, it's important to keep in mind that attaining a high level of maturity in a single area, such as requirements management, is easier than attaining overall organizational process maturity.

The five levels of maturity for our RMM are: 1) written 2) organized 3) structured 4) traced, and 5) integrated. We will use these categories to

partition requirements management practices, starting at the lowest level (One) and moving up through Level Five.

Chaos: No Requirements

Actually, there is one other level on the requirements maturity scale: Level Zero -- no requirements. At Level Zero, organizations fly by the seat of their pants; they make broad assumptions that they know what to build; they gamble that the time they save by not gathering requirements will not be squandered later because they deliver either too much or too little. Sometimes this gamble works, but more often than not, a product is released that is missing functionality, has functions that are not needed, or is of poor quality. These problems will have varying degrees of impact, depending on the customer and the criticality of the software, but if the consequences are severe enough, they may prompt the organization to start "doing requirements" (and to create an RM process along the way).

Level One --Written Requirements

The first step up from the chaos of no requirements is simply to write the requirements. White boards and sticky notes don't count. Any medium that cannot be backed-up and restored is so fraught with risk that we will not consider it here.

Once you write requirements, several benefits become obvious.

First, you have a real basis for a contract with your customer. If you write them well, the requirements can clearly state your understanding of what the customer wants you to build, and they can read the requirements and agree (or disagree).

Second, everyone on your development team has a basis for doing his or her work. This obviously includes the architects and designers, who can start thinking about how to architect the system to meet customer expectations, but it also includes the testers, who can get a very early start writing test cases, upon which they can later base test procedures and scripts.

Third, as you staff up the project, new members, too, will have a source for figuring out what the application or system is supposed to do. And because written requirements can be backed up and restored if necessary, you will have taken a big step toward reducing risk.

How about the cost? Someone -- or perhaps a team of people -- must take the time to do the writing. And unless the team is making up the requirements, there is effort involved in talking to the customer to find out what they want. Maintenance is also a time/cost factor; unless they are kept up-to-date, the requirements will become useless. And whoever is responsible for maintenance must learn and implement some "tool" to do the writing, even if it is simply Word or Notepad.

Is it worth the time and effort? To answer this question, you have to compare the cost to the potential consequences of not expending the

effort to record the requirements. What will happen if you don't build and deliver what your customer wants? What will happen if you build too much? If the answer is "not much," then it's not a problem. But if delivering the wrong system would mean a major drop in your stock price or an unhappy customer, well, then you might want to make the investment.

Level Two -- Organized

At this level an organization deals with things like the quality of the requirements, their format, their security, where they are stored, and how to keep track of different versions.

The goal of a requirement is to clearly communicate the proposed behavior of the software to a diverse set of constituents: users, customers, and other stakeholders, as well as the development team. This is a tall order and not easily accomplished. A good set of requirements does this job well; a bad one does not. Good requirements are understandable by stakeholders who specify them; they are also usable by the architect, developers, and testers. To achieve this, they must be not only readable, but also unambiguous and testable.

Formatting

Requirements must also be formatted effectively. Consistent numbering schemes, headers, fonts, and a good table of contents make your documents easier to read, understand, and use. Even a well-written requirement can be rendered useless if it's poorly formatted. Formatting is a simple thing, but it's often overlooked in the rush to get the requirements "done." Document templates can help, as can simple formatting standards for requirements documents.

Accessibility, Security, and Version Control

Have you ever become frustrated because you couldn't find a requirements document? At Level Two, you need a central, well known location for the requirements, one that is accessible by all users. Think about security, too. Whether you use simple file system security or a more sophisticated technique, limiting the ability to modify requirements to authorized persons only can help ensure the requirements' integrity.

Instituting version control for your requirements can also save time and prevent frustration by ensuring that you always know whether or not you are working on the most current versions of the specifications. In addition, you will always know who has made a change and why they made it.

These are not the only benefits of getting to Level Two, because you will automatically enhance the advantages of Level One. When your requirements are more readable and easier to understand (and more trustworthy), you will have a better basis for a contract with the customer, the development team will find the requirements easier to use, and new staff will be able to come up to speed more quickly.

Costs associated with getting to Level Two relate mostly to training and review. Writing quality requirements is not a simple task. Unless you are lucky enough to have skilled analysts already on staff, you will have to train your team. Getting to, and staying at, a given maturity level will require consistent review of requirements documents and some level of "process enforcement." These are additional tasks that take time.

Of course, these costs are counterbalanced by the obvious advantages in ensuring that the requirements are right: less rework and better customer acceptance, to name two. If you have to get it right the first time, the costs are probably worth it.

Level Three -- Structured

Getting to Level Three involves being more specific about the "types" of requirements you gather. Are they functional or nonfunctional? Business or system? Features or software requirements? How about customer, marketing, and user requirements? Making these distinctions helps you get a better understanding of the requirements and manage them better. Getting to Level Three also means adding information about the requirements, beyond the basic text. You can provide this additional information in the form of "attributes," which will help you take a big step up in managing and reporting on the requirements.

Getting Your Types Straight

As you think about the many possible types of requirements, you should consider two particular issues.

The first issue arises if you do not distinguish among different types of requirements. If your current requirements specification simply contains a big list of requirements with no indication of their type, it is likely that the list contains a mix of different types. For example, one requirement might be "The supplier's help desk shall respond to 90 percent of all trouble tickets within four hours," and another might be "The system shall support automatic checking of prerequisites."

Although these are both valid requirements, they are clearly directed at different concerns. The first is a requirement for the support organization of the company supplying the software; the second is a specification for what the software must *do*. Without indications of type, a long list of requirements can cause confusion and waste readers' time. Those looking for software requirements will get distracted by the support requirements, and vice versa. It is also difficult to run a report, for example, to show all of the software requirements.

The second issue relates to having requirement types but no agreement on what they mean. An interesting exercise in such a case is to ask team members who must use the requirements to write definitions for the various requirement types. You might be surprised at the variety of interpretations. Clearly, if one person defines a "user requirement" as "a business need the end user thinks the future system must perform so that he can do his job" and another defines it as "a requirement for the user

experience," then there is a problem.

Attributes

Let's turn now to the concept of requirements attributes, another Level Three capability. All requirements are not created equal: Some are more important than others; some are more stable than others; some may be intended for one release, some for another release. These are important things to keep track of, and adding attributes for requirements can help you do so. Attributes include information that supplements the requirement text and helps you better understand and manage your requirements.

But how do you know what attribute information is needed? It depends on the needs of those who will use the requirements information. One common mistake is to blindly use a predefined set of attributes from another project or requirements tool (like the one that comes with Rational® RequisitePro,® for example). Project templates are a great start, but you will likely have to modify the template to your needs. Otherwise, it may saddle you with attributes that you don't need and omit many you do need. For a large project that assigns requirements to different analysts, an "owner" attribute would be very useful; but such an attribute may not make sense for a small project on which one person "owns" all the requirements. You need to understand how the requirements will be used in order to understand what attributes are necessary. What reports and queries will you need to support? What metrics must you keep? Getting answers to these questions up front will help you start on the right foot.

Benefits of getting to Level Three revolve around better understanding and easier management. Well-structured requirements clearly identify different requirement types, and attributes provide the ability to query and filter groups of requirements; this means that team members have to do far less detective work and guesswork to identify their responsibilities and tasks. Better typing of requirements also provides greater assurance that the team has identified all important requirements. The main cost of getting to Level Three is in planning and maintenance. Determining appropriate requirement types and attributes is not a trivial task. A mini-project in itself, it involves talking to the requirements users to determine what they need. Usually this information is captured in a Requirements Management Plan (RMP). Then, requirements attributes are of little use if they are not kept up-to-date, so there is a maintenance burden that goes beyond the one for Level Two. There is an increased cost too, because determining the correct attribute values takes time. Often, when organizations get to Level Three, they institute the use of a requirements management tool (like Rational RequisitePro). This has large benefits that often outweigh the cost of purchasing, updating, and administering the software.

Level Four -- Traced

Implementing the previous three levels will get you to the point where you can determine and track requirements relationships. Most systems of any

significant complexity will have a hierarchy of requirements. In a systems engineering environment, this hierarchy starts with system requirements and moves down to subsystem requirements, program requirements, and element requirements. In the Rational Unified Process, the hierarchy starts with user needs, proceeds to features, and then goes on to use cases. The ability to keep track of these relationships is commonly called *traceability*, and it entails identifying and documenting the derivation path (upward) and allocation/flowdown path (downward) of requirements in the hierarchy. Traceability provides the ability to understand how changes to one requirement might affect others (impact analysis) and can also help determine if your requirements are complete (coverage analysis).

Usually, an organization at Level Four will develop an explicit traceability strategy prior to starting a project and then document it in the requirements management plan. The strategy will define the requirements levels and how they fit in the hierarchy. In addition, it will lay down some "rules" for requirements relationships. For example, one rule might be that for every "user need" there must at least be one "feature," and for each feature there must be at least one use case. Implementing a requirements management process like this sets up capability for sophisticated reporting, as we described earlier.

Coverage analysis reports show whether each high-level requirement has a lower level requirement associated with it. This helps you determine whether you have coverage holes. For example, if you have a feature that does not have a use case associated with it, the resulting software may be missing functionality. Or if you have a use case with no associated feature, you may be implementing functionality with no business value.

Impact analysis reports are primarily intended for managing change. They clearly show how a change to one requirement may impact others, which makes it much easier to either justify or resist changes. For example, if someone desires to change a feature, and an impact report shows that doing so will cause changes to several use cases (and slow down the project schedule), it is easier to see how to weigh costs against benefits for the feature change.

These benefits of tracing requirements are significant, but again, they are not without cost. The effort involved in entering and maintaining the trace relationships is not trivial. Defining, running, and analyzing the coverage and impact reports takes time and effort. Requirements tracing can be done manually via simple tables in Word or Excel in very small projects, but complex projects often need a requirements management tool like Rational RequisitePro. The benefits of using such a tool are significant, but as we noted earlier, there is a cost associated with purchasing, maintaining, and providing training for the tool.

Level Five -- Integrated

It is often the case that requirements are used up front to get agreement from the customer on what the software is supposed to do, but then those requirements are not really tied in to the way the software is developed. This results in stale requirements and software that doesn't meet its

objectives. Reaching Level Five means integrating the requirements management process with the rest of your software development environment. It means using requirements directly in software design, change management, testing, and project management.

Software that does what the customer expects is built to comply with the requirements -- that is, the team's software development process uses the requirements as a key input. The system's architects and designers follow a process to ensure that all of the requirements are implemented in the design; the Rational Unified Process does this by treating use cases as the input artifact for the analysis and design discipline.

Integrating requirements into your change management process helps ensure that no changes are made to requirements without review and approval. And relating each change request to an existing or new requirement helps to limit feature creep.

Requirements-based testing is also an important part of verifying that the software meets its objectives. Just as designers must use requirements to design the system, testers must use them to create test cases and other testing artifacts.

Since requirements are the basis for the whole development process, project managers should have direct access to a project's status in relation to the requirements. This includes metrics about new requirements, requirements implemented, requirements tested, and requirement change requests.

A comprehensive, requirements-based software development process as described here takes significant planning, training, and process enforcement. Software development tools are also an important part of implementing such a process. Visual modeling tools such as Rational Roseý or Rationalý XDE,ý change management tools such as Rationalý ClearQuest,ý requirements management tools such as Rational RequisitePro, and project status reporting tools such as Rationalý ProjectConsole, can all significantly enhance your organization's ability to get to the highest level of RM maturity. Again, although these tools have clear benefits, they also have associated costs for purchase, maintenance, and training.

Requirements Management Tool Support

Until Level Five, it is theoretically possible to do everything that we have talked about either "manually" or with general-purpose tools like a word processor and spreadsheet. However, starting at Level Two, an RM tool can help you be far more efficient and consistent. Table 1 shows how the important features of Rational RequisitePro support key characteristics of the five RM maturity levels.

Table 1: How Rational RequisitePro Supports RMM Levels

Rational RequisitePro Feature	Maturity Level
Dynamic integration between Microsoft Word ® and requirements database	One -- Written Two --Organized
Secure central requirements repository	Two -- Organized
User security	Two -- Organized
Requirements revision history	Two -- Organized
Web interface	Two -- Organized
Requirements project templates	Three -- Structured
User-defined requirement types, requirements attributes, and document types	Three -- Structured
Requirements attributes and query capability	Three -- Structured
Requirements traceability and coverage analysis	Four -- Traced
Impact of requirement change	Four -- Traced
Integrations with other software development tools	Five -- Integrated

RUP Support

The five levels we have been talking about encompass a requirements management process. A process determines and documents who does what, when they do it, and what things they produce. It takes time and effort to decide what the process should be and then to document it. An organization can save resources by not reinventing the wheel, and by adapting a standard process to suit their own needs. The Rational Unified Process (RUP), for example, is divided into disciplines, one of which is the Requirements Management Discipline. If you look at that discipline, you will notice that it contains a workflow detailing many of the concepts we have talked about here. Starting with RUP can give an organization a valuable head start on improving RM maturity.

Evaluate and Move Up

The goal of this article has been to describe the best practices organizations adopt as their requirements management efforts mature, and they move to new levels of sophistication. The five levels we described should provide a framework for you to evaluate your own organization and understand where you stand and what needs improvement. It should also provide a way for you to understand the benefits and costs involved in moving up to higher levels of requirements management maturity.

For more information on the products or services discussed in this

article, please click here and follow the instructions provided. Thank you!