



## *Why Are Requirements Critical?*

We all know that requirements are important. We've seen the statistics: 31% of software projects are canceled before completion. Requirements errors are the largest class of errors (41%) in large projects. Finding and fixing requirements errors consumes 70-85% of total project rework costs.

We know that these statistics are worth paying attention to, and many of us make a genuine effort to document our requirements. But, something must not be working. Many requirements efforts are doomed because they suffer under some crucial misconceptions. Perhaps the most important is the assumption that business owners know what they want right up front.

Have you ever been involved in a home or office building project? If your experience is like mine, I'll bet you can remember changing your mind once you saw the architect's drawings. I'll bet you remember changing your mind once the work began. And I'll bet you remember at least once when the plumber or the electrician or the carpenter made some assumptions that you didn't agree with and overruled.

Did you end up with the house or office you wanted? If you did, it's undoubtedly because you stayed involved and because, through the architect's drawings, you had a way to communicate. The comparison of building a house to building a software system is one you've undoubtedly heard before. Nonetheless, the comparison is extremely useful, and I believe that there are a few crucial lessons that we can draw.

- Lesson one: You may have expressed your initial ideas in words, but the architect turned the words into drawings. Those drawings helped you 'see' what the finished product would be like.
- Lesson two: The drawings you and the architect worked with were presented in your terms, not the electrician's or the plumber's. You could understand them.
- Lesson three: You, the one paying the bills, had the opportunity to be involved and make changes throughout the life of the remodeling effort, even when it was in the hands of the technicians.
- Lesson four: The requirements, the original drawings that you and the architect worked through, drove the rest of the project. They produced electrical specs, plastering specs, and so forth. If the assumptions in those drawings changed, the changes rippled through.

Just as in building a house, a good software requirements process *must* have a way to help the business owners extract, discover, and capture what they want in their own terms. A good software requirements process *must* have a way to communicate unambiguous requirements to the developers (the plumber, the electrician). And a good software requirements process *must* have a way to accommodate and even encourage change throughout the life of the project (you, the owner, are included in the process all along the way).

If your requirements process doesn't address these lessons, it won't succeed.



## ***Requirements Gathering Needs Both Words and Pictures***

Most business sponsors are used to expressing themselves in words, and it is important to capture their initial goals and objectives in a medium they're comfortable with. In addition, text statements are a good way to capture operational requirements such as the number of users that must be supported, or the necessary levels of security.

Text-based requirements tools let you write a document, designate text within the document as a requirement, and then create a repository of requirements. The repository provides a mechanism for change management and for traceability throughout the project life cycle.

Most important, these tools offer requirements *management*. Requirements will inevitably change and you must have a way to track those changes and understand the impact of one change on other requirements.

But what about process (or functional) requirements? There are situations where it might take several pages to describe the process of, say, taking an order. In this situation, a visual approach to understanding a business process flow seems preferable to a written one. (Again, think of those architect's drawings. Is it more effective to describe in words what the new bay window should look like, or to draw it?)

The solution is to find a way to marry the benefits of text requirements and requirements management with visual models. Requirements gathering needs to address visual *and* text requirements, and *both* kinds of requirements must be tracked and managed.

## ***Requirements Must Be Gathered and Presented in Business Terms***

For those requirements that would benefit from a visual approach, you will need to find models that speaks in *business* terms, not IT terms. Many business owners are impatient with entity relationship models and even more so with class models. These models may provide the information IT needs, but may not be particularly understandable or useful to the business side of the house. As others have noticed, the real world of business and the technical world of computers are inhabited by two different kinds of people. Each has its own culture and language.<sup>1</sup>

Remember that what you're trying to accomplish is to discover and document the business owner's requirements. If the business owner can't work usefully with the models, you've defeated your purpose.

## ***Business Owners Must Be Responsible for Requirements throughout the Life Cycle***

All the methodologists and researchers agree that process owners and system user input is critical to the success of a software project. According to research done by the Standish Group and reported in 1994 and 1995,<sup>2</sup> the top reason for project failure is "lack of user input."

---

<sup>1</sup> This idea was developed by my colleague Neville Haggerty of Compedia.

<sup>2</sup> Standish Group, *The CHAOS Report*, 1994.



# White Papers

---

Doreen Evans Associates, Inc.

At least some of those project failures included JAD sessions to collect user information. They may have even had the users themselves write the requirements. Shouldn't that have been enough? Apparently not.

As one business user said, "We've written requirements documents, but developers can ignore written instructions as well as verbal. What we need is someone to make sure each requirement is implemented as written."

Well, who should that someone be? I believe that it's the business owner him/herself. But if your requirements are 'thrown over the wall' to development, that won't be possible. You can document your requirements, you may even get your developers to read them, but then what? The development of the application is now out of your hands. Do you really want to abdicate that responsibility? If not, you need a way to keep your finger in the pie throughout the life cycle.

## ***Requirements Are Central to the Entire System Life Cycle***

How many legacy systems are out there with unknown business rules buried in the code? How many systems are we currently building without good documentation? Why can't we find a way to ensure that business people not only specify the requirements, but also have a mechanism to understand what the system when it's built will actually do?

Requirements analysis is often thought of as the first phase of the life cycle, with the implication that requirements are "done" before you go on to analysis, design or implementation. I believe that the models built during analysis ARE the requirements. These requirement models can lay the basis for the activities of many groups. Let me give you some examples.

- The user documentation group must have online help and written documentation ready to go when the application is ready to become operational. If you can give them models that portray the various functions of the application and what they're intended to accomplish, writers can get a good portion of the documentation written even before the application is in alpha.
- The group responsible for project management (at one of our recent clients, this group was known as the Project Control office) must know the number of requirements that are in process and their status. Are they approved? Are they undergoing change? In addition, this group will be interested if you can predict the number of function points so that they can estimate development time and costs.
- User testing must be able to write test scripts to show that the application meets user needs. What if the models you build for your requirements could generate these scripts, or at least a significant portion of them? What if, whenever the requirements were updated, the test scripts could be regenerated to reflect those changes?

The following figure illustrates the roles that I believe use requirements information.

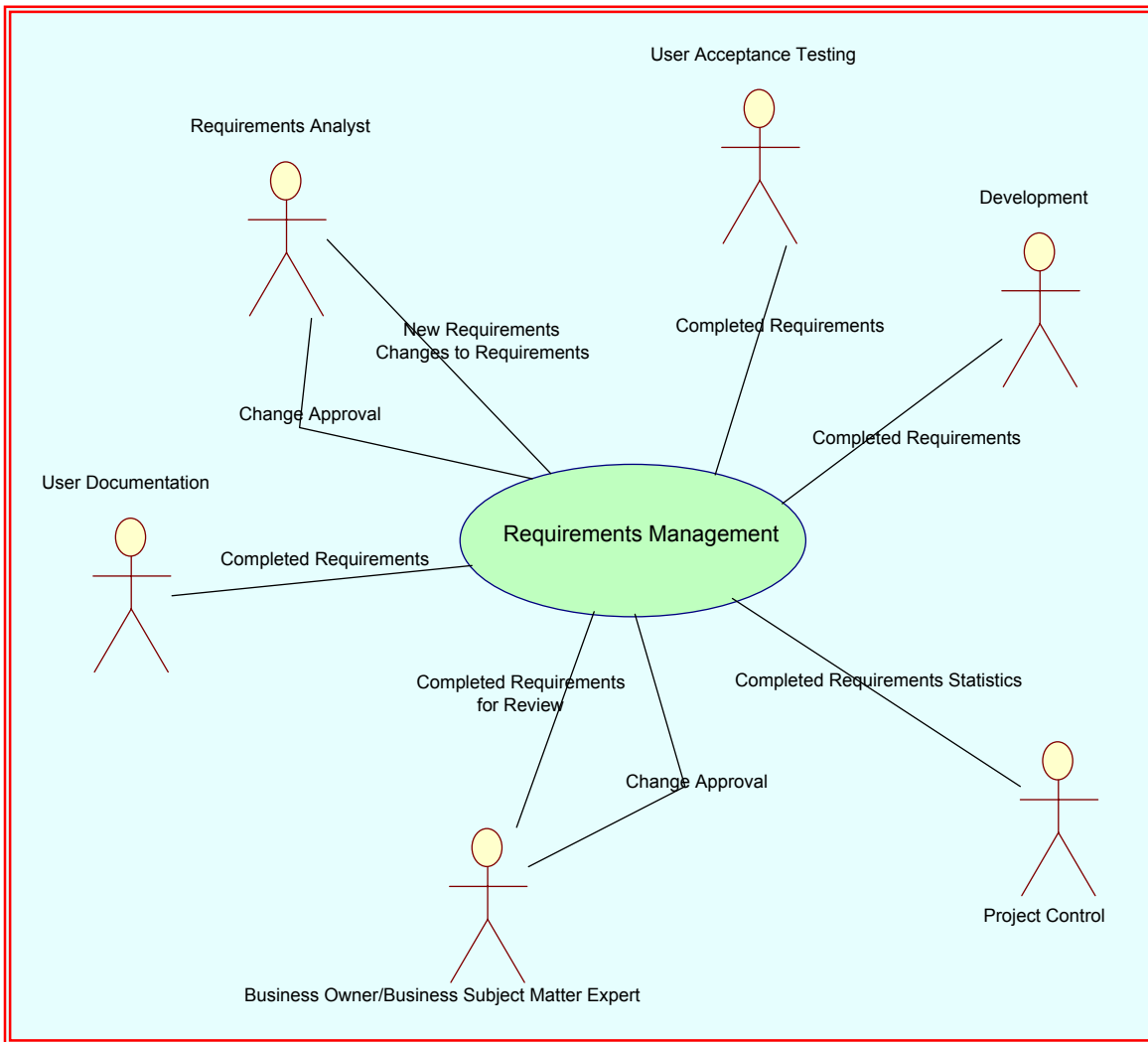


Figure 1: Users of Requirements Information

In order to support these roles, the requirements must be structured in such a way that they provide the right information. The user documentation group will find models and descriptions of the way the functions behave most useful. What kinds of users will be able to perform the functions? What kinds of data will those users need to input? What kinds of validation checks will be done, and what will happen if a check fails?

The project control office will want status reports showing all requirements and whether they're new, approved or undergoing change.

User testing will have similar needs to user documentation, in other words, models of the functional behavior. They too will need to know when a requirement has changed.



# White Papers

---

Doreen Evans Associates, Inc.

Notice that the requirements information needed seems to break down into two general categories: (1) models and text definitions; (2) status information. A good requirements process must have a way to communicate the information needed to each relevant group. And it must have a way to notify each group when something has changed.

## *A New Requirements Process*

Question: “What most often dooms IT projects?”

Answer (from a Chief Technology Officer): “First, we don’t know how to talk about requirements in any precise way, which is a matter of the business people not understanding what we need – and IT doesn’t clarify this very well, either. Specifying requirements is a failure on both the part of the business and IT departments. Users have trouble expressing their needs and desires in a way that the technologist can understand. There must be someone in the middle to interpret what the user is saying and translate it to IT.”<sup>3</sup>

Let’s return to the house analogy. When you embark on a building project it’s you, the owner, who has the vision of what you want and the money to pay for what it will cost. In all but the simplest cases, however, you turn to an architect for help. By using an architect you are getting the “someone in the middle” referred to above. Someone who has experience and can suggest alternatives for you to select from; someone who can model the solution in terms that you understand; someone who can interpret and translate that to the technicians. The architect is *your* representative. Unless you, the business owner, have lots of time and lots of experience, you will need a similar “someone in the middle” for your application development projects.

Once you have identified your “someone in the middle,” your requirements analysis can begin. The approach I recommend combines several components:

- a defined requirements gathering process for both text and visual requirements
- a requirements architect with the skills to facilitate the discovery of requirements whose job is to work with business owners
- use of a set of tools to build models and track requirements
- a template to generate documentation, both in Word and HTML
- a change control process when requirements are baselined
- a publication process to notify all stakeholders when requirements are changed
- a team of modelers, facilitators, and tool experts to participate in the project

The basis for the approach is a spiral, rapid application development (RAD) process. If the project is large, it can be separated into discrete, parallel activities for major groupings of functionality within the existing system. A *linkage* activity can then be responsible for taking the output of each of the individual channels and turning it into an integrated whole.

Each group follows a standard modeling approach by developing business *Scenarios* and system *Use Cases* for major business functions. Once this data is collected, you generate documentation,

---

<sup>3</sup> IT Roundtable, Software Magazine, April 1998.



# White Papers

---

Doreen Evans Associates, Inc.

both in Word and HTML format, and use the models to generate textual requirements statements. You can then establish requirements tracking to ensure that 1) all requirements of the existing system can be accommodated and 2) no additional requirements (otherwise known as scope creep) have been incorporated into the replacement system.

## *An Overview of the Project Approach*

Here's how it works:

1. Your business sponsors write the objectives, time estimates and cost estimates, and do the cost-benefit analysis that will precede a decision about whether to move ahead.
2. Your business users (subject matter experts and business process owners) participate in a modeling activity with analysts where their knowledge of the system is captured in visual models. The requirement analyst must be able to translate the business knowledge gathered from the business owners into structured models that accurately represent the business facts, rules and policies. Known deficiencies and recommendations for change can also be captured at this time. A set of business scenarios is defined and use cases are defined for the system.
3. As each use case or business event is modeled and understood, requirements documents are generated and text requirements are generated for a requirements management tool. These go off for review and, once accepted, are broadcast to other groups who need them (development, user documentation, and so forth) so that they can start on the work they need.
4. The accepted requirements are now in a change control process.
5. Your business users continue with the next use case or business event. If the new knowledge changes earlier work, specification documents are regenerated in Word. The current version is compared to the previous one and changes are highlighted and broadcast.
6. As earlier use cases are prototyped, your business users take the specification documents and use them to test that the prototype meets their needs. Perhaps the way the programmer has built the prototype is actually preferable to the way the business user envisioned it. If so, the models are changed and the requirements document regenerated. If not, the business user provides feedback to the developer.
7. Finally, once designers/engineers have prototyped and built the system, the scenario definitions can be used as test scenarios and all requirements can be traced to the design components in which they are implemented.

## *Conclusion*

Discovering, documenting and maintaining good requirements is the most crucial activity of the entire development life cycle. Don't fall into the 'why aren't we coding yet' trap, or allow your developers to convince you that they can build a good solution with little guidance from the business. Spending the time and the effort up front means a better solution in the long run.



Doreen Evans Associates, Inc.

**About Doreen Evans Associates**

Doreen Evans Associates (DEA) is a professional services firm that focuses on business process improvement. We can help you change a process, build an enterprise architecture, or define requirements for your systems and technologies. Founded in 1992 as a woman-owned, privately-held small business, our mission is to ensure that business need drives solutions.