

It Is Still The Requirements

Getting Software Requirements Right

By James Ward

Summary: Why are information systems requirements so difficult to define? What causes the yawning chasm between documented requirements and the actual implemented system? Requirements definition is difficult for two major reasons. First, the customer may have only the vaguest idea of what an information system should look like prior to implementation and use. Second, system developers lack sufficient knowledge of the business functions a system must support

Four Requirements Definition Assumptions

According to a recent article by Ed Yourdon in "Computerworld," software engineering literature and most requirements management methodologies often assume users understand their requirements perfectly; they just can't articulate or document them. The analyst needs only to clarify ambiguity to elicit the requirements. This prescription causes constant change, frustration, and failure to deliver systems that remotely address customer requirements.

Software development personnel who are responsible for requirements "elicitation" often start with four basic--usually erroneous--assumptions:

1. Customers can define their systems requirements.
2. The software development organization is a "customer"--not the "owner" of the process.
3. Requirements management starts after requirements have been defined.
4. The customer "owns" the requirements.

The dichotomy between those analysts who believe the customer is responsible for requirements definition and those who believe the customer is not capable of providing this information, at least to the level of detail and precision that is required, leads to widely divergent courses of action.

If you believe that customers are responsible for defining their requirements, you will primarily employ interviewing and "elicitation" techniques to obtain requirements. The definition of elicit is "to draw or bring out or forth; educe; evoke." The customer often perceives this as "extraction" or even "extrusion."

Using this method, you are at your customers' mercy--you know only what they tell you. Requirements will almost certainly be incomplete. Even if customers appear unable to define their requirements, you may believe that they are not only able to do so, but that they are responsible for doing so. You may believe that the correct elicitation tools and techniques will achieve success.

The same dichotomy exists between systems analysts who believe they are "customers" of the requirements definition process, accepting and documenting the output of that process, and those analysts who believe they must "own" the process, taking responsibility for defining the completeness, consistency, and quality of the requirements.

Too many organizations that produce and market requirements management tools self-servingly insist that requirements management begins after requirements have been identified. Requirements management begins at project inception, starting with problem identification, and continuing through a request or feasibility process. What problem are you trying to solve? What

business need are you addressing? These questions determine your requirements. Problem definition must be managed.

Requirements belong to the software developers. The developers, not the customers, must develop systems that meet those requirements. The customer must ultimately "own" the product, the deliverable system, but only if it meets their requirements.

What Are the Customer's Requirements?

Defining customer requirements is the most difficult task we face; it is also the most error prone. Over fifty percent of all errors enter the process at the requirements definition phase. How can information systems professionals meet customer requirements when no one--including the customers--knows exactly

What these requirements are?

A sincere software developer once told me, "You never know what the user's requirements are until the testing phase, so let's code something and see what happens." Fortunately, we have tools and techniques that assist in properly and completely defining customer requirements prior to the testing phase.

I use and recommend a six-step approach to make this process easier and less error prone.

Step 1: Know Your Markets

World class organizations are proficient at satisfying and delighting customers. They intensively study their customers. They develop joint-problem solving teams. They collaborate with customers in product design and development. They conduct extensive surveys. They are aware of their customers' strategic direction. And, importantly, they have excellent intelligence about their organization's competitors.

You cannot invest too much time in learning about your customers and their business. If information system organizations primarily concerned with technology are isolated from the mainstream of the business enterprises they serve, they cannot hope to meet customer requirements.

Step 2: Obtain Theoretical Business Knowledge.

In *Rethinking Systems Analysis and Design*, Jerry Weinberg talks about the time it takes a novice to amass knowledge comparable to that of an entry level graduate student in that field. The period is somewhere between one and three months. Most analysts are unwilling to invest even a fraction of this time on mastering the basics of the business discipline they are supposed to support.

You can't develop outstanding inventory management systems if you don't know inventory management. Accounts payable systems development requires accounting knowledge. Do not assume that your customers/users possess this theoretical knowledge. W. Edwards Deming said, "Experience teaches nothing unless studied with the aid of theory," but it is not unusual to encounter the absence of even basic theoretical knowledge about critical business functions in users. As an example, I worked with a client who managed finished goods inventory for an \$800 million multi-plant-packaging manufacturer. He joined the company right out of high school, worked on the loading dock, had been promoted internally to his current position. He had never taken one class in inventory management theory. How could he start to understand, let alone define, the advanced capabilities that new systems could provide?

How do you obtain this knowledge? Previously, this could require a great deal of time in the company library. Now, it's all available online. Remember, people write about problems and failures as much as successes. Take the chance to profit from someone else's mistakes.

Step 3: Study Existing Systems.

Use existing systems as a starting point in developing requirements. Analyze the existing system, but also examine other systems. These may include systems with which you have previously worked, those available from vendors, or those used by competitors. The best analysts can spot similarities between systems used for disparate business purposes and extrapolate their features to the existing environment.

For every feature offered by a vendor, ask yourself, "What requirement prompted this feature? Do we have this requirement; if not, why not?" These questions can be a tremendous help.

Step 4: Analyze the Using Environment.

The most logical and complete method of obtaining requirements is from a detailed analysis of the environment within which the system will be used. Observation, and--even better--actual working experience in this environment is invaluable if the analyst is armed with sufficient business knowledge, theoretical understanding of the business functions, and familiarity with existing systems. Without this background, you wouldn't know what to look for.

In Software Quality Professional, Alan Davis states, "Observing potential users in their natural environment. . .has resulted in significantly more accurate perceptions of the problem space than asking users what they do." Davis calls this ethno methodological studies. If systems analysts spend more than half of their time in the IT department, they aren't doing their jobs.

Leave your desk, get out, and work with your customers in their environment. If you don't understand how a system is to be used, you have no business building it.

Step 5: Interview Customers and Users.

The junior analyst begins defining requirements by asking the user what is wanted. Sadly, analysis typically stops at this point. Requirements definition interviews are not about "asking" customers what they want. These interviews should be a two-way exchange of information used to mutually define problems and opportunities for information technology application within a business function. Discuss problems with users. Clarify issues observed in the environment. Make suggestions and observations based on theoretical knowledge and familiarity with other systems. An information systems request from a customer is a request for help. The customer has a business problem and believes that information technology can assist in the solution. Usually the customer has only identified a symptom of the real problem, not the problem's root causes. Work with the customer, identifying root causes and developing opportunities to solve real business problems. Don't just satisfy a request to slap a bandage on a symptom.

Use the interview process to identify assumptions, constraints, and obstacles. Previous analytical work will have given you a basis for this. For reference, develop a high level design for a system "in a vacuum" before ever talking to the user. The interview process can then be used to tweak the design, uncover unique aspects of the problem space and discuss priorities.

Step 6: Utilize Prototyping.

Prototyping is most useful in user education and risk reduction. It is not a requirements definition technique, although it may be used to clarify expectations. Where a model does not exist for a system, where theoretical knowledge is unobtainable or where the environment within which the system must operate cannot be observed, the analyst, in conjunction with the customer, may utilize an iterative approach to requirements definition.

Prototyping can reduce the risk of building the wrong system. It can be a means of converting poorly understood requirements into well-understood requirements—educating both users and developers in the process. Prototyping can be effective in requirements definition, and let me emphasize this, only when people can articulate fuzzy (that is, poorly understood) requirements. Prototyping has the potential to clarify that fuzziness.

Know that prototypes are likely to miss the mark. Expect failures and throwaways. After all, you are sailing in uncharted waters. If the requirements were obvious, you wouldn't need a prototype. Software tools are available for building prototypes and mock ups quickly and relatively inexpensively, supposedly precluding expensive systems development fiascoes. However, I'll offer two cautionary notes regarding prototyping.

First, prototyping doesn't guarantee delivery. Recently I worked with a software firm that introduced a prototype at an industry conference. Customers went wild. This system, new and unique, would revolutionize a whole segment of their business. The software firm budgeted one year and \$1 million to turn the prototype into a working system. They booked orders against that schedule. Three years and almost \$4 million later, they still had not delivered a satisfactory system. They could develop the prototype, but lacked the system development methodologies and processes to build it.

Second, to re-emphasize, resist the trend to use prototyping as a substitute for business knowledge, analysis of existing systems, and analysis of the business environment and user interviews. Prototyping is a fallback position, not the answer in and of itself.

Own the Process

To be consistently successful, the project team must assume ownership of the requirements definition process. You cannot be the customer of the process--requirements are not a product supplied by someone else. Take responsibility for the quality and completeness of requirements. The NASA onboard shuttle project, the first software development organization to reach CMM Level 5, considers a request to change requirements as a nonconformance-- an opportunity to analyze the process, improve it, and reduce changes in the future.

The Capability Maturity Model (CMM) identifies Requirements Management as a Level 2 Key Process Area.

Are requirements identification taking place in your organization before requirement management begins? Contrast this with the iterative NASA requirements management process:

Conception

- identify need
- examine options
- develop solution

Generation

- define requirements
- produce requirements specification

Analysis

- assess technical and resource impact
- determine acceptability, implementation ability, testability
- examine requirements readiness

Inspection

- discuss proposed requirements in detail
- discuss operational scenarios
- identify issues and errors

- correct, resolve, rewrite

Approval

- evaluate risks and benefits
- decide on resource expenditures
- establish a requirements baseline

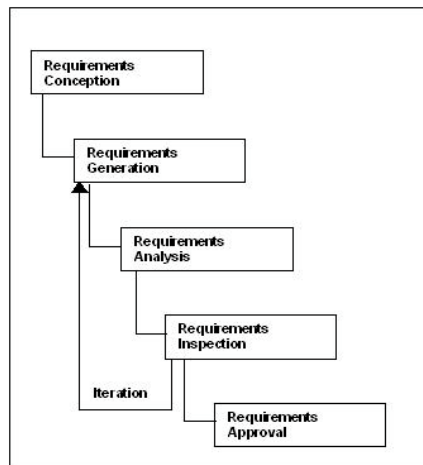


Figure 1: The Requirements Definition Process
Deliver Systems That Meet Requirements

Given the difficulty of requirements definition, expend the time and effort needed to ensure that the systems you implement do, in fact, meet the true customer requirements. Employing the six step method that I have described in this article, your efforts can yield complete, consistent requirements for systems that meet or exceed the expectations of your customers—the first time and every time.

Further Reading

Carnegie Mellon University, Software Engineering Institute (1995), *The Capability Maturity Model*, Addison Wesley, Reading, MA.

Davis, Alan M., *Achieving Quality In Software Requirements*, Software Quality Professional, June 1999, ASQ, Milwaukee.

Gause, Donald C. and Weinberg, Gerald M. (1989), *Exploring Requirements, Quality Before Design*, Dorset House, New York.

Robertson, Suzanne and Robertson, James (1999), *Mastering the Requirements Process*, Addison Wesley, Harlow, England.

Weinberg, Gerald M. (1982). *Rethinking Systems Analysis and Design*, Little, Brown and Company, Boston.

Yourdon, Ed, "Write Stuff For Users," *Computerworld*, November 20, 2000.

About the Author

James A. Ward is a management consultant specializing in project management, PMO implementation, interim IT management, implementation of quality, and process improvement initiatives in IT organizations. He offers seminars and workshops in project management, quality improvement, requirements definition, risk management, and Microsoft Project. Mr. Ward is a PMP certified project manager. He resides in Richmond, Virginia, and can be reached at (804) 303-4755 or via e-mail at soozward@earthlink.net. Further information is available on his web site at www.JamesAWard.com.