So You Want To Be a Requirements Analyst?[1]

Karl E. Wiegers
Process Impact
www.processimpact.com

Be it explicitly or not, someone always performs the role of requirements analyst on a software project. The official title may be *requirements engineer*, *business analyst*, *system analyst*, *product manager*, or simply *analyst ,* but someone needs to translate multiple perspectives into a requirements specification and communicate with other stakeholders. Perhaps most importantly, the analyst helps determine the difference between what customers *say* they want and what they really *need*—and that's easier said than done.

The Principal Conduit

The requirements analyst's primary responsibility is to gather, analyze, document and validate the needs of the project stakeholders. As the principal conduit through which requirements flow between the customer community and the software development team, you'll play a central role in collecting and disseminating *product* information, whereas the project manager takes the lead in communicating *project* information (Figure 1).

Requirements analyst is a project role, not necessarily a job title. One or more dedicated specialists can perform the role, or it may be assigned to any of a number of team members: the project manager, product manager, subject matter expert, developer or even a user. Nevertheless, a talented analyst can make the difference between project success or failure. In his book, *Software Cost Estimation with Cocomo II* (Prentice Hall PTR, 2000), Barry Boehm notes that seasoned analysts can reduce a project's required effort by one third compared to similar projects with inexperienced analysts, and projects with highly skilled analysts require half the effort of those using the least capable analysts.

The Analyst's Tasks

Straddling the gulf between vague customer ideas and the clear specifications that will guide the software team's work, the analyst must first understand the users' goals for the new system and then define functional and quality requirements that allow project managers to estimate, developers to design and build, and testers to verify the product. You can download a generic job description for a requirements analyst from http://www.processimpact.com/goodies.shtml. Here are the typical activities that you'll perform while wearing the analyst's hat.

Define Business Needs. Your work begins when you help the business sponsor or the product manager define the project's business requirements. The first question to ask is, "Why are we undertaking this project?" Business requirements include a statement of the organization's

---

[1] This paper was originally published in *Software Development*, July 2003. It is reprinted (with modifications) with permission from *Software Development* magazine.

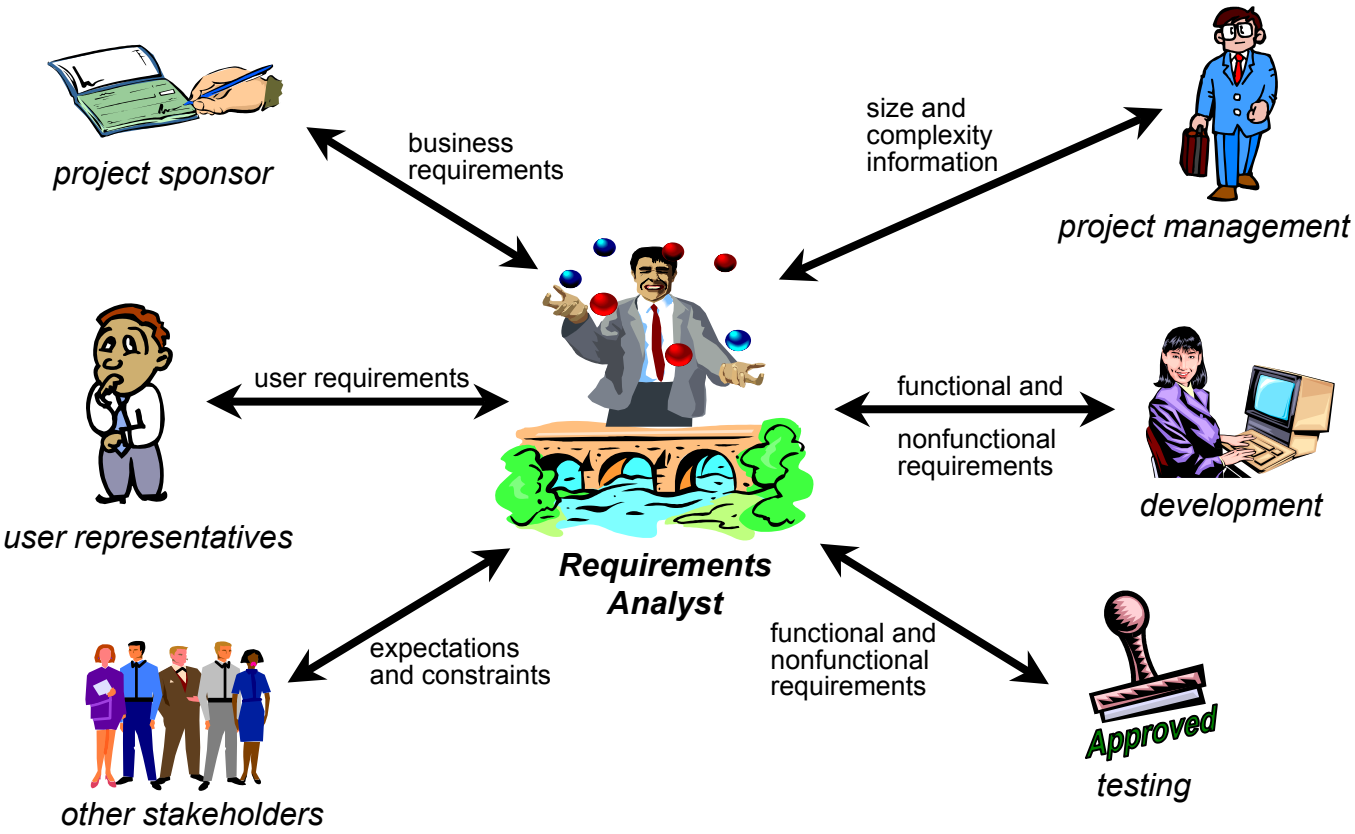

Figure 1. The requirements analyst bridges communication between customer and development stakeholders.

business objectives and the ultimate vision of what the system will be and do. Working with the people who hold this vision, you'll help them express it by completing a vision and scope document template (download a sample at www.processimpact.com/goodies.shtml).

Identify Project Stakeholders and User Classes. The vision and scope document helps you identify the product's important user classes and other stakeholders. Next, work with the business sponsors to select appropriate representatives for each user class, enlist their participation and negotiate their responsibilities. Write down the contributions that you'd like from your customer collaborators and agree on an appropriate level of participation from each one.

Elicit Requirements. Requirements for a software product don't just lie around waiting for someone to collect them. A proactive analyst helps users articulate the system capabilities they need to meet their business objectives. Users naturally emphasize the system's functional requirements, so steer discussions to include quality attributes, performance goals, business rules, external interfaces and constraints. It's appropriate to challenge assumptions, but don't try to force-feed users your own beliefs.

Analyze Requirements. Look for derived requirements that are a logical consequence of the customers' requests, as well as hunting for those implicit requirements that they expect but

haven't verbalized. Spot the vague, weak words that cause ambiguity and confusion. Point out conflicting requirements and areas that need more detail. Specify the functional requirements at a suitable level of detail for the developers who implement them. A website being built incrementally by a small, well-synchronized team can get away with limited requirements documentation, but a complex embedded system to be outsourced to an offshore supplier needs a precise, detailed software requirements specification (SRS).

Write Specifications. Effective requirements development leads to shared understanding and creation of a system that addresses the customer's problem. You're responsible for writing well-organized specifications that clearly express this shared understanding. Employing standard templates for use cases and the SRS accelerates requirements development by reminding you of topics that you need to discuss with users. You can find some sample templates at www.processimpact.com/goodies.shtml

Model the Requirements. You'll need to determine when it's helpful to represent requirements with nontextual media, including graphical analysis models, tables, mathematical equations, storyboards and prototypes. Analysis models depict information at a higher level of abstraction than does detailed text. To maximize communication and clarity, draw analysis models according to the conventions of a standard notation such as the Unified Modeling Language.

Lead Validation. You must ensure that the documented requirements satisfy customer needs and that they're clear, complete, correct, feasible, necessary, traceable, unambiguous, verifiable and so on. Analysts are the central participants in peer reviews of requirements documents. To ensure that requirements are interpreted correctly, you should also review designs, code and test cases based on the requirements specifications.

Facilitate Prioritization. You'll broker collaboration and negotiation among the various user classes and developers to ensure that the right people make sensible decisions.

Manage Requirements. After establishing the requirements baseline, your focus will shift to managing those requirements and verifying their satisfaction in the product. Storing the requirements in a commercial tool designed for this purpose can help.

You'll want to track the status of individual functional requirements as they progress from inception to verification in the integrated product. Collect traceability information from team members to connect individual requirements to other system elements. This data will aid in managing changes to the baselined requirements within a change control process and tool.

Soft Skills

An effective analyst combines strong communication, facilitation and interpersonal ability with technical and business domain knowledge and the right personality for the job. Patience and a genuine desire to work with people are key success factors. Here are 10 soft skills that you'll need to succeed.

1. *Listening*. Active listening involves eliminating distractions, maintaining an attentive posture and eye contact, and restating key points to confirm your understanding. You need to grasp what people are saying and also to read between the lines to detect what they might be hesitant to say. Learn how your collaborators prefer to communicate and try to avoid imposing your personal filter of understanding on what you hear the customers say. Watch for assumptions that underlie both what you hear from others and your own interpretation.

2. Interviewing and Questioning. Most requirements input comes through discussions, so an analyst must be able to ask the right questions. For example, users naturally focus on the system's normal, expected behaviors. However, every programmer knows how much code is needed to handle exceptions. Ask questions such as, "What should happen if...?" or "Could <some problem> ever arise?" so you can determine how the system should handle anticipated exception conditions. With experience, you'll become skilled in the art of asking questions that reveal and clarify uncertainties, disagreements, assumptions and unstated expectations. Donald Gause and Gerald Weinberg describe "context-free questions" in *Exploring Requirements: Quality Before Design* (Dorset House, 1989).

3. Analytical. You'll need to be able to operate at various levels of abstraction. Sometimes you must drill down from high-level information into details. In other situations, you'll need to generalize from a specific need that one user described to a set of requirements that pertain to a whole class of users. Critically evaluate the information gathered from multiple sources to reconcile conflicts, separate user *wants* from *needs*, and distinguish solution ideas from requirements.

4. Facilitation. Requirements elicitation workshops are a common technique; to succeed, they require a neutral facilitator. You'll need strong questioning and observational skills to help groups build trust and to improve the sometimes tense relationship between business and information technology staff. In *Requirements by Collaboration: Workshops for Defining Needs* (Addison-Wesley, 2002), Ellen Gottesdiener provides a wealth of advice for the workshop facilitator.

5. Observation. If you conscientiously watch a user perform his job or put a current application through its paces, you can detect subtleties that the user might not mention and thus expose new areas for requirements discussion.

6. Writing. Your main deliverable will be a written specification for customers, marketing, managers and technical staff; for this task, you need a solid command of the English (or other natural) language. Strive for clarity; avoid ambiguous words and phrasing, grammatical errors and overly idiomatic expressions.

7. Organization. You'll be faced with a vast array of jumbled information gathered during elicitation and analysis. Structuring all the rapidly changing bits into a coherent whole demands exceptional organizational skills, along with patience and tenacity.

8. Modeling. From the venerable flowchart through structured analysis models (data flow diagrams, entity-relationship diagrams and the like) to contemporary UML notations, diagrammatic tools should be included in every analyst's kit. Some of these techniques will be useful when communicating with users; others, with developers. You'll often need to educate other stakeholders on the value of these techniques and how to read them—and you'll find that these tools are useful ways to represent information even when you're not discussing a computer system.

9. Interpersonal. You must be able to get people with competing interests to work together, and you should feel comfortable talking with individuals in diverse job functions and at all levels of the organization. You might also need to work with distributed teams whose members are separated by geography, time zones, cultures or native languages.

10. Creativity. The analyst is not merely a scribe who records whatever customers say. In his article, "Eureka! Why Analysts Should Invent Requirements" (*IEEE Software*, July/Aug. 2002), requirements authority James Robertson suggests that the best analysts propose requirements. Analysts might conceive innovative product capabilities, imagine new markets and business

opportunities and think of ways to surprise and delight their customers. A really valuable analyst finds creative ways to satisfy needs that users didn't even know they had.

Domain, Business and Task Knowledge

Enthusiasm alone won't take you very far in requirements gathering; breadth of knowledge gained through experience is the only way to hone your ability. Start with a solid understanding of contemporary requirements-engineering techniques and how they fit in various software development lifecycles. Analysts need to know how to thread requirements development and management activities through the entire product lifespan. A sound understanding of project management, risk management and quality engineering can help prevent requirements issues from torpedoing the project. In a commercial setting, you'll benefit from knowledge of product management concepts and the way enterprise software products are positioned and developed.

Application domain knowledge is a powerful asset for an effective analyst, minimizing miscommunications with users. Analysts who understand the application domain often detect unstated assumptions and implicit requirements. They can also suggest ways that users could improve their business processes. Such analysts sometimes propose valuable functionality that no user thought of. Conversely, they do a better job of detecting gold-plating—that is, excessive or unnecessary functionality—than does someone who's unfamiliar with the problem domain.

Grown, Not Trained

There's no standard educational curriculum or job description for a requirements analyst, which is why most come from diverse backgrounds. If you migrate into analysis from a career as a system user, you'll need to balance your extensive knowledge of the business and work environment with supplemental training in software engineering and how to communicate with your technical counterparts. You'll also have to work against any tendency to believe your user experience is all-encompassing, to focus excessively on the user interface, or to ignore input from real users of the new product. On the other hand, if you're a former developer**,** you'll need to beef up your understanding of the business domain and guard against lapses into technical thinking and jargon.

Whatever your background, training and mentoring in the soft skills that the best analysts master, such as effective listening, negotiating and facilitating, helps new analysts of all stripes uncover users' unspoken desires.

Sidebar: Gathering Data

Try the following techniques to get the goods on requirements.

- Interviews
- Facilitated requirements workshops
- Document analysis
- Surveys
- Customer site visits
- Business process analysis
- Work flow and task analysis

- List of external events and corresponding system responses
- Competitive product analysis
- Reverse-engineering of existing systems
- Retrospectives performed on previous projects