

The Roles of the RA

Chapter 1 emphasized the importance of requirements. It was noted that customers, managers, and developers undervalue requirements engineering. The RA is in a strategic position to improve the practices in use on projects and in the organization. The analyst can have a positive impact on project success and also facilitate the organization's improvement results by performing in several roles. Making the RA's role explicit contributes to a smoother process. The RA's role can be linked readily to business goals, such as increasing customer satisfaction with the delivered work products; reducing the time to market of products; meeting cost, schedule, and quality objectives; and utilizing the human resources of the enterprise more effectively. The RA's role needs to be understood and valued in the minds of PMs and the technical communities (both computing and engineering). Table 2.1 summarizes the roles of the RA, noting the life-cycle phases in which each role is performed.

Suggested Roles of the RA

1. *Work collaboratively with customers, users, and system architects and designers to identify the real requirements for a planned system or software development effort to define the problem that needs to be solved.*

The concept of the real requirements was explained in Chapter 1. Experience has shown that the number one problem in requirements engineering is the failure to identify the real requirements prior to initiating system development activities. Anyone who has had some experience in developing systems or software will agree that identifying the real requirements is a significant problem. With respect to this role, the RA needs to create an awareness of the problem and also provide a suggested strategy to overcome the problem. This is a concrete example of a

Table 2.1
RA Roles and Life-Cycle Activities
Source: Richard Raphael.

situation that we know can be improved, but most often we don't act on this knowledge. We are impatient to get started on the so-called real work of programming. We are content to allow the development effort to proceed without taking the extra effort to evolve the real requirements. Note that I have used the word "evolve." This work involves more than identifying requirements. The essential task is to use the stated requirements articulated by customers and users as a base, couple this with a thorough understand-

RA Role	System Initiation	System Analysis and Design	System Component Design	System Implementation	System Integration, Test, and Evaluation	System Operations and Support
1. Work collaboratively with customers, users, and system architects and designers to identify the real requirements for a planned system or software development effort to define the problem that needs to be solved.	X	X	A	A	A	A
2. Work effectively with customers and users to manage new and changed requirements so that the project stays under control. Install a mechanism to control changes.			X	X	X	X
3. Be alert to new technologies that may help.		X	X			X
4. Facilitate the project in reusing artifacts and achieving repeatability.	X	X	X	X		
5. Assist the project and its customers in envisioning a growth path from the first release or version of a product through a set of staged releases to the “ultimate system or products.”	X	X	X	X	X	X
6. Advise the project (and customer) of methods, techniques, and automated tools that are available to best support requirements-related project work and activities.	X	X				X
7. Use metrics to measure, track, and control requirements-related project work activities and results.	B	X	X	X	X	X
8. Be able to facilitate discussions and to mediate conflicts.	X	X	X	X	X	X
9. Study the domain of the area in which the system or software is being used.	X	X				X
The life-cycle phases shown in the top row are not intended as a recommendation for a waterfall life-cycle model. Rather, the use of the phase terminology is intended to depict the activity being performed; this activity is equally applicable to virtually all life-cycle models (e.g., spiral, rapid application development, incremental, rational unified process), although the life-cycle phase names will differ.						
A—Continue to identify real requirements for subsequent releases and revisions, maintaining configuration control.						
B—System initiation or “project or task startup” is a confusing time. The experienced RA will be able to lend assistance. For example, the RA should provide a briefing to the project team that includes the topics noted in Table 5.4.						

Table 2.1
RA Roles and Life-Cycle Activities
Source: Richard Raphael.

ing of the business objectives, and iterate to evolve requirements that meet the criteria for a good requirement and address prioritized real needs for the system or software. Activities involved in performing this work include the following:

Identifying the stated needs of customers and users. This involves

reviewing things previously written about the proposed system, interviewing customers and users, studying relevant legislation, and so forth.

Studying the business objectives for the proposed effort.

Collaborating with customers and users in a joint or cooperative environment to analyze the stated requirements, evolve better requirements, and prioritize them (see the suggested techniques that follow).

Involving system architects in requirements development. Iterating the draft or proposed requirements will result in a candidate architecture with better requirements and a more robust architecture. For example, systems need to be able to accommodate changing business needs. The architecture should be designed and developed accordingly, or else the delivered system soon will be outdated.

Utilizing an industry-strength automated requirements tool to support this work.

The RA should work within the project organization to win the support of the PM in gaining commitment to investing added time and effort to evolve the real requirements. Here is a great opportunity for the RA to take responsibility and, drawing upon industry experience, convince project management and developers to invest more time and effort in the requirements process. Fortunately, data is available to help us manage by fact rather than by intuition or the way we have always done things. Refer to *Effective Requirements Practices* [1, p. 62] for these data.

Consider using collaborative requirements elicitation techniques that work well in group sessions. Examples of good requirements elicitation techniques are requirements workshops, electronic-based groupware or electronic collaborative development tools, high-level data flow diagrams, high-level IDEF0 diagrams (especially for business modeling), and high-level use case diagrams (especially to distinguish requirements that are

outside the system versus behavior expected from the system). All of these work well on a whiteboard, are easy to understand, and allow everyone present to participate. See Dean Leffingwell and Don Widrig's *Managing Software Requirements: A Unified Approach* [2] for good discussions of these and other techniques and how to use them. David Hay provides a useful comparison of techniques that can be used in *Requirements Analysis: From Business Views to Architecture* (see [3, p. 194] and the preceding discussion).

2. *Work effectively with customers and users to manage new and changed requirements so that the project stays under control. Install a mechanism to control changes.*

The next most serious problem in requirements engineering (after the failure to identify the real requirements) is failure to control requirements that are identified after system development (programming) begins, both new requirements and changes to existing requirements. Here we distinguish

between critical requirements (those that would have an impact on cost, schedule, or the development effort if changed) and noncritical requirements, such as a derived requirement that further defines the system being built, but serves to clarify a higher-level requirement and does not affect cost, schedule, or functionality. All stakeholders should welcome a “no-impact” requirement that further clarifies the system.

Again, we have data from industry experience to guide our actions: a 20% change in requirements will result in a doubling of project-development costs [4]. Therefore, it’s critical that a mechanism be put in place to evaluate and adjudicate changes to requirements. Without an effective mechanism to control changes to requirements, the project will soon be out of control in terms of schedule and cost. Several things must be done:

The importance of controlling changes to requirements must be explained to customers, users, and developers so that the partnership commitment to project success is maintained.

Developers must be trained not to accept unauthorized requirements changes. All requests for changes, no matter how trivial, must be funneled through the change control mechanism.

The change control mechanism should be a joint team that includes empowered decision makers representing the customer and the developer. The joint team should meet frequently enough to have a reasonable number of change requests to consider. A target metric of 0.5% requirements volatility is recommended to guide decisions made by the joint team once a baseline of validated requirements has been established.¹ “Whoa,” you say, “that’s not much!” Right! This is another

1. Chapter 10 of *Effective Requirements Practices* provides several ideas, suggestions, and recommendations for controlling requirements changes.

reason to invest the needed time to evolve the real requirements prior to starting the development activities.

Partnering with your customer, evolve ways to deal with change. We know the world is changing while we’re developing the system. What are some ways to deal with this without jeopardizing project success? Consider using releases, versions, and upgrades. Package increments of requirements upgrades and changes in subsequent releases or system upgrades.

Ensure that your contract provides for additional time and budgeting for all changes. This is a mechanism to maintain good relationships throughout the contract work—to partner for success. Changes cost time and money. This should be recognized up front and reflected in the contract.

3. *Be alert to new technologies that may help.*

A role that is often underutilized is advising our customers concerning evolving technology. While this is not solely the responsibility of the requirements analysts or engineers, many involved in developing systems for customers would be well advised to spend additional time and effort learning about new technologies and how they can be applied to our work. Customers are typically focused on what the system needs to do. We can serve them best by being familiar with evolving technologies that improve *how* the needed system is designed. This suggests that RAs will benefit from having system designers review their work products. Concurrently with requirements elaboration, involve a small team of designers to review the real requirements for cost, schedule, technology, and risk impacts. Use trade studies—the Decision Analysis Resolution (DAR) process in CMMI[®] terminology—to evolve alternatives. Keep the customer involved in these activities, so that when opportunities arise, the customer is there to partner with you in making recommendations for decisions. An excellent reference that describes the process of utilizing new technologies is Everett M. Rogers’s *Diffusion of Innovations* (4th ed.) [5].

4. *Facilitate the project’s reuse of artifacts and achieving repeatability.*

There has been a lot of discussion in the industry literature about reuse. Reuse has two meanings: (1) to take object X (e.g., an object, subroutine, or COTS software) that was done by Y and use it directly in another project, and (2) to tailor² a developed work product (a specification, a plan, or process, for example). Many organizations have invested in reuse strategies only to conclude that they are not viable or practical. Others are wary of

2. By “tailoring,” we mean modifying, extracting pieces from, elaborating, or adapting a process or document for another use. Reuse of tailored artifacts saves time and money and is an advantage of a process-oriented approach.

reuse because they believe it precludes unprecedented solutions and incorporates the errors of the reused work products.

We can consider requirements themselves as reusable artifacts. Books that discuss reusable requirement patterns include *Data Model Patterns: Conventions of Thought* [6] (for a relational viewpoint) by David C. Hay, *Analysis Patterns* (for an object oriented viewpoint) by Martin Fowler [7], and *Design Patterns* by Eric Gamma, et al. [8]. Michael Jackson’s problem frames (described in his book by the same name [9]) are in essence highly abstract requirements patterns that can be connected, nested, and built into real world models. The point is that many requirements are not unique; they have already been identified in someone else’s environment and problem space.

I have found in my writing activities that starting with an example work product gives me ideas about format, structure, content, and resources to reference or contact. An example work product you might want to consider is a requirements plan. As emphasized in Chapter 1, I advocate development of a requirements plan for any system or software development effort. This idea may be new to you, and it would be very helpful and instructive to review one developed previously in order to consider its potential value to your

work. Another example from my experience is reusing documented processes. If the organization or another project has a documented process for doing something, why not tailor it as needed and then reuse it, rather than create one's own process? Others who have performed the process in practice have incorporated their experience and the lessons they have learned using it. Related to this is the value of peer reviews. I advocate a peer review of every work product. (The extent of the peer review—the number of people requested to review the work product and the time invested to perform the peer review and report on defects and make suggestions—is a function of the importance of the work product.) If one can reuse the peer review process and checklists of another organization, this provides a jump-start in getting the process designed, accepted, deployed, implemented, and institutionalized.

An Example of Process Reuse

In teaching requirements courses and tutorials, I'm always interested to learn how many of the participants are using a documented requirements process on their project or in their organization. Typically, this turns out to be 15% to 20% of the participants. A sample requirements process is provided in *Effective Requirements Practices* [1, pp. 110–118]. This process has been tailored, deployed, and implemented on more than 50 projects. Its integration with the system architecture process is described later in the book [1, pp. 136–146].

Suggestion: Tailor this sample requirements process for your project or organization. Involve the stakeholders to make the changes that best serve their needs. Provide both flowcharts and narrative PDs as described in *Effective Requirements Practices*. Periodically update the documented process with continuous improvement ideas and suggestions.

5. *Assist the project and its customers in envisioning a growth path from the first release or version of a product through a set of staged releases to the ultimate system or product.*

This role is related to role 3. The RA can serve an important and valuable role in helping customers to envision and evolve a series of releases or versions of products. This approach is particularly appropriate in the situation in which requirements are not well understood at the outset or the requirements are changing rapidly. This suggests that an “incremental development approach” should be used, in which the full system is implemented over a period of time through increments of delivered functionality. In a sense, no system is ever done, so we have to help everyone see system development as a journey. Independently of the system development methodology used (waterfall, incremental, spiral, evolutionary, etc.), there has to be an agreed-upon process for managing changes and determining the scope of individual projects. No matter how much discussion and testing is done, there are some missing requirements that won't be discovered until the system is in production.

6. *Advise the project (and customer) of methods, techniques, and automated tools*

that are available to best support requirements-related project work and activities.

This is an important role. Experience has shown that methods and techniques vary in their applicability and effectiveness and that often automated tools purchased by projects and organizations are not used or are underutilized. Chapter 11 of *Effective Requirements Practices* [1] reports on industry experience and provides several recommendations. Chapter 8 of *Effective Requirements Practices* [1] recommends that the methods and techniques that are used by a project be familiar to the project participants and proven in their respective industry. It's not advisable to undertake a project with unproven, unfamiliar methods and techniques. The development work is challenging enough without introducing the complexity of methods or techniques that are not familiar and haven't been used successfully on previous projects in the organization. At the project level, the team should stick with the tools, processes, and techniques with which its members are familiar. At the organizational level, the project should try to use the tools, processes, and techniques that are known and proven in the organization. When contractors are brought into an existing effort, they should adapt to the tools that the customer already has in place (assuming they are working effectively). If the last five projects were done with tool X, and everyone is satisfied with the usefulness of the tool, then when you arrive, there are good reasons to use it. Note that a resource issue may be involved. Ideally, an RA would be a leveraged resource, moving from project to project and taking her experience with her. However, often in practice, a project team is built (or already exists) and someone from the current team with domain knowledge is tasked with being the RA. While tried-and-true techniques and tools exist, they may be unfamiliar to this person, requiring a lengthy and sometimes painful learning curve, with significant disadvantages to the

project. This argues for the organization to provide a set of experienced RAs that will provide a high return on the investment made to identify them, train them, and provide them with experience.

I also recommend challenging customer directions to use specific methods or techniques that are not familiar to the project team or not previously proven in practice. For example, a customer might direct that an object-oriented (OO) development approach be employed (see [10] for thoughtful guidelines on this topic) or that a particular automated tool or tool suite be used. It's valuable to be in a position to be able to advise your project and your customer of the methods, techniques, and automated tools that will best support the specific development situation. Draw on industry experience and don't pretend that "everything will work out."

7. *Use metrics to measure, track, and control requirements-related project work activities and results.*

The industry literature concerning metrics is vast. I'd estimate that perhaps 20% of it provides helpful counsel. It's easy to get into a situation of performing measurement activities for their own sake, rather than to help evaluate project work and take corrective actions. I recommend using a few useful metrics. I have developed the following axiom in my work over the

years:

The things that are measured and tracked and that management pays attention to are the ones that improve.

This suggests that it's not sufficient to have a few useful metrics—they must be tracked, and they must be used by management to guide project decisions.

There is a set of measures or metrics that should be used by all projects. See *Effective Requirements Practices* [1, pp. 255–261] for specific suggestions.

There is another level of sophistication that should be used by mature projects and organizations. As used here, “mature” means that processes have been defined, documented, implemented, used, institutionalized, and continuously improved over a period of at least two to four years. This involves quantitative management (QM) of cost, schedule, quality, and process metrics and baselines in support of specific business objectives. It is fulfilling to see projects and organizations move from the situation in which QM is not well understood to one in which QM is effectively used to achieve business objectives. This is especially satisfying to process engineers, because executives can see first hand the value of process improvement in meeting business needs.

8. *Be able to facilitate discussions and to mediate conflicts.*

This role stresses the “people skills” of the RA. We've learned that being well qualified technically is important, but that it's also necessary to have strong,

Summary

23

well-refined people skills. Experience has shown that two heads are better than one—whenever we take the time to explore ideas and approaches with others, we get even better ideas and approaches! Ergo, we can drop the point of view that “we know best.” And we can make great use of this principle by becoming good facilitators and mediators. There are courses available to assist (e.g., negotiating skills, team building, communications, relationships, and leading). Much can be gained by practicing these skills in our daily work. Having a “win-win” perspective is helpful—in fact, Barry Boehm et al. have developed a win-win requirements development approach in work done at the University of Southern California. See http://sunset.usc.edu/research/WINWIN/winwin_main.html.

9. *Study the domain of the area in which the system or software is being used.*

Be able to grasp, abstract, and express ideas quickly in the users' language. If the RA does not understand the user domain almost as well as the users do, he risks limiting his role to that of an order taker. I have seen different groups come and go whose specialty was communication, consensus building, and so forth. Populating those groups was a set of people who were trained facilitators, but who were not technically proficient. They moved from project to project so frequently that they never achieved any deep domain

understanding. For example, what if, on a network communications project, the only way an RA can explain any concept to the users is by giving analogies with building military aircraft? Answer: reduced effectiveness and credibility.

Summary

The RA performs several important roles on a project and in an organization. Nine important roles were identified and described in this chapter. The first two are paramount and essential to project success. Accordingly, study these, become proficient in them, and assist your project and organization in adopting, implementing, and institutionalizing related practices. Organizations should consider taking specific steps to develop and leverage their RAs, such as (1) ensuring that experienced RAs are assigned to each project; (2) providing appropriate training for RAs; (3) assigning experienced RAs to mentor new employees, junior RAs, and interns; and (4) having an organizational requirements working group to share expertise and provide a resource to the organization. The RA should be a trained, experienced, and strong performer. Unfortunately, I've seen many cases where the new employee or the summer intern is dispatched "to get the requirements." The role of the RA needs to be understood and valued in the minds of PMs and the technical community. At this point, you may feel overwhelmed with your responsibilities, as Figure 2.1 suggests. Be assured that with study and experience, you will provide a very positive contribution to the efforts that you support!



Figure 2.1 The challenges of the RA.

Case Study

This is the story of a project that failed because neither the customer nor the

contractor knew how to handle requirements. It is a negative example. Although the people involved were professionals and were well intentioned, things went horribly wrong because effective requirements practices were not applied.

The project approach was one that is used with alarmingly frequency and can be characterized as “get started programming and we will find out what they want as we proceed.” The customer, a military organization, handed the contractor a shelf-load of rules and regulations (“regs”) saying, “These are the requirements.” The programmers, all on-site employees of a contractor, were ready to start, and they did. Representatives of the military organization were aware of the project, but they did not participate in it until it was time to review the finished code.

While the code was being written, the contractor undertook to convert the regs to a set of shall statements. This was faithfully and painstakingly done, but as the code emerged, it was found that the verification of the shall statements—matching them to parts of the different code modules—was virtually impossible. They simply did not map.

Another complication encountered was a complete breakdown in communication between the contractor and the subcontractor producing the code. It wasn't that they did not understand each other; they simply did not communicate. The subcontractor viewed any inquiries as interference, and in an atmosphere of hostility, communication simply died.

As the code came into review by the customer, the military representatives were heard to say again and again, “No, that's what we do, but that's

Case Study

25

not how we do it.” And as the modules came into test by the contractor, they failed repeatedly. The right things had been written the wrong way, and they did not work.

After months of struggle, the first of about 20 modules was almost ready for release. It was still a little shaky, and lots of people were unhappy with it, but it was close to acceptance from the contractor's perspective. However, the customer gave up because relationships between the two parties had become broken during the code development period. Not only was the process broken, but also the platform and the operating system were outdated and inadequate. Three million lines of code were abandoned, the hardware was scrapped, and the whole project was started all over again.

What had gone wrong?

There was no partnership between the customer, contractor, and subcontractor.

There was no communication.

There was not an atmosphere of mutual respect.

There was no workable requirements plan.

There was no mechanism for joint resolution of problems.

The project was begun anew, on a different platform, in a different lan-

guage, by a different mix of subcontractors. An improved set of requirements practices was used, which included the following:

Mechanisms (similar to the joint team discussed in Chapter 1) to facilitate partnering, identifying real needs and requirements, and prioritization of requirements;

An approach that involved the users in the development effort, provided for collaboration with them, and gained the buy-in of the users to the project approach;

Use of methods including use cases that facilitated understanding and effective communication of user needs and requirements;

Incorporating appropriate and updated technology that better served the customer and the users.