CHAPTER 30

Tools for requirements engineering

Estelle finally got her SRS document completed and approved. Now James wants to add a requirement, but it messes up the numbering scheme, incrementing the labels for requirements that follow it in that section of the document. Estelle hopes that changing the requirement identifiers won't cause problems for anyone already working from those requirements. Sean requests to delete a requirement. Estelle suspects that the requirement might come back into scope in the future, so she wonders where to put it and how to keep the developers from working on it now. Antonio asked Estelle yesterday why a specific requirement was included, but she didn't have any way to answer that question.

One of the developers, Rahm, asked for a list of all the requirements that he was responsible for on the next release, but Estelle doesn't have any easy way to generate such a list. In fact, it's not easy to keep track of which requirements are scheduled for which release, because they are all stored in the same document. Estelle would like to know the status of requirements that are already under development, but she doesn't have an easy way to find that information either.

Estelle's document-based requirements approach is falling short of her requirements management needs. She needs a tool.

In earlier chapters, we discussed the creation of a natural-language software requirements specification to contain the functional and nonfunctional requirements, as well as documents that contain the business requirements and user requirements. We pointed out that these deliverables are just containers for sets of requirements information; they need not be traditional word-processing documents. Although still widely used, a document-based approach to developing and managing requirements has numerous limitations, including the following:

- It's difficult to keep the documents current and synchronized.
- Communicating changes to all affected team members is a manual process.
- It's not easy to store supplementary information—attributes—about each requirement.
- It's hard to define links between requirements and other system elements.
- Tracking the status of both individual requirements and the entire set of requirements is cumbersome.

- Concurrently managing sets of requirements that are planned for different releases or for related products is tricky. When a requirement is deferred from one release to a later one, a BA needs to manually move it from one requirements specification to another.
- Reusing a requirement generally means that the business analyst must copy the text from the original document into another document for each other system or product where the requirement is to be used.
- It's difficult for multiple project participants to modify the requirements, particularly if the participants are geographically separated.
- There's no convenient place to store proposed requirements that were considered but rejected and requirements that were deleted from a baseline.
- It's hard to create, trace, and track edits to analysis models in the same location as requirements.
- Identifying missing, duplicate, and unnecessary requirements is difficult.

Requirements development (RD) tools and requirements management (RM) tools provide solutions to all of these limitations. RD tools can help you elicit the right requirements for your project and judge whether those requirements are well-written. RM tools help you manage changes to those requirements, track status, and trace requirements to other project deliverables.

A team working on a small project might be able to get away without using any requirements tools, instead using documents, spreadsheets, or simple databases to manage their requirements. Teams working on large projects will benefit from commercial requirements engineering tools. None of these tools replaces a defined process that your team members follow to develop and manage their requirements. Use a tool when you already have an approach that works but that requires greater efficiency. Don't expect a tool to compensate for a lack of business analysis and requirements engineering process, training, discipline, or experience.

> **Trap** Avoid the temptation to develop your own requirements tools or to cobble together general-purpose automation products in an attempt to mimic the commercial requirements products. This initially looks like an easy solution, but it can quickly overwhelm a team that doesn't have the resources to build the tools it really needs.

This chapter presents several benefits of using requirements tools and identifies some general capabilities you can expect to find in such products. Dozens of commercial requirements tools are available. This chapter doesn't contain a feature-by-feature tool comparison, because the products are constantly evolving and their capabilities (and sometimes their vendors) change with each release. RD and RM tools often aren't cheap, but the high cost of requirements-related problems can justify your investment in them. Recognize that the cost of a tool is not simply what you pay for the initial license. The cost also includes annual maintenance fees and periodic upgrades, software installation and configuration, administration, vendor support and consulting, and training for users. Cloud-based

solutions eliminate some of these additional support activities and costs. Your cost-benefit analysis should take into account all of the expenses before you make a purchase decision.

Requirements development tools

Requirements development (RD) tools are used by business analysts to work with stakeholders to elicit and document requirements more effectively and more efficiently than with manual methods. Stakeholders will vary in how they best consume and share information: textually, visually, or audibly. RD tools can improve stakeholder collaboration by accommodating a variety of communication methods (Frye 2009). This section subdivides the development tools into elicitation, prototyping, and modeling tools. Some of the tools in the RD category provide all of these services. Some of them also offer requirements management capabilities. In general, RD tools are not as mature as RM tools, and their overall impact on projects is typically less than that of RM tools.

Elicitation tools

Elicitation tools include those used for recording notes during elicitation sessions. These enable the BA to quickly organize ideas and to annotate follow-up questions, action items, core terms, and the like. Mind-mapping tools facilitate brainstorming as well as organizing the information produced. Audio pens and other recording tools allow playback of conversations or provide visual reminders of what happened during an elicitation session. Some recording devices also tie the audio directly to the text that was written at the same time, enabling you to hear specific portions of the audio conversation as needed. Tools that support quality checks, such as scanning a requirements document for vague and ambiguous words, help a BA write clearer requirements. Some elicitation tools convert requirements from text to auto-generated diagrams. Certain tools also enable collaborative voting to help a team prioritize requirements.

Prototyping tools

Prototyping tools facilitate the creation of work products that range from electronic mock-ups to full application simulations. Simple prototyping tools come with basic shapes and designs to create low-fidelity wireframes (Garmahis 2009). Common applications such as Microsoft PowerPoint can be used to quickly mock up screens and the navigations between them or to annotate existing screen shots. Sophisticated tools might enable mocked-up functionality that a user can click through to see just how the application would work. Some prototyping tools support version control, feedback management, requirements linking, and code generation. See the cautions in Chapter 15, "Risk reduction through prototyping," to avoid investing more effort in creating prototypes than is needed to achieve your goals. If you use a tool to create high-fidelity prototypes, make it clear to customers that the prototypes are just possible models and that the final product might be different. Some prototyping tools can show screen mock-ups in a "hand-drawn" style to help manage customer expectations.

Modeling tools

Requirements modeling tools help the BA create diagrams like those described in Chapter 5, "Establishing the business requirements," Chapter 12, "A picture is worth 1024 words," and Chapter 13, "Specifying data requirements." These tools support the use of standard shapes, notations, and syntax for drawing diagrams according to established conventions. They might provide templates as starting points and examples to help the BA learn more about each model. Often these tools automatically connect shapes in diagrams to accelerate the drawing process and to help ensure that the diagrams are drawn correctly. They also enable you to create diagrams that look cleaner and more consistent than if you draw them manually. Specialized software modeling tools facilitate iteration by dragging along connected arrows and labels whenever you move a symbol in the diagram; general-purpose drawing tools might not provide that capability.

Many requirements management tools also provide some modeling capability. The most sophisticated tools allow you to trace individual requirements to models or even to specific elements of models. For example, analysts can create swimlane diagrams in the tool, and then after they write requirements, they can trace those requirements back to specific steps in the diagrams.

Keep in mind that no tool will be able to tell you if a requirement or a model element is missing, logically incorrect, or unnecessary. These tools enable BAs to represent information in multiple ways and to spot certain types of errors and omissions, but they don't eliminate the need for thinking and peer review.

Requirements management tools

An RM tool that stores information in a multiuser database provides a robust solution to the limitations of storing requirements in documents. Small project teams can get away with just entering the requirements text and several attributes of each requirement. Larger project teams will benefit from letting users import requirements from source documents, define attribute values, filter and display the database contents, export requirements in various formats, define traceability links, and connect requirements to items stored in other software development tools.

Requirements management tools have been available for many years. They are both more plentiful and more mature than requirements development tools. To be fair, the problem they solve is more tractable. It's easier to create a database in which to store requirements and provide some capabilities to manipulate them than to help a BA discover new knowledge, craft that knowledge into precise requirement statements and diagrams, and ensure that the resulting information representations are correct. Some tools combine both RD and RM capabilities into a powerful solution aid.

Benefits of using an RM tool

Even if you do a magnificent job of eliciting and specifying your project's requirements, you can lose control of them as development progresses. An RM tool becomes most valuable as time passes and the team members' memories of the requirements details fade. The following sections describe some of the tasks such a tool can help you perform.

Manage versions and changes Your project should define one or more requirements baselines, each identifying a specific collection of requirements allocated to a particular release or iteration. Some RM tools provide baselining functions. The tools also maintain a history of the changes made to each requirement. You can record the rationale behind each change decision and revert to a previous version of a requirement if necessary. Some tools contain a change-proposal system that links change requests directly to the affected requirements.

Store requirements attributes You should record several descriptive attributes for each requirement, as discussed in Chapter 27, "Requirements management practices." Everyone working on the project must be able to view the attributes, and selected individuals will be permitted to update attribute values. RM tools generate several system-defined attributes, such as the date a requirement was created and its current version number, and they let you define additional attributes of various data types. Thoughtful definition of attributes allows stakeholders to select subsets of the requirements based on specific combinations of attribute values. A Release Number attribute is one way to keep track of the requirements allocated to various releases.

Facilitate impact analysis RM tools enable requirements tracing by letting you define links between different types of requirements, between requirements in different subsystems, and between individual requirements and related system components (for example, designs, code modules, tests, and user documentation). These links help you analyze the impact that a proposed change to a specific requirement will have on other system elements. It's also a good idea to trace each functional requirement back to its origin or parent so that you know where it came from. For instance, you might ask to see a list of all the requirements originating from a specific business rule so that you can judge the consequences of a change in that rule. Chapter 28, "Change happens," describes impact analysis, and Chapter 29, "Links in the requirements chain," addresses requirements tracing.

Identify missing and extraneous requirements The tracing functionality in RM tools helps stakeholders identify requirements that are missing, such as user requirements that have no mapped functional requirements. Similarly, they can reveal requirements that cannot be traced back to a reasonable origin, raising the question of whether those requirements are necessary. If a business requirement is cut from scope, then all the requirements that trace from it can also be cut quickly.

Track requirements status Collecting requirements in a database lets you know how many discrete requirements you've specified for the product. As Chapter 27 described, tracking the status of each requirement during development supports the overall status tracking of the project.

Control access RM tools let you define access permissions for individuals or groups of users and share information with a geographically dispersed team through a web interface to the database. Some tools permit multiple users to update the database contents concurrently.

Communicate with stakeholders An RM tool serves as a master repository so that all stakeholders work from the same set of requirements. Some tools permit team members to discuss requirements issues electronically through threaded conversations. Automatically triggered email messages notify affected individuals when a new discussion entry is made or when a specific requirement is modified. This is a convenient method for visibly tracking decisions made about requirements. Making the requirements accessible online can minimize document proliferation and version confusion.

Reuse requirements Storing requirements in a database facilitates the reuse of them in multiple projects or subprojects. Requirements that logically fit into multiple parts of the product description can be stored once and referenced whenever necessary, to avoid duplicating requirements. Chapter 18, "Requirements reuse," describes important concepts regarding effectively reusing requirements.

Track issue status Some RM tools have functionality for tracking open issues and linking each issue to its related requirements. As issues are resolved, it's easy to determine whether any requirements must be updated. You can also quickly find a history of the issue and its resolution. Tracking issues in a tool enables automatic reporting on the status of the issues.

Generate tailored subsets RM tools allow you to extract and view a set of requirements that fits a particular purpose. For example, you might want a report that contains all of the requirements for a specific development iteration, all of the requirements that relate to a particular feature, or a set of requirements that needs to be inspected.

RM tool capabilities

The feature tree in Figure 30-1 presents a summary of the types of capabilities commonly found in RM tools. You can find detailed feature comparisons of many RM tools online (for example, see Seilevel 2011; INCOSE 2010; Volere 2013).

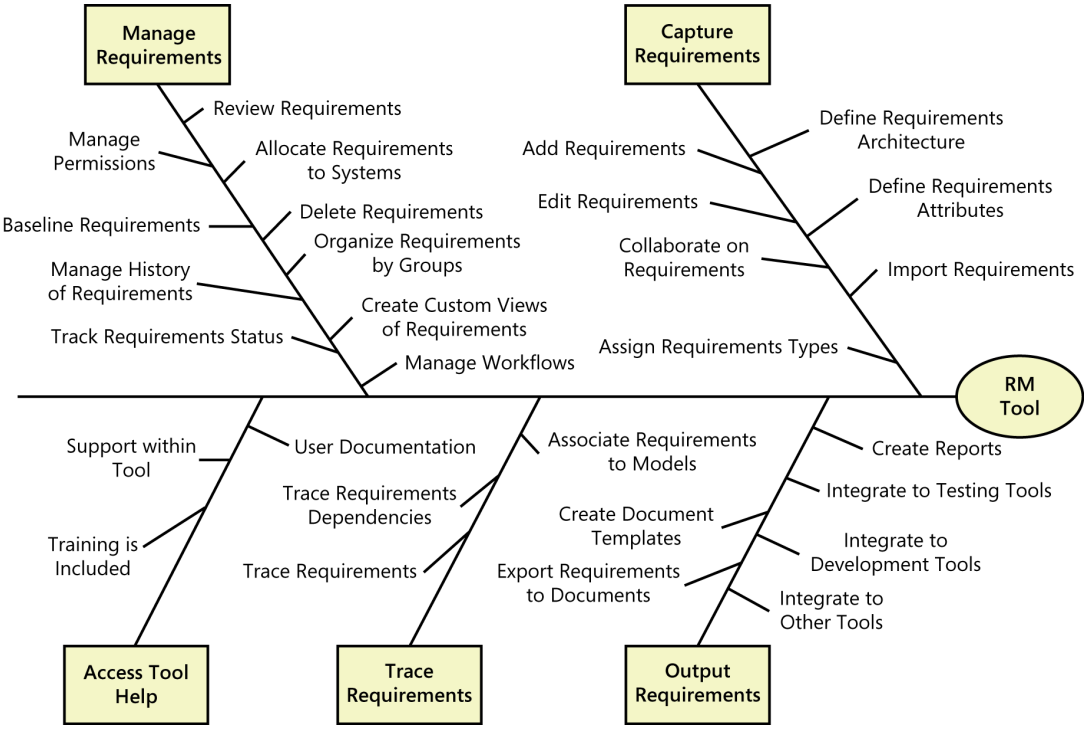

FIGURE 30-1 Common RM tool features.

RM tools let you define different requirement types, such as business requirements, use cases, functional requirements, hardware requirements, and constraints. This lets you differentiate all the types of information that are typically contained in an SRS. Many tools allow you to configure an information architecture (which defines how requirements types and other objects relate to one another) that is customized to your practices. Chapter 29 shows common traceability links that can be defined in the information architecture. Most of the tools provide strong capabilities for defining attributes for each requirement type, a great advantage over the typical document-based approach.

RM tools typically support hierarchical numeric requirement labels, in addition to maintaining a unique internal identifier for each requirement. These identifiers often consist of a short text prefix that indicates the requirement type—such as UR for a user requirement—followed by a unique integer. Some tools provide displays to let you manipulate the hierarchical requirements tree.

Requirements can be imported into an RM tool from various source document formats. The textual description of a requirement is treated simply as a required attribute. Several products let you incorporate nontextual objects such as graphics and spreadsheets into the requirements repository. Other products let you link individual requirements to external files (such as Microsoft Word files, graphics files, and so on) that provide supplementary information that augments the contents of the requirements repository.

Output capabilities from the tools generally include the ability to generate a requirements document in a variety of formats, including predefined or user-specified documents, spreadsheets, and webpages. Some tools allow significant customization for creating templates, allowing you to specify page layout, boilerplate text, attributes to extract from the database, and the text styles to use. Specification documents are then simply reports that are generated from the tool according to certain query criteria, formatted to look like a typical SRS. For example, you could create an SRS that contains all the functional requirements that are allocated to a specific release and assigned to a particular developer. Some tools provide functionality that lets users make changes in exported documents offline, which are then synchronized with the tool's database when the user is back online.

Most tools enable different views of the requirements to be generated within the tool or exported from the tool. Features typically include the ability to set up user groups and define permissions for selected users or groups to create, read, update, and delete projects, requirements, attributes, and attribute values. Setting up appropriate views and permissions facilitates the review of requirements and collaboration to improve those requirements. Some tools also include learning aids, such as tutorials or sample projects, to help users get up to speed.

Requirements management tools generally have robust tracing features. Tracing is handled by defining links between two types of objects or objects of the same type. Some requirements management tools include modeling capabilities that also allow the models to be linked at an element level to individual requirements or to other model elements.

Some agile project management tools also provide RM capabilities. These tools are used to manage and prioritize backlogs, allocate requirements to iterations, and generate test cases directly from requirements.

RM tools often integrate with other tools used in application development, as illustrated in Figure 30-2. Chapter 29 describes how individual requirements can be linked to objects that might reside in these other tools. For instance, you might be able to trace specific requirements to individual design elements stored in a design modeling tool, or to tests stored in a test management tool.

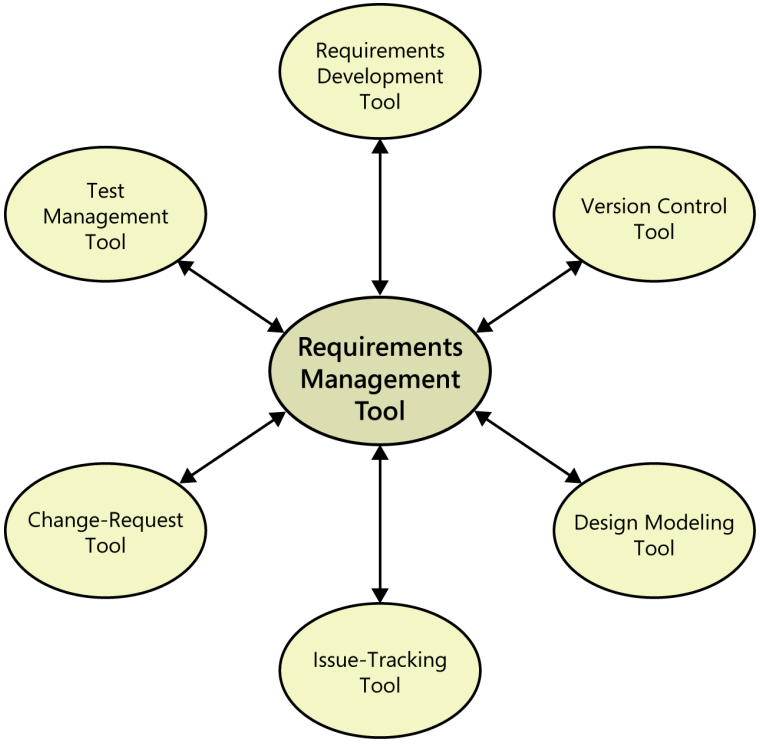

FIGURE 30-2 RM tools integrate with other kinds of software tools.

When you are selecting an RM product, determine whether the tool will be able to exchange data with the other tools you use. Think about how you'll take advantage of these product integrations as you perform your requirements engineering, testing, project tracking, and other processes. For example, consider how you would define trace links between functional requirements and specific design or code elements, and how you would verify that all tests linked back to specific functional requirements have been successfully executed.

Selecting and implementing a requirements tool

Any of these requirements tools can move your requirements practices to a higher plane of sophistication and capability. However, success depends upon selecting the most appropriate tool for your organization and getting your teams to adopt it as part of their routine practices.

Selecting a tool

Select a tool based on the combination of desired features, platform, and pricing that best fits your development environment and culture. Business analysts should lead the selection efforts by defining the evaluation criteria and performing the actual assessment. Some companies outsource tool evaluations to consultants who can assess a company's needs comprehensively and make recommendations from the available tool candidates. If you do the evaluation yourself, the suggestions described in Chapter 22, "Packaged solution projects," for choosing a COTS package also can be applied to selecting a requirements tool. Chapter 22 also offers a real story from one requirements tool evaluation. To summarize the selection process:

1. Identify your organization's requirements for the tool to serve as evaluation criteria.
2. Prioritize and weight the criteria according to what capabilities or other factors matter most to your organization.
3. Set up demos or acquire evaluation copies of the tools you want to consider.
4. Score each tool against the criteria in a consistent manner.
5. Calculate a total score for each tool by using your criteria scores and the weights you assigned to them.
6. For each tool that scored well, use it on an actual project to see if it behaves as you anticipated from the objective scores.
7. To make a final selection, combine the scores, licensing costs, and ongoing costs with information on vendor support, input from current users, and your team's subjective impressions of the products. Two good final questions to ask people who evaluate the tools are, "Which tool would you most want to use?" and, "Which tool would you be most upset about being forced to use?"

Setting up the tool and processes

Recognize that it will take effort to install a tool, load a project's requirements into it, define attributes and trace links, keep the contents current, define access groups and their privileges, and adapt your processes to use the tool. Configuring the tool can be complex; there is a steep learning curve just to set up a sophisticated requirements tool. Management must allocate the resources needed for these operations. Make an organization-wide commitment to actually use the product you select, instead of letting it become expensive shelfware.

There's little point in using a requirements tool if you don't take advantage of its capabilities. I encountered one project team that had diligently stored all its requirements in an RM tool but hadn't defined any requirement attributes or trace links. Nor did they provide online access for all the stakeholders. The fact that the requirements were stored in a different form didn't provide significant benefits, although it consumed the effort needed to get the requirements into the tool. Another team stored hundreds of requirements in a high-end tool and defined many trace links. Their only use of the information was to generate massive printed traceability reports that were supposed to

be reviewed manually for problems. No one actually examined the reports, and no one regarded the database as the authoritative repository of the project's requirements. Neither of these organizations reaped the full benefits of their considerable investments of time and money in the tools.

Even if you select the best available tool, it won't necessarily provide every capability that your organization wants or needs. It might not support your existing requirements templates or processes. You'll still likely need to adapt some of your existing processes to incorporate the tool in them. Expect to have to make some changes to templates, attribute names, and the sequencing of requirements development activities. Consider the following suggestions to overcome process issues as you strive to maximize your return on investment from a requirements tool:

- Assign an experienced BA to own the tool setup and process adaptations. She will understand the impact of configuration choices and process changes.
- Think carefully about the various requirement types that you define. Don't treat every section of your current SRS template as a separate requirement type, but don't simply stuff all of the SRS contents into a single requirement type either.
- Use the tool to facilitate communication with project stakeholders in various locations. Set the access and change privileges to permit sufficient input to the requirements by various people without giving everyone complete freedom to change everything in the database.
- Don't try to capture requirements directly in an RM tool during your early elicitation workshops. As the requirements begin to stabilize, though, storing them in the tool makes them visible to the workshop participants for refinement.
- Use RD tools during elicitation activities only if you are confident that they will not slow down the discovery process and waste your stakeholders' time.
- Don't define trace links until the requirements stabilize. Otherwise, you can count on doing a lot of work to revise the links as requirements continue to evolve.
- To accelerate the movement from a document-based paradigm to the use of the tool, set a date after which the tool's database will be regarded as the definitive repository of the project's requirements. After that date, requirements residing only in word-processing documents won't be recognized as valid requirements.

Provided you remember that a tool can't overcome process deficiencies, you're likely to find that requirements tools greatly enhance the control you have over your software requirements.

Important Don't even pilot the use of an RM tool until your organization can create a reasonable software requirements specification on paper. If your biggest problems are with eliciting and writing clear, high-quality requirements, an RM tool won't help you (although an RD tool might).

Facilitating user adoption

The diligence of the users of your requirements tools is a critical success factor. Dedicated, disciplined, and knowledgeable people will make progress even with mediocre tools, whereas the best tools won't pay for themselves in the hands of unmotivated or ill-trained users. Don't write a check for a tool unless you're willing to respect the learning curve and make the time investment.

Buying a tool is easy; changing your culture and processes to accept the tool and take best advantage of it is much harder. Most organizations already are comfortable with taking elicitation notes in a word-processing document or by hand, and with storing their requirements in documents. Changing to use software-based tools requires a different way of thinking. Using RD tools requires breaking old habits for running elicitation sessions. An RM tool makes the requirements visible to any stakeholder who has access to the database. Some stakeholders interpret this visibility as reducing the control they have over the requirements, the requirements engineering process, or both. Some people prefer not to share an incomplete or imperfect set of requirements with the world, yet the database contents are there for all to see. Keeping the requirements private until they're "done" means you miss an opportunity to have other pairs of eyes scan the requirements frequently for possible problems.

People are often resistant to change things that they're familiar with, and they usually have a comfort level with working on requirements in documents. They might have a perception—even if incorrect—that using a requirements tool will be harder for them. Also, don't forget that most of the tool users are already busy. Time must be allocated to let them get used to using the tool in their daily jobs. Eventually, the tool probably won't actually require more time from users, but they first need to get over the learning curve and develop new work habits using the tool. Following are some suggestions to help you deal with issues regarding user adoption and culture change:

- Identify a tool advocate, a local enthusiast who learns the tool's ins and outs, mentors other users, and sees that it gets employed as intended. This person should be an experienced business analyst who can be the single owner for ensuring tool adoption. This initial tool advocate will work with other users on their projects to ingrain the tool into their daily activities. Then he'll train and mentor others to support the tool as other projects adopt it.
- One of the biggest adoption challenges to overcome is that users don't believe the tool will actually add any value. Perhaps they haven't recognized the pain from limitations of their existing manual approaches. Share stories with them about where the lack of a tool caused a negative impact and ask them to think of their own examples.
- Your team members are smart, but it's better to train them than to expect them to figure out how best to use the tool on their own. They can undoubtedly deduce the basic operations, but they won't learn about the full set of tool capabilities and how to exploit them efficiently.
- Because you can't expect instantaneous results, don't base a project's success on a tool you're using for the first time. Begin with a pilot application of the tool on a noncritical project. This will help the organization learn how much effort it takes to administer and support the tool. Chapter 31, "Improving your requirements processes," describes the learning curve associated with adopting new tools and techniques.

The proliferation and increased usage of tools to assist with requirements development and management represents a significant trend in software engineering that will undoubtedly continue. Too many organizations, though, fail to reap the benefits of their investment in such tools. They do not adequately consider their organization's culture and processes and the effort needed to shift from a document-based requirements paradigm to a tool-based approach. The guidance in this chapter will help you choose appropriate tools and use them effectively. Just remember, a tool cannot replace a solid requirements process or team members with suitable skills and knowledge. A fool with a tool is an amplified fool.

Next steps

- Analyze shortcomings in your current requirements process to see whether a requirements development or requirements management tool is likely to provide sufficient value to justify the investment. Make sure you understand the causes of your current shortcomings; don't simply assume that a tool will magically correct them.
- Before launching a comparative evaluation, assess your organization's readiness for adopting a tool. Reflect on previous attempts to incorporate new tools into your development process. Understand why they succeeded or failed so that you can position yourselves for success this time.