Collaborate for Quality

Using workshops to determine your project's requirements

by Ellen Gottesdiener

Just how important is it to fully develop your project's requirements? After all, nailing down your requirements is usually only 8% to 15% of your overall project effort. Truth be told, it's not really something you'll want to spend your resources and energy on—*unless*, that is, you care at all about the quality of your product, your customers' level of satisfaction, and the amount of post-implementation repair you'll have to take care of down the road.

Why is it so important to get requirements right? For one thing, you're likely to introduce more defects into your software product in the requirements phase than in any other phase—and these defects account for as much as half of the product's total defects. Defects in requirements are harder to remove than defects originating in any other phase. But that's not all. Fixing them later

QUICK LOOK

- The benefits of requirements workshops
- Applying proven patterns for effective collaboration
- Examples from successful efforts

Workshop participants, sometimes called a *task group,* are productive because collectively they have the right mix of skills and knowledge to create the work products. Members of the task group act interdependently, relying on one other's knowledge, experience, skills, and perspectives.

in the project will *cost* you more, too—as much as 100 times more after implementation than if you detected and corrected them in the requirements phase. It's no wonder that rework due to requirements defects can eat up as much as 50% of your overall budget.

One other aspect of low-quality requirements is harder to measure, but just as treacherous. It's called "scope creep," and it's often cited as the most vexing problem in software development. Unrestrained by carefully developed requirements and mutual IT-customer or product development agreement, the scope of the project keeps creeping—expanding as the work proceeds.

For all these reasons, project teams are searching for ways to develop requirements that are as free from defects as possible. One way to develop high-quality requirements is based on the use of collaborative workshops along with walkthroughs and QA checklists. As this article will illustrate, that combination of best practices gives you a powerful and efficient way to deliver quality user requirements—and, by extension, quality software products.

What Makes Workshops Work?

In collaborative workshops, participants share a common goal, and they agree to join together to create work products in the pursuit of that shared goal. Workshop participants, sometimes called a *task group*, are productive because collectively they have the right mix of skills and knowledge to create the work products. Members of the task group act interdependently, relying on one other's knowledge, experience, skills, and perspectives. They are cohesive, meaning that they are motivated to act together. With appropriate direction, such a group can be highly productive.

If you're familiar with the acronym JAD (joint application design), then you're already familiar with one type of *collaborative workshop.* JAD workshops are structured events in which a carefully selected group of stakeholders and content experts works together to create, correct, and document a predefined *deliverable*, or *work product.* The group agrees ahead of time on the deliverables, and the participants often produce some of them before the workshop; this sort of timeboxing enables the group to focus on what's really important. The most successful workshops are composed of a healthy mix of business experts (or product development people representing those experts) and IT people, led by a neutral facilitator and a scribe who documents the group's work as it proceeds.

How are JADs and other collaborative workshops different from *other* kinds of business meetings?

On the surface, collaborative workshops are like "meetings" in that both types of gatherings involve people meeting together at the same time, and both (presumably) follow a logical flow. But there are significant differences. Among them is the presence of two people who fulfill two specific process roles: facilitator and scribe. The *facilitator* is responsible for managing the group's activities, dynamics, and work products. The *scribe* documents the group's work as it proceeds. Neither facilitator nor scribe operates as a content expert; nor does either collaborate in product creation. As a result, they are free to focus on the process. As the group becomes more familiar with the process, the facilitator's role in controlling the process can be relaxed.

Another important difference between a workshop and a meeting is that a workshop is authorized by a sponsor, who ensures that the right participants are present, verifies the workshop's purpose, and ensures that the workshop outcomes are implemented.

In effective collaborative groups, energy is high, individuals respect one another, skills are complementary, and responsibilities and roles are clear both inside and outside the group setting. To maintain energy, creativity, and motivation, the facilitator uses interactive as well as parallel group activities. For example, in a user requirements workshop, subgroups can be formed to work on portions of a single deliverable, such as business rules. Alternatively, subgroups might be assigned to work on entire work products—one group may work on use cases while another drafts a prototype and yet another creates a high-level class model. The subgroups then reconvene in a plenary (whole group) activity to share and critique their work.

The process works. Collaborative workshops have proven to be remarkably successful as a means to reduce risk, enhance quality, and increase productivity. They can reduce requirements creep by almost half. But that's only a few of their benefits. They can also

- commit users to the requirements definition process
- promote ownership for the deliverables and, ultimately, the system
- shorten the requirements phase
- eliminate nonessential requirements
- form or reinforce effective communication patterns
- build trust among project participants

By actively involving users in eliciting and testing requirements, successful workshops help you reduce defects in requirements—and en-

hance team communications along the way.

If you've ever experienced a poorly run meeting or workshop, you know how unproductive and negative an experience that can be. Successful ones, on the other hand, have a different feeling and definite flow. How can you learn from these successes, and reproduce them for your requirements workshops? One way, discussed next, is to apply proven "patterns" for effective collaborative work.

Collaboration Patterns

A pattern describes a known solution to a specific type of problem, documenting core insights or instructive information. A pattern is a best practice that can be applied in new, similar situations. Our software community has used patterns to solve problems related to software analysis, architecture, and design. More recently, patterns have been documented and applied to software development processes and organizations.

Similarly, *collaboration patterns* are recurring activities used successfully by collaborating groups. They are high-level blueprints for the behavior that these groups undertake to accomplish results together. When groups collaborate in a facilitated workshop setting, the facilitator often establishes the collaboration patterns. With experience, the group learns to incorporate these patterns, reducing the need for the third-party facilitator.

"Divide, Conquer, Correct, Collect" (see Table 1,) is one pattern that groups can use to elicit and test a requirement deliverable in a group. If you were applying this to a use case, the first step—*dividing* the use case—would be to partition it into its component parts, such as its name, header information, a brief description, a stepwise description, and exceptions. To *conquer* use cases in workshops in large groups (seven or more participants), each subteam focuses on one part of a single use case. Alternatively, multiple subteams can conquer different use cases at the same time and then reconvene for the next step.

In the *correct* part of the collaboration pattern, you test the use cases with other requirements, such as scenarios and business rules, and a QA checklist. Next, you *collect* all the parts of a use case along with all the use cases for the release. These, too, are tested as a whole. This final testing can be driven by scenarios and a use case navigation diagram (a visual diagram showing the relationships among all the use cases).

In the "Divide, Conquer, Correct, Collect" pattern, the group follows these general steps:

Name	**Divide, Conquer, Correct, Collect**
Context	A complex product needs to be created quickly by people with differing expertise. Individuals need to be familiar with the content to reduce the learning curve later in the project.
Problem	How do you make sure all participants have input and verify the product? How do you permit participants to work on aspects of a product with which they are familiar and also aspects with which they are unfamiliar?
Solution	**Divide**—partition each product into component parts and categorize them **Divide**—map out the relationships between the parts into logical order, ensuring that concurrency as well as dependency is present **Conquer**—allocate partitions to subgroups with expertise in that category of the product **Correct**—conduct whole group QA activities **Collect**—synchronize the product elements
Consequences	Closure on decisions; since participants have created the end product by portioning it, working in parallel, and sharing details of each partition, greater in-depth knowledge of product is obtained.
Entry Criteria	■ Shared time and space of knowledgeable participants ■ Ability to subdivide participants by deliverable or by experience ■ Ability to logically partition the deliverable ■ List of quality criteria for each deliverable ■ Active involvement of knowledgeable users
Exit Criteria	Creation of the end product that is agreed upon by all participants and to which all participants have had input in both creation and quality checking
Uses	Statement of business goals and objectives, creation of context-level use case, use case text, business rules, state chart diagram, project plan, migration strategy, communication plan, actor catalog

TABLE 1 The Divide, Conquer, Correct, Collect collaboration pattern

The *facilitator* is responsible for managing the group's activities, dynamics, and work products. The *scribe* documents the group's work as it proceeds. Neither facilitator nor scribe operates as a content expert; nor does either collaborate in product creation. As a result, they are free to focus on the process.

1. Define all the necessary deliverables
2. Determine the parts or components of each deliverable
3. Define quality criteria for each part
4. Devise questions for each part that will test its quality
5. Define related deliverables that can be used to test the part
6. Repeat steps 1–5 for all deliverables in a set (such as all requirements deliverables)
7. Arrange all parts into a logical sequence
8. Ensure that all parts to be defined concurrently are at the same level of detail
9. Define where and how all the documentation will be stored and kept up-to-date
10. Design groups of workshop activities from the logical sequence of parts
11. Define focus questions to lead the group into creating the parts
12. Design specific group processes for each activity
13. Lead the group through the pattern, continually checking for quality, common level of detail, and the view of the whole
14. Test the whole

In this way, you can use collaborative workshops to develop high-quality user requirements (see Figure 1). For your deliverables, you can choose from a number of representations—text, use cases, data models, business rules, events, scenarios, prototypes, scenarios, and dynamic models—instead of or in addition to use cases. To maximize productivity, I recommend using a combination of text and diagrams to represent requirements.

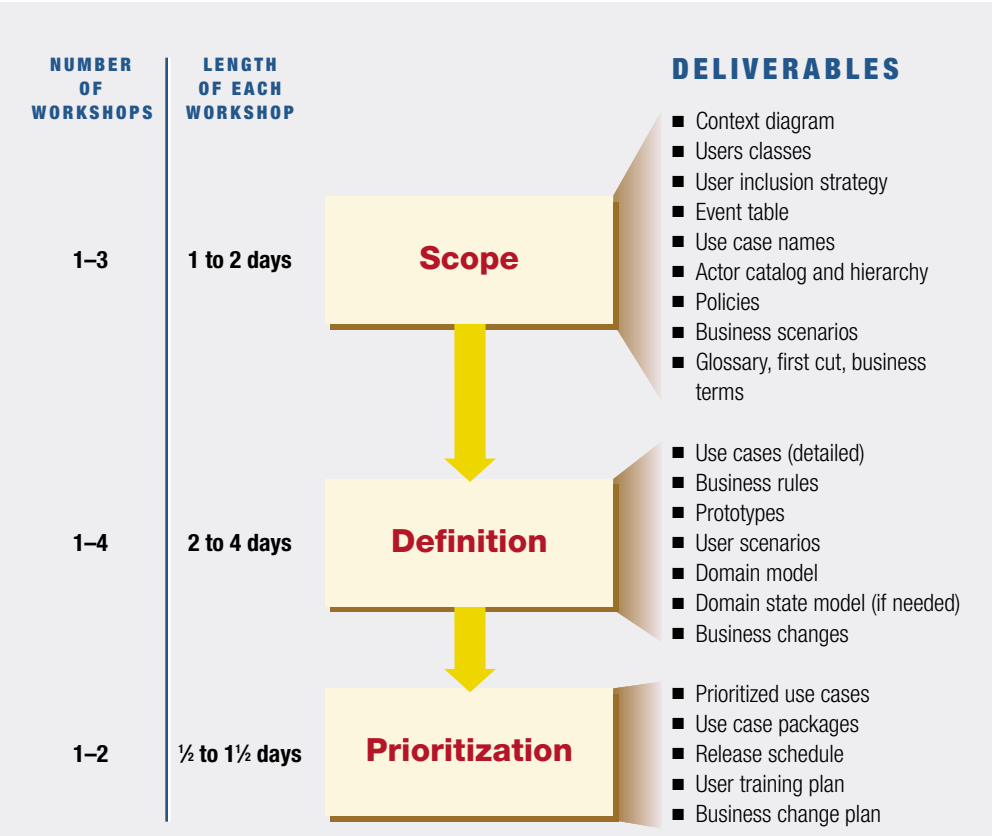
If you're keen on trying collaborative workshops, keep reading for an example of one successful workshop.

Workshop Walkthrough

In a two-day requirements workshop I recently facilitated, we applied Jerry Weinberg's concept of "egoless programming" to requirements. Weinberg's idea, cited in his 1971 book *The Psychology of Computer Programming* (reissued as a Silver Anniversary edition by Dorset House in 1998), was for participants to turn away from being defensive about their work products, focusing instead on the quality of the work. By "restructuring the social environment"—using peers to objectively evaluate the work—it's possible, Weinberg argues, to find not only defects but ways to improve the product. Because my group placed a high priority on quality requirements, the team members were interested in implementing this concept. To do that, we used walkthroughs.

In a *walkthrough*, a producer—the person who created the product—describes the product and steps through its contents while the team jointly takes responsibility for evaluating its quality. In addition to learning from our own and each other's mistakes, an added benefit of using walkthroughs is that we tend to do a better job of creating the product in the first place.

Before the workshop, the team had decided that use cases would be



Note: Some user requirements deliverables are created outside workshops (e.g., the glossary and nonfunctional requirements). As pre-work to each workshop, drafts of some of the deliverables are created.

FIGURE 1 Collaborative user requirements workshops

a good way to represent user requirements. (A *use case* is a functional requirements model that defines a specific use of the system and can be represented in a diagram, in a text specification, or both.) They had specified and documented a list of potential use case names and a set of scenarios. (*Scenarios* are descriptions of the system in action, including sample data, used to create test cases and, later, test scripts.) The group also drafted a glossary of terms, and the team's GUI designer created a set of screen mockups for each of the potential use cases.

At the workshop, participants created detailed use cases and wrote business rules. As they specified the use case steps, our scribe recorded them using a laptop. They were also posted on the wall in sequence, left to right, on big blue adhesive notes. Beneath each use case was a list of business rules. The team then used the scenarios to walk through each use case along with its associated business rules and the prototype screens.

Here's how it worked. Sarah, a business expert, played the role of a user. She selected a scenario and then walked through a use case posted on the wall, step by step. At the same time, Dave, the GUI designer, was poised at the overhead projector with a pile of prototype screens on acetates. Using an erasable colored pen, he wrote data from the scenarios on the acetate as Sarah walked through the steps. He not only modified the data on the screen shots but also rearranged and redrew rough screen shots to give Sarah the interface she expected to see at that point in the scenario.

We first worked through the "normal" flow of a use case. As we tested the use case with the data provided from each scenario, I would ask a question such as, "How do you decide which…?" or "How does the system select…?" or "What does the system have to know to…?" and so on. These questions were designed to test the business rules. When the group realized that a business rule was incomplete or unclear, the scribe recorded the revision un-

PERSPECTIVE

Drawing People Together

Maître d's take note. You may want to seat Becky Winant's collaborative lunch teams at a table with a flip chart if you want to preserve your pristine tablecloths; more often than not, she's got a magic marker in her hand, busy sketching out context diagrams and product features.

Technology, she says, provides amazing tools for communication and modeling—two of her specialties—but sometimes low-tech pen and paper is the best bet for helping people collaborate. Winant has been involved in software development for thirty years, twenty of those spent building analysis and design models and conducting workshops. "Sometimes just sketching a product or a feature on paper jump-starts the conversation," she says. "I've learned that people can't really go anywhere with a model unless you start at the very beginning—having some interaction with all the people involved in the project, who have some stake in it, to decide what they want to do with it."

Winant has found her early-stage collaborative workshops to be an ideal place to nurture basic understandings. "When I'm facilitating meetings with engineers on embedded products, I'll get them talking about a piece of equipment they're going to have to write operating software for," she says. "Something they're trying to wrap their minds around and understand the requirements for. Drawing a picture of it helps them understand the functions and components and gives them a chance to say, 'Hey, I guess I really *don't* know what happens there.'"

That kind of realization, she says, is what you want to happen during collaborative workshops—not halfway through development when you've already built in another direction, or worse, six months later when the product has shipped.

Basic face-to-face talking is key. "There's an unfortunate habit in software today," Winant notes, "to buy tools to make a system appear." Modeling tools are getting better and better, to the point that you can sit down in a room by yourself and build detailed and concise models that tie in to your final system. While those kinds of templates and automation are great, she stresses that they fall short of addressing the really hard problems in project teams: making sure everyone's communicating clearly about requirements and expectations. "The collaborative stuff," she points out, "is where everyone at the table gains deeper understanding of what you're doing, what targets you want to achieve, and how things will be different—better—when you're done."

—A.W.

der the use case steps for everyone to see.

After working through three or four normal scenarios, we attacked the *exceptions:* those scenarios that would occur less often or those that would cause errors. We had posted the use case steps that had exceptions using different-colored adhesive notes. These exceptions appeared as branches off the normal use case steps. Walking through the scenarios had already revealed that numerous business rules were missing. As we walked through an exception scenario, sometimes the participants realized that they needed to add another exception. These in turn yielded more use case steps, test data, and changes to the flow of the prototype screens. We kept track of the corrections on the actual models. You can also use a form to track all the defects in one place.

At the conclusion of each use case's walkthrough, we used a collaboration pattern I call "Decide How to Decide" to determine the disposition of each use case. Our scribe, working on his laptop, projected our Use Case Completion form on the wall. At this point, I polled the group for input. Using a predetermined decision rule, the sponsor made the final decision. As specifics were discussed, the scribe captured notes and the final decision on the form. (See this article's StickyNotes for a sample template for use case scenario testing, as well as a use case completion form.)

QA Checklists

A second technique you can integrate into collaborative requirements workshops is *QA checklists* (see Table 2). A checklist can provide more benefits than may be immediately obvious—the very process of creating and agreeing on the checklist helps IT and users clarify and define expectations for each deliverable clearly and precisely. Like walkthroughs, checklists push participants to create high-quality requirements in the first place. The checklist forces you to focus on the end from the beginning.

Using checklists is another example of the "correct" phase of "Divide, Conquer, Correct, Collect." The process is simple: You compare each requirement to the checklist questions and discuss and correct any discrepancies…right away, if feasible.

In one workshop I facilitated, the group created scope-level requirements in the forms of an *event table* (naming the events that trigger the system to act) and a *context diagram* (showing what the system gives and gets from things that interact with it). I divided the group of fourteen people into subgroups. Each subgroup was given a copy of the checklist illustrated in Table 2. As a facilitator, I've discovered that participants give you what you ask for. My experience is that taking a testing attitude toward deliverables helps workshop participants find more defects and find them earlier. So I told them, "Find 'boo-boos' in what we created."

Events
Is each business event unpredictable with respect to timing and frequency and does it originate from outside the system?
Does the list of events include both business and temporal events?
Are temporal events in the format of "time to…"?
Are temporal events truly temporal, that is, a specific time can be identified when the event can fire automatically?
Are business events in the format of "subject + verb + object"?
Does each event result in a response that is documented in the event table?
Are all the expected outcomes (responses) listed as responses?
Is each response listed separately?
Are responses that are custodial in nature (updating files) documented in the event table?
Is each term (noun) listed in each event also defined in the glossary?
Have out-of-scope (indirect) events been removed?
Have internal activities (internal events or actions) been removed from the event table?
Context Diagram
Does the label for the center describe the essential nature of the project's scope?
Does each inflow correspond to one and only one business event?
Is each inflow labeled with the information or product that is passed to the system?
Is each inflow labeled at the same level of detail?
Is each inflow labeled without verbs?
Does each outflow correspond to one noncustodial response in the event table?
If not, is that response custodial in nature? (updating files, databases)
Is each noun on inflows and outflows documented in the project glossary?
Event Table and Context Diagram
Does each event correspond to a response that is documented in the event table?
Is each inflow labeled with the information or product that is passed to the system?
Is there a single inflow on the context diagram corresponding to each business event?
Is there a single outflow on the diagram corresponding to one response to an event?
Can each inflow and outflow be corresponded to a row on the event table?

TABLE 2 QA checklist

Each subgroup indeed found defects, which were shared with the larger group and corrected. For example, they forgot that they had to get periodic updates from an employee database, and they realized they would need someone to play the role of approving certain types of queries to sensitive data. After that, they continued the workshop by defining detailed requirements for each event in scope.

Alternative Approaches

When should you *not* use collaborative workshops?

First, if you can't identify a workshop sponsor, you won't have the necessary commitment to ensure that the right players participate and prepare for a successful session. Second, in some organizations it isn't feasible to gather together users or their surrogates at the same time and place. In that case, however, you might consider using collaborative software (see this article's StickyNotes for information on collaborative software resources).

Collaborative software tools allow people to gather in a "different time–different place" virtual environment to generate, summarize, and prioritize ideas. Be aware that you will need a skilled facilitator who is experienced in that software product to chauffeur the tool for you.

Finally, don't use collaborative workshops unless you have a facilitator who is neutral to the outcome, experienced with group process, and knowledgeable about the deliverables you need to create in the session. These skills can be developed within your organization and shared across projects.

In lieu of workshops, you can use techniques such as observation, interviews, surveys, prototypes, competitive analysis, or product complaint data. Of course, these techniques can be powerful in combination with workshops, if workshops seem to be a fit for you.

Risks and Payoffs

Like other approaches, workshops come with some pitfalls. Here are some conditions to watch out for:

- participants don't share a common goal
- the sponsor and participants can't agree on the deliverables prior to a workshop
- participants aren't educated about the purpose and format of the workshop and the intended deliverables
- the sponsor doesn't budget enough time and money to allow participants to contribute to all phases of the workshop process
- the workshop doesn't include people with a healthy mix of perspectives and experience
- you don't have an experienced facilitator
- your facilitator is a key stakeholder in the outcome, and thus has difficulty being neutral and focusing on the group process
- project or company politics have poisoned the players' ability to collaborate
- participants can't agree on ground rules for conducting the workshop
- the work that gets created is not used or acknowledged by management

If you address these potential risks, you're ready to use well-run collaborative workshops. Combine these workshop suggestions with techniques such as walkthroughs and QA checklists. Those methods aren't new, but they're proven best practices in software development, and work well when integrated into the workshop process.

The payoffs can be dramatic. Even though gathering your requirements may seem to be a small part of the effort you devote to a project, getting your requirements right the first time through the use of collaborative workshops can have a big impact on your quality of work—and your team's sanity. STQE

Ellen Gottesdiener, principal consultant at EBG Consulting (www.ebgconsulting.com), *provides consulting, facilitation, and training to business-driven IT projects. Her book* Collaborative User Requirements *will be published in 2001.*

Design Your Own Workshop: Mind Your Six Ps

To design a workshop, you first define what I refer to as the "six Ps": purpose, participants, principles, products, place, and process. These six elements help you build the framework for a successful workshop. They answer the questions why, who, how, what, where, and when.

Purpose

The statement of the workshop's purpose outlines the reason and justification for the workshop. It explains why you're conducting the workshop and serves as a frame of reference.

It may seem obvious, but it's important to remember that a workshop delivers work products that are needed by the project. Because the workshop's context *is* the project, the workshop's purpose is linked to the project's purpose, which describes the business reason for undertaking the project. The statement of the project's purpose usually references the current business situation, the desired situation, the obstacles to achieving the desired situation, and the changes that are desired. Similarly, the workshop's purpose describes the business reason for gathering the players together. The link must be evident to all stakeholders. It's hard to overestimate the benefits of well-defined purposes for projects and workshops.

Participants

The people involved in a facilitat-

Framework for Designing a Collaborative Workshop

FOCUS	PURPOSE	PARTICIPANTS	PRINCIPLES	PRODUCTS	PLACE	PROCESS
Questions	Why are we doing the workshop?	Who is involved?	How should we function as a group?	What should the workshop produce?	Where should we gather and share space?	When should things happen, and in what order?
Concerns	■ Goals ■ Need ■ Motivation	■ Roles ■ Stakeholder ■ Experts	■ Guidelines for participation ■ Ground rules ■ Group norms	■ Work products ■ Dependencies ■ Models ■ Decisions ■ Next steps ■ Issues for resolution	■ Location ■ Time	■ Steps ■ Activities ■ Order ■ Concurrency

ed workshop are the workshop sponsor, the participants, a facilitator, a scribe, observers, and on-call subject matter experts. Another term often used to describe the various people involved in collaborative projects is *role*. When you explicitly define the individuals who are to play each role, the group can better plan the session, the participants are better prepared, and the event is more likely to succeed.

Not all of these roles are necessarily present during a given workshop. For example, the workshop sponsor may be present only at the beginning and end of the session.

Principles

Also called *ground rules*, principles are guidelines for participation, or the codes of conduct that participants agree to follow. Groups need these precepts to maintain socially acceptable behavior and to promote the goals of the workshop: delivering the appropriate work products in the allotted time. The principles serve as a process guide for the facilitator as well as the participants.

The facilitator works with the workshop sponsor and group to establish principles, which must be clear and acceptable to all participants. The facilitator and participants are responsible for monitoring the adherence to the principles.

Products

The work products, or deliverables, of a workshop take the form of text or diagrams. For user requirements workshops, they are lists of business policies, use case text, use case diagrams, atomic business rules, and the like. These workshop deliverables, in turn, serve as input to project activities, including design, development, or additional workshops, such as definition or design workshops.

To accelerate the delivery of workshop products, the participants need certain input products such as draft models, text, and documentation. These materials may already exist or may need to be created before the workshop. In one requirements workshop, for example, I asked the team to produce prototype screens and a list of free-form business rules for each of the use cases that had already been drafted. These additional input products proved to be critical in accelerating the delivery of higher-quality use cases.

Place

The location of a workshop can influence the outcome. If you're using a technique that requires the use of giant pieces of paper mounted on the walls, for example, the room needs plenty of wall space. If the room is crowded, subteam activities might be impossible. It's also important that the room has the space and amenities you need to serve refreshments.

- Should you hold your workshop onsite near participants' work areas, or offsite? It's good for participants to have easy access to files, documents, and materials, but it also means that they can get pulled away from the workshop to deal with email, voice mail, or peer questions.

- In evaluating the best place to hold the workshop, consider room size, wall space, support for refreshments, seating (a U-shape arrangement is best), availability of outlets, access to on-call subject matter experts (via physical proximity or phones), and access to phones during breaks and lunch.

Process

Plan the workshop's activities so they follow a logical flow in which the outputs from one activity are used by the next activity in the session (or in a post-workshop project task). An activity may consist of several steps in which members work individually, in small teams (subteams), or as a whole group.

To streamline the process and improve the quality of deliverables, it's a good idea to follow a collaboration pattern. For a list of best practices for integrating workshops into the requirements phase, see the StickyNotes feature at the end of this article.