419 Hudson Road • Sudbury, MA 01776 USA
Phone: 978.261.5552 • Fax: 978.261.5553 • www.ebgconsulting.com

Scope Keep, Not Scope Creep©

By Ellen Gottesdiener, EBG Consulting

Author's note: this article was first published on Stickyminds.com as a Stickyminds Original: December 16, 2008.

Summary: *An often-cited bugaboo of many projects is scope creep--the unrestrained expansion of requirements as the project proceeds. Yet requirements development is about gaining an ever-growing understanding of requirements. So, isn't scope creep normal? Find out in this column as Ellen Gottesdiener explores exactly how to keep scope under control.*

It's true that requirements evolve as you explore, analyze, prototype, and build a product, so some growth is expected. The key concept is "unrestrained expansion." As you learn more about requirements, you have to make sure they are in scope--and if they are not, whether they should be.

All teams need to establish scope and reasonably react to changing needs. In working with teams starting up their product development efforts, as well as those in the midst of contentious projects, I have found a set of efficient, simple ways to explore, communicate, and adapt product scope.

Your Product Vision

To paraphrase the Cheshire Cat of *Alice in Wonderland*, "If you don't know where you're going, any road will take you there."

Having a product vision keeps scope in check because it provides all stakeholders with a common understanding of the final product for all stakeholders. It serves as a rallying point or "elevator test" (something that anyone on the project can explain in a minute or so, as if to someone between floors in an elevator).

Derive your product vision by collaboration--gather key stakeholders together to co-create the vision. You can use a vision statement or a visual diagram to articulate the vision.

419 Hudson Road • Sudbury, MA 01776 USA
Phone: 978.261.5552 • Fax: 978.261.5553 • www.ebgconsulting.com

A product vision statement articulates three things: what you need to build, for whom you need to build it, and how it's different from what now exists. (For a fast and effective vision statement format, see my book *The Software Requirements Memory Jogger*. This template is based on Geoffery's Moore's product differentiation statement from his book *Crossing the Chasm*.)

You won't know whether new or changing requirements are valid unless you know whether they're aligned with the product vision. In addition, I recommend that teams create a visual diagram, metaphor, or product box (a low-fidelity physical package for the product, as if you were going to buy it at a computer store).

I noticed while working recently with an agile team that their focus was on delivering stories to test the architecture. Priorities were being determined by the team because the business customer (product owner) was disengaged and rarely attended their iteration planning workshops.

It wasn't until we conducted a backlog-development workshop in which an early activity was to build the product vision (in both words and drawings) that the team and customer converged on what requirements were most important to deliver.

The customer thereafter attended each iteration-planning workshop as well as the daily standup meeting. The vision statement and visual posters remained in the team room. At the start of each iteration-planning workshop, the team referred to the vision as the way to start the discussion with the customer about the theme for that iteration, which had to align with the product vision.

Useful Scope Models

When the stakeholders have agreed on the vision, it's time to create requirements models (representations of requirements using text and diagrams) to articulate the boundary between what is in and what is out of scope for the product. Sample scope models include a context diagram and an event-response table. These models should be developed collaboratively with the product owner and other business experts.

419 Hudson Road • Sudbury, MA 01776 USA
Phone: 978.261.5552 • Fax: 978.261.5553 • www.ebgconsulting.com

One infrastructure team I worked with was having difficulty getting started on the project and knowing which people were needed to deliver enhancements to their document-management platform. Once we started to model the needs using a context diagram and event-response table, they began to question, argue (in a good way), and then agree on what interface changes were needed and what skills were required to get the job done.

If your product will support a large business process, part of your scope modeling might include business modeling. A business process is a set of related tasks that creates something of value to the business. Business modeling helps you understand how the software product will support business processes and which processes should be allocated to software and to business people (e.g., updating documents, reworking guidelines, conducting training, revising standard operation procedures, and creating marketing collateral).

Business modeling also helps you define efficient processes for using new software. It is important to have when selecting commercial, off-the-shelf (COTS) software.

Another team found that modeling their supply-chain business processes enabled them to prioritize better which requirements should be addressed first for their large enhancement project. How? Having all product managers along the entire supply chain collaborate to build an end-to-end process model revealed errors in how they made decisions about reallocating items to invoices.

Whether or not you need to conduct business modeling, a clear definition of product scope also narrows the project's focus, thereby enabling better planning, use of time, and use of resources.

If the scope is unclear, it is better to cancel or delay the project until the key stakeholders can clarify and agree on the scope. Canceling a project (early!) might be the most successful possible outcome.

Transparent Prioritization Criteria

Labeling the priority of each requirement with a ranking scheme such as "high, medium, or low" or “essential, conditional, or optional" is not enough. The scheme must include clear criteria for determining what makes a requirement "essential" or "optional." This means you need to tailor those definitions for each project.

On some projects, essential requirements (those rated "high" or "must have") relate to requirements that return high business value, reduce regulatory risk, delineate the product in the market space, and so on. The product owner or business sponsor owns these criteria. In collaboration with the technical team, the criteria owner should write them, get agreement on them, and continually use them to prioritize new and changing requirements.

For example, on one project for reporting incident data in a regulated environment, the "essential" ranking was reserved for requirements that, if not delivered, would put the company in jeopardy of FDA violation. A violation could result in fines and negative publicity. The stakeholders worked together to arrive at this and two other ranking schemes and their definitions. They then used these ranking schemes throughout their requirements workshops to decide which data *really* was needed for the data warehouse being built to meet the reporting needs.

Change Management

Change management is *not* change prevention. You need a way to collect and evaluate new and changing requirements as they come in. Agile teams use a product backlog, and traditional teams use a change-control board and change-control process.

Agile teams work on a small subset of requirements within a time box or iteration. During any given time box, they use a lightweight model (such as a story or scenario) to capture new or changing requirements and then stack them into a product backlog. When the iteration time is reached, the team conducts a demonstration and then an iteration retrospective.

Next, they plan the next iteration by looking at the requirements in the product backlog as well as new requirements that came out of the demonstration and retrospective. The customer or product owner has the responsibility--with input from the delivery team--to select backlog items to be delivered in each iteration.

Traditional teams use more "ceremony" in their development activities, such as large specification documents, planning documents, formal reviews, signoffs, milestone or quality "gate" meetings, and so on. New and changing requirements are captured (perhaps in a requirements management tool). Periodically a group of decision makers, called the change-control board, meet to review and decide what to do with these requirements.

Each project adapts its own change-management process. The key is to keep it simple and timely.

Decision Rules and Transparent Decision Making

My guess is that you've been through this: A lack of clarity about when and whether a decision has been made slows things down and causes team members to spin their wheels.

Making decisions about what is "in" and what is "out" of scope is essential. Furthermore, those decisions are not static. Teams need continually to revisit scope as development unfolds and they learn more about what they are building and how well it will satisfy needs.

Yet, many teams don't make decision making explicit and transparent. Identifying *what* decisions need to be made, *who* will make them, and *how* they will be made is essential for effective teams. You can obtain clarity by having decision rules and a decision-making process.

A *decision rule* is a rule that spells out how the project team will make a decision. The decision leader is the person with authority to implement (or obtain resources for) the decision, as well as the responsibility to ensure that the decision is supported in the organization. This decision leader chooses the decision rule, and the team then uses a quick, transparent decision–making process. This involves consulting with stakeholders who can provide informed and valid input about any requirements decision.

For scope decisions, the decision leader is often the product owner (product manager, business sponsor, or business project manager). When requirements changes reach beyond scope-level decisions, the decision leader may also be a business manager closer to the content who can also balance known project constraints. Sometimes a decision leader delegates decision-making authority.

I recommend that the decision leader be as high as necessary and as low as possible in the organization structure.

Keeping, Not Creeping, Scope

Scoping your product with a vision and scope models early in your project helps you establish shared understanding among the delivery team and customer. Yet, changes in requirements are inevitable. Finding practical ways to recognize and address them takes collaboration and transparency. Using clear prioritization criteria, transparent decision making, and simple yet timely mechanisms for responding to changing requirements will help you build the right product and keep the scope right.

About the Author

Ellen Gottesdiener, principal consultant of EBG Consulting, helps teams to collaboratively explore requirements, shape their development processes, and plan and review their work. Her book Requirements by Collaboration: Workshops for Defining Needs (Addison-Wesley, 2002) describes how to use multiple models to elicit requirements in collaborative workshops. Ellen's most recent book, The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements (GOAL/QPC, 2005) is becoming the "go-to" industry guide for requirements good practices for business owners and analysts. She is a Certified Professional Facilitator and a member of IEEE, IIBA, IAF, ACM, and DAMA. Visit EBG's Web site for more articles, resources, to register for her free eNewsletter "Success with Requirements," and more. You can contact Ellen at: ellen@ebgconsulting.com.