A View To Agile Requirements

By Ellen Gottesdiener

EBG Consulting, Inc.: www.ebgconsulting.com

This article was first published in the November, 2009 Issue of *Modern Analyst*

The patterns, workshop design guidelines, and collaboration techniques I describe in *Requirements by Collaboration* are agnostic to product or software development method. When I wrote the book, agile methods were beginning to gain in popularity. Fellow agilists tell me they find the book useful when they design and facilitate workshops for their agile teams.

Yet some agilists ask me, "We have to constantly figure out our requirements—what to build, and when. How do I adapt requirements and requirements workshops to fit my agile project?"

They're right about the need to constantly adapt requirements and elicitation practices. In agile projects, you deliver the product in a series of successive and sensibly staged *releases.*[1] Each release represents the culmination of a series of requirements decisions. Working on cross-functional teams, you plan and deliver working software in a series of fixed time frames, with "business value" as your mantra. Your team has a ferocious focus on sculpting portions of the product from a mass of backlog possibilities.[2]

One of your biggest challenges is ongoing—how to group and sequence requirements for optimal delivery. Let's take a look at adapting requirements workshops to meet that challenge.

The Timing, Rhythms, and Patterns of an Agile Cycle

---

[1] A *release* is a shippable product version; it is ready to be deployed to the customer. Time frames for a release vary but typically fall every one to four months.

[2] The *backlog* is a catalog of prioritized and sized sets of work to be performed, including requirements as well as work needed to implement the product. Backlog items may vary in their degree of clarity. Some are defined in more detail than others.

Agile development and delivery rely on many short cycles with a clear cadence: *iterations* occur within multiple releases as you deliver the product successively. Each cycle—product, release, and iteration—necessitates its own view of the requirements.

Furthermore, it's wise *not* to conduct highly detailed planning at the start. Instead, you allow for the possibility that you might not build the entire envisioned product. You also take into account that the sequence, timing, and specifics of delivering requirements can change.

Thus, time drives requirements decisions, and time drives how you structure your agile requirements workshops.

Agile product development takes on a pattern of plan-do-inspect-act:

- **Plan:** define the product, release, or iteration.
- **Do:** define, build, and test working software.
- **Inspect:** review and retrospect your process.
- **Act:** adapt team practices, prune and prioritize the product backlog's items, and prepare for the next planning cycle.

You want to learn from each product cycle delivery and, based on what you learn, adjust your plans for the next cycle. These best practices optimize both your product investment and your development process. To this end, you punctuate the cadence with a starting ritual (planning) and ending rituals (reviews and retrospectives).

You do your planning, reviewing, and retrospecting in collaborative workshops. These workshops use group interaction rituals led by a skilled facilitator to deliver agreed-upon outcomes that feed the team's work. (For insight into those rituals and facilitation techniques, see Derby and Larson, Gottesdiener and Tabaka in Additional Reading.)

This article focuses on planning, when you explore and confirm requirements. The two key approaches I recommend my agilist clients are (1) to deliver requirements in collaborative workshops and (2) to organize agile requirements planning according to three product views.

Successful agile teams use requirements workshops to collaborate on the key planning cycles, holding a series of integrated workshops to deliver requirements. You hold collaborative requirements workshops for

- The entire *product* (once)
- Each *release* (once for each release)
- Each *iteration* within a release

Requirements by View

On an agile project, you don't attempt to understand or predict all product requirements up front. But you do need to sketch out the long view of the product to establish a common focus and marshal organizational resources (people, money, space, governance). From that vantage point, you define what to build in each release, and then in each iteration.[3]

These three levels—product, release, and iteration—correspond to three views (what I call the Big-View, the Pre-View, and the Now-View) of the product requirements, as shown in Table 1.

Table 1
Views of Agile Requirements

Requirements View	Purpose
Big-View (Product)	Gain an overall understanding of what the product will be, and plan the sequence of delivery. Agree on vision, scope, and time line for the entire product.
Pre-View (Release)	Define what product functionality to deliver in a given release, and obtain agreement on the backlog items to deliver in the first few iterations in the release.
Now-View (Iteration)	Identify enough requirements to deliver in an iteration to enable the team to make a commitment for delivery.

Let's review each view.

[3] An *iteration,* or *sprint* in Scrum vernacular, is a single development cycle, usually one to four weeks, resulting in a set of working software. Each iteration encapsulates a cycle of requirements exploration, design, coding, and testing.

The Big-View

The *Big-View* provides a holistic, evolving *product roadmap* of all the product requirements you will deliver over time to satisfy the product vision.[1] Because the requirements represented in your product roadmap are high-level, the roadmap shows *features:* "cohesive bundles of externally visible functionality that should align with business goals and objectives. Each feature is a logically related grouping of functional and nonfunctional requirements described in broad stokes."[2] The roadmap shows features arrayed across time and releases. The features in the roadmap are used to build or extend you product backlog.

The roadmap enables senior managers to plan their sponsorship of the development effort and establish governance and feedback loops for their funding decisions.

The product roadmap is an evolving framework, because you want the flexibility to end the product development journey before the originally envisioned development time frame. You might achieve sufficient value sooner than planned, or you might learn that you haven't gained the expected value and so it's best to cut your losses and kill the project. Because the product's payoff is uncertain, your plan needs the flexibility to change the routes at any time based on user feedback and market conditions. The roadmap has ongoing value as a tool for planning, requirements discovery and communication among product stakeholders as enhancements and extensions are added to the product until its retirement or replacement.

Types of Product Roadmaps

Product roadmaps can take on different perspectives, communicating different things to different audiences. For example, you might have a time-based roadmap that shows the functionality you plan to release at certain key milestones, industry occurrences, or market events. Another version of your roadmap might illustrate key market-worthy features or themes. If you are building commercial software, you will want both an internal and an external roadmap. Your internal roadmap supports funding and reveals the features you will use to compete in the market. Your external roadmap (which contains less detail) is helpful for engaging your customers, promoting market innovation, or shaking up the marketplace.

Pre-View

Whereas the product roadmap (Big-View) provides guideposts for the features to be delivered, the *Pre-View* reveals details at the next level: the release. Each release should contain cohesive chunks of consumable, marketable, valuable features.

To develop the Pre-View, business and technical stakeholders explore and define possible delivery sequences and dependencies for a release. A good release plan reflects analysis of functional as well as nonfunctional requirements (the quality attributes, design and implementation constraints, and external interfaces that the product must have).[3] This analysis exposes risks and unknowns about the product that need to be investigated, allows you to tune and prune your product backlog, and helps you focus on the backlog items that need the most attention in the next few iterations.

Now-View

The *Now-View* defines the requirements for a time slice within a release: the next iteration. This provides the iteration's requirements in sufficient detail that your team can make reasonable estimates of the work needed to deliver those requirements to a predefined set of acceptance criteria (conditions of satisfaction, or "doneness").

Together, these three views help you form a coherent, focused, flexible plan for agile development.

A Strategic and Tactical Product Owner (aka Customer)

On agile projects, team members need to make tough decisions about requirements throughout development.

You work with your product owner to answer questions such as, Is this requirement necessary? Is it more important than another? Can we deliver part of a requirement and still get value? Do these requirements go together? How can we group requirements to best serve our end users, or to deliver optimal value and return on investment? Will building one requirement (or set of requirements) before another cause too much rework later?

These are critical questions, and large agile projects need more than one product owner to answer them. It's almost impossible for a single person to handle the Big-View, Pre-View, and Now-View day in and day out. Consider that the "complete" product owner must do all this:

- Manage the product backlog.
  - Prioritize and prune a large backlog (or set of backlogs).
  - Participate in planning meetings at all levels.
  - Attend the demonstration/review and retrospective.
- Provide requirements expertise to the team.
  - Answer requirements questions about the current iteration.

  - Attend daily stand-ups.
  - Develop (or assist in developing) user acceptance tests for the current and next iteration (or provide details so that others can develop them).
- Participate in formulating business strategy.
  - Conduct market and competitive analysis.
  - Assist with marketing the product to the user community.
- Prepare for deployment.
  - Determine how to market to, train, and communicate with the customer.
  - Determine how the organization or customer will manage change.

One way to handle this overload is to split the responsibilities among two roles, having a strategic product owner and a tactical product owner. The strategic product owner owns the Big-View and the Pre-View, and the tactical product owner owns and maintains the Now-View.

The strategic, or Big-View, product owner . . .

- Is external facing
- Handles product marketing and portfolio analysis
- Is responsible for the product vision, product roadmap, "voice of the (end) customer" (balanced with the "voice of the business")
- Defines and adjusts the roadmap and release plans in response to market needs

The tactical, or Now-View, product owner . . .

- Selects and negotiates backlog items to deliver for each iteration
- Elaborates on requirements for backlog items (just-in-time)
- Defines dependencies, balances value with risk
- Defines criteria of acceptance ("doneness") for each item

The Long View

Collaborative requirements workshops are a useful way to plan and manage the changing and changeable requirements of your agile project. By organizing your tasks and conceptualizing around the three views discussed here, you can learn how to manage and improve your agile process and maximize agile's business value for your organization.

References

EBG Consulting, "Agile Requirements: Collaborating to Define and Confirm Needs," Course Guide, Version 09.08, 2009

Gottesdiener, Ellen. *The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements*. GOAL/QPC, 2005.

Ibid.

Additional Reading

Cleland-Huang, Jane, and Mark Dean. *Software by Number: Low-Risk, High-Return Development.* Prentice Hall, 2003.

Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.

Cohn, Mike. *Agile Estimating and Planning*. Addison-Wesley, 2006.

Derby, Esther, and Diana Larsen. *Agile Retrospectives: Making Good Teams Great.* Pragmatic Bookshelf, 2006.

Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley, 2005

Gottesdiener, Ellen. "Agile Requirements, in Context," *Success with Requirements,* Vol. 2, No. 1 (2008). Available at http://www.ebgconsulting.com/newsarchive.php?pid=agile-business-analysis-jan08#agile

Gottesdiener, Ellen. "Requirements Practices on Agile Projects," *Success with Requirements,* Vol. 1, No. 8 (2007). Available at http://www.ebgconsulting.com/newsarchive.php?pid=agile-business-analysis-sep07#3-mainarticle

Hohmann, Luke, Steve Johnson, and Rich Mironov. *Living in an Agile World: The Role of Product Management When Development Goes Agile.* Kindle Book available on Amazon.com.

Smits, Hubert. "Five Levels of Agile Planning: From Enterprise Vision to Team Stand-Up," Rally Software Whitepaper, October 2006, available at rallydev.com.

Tabaka, Jean. *Collaboration Explained: Facilitation Skills for Software Project Leaders.* Addison-Wesley, 2006.

Lawley, Brian. *Expert Product Management: Advanced Techniques, Tips & Strategies for Product Marketing & Product Management.* Happy About, 2007.

Comprehensive Agile References

Author

Ellen Gottesdiener, Principal Consultant, EBG Consulting, helps you get the right requirements so your projects start smart and deliver the right product at the right time. Ellen's company provides high-value training, facilitation, and consulting services to agile and traditional teams. An agile coach and trainer with a passion about agile requirements, she works with large, complex products and helps teams elicit just enough requirements to achieve iteration and product goals.

Ellen's book *Requirements by Collaboration: Workshops for Defining Needs* describes how to use multiple models to elicit requirements in collaborative workshops. Her most recent book, *The Software Requirements Memory Jogger* is the "go-to" industry guide for requirements good practices. In addition to providing training and consulting services and coaching agile teams, Ellen speaks at and advises for industry conferences, writes articles, and serves on the Expert Review Board of the International Institute of Business Analysis (IIBA) Business Analysis Body of Knowledge™ (BABOK™).

You can subscribe to EBG Consulting's offers a free monthly eNewsletter "*Success with Requirements*" offering practical guidance and requirements-related news. When you sign up, you'll receive a free article on essentials for scoping your requirements. You can follow Ellen on Twitter or contact her via email (ellen@ebgconsulting.com).