Chapter 11
Role of the Product Owner

"What's in a name? That which we call a rose, by any other name would smell as sweet."

-- Romeo and Juliet (II, ii, 1-2)

Chapter 11 Table of Contents

Is This a New Role?

In Chapter 1, we described the evolution of agile methods over the last decade or so, noting that, when it come to market share at least, the market is currently dominated by Scrum, followed by a Scrum/XP hybrid, then XP, then custom methods, agile unified process, etc. We are also seeing a growing influence of lean-agile hybrids, and lean-kanban implementations. So, while the market is maturing, agile methods continue to flourish. That has both its positives (advancing practices, new methods, increasing scale) and negatives (differing practices hinder standardized adoption) for the industry.

In all these methods, as well as in traditional requirements practices, it has always been clear that *someone*, or some small *group of someones*, must have a definitive say as to what the relative priorities for the solution requirements are. This focuses the team on the highest value work, and minimizes thrashing, wherein team members receive conflicting inputs from multiple sources[1]. In that case, they can't possibly satisfy all the stakeholders, so frustration and dissatisfaction is a likely outcome.

In Scrum, the responsibility for these activities fall on the role of the *Product Owner* and these responsibilities are fairly well articulated in Scrum books and trainings. However, we also recognize that there have been, and still are, various other titles ascribed to this role. For example, the role has been ascribed as a *product champion* [Leffingwell, 2000&2003 and Shalloway 2009]. In XP, the role is the responsibility of the *on-site customer*. In IT shops, it's typically the *business owner* that has those responsibilities.

The Product Owner title and responsibilities, as largely defined by Scrum, is emerging as a default standard for the role and function in most agile implementations. We feel this is a healthy trend in the industry as it simplifies agile adoption. Also, the availability of specialty training for that role improves the individual's and team's skills in prioritizing and elaborating requirements. In turn, this empowers teams to take stronger control of their destiny, and ultimately increases the velocity of the team by accelerating local decision-making, which otherwise slows down value delivery. In other words, the emergence and codification of this key role helps products deliver better software, faster.

However, as agile scales to larger programs and to the enterprise, there are some challenges with the role as defined in Scrum, and some modifications are typically necessary.

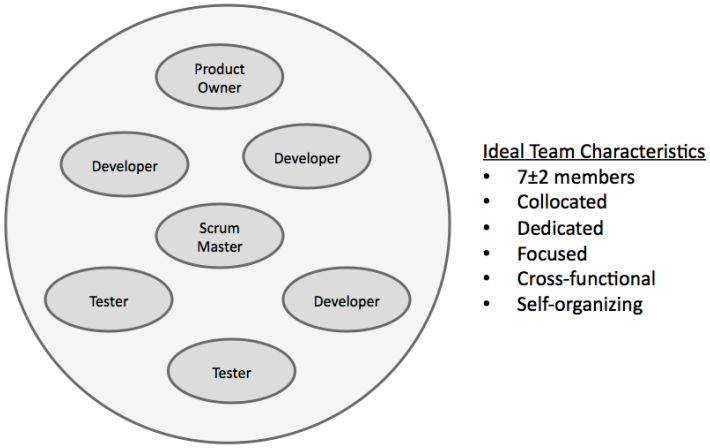
[1] In one pre-agile implementation, we asked a developer where they got their input for what to do. The answer was: project manager, tech lead, system architect, engineering manager, product manager and program manager. Six sources-one developer. What's wrong with this picture?

Perspectives on the Dual Roles of Product Owner and Product Manager

As we described, the Scrum Product Owner

> *"is responsible for representing the interests of everyone with a stake in the resulting project ...achieves initial and ongoing funding by creating the initial requirements, return on investment objectives and release plans." [Schwaber 2007].*

But the Product Owner's responsibilities don't end with the broad statement above. At the same time, the Product Owner is a resident of the ideal Scrum team as Figure 11-1



illustrates.

Figure 11-1 An ideal Scrum team

It can be seen in the figure and as it follows in support of Agile Manifesto Principle #4 - (*Business people and developers must work together daily throughout the project*[2]) the Product Owner is ideally collocated with the team and participates *daily* with the team and its activities.

We also note the "7 ± 2 members" recommendation for the ideal team. Having experimented with larger teams, mostly unsuccessfully, we personally support this recommendation as well. Further, as taught and commonly practiced, the Scrum Product Owner has additional, tactical activities that directly contribute to the team's success. These include:

- Setting objectives for the Sprint (or iteration)
- Prioritizing and maintaining the backlog
- Participating in the Sprint planning meeting
- Elaborating stories on a just-in-time basis with the team
- Accepting stories into the baseline
- Accepting the Sprint

[2] Agile Manifesto, agilemanifesto.org/

- Driving release planning

In summary, there are two primary sets of responsibilities that can be implied from the above:

> Responsibility Set # 1 - The Product Owner sets the vision and product objectives, manages to ROI, defines pricing and licensing policies, and also works with marketing to position the product in the marketplace
>
> Responsibility Set #2 - The Product Owner is a member of the team and works daily with developers and testers to help the team meet its objectives.

Given these responsibilities, when Scrum is introduced into a larger enterprise context, there often occurs a *role and paradigm mismatch* between the Scrum teachings and the existing organization's structure. Specially, the enterprise is certain to already employ product managers or business analysts who have the requisite skills, training, and existing responsibilities for Responsibility #1, above:

- They work directly with customers; their responsibilities include product definition and their reward system may contain an ROI element.
- They are trained professionals[3]; they have experience in the broader domain of defining and launching successful products;
- They have extensive domain and marketing knowledge;
- They have influence and authority over what gets built and why.

This can create a significant conflict with a Scrum rollout, a conundrum that is now being addressed by both sides:

> **From the Scrum community**. As Scrum advances into enterprise settings, the Scrum community is expanding its view of Scrum to include more of the outbound nature of the role, including market research, defining pricing and licensing policies, etc. This can be seen in books such as Pichler's Agile Product Management with Scrum [Pichler 2010] and the newer Scrum Certified Product Owner course. There, we are also seeing the introduction of a new Scrum *product owner hierarchy*, including the newly invented roles of *Product Line Owner* and *Chief Product Owner*.
>
> As these new titles and their implied responsibilities impinge even further on already well-established roles and titles in the enterprise, it remains to be seen whether this will help or hurt agile adoption at large. In any case, the viewpoints are simultaneously *convergent* (general agreement on the activities necessary to build the agile enterprise) and *conflicting* (what individuals and departments have that responsibility) at the same time.
>
> **From the professional software product management community**. To those in the ISV product management and product marketing community, this looks like a new version of "barbarians at the gate" - nothing less than a land grab whereby the development community is attempting to extend their fingers into areas where

[3] See for example Pragmatic Marketing Institute: www.pragmaticmarketing.com

they lack distinctive competence, appropriate background, and training. To them, Scrum is a process, built by developers, for developers, but who says that processes and those new roles will now be used to drive product and market policy? In their view, that's what product managers do now, and have always done, so why would they let the software development types take assume their responsibilities. In one pointed article[4], for example, Rich Mironov comments:

Product Managers are responsible for the overall market success of their products, not just delivery of software. In the Agile world, a new title is emerging-the Product Owner-which covers a small subset of the Product Management role. While this makes sense for internal IT groups that have traditionally gone without Product Management, ...agile product companies need full-fledged Product Managers to drive strategic activities and manage organizational/external participation.

Given this dichotomy, the enterprise has appears to have two choices when Scrum is introduced, neither of which actually work very well.

a) product management will pick up the additional, tactical responsibilities of the Scrum Product Owner and become Product Owners as well, or
b) individuals from the development teams will be trained to become Product Owners, only to discover that they may have an overlapping responsibility with the people who currently fill this role, who are in another department. To exacerbate matters, if more extensive Scrum practices emerge, development may likely create the new roles and responsibilities for *product line owner* and *chief product owner*, and the political fur will really begin to fly.

Choice #1 – When product managers assume the roles of Product Owners, there are issues galore:

- It doesn't scale. There may be a significant number of agile teams who now require this intense, daily tactical support and there are typically not nearly enough product managers to go around[5]. And they were quite busy *before* agile came to their enterprise. Now there is far more work to do. Current Scrum guidance is "one team-one Product Owner" [Cohn 2009], so where would all these new Product Owners come from?
- Even if you had enough product managers to fill the roles, they may be ill-suited, ill-inclined, and downright uninterested in these increasingly technical and development team-bound responsibilities. (If they had wanted to live with development, they would already have figured out a way to do so). For those with a thicker skin, see the footnote from

[4] www.enthiosys.com/insights-tools/pm-prod-owner/

[5] In one project Leffingwell coached, prior to agile, there were six product managers supporting approximately 250 developers, which became about 30 agile teams. That didn't work very well to begin with. Then, when one of the most talented product managers went on maternity leave, *it didn't work at all*.

the *Cranky Product Manager* blog below[6], but remember, we did say "cranky".

- Product managers often have insufficient technical depth and interests to add significant value to the team's highly technical language, activities and responsibilities. They simply may not be either respected or effective in the role.

In summary, Choice #1 doesn't paint a very pretty picture. So, let's try choice #2.

Choice #2 - Newly provisioned team-based Product Owners assume some of the role and responsibilities of the product managers. After all, that's what they were taught and if they are empowered to make decisions on behalf of the team, don't they have to drive the vision, and do all those other things necessary for product success? Again, issues galore:

- There is now an overlapping set of responsibilities, including the most important one, "who decides what the product is supposed to do?" What a fun place to have disagreement in the agile enterprise - the place where people decide what we are supposed to build! Worse, this conflict can occur in an area of historical dysfunction (see last section), the relationship of product management to development.
- Team based Product Owners are unlikely to be trained or skilled in all the other aspects of the traditional product manager or business analyst role. What makes us think they would be very good at it? If they wanted to live with marketing and customers, rather than development, wouldn't they likely already be in another role?

Choice #3. Dual agile roles.

Fortunately, there is another choice. As experience has shown that neither of these approaches is particularly effective, many enterprises take a more refined approach, one that supports *dual roles* of agile product manager *and* agile product owner:

- The more ***market (customer) facing*** product managers continue in their role along with most of their existing responsibilities, but they also evolve a far more agile set of practices, including *taking on a tighter relationship with the development teams.* That is the topic of Chapter 14, *Role of the Product Manager.*
- The ***product (technology) facing*** product owner role is assumed by either

[6] [Some] argue that in Scrum the product manager is the same as the Product Owner, and therefore the Cranky Product Manager needs to be constantly available to the team in order to make on-the-spot decisions within minutes of the asking. Ergo, you demand the Cranky Product Manager sit in that sticky-note-encrusted, windowless tomb with you all … day. Uh, no way. Not gonna happen. Why not? Because the Cranky Product Manager needs to be the Voice of the Customer and the Voice of the Market. How is she to do that without actually VISITING some customers and prospects? And VISITING means that she actually needs to leave the office, hop on airplanes, and fly far, far away." crankypm.com/category/agile-scrum/

the more technically inclined product managers or business analysts, or by development team members who are interested in that new role; they assume the agile team product owner responsibilities but also *take on a tighter relationship with Product Management.* This approach empowers the development team to take additional control of their destiny and also provides a new career path for those team members who would like to "grab hold of the steering wheel" in the increasingly agile enterprise. That is the topic of this chapter.

We strongly advocate for Choice #3, though it diverges from some current Scrum philosophies. We believe that this puts the right people in the right roles – team-based product owners that work their wonders with the technology – market-based product managers that work their wonders in the market, and it does so with minimum disruption to the enterprise's existing organizational structure. After all, with all we have to change to improve the value stream, why change anything we don't have to?

Important Note: Throughout this book, we'll operate under the *Choice 3, dual role* assumption, and we'll use the generic term *product owner* for the development team-based role, and ascribe a subset of the Scrum Product Owner responsibilities to it.

We'll also use the generic, traditional role and title of *product manager* (you can substitute business analyst for IT shops), but we'll also describe how this traditional role has to assume a new set of responsibilities to truly enable the agile enterprise. This is the subject of Chapter 14.

However, no matter the choice of labels, enterprises are free to assign the people they feel will be most effective in fulfilling the responsibilities, and call them whatever makes the most sense in their context, *so long as they do the work that is necessary to accelerate the value stream*.

The Name Game—Experimenting with the Product Owner Role/Title

Given this bit of confusion, it doesn't usually take very long for the loaded role/title and implied responsibilities of a new "product owner" to portent a major crisis in the prospective agile enterprise receiving Scrum training. Indeed, it can even be such a sensitive matter that the agile adoption can be seriously endangered by the potential conflict in role, title, and responsibilities. Serious caution is warranted.

One way to address this problem is by changing the title of the person assuming Responsibility #2. For example, the *role* of agile product owner may simply be assumed by the existing role and title of the *business systems analyst*. In other cases, the historical title of *requirements analyst* was assigned to the role. In another case, a new title/role of *requirements architect* was invented, primarily to avoid conflict with existing titles role and responsibilities. In still other cases, a small product owner *team* is created.

Of course, none of these titles fit every context perfectly and, in most enterprises, changing the title of product owner is more trouble than it's worth, since it isn't reflected in industry literature or on the standard trainings available. For example, Keith Black, VP Product Development, at TradeStation Technologies, described the problem this way:

> *"We've clearly found that the role of PO is NOT a PM. This has led to difficulty in us even describing the position in help wanted ads. If you run an ad for Product*

Owner, you get responses from people that want to be a Business Owner. If you run the ad for PMs, you get traditional PMs that are not technical. We could call the role "Analyst" or "Requirements Analyst", but that attracts more of a Systems Analyst profile. And given the fact that there's a void in the PO role in the Scrum world, it creates a situation where no one is out there who understands what the title means."

Responsibilities of the Product Owner and Product Manager in the Big Picture

In the context of this book, we'll assume that one of two cases exists:

1) The project scope is small enough that the two roles are combined; someone assumes the responsibilities of both the product owner and the product manager, as we describe them.
2) The enterprise is large enough that the roles are split per Choice #3.

In this latter case, we suggest that the following table is a reasonable division of responsibilities for the dual roles to support the agile development process:

Agile Product Owner	Agile Product Manager
Product/technology facing	Market/customer facing
Collocated & reports into development/technology	Collocated & reports into marketing/business.
Focuses on product and implementation technology	Focuses on market segments, portfolio, ROI
Owns the Implementation	Owns the Vision and Roadmap
Drives the Iterations	Drives the Release

Table 10- 1 Product manager and product owner roles and responsibilities

The agile product manager assumes most of the outbound and ROI responsibilities; the agile product owner assumes the inbound, product, or technology facing responsibilities. In this Chapter, we'll describe the responsibilities and activities of our more generic *product owner* (as we define it within the enterprise context). In Chapter 14, we'll do the same for the broader, product manager role. (Interestingly, it works out that no matter which case you pick, you still have to read both chapters!)

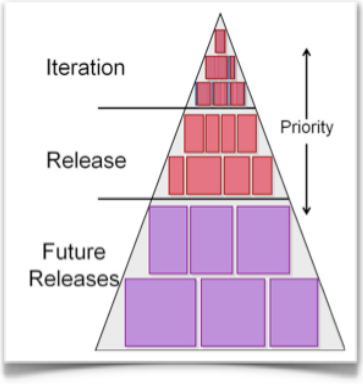
Responsibilities of the Product Owner in the Enterprise

Within this context, the responsibilities of the product owner can be divided into five primary areas:

- Managing the backlog
- Just-in-time story elaboration
- Driving the iteration
- Co-planning the release
- Collaborating with Product Management

We'll describe each of these in the sections below.

Managing the Backlog



In Chapter XX, we described the project backlog as the primary organizing technique for all the to-do work for the team. As it is the primary artifact that helps the team control and prioritize their work, it's hard to overstate the value of this simple construct. While the backlog primarily contains user stories, it also contains other work that the team needs to do to complete the current release as well as build architectural runway for future releases.

Building the Backlog

With input from the identified project stakeholders (Chapter 7), the product owner has the primary responsibility to build, prune, and maintain the backlog. Since teams use the

backlog to capture and manage all their work, any team member can put something in the backlog, so the resultant backlog is likely to include a number of different types of things:

User Stories. The primary content of the backlog is the user stories that have been defined to deliver value in the next iteration or release. As such, the user story content constitutes the product definition, and it can be sourced a number of ways.

- Teams may work directly with stakeholders to understand solution requirements. There are a variety of discovery and elicitation techniques that product owners and product managers use to determine these requirements; that is the subject of the next Chapter.
- User stories will be defined during release planning. Features are presented at release planning boundaries; teams decompose features into the various stories they will need so that the feature, in aggregate, can be implemented.

Defects. Defects are generally obvious to the team, they may have found the defects themselves in their testing, or they may have been reported from support or from the customers. They are referenced in the backlog so as to be prioritized with all the other work.

Refactors and Technical debt. Agile teams typically keep a list of refactors (rework) items that they need to do in order to either improve the quality of the existing product or to build architectural infrastructure necessary to support upcoming user stories.

Infrastructure work. The teams are also typically responsible for building the infrastructure they need to successfully build code in a timebox. This can include ongoing investments in continuous integration and build environment, project management tooling, test automation, and the like. Since it nibbles at the team's capacity, this type of work must be visible to the product owner and other interested stakeholders.

Prioritizing the Backlog

In general, the topic of prioritizing the backlog is always a tricky one, as many non-quantitative and (many times, arguable) factors potentially enter into determining priorities. These can include the *user value* of a thing, the *penalty for not doing* the thing, *risk of doing* the thing, *cost of doing* the thing, and *potential financial return* for doing the thing. So when it comes to prioritizing the program-level *features* that drive the project teams and ultimately, determines the overall fitness for use of the solution, we'll need to do a pretty thorough job of describing a process that can be used to prioritize features. We'll cover that topic later in Chapter 15.

However, in the context of this chapter, and in the context of a product owner facing a near-term iteration start boundary, the problem of prioritizing stories for an iteration is a lot less daunting. This is because there is so much current context to help in the decision-making:

- The goals for the next iteration should be clear (from the release plan)
- The results of the last iteration are known

- The backlog contains things that, while they may be unlike, are roughly the same size, (i.e. small enough to fit into an iteration). We don't have to prioritize huge things against small things.
- The scope of the prioritization problem is limited to just the next iteration; the team doesn't have to be right forever, just right for the next iteration.

In other words, the team has excellent, localized, current context in which to make priority decisions. The product owner can probably set these priorities fairly easily with knowledge of this context and a vision of where the team needs to be a few iterations later.

For example, as is illustrated in the Tendril example to the right, we see some defects at a high priority, user stories ranked on a relative basis, and then more defects, etc, further down the backlog list. In this context, teams rarely need a weighting system to determine the priorities. In this context, a simple discussion with the teams will likely create an agreement on priorities, and if not, the product owner makes the call.

All	Rank ▲	ID	Name
⊞ ⊟		#	
	0.001	DE2763	Unable to register IHD with new 3.5 stack against tree broadband
	0.001	DE2760	No Price displayed on Backhaul IHD
	0.001	DE2765	Cancel All on the DRLC Portal Manage Events page does not cancel any event
	0.001	US3093	SW - Tunnel True-Up to IHD via Backhaul
⊞	0.01	US2929	Backhaul consumer portal reflects device registration status
	0.06	US2971	[SPIKE breakdown on BH] User turns on previously provisioned device - DR
	0.07	US2972	SW - User views Price on PCT
	0.07	US2921	DRLC operator views network IDs of ingested DRLC users in DR portal network ID filter
⊞	0.5	US3088	Update Android to 1.6
	0.99	DE2753	DRLC Portal - Reporting - Criteria and GUI Report (GMT-6), download Report (GMT -5)

A Richer Scheme for Prioritizing the Backlog

However, when the situation is a little more complex and the decisions are not so obvious, we suggest a simple rating system which rates each backlog story on three factors:

- **Independent user value.** This attribute rates the value of the story to the user, independently of any other factors. In other words, how much benefit would this particular story deliver to a user, relative to other stories in the backlog? Based on domain knowledge, and the fact that the rating is only relative to other near-term stories, the product owner can usually just set this.
- **Iteration value.** While the story will have its own independent value, another consideration is at work: how the story will help the team meet its objectives for the iteration. Because program commitments are dependent on individual teams meeting their iteration objectives, this value is important to the team and the larger program. For example, if some other team has a critical dependency on a story, even when that story may not otherwise be particularly important to the team that is implementing it; then it would rank much higher.
- **Information discovery value**. The final factor to consider is the value of information discovery. This factor gives the team a way to put a value on gaining knowledge that reduces risk of implementing stories that are going to be necessary further down the road.

With this simple system, total value of a backlog items is simple the sum of the three as Table 10- 2 shows (scale is 1-9).

Unsorted				
Backlog Item		Value		
	Independent	*Iteration*	*Information discovery*	*Total*
Story US 17	*6*	*4*	*2*	*12*
Story US 32	*8*	*2*	*1*	*11*
Spike S43	*2*	*5*	*7*	*14*
Defect DE311	*6*	*2*	*1*	*9*
Story US53	*5*	*5*	*6*	*16*
Sorted				
Backlog Item		*Value*		
	Independent	*Iteration*	*Information discovery*	*Total*
Story US53	*5*	*5*	*6*	*16*
Spike S43	*2*	*5*	*7*	*14*
Story US 17	*6*	*4*	*2*	*12*
Story US 32	*8*	*2*	*1*	*11*
Defect DE311	*6*	*2*	*1*	*9*

Table 10- 2 Rated, unsorted, and sorted backlog items

From the table, we can see that story US53, which was lowest on the ordinal list, became the highest ranked priority story for the next iteration. Spike S43 was elevated to the second position, based on the high value the team placed on information discovery.

If the team so desires, these three values could also have weightings assigned, based on the project and release context.

Just-in-time Story Elaboration

In the context of an iteration timebox with a set of story priorities and an iteration objective, professional developers and testers take their objectives seriously, and they do not like it when they fail to meet them. In practice, countless iteration retrospectives have surfaced the common feedback:

> *"We failed to deliver these stories that weren't understood before we committed to it.*[7]*"*

Therefore, once the priorities are clear, it's likely that some of the stories in the backlog are going to need additional work before the team is comfortable putting them in the iteration. In

[7] We once had the experience of working with a proud and capable developer, new to agile, who takes his personal commitments seriously. In one iteration, he wrote a detailed, *four-page narrative* of how an apparently simple story blew up on him to take most of an entire iteration. His point was "I couldn't possibly have anticipated this, how am I supposed to estimate anything….."

other words, it's time for the *conversation* and *confirmation* parts of our Card, Conversation, and Confirmation metaphor.

From a timing perspective, there are three opportunities to do this:

During prior iterations. Once, a story has reached the point where it is likely to be implemented in a near-term iteration, the product owner can elaborate the story by doing appropriate research, talking to the team, users, product managers, or other stakeholders. Since the story is very likely to be worked on, this does not create any waste of unnecessary or too-early investment in specificity.

In Chapter 9, we also described the regular *iteration preview meeting*, wherein the product owner discusses stories that are anticipated for upcoming iterations. This gives the team a chance to discuss specific upcoming stories as well as to take a break from the "tyranny of the urgent iteration" and a chance to think about future work. In addition, a better understanding of *future* stories gives the developers the ability to implement *existing* stories in a more synergistic way.

During the iteration planning meeting. The next opportunity is during the iteration planning meeting. In this meeting, the team can take the time necessary to discuss the story itself and to understand the acceptance criteria. Some teams spend significant time in this meeting in order to do this (though a few hour timebox is recommended). If a team cannot be comfortable enough to commit to the story before the meeting is over, then there are two options: 1) postpone the story and have the product owner take the action to clarify the story before the next iteration, or 2) put in a spike to "figure out the story" in the current iteration, so that the story can be implemented in a later iteration.

During the iteration itself. Lastly, if the team is sufficiently confident that the story can be implemented within the iteration, or if the story is a relatively lower priority, then the conversation can happen during the iteration itself. Of course, at that point, the risk of not completing the story is material; however that is often an acceptable case.

In any case, how the team addresses story maturity on a just-in-time basis depends on the risk of the particular story and how the team addresses risk, as can be seen in Figure 11-2.

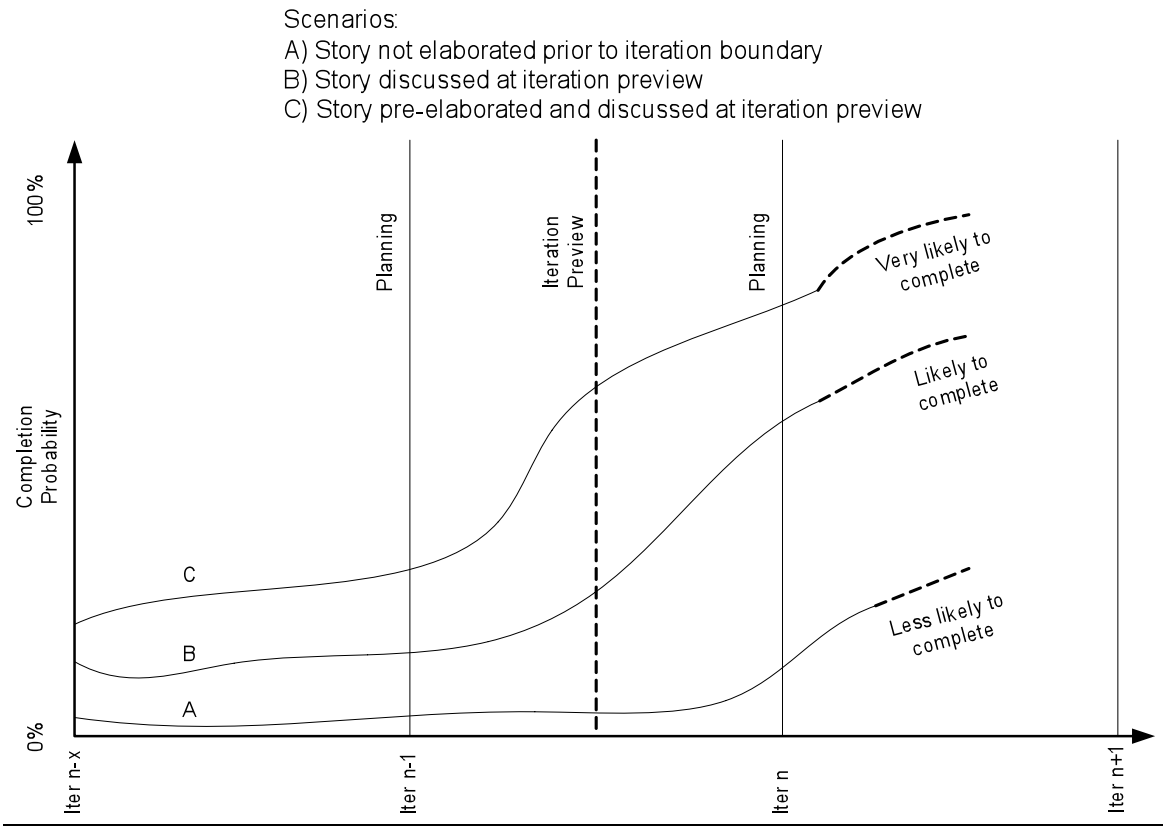


Figure 11-2 Story maturity as a function of elaboration

You can see from this figure that the probability of completing a story is simply proportional to how well the story is understood by the team prior to including the story in an iteration commitment. However, the story can't be too well elaborated prior to meeting a likely iteration boundary, since the team can't be certain it will actually be implemented as defined. Instead, it could be "trumped" by a higher priority story, or the nature of the story itself may have evolved after it was elaborated, which creates waste. Just-in-time is just agile common sense.

Driving the Iteration

In Chapter 9, we described the iteration itself at length. Since every iteration delivers some incremental value, and value is in the eye of the beholder (in this case, the product owner), the product owner will be integrally involved in this process.

In this chapter, we'll discuss the product owner's role in helping the team "deliver the goods". For context, the basic iteration pattern is repeated in Figure 11-3.

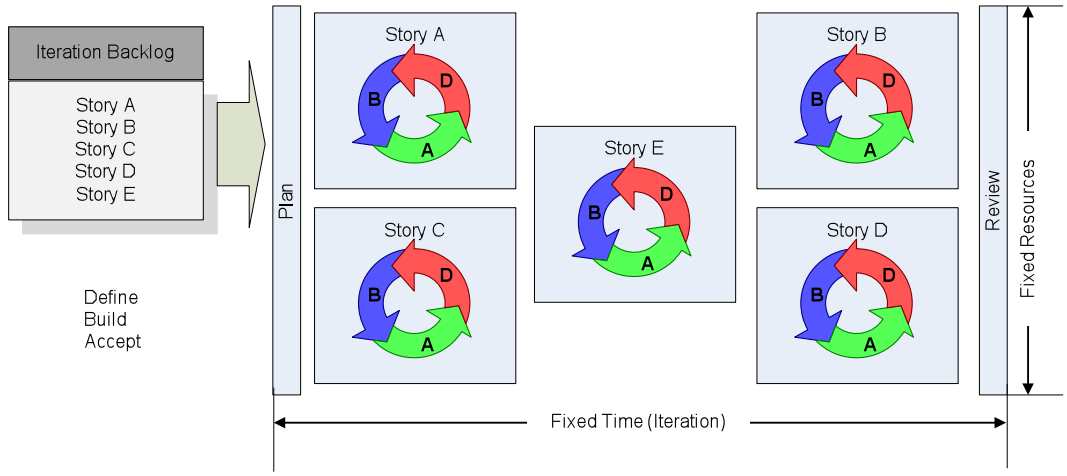


Figure 11-3 Basic iteration pattern

Preparation for Iteration Planning

The planning meeting is an important ceremony for the team. Given that the objective is to *agree on the content for the upcoming iteration*, some product owner preparation is likely to be required:

- Further elaborate higher priority stories as necessary
- Prepare a draft objective.
- Coordinate any common objectives and dependencies with other product owners and product managers.
- Review and reprioritize the backlog. This includes stories that (a) were already in the backlog; (b) failed acceptance in the current iteration; (c) are generated from defects or bugs.
- Consider necessary refactorings, defects, constraints, and dependencies.
- Understand the team's velocity for the upcoming iteration.

With this preparation in hand, the product owner is ready to participate in an intense, productive, and timeboxed planning meeting.

The Planning Meeting

The product owner begins the meeting by reviewing the objective for the iteration. Thereafter, the process is as follows:

- Product owner presents a backlog item for discussion
- The team discusses each item until it is well enough understood for the development team to detail and estimate engineering tasks.

This process is repeated for each story on the backlog until the team runs out of capacity. At that point, the team reviews the stories against the objectives and revises the objectives or the stories as necessary. Some stories may be dropped; other stories may be added. Therefore, the final, agreed-to scope of the iteration is the typically the result of some negotiation between the product owner and the development team.

Iteration Commitment

The end result of an iteration planning meeting is an iteration plan that contains:

- A committed iteration objective.
- A prioritized list of stories with estimated tasks and owners.

In any case, however, the product owner's primary role and goal is to *help position the development team for success in the iteration*. For if they fail, *they fail together*.

Executing the Iteration

Thereafter, the primary responsibility for successfully executing ("landing") the iteration lies with the development team. The team members deliver the stories to the code baseline in *priority* order:

Define - elaborate the story and its acceptance test.

Build - build the code and the test.

Test - get the code to pass the test and ready for final acceptance

Accept – as soon as a story is ready, it should be reviewed by the product owner. If accepted, the story is *done*. If not, the story definition, code, or acceptance test must be revised and then the story is presented again.

Note: It is important to do this serially; otherwise, the entire acceptance process is deferred until the end of the iteration. The result is that the team may tend to "waterfall" the iteration, leaving all stories for acceptance on the last day, and nasty surprises are a likely result.

This cycle repeats within the iteration until the end of the timebox, with an objective of getting all stories completed and accepted. During this time, the primary responsibility for completing all the stories rests with the development team members. As a member of the team, the product owner has a critical daily role as well:

- Work with developers and testers to elaborate each story
- Re-scope where necessary to better meet the iteration objectives
- Attend the daily stand-ups
- Review stories that are ready for acceptance
- Accept those stories that pass the acceptance criteria

Iteration Review

At the end of the iteration, a demo of the working, integrated, software is held for all interested stakeholders. The format is as follows:

- Presentation of each story by the responsible party.
- Discussion and feedback with stakeholders.

Based on stakeholder review and team feedback, the product owner may move the story to "accepted state" (if it wasn't already) or leave the story in the backlog if incomplete.

At the end of the review process, the product owner reviews the objectives of the iteration and decides whether to accept the iteration or not based on how well the team (inclusive of the product owner) did against the stated objectives.

Retrospective

The final activity is the retrospective, where the team takes the time to reflect and assess on the results and then adapts the development process accordingly. The product owner participates, just like every other team member.

A Product Owner's Iteration Calendar

Taken together, the activities and meeting commitments can fill up a product owner's daily diary pretty well, as Figure 11-4 indicates.

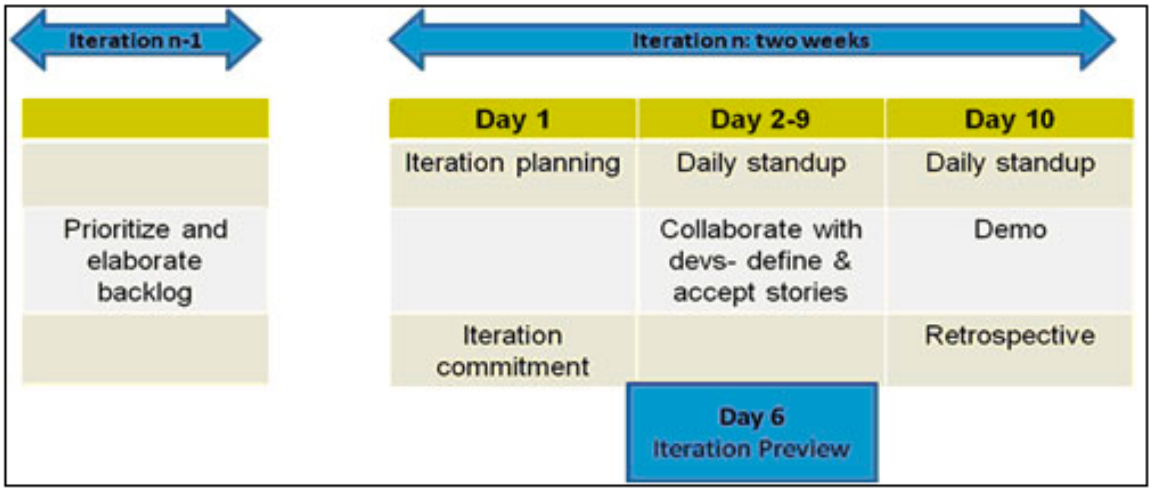
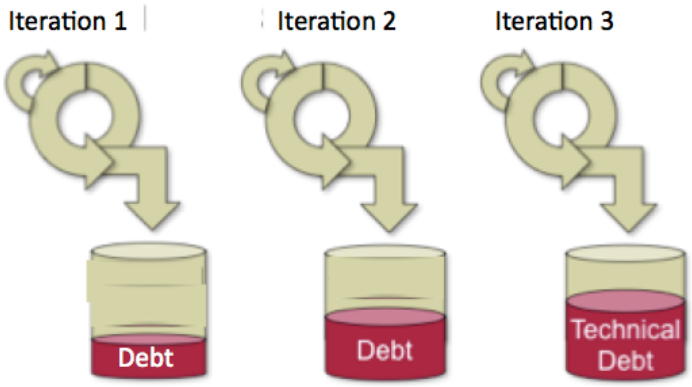


Figure 11-4 *A product owner's schedule*

The Problem of Technical Debt and the Value Stream

Agile is an empowering software development model and it often creates a sense of excitement and reward for completing work for the product owner and users. However, this energy level – coupled with an "insane focus on value delivery" - sometimes becomes dysfunctional and counterproductive. That can occur as the team feels the pressure to continually meet their commitments and in doing so, they may begin to let slip, or fail to improve, their quality engineering practices. This is the "tyranny of the urgent, iteration style".

In the process, the team creates what agile describes as *technical debt.* For example, some of that brand new code may not have automated test coverage or adequate documentation. Perhaps it has more complexity than it needs because the team couldn't afford the time to clean it up and refactor it.



Now it's going to be harder for someone else to work on it, because it's inadequately documented and complex and it's even hard to refactor it without introducing bugs because you don't have the automated tests to help you know if you covered everything!

Other examples include not refactoring code for new stories, leaving defects in the system, taking shortcuts, not unit testing, not automating tests, not writing code to standards, etc. Worse, despite the team's hard efforts, bugs could actually be increasing, too. Systemically, the development process has become less lean as more time is spent on overhead and waste.

Analogy: Credit Card Debt

Technical debt works like credit card debt and its compound interest. If you keep racking it up, it could eventually cause the project to default. Because the team is now working slower, the team may be falling farther and farther behind and not being able to finish everything by the end of the iteration; they rack up even more technical debt! Product owners may see this because the team seems to be getting less done per unit time than at the beginning of the project, when they had fewer people.

Paying Down the Debt

Unfortunately, getting out of technical debt is similar to getting out of credit card debt, and is psychologically just as hard. The first step is cut up the credit cards – *simply refuse to accrue any more technical debt from this time forward*. Instead, demo only things that meet the *real definition of done*, even if this means the team can't claim credit for everything they thought they would. This takes a certain degree of courage, and is probably something to do right after a major release, rather than just in front of one.

What's a Product Owner to Do?

Since the product owner sets the priorities, they must be part of the solution. When a team pushes back on getting stories done "right", the product owner has to listen and adapt. After all, velocity may now be decreasing – so it is in the product owner's best interests to take corrective action and start allowing additional investment in defects, infrastructure, refactors, etc.

Each time the team says "we could do it the fast way, or we could do it the right way", listen. Discuss what it means in terms of building more technical debt. A general guideline is that a team should be able to devote *as much as 15%* of its capacity to work that accelerates future, rather than current, velocity.

Co-Planning the Release

Of course, the iterations serve a larger purpose - frequent, reliable, and continuous *release* of value-added software to the customer or marketplace. Pictorially, this is seen in the Big Picture as the Release (or PSI - Potentially Shippable Increment) boundaries in Figure 11-5 indicate.

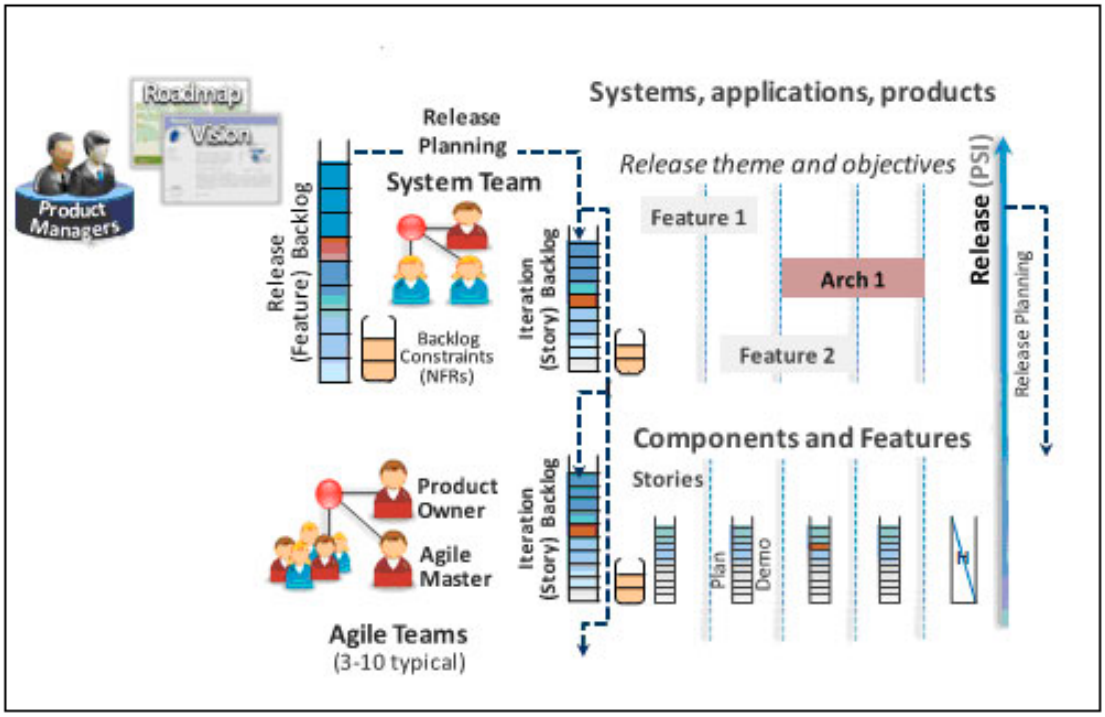


Figure 11-5 *The objective is to "Release"*

As we will discuss in Chapter 15, *Release Planning* is the seminal enterprise event that regularly aligns the teams to a common vision and release objective and the product owner plays an active role:

- Update the team's local backlog.
- Meet with other product owners to understand overall system status.
- Meet with product managers to understand the vision for the upcoming release.
- Brief the team on the upcoming release objectives.

During the release planning event, the product owner will typically:

- Help identify, prioritize and estimate stories that will be necessary to achieve the release objectives.
- Help design the release plan by laying the stories into iterations.
- Participate in the team's discussion, impediment and risk identification.
- Identify and coordinate dependencies to ensure a cohesive solution
- Participate in refining the release objectives and making a commitment to the release.

Once the release plan is committed, the product owner assumes the responsibility with the team to deliver on each iteration's objectives. If all teams meet their objectives, then all is well and the release will go out on time. However, it isn't quite that simple, but that is the topic of Chapter 15- Release Planning and Execution

Five Essential Attributes of a Good Product Owner

In this Chapter, we've seen that the product owner, whether as applied in Scrum per se, or as we have applied it more generically, is a key figure in our agile process. While the ultimate velocity of the team depends on many factors, the ability for the team to quickly determine what the system is supposed to do hinges on this important role. Given that importance[8], we'll take a moment to characterize the key skills and attributes of someone who is likely to be successful in this role.

Communication skills. The product owner is the "glue" that binds the product management function and all the other project stakeholders to the development team. Doing so requires good communication skills as the product owner translates user and business objectives into the level of detail suitable for implementation. Moreover, the product owner will almost certainly be involved in customer demonstrations, sales support, and other outbound activities, so some customer communication skills are beneficial.

Good business sense. Agile's focus on value delivery also demands that product owners have working knowledge of the business domain. In this way, the product owner can better understand and define user stories that deliver real value to the end users and establish priorities and appropriate tradeoffs for system functionality and performance. In addition, they have their own business' best interests at heart, and with this knowledge, they can make decisions that balance the customer's and business' needs.

Technical foundation. Effective scope triage requires the constant evaluation of technical, functional, performance, and user-value tradeoffs. In turn, this requires a degree of technical competence, as the foundation for effective decision-making is an understanding of the implementation technology. In addition, the ability to intelligently prioritize refactors, defects, and technical debt versus new value stories requires both an understanding of the technology as well as empathy and respect for the technical competence of the team.

Decisive. At the pace of agile development, *no* decision is worse than any other kind. The product owner must be able to make decisions, every day, in the presence of far-from-perfect knowledge. This requires empowerment, courage, and the ability to admit when one is wrong. This is agile after all - you'll get another whack at it if you need it.

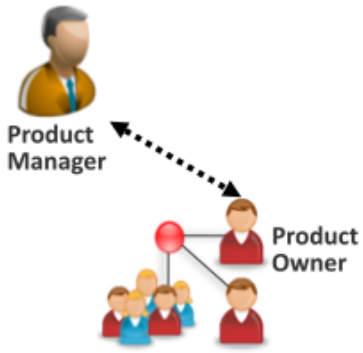
Trustworthy. Since the primary responsibility for prioritizing and managing the backlog (i.e., what *will* and *will not* be done) falls to this role, the most essential attribute of the

[8] In [Cohn 2010] (which we just received), Mike describes a similar set of five attributes as "ABCDE", Available, Business-savvy, Communicative, Decisive, and Empowered. We've assumed *availability* and *empowerment* throughout, by virtue of the one product owner-one team design.

product owner is *trustworthiness*. The teams have to *trust* the product owner to make the hard calls on scope triage and to defend their interest in quality *and* functionality; the product managers have to *trust* the product owner to faithfully represent their feature priorities to the teams. The keys to building trust include in this role include transparency, honesty, meeting commitments, and admitting when you are wrong.

Collaboration with Product Managers

As we've described above, while the release planning event is one structured and routine collaboration opportunity, enterprise agility is most effective when product owners have a far more closely coupled relationship with product management. To help address this problem, we often recommend that product owners "report on a fat dotted line" into the product management team, even if they are in the development line organization.



From a line-management perspective, product owners

- are collocated with the team.
- share managers, incentives, and culture with development.
- should be rewarded based on how the team as a *whole* performs.

But they are also honorary members of the product management organization. Here they:

- receive overall product direction.
- attend most relevant PM meetings, functions, and planning sessions.
- receive input with respect to career growth and performance.

In fact, product owners live in *two* teams, the development team and the extended product management team. Neither team is less or more important than the other. And for either to succeed, they must both succeed, so there can be little or no political infighting within the system. It's not an easy job.

Making this work fairly seamlessly creates its own set of challenges but is a worthy endeavor for the enterprise. As Jennifer Fawcett[9] notes:

> *Creating the Ultimate Product Team does not come without emotional challenges of adopting, coaching, and nurturing this high-performing team. Past processes, roles, and behaviors do not change overnight.*

We'll discuss that at length in Chapter 14 – Role of the Product Manager.

[9] www.agileproductowner.com

Seeding the Product Owner Role in the Enterprise

As we have seen, an agile transformation is no small feat for a single team, a larger program, or an entire enterprise. Since the product owner is the linchpin to a timely flow of value, putting competent people in the role is one key to enterprise success.

In the smaller project context, finding someone for the role is usually not too hard – indeed, someone has probably already been playing the role; management and the team will likely recognize them fairly quickly. Throw in a little training and coaching and you should be well on your way.

However, in the larger organization, finding and growing some number of individuals (5-10-20, even up to 100 in the largest cases) to assume this key, but previously underserved and undeclared, role is a challenge into itself. Moreover, since the enterprise has probably grown into various silos, the solution requirements have been "handed off" from product management to development and there may be no natural interpreter on each team to naturally gravitate to the role. But find them we must, for if the product owners aren't there - the requirements don't flow - if the requirements don't flow - the enterprise can't be agile.

In the sections below, we'll highlight some real-world large-scale transformations, and illustrate how these enterprises found the product owners *to get those agile requirements flowing*.

TradeStation Technologies. TradeStation is a premier brokerage-trading platform for rule-based securities trading. At TradeStation, Keith Black, John Bartleman, and their teams have been driving a comprehensive, all-in agile transformation that affects 100+ practitioners. John described their approach to filling the product owner role as follows:

> *"Before transitioning to Agile, our Product Management team was made up of ten product managers who reported into development. In general, we have always had an internal/technical focus as opposed to the traditional external/marketing focus. When we transitioned to Agile, seven of the ten Product Managers became full time Product Owners; the other three now focus on the market-facing Product Manager role. This separation of labor and concerns has helped us bring additional focus to both the market and technical aspects of our solution."*

John then comments on the staffing challenge:

> *"When staffing the Product Owner role, I would have preferred to use a few lead developers and/or testers since they have the domain knowledge and technical expertise; however, we are reluctant to do this because of the impact on development resources. Therefore, we hired a few additional Product Owners from outside the company. These people need to be technical, but also need to have good industry specific experience, and that is a difficult combination to find. So far, former developers/tech leads with business sense and good project management skills seem to be the best fit…. in my view at least, technical skills are mandatory, and domain experience is a plus whenever I can get it."*

CSG Systems. CSG Systems is a customer interaction management company that provides software- and services-based solutions that help clients engage and transact with

their customers. In 2007, CSG began transforming their ACPx product development (>100 practitioners) efforts using enterprise agility best practices. Mauricio Zamora, whose team is responsible for defining the ACPx vision, roadmap and agile processes leveraged to deliver the vision, noted that they established product owners through a series of phases:

> *"We first had to leverage a combination of product analysts originally responsible for waterfall requirements, architects originally responsible for designing our software and a few Product Owners already experienced in agile execution. Over the course of a few releases, we used the really good Product Owners to set the standard for others. In the process, we discovered candidate Product Owners that weren't a good fit for the role fit, moved them to other slots, and replaced them with more appropriate internal resources and a few additional external hires."*

Mauricio went on to note that the transformation wasn't easy and it took time to help people see all that needed to change.

> *We first educated everyone on the differences between the traditional Product Management, agile Product Owner and architect roles. We had to convince management that the Product Owner role required dedicated focus. The visibility agile provides made the increasingly obvious gaps in Product Ownership easier to see and address. Finally, we had to revisit and revise organizational titles and compensation because the new Product Owner role didn't map well into our existing organization."*

BMC Software - Business Performance Manager Project. BMC Software is a leading provider of IT Infrastructure Management Solutions. In 2006 and 2007, the Infrastructure Management Group (IMG) transformed their development organization (>250 practitioners) using agile development practices to deliver "a major product to the market in less time and with higher quality than previously possible" Israel Gat, then with BMC and now a Cutter Consortium consultant, led the transition. Recently, he described how they filled the product owner role (which they re-titled as "requirements architects") as follows:

> *Between the product managers and the requirements architects we had enough "hands on deck". We must have needed half a dozen architects to become requirements architects, and we were able to generate them. We might have borrowed a little from the project leads, but the architects were our #1 pool.*

Symbian Software Limited. When it comes to embedded systems and an even *much* larger enterprise scale, consider Symbian Software (now a part of Nokia) develops and licenses Symbian OS, the market-leading open operating system for mobile phones. Symbian initiated an agile transformation in 2008 that will ultimately affect *thousands* of practitioners.

The development of a mobile phone operating system is a highly technical endeavor and one where the ultimate user (mobile device user) is fairly far removed from the major technologies (OS, device drivers, media players, etc.), which are the primary focus of the implementation. As such, the development process does not lend itself quite so easily to the traditional customer/user facing agile product manager/product owner roles.

Mathew Balchin was integrally involved in the design of the agile rollout model for Symbian. The scope of the challenge (i.e., finding *lots* of product owners) was daunting. Matthew described their approach this way:

> *We applied the product owner role pretty much out of the box (per Scrum training) and defined the interaction at the outset with our traditional product management functions. Senior stakeholders identified it as one of the pivotal roles for success early on and we also identified the need for role-based soft skills training in addition to the standard agile training. All our POs come from engineering teams and are senior engineers with product or customer experience. We have a one PO to one-two team mapping typically, rarely 3 teams.*

Discount Tire. Discount Tire is America's largest independent tire dealer. Chris Chapman, Application Services Manager, and his teams develop the internal software that keeps Discount Tires corporate and store operations (750+ stores in 21 states) running. In 2008, they implemented an agile transformation that affected the entire information systems team. Chris noted how they addressed the product owner challenge:

> *Our Business Systems Analysts in IT are filling the role of Product Owner. They are the liaison to the business and in many cases speak for the business. Their previous responsibility of documenting detailed business requirements and rules now falls to the entire team in the form of user stories and acceptance tests (which is still a major "cheese moving" event for us).*

Summary

In this rather lengthy chapter, we‘ve described the product owner, the individual that has the highest impact on the flow of value stream to and through an agile team. We’ve described how, at least on the context of the larger program or enterprise, the role of the product owner, as originally defined by Scrum, is more likely to be shared between some number of product owners and product managers, whose activities and behaviors will also have to adapt in the new agile paradigm.

We’ve described the responsibilities and activities of the product owner in driving the content and priorities of the iteration, and the role they play there in helping the team build small increments of value in a timebox. We’ve also described the problem of technical debt, and the role the product owner can play in increasing *or* decreasing it.

Since it such a critical role, we’ve also described the essential attributes of an individual who can effectively fill the role, and the part they play in building an effective product owner and product manager team. Finally, since the role is likely to be new to a team and enterprise, we’ve provided some case studies of how a number of agile enterprises have found the right people they need to fill the role.

In so doing, we’ve elaborated on the *how* - the activities and role the product owner plays in the agile process, but not the *what* – the requirements content that the product owner feeds into the system. In the next Chapter, *Requirements Discovery Toolkit*, we’ll describe a set of tools that the product owner and team can use to discover the *what*.

References

Cohn, Mike. (2010). *Succeeding with Agile: Software Development Using Scrum*. Reading: Addison-Wesley Professional.

Leffingwell, Dean. & Widrig, Don. (2000). *Managing Software Requirements*: *A Unified Approach*. Boston: Addison-Wesley.

Leffingwell, Dean. & Widrig, Don. (2003). *Managing Software Requirements: A Use-Case Approach*. Boston: Addison-Wesley

Pichler, Roman. (2010). *Agile Product Management with Scrum*. Boston: Addison-Wesley.

Schwaber, Ken. (2007). *The Enterprise and Scrum*. Redmond: Microsoft Press, 2007.

Shalloway, Alan et.al. (2009). *Lean-Agile Software Development: Achieving Enterprise Agility.* Reading: Addison-Wesley Professional.