# *Chapter 14*
# *Role of the Product Manager*

## *Chapter 14 Table of Contents*



## *Introduction*

In earlier chapters, we've described why, at least in the context of the mid- to larger-sized software enterprises, the responsibilities of the agile product owner (as primarily defined by Scrum) is really a set of responsibilities that are typically shared between a significant number of agile product owners and a smaller number of agile product managers. In

Chapter XX, we described the specific activities of the product owner role, which in the context of common agile rollouts, is a relatively new role which is tightly coupled to the team and the implementation. We also described how this modest separation of concerns can improve the value stream by avoiding bottlenecks in decision-making relative to priorities or product definition. In this Chapter, we'll describe the other half of that equation, which is the role of the product manager in the agile enterprise.

In the Big Picture, the product manager operates a little above the fray of the iterations and the tactical work the teams do to actually develop and deliver the code. Instead they operate primarily at the *Program* and *Release* level, where they focus on program vision, features, and releases, as Figure 14-1 indicates.

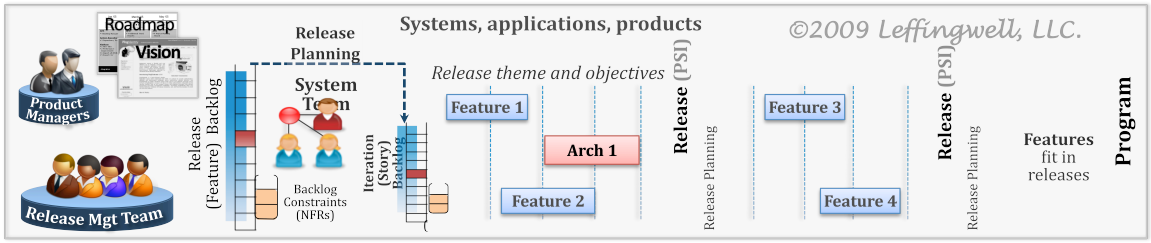



**Figure 14-1** Product Managers operate at the Program and Release level

## *Product Manager, Business Analyst?*

Before we proceed, however, we pause for a second to reflect on the title of the role. In different industries and organizations, the role can be responsible for a variety of activities, from strategic to tactical. Individuals with a variety of different titles, who may come for marketing, development or IT operations, can fulfill the role.

In addition, the titles associated with these responsibilities tend to differ based on industry segment.

- In *product oriented companies*, including Independent Software Vendors (ISVs) and product and system manufacturers, *Product Manager* is a common role and title which usually has a fairly clear set of responsibilities for defining and positioning the product in the marketplace. However, we've also seen other titles carry that same responsibility within product companies, including such titles as *Solutions Manager* and even *Program Manager.*
- In enterprises which have large *Information Systems/Information Technology* (IS/IT) responsibilities, the title is most typically *Business Analyst*, though the term *Business Owner* may also be used to describe this function.

These two different industry segments have different objectives, focus and activities, which we will highlight briefly.

## *Responsibilities of the Product Manager in a Product Company*

The Product Development and Management Association[1] has published a Body of Knowledge whose content suggests the responsibilities of those who perform the various functions of product management in a product-oriented company. The body of knowledge is organized into three *lifecycle phases* and *six knowledge* (competency) areas:

The lifecycle phases are:

> ***Discovery Phase*** *– Discovery covers the process of searching for and identifying opportunities - whether market-based or technology-based - and all of the planning and strategy to accomplish this. It requires the identification of customer needs, problems, and benefits, and the conceptual features that are envisioned for the products it wishes to build.*
>
> ***Development Phase*** *- The second phase is primarily about realization. It covers the process of converting specifications into designs - whether for an individual product or a complete portfolio of products - and all of the processes to accomplish this. It usually requires detailed resource management, creative engineering and process design capabilities, and sophisticated information technology. It ends when the products or services achieve their first commercial availability.*
>
> ***Commercialization Phase*** *- The third phase is primarily about fulfillment. It covers the entire process of new product introduction and the organization's management of its product and service portfolio as it attempts to fulfill its financial potential. It ends when the products or services have reached the end of their useful lifecycle and are to be considered as candidates for retirement, renewal, and regeneration. At this stage, the process begins anew with the undertaking of a new product development initiative and a return to the Discovery Phase."*

The six knowledge areas are:

> ***Customer & Market Research*** *- bringing external insight into product innovation, development, and growth especially insight about customers (buyers and end users) but also information about channels, competitors, markets, alternatives, etc.*
>
> ***Technology and Intellectual Property*** *- the invention, development, acquisition, licensing, and management of technologies and intellectual property that enable and become part of products.*
>
> ***Strategy, Planning & Decision Making*** *- strategies, plans, and decision-making around product innovation, development, and growth. These would include strategies, plans, and decision making at the business level (as relates to product*

[1] http://pdmabok.arcstone.com/description.php (Permission needed)

*innovation, development, and growth), as well as for platforms, product lines or product families, and products.*

***People, Teams, & Culture*** *- the people side of product development across the lifecycle including organization/team structures, people management, skills development, culture, organization change management, human interaction, etc.*

***Co-development & Alliances*** *- innovation, development, and growth activities that take place in unison with external partners, including customers, suppliers, service providers, and channels. This would include co-development or development chain strategy, partner management, co-development execution processes, co-development teams, etc.*

***Process, Execution, Metrics [&Financial]*** *- pricing, positioning, promotion, channel management, financial management, the customer support operational dimension of product innovation, development, and growth including: processes and tools for requirements development and management, design, manufacturing, supply chain, (engineering) and change management.*

While there are many viewpoints on the role of product management in a product company, we feel this body of knowledge is reflective of a typical, albeit generic, set of responsibilities for people who fill the role.

## *Business Responsibilities of the Role in the IT/IS Shop*

In the IS/IT shop, the focus is on helping the enterprise meet its business objectives through either

- development of new systems for internal use
- integrating internal development capabilities with acquired solutions, and
- installing and configuring commercial, off the shelf software.

The Business Analyst plays a product manager-like role in needs assessment, defining solutions, build vs. buy decisions, etc.

The International Institute of Business Analysts (IIBA) has developed a Guide to the Business Analysis Body of Knowledge (BABOK Guide)[2] to guide practitioners who fulfill this role. BABOK consists of six *knowledge areas* and eight underlying *competencies*. The six knowledge areas are the most relevant here, as they describe the activities of a Business Analyst:

***Business Analysis Planning and Monitoring*** *...covers how business analysts determine which activities are necessary in order to complete a business analysis effort. It covers identification of stakeholders, selection of business analysis techniques, the process that will be used to manage requirements, and how to assess the progress of the work.*

[2] IIBA Guide to the Business Analysis Body of Knowledge [source]

> ***Elicitation*** *describes how business analysts work with stakeholders to identify and understand their needs and concerns, and understand the environment in which they work. The purpose of elicitation is to ensure that a stakeholder's actual underlying needs are understood, rather than their stated or superficial desires.*
>
> ***Requirements Management and Communication*** *describes how business analysts manage conflicts, issues and changes in order to ensure that stakeholders and the project team remain in agreement on the solution scope, how requirements are communicated to stakeholders, and how knowledge gained by the business analyst is maintained for future use.*
>
> ***Enterprise Analysis*** *describes how business analysts identify a business need, refine and clarify the definition of that need, and define a solution scope that can feasibly be implemented by the business. This knowledge area describes problem definition and analysis, business case development, feasibility studies, and the definition of solution scope.*
>
> ***Requirements Analysis*** *describes how business analysts prioritize and progressively elaborate stakeholder and solution requirements in order to enable the project team to implement a solution that will meet the needs of the sponsoring organization and stakeholders. It involves analyzing stakeholder needs to define solutions that meet those needs, assessing the current state of the business to identify and recommend improvements, and the verification and validation of the resulting requirements.*
>
> ***Solution Assessment and Validation*** *describes how business analysts assess proposed solutions to determine which solution best fits the business need, identify gaps and shortcomings in solutions, and determine necessary workarounds or changes to the solution. It also describes how business analysts assess deployed solutions to see how well they met the original need so that the sponsoring organization can assess the performance and effectiveness of the sol*ution.

The underlying competencies describe the expectations for an individual's skills and abilities in: Analytical Thinking and Problem Solving, Behavioral Characteristics, Business Knowledge, Communication Skills, Interaction Skills and Software Applications.

## *From an Agile Perspective, the Role is Largely the Same.*

Fortunately, from the standpoint of the actual software development process, this is probably more than we need to know. However, from our perspective, the business analyst and the product manager (we'll simply use the generic term *product manager* from here forward) have the organizational responsibility to:

1. Understand stakeholders, needs, gaps, and opportunities
2. Define products and solutions to address those needs

3. Work within the organization to address all the other issues (internal and external) that are necessary for successful deployment, and finally:
4. *Work with the development team to communicate vision and features and help assure the solution evolves to meet the real needs of the stakeholders.*

## *What's So Different About Product Management in the Agile Enterprise?*

It is clear that the role of the product manager is a highly leveraged role in agile development and one that is necessary for enterprise success. Indeed, the criticality of the role is implied directly in Agile Manifesto Principle #4 – *Business people and developers must work together daily throughout the project*. But the question we must address here is not the role itself, but how the role *changes* when the team moves to an agile development paradigm.

While it would certainly simplify agile adoption if this role was largely unchanged, the reality is that the role, behaviors and activities of the people who fill that role undergo a substantial transformation in the agile development model, as we summarize in Table 15-1 below.

| PM Responsibility | Traditional | Agile |
|---|---|---|
| Understand customer need | Up front and discontinuous → | Constant interaction |
| Document requirements | Fully elaborated Documents → | Constant communication with team. |
| Scheduling | Plan a one time delivery, way later → | Continuous near term roadmap |
| Prioritize requirements | Not at all or one time only in PRD → | Reprioritize every release and iteration |
| Validate requirements | Not applicable. QA responsibility → | Involved with iterations and each release. Smaller, more frequent releases |
| Manage change | Prohibit change – weekly CCB meetings → | Adjust at every release and iteration boundary |
| Assess likelihood of release date | Milestone document review → | Release dates are fixed, reliable. Manage scope expectations |

**Table 15- 1** Product Manager's Changing Role in the Agile Enterprise

Adapting to all the behaviors in the right column in Table 15- 1 is a substantial change for the newly-agile, enterprise product manager so we'll discuss each of these briefly:

### Understand Customer Need

| | | |
|---|---|---|
| Understand customer need | Up front and discontinuous → | Constant interaction |
| | May be developed over many months. Perhaps annually.<br>Write it down. Hope for the best | Continuous and incremental discovery<br>Continuous communication to customer and team |

The traditional months and months spent up front to determine customer needs are largely eliminated as they cause big delays in the value stream. Instead, we start by implementing what we do know, and we evolve the system from there. Interactions with customers are continuous. Communication: customer – product manager – product owner/development team – replaces much of the documentation.

### Document Requirements

| | | |
|---|---|---|
| Document requirements | Fully elaborated in Documents → | Constant communication to team |

| | | |
|---|---|---|
| | Marketing requirements documents | |
| | Product requirement documents | Vision statements |
| | Sign offs, approvals and feature "freeze" | Release planning briefings |
| | Throw over the transom to development | Mockups, screen shots, videos, lightweight tools |

Traditional marketing requirements and product requirements documents are often eliminated, reduced in scope, or lightweight substitutes such as briefings, videos, mockups, etc. are used instead. Product managers communicate at the feature level, stating intent and avoiding unnecessary specificity (requirements). Intense communication to development teams occurs at iteration and release boundaries, coupled with constant, daily, communication with product owners.

**Scheduling**

| | | | |
|---|---|---|---|
| Scheduling | Plan a one time delivery, way later | → | Continuous near term roadmap |
| | Annual release schedule (at most). | | Date and quality fixed. Content variable. |
| | Date and feature fixed. Quality variable | | Rolling wave planning |
| | Hope they meet the date but doubt they will | | Potentially shippable increments every quarter. |

Scheduling is continuous; deliverable plans are updated on about 90 day boundaries. Quality is fixed and not a variable. Teams will always have a high confidence, current plan of intent for the next release increment. Longer-term roadmap deliverables are more vague.

**Prioritizing Requirements**

| | | | |
|---|---|---|---|
| Prioritize requirements | Not at all or one time only in PRD | → | Reprioritize every release and iteration |
| | All features created equal. | | Constant re-prioritization. Not "what is the highest priority", but "what do you want next" |
| | All "Must haves". | | |
| | No prioritization | | |

It has always been difficult to prioritize requirements in document form. Spreadsheets and requirements management tools work better mechanically, but even then, it can be difficult to get product managers to prioritize. Perhaps this is because we've taught them that what "prioritization" really means to developers is: "here's the stuff I really need you to do, and here's the stuff I invested all these words in that you probably will never do anyway". In agile, we break that scenario with the availability of near term incremental releases. It's far easier to get a product manager to commit to "what we should deliver next" than to "this is more important than that".

### Validating Requirements

| | | | |
|---|---|---|---|
| Validate requirements | Not applicable. QA responsibility | → | Involved with iterations and each release. Smaller, more frequent releases |
| | Handoff to development and QA. Assume system will meet requirements.<br>Pretend like they haven't changed in the year of development. | | See, validate, and adjust at every iteration. Reset priorities at quarterly release plan. |

Validating requirements was never a responsibility we ascribed to the product management role. Delivery was late anyway, and we came up short on requirements, so it was easy to hide behind what wasn't done whether it would have satisfied the customer or not. With more rapid delivery, and the availability of intermediate code increments that can be externally validated, product managers have the opportunity *and* responsibility to help the team *evolve* requirements to better meet the customer's need, no matter what we thought they were back when the project started.

### Managing Change

| | | | |
|---|---|---|---|
| Manage change | Prohibit change – or weekly CCB meetings | → | Adjust at every release and iteration boundary |
| | Manage and try to control change<br>Code and feature freeze<br>Defect triage as end game | | Embrace change<br>Code has quality<br>Can add new things up to last responsible moment |

In some ways, managing change was conceptually easier in the old days. Simply, we would freeze the requirements at some point and then minimize, or even eliminate, change in order to assure we could deliver something. Of course, the customer's needs were changing whether the code was changing or not, so value decayed while we rejected change. Now, we are engaged in a constant process of embracing change, refactoring code, and trusting our automated test assets to keep us out of trouble.

### Assessing Status

| | | | |
|---|---|---|---|
| Assess status and likelihood of release date | Milestone document review | → | See working code every iteration and every release |

| | |
|---|---|
| Attend milestone reviews.<br>Look at indirect artifacts.<br>Pretend the data presented is understandable and can actually help predict delivery<br>Look in crystal ball - what does 90% complete mean? | Attend iteration demos.<br>See iteration goal progress.<br>Evaluate working code in PSI.<br>Get interim customer feedback.<br>Understand velocity and feature status. |

Since product managers interface directly with customers, they are usually quite curious about delivery dates. While the agile truth is to "admit what we don't know", customers ask, even demand, to know what we don't know - and that puts the product manager in the hot seat. ("Hmmm… we don't know, but 'we don't know' is not an answer").

So if necessary, they might even make something up. Thereafter, they hope that developers will deliver *something a lot like they said*, *a lot like when they said we would*. Unfortunately, however, our historical means to know - milestone reviews, status reports, documents and the like - didn't really tell them much, except "no, it isn't done yet".

Now we have real working code, and they can now tell their customers "here's what's working now and here's what's going to be working in the next iteration (or PSI)". It still isn't perfect, and it isn't long range, but *it's better than what we had before.*

## *Responsibilities of the Agile Product Manager in the Enterprise*

This is a lot of change for a product management organization to address, just because *we* changed *our* development model. Fortunately, however, others have gone before us and the product management team can achieve most all of the change behaviors above by interacting with the agile teams in a number of fairly well defined patterns.

Specifically, in the agile enterprise, the product manager's role evolves to fulfill the following primary responsibilities:

1. Own the Vision and Release Backlog
2. Manage Release Content
3. Maintain the Product Roadmap
4. Build an Effective Product Manager/Product Owner Team

We'll define each of these responsibilities in the sections below.

## Own the Vision and Release Backlog

### *The Agile Vision*

Though the instantiation and delivery models (Vision vs. product requirements document) are different, the responsibility for owning the product or solution vision is not new to the role. After all, if the product manager:

- Has a continuous, in-depth understanding of the current solution
- Stays abreast of the latest industry trends
- Understands the changing needs of the market and the customer base
- Articulates a clear direction for addressing gaps and opportunities

Then building a *Vision*, which articulates a clear direction for addressing gaps and opportunities, is a logical outcome.

### *Communicating the Vision*

In agile, the traditional *product requirements documents (PRD), system specification, software requirements specifications (SRS)* and the like are typically eliminated entirely. In their place, agile enterprises take a leaner approach better suited to the last-responsible-moment, delayed decision making and artifact-light development practices of the agile enterprise. However, since the PRD and SRS documents no longer exist to specify system behavior, communicating the *Vision* to the agile development teams becomes even more critical, because the documents aren't likely to be around to do it for you.

Doing so is product management's responsibility, because no matter how empowered and energized the agile teams may have become, it is the product manager's responsibility to set strategic direction. The Vision answers the big questions for the *system*, *application,* or *product* under development, including:

- Where are we headed with this thing?
- What problem does it solve?
- What features and benefits does it provide?
- For whom does it provide it?
- What performance, reliability, etc. does it deliver?
- What platforms, standards, applications, etc. will it support?

### *An Agile Vision Can Take Many Forms*

Agile teams take a variety of approaches to communicating the Vision. These can include:

- Draft Press Release - highlighting the new features and benefits of the next release of the solution
- A "Very Preliminary Data Sheet" - which accomplishes the same thing in a concise, marketing presentation-like form
- "Product Vision Box" [Highsmith ref] - which uses the product packaging box metaphor to capture similar elements

- RUP-like Vision Document - which is a standardized template which defines users, features, system qualities and the like.

In the context of rolling wave release planning which we discuss in Chapter XX, these documents take on more of a "delta" function, in that each new vision statement need carry only what's *new*, and what's *different* about the upcoming release. This keeps them lightweight, digestible and team and stakeholder friendly,

### *It Doesn't Even Have To Be That Structured*

Each of these forms has proven their worth in a variety of agile projects, but it doesn't even have to be that well structured. In one release planning session, there wasn't an opportunity for the four product mangers to collaborate prior to the release planning session. Even if there was an opportunity, it's doubtful that they could have necessarily come up with a harmonized, force-ranked feature set anyway. (Question: which product manager wanted their number one feature to be placed fourth on the release backlog?) Instead, we allowed each product manager approximately 45 minutes to present. Each presented a briefing, including of the top ten new features proposed for their solution. Based on this context, the teams then went into the more detailed planning session.

Clearly, this was not the ideal forced-rank prioritization and it was left up to the teams to decide what to do about the fact that four product managers had separate priorities. But software, like life, can be a little messy. But it worked really well, in part because the product managers were part of the process, in part because they had visibility into what the teams could and could not achieve in the timebox, and in part because they were able to see and empathize with the other product manager's priorities.

### *The Primary Content of the Vision is a Set of Features*

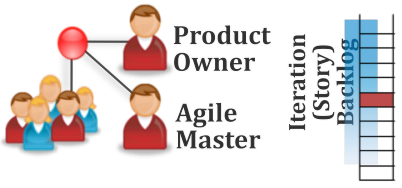
No matter the form, the main content of the vision document is a prioritized set of *features*. Features are high level system behaviors that can be described in a sentence or two and which are written so that customers can actually understand, debate, and prioritize them.

Of course, in so doing, we didn't invent either the word "Feature" or the usage of the word in that text. Rather, we simply fell back on industry standard norms to describe products in terms of, for example, *a Features and Benefits Matrix* that is used by product marketing to describe the capabilities and benefits provided by our new system. By applying this familiar construct in agile, we also bridge the language gap from the *agile project team/product owner* to the *system/program/product manager* level and give those who operate outside our agile teams a traditional label (Feature) to use to do their traditional work (i.e., describe the thing they'd like us to build).

We have also posited that by managing the level of abstraction, a system of arbitrary complexity (from the space shuttle to the spellchecker on this editor) can be described in a list of 25 or so features. That still works for agile teams describing a Vision today and a simple list of *features* can still be used as the primary placeholder for user value.

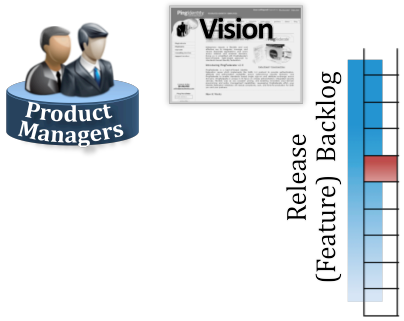
### *Undelivered Features Fill the Release Backlog*

In the Big Picture graphic and in Chapter XX (Role of the Product Owner), we noted that the primary currency



of requirements expressions for the agile teams is the user story, which is contained in the *Iteration (story) Backlog* (product).




In a like manner, the *Release Backlog* contains the prioritized set of features that have not yet been implemented. Like stories, features can be *scheduled* (in a release) or *unscheduled* (waiting for future attention). They are prioritized and estimated. Estimates at this scale are coarse grained and imprecise, which prevents any temptation to over-invest in feature elaboration and estimating. If and when a feature reaches a priority such that it hits a release planning boundary, it will be broken into user stories prior to implementation.

### *Nonfunctional Requirements*

In addition, the enterprise is too large to assume that all the global development teams will naturally understand the various constraints and "ilities", such as reliability, accuracy, performance, quality, etc, that are imposed on the system as a whole. Therefore, these *nonfunctional* requirements must also be known and communicated.(This is the topic of Chapter XX).

## Managing Release Content

In accordance with the Big Picture, the Vision for the product is delivered to the market in a stream of continuous and frequent, small releases (typically, every 60-120 days). Each release is defined by a fixed date, theme, planned feature set, and fixed quality requirements- *scope* is the variable. Product managers deliver the vision to the development teams face-to-face in the periodic Release Planning events.

### *Release Planning Event*

In the Big Picture, all agile teams who build cooperative subsystems operate on a synchronized *Agile Release Train* model. Periodic release planning is the seminal event that aligns the individual teams to the overall strategy of the enterprise. As such, the Release Planning event is to the enterprise what iteration planning is to the team, the fixed pacemaker that drives the enterprise's delivery cadence. As such, release planning is the focus of much preparation, communication, coordination, and commitment. Teams whose software must cooperate will have to plan together, and they do so face-to-face, at least so far as travel budgets will allow.

Since the release dates are fixed in advance on the Agile Release Train, release planning events are routine and can be pre-scheduled as far as a year in advance. During release planning, team members attend in person *if at all possible.* During the event:

- Product Management owns the feature priorities
- Development team/product owner owns story planning and high-level estimates

- Architects work as intermediaries for governance, interfaces and dependencies

### *Preparation for the Release Planning Event*

Since the Release Planning event is the primary communication and coordination point for product strategy for the upcoming release period, product managers should be come well prepared:

- Understand the status of the current (in process) release
- Update the release backlog
- Meet with other business owners and other Product managers to coordinate initiatives and priorities
- Meet with Product Owners and discuss the preliminary Vision for the upcoming release.

The event is typically a full day minimum, more likely two, and often follows a pattern as illustrated in the two figures below.

**Figure 14-2** Release Planning, typical day 1

Day 1 is focused on delivery of the Vision, which the product managers deliver via whatever medium suits them best, followed by initial planning by the teams.

**Figure 14-3** Release Planning, typical day 2

In most cases, the Vision doesn't "fit" in the release time frame provided, (after all, what self-respecting product manager would bring *less* vision than the teams could likely accomplish). Thus, some Day 2 scope management triage and out-of-box thinking is required. In any case, Day 2 should bring a commitment to the release objectives for the next release. Even then, the commitment has to be flexible (otherwise you have fixed time-fixed-scope-fixed quality; in other words a short waterfall-constrained release) and many agile teams communicate the feature release priorities on a flexible basis like this:

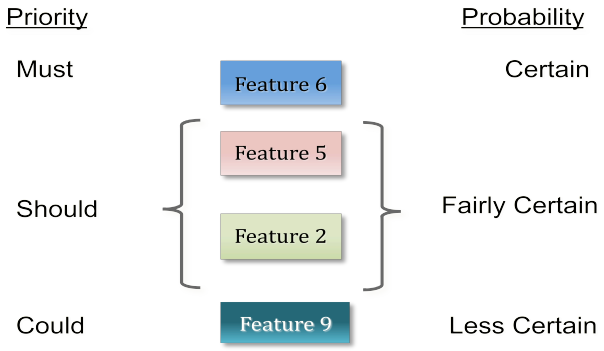



**Figure 14-4** Prioritized release features with probability

In this manner, the Product managers control the release content, as well as the delivery priorities. And since teams become far more reliable on their release commitments as they master the new agile paradigm, this gives the product managers and external stakeholders a more certain planning basis on which to make key decisions.

### *Release Messaging*

To be filled in later.

### *Tracking the Release*

All the hard work the enterprise has put into the agile transformation should now start paying dividends. Once a commitment is achieved, the primary responsibility for *delivering the release* in a series of short iterations resides within the project teams. After all, they write and test all the code, and in agile they are both *empowered* and *accountable* to achieve the agreed-to objectives.

Even then, however, the agile enterprise recognizes that continuous tradeoffs of changing requirements and scope is inevitable, so the product manager also plays a pivotal role in *helping the teams meet the release objectives*. Doing so requires ongoing tracking and managing of the release and its content. Fortunately, our agile enterprise model is replete with visibility, so understanding the actual vs. planned status of a release is no longer a crystal ball activity. Instead, there are four primary mechanisms, which help the enterprise and its key product manager stakeholders keep the agile release train on its rails.

#### *#1 - Constant Informal Communication with the Product Owners*

The product manager has a direct liaison to the teams via the team's product owners. The fan-out is not too extreme – a single product manager may typically

interact with some small number (3-6) of product owners– and thereby have ready status from all the project teams who must collaborate in meeting the release objectives

#2 - Participation in the Release Management Team

In the Scaling Software Agility book and blog, I've described a typical construct where development management, agile/ScrumMasters, product managers, quality assurance, release managers, and other key stakeholders meet weekly to assess the status of a release. This Release Management Team (or RMT) has the primary responsibility for shepherding the release to market. This weekly forum is a key touch point through which product managers can assess release status, and adjust scope where necessary.

*#3 - Attendance at iteration demos*

The inherent visibility of agile development also provides an inherent opportunity to *see the working code as it develops*. This happens in the context of the iteration (Sprint) review, which contains both a product demo and retrospective. The every-other-week product demo is key to the product manager's insights into status and progress and is a not-to-be missed opportunity to see the product, interact with the teams, provide feedback and make midcourse adjustments where necessary.

*#4- Status via agile project management tooling*

To have even reached this point, the enterprise will assuredly have rolled out appropriate agile project management tooling which provides support for the higher level status views needed by product managers and other stakeholders. This tooling should provide some form of hierarchical, feature-level burn down so that the product manager can assess, *on an aggregate basis*, exactly where they stand within the release as Figure 14-5 shows:

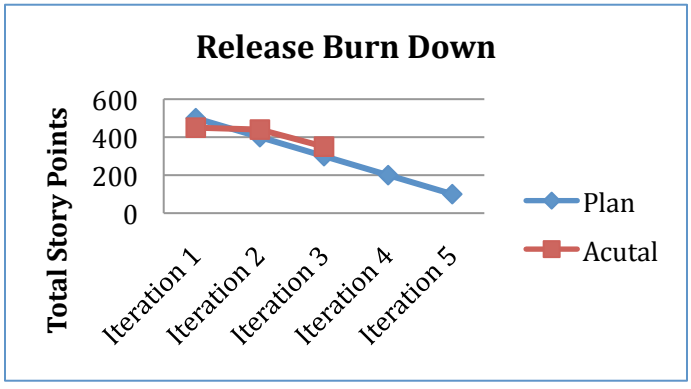



**Figure 14-5** Release level burn down

This chart provides a sense of the probability of landing the release. However, it doesn't give any notion of what features may or may not be delivered. For that, the tooling should also provide default reports on the status of each individual feature, perhaps something like Figure 14-6 below.

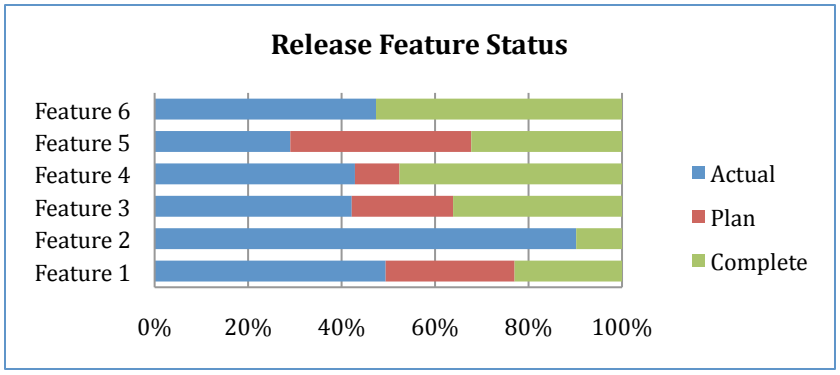



**Figure 14-6** Release status by feature

Together, this information should give the RMT and the product managers some objective knowledge of where they are, and even more importantly, what changes might be necessary to "land" the release on time.

## Maintaining the Roadmap

As we have described the Vision so far, it has been presented as time-independent. This is appropriate as the objective is to communicate the gestalt of "what this thing is we are about to build" and overloading the Vision with prospective timelines will likely quickly derail the discussion of the "what".

However, in order to set priorities and plan for implementation, we need an additional perspective, a product *Roadmap* that provides a view we can use to communicate *future* objectives to our outside stakeholders. An agile Roadmap is not a particularly complicated thing, nor is the mechanical maintenance of it difficult. For example, a typical Roadmap might be communicated in a single graphic as shown in Figure 14-7:


**Figure 14-7** A sample product roadmap

The Roadmap consists of a series of *planned release dates*, each of which has a *theme* and a prioritized *feature set*. While it is a simple thing *mechanically* to represent the Roadmap, *figuring out the content for anything beyond the next release is another matter entirely*. The topic of *what* else the team *plans to ship* and *when* can be a fascinating and contentious topic in agile. However, the easiest way to think about the Roadmap is that it is an *output*, rather than an *input* to the Release Planning process.

- The dates and themes for the next release are *fixed*. The features are *prioritized* and *variable*.
- The teams can commit only to the features in the next upcoming release. Releases beyond the next, represent only a best estimate.

The Roadmap, then, is a "plan of intent" and is subject to change as development facts, business context, and customer needs *change.* With respect to the upcoming release, perhaps the most important guidance is this:

> Even though the team has committed to the objectives and we have agreed that the feature set cannot be guaranteed, it is a reasonable expectation that the agile teams will:
>
> 1. meet the date
> 2. accomplish the theme
> 3. deliver most of the features, and certainly the highest priority ones, with the requisite quality.

Anything less would be unprofessional and belie the power, discipline, and accountability of our agile enterprise model. Moreover, it will eventually threaten our own empowerment, as failure to deliver will inevitably cause the implementation of various controls to "help us"!

## Building an Effective Product Manager/Product Owner Team

The "steering wheel" that guides the enterprise to its product outcomes is the relationship between the agile product owner and agile project manager. From a reporting standpoint, we often described the most typical relationship as a "fat dotted line".


Product manager typically report into marketing or business- product owners typically report into development. However, product owners are also honorary members of the product management organization from whence they receive overall product direction. They also:

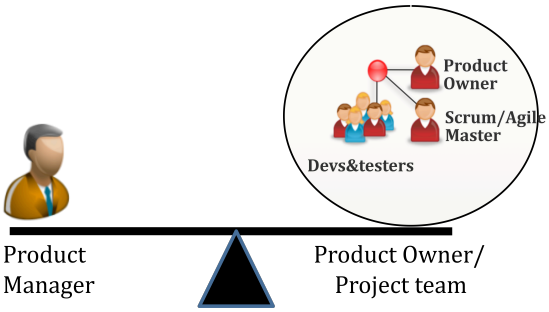
- Attend most relevant meetings, functions, and planning sessions.
- Receive input with respect to career growth and performance.

Of course, there is a natural tension in this relationship as the needs of the key constituents (development team vs. customers/market) are different. The product managers naturally want *more product-more quickly* and there is no upper limit to their demands; product owners and development teams naturally want that too, but are sensitized to the inevitable technology, architectural, and quality constraints endemic to every software application.

### *Needed – A Sense of Balance*

This natural tension is a direct delegation of the larger business-versus-technology enterprise balancing act.




If the business (product management) solely has its way, it's likely that expediency of value delivery will rule and technology will get the short stick. After all, what business owner would not want to accelerate value delivery at every opportunity?

If technology (product owner/team) solely has its way, there can be little doubt that the product will be built on sound (and the latest!) technology without shortcuts or quality compromises. But, the business value delivery may get the short stick. After all, what technologist wouldn't want to build the most extensible and reliable platform to support *future* customer needs?

So, the best we can do is first recognize and balance these competing interests and then occasionally let the balance tip a little this way or that (refactoring vs. new value stories) based on the current business context. Perhaps more importantly, it is in the heat of this natural friction and its constant resource constraints that the sparks of true innovation and creativity are born.

### *Essential Ingredients*

In a series of blog posts [3]Jennifer Fawcett has commented on how important it is to build an effective relationship amongst these teams. She noted that the essential ingredients for building the ultimate agile product owner/agile product manager team are *Collaboration*, *Partnership,* and *Trust*. Together, we provide a few tips for building these essential ingredients:

**Collaboration**

Synchronize and communicate the ever-changing corporate priorities daily. Invite product managers to attend daily stand-ups on occasion. Politely insist on attendance at most/all demos. Have an "open door" policy for all development and product management meetings. If one party cannot attend, summarize results in minutes or email.

**Partnership**

Ask each other "How can I help?" Create and participate in after-hour events to eliminate any us – vs. – them thinking. Prepare for release planning together. Operate under the "never surprise each other in front of others" rule.

**Trust**

Trust each other to make the right decisions. When (not if!) your product manager/product owner partner makes a decision that is opposite of yours, support that decision. Teams will know you are both empowered, will "cover each other's backside"

[3] www.agileproductowner.com

and they will also know that you can both be trusted with business decisions and authority.

## *Summary*

# Possible addendum

Ed note: this is an old blog post which could be a) integrated as is in the body, b) left as an addendum, c) key points made in the body without this entire post or d) not needed.

---

## *Phases of Product Management Disillusionment in Waterfall Development*

Prior to agile, many enterprises have followed a sequential, stage-gated, waterfall development model. In these cases, it's likely that the Product Manager's mindset has moved through a series of increasingly foreboding attitudes, as the figure below shows.

### Phase 1 – Unbridled Enthusiasm

In this phase, the Product Manager spends their time with customers, interviewing, running workshops, and using whatever other tools they have at their disposal to define and document the requirements for the prospective new system. These requirements are typically captured in a MRD (Marketing Requirements Document (MRD) or PRD (Product Requirements document). After that, the development team typically response with a Software Requirements Specification, or (SDS) System Design Specification, which further refines and documents the intent of the PRD.

At the end of this phase, which may take from 3-6 months to document and gain the requisite approvals, the development effort is launched and there is a *hand off to engineering*.

### Phase 2 – False Sense of Security

This initial, upbeat phase is followed by a period of relative calm. I call this the *period of a false sense of security*. During this phase, development proceeds apace. The product manager may be uninvolved, or may attend milestone reviews where models, documents and project plans are reviewed and inspected. Towards the end of this period, which lasts typically from 6-12 months, the software is declared to be 90% complete and launch plans are put into place. Customers are notified that the release is impending and external and internal commitments are solidified.

### Phase 3 – Rude Awakening

Of course as we've discussed in the book and blog, the next phase is a painful one indeed. During system integration, many defects are discovered, some design related (think – uh oh, lots of rework…) , and the dreaded defect triage process begins.

Now it is obvious that the schedule will not be met and communication of that prospect begins. Worse, in the early period, the defect "find rate" exceeds the "fix rate" and the schedule becomes *less and less certain* and product delivery looks *further and further out*. Even worse, the customer typically now has their first opportunity to see the actual

software and they discover that it is not what they currently need. This is because either a) they didn't know what they wanted back then, or b) their needs have changed in the last year. No matter the cause, substantive, additional rework will be required before it can be deployed.

This a period of substantial pain for the Product Manager and for all key stakeholders in the process.

### Phase 4 - Resetting Expectations

Fortunately, we didn't get this far in the tech industry without most program teams eventually figuring out how to deliver software, so a period of rework and recovery begins. During this period, the scope is typically slashed dramatically, and commitments for schedule and functionality are renegotiated. Of course, credibility has now been lost throughout the enterprise – the development team to its stakeholders – as well as the product managers to their external stakeholders.

### Phase 5 – Season of Perpetual Mistrust

What follows is a long, persistent period of mistrust between the development and product management organizations, which is often characterized by cross-department hostility, *reduced* communication and even complete dysfunction.

Worse, as the Product Mangers have learned that they only get about half what they ask for, they often resolve to ask for twice as much next time, to assure that they get something meaningful delivered. And we all know how well that works out. So the vicious cycle continues.

I describe all this, not to further belabor the pain, but simply to point out this may well be the environment when the agile transformation is initiated.

Clearly, this waterfall development model has not served the needs of the Product mangers or other stakeholders particularly well, so at some point, the lucky enterprise commits to a new, agile development paradigm. Often times, the development teams themselves sponsors the agile initiative and delivers a couple of messages to the Product Management organization:

Message 1 " We are heading down a new path, and this type of thing won't happen again"

Message 2 – "but in order for us to be successful, you'll need to change many of your behaviors, too".

And finally, message 3- "with our new agile model, we are going to stop predicting what will be delivered and when and we don't really need those Marketing Requirements Documents any more, and we also won't be creating those software design specs either."

And now finally, for those developers and managers leading an agile transformation, and in the context of the Phase of Persistent Mistrust, I ask you to perform the following thought experiment.

*If you were a Product Manager operating in this environment, how would you react to such a message?*

Clearly, we are going to have to approach this change with a more mature thought process and a credible strategy to get these key stakeholders on board with our new model.

## Summary

Well, that's a statement of the problem and that is it for this post. In the next post, I'll move on to describing a more helpful and enlightened approached to the transition, and to the changing role of the Product Manager in the increasingly agile enterprise. But in this post, I wanted to set the context so we will at least recognize the hole we have likely dug for ourselves in the meantime.