

# Extreme Product Management

How to deliver products people want to buy in an agile development environment

By Barbara Nelson and Stacey Mentzel

## Definition of Extreme Product Management (XPM):

XPM is using the minimum process and creating the minimum artifacts to deliver products people want to buy.

Writing this article feels like a development project. We are buried in the details. We each have hundreds of user stories in our heads on 3x5 cards. Some are related, others are not. Some are from personal experiences, others are from interviews and discussions with product managers. We could take an agile approach and just start writing content for each card. That's basically what we've done. But so far, it isn't readable by anyone but us. To further complicate things, we are writing this article together, have not seen each other in a few years, have only spoken a couple of times, and live on opposite sides of the country. In order to turn those user stories into a coherent article, we need some organization. We use FreeMind, a mind-mapping tool. Before you know it, we have almost as many nodes in FreeMind as we have user stories.

## This article needs Extreme Product Management (XPM)!

### Minimally, XPM needs to define:

- What problem are we trying to solve?
- For whom are we trying to solve the problem?
- How pervasive and valuable is solving the problem?
- What is our vision of the solution?
- What context do we need to communicate to drive the solution?
- Can we solve it in its entirety or can we take a phased approach?

An overriding principle is that the solution has to be practical. As we implement, we move from more general to more specific. Here goes.

### What problem are we trying to solve?

One of the biggest challenges today for product managers is clarifying the role of Product Management. In software companies, this is especially challenging as Development teams are adopting more agile or iterative development methodologies to meet the increased demands to develop more code faster. Where does Product Management end and Development begin?

The strategic role of Product Management is president of the product. Running the product as a business. Taking a market-driven approach to product management. Making decisions about the product to increase adoption across the market, delight customers, and ultimately deliver profitable products. Even if your product is "free," it should contribute to profits somewhere in the company. This is what we mean by Extreme Product Management. Staying focused on the point. What is the single overriding point of what we do?

But instead of being "president of the product," product managers often feel like "janitor of the product." Cleaning up after everyone instead of leading the team. Picking out icons, writing error messages, answering 30 questions a day because the developers don't want written requirements.

### Product managers say:

"My developers are using Extreme Programming and it is creating chaos."

"My developers want me available every minute of the day to answer their questions. I have no time to visit customers."

"I'm being asked to rewrite our user stories to fit the sprint."

"The developers are delivering code every two weeks, but it doesn't seem like we are making any progress towards delivering a useful product."

"My developers are being told to watch out for the product manager. What is my role in agile development?"

"It seems like agile development is the unionization of the developers. Resource allocation is done based on a 40-hour work week. Why am I putting in 60-70 hours a week to make this work?"

"How can I market a product when I have no idea what will be in it or when it will be available?"

In fairness, we also have to listen to the developers and why agile methodologies have sprung up in the first place.



## Angst. Frustration. Conflict. Mistrust.

Let's get back to basics here. What is the real problem? On the surface, the problem seems to be that Product Management and Development are each trying to wrestle control from the other. Product Management is frustrated at what appears to be a lack of commitment from Development. Development is frustrated because Product Management won't commit to what they want and keep changing their minds. Even customers don't know what they want.

### Developers say:

"Product managers keep changing their mind. The churn is killing us."

"We're being asked to stop what we're doing to respond to the deal of the day. We'd get something done if you'd just let us finish what you already asked us to do."

"I have no idea what the product manager is asking for. Flexible and scalable are not requirements!"

"If the vision is going to change every quarter, what good is the vision?"

"Market Requirements Documents are so eighties. We're agile." [Translation: "We don't need no stinkin' requirements!"]

"How can I estimate how long something is going to take when I have no idea what I'm being asked to do?"

"We're tired of getting beaten up by management for not meeting our dates. How can we be blamed when the target keeps moving?"

"It's about time we took back control. Product managers don't know what they want."

### How pervasive and valuable is solving the problem?

Every week in the Pragmatic Marketing® product management seminars, a show of hands often yields 30-40% of the students struggle with what their role is in agile development. XP and other agile development methodologies have been enthusiastically adopted by multitudes of development teams in an attempt to address the problems of developing software. (On the surface, to someone not involved in software development, building products doesn't look that hard. What's the big deal? It's just code! And yet a Google™ search of "software development challenges" yields 179 MILLION hits!)

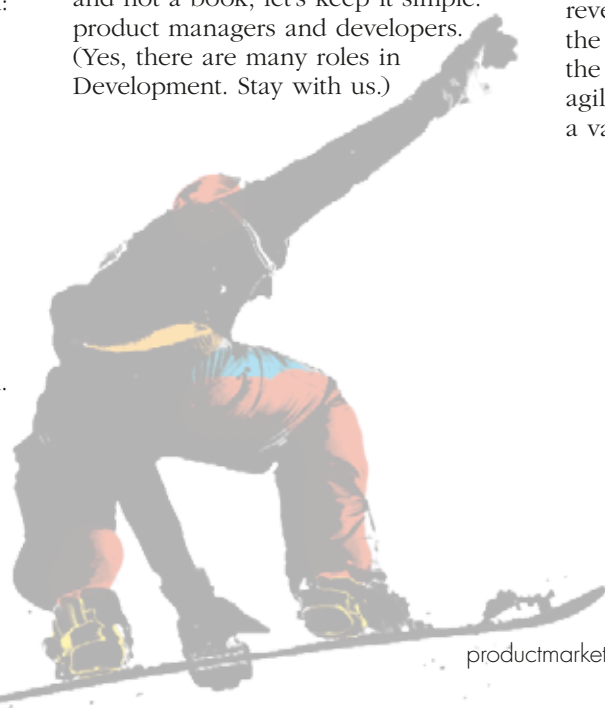
If agile methods by themselves were solving the problem of **building products people want to buy**, we wouldn't need to write this article. We'd say, "Be agile!" But over and over again we hear that agile has helped deliver **something** sooner, but without some planning and overall vision, what is being delivered (let's call it user stories) may not actually help us sell more software. We have decomposed the giant, complicated the project into such minute pieces in order to deliver **something**, that we can't market the product because it never seems to be done. And we can't predict when it will be done so we can project (and deliver) revenue. Ask any product manager in the middle of the quagmire if clarifying the product management role in an agile development environment is a valuable problem to solve. →

## Stop!

The real point of all of this is to **build products people want to buy**. Ultimately, we're on the same side. Ultimately, we all want successful products that help us all make lots of money.

### For whom are we trying to solve the problem?

There are a number of players or "personas" in this situation. But to bound this problem to fit into an article and not a book, let's keep it simple: product managers and developers. (Yes, there are many roles in Development. Stay with us.)



## What is our vision of the solution?

### **Build products people want to buy.**

This can be achieved any number of ways. Two guys in a garage. An agile team of 5-10 people. 18 teams across continents producing 62 products which have a high level of interdependence. In fact, to build products people want to buy, we may need different methods for different situations and different types of projects. XPM doesn't dictate the method (that's right), but must plug into whichever method has been adopted (this is the hard part). An extreme product manager is the evangelist on the team that keeps the team focused on the overall vision.

XPM is a response to Extreme Programming and agile development. Without XPM in an agile environment, we run the risk of building the wrong product. We might actually delight the one customer we involved in our agile process, but we might not be able to sell the product to anyone else.

XPM is needed when you are trying to build products the **market** wants. The "market" is more than one installation that you will sell over and over again. It implies **many**. When you are building products for a single project, you need customers directly involved in the process. Extreme Programming accounts for that by requiring the customer to be onsite with the developers. But when you are building products for many customers, you need a representative of the "many" (the

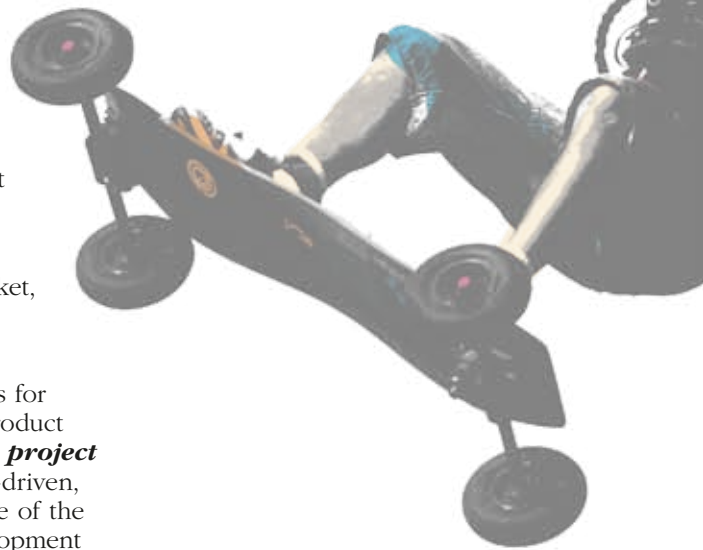
market). This is a product manager. This isn't to say that we won't have user-centered design and real users also involved. (The number one attribute of successful products is user involvement.) But when you're building a product for the market, the product manager needs to make sure we know **which** users are important for us to work with. Who are the representative users? And to make the product commercially viable, the product manager guides the team to solve problems **worth** solving.

Are "market-driven" and "agile development" mutually exclusive? If Development has taken the agile reins, does this preclude Product Management playing a strategic role?

By the way, if you are not building products for the market, you don't even need product managers. Let me repeat that.

If you aren't building products for the market, you don't need product managers. What you need are **project** managers and if you are deal-driven, you need sales engineers. One of the problems with so many development and requirements methodologies is that they are for one-time projects, not building products for the market.

James Shore who was the tenth person to sign the Agile Manifesto ([www.agilemanifesto.org](http://www.agilemanifesto.org)) after it was originally written, says "Product managers are critical to success." On February 14, 2006, he wrote a blog entry ([www.jamesshore.com/Blog/Product-Managers-Are-Critical.html](http://www.jamesshore.com/Blog/Product-Managers-Are-Critical.html)) detailing why a product manager is critical to successful software development. He values agile. He values Product Management. Product Management and agile development are not mutually exclusive!



### **To be an extreme product manager:**

- Know the market.
- Find valuable problems to solve.
- Articulate the vision and strategy to keep everyone aligned.
- Provide just enough detail to the team but keep the backlog of problems prioritized and ready to deploy as each iteration begins.
- Encourage the development group to innovate within the boundaries of market segments, personas, goals, and problems to solve.

If someone isn't doing this, who is? Are the inmates really running the asylum?



## What context do we need to communicate to drive the solution?

- We need to clarify terminology.
- We need to define activities.
- We need to clarify roles.
- We need to communicate context to drive the solution.

### Terminology

Let's start with two development approaches (to keep it simple). Waterfall and agile.

- **Waterfall.** In a waterfall model, Development is seen as a waterfall that steadily flows downward through the phases of implementation.

Requirements

Functional Specifications

Detailed Design

Development

Test

We finish one phase before beginning the next. Consequently, it is very difficult to go back to a prior phase once we've begun the next. Waterfall is theoretically less risky because more planning takes place up front. Less rework is needed if the design is well thought out. Estimates and schedules are more predictable. The disadvantage of a traditional waterfall is that all of the work in a phase must be completed before starting the next. It requires more formal documents (Market Requirements, Business Requirements, Product Requirements, Functional Specifications, Detailed Design, Test Specifications, and so forth). Creating and maintaining documents is time consuming and with a waterfall approach it is more difficult to be flexible to changes in the market. It also takes a long time to have working code to show customers for feedback. By then, it is usually too late to make significant changes without significantly impacting the cost and schedule.

- **Agile.** In an agile environment, the phases are smaller, iterative "chunks" rather than finishing an entire phase before moving on. It is a cyclical path through the process rather than linear, involving customers early on to get better clarity to end-user requirements. Requirements might be communicated through use scenarios or user stories. Prototypes, experiments, and conversations with customers provide feedback throughout development. Spiral, Iterative, Extreme Programming (XP) and Scrum are all forms of agile development. Each has its own nuances and principles. A major advantage of an agile method is quicker delivery of functionality which allows earlier feedback from customers. It also allows more flexibility when requirements or priorities change. A disadvantage is difficulty getting estimates and schedules of when the entire product will be complete. It is easier to get lost in the small details when the entire product is broken down into its subatomic pieces in order to fit the work into small sprints or iterations.

Any development methodology can work and any development methodology can fail. If we don't get all members of the team going in the same direction with the same vision it doesn't matter what approach you have. XPM is an approach to articulate the vision and direction and to foster the team spirit to **build products people want to buy.**

Agile development is often just programming without requirements. It is Development's response to poor product management. Without knowing market requirements, developers propose fast, small iterations to put in front of a customer for quick verification before proceeding. The promise of agile development is better product delivered faster, but if there is no clear vision of what the overall product is, it is unlikely that the right product will be built. We need to understand the whole before we start decomposing it into its actionable tasks. If we don't know what is shipping and when, we can't market the product.

Does it have to be only one way or another? Waterfall versus agile? What about a third alternative?

- **Agile waterfall.** With a solid roadmap in place and stable market requirements, we should be able to review and complete market and functional requirements at the start of a project. Then, Development can iterate through implementation and testing while Product Management is preparing for the next release cycle. Since Development reaches a point where the entire release is designed, they can set a good target date and free Product Management to focus on promotions and the next big thing.

The agile waterfall model would consist of small, quick iterations plus a multi-year product roadmap. This gives Development a series of small successes and allows the company the ability to forecast delivery. It combines the best of both worlds: the flexibility of agile development, and the planning cycles needed to maintain exciting, innovative, revenue-driving long-term vision.

Alistair Cockburn, one of the original authors of the Agile Manifesto says, "Different projects have different needs.

*Sure, if your project only needs 3-6 people, just put them into a room together. But if you have 45 or 100 people, that won't work. If you have to pass Food & Drug Administration process scrutiny, you can't get away with this. If you are going to shoot me to Mars in a rocket, I'll ask you not to try it.*

- As the number of people involved grows, so does the need to coordinate communications.
- As the potential for damage increases, so increases the need for public scrutiny, and so decreases the tolerance for personal stylistic variations.
- Some projects depend on time-to-market and can tolerate defects (web browsers being an example), while other projects aim for traceability or legal liability protection."

(continued on page 10) →



### Activities

Here's the work that needs to get done:

1. Problem definition
2. Analysis and design
3. Build and test

These are the necessary phases to building good software, regardless of the development methodology used. Depending on which approach we use, the level of written documents and the scope of what is done will change. With waterfall, we will do a lot of work in each phase before moving to the next phase. The written documents will be more detailed. With agile, there is less written documentation and the scope for each phase is smaller.

### Roles

A key success factor to move through the phases is defining who does what. Unless one person is doing everything, you have to divide up the work. With two guys in a garage, one person might define the problem, and the other might analyze, design and build. Maybe the first guy tests. But each knows which part to do because they talk it through. It might alternate from task to task, but whenever there is more than one person, role definition and communication are essential.

Whoa, whoa, whoa! Who has this many roles? It's hard enough getting a product manager and a product designer added to the team. Let's get back to simplification. Back to what needs to be done. Let's assign the following roles.

1. Problem definition → Product manager
2. Analysis and design → Product designer
3. Build and test → Developer and Quality Assurance

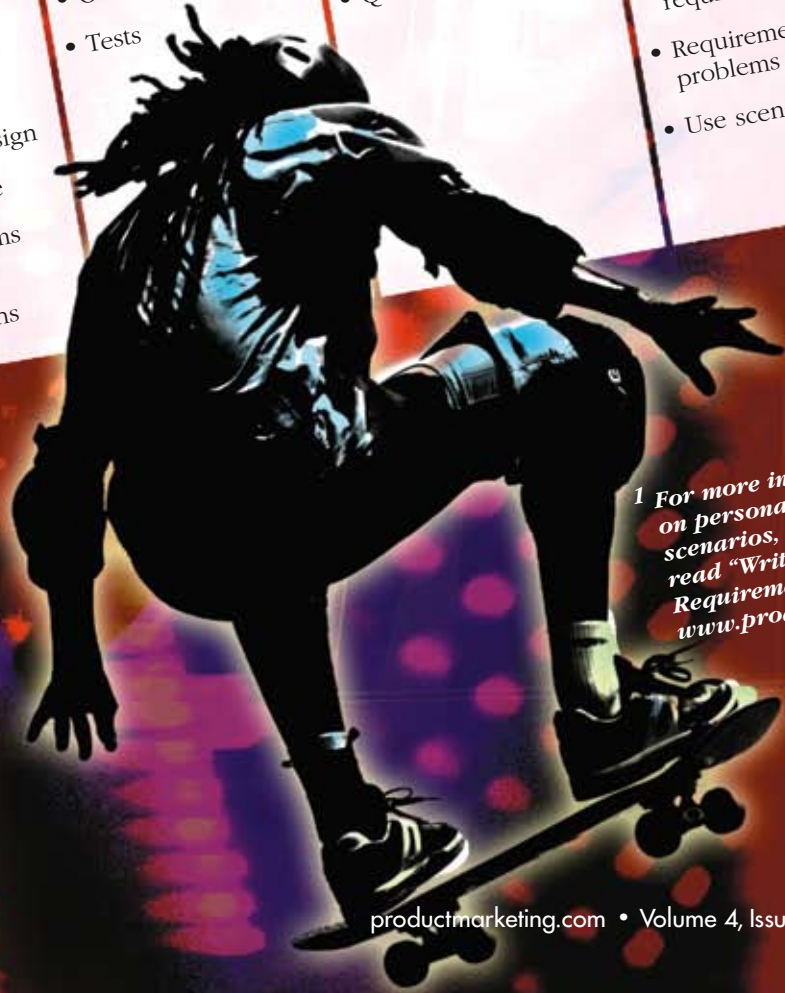
To deliver products people want to buy, we also need to involve the "customer"—possibly at every step. Product managers should ensure the validation is done with representative customers. Many agile methodologies require the customer to be onsite with the developers. Getting representative customers onsite may not be practical so product managers often play this role. But even product managers cannot be onsite all the time if they are to have time to actually visit the market.

The inputs and outputs might be written, might be communicated face-to-face, or a combination of both. It depends on how "agile" you are. It depends on whether members of the team are in the same location. But keep in mind, while the developers may not need written documentation in order to develop, others might—Support, Training, Professional Services, Operations, and customers. In an agile environment where user stories communicate requirements, you might need a user story articulating the need for the documentation. →

### Here are some possible roles:

- Product manager
- Project manager
- Program manager
- Product designer
- Systems engineer
- Product architect
- Systems architect
- User interface specialist
- Interaction designer
- Usability expert
- Business analyst
- Lead developer
- QA lead
- Developer
- QA engineer
- Release manager
- Executive sponsor

Step	Inputs	Outputs <sup>1</sup>	Role	Validate
Problem definition	<ul style="list-style-type: none"> <li>• Call reports</li> <li>• Enhancement requests</li> <li>• Competitive analysis</li> <li>• Win/loss</li> <li>• Support incidents</li> <li>• Defects</li> </ul>	<ul style="list-style-type: none"> <li>• Overall vision of the solution</li> <li>• Market segments</li> <li>• Personas</li> <li>• Goals</li> <li>• Problems</li> <li>• Use Scenarios</li> <li>• Requirements</li> </ul>	Product Manager	<ul style="list-style-type: none"> <li>• Which segments to pursue</li> <li>• Problems worth solving</li> <li>• Understand problems and use scenarios</li> <li>• Priorities</li> </ul>
Analysis & design	<ul style="list-style-type: none"> <li>• Vision</li> <li>• Segments</li> <li>• Personas</li> <li>• Goals</li> <li>• Problems</li> <li>• Use scenarios</li> <li>• Requirements</li> </ul>	<ul style="list-style-type: none"> <li>• Functional specifications</li> <li>• Prototypes</li> <li>• Interactive design</li> <li>• User interface</li> <li>• Flow diagrams</li> <li>• Technical specifications</li> </ul>	<ul style="list-style-type: none"> <li>• Product Designer</li> <li>• Other related design roles</li> </ul>	<ul style="list-style-type: none"> <li>• Prototypes</li> <li>• Usability</li> <li>• User interactions</li> <li>• User interface</li> <li>• Specifications</li> </ul>
Build & test	<ul style="list-style-type: none"> <li>• Functional specifications</li> <li>• Prototypes</li> <li>• Interactive design</li> <li>• User interface</li> <li>• Flow diagrams</li> <li>• Technical specifications</li> </ul>	<ul style="list-style-type: none"> <li>• Code</li> <li>• Tests</li> </ul>	<ul style="list-style-type: none"> <li>• Developer</li> <li>• Quality Assurance</li> </ul>	<ul style="list-style-type: none"> <li>• Product meets specifications</li> <li>• Specifications meet requirements</li> <li>• Requirements solve problems</li> <li>• Use scenarios can be met</li> </ul>



<sup>1</sup> For more information on personas, goals, use scenarios, and requirements, read "Writing the Market Requirements Document" on [www.productmarketing.com](http://www.productmarketing.com)

## Can we solve it in its entirety or can we take a phased approach?

As with anything we're trying to do, get back to the problem. Prioritize the problems and start working your way through them in a phased approach. These are complex problems. Even if we could fit the entire solution into an article, implementing it will not happen in one day, or one week, or one month. It takes time (and iterations). People are involved.

Back to the problems we articulated earlier in this article:

### Challenges

PM: "My developers are using Extreme Programming and it is creating chaos."

PM: "My developers want me available every minute of the day to answer their questions. I have no time to visit customers."

Dev: "Market Requirements Documents are so eighties. We're agile." [Translation: "We don't need no stinkin' requirements!"]

PM: "I'm being asked to rewrite our user stories to fit the sprint."

Dev: "I have no idea what the product manager is asking for. Flexible and scalable are not requirements!"

### Look for Root Cause

- Why are they using Extreme Programming?
- What are the specific problems?

- Do they understand the customer's domain?
- Are representative customers available to answer questions?
- Is there a product designer on the team?
- Are you playing the role of product designer?
- Why do they resist written artifacts? Does it take more or less time with no artifacts?
- What types of questions do you commonly get?
- What level of detail are you getting pulled into?
- Do you trust Development to bypass you for the answers?
- Does Development trust you?

- Why?
- Is the story too big because there are multiple stories in it?
- Is the story too high level?
- Are you being asked to put a complex (yet legitimate) story into subatomic "tasks" so they fit the iteration?

### Approach

Do some problem discovery to break this down into manageable "chunks." What are the specific situations (use scenarios or user stories) associated with the chaos?

Provide market context with personas, goals, problems, use scenarios, and requirements.

If the developers refuse to read artifacts, then at the beginning of an iteration or sprint do a white board presentation to give context to the team. Who are you building for (personas), which goals are we addressing, what are the problems they have that if solved would allow them to achieve their goals, and what are the scenarios (user stories) they will be implementing?

Look for ways to transfer knowledge in a repeatable way whenever you can. (If the artifacts provide valuable context, maybe the Developers would be more willing to read them.)

To build trust, work with the other members of the team on a personal, individual basis.

If a user story is legitimately bigger than an iteration, recommend it span multiple iterations. It isn't usable by a customer until the whole story is complete.

Suggest a product designer break the story into multiple tasks that could potentially be built over multiple iterations. Breaking it into the implementation tasks is not product management.

PM: "The developers are delivering code every two weeks, but it doesn't seem like we are making any progress towards delivering a useful product."

Dev: "Product managers keep changing their mind. The churn is killing us."

PM: "My developers are being told to watch out for the product manager. What is my role in agile development?"

PM: "It seems like agile development is the unionization of the developers. Resource allocation is done based on a 40-hour work week. Why am I putting in 60-70 hours a week to make this work?"

Dev: "It's about time we took back control. Product managers don't know what they want."

PM: "How can I market a product when I have no idea what will be in it or when it will be available?"

Dev: "We're being asked to stop what we're doing to respond to the deal of the day. We'd get something done if you'd just let us finish what you already asked us to do. How can I estimate how long something is going to take when I have no idea what I'm being asked to do? If the vision is going to change every quarter, what good is the vision? We're tired of getting beaten up by management for not meeting our dates. How can we be blamed when the target keeps moving?"

- What is in an iteration?
- How is the work prioritized?
- Are smaller, lower priority items being added because the larger items don't fit an iteration?
- Is the cumulative effect of many lower priority items impacting the ability to deliver useful functionality?

- Do you know your overall role in the company?
- Are you more like the janitor or the president of the product?
- What are all of the roles in the development process?
- Who plays each role?
- Do both Product Management and Development understand each other's methodologies?

- Is this true?
- If this is true, and if Development is not delivering enough value with this philosophy, won't it eventually catch up?
- Are you in the right job?
- Are you in the right company?

- What is preventing the team from estimating completion of key milestones?
- Do things change too much for Development to commit to a schedule and then deliver on it?
- Does the team understand the impact of not knowing?
- Does the team need a brief primer on both inbound and outbound activities of Product Management?
- Does the team know the revenue implications of not knowing?

Prioritize, prioritize, prioritize!  
Be involved in iteration planning and iteration reviews. Communicate market facts, not opinions. To influence without authority you need credibility.

Educate each other on your methodologies: roles, responsibilities, terminology, goals.  
Do a gap analysis of the Pragmatic Marketing Framework of the Product Management role.  
Define roles and responsibilities between Product Management and the other team members.

Do your job and let go of the things over which you have no control. Life is too short.

Minimize "deal of the day" churn on Product Development.  
Prioritize, prioritize, prioritize.  
Define what the minimum viable product or deliverables are to successfully drive revenue. Use a product roadmap to articulate high level phases of implementation.  
Keep revenue and profit targets visible to Development.



## Summary

Here are the minimum activities to have a viable approach to Extreme Product Management. The rest can come later.

- **Build trust.** (Beer and fajitas is a practical way to get started.) Share methodologies. Learn each other's terminology, processes, artifacts, principles, and goals.
- **Communicate the vision.** Who are you building the product for, why would they want to buy it, and what key problems will you solve? Speak in market facts, not opinions.
- **Clarify roles.** Do a gap analysis of the Product Management role using the Pragmatic Marketing Framework. Identify roles and responsibilities on the Development project. Who does what?
- **Prioritize the work.** XPM is a market-driven approach. To participate in what is being built, you need to have a quantified, prioritized list of problems to solve. Feed this to Development in the form of personas, goals, problems, use scenarios, and requirements. Work together on the scheduling challenges to fit the work into agile, iterative pieces, but don't get dragged into the subatomic level.

pm.c

### About the authors:

We both have been product managers who have worked with many, diverse development teams. We've worked on entirely new technology, and very mature offerings. We have worked with no roadmap, no defined destination—and product lines with clearly defined strategy. We have both been honored to have worked with engineers dedicated, motivated, and driven to succeed.



**Barbara Nelson** is an instructor for Pragmatic Marketing. She has 21 years in the software industry, including VP of Product Marketing for a leading provider of business and accounting applications.

Contact Barbara at  
[bnelson@PragmaticMarketing.com](mailto:bnelson@PragmaticMarketing.com)



**Stacey Mentzel** is a Director of Product Management at Business Objects, within the Enterprise Information Management area. Stacey has more than six years of experience in Product Management, along with experience in Product Testing.

Contact Stacey at  
[Stacey.Mentzel@businessobjects.com](mailto:Stacey.Mentzel@businessobjects.com)