Sliger Consulting Inc.

Selling Agile White Paper

How to Respond to Concerns from Management, the Business, and the Team

Michele Sliger and Stacia Broderick

ABSTRACT

Are you excited by the potential of agile software development, but find that your colleagues are a bit reticent? Is your whole team ready to dive in, but your business partner is only interested in dipping in a toe—if that? Are you struggling as a project manager, wishing for the right way to help your management see that agile is the way to go, and wishing for the words that will help your teams feel more confident about trying agile? Or maybe you're wishing you could find a way to convince your clients that there's a better way to contract for a software development job—without having to do a full-blown detailed design spec up front. This paper will look at all of these questions surrounding how to best sell agile in your organization.

Note: This paper is an updated excerpt from the book *The Software Project Manager's Bridge to Agility* by Michele Sliger and Stacia Broderick, published by Addison-Wesley who retains the copyright.

Table of Contents

Introduction....3

Some General Ideas About Selling....3

Selling to the Team 4

There Are Too Many Meetings....4

We Don't See the Point in Gross Level Estimating 6

If We Don't Do Any Technical Planning Our Architecture Will Fail....6

We Aren't Co-Located So We Can't Be Agile....7

Some Unspoken Reasons....7

Selling to Management 8

Agile Doesn't Allow for Long-Term Planning....8

It's Worked So Far – Why Do We Need to Change? 9

Our Situation is Just Too Complicated for Agile 9

We Need to Matrix Resources to Get Maximum Efficiency 10

Our People Can't Be Trusted to Self-Organize....10

How Can We Make Strategic Decisions Without Gantt Charts?....10

Selling to Customers/Product Owners....11

You Just Want Us to Contract With You on a T&M Basis So You Can Bill Us For Eternity....11

I Don't Have Enough Time to Work With the Team Every Iteration....12

I Can't Wait an Entire Iteration for That Feature! 12

Selling to Other Departments in the Organization 13

Other Ways to Sell Agile....13

Summary 14

About the Book: *The Software Project Manager's Bridge to Agility* 15

About the Co-Authors 15

Bibliography....16

Introduction

> *Are we selling or building trust? – Hubert Smits*
>
> *I really don't "sell" anything, and don't even think that I can do so very well. What I can do, on my best days, is to listen to what someone wants, what they are having trouble with, and suggest a specific thing to do to make that part of their life better. – Ron Jeffries*

As an agile practitioner (or agile practitioner wannabe) out to convince others that adopting agile is the way to go, you will ultimately be asked by several people – whether they be team members, management or customers – "What’s in it for me?" Congratulations! Part of your role as an agile advocate is to help others understand what agile is and is not, and the benefits that can be derived from a successful agile adoption.

This chapter excerpt is intended to provide some helpful hints when thrown into the situation of selling, persuading, or convincing others about agile. What we call “selling” is really the act of finding what interests people and seeing if an agile approach can help meet their needs. You might be involved in an organizational change to agile methods, and understanding how to relate information to certain groups of people will greatly help in the transition.

We look at selling agile in several ways: selling in general, selling it to the team, selling it to management, selling it to the customer/product owner, and selling it to other departments in your organization. We finish with a look at *not* selling, i.e., finding other ways to share and promote agile practices.

We also assume that you already know the basics of agile frameworks, principles, and practices, and thus are familiar with agile terminology.

Some General Ideas about Selling

Ask any salesman and he’ll be sure to tell you that it helps if you believe in the product yourself before you try to sell it to somebody else. The same is true for agile. How can you convince someone of the benefits of something if you do not receive or have not observed these benefits yourself? It can be a little difficult to do.[1]

In addition to believing in an idea, there’s another aspect of selling: know what is motivating the buyer. Some refer to this as "pain points," which is a level of difficulty sufficient to motivate someone to seek a solution or an alternative. In other words, knowing enough about agile to position it against various pain points of the organization will surely help the sell. Our standard response to most protests is “How do you handle that now?” followed by “And how is that working out for you?” After a momentary stammer and some nervous laughs from those witnessing this exchange, the protester usually admits that their current method isn’t working out very well. An admittance that a problem exists and a solution is needed is the best way to start the discussion.

[1] If you lack experience but still want to make a pitch for agile in your organization, you’ll need to use metrics and studies to make your points. We recommend you start with VersionOne’s annual State of Agile Development Survey at http://www.versionone.com/whitepapers.asp.

While it’s nice to present a perfect picture of an idea you’re wild about, it’s also important to acknowledge the weaknesses. Nothing is bulletproof, there is never a one-size-fits-all solution that will fix everyone’s pain and make lives perfect. The same is true with agile; although it does improve the management of complex projects, it does not fix everything. Difficult situations and trade-offs will still exist, no matter the approach.

Part of selling is learning when not to sell. Sometimes you have to plant the seed of an idea within people’s minds and then let some calendar time pass. Learn when to push and when to back off; and educate throughout the process.

Now let’s discuss some of the groups you may wish to sell agile to as it takes hold in your organization. We’ve compiled a list of the most popular questions and have provided some food for thought for each. Please note that this is not an exhaustive list, and the ideas are not prescriptive. We just want to expose you to many of the questions that we are often asked and ways these questions can be addressed.

Selling to the Team

The team can be your hardest sell, especially in the situation where agile has been brought in from top management. Developers, quality assurance staff, technical writers, and other professionals do what they do because they are smart. They will challenge you, and they have every right to. Arming yourself with information about principles and values is vital to helping a team understand why the discipline in agile approaches is necessary and how that discipline can ultimately benefit them.

There Are too Many Meetings

After we dispel the notion that agile teams don’t do any real planning, there is a general horror expressed by many of our clients at the number of meetings that agile requires. We’ve found that this negative view of the meetings is usually based on their own experience with their current ongoing meetings. Usually starting late and lasting too long, with no agenda, and too much politicking and finger-pointing, these nightmare meetings set the tone for what they expect agile meetings to be like.

When we encounter the pushback regarding the number of meetings involved in agile – iteration planning, iteration demos, reviews, retrospectives, daily stand-ups, not to mention release planning and backlog grooming – it is time to be clear about the differences in how these meetings are run, and how all the “overhead” as they see it is actually quite productive.

The first question to ask is “In the process you’re using now, how do you know what to work on?” The answer generally centers on documentation: “I’m given a spec and I code to what I interpret the spec to mean.” The second question to ask is, “Yes, and how is that working out for the customer? Are they getting software they can use?” What we’ve heard from hundreds of teams is that this process generally doesn’t work out well. With this acknowledgement comes the uncertainty of how to lighten up the heavy documents. Well-timed meetings help us accomplish this.

First, agile meetings are in keeping with one of the basic agile tenets, that face-to-face communication is the most efficient and effective way to communicate. Combine this with another agile tenet, that of "the art of maximizing the amount of work not done,"[2] and the meetings become the most efficient way to communicate information, plan the work, make improvements to both the product and the process, and eliminate quality errors caused by misunderstandings. These meetings require team involvement and focus on the next best thing to do. They are efficient in that there is an agenda, the planning is limited to the timebox at hand, and information does not need to be repeated or heavily documented, as everyone who is affected is there and participating in the decision-making.

Sometimes it helps to do the math. Five daily standups = 75 minutes, not much more than a weekly status meeting, so that's a wash, especially when you consider that there is no daily stand-up meeting when the team is already all together in an iteration planning meeting or an iteration review meeting. So if your team is doing two-week iterations, they really only have four daily stand-up meetings for a total of 60 minutes. More math: four hours in an iteration planning meeting * 6 two-week iterations = 24 hours, which is less time than you would normally spend planning a quarterly release. And don't forget, you're not supposed to be attending agile meetings AND other non-agile project meetings! Asking the team to follow two different processes isn't efficient at all—the agile meetings should replace existing ones.

Probably the most contested meeting in an agile setting is the daily standup. Teams often complain that they feel as if this is a mechanism only in place so that managers can micro-manage. Remind the team that this meeting is *their* meeting, designed for them to inspect and adapt their work tasks in response to how the iteration is unfolding, and to coordinate with one another.

One technique that we've used in the past to push the team to run its own meeting is to walk away and let the team continue the meeting solo. This sends the message that "I trust you as professionals to run this meeting on your own." Afterwards, stop by and talk to a team member or two to see if any obstacles were brought forth in the meeting and take these down. Of course, we wouldn't do this with a brand new team. We would do it with one that has the hang of how to run this meeting (which doesn't take too long). Mike Cohn tells a story of how he had to attend daily stand-up meetings with a magazine and pretend to read during the meeting. Deprived of his eye contact, the team started speaking to each other, which is the whole point of these stand-ups.

Often, the "roll up your sleeves" approach is very beneficial. Give the status of the impediments in your own backlog as the agile project manager. How are things going with getting Support's presence at the product review meeting? How did the discussion go with the vice president regarding publishing agile reports to the dashboard? Visibility into this information will build trust with the team. They will come to understand that you have work that you are doing for the good of the team, and you are open to sharing it in the daily standup meeting. When the team feels like you are supporting and serving them, they will begin to trust you.

It's just fifteen minutes. Remind the team that their engagement in this meeting is meant to keep them out of other meetings, hopefully saving them some time. Also, remind them that the daily standup is also a mechanism to promote visibility into work and that anybody is invited. This meeting, while primarily to serve the team, also serves the greater organization by promoting transparency.

[2] From the "Principles Behind the Agile Manifesto," the full quote is "Simplicity—the art of maximizing the amount of work not done—is essential." In other words, by keeping things simple you don't end up wasting time doing work that did not need to be done. You can read all of the principles here: http://www.agilemanifesto.org/principles.html.

We Don’t See the Point in Gross-Level Estimating

Teams will resist high-level (gross) estimates of feature complexity, mainly because they’ve felt pressured to give "perfect estimates" in a plan-driven environment. Gross-level estimates feel like “fuzzy logic” to engineers.[3] They’ve on occasion received punishment for "bad" estimates by having to work late to make up the time. Sometimes, they don’t want to give estimates at all because sales and management make promises based on these numbers or replace the team members’ estimates with a "better estimate" anyway.

Remind the team that gross-level estimating was devised as a way to abstract time from the estimates in order to focus on the complexities surrounding implementation of the feature; additionally, this level of estimating provides a look into longer-term planning. Our experience is that over time teams become very effective at estimating using this technique. It is important to remind the team that the time to implement a feature is important, but we must not confuse expected duration with estimates of the level of effort. In agile, the durations are always the length of the iteration. Instead, explain to the team that the time that a feature takes to implement is impacted by many things: skill, technology, culture, distributed teams, code complexity, and so on. The gross-level complexity estimates provide a way of factoring in these elements as part of the overall complexity of the feature, and understanding that complexity helps us to better determine a range of time and resources that will need to be expended in implementing that feature. History has shown us that trying to predict time rarely leads to accurate estimates; predicting complexity, however, does give us a better view into what to expect.

This gross-level complexity estimating is also important to the product owner who manages the product backlog. Knowing the level of complexity involved helps the product owner make decisions based on the return on investment the company would receive from the implementation of the feature. If a feature receives a high complexity estimate from the team, the product owner may decide to prioritize it lower than another high-value feature with a lower complexity estimate. It’s about getting the most return for the investment, and providing team-generated gross-level complexity estimates is the fastest and simplest way to indicate potential risk, effort, and size.

If We Don’t Do Any Technical Planning, Our Architecture Will Fail

We would tend to agree! While the Agile Manifesto states that “the best architectures, requirements, and designs emerge from self-organizing teams,”[4] that’s not to say that emergent design simply pops into existence from thin air. The design begins with a high-level plan, appropriate to the time horizon and in keeping with the overall product vision.

In the same way the product owner is planning the vision around the product and how it will serve the customer, the architect (or members of the agile team) should plan for how the system can support that vision and provide enough flexibility to meet the inevitable changes in requirements. In his book *Scaling Software Agility*, Dean Leffingwell refers to this agile architectural planning as the “architectural runway.”[5] Architectural planning is still emergent, but the idea of having a “runway” in place for the team means that the architecture, like the high-priority items in the product backlog, is being designed and detailed in the existing iteration for the next iteration or next few iterations. The idea is that an architectural foundation, or runway, will be in place and ready for the team to use as they take off on their next iteration. Portions of

[3] Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based.

[4] Kent Beck et al., “Principles Behind the Agile Manifesto,” http://www.agilemanifesto.org/principles.html.

[5] Dean Leffingwell, *Scaling Software Agility*, (Boston: Addison-Wesley, 2007), 204.

the runway continue to be built out as the project progresses and the “plane,” or product, gets larger and heavier with features.

While considering the architectural runway, it is important to remind the team of the statistics: sixty percent of software features are rarely or never used![6] Many have attributed this to the fact that in traditionally managed projects, every conceivable feature is planned for and expected to be implemented. This is frequently a symptom of fixed-price, fixed-scope projects with little to no room for change. Since these types of products historically were created in horizontal layers, one layer at a time, by hand-off from one group to the next, the architecture would be built to support any and all features outlined in the analysis phase. Unfortunately, many companies realized that after anticipating and building for all of these possibilities, by the time the system was complete, the customers’ business had changed, making some (or all) of the features and underlying architecture obsolete. In agile we build product increments as we go, and this also means the supporting architecture; however, we keep an eye on the horizon, making sound architectural decisions based on what we know today. Agile teams don’t want to waste time building something that will never be used, as well as add to future waste of time maintaining that functionality for the life of the product.

We Aren’t Co-located So We Can’t Be Agile

When you have a geographically dispersed team, communications simply take longer. Teams have to figure out ways to work together so that they communicate verbally as often as possible. Technology exists to support this: telephones, Skype, video conferencing. Throw in a few face-to-face meetings to build understanding and team cohesion, and even non-verbal instant messaging can then work well.

You can help build a sense of team and collaboration by helping team members learn about each other. Create trivia quizzes that allow team members to guess little-known facts about each other. Post photos on the wiki, along with favorite programming languages or hobbies. Invite remote team members to key planning meetings to kick off the project with everyone on board. With a little effort, virtual teams can be just as effective as co-located teams.

Some Unspoken Reasons

We’ve talked about some of the more obvious push back from teams who don’t wish to adopt agile (and by no means have we covered all of them) – but there are also many *unspoken* thoughts that contribute to a person’s reticence. They are all centered on fear and discomfort, and while we have no pat answers as to how to address these concerns, it is good to at least be aware of them. You’ll find that despite what team members (and project managers) are saying, they are often really thinking:

- I’m afraid of change
- I’m afraid I will have nothing to do
- I’m afraid I will lose my job
- I’m afraid people will see how little I actually do
- I’m afraid I won’t be able to keep up
- I’m afraid I won’t be able to learn the new software
- I’m afraid this will mean hard work
- I’m afraid I’ll be fired if the decisions we make don’t work out

[6] The Standish Group International, *The Standish Group’s CHAOS Report 2004,* (West Yarmouth, MA: The Standish Group, 2005).

- I'm afraid I won't get raises or promotions anymore
- I'm afraid of conflict and trying to reach consensus
- Nuts! there go my 3-hour lunches
- Nuts! that means I can't mosey in at 10:30 anymore
- Nuts! that means I'll have to really think now
- Nuts! that means I'll actually have to talk to people now
- It's just so much easier and safer when someone else tells me exactly what to do
- It's just so much easier and safer when I can tell them exactly what I want them to do

The best thing you can do to help people through these issues is to create and maintain an environment where failure is viewed as an opportunity for learning and growth. While you can make promises and clearly communicate that this is your intent, it has been our experience that people don't begin to really relax until they see the truth of it in action. Additionally, provide hands-on coaching and leadership for team members to help them envision their new roles and how to add value in new ways. For example, working with the developers to write unit tests can help them see how they are contributing to overall product quality. Helping technical writers learn incremental writing and editing techniques can help them scale to multiple teams. Assisting customers with product backlog preparation and inviting them to iteration demos can show them the influence that they really do have over development.

Selling to Management

Management sometimes has the same concerns as the team, but often their issues center around perceived lack of control and loss of identity. Many managers feel personally tied to the current state of development because they helped create this current state. Asking them to change to an agile approach is sometimes perceived as a criticism.

Agile Doesn't Allow for Long-Term Planning – How Are We Supposed to Do Our Budgets?

This is a misunderstanding born from the initial focus on doing iterative and incremental delivery. Because this is such a new way of thinking, it takes a while before newcomers learn about more strategic and long-term levels of planning, like release planning and defining a product roadmap.

Gross-level estimating is crucial for long-term planning. The team has backlog grooming meetings, where items in the backlog are estimated at a high level jointly by the entire team, using an abstract indicator designed to denote complexity. The Fibonacci sequence as tweaked by Mike Cohn, author of *Agile Estimating and Planning* is quite popular and is executed using a wide-band Delphi estimation method called planning poker.[7]

Once backlog items have a gross-level estimate assigned to them, and the team knows what its average velocity is (the amount of work a team can complete in an iteration—let's say it's indicated in points), a quarterly forecast can be planned by the team. The entire team conducts a quarterly or release planning meeting, and takes the items from the backlog and places them into the iterations that make up the larger timebox, until each iteration is considered full. Teams can

[7] For more information on this technique, read Mike Cohn's book *Agile Estimating and Planning.* And if your team would like to try planning poker but isn't co-located, try Cohn's free site http://www.planningpoker.com.

do this either in a feature-driven fashion, where the expected completion date is derived from the placement of all the needed features into iterations, or it can be done in a time-driven fashion, where the deadline is fixed and the team determines how many features they expect to complete by the deadline.

Once this planning has occurred, budgets can be determined by calculating the cost per point. The loaded salaries of the team members for a given timeframe divided by the number of points the team can complete in that timeframe will give you the cost per point, and budgets can then be derived for a set of estimated backlog items.

It's Worked So Far, Why Do We Need to Change?

Well if this is really true, and your customers are satisfied, then you probably don't need to change. However it has been our experience that this type of general resistance to change hides a different issue.

Traditionally, managers create metrics, reports, budgets, and career paths. They have goals and they try to meet those goals, not only for themselves but for their departments, and they are rewarded on how well their departments perform.

The focus in agile development shifts to that of cross-functional teams of people, not a group of individuals with similar skills, otherwise known as skill silos. The focus is also on creating self-managed, empowered teams. This notion troubles many managers because they feel like they will lose power or control. Because they must redefine their existence within agile, they will often resist the very idea of it. Sometimes, the personal change is just too great.

You can help managers by showing them how they can support a team. You can partner with them to help remove the biggest impediments. You can have discussions with them to find out what they are being asked to measure, and offer to talk with their bosses about how measurement is conducted on an agile project. You can also discuss ideas for career paths in an agile setting.

At the end of the day, you want to create a partnership with the manager. Help them see the results of the team, and engage them to help you help the team get past difficult times. Help them re-conceive the value they provide by learning about agile engineering and testing practices, creating centers of technical excellence, coordinating releases, focusing on vision and strategic direction, etc., so that product development "flows".[8] In an organization that's Agile, the work of continuous improvement never ends.

Our Situation Is Just Too Complicated For Agile

We hear this all the time: "Our situation is so special, so unique, so complex that we can't *possibly* change!" This is a mindset that is difficult to get past, and it is often only by demonstrating results that people can begin to think differently.

Help managers by referring to case studies of companies in similar situations. Remind them that change does not happen overnight. Ask what things they think they *could* change, or help them envision the desired state and what must change in order to get there, no matter how wild or wacky. Encourage them to create an impediment backlog and work through these issues in order of priority, in iterations. Offer to help.

[8] "Flow" is an idea from Lean. Introduce management to Lean concepts without the software development jargon by suggesting they read *Lean Thinking* by James P. Womack and Daniel T. Jones.

Know that there are agile teams who have passed FDA audits, Sarbanes-Oxley audits, and CMMI Level 5 assessments. NASA has teams using Scrum. Department of Defense (DOD) teams that shall not be named (or they'd have to kill us) are also using agile practices. There is even an IEEE standard, "P1648 - Recommended Practice for Establishing and Managing Software Development Efforts Using Agile Methods," that is in draft form and currently being developed. The more complicated the project, the more likely it will benefit from an iterative and incremental approach that incorporates feedback and learning into each iteration.

We Need To Matrix Resources To Get Maximum Efficiency

The industrial and maximum efficiency mindset of the 1920s and 1990s, respectively, insist that software can be created and managed by matrixing people across multiple endeavors so that nobody is ever idle. While software *can* be created this way, it is not the most effective approach. What hundreds of teams across the world are learning is that the dedicated team approach is the best possible scenario for focused work and innovation.

You can arm yourself with data here. There are plenty of reports that show the productivity hits when people are time-sliced across multiple initiatives – up to 40%![9] This statistic does not include the immense overhead of the management necessary to orchestrate the dependencies between matrixed staff. Moving to an environment of dedicated teams means a big leap in organizational structure to support this setup. Not everyone will see the benefits immediately, and experts will suddenly become constraints to building functionality every iteration. Investment must be made in cross-training and knowledge-sharing from senior to junior resources and across skill sets. The working environment must be safe for people to jump into another role when the team must meet its commitments. Management will need some convincing, however, that the learning and ramp-up hits to productivity in these early days will be worthwhile in the long run.

Our People Can't Be Trusted to Self-Organize

Agile won't solve this problem, but it will certainly highlight it on a daily basis. The problem is usually with management at this point, and not with the team. When there is no trust or respect from managers to workers, agile implementations will fail in that teams won't be able to self-organize. You will have to decide whether or not you want to stay and work to build a trusting relationship and safe environment, or flee.

How Can We Make Strategic Decisions Without Gantt Charts?

Executives need timely and accurate information in order to make sound strategic decisions about the future of the business. They really don't care how product is created, as long as it meets the functional and quality needs of the customer, there is reliable delivery, and it is done in such a way as to meet legal and ethical obligations. But they do care about getting information about the effects of product development: the ROI, the market share, the customer satisfaction rates, and so on. When they log into the project dashboard or look at their weekly reports and see burndown charts instead of schedule and cost performance indices, it can be incredibly annoying and frustrating for the executive. They feel that they don't have the information they need to steer the business. This is when educating them about agile is critical.

Let them know they'll still have access to all the information they've come to expect from projects, just in a different format. If you can get them to agree to support an agile pilot team, then the best way to do this is to have the teams create information radiators and then invite the executives to the team room and help them understand what's being built and how we can track

[9] Mike Cohn, "The Dark Side of Multi-Tasking," *Better Software*, July/August 2005, 10.

progress of the team. Ask what information they get the most value out of, which information is confusing and work to improve the value and visibility of this information. There are also several tools on the market specifically designed for agile project management, and these should also be investigated as possible solutions to the reporting problem.

Another effective approach is to invite the executives to the pilot team's product review meetings. Often they are very impressed to see working code so early. This transparency helps create trust by the executives in that the process is working and that delivery is reliable and consistent.

Over time, education and making results visible will greatly increase the level of trust with the executives, enabling them to make timely decisions about products with relation to strategy.

Selling to Customers/Product Owners

Customers want value. They want results. They are paying for a product and want to be able to use it exactly as they've envisioned. We've included a few statements that you'll hear from some customers when they find out the team is using an agile approach.

You Just Want Us To Contract With You On A Time And Materials Basis So You Can Bill Us For Eternity

This is probably the number one reason why customers will be reluctant to engage with the team in an agile fashion. In order to understand this, one must examine it from a contract perspective. It is important to understand the nature of contracts; they protect somebody from risk. Sometimes are they written as mutually binding agreements, but usually they favor one party over another. Fixed time/fixed scope contracts are written to place most of the risk on the vendor; a penalty is applied if the vendor does not deliver to the specified scope by the specified time.

This type of arrangement puts pressure on the customer to specify all of the scope up front, and calls for change control meetings to manage any change. The vendor doesn't want to change the scope because there are uncertainties of the related impacts in doing so. The customer wants the change, but has to fight for it, and the vendor retaliates, feeling as though the customer should have thought about it in the first place. It is usually an uncomfortable situation for both parties.

Changing contracts from fixed scope/fixed price to a pay-as-you-go time and materials (T&M) means that customers are now more responsible for the success of the product. They directly control the cost of the contract because they authorize its existence on a recurring basis, every iteration. As Ron Jeffries so succinctly states in a posting on the XP Yahoo discussion board on July 1, 2004:

> Right now, this appears to be a 200-point project. Based on our performance on other projects (or a random guess), with N programmers on it, and your intimate involvement in the project, a project of this size will take between 4 and 6 months. However, we will be shipping software to you every two weeks, and we'll be ticking off these feature stories to your satisfaction. The good news is that if you're not satisfied, you can stop. The better news is that if you become satisfied before all the features are done, you can stop. The bad news is that you need to work with us to make it clear just what your satisfaction means. The best news is that whenever there are enough features working to make the program useful, you can ask us to prepare it for deployment, and we'll do that. As we go forward, we'll all see how fast we're progressing, and our estimate of the time needed will improve. In every case, you'll see what is going on, you'll see concrete

> evidence of useful software running the tests that you specify, and you'll know everything as soon as I know it.

With agile projects, the customer drives the priority of the requirements, so one benefit to your customer is that the priority can change at any given moment (except for the iteration at hand). No more uncomfortable change control meetings. We can trade a new requirement for a requirement not yet started that is of equal value.

Also, the customer has the ability to stop production at any time he feels that the product meets his needs, and he will see new functionality at the end of each iteration, which provides for a natural decision point. This does not mean that there is no overall plan going into the relationship—it would be very difficult for the vendor to do their resource planning if they had no targeted end date! Release or quarterly plans are defined at a high level and revisited throughout the engagement.

You may still run a fixed time/fixed scope project using agile, and you may still ask for prioritization of the requirements, even though they are fixed up front. You may still ask if the customer is interested in an early review of functionality (they usually are). Jeff Sutherland has done several presentations on agile contracting titled "Money for Nothing and Your Change for Free," where the vendor starts with a fixed price contract that includes T&M clauses for changes. If the customer agrees to work with the team every iteration in an agile fashion, then they get "changes for free." The "money for nothing" clause allows the customer to terminate the contract early if value has been satisfied, for 20% of the remaining unbilled contract value.[10]

I Don't Have Enough Time To Work With The Team Every Iteration

We hear from product owners, especially, that they are too busy working with the end users to have the time necessary to dedicate to the delivery team. Granted, agile puts a new set of job requirements on the product owner; whereas before agile they didn't really have to interact with the team much, except for in the beginning when scope was being identified and solidified.

The collaborative nature of agile – product owner working with the team – can be a stressful situation for the product owner, merely from a time management perspective. Try to understand that the product owner is going through the agile change just like everyone else on the team, and try to partner with him or her in order to keep the backlog staged for the team's next iteration. One popular alternative to working daily with the team is for the product owner to establish office hours, like university professors do. This way the team knows precisely when and how to reach the product owner. Obviously, the more office hours the product owner can arrange, the better it is for the team and the product.

I Can't Wait An Entire Iteration For That Feature!

The idea of the iteration is deliberate in its attempt to control the anarchy associated with forever fluctuating requirements. This chaotic environment, one that does not allow the team to focus, often means that many things get started, but none are completed. The idea of the timebox – the iteration – was created as a way to control the chaos in complex project, as well as forcing the business to keep its work in a backlog, staged and ready to be worked upon by the team. Using both of these concepts together allows the right product to be created iteration to iteration.

[10] Jeff Sutherland, "Agile Contracts: Money for Nothing and Your Change for Free," http://jeffsutherland.com/scrum/2008/10/agile-contracts-money-for-nothing-and.html. Slides available here: http://jeffsutherland.com/scrum/Agile2008MoneyforNothing.pdf.

Product owners, customers, and other stakeholders can get antsy when forced to wait an entire iteration for work on the new requirement to begin. You may want to discuss with them the possibility of shortening the iteration length to better accommodate their needs. Usually, once the stakeholders see that delivery is reliable and consistent, this pressure subsides.

Selling to Other Departments in the Organization

When new agile teams begin to mature they become responsive, create working software reliably and consistently, and establish a flow of product creation. This inevitably starts to affect other departments in the organization. The finance managers may have to rethink how they determine capitalization of agile projects, matrixed businesses will want to look at reorganization, and production departments and customers may not be able to keep up with the speed of the delivery teams, just to name a few examples.

While we cannot cover every scenario in this paper, we urge you to work with the other departments to discover their needs and then work with the team to see how to best meet those needs. Additionally, the team may want to invite certain representatives from these organizations to a retrospective or working meeting in order to identify and work out the issues together.

The ultimate goal is to provide reliable delivery to customers and to respond accordingly to their needs. The other departments in the organization are part of that value stream – all of the steps required to bring a product or service from raw state through to the customer – and it is important to work together to understand how to make that stream flow as efficiently as possible.

Other Ways to Sell Agile

Information must be communicated repeatedly in different forms before it takes hold and is ingrained in the minds of the message recipients. We hear teachers say that their mantra around education is to teach a new concept “seven times, seven ways.” Sometimes, even the most basic message – a product vision, for example – must be communicated repeatedly until the team and other stakeholders can really internalize it.

We wanted to outline some ideas for you to think about in communication of agile or the artifacts of the teams. Always err on the side of over-communication. The alternative just isn’t sufficient.

- **Socialize, Don’t Evangelize** – Nobody likes to have change forced upon him or her. Sometimes it’s better to simply share your ideas about agile processes without using the word “agile” and without demanding change. Just make some suggestions for ways things might be done in a more effective manner, using terminology they readily understand, and see what happens. One way to do this is to simply make some changes in your own team – when they start delivering product faster than expected, people will notice and will come to you to find out how you did it. That’s the time to share.
- **Brown Bag Sessions** – These lunch sessions are intended to gather anyone who may be interested in a particular topic. It is a wonderful way to promote education about agile (or any other topic) in a comfortable, informal session.
- **Entice Colleagues to Join You at Local Chapter Meetings** – If you’re lucky enough to live in a city with a local agile chapter (see www.agilealliance.org, www.scrumalliance.org or www.apln.org) you’ll find they usually have monthly meetings, and some even provide free sodas and pizza.

- **Get an Outside Consultant to Bring in the News** – Sometimes, hearing the message from an objective third party is more effective than hearing the message from within. An experienced coach can help the team identify areas of improvement and can communicate difficult messages to others in the organization.
- **Don't Bother Trying to Sell Anything –** As Jim Highsmith once said in response to a question asked of him at a conference, "Don't bother trying to sell agile to management. Just do it. They don't care what you're doing anyway." Management usually doesn't care, as long as you deliver working software when you say you will. So go stealth. Do as much as you can and see what happens.

Part of selling, persuading, and convincing is knowing enough about your audience to do this effectively. We recommend that you study teambuilding, team dynamics, motivation theory, Myers-Briggs, etc. in order to prepare yourself for the situations that you may encounter in building teams and working with the rest of the organization. After all, agile is about people, and the more you know about people, the better off you and the team will be. Sometimes it's as simple as listening and creating a safe environment for change, and not trying to sell anything at all.

Summary

The takeaways from this paper include the following:

- Selling agile is usually best done by not selling per se, but instead by listening, offering alternative solutions, and by being (or providing) a great example of what agile teams can accomplish.
- When presenting agile to the team, they will usually object to the number of meetings, having to do gross-level estimating, the perceived lack of architectural planning, and the issue of needing a co-located team. While all of these can be addressed, don't forget to look for the hidden objections—like fear of change—that may be driving the resistance.
- Management resistance is frequently centered around the perceived loss of control and personal identity. While objections like the concern for lack of long-term planning, missing Gantt charts, and matrix organizational issues can all be addressed, other issues may be deal-breakers: insistence on the success of the current method, lack of trust, or refusal to change for personal/political reasons.
- Issues of resistance from the business or customer that can be addressed include a reluctance to contract on a T&M basis, a lack of time or willingness to spend it with the team, and a need to make changes at the drop of a hat rather than on an iterative basis. All of these issues can be addressed and compromises can be reached, as long as the customer is willing to participate in the agile process.
- Other methods of selling agile include socialization, brown-bag sessions, attendance at local chapter meetings, outside consultation, and simply not bothering to sell at all (go agile in "stealth" mode).

About the Book

When software development teams move to agile methods, experienced project managers often struggle—doubtful about the new approach and uncertain about their new roles and responsibilities. In their book *The Software Project Manager's Bridge to Agility*, Michele Sliger and Stacia Broderick have built a bridge to this dynamic new paradigm. They show experienced project managers how to successfully transition to agile by refocusing on facilitation and collaboration, not "command and control."

The authors begin by explaining how agile works: how it differs from traditional "plan-driven" methodologies, the benefits it promises, and the real-world results it delivers. Next, they systematically map the Project Management Institute's classic methodology-independent techniques and terminology to agile practices. They cover both process and project lifecycles and carefully address vital issues ranging from scope and time to cost management and stakeholder communication. Finally, drawing on their own extensive personal experience, they put a human face on your personal transition to agile—covering the emotional challenges, personal values, and key leadership traits you'll need to succeed.

Copyright Disclaimer:
This chapter is excerpted with permission from the book *The Software Project Manager's Bridge to Agility,* authored by Michele Sliger and Stacia Broderick, and published as part of the Addison-Wesley Professional Agile Software Development Series, Copyright 2008 Pearson Education, Inc. ISBN0321502759. For more information, please visit: www.awprofessional.com. It may not be reproduced, in whole or in part, without express permission from the publisher.

About the Co-Authors

Michele Sliger has extensive experience in agile software development, having transitioned to Scrum and XP practices in 2000 after starting her career following the traditional waterfall approach. A self-described "bridge builder," her passion lies in helping those in traditional software development environments cross the bridge to agility. Michele is the owner of Sliger Consulting Inc., where she consults with businesses ranging from small start-ups to Fortune 500 companies, helping teams with their agile adoption, and helping organizations prepare for the changes that agile adoption brings. A frequent conference speaker and regular contributor to software industry publications, Michele is a strong advocate of agile principles and value-driven development practices. She is a certified Project Management Professional (PMP®) and a Certified Scrum Trainer (CST), with an undergraduate MIS degree and an MBA. Visit her website www.sligerconsulting.com to learn more about agile.

Stacia Broderick has worked as a project manager for 14 years in commercial manufacturing and software development. Stacia was fortunate enough to be cast in the role of ScrumMaster in 2003, and ever since has helped teams all over the world embrace the principles of and transition to agile. Stacia founded her company, AgileEvolution, Inc., based on the belief that agile practices present a humane, logical way for teams and companies to deliver products. In addition to being a Certified Innovation Games Facilitator, she is a Certified ScrumTrainer and Practitioner and a PMP®, a mix that proves helpful when assisting organizations' transition from traditional to modern practices. Visit her website www.agileevolution.com to learn more about agile.

Bibliography

Beck, Kent, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. "Principles behind the Agile Manifesto." http://www.agilemanifesto.org/principles.html.

Cohn, Mike. *Agile Estimating and Planning*. Upper Saddle River, NJ: Pearson Education, Inc., 2006.

Cohn, Mike. "The Dark Side of Multi-Tasking." Better Software, July/August 2005.

Leffingwell, Dean. *Scaling Software Agility*. Boston: Addison-Wesley, 2007.

The Standish Group International. *The Standish Group's CHAOS Report 2004*. West Yarmouth, MA: The Standish Group, 2005.

Sutherland, Jeff. "Agile Contracts: Money for Nothing and Your Change for Free." http://jeffsutherland.com/scrum/2008/10/agile-contracts-money-for-nothing-and.html.

Womack, James P. and Daniel T. Jones. *Lean Thinking*. New York, NY: Free Press, 1996.