Kenny & Company
Management Consulting

Using Agile Methods for Enterprise Software Implementations

Thirty-five percent of IT departments follow agile delivery methods, according to Forrester [1]. No, they're not just following iterative methods and calling it agile, but a little over twenty percent are and only thirteen percent are still following a waterfall method, including the famed CMM. These thirty-five percent are adopting a working style that empowers our highly skilled and highly paid technology professionals to solve business problems creatively and autonomously in self-managing, self-organizing, accountable teams. Beyond IT, product development at software companies all over the world have also begun to abandon the barriers of innovation and speed moving to Scrum teams that consist of people from product manager to test engineer.

Continuing to manage projects following a waterfall method invites failure and inconsistent customer satisfaction. In fact, Dr. Winston W. Royce, who sent us all on the waterfall course explicitly stated, “I believe in this concept, but … [it] is risky and invites failure.”[2] Too bad we've spent the last fifty years holding tight to the belief that we can manage software development with a project plan that presumes it can predict how long it will take us to elicit requirements and design a solution that meets the customer needs.

At this point you're probably thinking, “Yeah, it's easy for our R&D and IT teams to follow agile as they don't have an external client to deal with.” But, we believe you can and here's how.

Why should a professional services organization adopt Agile?

Several years ago Darin was mentoring a young consultant who was ambitious and eager to make her mark on the world but had become disillusioned quickly after her brief project experience. She couldn't understand how so much time was spent on planning, discussing what should be done, and ruminating on how it would be done. When she had started the actual work, she found that team after team would fall behind schedule and reduce scope or cut quality to meet deadlines. Sound familiar? She shared a great metaphor outlining how the project was progressing.

About Us

Kenny & Company is a management consulting firm offering Strategy, Operations and Technology services to our clients.

We exist because we love to do the work. After management consulting for 20+ years at some of the largest consulting companies globally, our partners realized that when it comes to consulting, bigger doesn't always mean better.

Instead, we've created a place where our ideas and opinions are grounded in experience, analysis and facts, leading to real problem solving and real solutions – a truly collaborative experience with our clients making their business our business.

We focus on getting the work done and prefer to let our work speak for itself. When we do speak, we don't talk about ourselves, but rather about what we do for our clients. We're proud of the strong character our entire team brings, the high intensity in which we thrive, and above all, doing great work.

What's Inside

Why should a professional services organization adopt Agile? .... p 1
How do you convince a client? .... p 3
Estimating the Work .... p 4
Statement of Work .... p 5
Staffing & Training .... p 5
Project Tracking & Governance .... p 6
Quality Assurance (Testing) .... p 7
How do you get started? .... p 7
About the Authors .... p 7

For a complete list of **Kenny** & **Company** publications and information about us, please visit our website www.michaelskenny.com or email us at info@michaelskenny.com.

> *"We started the project getting the client really excited about what we were going to deliver. It was a Lexus. It was going to solve all their problems and get them there in style. Once the project began, however, during requirements gathering we realized that we wouldn't be able to build the Lexus in the first release given the timeline that was committed to. We would be able to build a Toyota Camry though, and it was pretty much the same thing as the two shared the same frame, body, engine and other parts. They'd just have to let go of a few bells and whistles, and if they did, we'd be right on plan. On it went through the software delivery life-cycle where at each stage it seemed necessary to reset expectations and continue to downgrade. By the time testing began we were now only committing to a Toyota Corolla. When UAT began, we realized that Release 1.0 was not going to work and that it would now only consist of the frame and body, and the engine wouldn't be put in until Release 1.1, when we'd be able to do a soft launch."*

All too often we work with our clients enthusiastically espousing how our software solution is going to radically transform how they do business. What we don't admit is that it's difficult to change a company's processes, gain buy-in from diverse stakeholders and more often than not our systems are not as flexible as we need them to be.

An agile approach follows an empirical process model that requires frequent inspection and adaptation, recognizing that software development is not like stamping out toys on an assembly line. Rather than spend a client's money and time planning, you begin the work focusing on the highest priority items that will deliver business value. When those items are delivered, you move on to the next group of prioritized items. The outcome is that you deliver the Lexus in Release 1.0.

How do you convince a client?

Professional services organizations, consultancies, law firms, ad agencies, and any other services business have one thing in common – the contract. We call them work orders, statements of work, services contract, work request, etc. It's our way of comforting a client given what we are selling is intangible. It's not an invoice that comes after the work has been done. That would be all we needed if they were buying a hard-good from us as they'd be able to see it, touch it and say, "I want two hundred." No, it's not easy to convince a client to trust that we're going to deliver on a promise given none of the work can be seen and in most cases understood. So, we draft a piece of paper that attempts to illustrate what we jointly agree the services will be. And when it comes to software, we're asking them to imagine something they've never seen and then sign on the dotted line to indicate they understand it well enough to commit to a long engagement. Let's be real, all you're selling at this point is your reputation as both parties know that the piece of paper doesn't even come close to defining what they're going to get from the engagement.

An agile approach, however, only requires the client to hold their breath for one sprint at a time. You will continue to take time during the sales process to educate them on your approach. You'll walk them through how you will take a backlog of work and following their prioritization deliver work in small increments that demonstrate the software working, not provide hundred page specifications that they have to then envision what the software will look like, but real working code. This is dramatically different. This is a new commitment. This says, "I'm going to focus your money and energy on getting you up and running so we can solve those business challenges we convinced you our software solves."

Still not convinced? Ask them to sign up for two sprints. If your software is a large, traditional implementation then your sprints are probably going to be thirty days each. This means you're only asking them to hold their breath for thirty days and will only require an investment of a small portion of the engagement. If they don't see results and feel engaged more than on any other project they've had, then let them terminate the agreement for convenience. We're convinced that if you follow agile methods, you'll demonstrate value in the first sprint and they'll be hooked. They'll be hooked because they're going to see a level of transparency they've never had before and the data will be real. They'll be hooked because you're going to deliver working code after the first sprint. No, not Visio diagrams, HTML mock-ups or wire frames, but real, working software.

Practically speaking, you should break the work up into two engagements. The first is your Discovery Phase. During this short engagement you should have pre-packaged questionnaires, process flows and a vanilla version of your system to demonstrate. The exercise then would be to walk through your system similar to how you would in a detailed RFP life-cycle, but the focus would be to identify the areas that may need customizations to meet the client's needs and gather the basic information that you need to understand what the effort will be to configure your system. We're making the assumption that it's packaged software (a.k.a., Commercial off the Shelf [COTS]) and so you should know what areas you have to setup and configure. We have seen many companies use a lengthy questionnaire that can be completed by the client themselves, to gather the key insights to allow the team to estimate the engagement. These details now become the backlog. Once you have this you can facilitate a prioritization with the client such that you finish the phase knowing what the work is you team needs to do and have something to estimate.

Estimating the Work

As with other areas of this article, we expect you to have read about agile and Scrum, and in this case we would recommend that you dive deeper into the books on estimating with agile. But, we do want to provide some highlights around the approach. Estimating software implementation efforts is hard. No methodology in the world can predict what your client's demands and dreams will be once they become more familiar with your software. The difference between agile methods and traditional waterfall methods is agile approaches don't pretend they can. Estimating following an agile approach is composed of two main processes. The first is when you are estimating backlog items and the second is when you're estimating throughput of completing a grouping of backlog items.

Following the workshops and questionnaires of the Discovery Phase, you'll have a backlog of items that represent specific configuration tasks and customizations. Each of these needs to be estimated for not only the "dev time", but all the time related to completing the task from further requirements gathering, to integration testing and deployment. Teams use a method of comparison to something they know and use a metaphor such as "t-shirt sizes" to keep them focused on referencing something they know versus the new subject material. An example is when they look at a backlog item that requires them to setup the login screen. They may see that there are no changes to the software and thus be able to flag that item as Extra Small. In another case, they may be looking at an item that is typically a couple days of work, but in this case there are a number of customizations. So, what might normally be flagged as Medium or Large may now be increased in t-shirt size. The t-shirt sizes do represent time and thus later can be converted to hours or days. It will still remain difficult to estimate the throughput and so teams are best served using a comparative method that references past experiences with similar clients. Once the project begins, however, estimating the completion time becomes increasingly accurate as you inspect actual burn-down of the backlog and the velocity of work completed. This becomes valuable when the client makes a request for something new in the middle of the project.

Regardless of what methodology you're using today, your team should have a list of tasks or deliverables that if checked off as delivered mean your system is setup. Use this list to keep track of estimates and "actual" from client to client and estimating will become easier and easier for your team during the fast paced sales process.

Statement of Work

Well, we still need a contract. And yes, introducing a new methodology is likely going to be difficult – at least until everyone wakes up and starts challenging waterfall contracts more. Until then, we propose a few modifications to your contract to help alleviate client concerns. The first is to define Scrum and agile in the document itself. Include an overview of the process, your client's involvement and highlight the level of transparency provided with this approach. We used illustrations to show how the sprints work pointing out that they get to have a formal review of working software at the conclusion of each sprint. Emphasize that they will also be able to control the prioritization and change direction as you go – something you won't find in a waterfall based contract. Include the backlog and each item's estimate. This should be translated to engineering hours or days, whichever is most relevant to your efforts.

A key selling point to agile is that it is much more flexible to change. Rather than scare everyone into the stone tablets that are typical requirements of waterfall engagements that require senior councils to approve any change once the project has begun and typically a major "sign-off" of a change order or revised contract, agile methods allow the client to add, modify and remove items from the backlog as they go. So, if they get into the project and having seen demonstrated working software after a few sprints realize that your solution is amazing and they just want to get using it and no longer are caught up on all their customizations, those items are simply de-prioritized and not executed on. Now, you've probably been wondering at this point, "yeah, this sounds all fine and good, but my client has to get approval for a budget and there isn't all this infinite flexibility!" You're right, which is why we propose introducing contingency. Yes, we still need contingency. The difference here is that we're only going to use it if the client approves it versus adding in an arbitrary number to each waterfall phase because we know we never get through it like we predict. In this case we create a bucket for "refundable contingency" that gives the client some flexibility later to change the prioritization of a backlog item and / or add new scope once they've become more educated on our system. The best way to use this is to keep the client focused on using the software as intended and minimize customizations. When in the heat of the moment you can say, "we can definitely add a new backlog item to the next sprint that will allow for that customization if you approve us decrementing the refundable contingency bucket." So, the client gets a contract with one number, which is really made up of what you believe the work to be plus a bucket of refundable contingency so that they can make their budget ask with some room. Can you always get this? No, but it's worth a try and a lot easier to get approval for when they don't just think contingency is being added to each scope item. Remember Parkinson's Law: "Work expands so as to fill the time available for its completion."

Again, if they're not ready to sign up for the whole engagement, ask them to sign up for two sprints, then prove it.

Staffing & Training

Agile teams require deeper skills and competency than most managers are ready to recognize. They also require more multi-discipline resources that are comfortable doing different tasks. In a scrum team, the whole team commits to delivering the scope of a sprint, how they get there is up to them. The majority of coding tasks will be done by the developers and the majority of testing tasks will be done by the test engineer, but you need test engineers that can jump in and help coding if the team falls behind and developers that test their code. Additionally, you need to continue to invest in their skills. Every company should be doing this anyway, but let's face it, most don't invest in their human resources. We spend tons of money upgrading computers, changing out old furniture, upgrading plants and equipment, but rarely have the same focus on our people. Agile teams require deeper expertise and thus need more investment. The benefit is that you accomplish significantly more output

Recommended Reading:

- Succeeding with Agile: Software Development Using Scrum by Mike Cohn
- Agile Project Management with Scrum by Ken Schwaber
- Agile Estimating and Planning by Mike Cohn

with significantly less people. You'll find scrum teams of seven to nine people that can outperform a forty person team from the top system integrator in the world. So, if your team uses a scripting language to configure your software, or have to code in Java typically or just use your proprietary configuration tools, make sure they are experts in these skills and invest in this over time.

Staffing an agile team requires breaking down boundaries between disciplines. It means a team must work together, even though team members may have reporting relationships that go up to different executives in your company. This becomes very difficult for most managers as they begin to worry about where they fit into this new world, which is unfortunate because if they were doing their jobs they'd realize their value is not in trying to control the flow of information or who works on what, rather they'd focus on improving the talents of their team by designing training programs for them, coaching them in their day to day work and finding new ways to share their skills across other teams so that everyone begins to build some multi-discipline experience. This is a huge job and if done right can produce truly high performing teams.

The other key to staffing an agile team is to create a new working environment for these teams. The cube farms and closed door offices don't inspire creativity, innovation or collaboration. You need open work spaces, sometimes called "team rooms" where each team member can see one another, work together and collaborate on the commitments of each sprint. This doesn't mean throw everyone in bullpens and save a ton of money on office space. You still need private areas and people need quiet places.

And most importantly, get everyone trained on the agile methods you're adopting. If you choose Scrum as your project management framework, which is becoming the standard, then get everyone trained. Even if they won't always play the role of scrum master, they all should be able to step into it and know how the process works. For those that will play a more traditional role of business analyst or client engagement roles, get them trained in what is called the product owner role, this role manages the backlog and helps the various stakeholders of a project to define their requirements. Engineers should also get training on software test automation tools as this is a critical pillar of agile delivery teams.

Project Tracking & Governance

Most people think agile is a free for all. They think it means developers get to just begin coding without requirements defined and no documentation. What most don't realize is that agile methods actually have more planning and with more team members than any other method. And yes:

- You still need a project work plan; the architecture of it is just changed
- You still need to track what everyone is working on and assigned to
- You still get to track estimates and actuals,
- You still get to keep track of deliverables and milestones
- You will have amazing status reports, and
- You still have change control.

Project plans are your backlog. You can pre-arrange the backlog items into buckets that you believe will be future sprints to allow you to forecast roughly when the engagement might finish. This also allows you to illustrate to the client the order that might be followed. These tasks though should be deliverable-based rather than activities-based. This allows you to demonstrate "earned value" rather than simplified metric of percent complete. The real value of agile in project tracking is the level of transparency provided. Prior to each sprint, a sprint backlog is created that is a decomposed group of backlog items that are made up of smaller tasks or work items. Each day the team then marks off only those that are complete, not partially complete, but 100% ready for deployment. This allows the team to provide a "burn-down chart", daily illustrating progress. This along with standard impediment logs (a.k.a., issues log) can be aggregated to provide a daily status report. Each backlog item has an effort estimate and during the sprint planning meeting, team members assign themselves to each item and so you also have visibility into who's working on what.

Within a given sprint the rule is that you don't introduce change. This is typically not met with too much resistance as the client only has to leave the team alone for a max of 30 days. Any changes proposed are converted into backlog items and then prioritized against everything remaining. If the team is completing items ahead of plan, then you may be able to absorb the new backlog item. If the new or modified item introduces more effort than originally planned, you would then remind the client of their refundable contingency and ask if it's important enough to decrement that budget. This activity alone often gets them thinking more about business value of their ideas than any other approach you've used in the past. It puts the onus on them to make the call.

Quality Assurance (Testing)

Testing typically makes up most of the effort of any project. Many project managers like to pretend that testing will be some percent of development and most executives believe it should be less effort as we all want more features, less overhead. The reality is testing often takes almost twice as much effort as development and breaks schedules because of two reasons. The first is that developers in the waterfall approach get trapped in the telephone game and have to make guesses at what the client wanted and focus on meeting schedule milestones over validating assumptions and thus cut corners when it comes to the quality of their work. It will all get caught in testing, right? Secondly, when the project does decide to move to the test phase(s) of the project, the team typically has planned for manual testing that takes a significant number of resources and hours to repeat tests over and over after each bug is resolved. To become an agile team requires the team to validate their assumptions often with their client through daily check-ins and formal sprint review meetings and requires a team to automate their tests so that they can be run daily and constantly to continually verify the system works as expected.

This will require test engineers that have deeper technical skills and can build tests in the various automation tools. This also requires a better understanding of how the code actually works so that they create efficient tests rather than only black box tests because they don't know that executing the four different scenarios is actually just executing the same function in the code four times.

The goals of an agile team are not to produce documentation and test cases and scripts, but rather to demonstrate working code that is potentially shippable after each sprint. This requires test driven development and automated tests to be built in the beginning. It also means a lot less overhead in your test management efforts as the team is more focused on making the system work then showing how the dev team missed requirements and/or have bugs.

How do you get started?

Don’t try to do this on your own. Get a coach and some practitioners to join your team. Supplement the teams with people that have done this before and get everyone trained up on the basics. You should also show your commitment by getting trained yourself. Unfortunately there is no “dipping your toe in” when it comes to agile. As a professional services organization, this makes it difficult to begin. If you have an existing client that has follow-on project work, find a way to get one of those projects following this new approach. If you have a project that has stalled out or is failing, stop what they’re doing, create a backlog of what’s left and begin getting it back on track one sprint at a time. And however you start, don’t forget to employ the basic agile engineering practices out of the gate such as continuous integration and automated testing. This is your foundation.

About the Authors

Darin Archer is a Senior Manager of ecommerce at Adobe Systems where he is currently transforming an online team to agile methods. Mr. Archer has been a professional services executive for over 15 years, at ISCS, a software company focused on the Insurance Industry and at Accenture, a Global Consulting & Technology Services and Outsourcing company. At ISCS, he was a key architect of its industry-leading services delivery methodology and directed implementations, technical support, and training, as well as overseeing alliances with service partners. Prior to ISCS Mr. Archer spent over a decade in consulting, where he led large custom and packaged software implementations for Fortune 500 companies. Within Accenture’s system integration and technology practice, he advised clients on product and IT strategy, development methodologies and global sourcing. Mr. Archer received his Bachelor of Science degree in business administration with an emphasis in information systems from the University of Montana.

Will Yen is a Partner and the Chief Marketing Officer at Kenny & Company. He has over 18 years of experience delivering business solutions for Fortune 1000 companies. His range of experience includes supply chain strategy, marketing strategy and planning, product management and development, IT strategy and planning, mobile computing, and financial services software development. Will has been published in Baseline Magazine, Computer Technology Review, and PS Village, and is the author of several research whitepapers and blogs. He holds a Bachelor of Science in Managerial Economics from the University of California, Davis, a Master of Science in Applied Economics from University of Georgia, Athens and a Master of Business Administration from Duke’s Fuqua School of Business.

References:

1. http://agilescout.com/learn-more-agile-software-development-methods-this-year
2. “Managing the Development of Large Software Systems”, Dr. Winston W. Royce, University of Maryland, 1970

This article was first published on www.michaelskenny.com on February 22nd, 2011.The views and opinions expressed in this article are provided by Kenny & Company to provide general business information on a particular topic and do not constitute professional advice with respect to your business.

Kenny & Company
Management Consulting

Kenny & **Company** is a management consulting firm offering Strategy, Operations, and Technology services to our clients.

Who We Are

Partner Led

Our Partners are personally committed to our clients and lead every engagement.

Experience, Perspective and Passion

We average over 20 years in professional services and bring tailored approaches to every client engagement.

Focused, Collaborative, High-Impact

We work side-by-side with our clients in highly focused teams to solve complex business problems.

Client First

Our highest priority is our client’s professional and personal success. We believe clients should expect more.

Guarantee Our Work

We guarantee our clients complete satisfaction every engagement every time.

Contact Information

Firm Headquarters
Serving San Francisco, Silicon Valley & Los Angeles
1710 South Amphlett Blvd.
Suite 302
San Mateo, CA 94402

Northwest Office
Serving Portland & Seattle
707 SW Washington St.
Suite 925
Portland, OR 97205

For inquiries: info@michaelskenny.com

To see additional publications and learn more about us, please visit our website at: www.michaelskenny.com.

Also, follow us on: