NET OBJECTIVES EZINE
VOLUME 2, ISSUE 6 OCTOBER 2005

The Lean-Agile Connection:

Developing Quality Software Efficiently

Alan Shalloway, Net Objectives, alshall@netobjectives.com

There is a new risk in the software development community today. It is building the wrong thing. In fact, the industry now builds the wrong software 64% of the time (or at least 64% of what is built is not used). How does this happen? What can we do to be more effective? We'll talk about that here.

This article is about how to develop software. We'll first look at what we are trying to accomplish when we build software. We'll then investigate what we have to do to do this – including the nature of the problems inherent in building software. This may sound a little basic, but there are many inherent flaws in common approaches to building software because people haven't stepped back far enough to see the big picture. Planting the right tree in the wrong forest does not make for good forest management.

Once we have a clearer view about we are trying to do, we will move on to an overall view of the tasks needed to build software that is useful and cost effective. We investigate different ways these are accomplished with both standard processes and a lean-agile approach. Finally, we come up with a set of core practices that we should follow to build cost-effective software that give us appropriate solutions so we get the return on investment we want and need.

What We Are Trying To Accomplish

Let's start with what we are trying to accomplish when we build software:

About the author --
Alan Shalloway
Alan Shalloway is the CEO of Net Objectives. Since 1981, he has been both an OO consultant and developer of software in several industries. Alan is a frequent speaker at prestigious conferences around the world, including: SD Expo, Java One, OOP, OOPSLA. He is the primary author of Design Patterns Explained: A New Perspective on Object-Oriented Design and is currently co-authoring three other books in the software development area. He is a certified Scrum Master and has a Masters in Computer Science from M.I.T.

Articles and events of interest to the software development community

In This Issue:

NET OBJECTIVES
Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email: info@netobjectives.com

We need to build software that returns value to the business. We might measure this with standard Return on Investment measures (ROI) but clearly we are making an investment and expect a greater return. The return is often measure by how much value we bring to our customers (whether they be internal or external customers). The cost is determined by both our effectiveness in building software and in building the right software. Software that isn't used by our customers (provides no value) is essentially waste. We'll discover later that this is a very large hidden cost in many projects.

We need to build software that the customer wants. Studies by the Standish group show that 64% of the software features developed over the last decade are either not used or only rarely used. This represents a huge waste for the industry. Besides the cost of building features that aren't used, we have to maintain those features after the software is released – causing even more waste.

We need to build software with high quality so that it can be maintained and modified. I would suggest that this is an obvious need, but one that gets lost in the pressures of the development cycle. Poor quality software has many costs. Customers think more highly of products that function well (as well as the companies that developed them) than products that don't. Lower quality software causes lots of extra work in the maintenance of the software and makes it more difficult (expensive) to extend.

We must build software quickly. Not only will this drive the cost of development down because fewer resources are tied up, but we can get a competitive advantage by being able to respond to our customers and competitors most readily.

Be able to adapt to things learned during the development process. In other words, we want to continue to get smarter as we go along.

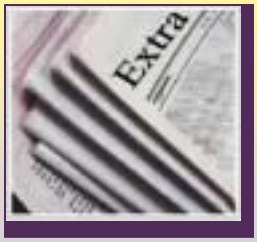
Give management exposure to what is going on through the development process. This enables management to make key decisions such as adding or removing resource from a project, or even canceling the project if it is not going well.

Create teams where people want to work and where the workload is steady. It is important that our people want to continue working with us – and that they are able to do so. This means we need a good, sustainable, work environment.

We need a cost-effective development process. But what does "cost effective" really mean? Given the items above, I'd suggest it must include the cost of maintenance, application extension (new features) as well as the original development cost.

Volume 2, Issue 6
October 2005

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com



Breaking News!

Net Objectives has been selected as one of the 100 Fastest Growing Private Companies in Washington by the Puget Sound Business Journal.
More news in the next mailing.

What Do We Have To Do To Do This?

Before we answer this question, let's look a little about into the problem, or should I say, problems, facing us. We'll take a short look at:

- The process of software development itself.
- Some things we know about teams (don't want an I mentality – anti roles)
- Some things we know about projects of any type (we learn more about them at the end)

The process of software development. In Agile Software Development with Scrum, Ken Schwaber, relates how he brought several systems development methodologies to process theory experts at DuPont – the most highly respected theorists in industrial process control, led by Babatunde "Tunde" Ogannaike. They inspected these processes and were amazed and appalled that our industry, systems development, was trying to do its work using a completely inappropriate process control model. They said systems development had so much complexity and unpredictability that it had to be managed by a process control model they referred to as "empirical". They said this was nothing new, and all complex processes that weren't completely understood required the empirical model.

In a nutshell, there are two major approaches to controlling any process. The "defined" process control model requires that every piece of work be completely understood. Given a well-defined set of inputs, the same outputs are generated every time. A defined process can be started and allowed to run until completion, with the same results every time.

Standard software methodologies attempt to use the defined model, but none of the processes or tasks are defined in enough detail to provide repeatability and predictability. Tunde said our business is an intellectually intensive business that requires too much thinking and creativity to be a good candidate for the defined approach. He theorized that our industry's application of the defined methodologies must have resulted in a lot of surprises, loss of control, and incomplete or just wrong products. He was particularly amused that the tasks were linked together with dependencies, as though they could predictably start and finish just like a well defined industrial process.

Tunde told me the empirical model of process control, on the other hand, expects the unexpected. It provides and exercises control through frequent inspection and adaptation for processes that are imperfectly defined, and generate unpredictable and unrepeatable outputs.[1]

In a nutshell, we must control our software development through an empirical, not a predictive process.

[1] Agile Software Development With Scrum, 2001, Schwaber, Beedle

Personal Book Recommendation from Alan Shalloway:

***Living With Joy: Keys to Personal Power and Spiritual Transformation* by Sanaya Roman. (ISBN: 0915811030) OK, I'll warn you right up front. This is advice purportedly from a non-physical entity. However, don't worry about that – just read it. Great advice and very helpful. The book gave me insight after insight that I have been able to translate into action and a better state of mind.**

Volume 2, Issue 6
October 2005

NET OBJECTIVES

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email: info@netobjectives.com

Some things we know about team. While it is somewhat of a cliché now, there is a lot of truth to "there is no I in team". Good teams have team members concerned more with the results of the team than the actions of the individuals. Cross-functional teams, where people work together to get the desired result, are more likely to produce good results than teams where people work out of well defined, restrictive jobs thinking – "I don't do that because it's not my job".

Some things we know about projects of any time. A simple truth: we know more about a project half way into it than we did at the beginning (and even more at the end). We should find some way to take advantage of this gathering knowledge.

What is Lean-Agile?

Lean-Agile is a combination of best practices from Lean Software Development[2] and Agile Software Development. Lean development principles have been tried and proven in the automotive industry, which has a design environment arguably as complex as most software development environments. Moreover, the theory behind lean development borrows heavily from the theory of lean manufacturing, so lean principles in general are both understood and proven by managers in many disciplines outside of software development.

Lean software development is about the application of lean principles to software development. The main principles of Lean Software Development:

- **Eliminate waste.** Waste is anything that does not add value to a product, value as perceived by the customer.
- **Amplify learning.** Development is an exercise in discovery, while production is an exercise in reducing variation.
- **Decide as late as possible.** Delaying decisions is valuable because better decisions can be made when they are based on fact, not speculation.
- **Deliver as fast as possible.** Without speed, you do not have reliable feedback. In development the discovery cycle is critical for learning: Design, implement, feedback, improve.
- **Empower the team.** Top-notch execution lies in getting the details right, and no one understands the details better than the people who actually do the work.
- **Build integrity in.** Software with integrity has a coherent architecture, scores high on usability and fitness for purpose, and is maintainable, adaptable, and extensible.
- **See the whole.** We need the system to provide value to the customer – not just have locally optimized parts.

[2] The information on Lean software development was extracted from Lean Software Development: An Agile Toolkit For Software Development Managers by Mary and Tom Poppendieck. An excellent book we highly recommend.

Management Book Recommendation:

***The Goal: A Process of Ongoing Improvement* (ISBN: 0884270610) by Eliyahu Goldratt. When I first read this book years ago it had a profound impact on my thought process. Now that I have gotten into Lean Software Development, I am seeing how it is even more useful. The Theory of Constraints is a very important theory to understand and Eliyahu's writing makes it easy and enjoyable to understand.**

Volume 2, Issue 6
October 2005

NET OBJECTIVES
Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-7011
Email :info@netobjectives.com

Check www.netobjectives.com/events/pr_main.htm#FreeSeminars for Public Seminars in Western Washington State, Midwest, and Northern California

Seminars We Can Give

Agile Planning Game – The Planning Game was created by Kent Beck and is well described in his excellent book: Extreme Programming Explained. Unfortunately, the Planning Game as described is not complete enough - even for pure, XP teams. This seminar describes the other factors which must often typically be handled.

Comparing RUP, XP, and Scrum: Mixing a Process Cocktail for Your Team - This seminar discusses how combining the best of some popular processes can provide a successful software development environment for your project.

Design Patterns in an Agile (even XP) Environment – The Object Pool – This seminar presents a project done by following the guidelines of Design Patterns, Agile Development, and Refactoring to show how these ideas can inform each other.

Emergent Design: Design Patterns and Test-Driven Development - This seminar illustrates why design patterns, Test-Driven Development (TDD) and refactoring, which can seem opposed to each other, are actually based on the same principles.

Executive Briefing on Agile Software Development – This seminar describes agile methods from the perspective of the business needs of the company. In other words, instead of focusing on the technical details of agility, this talk focuses on the business motivations for it. These include the need for eliminating waste, understanding the business needs of the customer, how teams can coordinate efforts, and more.

Introduction to Scrum - This seminar gives a brief description of the philosophy behind Scrum, the Scrum method, the roles involved, and additional topics that augment and enhance Scrum. The attendees leave with an understanding of how and why Scrum works, and hence why Scrum is a popular management method.

Lean Software Development – Lean Software Development builds on the principles of lean manufacturing – the techniques that Toyota and Honda pioneered in the 80s. One thing that differentiates Lean from other agile approaches is that Lean is very management friendly. This seminar will give an overview of how to apply lean principles to software development.

The Product Owner/Customer Role in Agile Development – Agile development has proven itself to be successful, especially with small teams of 6-10 developers. The existence of a qualified, empowered, Product Owner / Customer is one of the major success factors. In this seminar we describe the basic responsibilities of the Product Owner / Customer, challenges and clues about how to solve them.

Transitioning to Agile – More and more companies are beginning to see the need for Agile Development. In this seminar, we discuss what problems agility will present and how to deal with these.

If your user group or company is interested in Net Objectives making a free technical presentation for them, or if you would like to be notified of upcoming Net Objectives events, please visit our website, or contact us by the email address or phone number below.

www.netobjectives.com • mike.shalloway@netobjectives.com • 404-593-8375

Volume 2, Issue 6
October 2005

NET OBJECTIVES

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email: info@netobjectives.com

Agile software development is completely consistent with these principles but focuses more on the project management, design, coding and testing aspects of software development. It believes in developing software iteratively, with as much feedback as possible with the customer and maintaining high quality of code through automated testing techniques.

There is both a relationship between these two methods and, in fact, all of the aspects of software development that they suggest. An holistic view of software development defined by Lean and Agile is shown in Figure 1:

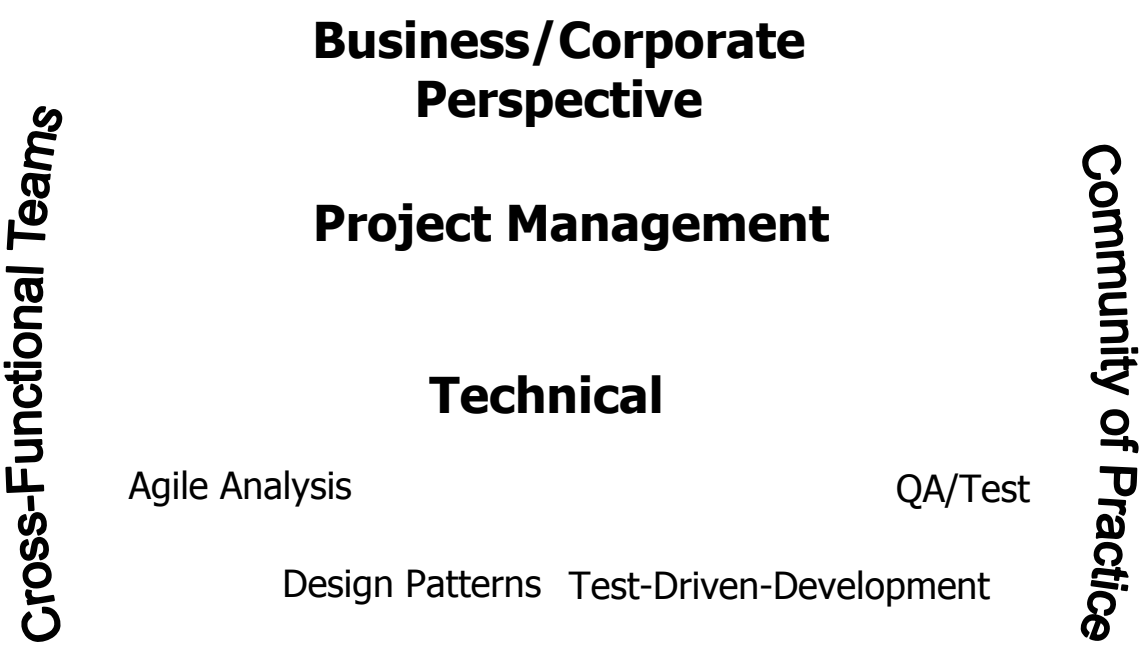


Figure 1: An Holistic View of Lean-Agile Development

Employee Spotlight: Alan Chedalawada

Alan has been with Net Objectives for over a year as Senior Consultant in the Consulting division, but the need and his experience mandated that his role expand to COO.

Alan brings 25+ years experience at various corporations ranging from entrepreneurial startups to Fortune 1000 companies. His experience spans the Manufacturing, Telecommunications, Technology, Finance, Energy, and Distribution industries.

Alan is a Certified Scrum Master, a member of TIE – The Indus Entrepreneurs organization, and a graduate with honors from Columbia University's Computer Technology & Application Masters program.

Volume 2, Issue 6
October 2005

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com

We will explore these more fully below.

What Do We Have To Do?

Deciding whether to start a project or not. If we are guiding our decisions as to whether to do a project based on ROI, it seems clear we need a good idea of the cost of the project. Unfortunately, having a high view of what the customer wants often leaves many risks that can't really be seen early on. Standard approaches tell us to learn as much as we can about a project early on. While we don't disagree with this, we think that this means learn something about design and implementation as well as what the customer wants. Agile methods often include the concept of a spike – or as other call a "tracer shot" through the application. This technique of building a very thin scenario of the application can often be very useful in learning about the true cost of building the system. Looking at a system for months and months, trying to decide what and if you should do it often results in much less information than you would hope.

Lean techniques such as value-stream mapping (to determine where wastes are in the process) and creating multiple profit and loss statements for different scenarios also can be invaluable in deciding whether to proceed with an application or not.

Figuring out what the customer wants. It is virtually axiomatic in our industry that customers change their minds. Instead of resisting this or trying to prevent it, we have to accept it. Customers change their minds about things for good reasons. As the project proceeds:

- They understand their needs better.
- They learn new things that are possible.
- They learn about how computer systems and their processes interact.

This is true for the development team as well. As the project proceeds:

- They understand the problem domain better.
- They come up with new ideas that can be useful to the customer.
- They learn how to implement features faster.

The fact that our customers and our developers are getting smarter as time progresses should be considered to be a good predicament, even though it implies new ideas will generate new requirements on an ongoing basis.

Build the software with high quality. Low quality software is problematic at all levels. Defects that get released not only causes much

Volume 2, Issue 6
October 2005

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email: info@netobjectives.com

More Seminars We Can Give

Agility and Ceremony: How Can They Co-Exist? – This seminar provides a short overview of Agility and the challenges one faces in a High Ceremony environment.

C# for Java and C++ Developers – This seminar will give a complete overview of the major language features of C#, and will also examine some best-practice issues.

Design Patterns and Extreme Programming – Patterns are often thought of as an up-front design approach. That is not accurate. This seminar illustrates how the principles and strategies learned from patterns can actually facilitate agile development. This talk walks through a past project of the presenter.

Effective Coding Practices – This seminar introduces three straightforward coding practices that support design patterns, refactoring and test-driven-development: Encapsulating Construction, Coding by Intention, and Considering Testability before Coding.

Effective C# – This seminar is intended for programmers who are not object-oriented experts and who want to learn how to use C# effectively.

Introduction to Use Cases – In this seminar we present different facets of the lowly Use Case; what they are, why they're important, how to write one, and how to support agile development with them.

The Need for Agility – This seminar is designed to demonstrate how to balance the necessity for up-front analysis and design with the need to get feedback about how the project is going.

Pattern Oriented Development: Design Patterns From Analysis To Implementation – This seminar discusses how design patterns can be used to improve the entire software development process - not just the design aspect of it.

Test-First Techniques Using xUnit and Mock Objects – This seminar explains the basics of unit testing, how to use unit tests to drive coding forward (test-first), and how to resolve some of the dependencies that make unit testing difficult.

Unit Testing For Emergent Design – This seminar illustrates why design patterns and refactoring are actually two sides of the same coin.

Use Case Driven Agile Development - Capturing functional requirements with Use Cases is a software development best practice. Agile development processes are proving themselves to be effective. They work very well together by combining the iterative nature of agility with incremental development of use cases, as this seminar explains.

The XP Planning Game and Iteration Retrospectives – Tools, techniques and tricks that aid iteration planning and team learning will be discussed.

If you are interested in any of these offerings, if your user group or company is interested in Net Objectives making a free technical presentation to them, or if you would like to be notified of upcoming Net Objectives events, please visit our website, or contact us by the email address or phone number below:

www.netobjectives.com • mike.shalloway@netobjectives.com • 404-593-8375

Volume 2, Issue 6
October 2005

NET OBJECTIVES

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com

wasted effort, it also lowers the image of the group that developed it. Low quality software is also more expensive to maintain and to extend. We therefore pay a quadruple cost with poor quality, maybe worse when you consider that poor quality tends to demoralize teams as well.

Be risk aware. Risk mitigation is a critical aspect of software development. Saying bad things won't happen is not the proper way to mitigate risk. In today's development world, one of our biggest risks is building software the customer won't want at the end of the effort (or, more likely, that there will be some other software that they want more). Another big risk is the software will take significantly longer to build than expected. We'll find that Lean-Agile methods can help mitigate these risks better than standard methods.

Know where we are. It is important that management understand where a system is, how well it is progressing and how far it must go to complete. If we can't provide management with this information, they really can't manage (they can only aim, shoot and hope for the best). We've all been (or at least heard of) projects that go well for several months but then in a 30 day period go from being on schedule to being several months behind schedule. Assuming people weren't actually sabotaging the project, it is most likely that the team really wasn't as on schedule as they thought.

Create a cross functional team. Since software development is a team effort, effective teams are very important. Creating stove-piped teams that have separate metrics and different interests does not work as well as one team with a common goal. Again, we'll see that this is a critical aspect of lean software development.

How Do We Accomplish This?

Comparing standard techniques with Lean-Agile techniques. Standard techniques ignore the advice of systems experts and assume that, in fact, we can get repeatability and predictability of a very complex project (in the face of repeated evidence to the contrary). By focusing on analysis and architecture early in the project, standard methods make it very difficult to get feedback from customer (because nothing is built for the customer to see for quite some time). Lean-Agile methods take a different tact – get feedback from the customer as soon and as often as possible. Lean-Agile methods revolve around the concept of building software in iterations. Each iteration is a cycle of detailing some requirements, building it and verifying with the customer that the team has built something useful. Because this causes code to change on a regular basis, there is also a very strong emphasis on code quality being as high as possible. Hence, automated testing and the use of proven design/coding techniques (such as design patterns and refactoring) are espoused.

Let's continue this contract in methods by looking how standard methods and Lean-Agile methods attempt to accomplish the seven items we suggested in the "What We Are Trying To Accomplish" section.

We need to build software that returns value to the business. Standard methods assume we can know all the right things to build at the beginning of the project as well as which aspects

> *"There are two things to aim at in life: first, to get what you want; and, after that, to enjoy it. Only the wisest of mankind achieves the second."*
> *-- Logan Pearsall Smith*

Volume 2, Issue 6
October 2005

NET OBJECTIVES
Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com

of the system will return the greatest value. Lean-Agile methods assume that we may know the general big-picture of things, but that we want to take advantage of what we learn as we go.

We need to build software that the customer wants. I have heard it said that customers can't tell you what they want until after you've shown them what they don't want. Although anecdotal, it seems to ring true for many of us. Building software that can evolve as we understand a problem is better than assuming we know all there is to know at the beginning. While this causes some problems, these problems can be handled. In Figure 1, the Technical section had four aspects to it:

- Agile analysis – learning what requirements are over time
- Design patterns – maintain high code quality through the use of best practices
- Test-Driven Development – ensure high code quality through good design and tests
- Quality Assurance / Test – have QA define our requirements and ensure we are implementing them

By using the above techniques properly, we can, in fact, evolve code as our understanding of the problem expands.

We need to build software with high quality so that it can be maintained and modified. Ironically, both standard and Lean-Agile methods espouse this. They take two completely separate tacts, however. Standard methods assume that by looking at a problem up front in great detail, an architecture can be built that will enable quick, efficient, high-quality implementation. Lean-Agile methods assume that we want to take advantage of what we learn as we move forward. Lean-Agile developers have come up with a plethora of techniques that support this.

We must build software quickly. This has two interpretations. Done quicker? Or quicker to show something? Both are good. If we don't have something quick to show, however, then we run the risk of building the wrong thing (or at least, the customer will want something else after being shown, just before release, what we built). Quick turnaround is essential.

Be able to adapt to things learned during the development process. Standard techniques are more about being able to prevent the need for adaptation. Lean-Agile techniques accept the fact that we live in a world where our understanding continues to evolve and improve.

Give management exposure to what is going on through the development process. Management cannot function well without steady, consistent, accurate information from the development team. Standard techniques rely on artifacts and milestones that tell us that:

Technical Book Recommendation:

Fit For Developing Software: Framework for Integrated Tests* by Rick Mugridge, Ward Cunningham. (ISBN: 0321269349) This book is not just about FIT, but why integrating test and development together is a necessary part of the analysis process. Explains very nicely how tests are really executable requirements. Get it, read it, use it.

- Requirements are done
- Analysis is completed
- Architecture is completed
- Design is done
- Coding is done
- Testing is done

The problem with basing progress on artifacts and milestones such as these is that they don't necessarily accurately reflect where the team is. Completing code does not mean you are completed by a known amount. How well is the code implemented? Does it do what the customer wants? How long will it take to get the system deployable? These are questions that must be answered in order to see where you truly are. Lean-Agile methods measure where you are by how much value you have actually implemented – implemented in a way that it can be deployed. This is done code and this is an accurate measure of progress.

Lean-Agile projects have many methods of disseminating information to management as well as other teams. These information radiators, as they are called, show how much has been done, how much remains to be done and what rate things are progressing. This is done in a manner that is reality based, not artifact based.

Create teams where people want to work and where the workload is steady. Standard techniques basically say – we don't trust the team to do what's best – we'll have a process they must rigorously follow. Lean-Agile methods assume teams are motivated and well-qualified. A team that can make its own decisions should produce predictable, repeatable results. They will decide the best way to do it, based on best-practices. Trusting on rote methods is unlikely to work as each project is different.

We need a cost-effective development process. Cost effective means building what returns the best ROI. Standard techniques require us to build features we anticipated we would want to build at the beginning of the project. Lean-Agile techniques give us the opportunity to build newer, better ideas if the customer wants us to.

Summary

In summary, we believe Lean-Agile methods take advantage (or rather, address) the reality of how customers and developers interact in the real world and how humans operate. Although it'd be great to be able to decide everything up front in an efficient, effective manner, past experience tells us that this hasn't worked very well.

Focusing on eliminating waste is where to start. Getting feedback from customers as often as possible goes a long way toward facilitating this. This typically requires some sort of iterative process. Being risk aware is also essential. This allows you to pay attention to aspects of the development effort that might hurt you later. Maintaining high quality code is essential – not only for this release, but for subsequent ones. Finally, letting management see where you are, where you are going and how you are planning to get there is essential.

Volume 2, Issue 6
October 2005

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com

To join a discussion about this article, please go to
http://www.netobjectivesgroups.com/6/ubb.x
and look under the E-zines category.

Volume 2, Issue 6
October 2005

NET OBJECTIVES

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email: info@netobjectives.com

Go to http://www.netobjectives.com/subscribe.htm to subscribe to our mailing list and receive future ezines and seminar announcements

Net Objectives – Courses

New Course: Implementing Lean Software Development: Practitioners Course

2-day Workshop. Of the many methods that have arisen to improve software development, Lean is emerging as one that is grounded in decades of work understanding how to make processes better. Lean thinking focuses on giving customers what they want, when and where the want it, without a wasted motion or wasted minute. In this workshop you will learn how to apply Lean principles to software development.

New Course: Lean Software For Managers

1-day Workshop. Lean Thinking is an approach that can help today's Software Technology leaders succeed in making difficult strategic decisions. It has a long history of generating dramatic improvements in fields as diverse as health care, manufacturing, retailing, and now information technology. Learn first-hand from Lean thought-leader Mary Poppendieck how to apply Lean Principles to your organization.

New Course: ScrumMaster Certification Training

Agile project management is as radically different from traditional project management as agile processes are different from traditional methodologies. One of the most popular agile methods is called **Scrum**, and this course is about managing Scrum projects. In this course you will be certified as a ScrumMaster and learn how to make a development team, a project, or an organization agile.

Agile Software Development Simulation - This 1-day course is a simulation of our agile software development process. Its purpose is to show anyone who is associated with software development why agility works by simulating a software project. No coding experience (or even direct experience of development) is needed.

New Course: The Product Owner / Customer Role in Agility

Agile development has proven itself to be successful, especially with small teams of 6-10 developers. The two most popular small-team agile methods are eXtreme Programming (XP) and Scrum, and they have much in common. One of the major things they have in common is a close working relationship with their Customer, or Product Owner. In fact, the existence of a qualified, empowered, Product Owner / Customer is one of the major success factors in these processes. In this course we explore the basic responsibilities of the Product Owner / Customer, arguably the hardest job in software development.

New Course: Test-Driven ASP.NET

Test-Driven Development (TDD) is a powerful tool for combining software design, testing, and coding to increase reliability and productivity. Co-developed by James Shore, the primary developer of NUnitAsp and winner of the 2005 Gordon Pask award for contributions to agile practice, and Net Objectives, this course is the most authoritative test-driven ASP.NET course available.

Go to the URL below for information on how to register
www.netobjectives.com/events/pr_main.htm#UpcomingPublicCourses

Volume 2, Issue
October 2005

NET OBJECTIVES

Address:
Belle-View Office Park
275 118th Avenue SE, Suite 115
Bellevue, WA 98005
Telephone: (425) 688-1011
Email :info@netobjectives.com