Surfing the Net for Software Engineering Notes

Mark Doernhoefer

The MITRE Corporation
7515 Colshire Dr.
McLean, VA 22102
mdoernho@acm.org

Agile Surfing

Agile methods are one of the most controversial topics in the software development community today. The homepage for this year's Agile Development Conference remarked that one of the purposes of this year's conference was to dispel the notion that agile development methods are "a bunch of hot air". And indeed, the amount of hype surrounding this new approach to writing software would lead one to believe that agile methods are long on promises and short on rigor. Even some of the terms surrounding agile methods; Extreme Programming (XP), Crystal, Scrum, etc., seem like they were developed by an advertising agency instead of a software process improvement group.

As I recall there was a similar amount of hype surrounding the introduction of object oriented software development. For those of us dutifully decomposing functions into data flow diagrams using structured analysis/structured design under the waterfall development cycle, the thought of a bunch of objects running around our systems with attributes and methods seemed wacky. Early OO adopters were advised that they could easily locate the objects in their systems by highlighting the nouns in the requirements specifications. Over the years, the object-oriented approach to development was refined, UML modeling matured, OO CASE tools became available and today object-oriented software development has largely replaced the old SASD methodology. Could agile methods be the replacement for object-oriented methods?

Kent Beck's book, *Extreme Programming Explained: Embrace Change*, published in 1999, launched the agile revolution. Since that time, Extreme Programming (XP) has gained a huge following in the web development community. Programmers found they were free to design on the fly and the XP philosophy of continuous testing was rigorous enough to produce good quality results for web apps. There was little need for program documentation because in the dynamic web environment, applications were changed frequently to keep fresh content on the web site. The programmers didn't need to document code or design since the team was immersed in the code and knew it by heart. The Extreme Programming approach fit the bill in this dynamic environment.

As more development teams experimented with XP, they added additional techniques and the family of agile methods started to grow. Unlike the other methods for software development, agile methods include a fair degree of sociology in their approach to development. In fact, the first principle of the agile manifesto states the goal of emphasizing individuals and interactions over processes and tools. Other agile principles require projects with no overtime so as to not burn out the development team. Authors such as Tom DeMarco have previously recognized the role of people in the development process, but the agile approach integrates the people processes into the agile processes to a much greater degree than any previous methodology.

The agile developers realized that the agile approach might not be suitable for all types of development. Early adopters are contributing experience reports and fine tuning some of the agile methods. Unlike the other methodologies that claim to work for all types of application development, agile methods advocates are finding that the agile approach works better for certain types of projects. There has been a lot of effort directed to identifying the characteristics of applications development that lend themselves to the agile approach and even more effort to identify those situations where agile methods may not be the best choice.

So all you old process practitioners keep an open mind, you're about to learn something. XP and agile methods have been around long enough to garner some real successes. You'll be interested in the case study links on the web sites listed below.

You agile "old hands" will find a lot of new content on the pages below. Since the agile approach is constantly being refined, chances are you'll find some new articles of interest.

Before we start our surfing safari, let me apologize in advance for mixing the terms process, practice, methodology, technique and approach. I know that each term is different and that each has a precise definition in the engineering community. I will be using all of these words interchangeably without respect to their exact meanings. It's called poetic license and I do this to keep the text from becoming repetitious. And, since this is not a refereed journal, I can get away with it.

The New Methodology

http://www.martinfowler.com

Martin Fowler's web site is a good place to start talking about agile methods. His home page at the URL listed above has a collection of updates on various topics in software development with an emphasis on agile development methods. Each item is an in depth discussion of the topic, either as a magazine article, or a set of links to other pieces.

I've placed this site first due an excellent article entitled "The New Methodology". This older article was not listed on the home page, but can be located through the articles page or directly at:
http://www.martinfowler.com/articles/newMethodology.html
"The New Methodology" offers a comprehensive look at many of the tools and techniques used in the agile approach. It's a great place for a beginner to start learning about agile methods. An international resource, the article and the underlying pages have been translated into French, Russian, Japanese, and several other languages.

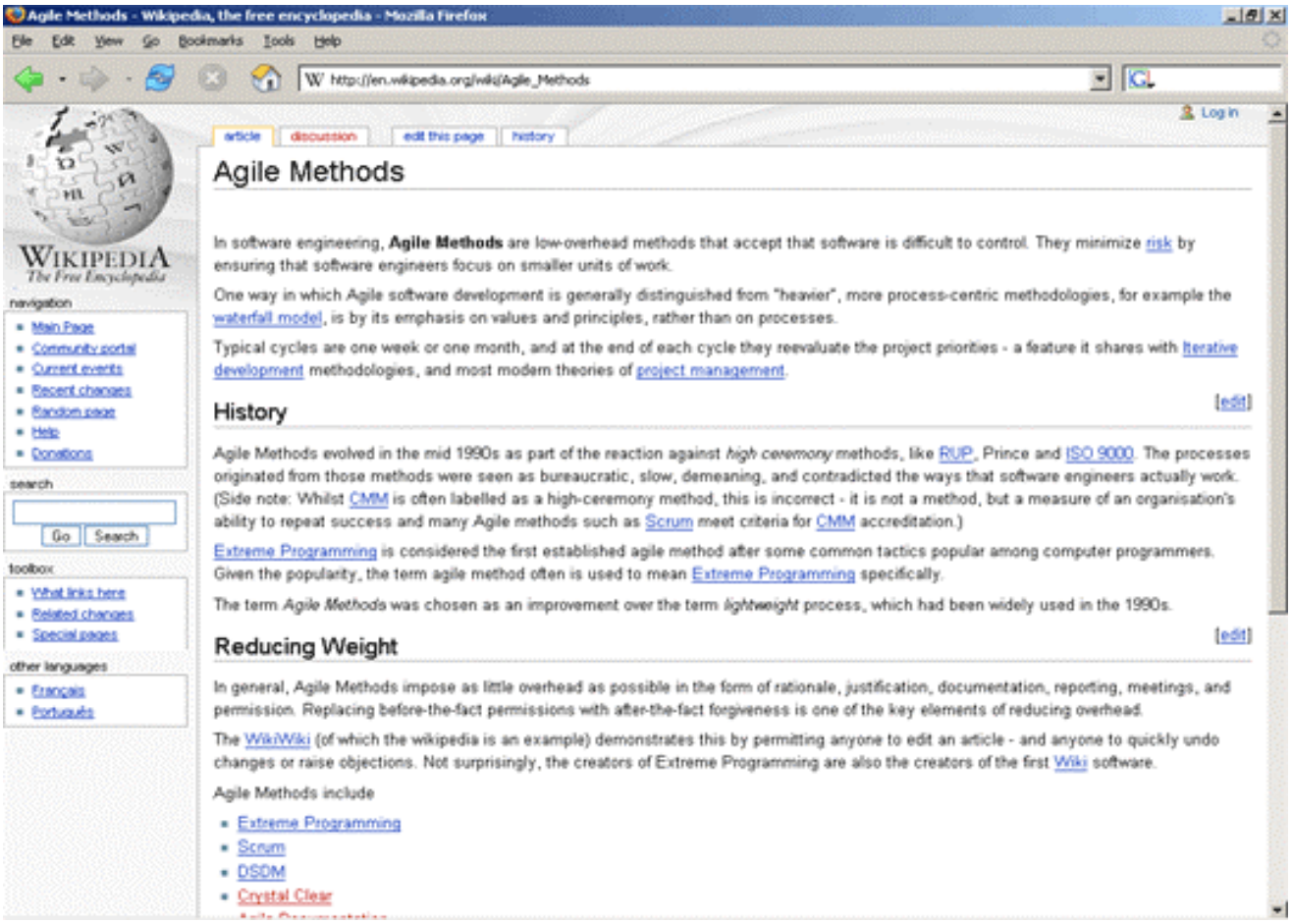
Martin Fowler is a prolific author who offers reasoned, practical advice to those actually writing applications. Although he uses a lot of the XP and agile method buzzwords, you won't find any academic discussions of the science behind agile methods. This is straightforward, get-it-done advice. This practical approach to programming is characteristic of the agile movement.

Agile Methods Wikipedia

http://en.wikipedia.org/wiki/Agile_Methods

The Wikipedia is quickly becoming a standard reference site for just about any topic imaginable. For those not familiar with the site, Wikipedia, is a free online encyclopedia where the encyclopedia citations are open source. Wikipedia currently holds over 300,000 encyclopedia articles.

The agile methods Wikipedia article does not go into great depth on the agile practices, but has some nice, concise definitions of the agile practices. If you are looking for a quick answer to an agile methods question, check it out. And, if you are an agile methods practitioner who wishes to contribute an item to Wikipedia, stop by. At this writing there were a number of empty placeholder links awaiting contributions by subject matter experts.



The Agile Manifesto

http://agilemanifesto.org

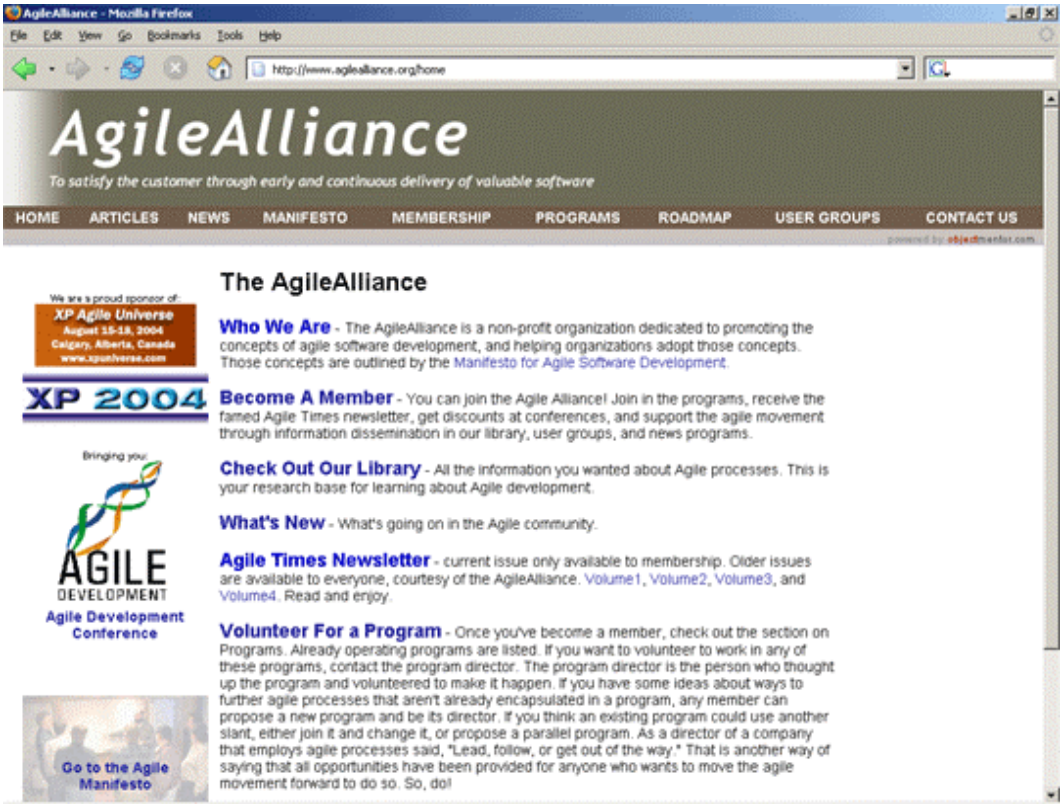
Now that you have some idea of what agile methods are all about, surf by the Agile Manifesto page to see how and why this new approach to software was developed. The brief manifesto sums up the rationale behind agile software development:

> We are uncovering better ways of developing
> software by doing it and helping others do it.
> Through this work we have come to value:
>
> **Individuals and interactions** over processes and tools
> **Working software** over comprehensive documentation
> **Customer collaboration** over contract negotiation
> **Responding to change** over following a plan
>
> That is, while there is value in the items on
> the right, we value the items on the left more.

The term "manifesto" implies a radical statement and, as you can see from the screen shot, the manifesto was created by a radical group of programmers standing around a whiteboard. For a traditional, process-based developer like myself, this is dangerous stuff. How can you create quality software when you're writing the tests at the same time you're writing the code? Whoever heard of trusting a software development to interactions between individual developers instead of documented processes written in a software development plan? If we spend all our time responding to change, how can we ever complete that plan? And how can you possibly maintain software without the accompanying comprehensive documentation? Read on.

The manifesto site also has pages that describe how the manifesto came to be and mini-biographies of the folks involved. The agilites, as they are sometimes called, come from a wide variety of backgrounds, but are now consultants, educators, or mentors, all serious about building good software quickly. The site has a feature to allow you to sign on to the manifesto. I don't exactly know what that means other than your name will be posted to the world as a signatory to the manifesto. There's no count of names, but well over a thousand people have signed on to the principles of agile software development.

If you visit the manifesto page, make sure you check out the link describing the genesis of the manifesto, "About the Manifesto" by Jim Highsmith. The description of the meeting at a ski lodge in Utah provides great background to the agile movement.

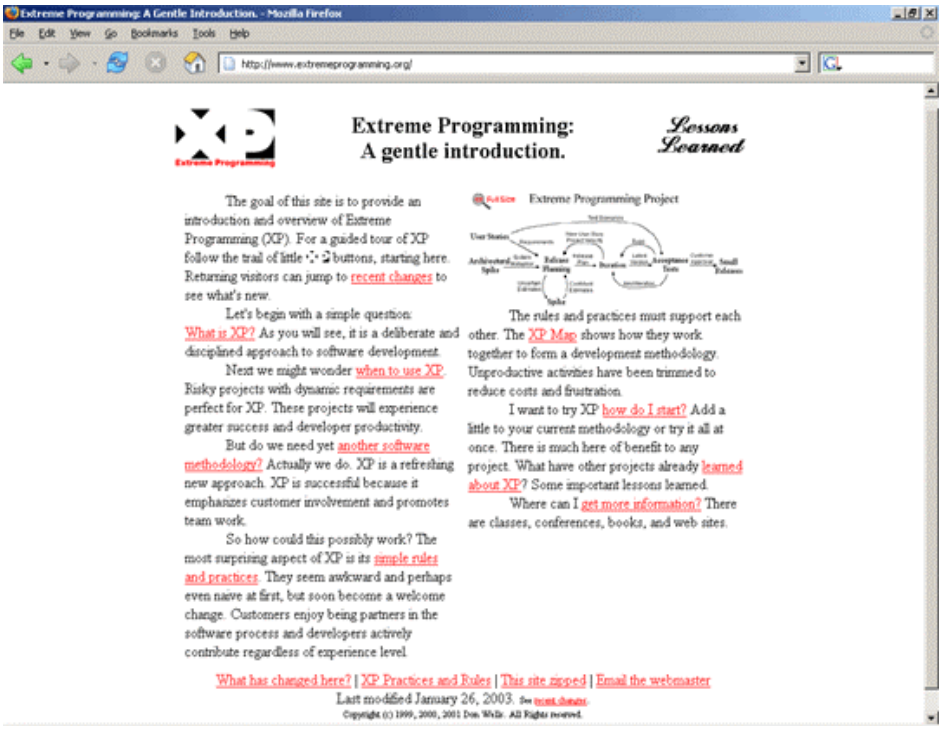


Extreme Programming

http://www.extremeprogramming.org

One of the most important core processes (and perhaps the most controversial) in agile development is Extreme Programming. This approach to software development uses pairs of programmers who typically write unit tests at the same time they write the application code. I am greatly oversimplifying how XP works, so I would encourage you to visit the XP site to get the full story. The subtitle for the XP site is "A Gentle Introduction", and that's exactly what it is. All of the core XP practices are laid out in simple diagrams to help you navigate the overall structure of the methodology.

A successful XP project requires a shift in programmer attitude. As you navigate through the XP site, this gentle introduction will also talk about the rationale and motivation behind the various XP techniques. Most older methodologies focus on the mechanics: flowcharting, modeling, or code constructs. Because the agile focus is on "writing good code" XP places an emphasis on the social aspects of the software both from the user and developer perspective. It all sounds a bit "touchy-feely", but another goal of agile methods is to produce software without



The Agile Alliance

http://www.agilealliance.org/home

Similar to the manifesto site, the alliance is where the manifesto signatories hang out, trading ideas and best practices with each other. Inasmuch as agile methods are still being developed and tweaked, this forum for practices and approaches is where the action is. For those just stating out, there is a "Roadmap" section. Annual membership to the alliance is $100, $40 for students. Membership entitles you to receive the Alliance newsletter and participate in the various Alliance programs.

The Agile Alliance site features a great collection of articles, grouped by topic area, on various aspects of agile development. The articles vary from case studies to very technical discussions of specific implementations of the agile core practices. The collection is kept current with the frequent addition of new articles. Overall, it's a great resource for anyone involved with agile development.

In addition to the collection of articles, the Agile Alliance site also offers current news items (the news is also available as an RSS feed), user group information, and information on agile development conferences.

killing the programmers. (See Ed Yourdon's *Death March* for more on killing programmers.) Your XP experience should be pleasant as well as effective so the XP site talks a lot about attitude and team building in addition to introducing you to the overall XP approach.

Scrum

http://www.mountaingoatsoftware.com/scrum

Scrum is a process that provides the day-to-day ground rules for the inner workings of the development team. Scrum formalizes programming team dynamics using funny names for the process procedures. For example, the month long development iterations in Scrum are called "sprints". So every month, the development team sprints to a new software release.

During the sprint, the development team holds a morning meeting, the daily Scrum, to set the work program for the day. At the Scrum, each participant is known as a chicken or a pig depending upon his or her level of involvement in the program. These roles are derived from the old joke that when preparing a bacon and egg breakfast, the chicken is involved, but the pig is totally committed. The pigs are required to attend the daily Scrum and are the only ones permitted to speak at the Scrum.

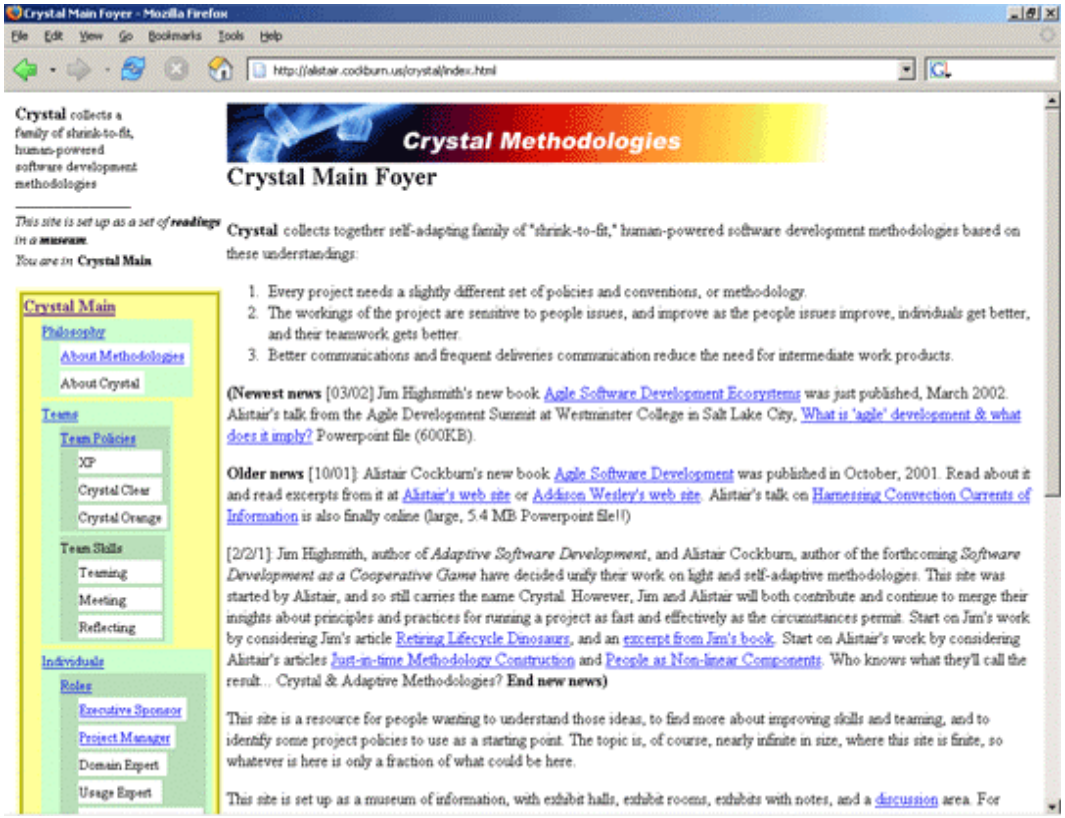
Had enough yet? No, this isn't a joke; this is an actual agile methodology that is gaining popularity for its success at turning out quality software in a short period of time. As you can see from the screen capture, there's a lot more to the Scrum and the supporting processes. Check out the Mountain Goat site for the full story. You'll find all you need to know to get started with your own Scrum.

Crystal

http://alistair.cockburn.us/crystal/index.html

Crystal introduces elements of sociology into the agile methods. As I mentioned above, the agile approach is remarkable in that the human interaction of programming and development teams play an important role in the agility. This collection of human factors in the development process has been collected in Crystal. The primary Crystal proponent is Alistair Cockburn. Cockburn calls Crystal "a family of shrink-to-fit, human-powered software development methodologies".

The Crystal family of methodologies focuses on the people participating in the development and the interactions between them. Over years of observing and participating in developments, Cockburn noticed certain behaviors in the programming team that are characteristic of good development teams and other behaviors that detracted from the success of the project. Cockburn has cataloged these behaviors and describes



Crystal collects a family of shrink-to-fit, human-powered software development methodologies

This site is set up as a set of readings in a museum.

You are in Crystal Main

(Newest news [03/02] Jim Highsmith's new book Agile Software Development Ecosystems was just published, March 2002. Alistair's talk from the Agile Development Summit at Westminster College in Salt Lake City, What is 'agile' development & what does it imply? Powerpoint file (600KB).

Older news [10/01] Alistair Cockburn's new book Agile Software Development was published in October, 2001. Read about it and read excerpts from it at Alistair's web site or Addison Wesley's web site. Alistair's talk on Harnessing Convection Currents of Information is also finally online (large, 5.4 MB Powerpoint file!!)

[2/2/1] Jim Highsmith, author of *Adaptive Software Development*, and Alistair Cockburn, author of the forthcoming *Software Development as a Cooperative Game* have decided unify their work on light and self-adaptive methodologies. This site was started by Alistair, and so still carries the name Crystal. However, Jim and Alistair will both contribute and continue to merge their insights about principles and practices for running a project as fast and effectively as the circumstances permit. Start on Jim's work by considering Jim's article Retiring Lifecycle Dinosaurs, and an excerpt from Jim's book. Start on Alistair's work by considering Alistair's articles Just-in-time Methodology Construction and People as Non-linear Components. Who knows what they'll call the result... Crystal & Adaptive Methodologies? **End new news)**

This site is a resource for people wanting to understand those ideas, to find more about improving skills and teaming, and to identify some project policies to use as a starting point. The topic is, of course, nearly infinite in size, where this site is finite, so whatever is here is only a fraction of what could be here.

This site is set up as a museum of information, with exhibit halls, exhibit rooms, exhibits with notes, and a discussion area. For

how to select the appropriate practices from the collection to provide the best possible set of methods to leverage the abilities of the team.

If you happen to stop by this site, don't miss the articles on "Software Development as a Cooperative Game" and "Software Development as Community Poetry Writing". These are two interesting articles that every developer who has been involved in a programming team will identify with.

Dynamic Systems Development Method (DSDM)

http://www.dsdm.org

DSDM is a flexible methodology that was derived from the work done with Rapid Application Development (RAD) methods. A single RAD approach was never codified and RAD grew to mean different things to different people. DSDM took those independent RAD techniques and placed them in an industry standard framework to yield a single consistent application development approach.

The resulting DSDM framework has been in use and refined for over 10 years. Now in Version 4.2 the DSDM framework is licensed for use on a personal, developer, or project basis. Special licenses are available at reduced cost for students and charities. This is probably why DSDM hasn't gained a great deal of popularity. The licenses and annual renewal fees can be quite costly.

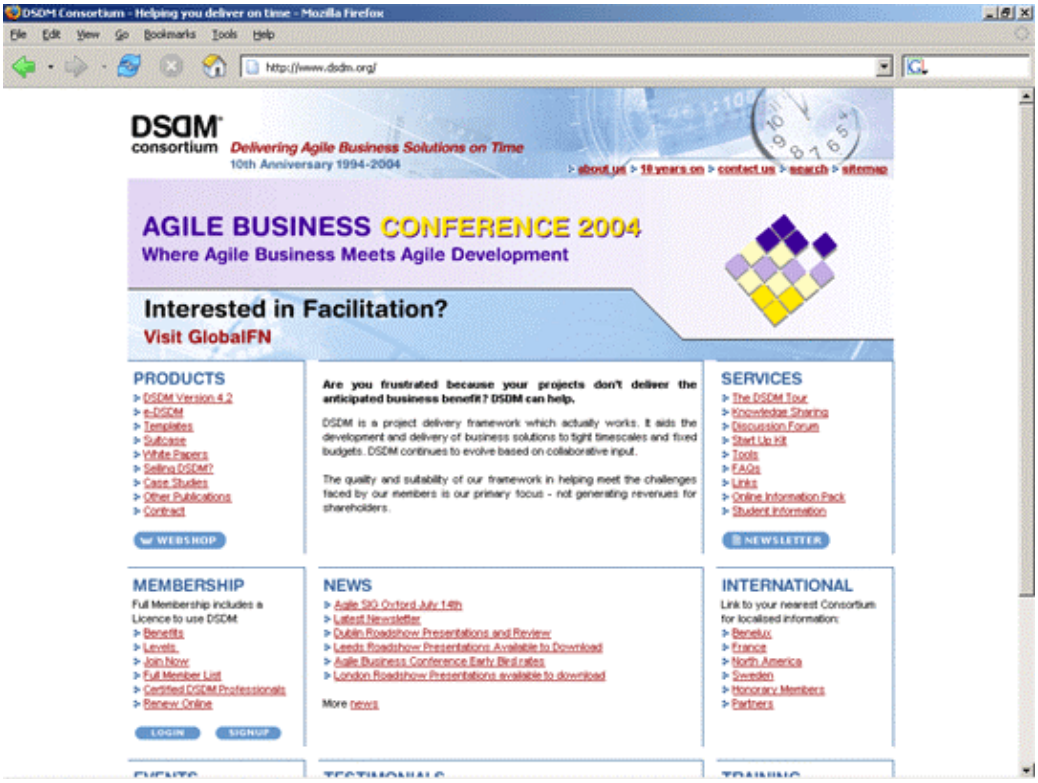
Refactoring

http://www.refactoring.com

In addition to his own personal site, Martin Fowler also maintains the refactoring.com site. He defines refactoring as follows:

"Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior."

Refactoring is a very important procedure for an agile development. Since there is no initial design or planned architecture, a continuous design process is used to develop the application. Under this approach, the application architecture evolves in response to the evolving requirements. Complex requirements identified late in the coding cycle may dictate the



web site you can find lots of great sources on refactoring, books, links, white papers, doctoral dissertations, even people who can help out with refactoring. The site is a companion to Fowler's book, *Refactoring: Improving the Design of Existing Code*, so in addition to news and tools, you can also find the graphics and code illustrations amplifying the book.

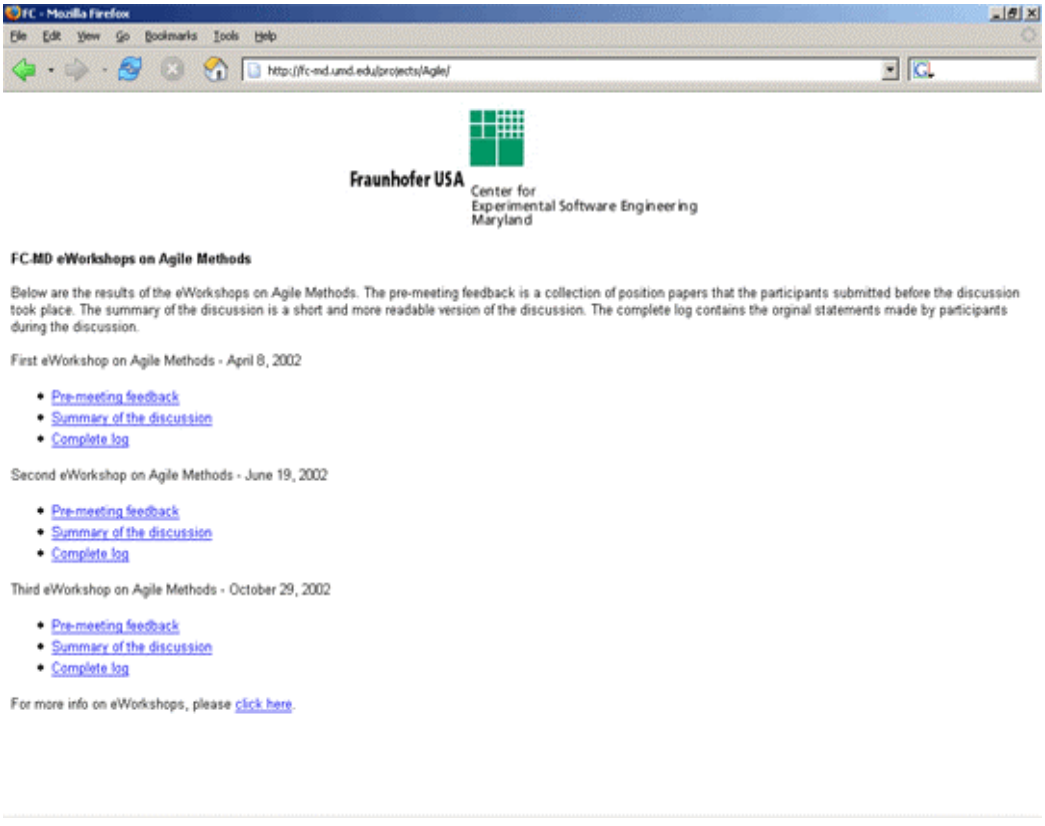
The Fraunhofer Center eWorkshops

http://fc-md.umd.edu/projects/Agile

The Fraunhofer Center for Experimental Software Engineering at the University of Maryland hosts current research programs in a number of new and innovative software engineering processes and techniques. Their project on agile methods has sponsored three eWorkshops where leading proponents of agile development discussed various topics surrounding the new methodology. The Center has published the transcripts of those workshops along with summaries of the issues discussed during each session. I recommend reading at least the summaries if not the entire text of each workshop. Although there is disagreement among the experts, many of the criticisms of agile methods are discussed and answered with

need for massive architectural changes to the software. Here's where the agility in the development process comes in.

Those critical of agile development point to refactoring as one of the weaknesses of the agile approach. Since you don't alter the external behavior when you refactor, the only benefit of refactoring is to improve the design. It's time wasted because you didn't take the time to design the system before you started coding. This is analogous to the exercise we when through at the Y2K rollover. Lots of people spent lots of money just to make sure their applications worked the same on Jan. 1, 2000 as they did on Dec. 31, 1999. The agilites respond to this complaint by pointing out that many traditional developments are required to refactor (or redesign) due to changing requirements or an improper initial design. So the agile refactoring isn't that big a deal. In fact, since refactoring is an established agile process, agile developers are much better at effecting design changes than engineers using a traditional approach.



So that brings us back to refactoring.com. On Fowler's

practical advice on how to implement this new approach to development.

Topics included in the eWorkshop discussions are:

- Project size
- Refactoring
- Agile methods for safety-critical software
- Levels of testing
- Documentation

and several other controversial topics. The discussions provided insight on both sides of the issue; how agile development supports or fails to support each topic area. I felt many of the concerns of "traditional" methodologists (myself included) were answered in these workshops. However, you should read the monologues and decide for yourself.

The Official Agile Modeling Site

http://www.agilemodeling.com

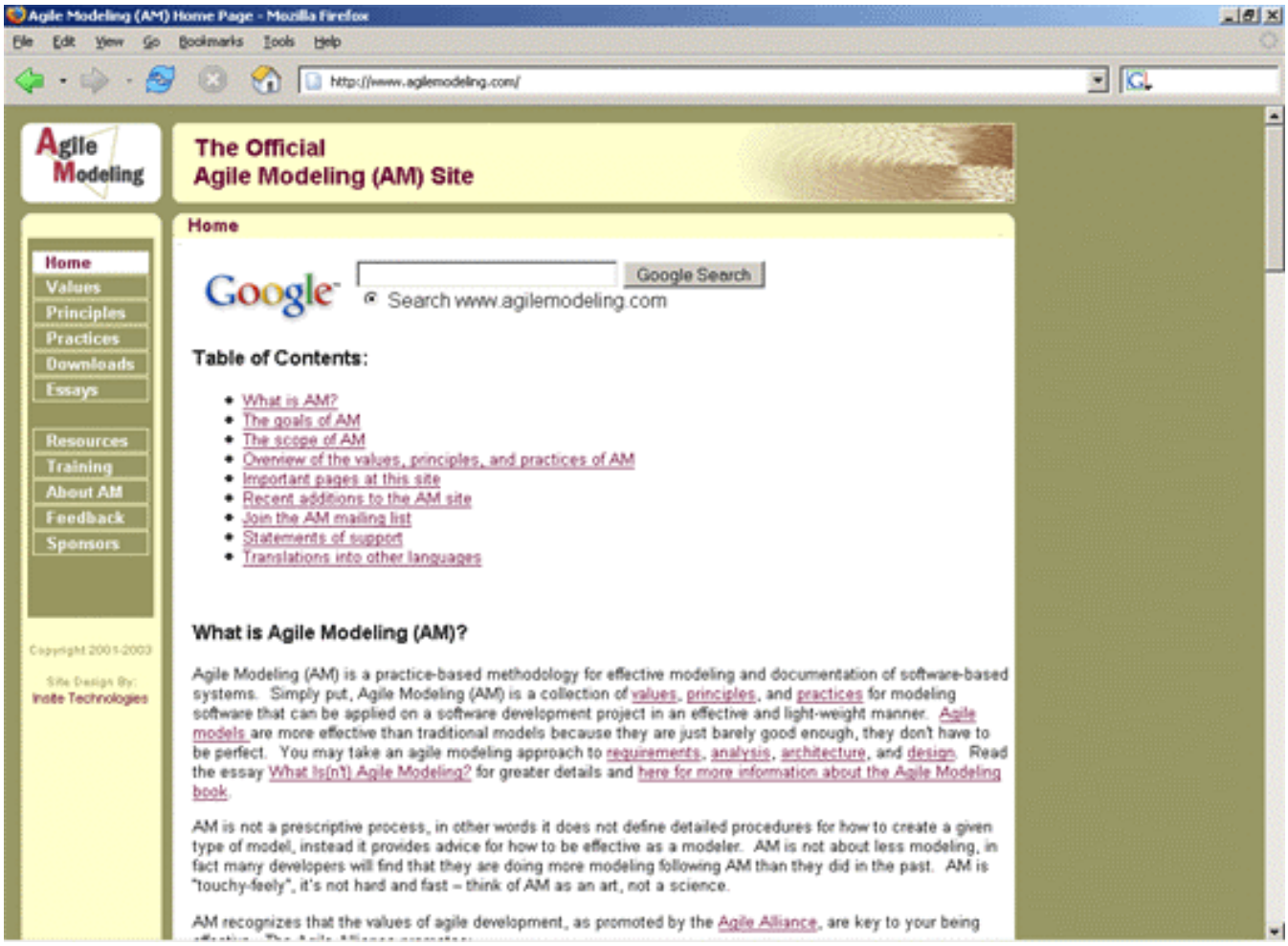
Authored by Scott Ambler, one of the original signers of the Agile Manifesto, the Agile Modeling (AM) site provides an introduction to some lightweight modeling techniques that support agile methods. As the site says, "Agile Modeling (AM) is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner." In addition, the site discusses how an agile approach can be combined with other techniques, such as Model Driven Development, and also provides guidance for the use of CASE tools in an agile environment.

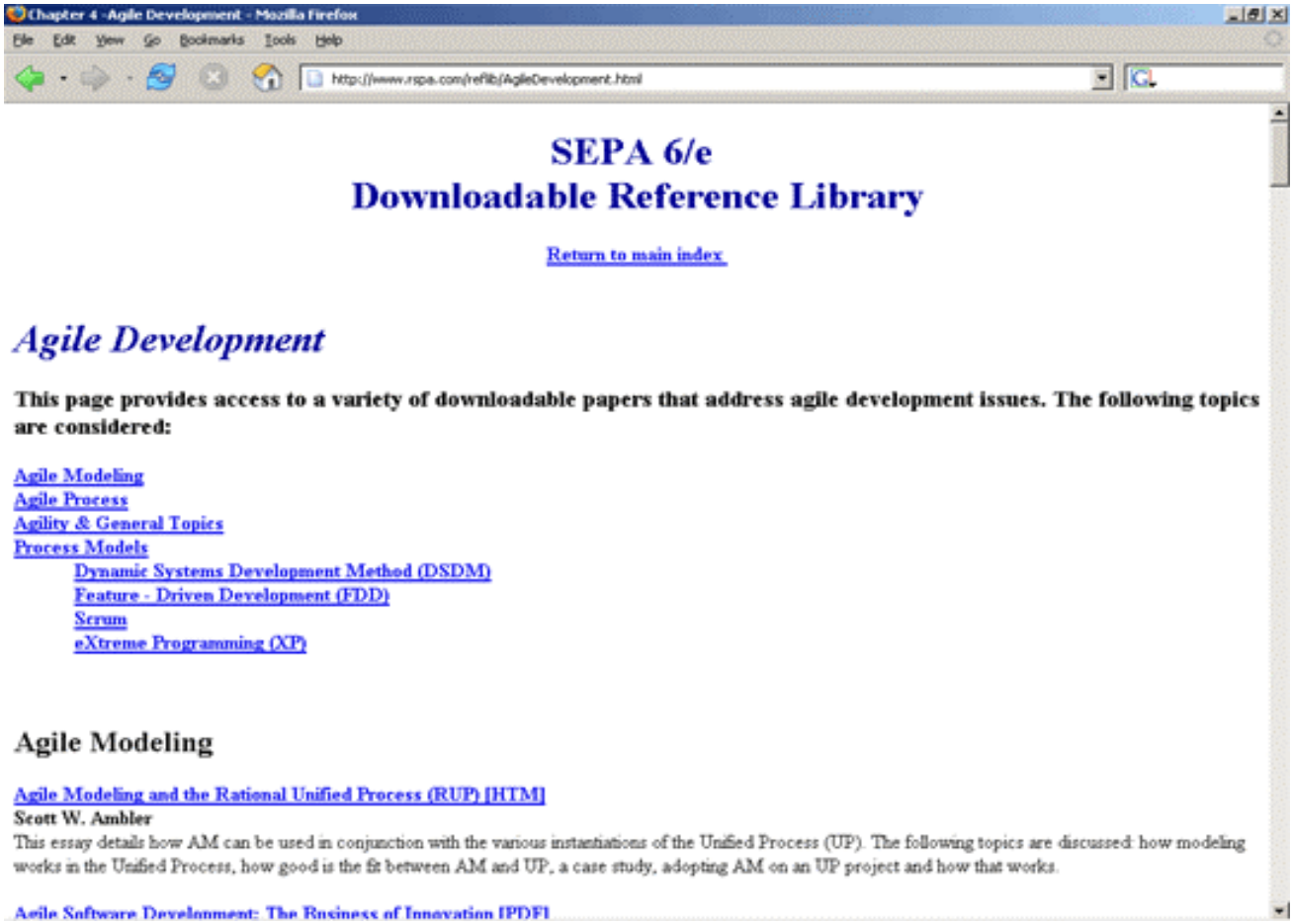
Ambler's essay on agile documentation at:
http://www.agilemodeling.com/essays/agileDocumentation.htm
answers the "you don't document" criticism. Don't expect to find a description of state-charting, data flow diagramming or UML notation. Instead, you'll find practical advice on what needs to be documented and how to create useful documentation as opposed to useless piles of paper. For example, Ambler suggests focusing on content rather than format and writing documentation for a specific audience. A few years back I was a programmer in a large multi-year development program. We were sure that the documentation we wrote to accompany the code would never be read by anyone. The standard joke in the project team was that we captured our documentation in write-only memory. The AM site will show you how to avoid this pitfall.

The Pressman Downloadable Reference Library

http://www.rspa.com/reflib/AgileDevelopment.html

The website accompanying the new edition of Pressman's excellent book, *Software Engineering: A Practitioner's*





Approach, provides a collection of agile development articles beginning with XP and moving into agile processes and agile modeling. Many of the links you'll find here are also linked from the other sites mentioned above, but Pressman brings them all together in one spot organized by topic. Furthermore, if you back up to the index at:
http://www.rspa.com/reflib/Index.html
you'll find the full reference library providing links supporting the chapters of *Software Engineering*.

I wanted to mention this site because Pressman is not one of the signatories of the Agile Manifesto nor is he a member of the Agile Alliance. In fact, the Pressman book, now in its sixth edition, is a classic reference for traditional structured software methods. Inclusion of agile methods in Pressman's book is acknowledgement that agile methods have matured to the point where they form viable, valuable alternatives to the other methods discussed in the book. You gotta love it, Chapter 2 discusses the SEI CMM and Chapter 4 covers Agile Development. Talk about oil and water…

In Conclusion

My simple conclusion is that despite the glitz, these agile folks are onto something. Like any process or methodology, an agile approach can be compromised with bad decisions, but given the nature of software development and the wide range of projects, it seems clear that no one approach to building software fits all situations. Several of the web sites mentioned above describe the types of projects that work well under agile methods. For those projects, the agile approach provides a lightweight, systematic approach to building applications quickly. Sure beats sitting in front of a keyboard and flailing away.

I do have a few concerns about agile methods. For a larger development, there does not seem to be a good way to coordinate application-wide strategies. For example, how do you communicate the standard approach to functions such as security, performance monitoring, and fault localization, to all developers across a large team? Developers must have some discipline (and smarts) before they begin an agile development otherwise each refactoring could become a nightmare. There are techniques at the agile methods sites that address these concerns, but after seeing a number of traditional, highly process driven approaches yield poorly designed and badly documented results, it would be nice if the agile methodology provided additional guidance to influence developers to do the right thing.

There are examples of successful projects using any of the past or present methodologies. I believe that any approach can work if you plan your development in advance and adjust your methods in response to specific issues raised during the course of development. Agile methods seem to address the major complaint against the "traditional" methodologies, the lack of responsiveness to requirements changes. With rapidly changing web-based applications, requirements change has become a way of life. Given this reality, the agile approach just might be the right thing at the right time.