XP Universe 2002

Mark Windholtz, ObjectWind Software Cincinnati, Ohio; September 2002

In this report …

Conference Quotes:

- It's called "extreme programming" not "stupid programming".
- When was the last time you were out of your comfort zone?
- Steering the Big Sheep
- So is Watts Agile ?!?! crowd: Yeaahh!
- CMM Certification depends on the person doing the evaluation.
- There is no reason you couldn't use XP in PS/TSP
- Space Shuttle Software cost $400/Line of Code with up front design, yet second launch was scrubbed because of software error.
- We don't want an Agile religion we want a craft!
- Un-deployed detail is inventory. At [one project] we went to production every night. – Kent Beck.
- It's the ineffable stupid! – David West

This report will step through my daily sessions and then provide some overall impressions at the end of the report. Agile and XP Universe 2002 had 330 attendees - a 40% increase from last year. Attendees were 80% from industry, 20% from academia; 60% were programmers, 35% management, Testers, Customers and Analysts made up the rest. 25% rated themselves as competent Agile practitioners, 25% were learning and 50% were still evaluating the approach.

Saturday 3-August-2002

The Conference Started with a lively reception given by ObjectMentor at their Chicago offices. The tour included a look at the numerous, humorous framed posters from www.despair.com. They also showed off their large training area and the specially constructed Pair Programming tables.

Sunday 4-August-2002

Sunday I attended two Tutorials:

Change Wizardry: Tools For Geeks by Joshua Kerievsky and Diana Larsen

Using a combination of simulations and discussions this tutorial illustrated and explained the obstacles and opportunities when guiding an organization through change. It addressed many of the particular challenges in preparing and leading a software development team in a change to Extreme Programming practices. The simulations in the tutorial provided participants an experience of what change feels like. To convince a group to change one must get their confidence. To get their confidence it is best to sympathize with their fears of change. To do that one must experience a similar fear and become conscious of its impact on ones own behavior. These steps occur in a good simulation, even if you already know they are about to happen to you. This tutorial provided good simulations that walked you though the phases and reflection in a very valuable way.

Next we discussed change models from Virginia Satir and William Bridges. The models outline the stages moving from Status Quo, to Loss & Letting Go, to Exploring the Chaos, and finally to Starting Anew with the new Status Quo.

Joshua & Diana suggested that potential XP projects spend 2-4 days to conduct **Change Readiness Assessment**. Some environments are not conducive to a successful XP project and XP should not be forced to fit. The tutorial provided a template for the questions that need to be asked. Next the project should begin by developing a **Project Charter.** A Project Charter defines the vision & purpose, mission, objectives, committed resources, players, team members, and preliminary schedule of iterations and milestones. All metrics must be strictly quantifiable so that success or failure can be easily distinguished.

A software project is an expensive undertaking. At the outcome of a project the team has experienced many events. An organization stands to benefit greatly by pausing at the end of the project to collect the experiences. A **Project Retrospective** gives an organization the opportunity to reflect and learn. A retrospective can also provide a connection back to the Project Charter to evaluate to what degree the original goals attained. This also provides a strong jumping off point for the next project.

Steering the Big Ship: Succeeding in Changing an Organization's Practices by Lowell Lindstrom

This tutorial took a broader approach to organizational change, one not merely focused on XP. Lowell presented the Satir Change Model and an extended case study of an organization undergoing various changes. There were multiple group discussions about how the model helped inform us of the stages of change the organization experienced. There was also a chance to get a perspective from the top, by interviewing the CEO of Mock Corporation to find his view of the state of the company. Lastly, Lowell provided an model that showed how to design an organizational Infrastructure for Change that included Goals, Training, Promotion, Practices, Best Practices, Incentives, and Monitoring to get Results.

At 5:00pm there was a reception in a large tent set up on the side. It was great to catch-up with lots of good acquaintances from around the world and meet new people so that next conference there will be more people to catch-up with.

Monday 5-August-2002

Conference Chairs: Laurie Williams and Don Wells opened the conference.
Laurie and Don opened the conference and provided some of the statistics that appear in the first paragraph of this report. They introduced the conference sponsors: RAD Soft, ThoughtWorks, ObjectMentor, and TogetherSoft.

Welcome Address: Martin Fowler
Martin began with a poll of the audience, which produced the some of the statistics in the opening paragraph. Martin then stressed the importance of the XP practices. The practices allow us to respond to change. They are targeted at being Agile. But we need to practice them. He expressed concern that there are growing descriptions of Agile and XP by people who are not actually doing them. He drew attention to one published account of a supposed Agile team that was doing *nightly builds* when for any real XP team nightly builds are glacial timeframes.
The practices are not new. But most projects have not done them. The Agile and XP movement are a large shift in how people think about software development. (Thankfully he did not use the word: *Paradigm*). The change in Attitude is twofold: (1) a change in attitude toward Change (2) and a change in Attitude toward People.

On the first change he said that most processes think change is bad. XP requires a Brain Flip: we welcome valid changes in Business needs. Requirements changes happen for good business reasons. We provide the ability to respond to it. And the kicker is: a late change in requirements is a competitive advantageto an XP team!
On the second change he said that Software development is a human activity. Every problem in programming is a people problem. All of the original framers of the Agile manifesto agreed on what should be the first item: *People over process and tools*. Other processes want any people. XP focuses on a particular team to give them the tools and process to be the best they can be.

So what is the next big step of XP? People ask, how prevalent will it be? Will Agile be dominant? Where on the growth curve is Agile? These questions arise because lots of companies want to be doing what other companies are doing. We are at the stage that cricket players call: *Playing yourself in*. Early in a match they get used to the field and light conditions. That's where XP is. Where Objects were in the 1980's. This is a long-term thing. Be patient it may take 10-15 years or more. XP's strongest advantage is all the XP people in the field. But the risk is that lots of so-called XP projects are not doing the practices. We must practice the practices!

Keynote Address: Watts Humphrey
Title: Setting the Agile Context.

Watts lead the IBM 360 project. He started by saying, “I never used XP but I’ve got an opinion about it!” That got the audiences attention. His made his presentation in a bullet item style that I try to echo here.
Then he asked,

1. Why is software so troublesome?
2. Why can’t we agree on how to do it?
3. Why do we argue with each other?

Why is software so troublesome? Because…

- Its complex
- Large range of scale from few hours work to 1000s of people
- Software is everywhere

Why can’t we agree on how to do it? Because …

- There is no single answer
- But a great need for principles and methods.

Why do we argue with each other?

- The industry is in needs new models. We repeatedly invent new and better stuff.
- But attacks are a sign of weakness.
- If facts support the case argue facts. If all else fails attack the opposition.
- In 1972 Winfred Royce was really wise when he proposed the Waterfall process. But people misused the Waterfall.
- We should not waste time attacking it.
- Produce something.

What are the principles we need?

1. Our job is to transform a problem from poorly understood to precise machine instructions.
2. When we know what is needed then build it
3. When we think we know then we are wrong
4. We must experiment
5. We must get expert feedback to experiment
6. We must know we know

Important to Track and Report

- Business needs schedules
- Business must know status
- To report we must measure
- To measure we must Plan

Plan Every Project

- Never commit without a plan

- People who do the work must do the plan
- If you can't plan accurately, plan often
- Plan what you know at the level that fits the job
- If plan does not fit the work revise the plan
- Base plans on historical data
- To have comparable data you must have similar plans
- To make similar plans you need similar processes.
- Can't make a detailed plan for more than 2 months. Sometimes not more than a week.

The best projects provide a balance between Business & Technical Capacity & Customer desires. "CMM does not tell you to do anything". It tells you what to do but not how. Agile is about How. There is "no reason people couldn't use XP in PSP/TSP".
[Unfortunately, my notes are not complete for the last half of his presentation.]
During the Question and Answer session the last questioner asked three good natured questions intended to quiz Watts to see if he was Agile compliant. Although I missed the questions, after the three answers from Watts the questioner asked the audience: ***So, Is Watts Agile?*** And there were approving shouts of: ***Yes!***

Keynote Address: Berry Boehm
Berry started by stating that Agile " is the biggest breath of fresh air software engineering has had in a while". And then went on to compare an Agile approach to a plan driven approach from the perspective of Risk reduction. He compared a case study of a ThoughtWorks project to a Missile detection system. Unfortunately, as he spoke the audience slowly became aware that he was not very familiar with the basic economics of Agile teams. His observations and concerns were not based on a mere caricature of Agile.

OpenSpace
As I was getting ready for my trip to Chicago, I admitted to a colleague that I had not yet decided which tutorials and sessions to attend. He poked fun at XP and asked, *if it's so Agile why do they need to work out a whole week's schedule in advance? Couldn't they just refactor along the way?*
So I wondered to myself: How would you hold a flexible conference? OpenSpace was the answer that the conference organizers provided. A large room was set aside. Anyone could propose a topic and reserve a time slot.
Some of the rules of OpenSpace are:

- The people who show up are the right people since they are the ones who care enough to be present.
- It's not rude to leave. If your mind wonders take your body with it.

This lightweight structure insured that the topics proposed matched perfectly with the interests of the participants. The energy level in the discussions was high. The topics were summarized after the meetings on an ObjectMentor provided Wiki site and can be reviewed at:
http://monet.objectmentor.com/cgi-bin/openspace/wiki.py?OpenSpace

One of the more provocative topics presented at OpenSpace was: *Do you still use a database?*

How fast could an XP team go without using a database? Find out more about this at: www.prevayler.org

Invited After Dinner Talk: Robert Martin

We in the Agile community don't want a religion we want a craft. Two crafts can exist side by side without conflict like the various Martial Arts. Only kids ask: *which is better Kung Fu or Tai chi*? We should not discount other approaches without giving them a try. Has anyone in the audience tried Watt's Humphrey's PSP/TSP techniques?

What is our school? Bob suggested the following scale based on the Martial Arts belt colors.

To earn a belt level the practitioner should demonstrate knowledge in the following areas.

White belt: Knowledge about what a computer program is.

Yellow belt: Knowledge about programming languages and APIs

Green belt: practices of project work: Pair Programming, Refactoring…

Blue belt: project planning: estimating, iterations

Brown belt: design principles.

Black belt: Patterns, and synthesis.

The best part of the talk was the demonstration of techniques where Bob's son Micah placed Bob in various arm holds. It probably wasn't the best part for Bob though.

Tuesday 6-August-2002

OpenSpace: Lean Manufacturing

This topic was proposed and moderated by Kent Beck.

Lean Manufacturing is Toyota's term for how they build cars. It turns all the assumptions backwards. Rather than pushing parts down an assembly line, they pull parts based on need, at the time of need. This gives them the ability to customize orders and to eliminate costly inventories upstream of the assembly point.

Rather than relying on economies of scale, lean manufacturing relies on economies of reducing waste and carrying cost. There is paradox with the economies of scale approach in that while only a manager can stop the line, the line ends up stopping a lot. Toyota encourages any worker to stop the production line when a defect occurs; the line rarely stops, and achieves 99% up time.

The book by Taiichi Ono (see below) has a rant about warehouses and the cost of storing inventory. In programming we can consider un-deployed detail as our inventory. Lifeware is a Swiss insurance company (www.lifeware.ch). They start with zero code for each feature and build it from scratch. They have hold no inventory. They never rely on reuse to develop the initial implementation of a feature. Only after implementation do they explore avenues for reuse. Through Refactoring we find that reuse within a software product that works well, but s often not so good between products. The Lifeware approach indicates how this could work well. Kent advises that the best way to find good reusable code is to re-implement and then refactor. In business terms: Refactoring is a way software projects can reduce inventory and eliminate waste. Since un-deployed detail is the *inventory* of software projects, Lifeware deploys new work into production every evening.

One thing becomes immediately clear; a team must work at very high levels of quality for lean manufacturing to be feasible. In the 1970's Japan's challenge was to get the American buying public to accept the idea that cars should *just work.*That quality should be expected. One participant in the discussion talked about how their company went from having 200 defects in the system to finding a defect every 2 months. To get to this level a TQM approach is to ask the five-whys and develop a fishbone diagram of causes, and causes of causes to five degrees.
Kent suggested that the critical path in an XP project is getting from an idea to it's failing Acceptance test.
This leads to the paradox to go fast you must get do things correctly. To do things correctly you must initially slow down and do things slowly. When Toyota first tried the Ono approach it took them a week to make the first car. After a wile they could make one car in a day. Perhaps we should use the Toyota way to start new XP teams. Zero defects from day one. Hippocrates might suggest: first code no defects.

Lean Manufacturing on the Wiki
Toyota Production System: Beyond Large-Scale Production, Taiichi Ono,
Managing the Design Factory: The Product Developer's Toolkit, Donald Reinertsen
Hippocrates

OpenSpace: Why Too Little Refactoring?
This topic was proposed and moderated by Mark Windholtz.
The question to frame this discussion was: Has anyone ever seen a team that did too much Refactoring? If so what was the bad result that occurred? As it turned out no one in the group of 30 had seen a team that suffered from too much Refactoring. There were some cases where and individual on a team did too much Refactoring and did not get their tasks completed. This seemed like a lesser problem than not enough Refactoring by the whole team.
The group then discussed the symptoms of not enough Refactoring. Team velocity is reduced. Change in the code hurts, so change becomes more painful. The discussion then moved to how to encourage Refactoring.
One problem is that programmers simply do not recognize bad code smells. Some of this is caused by the fact that programmers have had to put up with disorganized source code for so long they have a hard time seeing at the micro level what changes are necessary in order to make the small steps toward a clean code base. To sensitize a programming team the coach can hand out cards with smells on them, and organize a Smells Scavenger hunt. Or spend a hour a day on targeted learning on techniques of good code development. The best situations occur when the Customer, coach, and programmers all agree on the importance of Refactoring for sustainable progress. There was a discussion about small verses larger scheduled Refactoring. Someone raised the warning that scheduled Refactoring tends to be delayed while causing ever more problems.
As I write this the Toyota approach to starting a project becomes ever more useful. If the team takes Refactoring seriously from the start then these issues of remedial Refactoring will not occur. That is the lean manufacturing approach to quality. Don't fix just the problem fix the process so the problem can't occur.

Why Not Enough Refactoring on the Wiki

So you say you want a Revolution?: David West

In the Agile community (XP, Scrum, ASD, Crystal) no one is using the word *Revolutionary*. Yet there is an underlying tone that hints at revolution. Kent says we should leave the mountain culture of scarcity and move to the forest of abundance. And the founders of the Agile Alliance published a *manifesto.*

Why is there this undercurrent of revolution?

Some are just mad as hell and aren't going to take it anymore. Some want better programmers in order to create better products. Some want to transform the world to be a better place. Some want to inherit the momentum of the declining Object Oriented movement and redirect it toward better processes.

How do you properly conduct a revolution?

The people involved must (1) understand the vision and the dark side; (2) convert the masses; (3) establish a cadre; (4) guard against counter-revolution.

The vision of the Agile community is not yet well enough defined. XP has the 4-Vaules and the Agile Alliance has the manifesto but these are not compelling enough. It is the ineffable that is missing. Art and Creativity were driven out of programming 20 years ago. Two recent concepts have helped reintroduce the ineffable - the introduction of *the Quality without a Name* and the growing understanding in the role of *Evolution* in creating complex systems.

Convert the masses. But who are the masses? West suggests not the programmers, accountants, or managers. The masses are the Users - the real end users. When they begin demanding high quality software the revolution will be complete.

To establish cadres we need to build a community. How do the techniques of test first development and pair programming get passed on? Already we see projects called XP that are not using the practices! West seemed to advocate some form of apprenticeships, guild, or certification. While this has been discounted by most of the Agile leaders, he insists that throughout the history of movements a symbolic passing of knowledge has always been necessary in maintaining momentum and clarity. He takes a step further and provides a business case for guild-like status. He suggests that there is leverage and value in being part of a group.

To prevent counter-revolution the movement must guard against five typical dangers. *Co-option* occurs when the heavy processes call themselves Agile but change little else. It also occurs when hacker teams call themselves XP but do not build tests or integrate often. *Marginalization* occurs if Agile is driven into niche, such as only teams under 20. *Nesting* occurs if the community accepts the Marginalization, and becomes content with a niche. *Exegesis* occurs if a movement becomes overly academic. *Ossification* occurs is the movement gets overly concerned with

static definitions as in the discussion of what is pure XP versus standard XP. Change continues we must be in tune with it rather than with the current situation.

To conclude, David West advised that agile community should give up on revolution or do it right. He believes that XP could do it, but that XP is not yet engaged.

XP Fest
Extreme Programmers pride themselves in being hands-on the code. So how can you have an XP conference that talks about code but does not do it? You can't! XP-Fest was a live demonstration of the practices at the edge of chaos. A room was set-up with PCs and development environments. A customer was found from a local non-profit organization who wanted a web based Event Registration System for a non-profit counseling services. The time available was 9 hours of conference time split into 6 iterations each 1.5 hours long. Each iteration had a planning session to schedule the features to be completed. And if that was not enough, programmer turn-over per iteration was 75%. On top of that 75% of programmers did not know the Ruby language. So what was produced under these ridiculous constraints? 13 Story cards were done, 4 partially done, 4 never scheduled. The customer said he was really happy with what was accomplished and believed that with just a little more work it would be ready to put into production.

Sponsors reception

Wednesday 7-August-2002

XP Fishbowl
Provide bits of badly written partially tested code.
Have two commentators, a coach, a customer, and two pairs of XP programmer volunteers from the audience.
Rotate the programmers every 10-15 minutes during the 1.5 hour fishbowl. And see what happens!
The commentators humorously harassed the XP programmers yet within 40 minutes they get the first green bar passing test. Five minutes for the next passing test. Pushing XP over the limits seems to be an interest in this community. And while this is not the way anyone would prefer to run a project it is interesting to experiment to see how the rules function at the far-out margins.

Panel Discussion: Agile Experiences
Managers from lots of companies discussed their XP experiences and shared learning's.
Companies doing XP: ABB; Saber the travel company has 300 XP programmers; Symantec uses XP for1/3 of enterprise products; Quest Communications; Motorola has 200 XP programmers on 3 continents.
Other companies benefiting from XP: Pepsi, Chicago Board Of Trade, Chicago Fed, Freddie Mac, Dow Jones, TRW

The **VP of Saber** (I failed to note his name) gave a great list of learning's including the following:

Saber has 300 programmers, 200 technical support, and 100 product managers using XP.

- The customer should have a meeting before the iteration planning game meeting
- Don't talk about 40-hour week to upper management. Talk about *optimal pace*.
- Don't emphasis pair programming
- Don't over market
- Explain that XP takes a long learning curve
- Say it's OK to be skeptical just be open-minded.
- Initially: 1/3 people bought in to XP, 1/3 waited, 1/3 were skepital
- Eventually: 85% eventually bought in, 10% agreed to play along, 3% never buy in.
- Cut that 3% quickly!
- The rest of the team will support this cut.
- Try XP with a flat lined, important project.
- "XP was a god send" – VP at Saber
- Hold frequent executive briefings
- Partial investments in XP projects are never lost.
- Get customer buy-in
- Walk into the XP lab and see "intensity in the Air"
- Watch out for not completed stories.
- The customer must sign off.
- Do full XP, not just parts.
- Use traditional project management for the non-software tasks.
- Unless all of your code is XP you still need 2-4 iterations to deploy.

Independent consultant, **Jutta Eckstein** coached 170 programmers in coordinated teams of 20-40. And started by suggesting that "take responsibility" should be an additional practice. She made the following points.

- People must be able to take responsibility.
- Responsibility is more difficult in larger groups
- Common room issues are different with a large team
- Communication is key
- Move people around
- Do not build architecture up front.
- Up front architecture stops learning
- Put some technical service personnel on the customer team.
- Architectural stability means death.
- Large organization must break the rules to make progress.
- Don't be dogmatic.

Invited After Lunch Talk: Kent Beck

The Conversation in Agile Software Development: Alistair Cöckburn

Alistair provided a very entertaining talk that suggested that software projects are faced with a diversity of problems and so software developers should have diversity of techniques to address those problems.

Closing Address: Ron Jeffries
Ron reviewed Arthur C. Clark's Laws.
XP is a community of Values like fly-fishing.
XP is what you know after you have done the practices for a long time.
He challenged the group to find holes in the XP model and then "push on".

Farewell: Angelique Martin, Laurie Williams, Don Wells

Summary Impressions

I noticed three shifts from last years conference.
One shift from will it work, will it sell to what holes need to be plugged.
A second shift was from thinking that Re-work is wasteful to understanding that Bug Fixing is the primary wasteful activity.
A third shift is from worries if XP is CMM compliant to asking if Watts Humphrey Agile compliant?

The major discussion was between differentiating techniques versus finding commonality. One side says Agile and XP is different and needs to assert itself or it will die as a movement. The other says, Agile is not so different and we should be able to get along. They are both right but they are answering two separate questions. One question is how does Agile relate to things non-Agile (but who admits to membership in this group these days anyway?). The other question is how does Agile relate to XP in particular.

Next Year XP Universe is tentatively scheduled for 10-14 August in the Eastern USA.

Hope you found this as interesting,
Send your comments to: Mark@Objectwind.com

-Mark Windholtz
www.objectwind.com Cincinnati, Ohio; September 2002