

Report on the Agile Development Conference

June 22-26, 2004 Salt Lake City, Utah

(provided by Scott Duncan)

This was the second ADC conference. Next year, ADC and the XP Agile Universe conference (held in August in Calgary, Canada) will be combined into one. Both focus on experiences using agile methods and attract many of the same speakers/attendees.

Unlike most conferences, the ADC did not rely on standard 40-45 minute “presentations,” but used a mixture of half-day tutorials and 90-minute “peer-to-peer” discussions for most sessions. There were “experience reports” and research papers that were more like standard presentations but they did not seem well attended. Daily “Open Space” sessions allowed attendees to discuss topics of their own invention. A Retrospective session was held at the end of each day. However, as the week progressed, these latter two sessions had fewer and fewer attendees.

Wednesday, June 23rd

Keynote: “Adult Behavior on Projects” (Tim Lister)

Tim Lister, co-author with Tom DeMarco of *Peopeware* and *Dancing With Bears*, delivered this presentation. The former book addresses the human side of software development, fitting well with the people-centric approach of agile methods. The latter, addresses project risk, and this was the theme of Lister’s presentation.

Lister’s main message seemed to be that “you have to be able to talk out loud about risks.” He used the metaphor of the “dead fish on the table” (or the “elephant in the living room”) to describe those things about or on a project that everything knows smell bad, but people act like don’t exist because the culture doesn’t really permit/encourage talking about them until it is too late. (At a later project management session by Jim Highsmith, one of the better-known agile methods authors, attendees took on the name of the “Dead Fish Society” with a URL already registered though a true site with content does not yet exist.

Lister said, “Avoiding risk lowers the value of the product”; however, “the better you get, the more risk you can take.” This is an important theme of *Dancing With Bears*, as DeMarco and Lister believe that all the predictable and easy software has already been written. So taking on riskier projects is important to achieve significant value. It is important then, not to manage a lot of predictable projects well but to be able to know if you are working on projects that are really worth doing.

To better manage the risk on such projects, Lister recommended “incremental-ism” as “a great risk management strategy.” Incremental (and frequent) delivery of software that has valuable, working functionality is a cornerstone of agile methods. Lister asked, “How does the customer know what they need if they don’t know what they can get?” As both customer and developer learn more about what is possible, both learn that “all the really interesting requirements are invented,” i.e., not fully-specified in detail ahead of time.

To make this kind of risk-management and agile delivery approach work requires a lot of face-to-face contact between customer and developer and an “immersion” in a project (or perhaps two at most). This allows people to “work hard and go home.” It also avoids the too typical lifecycle in contracted software: Requirements, Analysis, Design, Code, Test, LITIGATE!

Peer-to-Peer Sessions: Foundations for Agile Development (Steve Berczuk) Agile Enablement Patterns (Dan North and Richard Watt)

As with most of the PtP sessions, though the idea was to have people “experienced” with agile methods as participants, most attendees seemed not to have a strong background. Hence, much what occurred in both tended to be presentation by the speaker(s) followed by some brainstorming. Some interesting concepts were highlighted and presenters were asked to post their results to the ADC “wiki” (www.agiledevelopmentconference.com/wiki). [A “wiki” is server software that allows users to create and edit Web page content using any browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal pages on the fly.]

Open Space “Kickoff”

The idea of the Open Space sessions was to allow anyone attending the conference to suggest topics for impromptu small group discussions, some of which might be continued throughout the week at other, shorter, end-of-day Open Space periods. About 12 topics were suggested. The interesting feature of this kind of session was that there were 5 chairs placed in the center of an area devoted to the discussion. Only people sitting in a chair could speak and one of the chairs had to be open at all times. So if a person sat down, at least one person had to stand up making a new empty seat. (Not sure how this might work in routine meetings within a company. Some agile methods advocate “daily stand-up meetings” (i.e., no chairs for anyone) as a way to ensure constant communication but make sure it is to the point and brief.)

I sat in on a topic, which asked, “Are agile methods just an excuse for hacking?” As you might imagine, many people gathered around this one. The person proposing the topic was being deliberately provocative. However, he had a work situation where people seemed to be using agile methods as an excuse for pure ad hoc behavior without understanding the methods and wanted arguments against what they were claiming was “agility.”

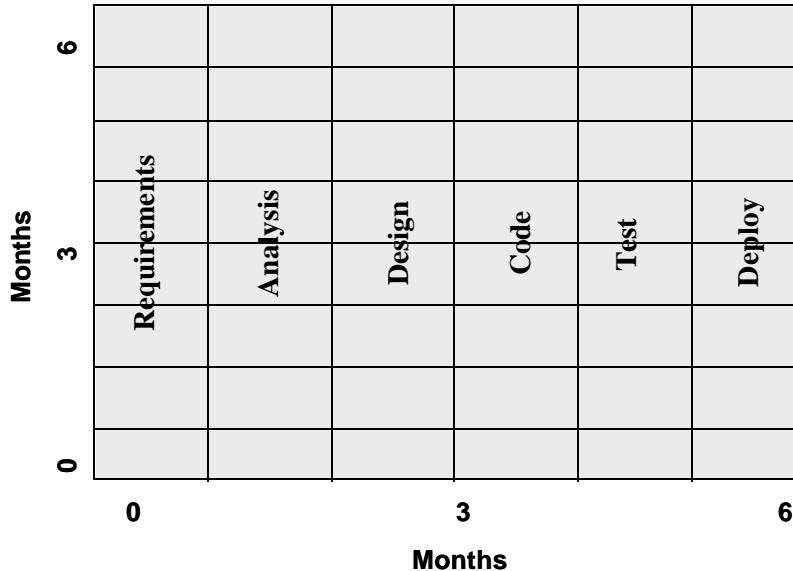
Key ideas from the first day

A key project success factor, because of the impact on requirements at the front end and satisfaction at the back end, is the existence of “lots of systems with lots of people who have a stake in parts of” them. The agile principle of constant communication and feedback seeks to address this.

It was noted several times during the Conference: “There’s nowhere to hide in agile projects.” Compared to typical project development approaches where people can “hide behind someone else’s schedule,” agile projects require people to make their status known daily. This helps maintain the constant pace, acceptance test feedback, and integration advantages agile methods pursue.

An unusual comment was that “to create change, follow the line of most resistance, i.e., the next person in line who won’t let it happen.” As opposed to finding the line of least resistance, it may be better to deal with major resistances, which, once overcome, eliminate dangerous backtracking late in a project.

The following chart illustrates how one group presented the agile concept of incremental delivery. If one ignores the horizontal lines and uses the X-axis to represent project time, the chart illustrates the traditional “waterfall” approach where each phase is completed before the next begins (or close to that even if some overlapping and some feedback between adjacent phases exists). Envision the agile approach as using the Y-axis such that the horizontal lines cutting across all phases of a project



represented incremental development and delivery. An incremental approach is fundamental to agile methods and demands a dramatically different model for how a project is conducted. This involves a different model for the delivery of project value and an equally different model of customer and developer involvement.

Finally, it was emphasized that an agile approach to large projects should turn people’s attention to values and patterns over practices. That is, instead of battles over individual programming techniques and behaviors, stress getting strong agreement on the higher-level agile values and principles to reduce risk, increase communication, and keep project status very visible.

Thursday, June 24th

Tutorial: Large-Scale Agile Development (Ron Crocker)

Crocker is a manager of software development at Motorola and used the tutorial to describe how he adapted practices from XP to 2 projects done by 9-10 teams located in India, Japan, Singapore, China, Australia and the United States. Crocker’s projects were “demonstrations” or proof-of-concept systems with reasonably well-known requirements as they involved wireless communication and prevention of interference between frequencies. But the systems, though not commercial, were fully functional.

Many of the XP practices were not pursued because Crocker did not feel they were “scalable.” He settled on XP’s approach to sustainable pace, refactoring, simple design, testing, and continuous integration while having to forgo things like an on-site customer, collective ownership and pair programming, coding standards, and short releases. Some of the latter, he felt were very “local” matters for the individual teams and he simply did not want to try to enforce, for example, a project-wide coding standard as it was a “battle” he was not prepared to fight.

One of Crocker's major cautions was that it is hard for agile teams to work with non-agile teams because the latter's dates/process tend to dominate the effort. For example, one team insisted they had to deliver certain things according to their methodology, feeling they could not follow an incremental lifecycle. Crocker told them what they had to deliver by a certain date for his project and left how they did this up to them. The group ended up delivering the total functionality assigned to them by that date. Crocker said they then spent the rest of the project's development time, as many projects do, correcting bugs, clarifying requirements misunderstandings, responding to client changes, etc.

Crocker advised taking an approach that would result in doing "just enough work to let the next person do their job." This somewhat relates to the XP concept of "You Aren't Gonna' Need It" (YAGNI) which argues against architecture and design work that does not support the immediately known (and committed to) functionality for the foreseeable future. The latter could be the next iteration or a series of near-term iterations. But YAGNI argues against a fully committed architecture and design up front when customer changes are likely to obsolete many of those decisions. Thus, doing enough work for the next person to do their job constantly focuses on exactly what is needed for the work to progress (incrementally) and for functionality to be (incrementally) verified.

Crocker finished by describing what he called "strata-based development" where a team would focus on building functionality across all subsystems so they could locally integrate, and then the project, as a whole, could integrate the various levels of functionality. The typical approach in organizations might be to have all work in a given subsystem done by the same team, but Crocker pointed out that this led to "silo" mentality and improper "ownership" at a structural, not functional level.

Peer-to-Peer: Non-XP Words for Agile Development (Alistair Cockburn (pronounced "Coburn") and Jeff Patton)

Cockburn is the founder of the Crystal series of agile methods. This session, though, was dedicated to taking many of the XP practices and seeking attendee feedback on alternative ways of describing the practices applied at a "dial setting" of less than 10. XP proponents have described the idea of XP practices as taking good practices to the extreme. For example, if code reviews are a good thing, do them all the time, which results in pair programming. If customer feedback is important, have it all the time by having the customer on the development team. This sessions was to find terms to describe a less extreme approach to XP practices than "turning the dial all the way up to 10." Indeed, Ward Cunningham, who originated many of the agile practices in XP, said he "had the dial set at about 8 or 8.5" when he developed the practices.

Friday, June 25th

Tutorial: Collaboration Works! Facilitation Skills for Agile Teams (Ellen Gottesdiener)

Despite the title, this tutorial covered typical facilitation material/techniques. It was, however, quite a good tutorial and the most "organized" session I attended during the week. It was clear that this was a tutorial Gottesdiener presented often. Her early introduction to the tutorial and our initial exercises were actually examples of topics she covered later in the tutorial. In effect, she demonstrated the technique, then reflected on it after we had personally experienced (using) it (as opposed to talking about it first, then having an "exercise" afterwards.)

There was much more material provided in the handout than was covered in any detail during the session. What follows are ideas that were covered in some detail and approximately in the order in which they were discussed.

We began by being asked to imagine leaving the session satisfied and feeling able to use the information. Then, we were asked to write down a question that, if addressed during the tutorial, would make us really feel that way. Many questions offered had to do with handling difficult situations when facilitating meetings or discussions, so Gottesdiener focused on that subject for most of the detailed material.

She began by suggesting that, rather than avoiding conflict, we try to “move conflict out of energy that stays negative.” That is, let people express what bothers them, and focus the group on coming up with a solution, turning the negative energy into something directed toward change. It is critical that the facilitator not allow the group to place the responsibility for solutions on the facilitator. The group must come up with the solutions, which is why it can be a mistake to have the facilitator be someone with a stake in the outcome of decisions. This can be difficult to overcome in many business situations where an impartial facilitator may not often be available. On the other hand, Gottesdiener noted that, if the facilitator can just make his/her position known in such cases, the group can then judge if the facilitation is even-handed.

One of the common themes people noted was the difficulty in coming to a decision or having the decision followed effectively once the meeting ends. Gottesdiener said it is important, at the outset, to “delineate the decisions that have to be made” and “clarify the ‘rule’ to be used to make a decision,” e.g., consensus, majority, 500lb gorilla, etc. In this way, people can be kept on topic and not be “surprised” by how the decision is made. It was noted, however, that it was not effective to have a group brought together solely to express public agreement with what a decision-maker has already determined will be done.

Referring back to the opening “exercise,” Gottesdiener suggested beginning a meeting by giving people an image (e.g., “Imagine you leave here saying ‘I can go do that’.”), then asking a focus question (e.g., “What issue do you need answered most so you’ll feel that way?”). She said this allows people to envision the end result they want and then identify the issues they feel are most important in arriving at that result. This puts individual “agendas” squarely on the table and helps remove the negative connotations of “politics” from meetings. Everyone has an “agenda,” but it is the hidden ones that produce dysfunctional teams and decisions.

Some time was then spent discussing the team-building cycle known as “Forming, Storming, Norming, and Performing.” Gottesdiener noted that a group psychologist named Tuckman who originated this cycle, later added “Adjourning” to address how teams come to a satisfactory end, rather than having the team “disbanded” and people silently move off to other projects and teams. It was noted that the cycle is not “linear,” i.e., a team can move back to a prior stage given introduction of new people/issues. When someone mentioned teams where people seemed to have “given in” on issues rather than the team coming to accepted norms, someone else suggested a stage called “Conforming” might be added to the model. Gottesdiener said that might also be a passive type of “Storming” since people could then end up resisting team decisions/progress later on.

When the attention of the techniques turned to “difficult” situations for facilitators, Gottesdiener suggested the one should point out the behavior to verify awareness of the behavior, but without trying to “interpret” it, i.e., without trying to say why the facilitator might “think” the person is acting that way.

Then, the facilitator can “verify the inference they perceive from the behavior,” i.e., suggest what they think it might mean to make the person aware of how it could be affecting others. Again, this helps “make difficult behavior the group’s problem, not the facilitator’s.”

Five typical forms of “resistance” exhibited to/in meetings were discussed:

Domineering behavior tends to take over the discussion – the facilitator can physically move away from that person toward others without other comment on the situation to show an effort at changing the focus to other people; the facilitator can use a “parking lot” approach to get the person’s issues down without having to deal with them at that moment; or, more forcefully, the facilitator can introduce the idea of a “red card” at the outset so anyone can raise a red card as a physical, but no verbal, sign of discomfort with such behavior. (One person noted that in their luncheon meetings, domineering people tended to eat slower and less than others. He said their group found the phrase “Are you gonna’ eat that?” as a non-threatening “clue” to indicate a person taking over the conversation.)

Complaining behavior tends to introduce more issues than resolving any – the facilitator can establish the expectation early on that any problem must also have some initial recommendation accompanying it and that the group will be asked if anyone will take responsibility for the issue. (It would be agreed ahead of time that if nobody takes ownership, the issue must be dropped, regardless of how valid someone thinks it is. An issue with no ownership will produce no decision or action.)

Silent One behavior allows people to stay out of the decision – the facilitator needs to “create a safe structure for participation, including non-verbal chances to participate. (The additional handout material addresses some techniques for doing this, but one idea mentioned was to “practice using silence to open space for people to speak up.”)

Latecomer behavior leads to reduced time to address issues – the facilitator must establish the expectation that meetings start on time (including coming back from breaks), but they can also ask the group what they think such behavior may mean. (Again, this makes the problem one for the group and does not turn the facilitator into an “authority figure.”)

Preoccupied behavior allows people to separate themselves from the meeting – the facilitator can address this through asking people to do small group work, giving those exhibiting such behavior a specific assignment, along with everyone else.

Finally, one of the last ideas mentioned, though not discussed at much length, was that “to think well together, we must think well individually.” That is, an effective team cannot be made up of people who, individually, find it difficult to come to decisions and pursue them consistently, including recognizing when changes to them are required.

Friday Afternoon

Most of the afternoon was “free time” when people went off to do things individually or in groups, with or without relationship to Conference topics, though the former was encouraged. For some, this was an opportunity to pursue more in-depth Open Space discussions. I chose to pursue lining up key people for my own Peer-to-Peer session the next morning given what I had seen at other PtP sessions. Fortunately, I had support from some of these key people in getting others to commit to come to the session.

Saturday, June 26th

Peer-to-Peer: What a Customer of Agile Methods Should Understand (Scott Duncan)

My justification for being at the Conference to begin with was to solicit answers to this topic from people with experience in using agile methods, especially in situations where customers were used to large project process and documentation, such as government contracting situations. Alternatively people were asked to suggest the problems they had seen when agile methods met more formal ones. This material will be input to the IEEE 1648 Working Group whose goal is to develop a Recommended Practice for customers/acquirers who would be initiating and conducting projects using agile methods. (An RP document uses the word “should” in its statements where a full Standard uses “must” and “will” and a purely guidance document says “may.” This puts the RP in between the other two types, but does allow it to be referenced in a contract.)

Here, in approximately verbatim terms and in the order suggested, is the feedback from the session:

Understand the iterative development roots of agile methods.

Understand agile methods depend on better people.

Customers, you have the right to:

- an overall plan, to know what can be accomplished, when, and at what cost
- get the most possible value out of every programming week
- see progress in a working system, proven to work by passing repeatable tests that you specify
- change you mind, substitute functionality, and change priorities without paying exorbitant costs
- be informed of schedule changes, in time to choose how to reduce scope to restore the original date
- be left with a useful working system reflecting the investment to date should the project need to be stopped

Developers, you have the right to:

- know what is needed, with clear declarations of priorities
- produce quality work at all times
- ask for and receive help from peers, superiors and customers
- make and update your own estimates
- accept your responsibilities instead of having them assign to you

Early delivery of business value.

Customer involvement throughout the project. (Frequent feedback and reference points.)

Contract scope negotiable throughout.

Tolerance for non-specification level of requirements (demonstrated through verification).

Measure progress in terms of delivered features instead of completed/partial activities.

Customer will be “happier” with both the relationship and results.

Spend an enormous amount of time interacting with the developer.

Testing is an integral part of the whole schedule, not just at the end.

Hard to tell if the “process” is being followed. (How to verify this? *By what’s delivered and how, perhaps?*)

Customer responsibility in defining acceptance criteria.

Delivery of shippable/testable system at least every 90 (“n”?) days.

Either party can cancel after any delivery/iteration.

Stakeholder and developer contact every “m” days/weeks.

Requirements renegotiation every “p” iterations.

How well are “best practices” are being followed?

How to decide which “method” to choose.

Different advice/guidance for different (kinds of) customers.

Customer has to give cost-benefit of changes.

How customer defines “done” or completeness throughout project.

How to guide and educate the customer in how to do these things.

Flexibility in scope but binding in current iteration.
Will contract define what we can foresee.
Payment milestones against acceptance tests.
No 3rd party arbiters of the meaning of what is contracted (i.e., no compliance officers).
Commitment to a team for “k” iterations.
Flexibility in work practices.
Automated tests – test coverage.
Developer should say what “standards” they have for defining requirements, design, implementation, and test definition/practice.
Methodology does not solve all problems.
Customer does not have the silver bullet.
Very clear about what is/is not covered in the RP.
Allow team to refactor.
Receive the documentation that the customer wants.
Will pull together methods but not in a pre-defined sequence.
Consider needing less documentation than you are used to.
System growth over time, not (fully) pre-defined.
How you are going to calculate earned-value, cost, schedule, etc.
Early release. (Problem of not putting releases into production. Key to benefits is having system really used.)
“Right to ship”
Value of feedback
Value of trust
“Bills of Responsibility”
Where management has hands off of developer’s rights.
Expect high efficiency and low ceremony.
Identify any RP content related to contracts, as not applicable to internal projects.
How to negotiate changes (as part of the contract).

Saturday Afternoon

Unfortunately, illness prevented me from attending Alistair Cockburn’s tutorial on Crystal methods; however, I have a draft of his upcoming book on this subject.

Final Thoughts

This conference and the reading I’ve done since then of articles and books by Cockburn and Highsmith have helped clarify the philosophy behind agile methods for me. Alistair Cockburn writes, in his upcoming book entitled *Crystal Clear*: “I hope to reach into the *feeling* of the project.” Something “most methodology descriptions miss....” This characterizes one reason I find his writings on agile methods enlightening: he describes “how to do,” of course, but also how to tell you’re doing it right. Without this, using any methodology becomes a purely mechanistic activity whose “rightness” gets defined by compliance to practices rather than effectiveness in results. Highsmith, in an article entitled “Software Ascents,” describes key software development concepts by paralleling them with activities and preparation associated with rock/mountain climbing. I refer to you material available through their websites (alistair.cockburn.us and www.jimhighsmith.com) and that of the Agile Alliance (www.agilealliance.org).