

## **Notes from the “Introducing Agile” Goldfish Bowl, Agile Development Conference, June 2003**

Invited panelists were Mary Lynn Manns, John Goodsen, Jeff DeLuca, Dane Falkner and Mike Cohn. Just about everyone joined the panel at some time during the session and contributed their experiences and ideas.

Mary Lynn Manns explained the goldfish bowl mechanism. We started with a brief statement from each of our panelists;

Mary Lynn has a new book (with Linda Rising) on the subject of introducing new ideas to organizations, so she’s been doing a lot of research in this area. She has seen people make two main mistakes:

Talk about a new idea generically – “What I have heard in general” – instead of finding the pain and saying how the new idea will solve the problem.  
Try to be the change leader all by themselves.

John Goodsen has worked on XP projects for several years. He sees change as occurring from the bottom up. His suggestion: Show ROI. Compare the old way, for example a long term project, to an iterative project that brings quicker ROI.

John also sees fear about “not doing design”. In fact, with agile processes, we value design so much we do it every day. You need buy-in from managers who come from an engineering background.

At the developer level, show how agile will help them gain control of their life, let them give realistic estimates, gain quality of life.

At the line manager level, show how agile will help them gain control of their project.

At the CEO level, show the ROI.

Dane Falkner has a white paper on <http://www.dsdm.org/> which can be applied to any agile process, not just DSDM. You are always selling. Analyze your market, what stage is it in? Maturing, declining, growing? Identify your stakeholders and a champion.

Mike Cohn works a lot as a “salvage” consultant, salvaging messed-up projects. He has an article in the IEEE journal which came with our conference materials. When he comes into troubled projects, he finds that problems make it easy to “sell” agile. What is hard is going into the engineering groups and getting them to buy in.

Jeff DeLuca uses Feature Driven Development. He talked about going from the current status quo to the new status quo. The change model has several stages. The first is chaos, where your team isn’t sure what the new things are, some groups are loyal to old

ways, some people are unsure and disruptive. People mistake this chaos stage for the new status quo. To avoid this, explain the change model up front, set expectations.

Mary Lynn brought up the subject of skeptics. They can be good for your project or they can shanghai your efforts. You can't keep them out of the way. Instead, use the skeptics. They are a great way to find problems so you can address them.

John said to approach people according to their level of training. Those who have only the basic level need things spelled out. Integrate training throughout the change process.

The rest of these notes don't attribute who said what, because I didn't always note it. The ideas are the important thing.

50% of the team should already know how to be agile, in other words, actually have experience working in an agile team. Hire people who know how.

There's never a technical problem, there are only people problems.

Not everyone can afford to hire experienced people. After two or three iterations, the team usually gets the hang of agile processes.

Management is convinced by seeing results. How do you predict the economic impact?

When you start iterative development, use retrospectives to provide people a place to talk about their fears and give feedback. Let people know that it's normal to be fearful. Be open, teach them it is ok to say they are fearful or uncomfortable. Discuss it, learn, make decisions and move on.

Demand delivery at the end of the determined time period, whatever it is. If it's a month, every month there will be an integration, a customer review, and so on. It forces people to choose.

It's ok to make an error once, but don't make the same one twice!

One panelist described a new agile project where 16 teams work closely together on one project. It's a challenge to get them communicating and integrating. They plan iterations one month in advance. Architecture is a big issue. Training is another big issue, there are 150 people. As a QA person, he's looking at how they can improve the process.

One approach to giving the team expertise is to identify one person for one skill such as unit testing, let that person learn it and pass the knowledge along. Plant seeds.

One team does iteration assessments and has a way for people to provide feedback either anonymously or by name. Over time, fewer comments have been anonymous. When someone breaks a build, they know who it was but it isn't finger pointing.

If people who have gripes are given a forum to express it, that can generate change. If you're having problems, chances are that other people on your team do to.

One approach to introducing agile is to plant seeds, don't push too much.

Mistakes organizations make: they come up with a long term plan for change, and if it goes awry, they throw up their hands. Try shorter plans, then assess what worked and what didn't. This allows you to celebrate small successes along the way and solve problems.

One manager of QA, test and process had the dilemma of how people trained in a waterfall process could help the XP team. He feels they can't use the term 'process' anymore. They teach the principle and let programmers choose the best way.

When you are deep in chaos, how do you iteratively learn and grow while making deadlines?

Need change agents at top and bottom. You have to just get through the chaos.

One participant has been doing agile development informally with three week cycles. They have a new CIO. They'd like to be a bit more formalized.

Having a champion is important. Having a dynamic user organization that is involved helps. Bottom up effort can win over top management.

"Change your organization or change your organization"!

Not all developers have equal productivity, there can be up to a 10x factor difference. Less productive developers can feel exposed by agile development. You need a lot of courage to face this problem. On one project, they had to cut the network access off except in the 'pit' where the team pair programmed. This showed who was avoiding pairing. Sometimes you have to make tough decisions and get rid of people who don't want to try the new way. A smaller team with productive developers can be more productive than a large one with less effective developers.

A participant who came to the software business more recently found an article on Scrum and decided to try the 30 day cycle himself. It worked well, management saw this and made him a lead over more senior programmers. He suggested just getting people to try a new process for a few weeks and see how they like it.

Forcing people to do something they don't want is detrimental. Lead by example, show results. You produce better code and form better relationships.

Less productive people would rather be in a meeting where everyone is unproductive!

Unproductive people may have other qualities. If they are motivated, they can improve their skills by pairing. If they have no desire to improve, they can be detrimental.

Approach of one team using RUP for three years with mixed results: they didn't want to abandon everything, but chose best practices from agile approaches and built step by step.

A toolkit approach might even consider retaining some practices from the old waterfall style process, such as a plan for the overall project with high level estimates.

Opposing this toolkit approach is the idea that you should try to do all the practices of your chosen process correctly, so that you can intelligently pick and choose the practices appropriate for you. Practices don't stand on their own, the synergy is the important factor. Go whole hog if you can and avoid rationalizing why you aren't doing some practices.

A critical success factor is that the team must have ownership and ability to customize their approach.

Some processes don't demand particular practices. If you want the team to do test first development, Scrum by itself wouldn't prescribe that practice, but XP would.

Compared to what most organizations do, any agile approach will help.

All organizations have problems and some are similar. Accept and expect chaos as you implement agile processes.