2008

PACIFIC NW
SOFTWARE QUALITY CONFERENCE

Collaborative Quality

October 13-15, 2008

Conference Paper excerpt
From the

CONFERENCE PROCEEDINGS

Actually, It IS All About You!

Jim Brosseau

Biography

Jim Brosseau

Jim has been in the software industry since 1980, in a variety of roles and responsibilities. He has worked in the QA role, and has acted as team lead, project manager, and director. Jim has wide experience project management, software estimation, quality assurance and testing, peer reviews, and requirements management. He has provided training and mentoring in all these areas, both publicly and onsite, with clients on three continents.

He has published numerous technical articles, and has presented at major conferences and local professional associations. His first book, Software Teamwork: Taking Ownership for Success, was published in 2007 by Addison-Wesley Professional. Jim lives with his wife and two children in Vancouver.

Actually, It IS All About You!

Jim Brosseau

August 11, 2008

Most branded approaches for software development are pitched with little active consideration for the team that will be dealing with that new approach. One could cynically suggest this is because these approaches are hailed as silver bullets that are independent of the target team, but it goes deeper than that.

If we really want to improve the way we develop software, we need to start by understanding the team itself: what each participant brings to the table. We need to bring all this out into the open, discuss our shared goals, reflect on our cultural distinctions, and leverage our combined strengths. The better we understand each other on the team, the stronger our trust will be. Our relationships will be more genuine, and our interactions will be more effective. With a consciously managed inventory of our skills, experiences and attitudes, we are better equipped to select which approach will be most effective and how to deploy this refined approach with the team.

This all takes time, to be sure. Consider, though, the cost of miscommunication, disappointments, broken trust and other forms of interpersonal dysfunction on most projects. In that light, the effort to better appreciate everyone involved on the project starts to look much more like an investment than a cost.

Relationships are hidden behind tools and techniques

There are several reasons for neglecting details about the human element, but deeper challenges in failing to do so.

First, the complexity of dealing with the human element when we are dealing with teams and processes can be daunting. Teams come in all shapes and sizes. There are at least as many different motivating factors for participating on projects as there are people. There are many complex emotions that come into play, and this entire mix is constantly changing. It is impossible to codify this complexity and provide a generalized solution as we often do with practices and methodologies. It would need to be simple enough to be easily communicated without losing the audience's attention, but there are too many variables here.

Secondly, most of these branded approaches got their start with a handful of people who had success on a handful of projects. While there may be some discussion of elements such as trust and fearlessness, these are generally dealt with through hand waving and generalities, such as "get along with your teammates" or "have fun". For the most part, concrete emphasis is on the practices themselves. In almost any retrospective I have been a part of, discussions about what worked and what needs to change focus almost exclusively on 'best practices' such as defining scope and

planning, design practices, implementation and validation approaches. Even if symptoms of dysfunction such as the team not getting along are raised, the solution space usually consists of practices we see as within our control – those same practices described above.

Implicit in most successful teams is the relationships that have developed among the team members. Indeed, when you are asked to think back and describe the ***approach*** used on a project that you would have considered successful, there is a good chance that you would describe elements like a change management process, or nightly builds, or test-first development, or tactical access to a customer. If you look deeper, though, and consider ***how*** the team worked together, you will find other, more important elements. The team surely cooperated well with one another, demonstrated a level of trust and shared a common goal. Everyone felt like they belonged, and actively participated in working toward that project's success.

In working with teams, I explicitly asked them to identify the characteristics of their best project memories, carefully avoiding biases towards approaches or relationships. The one common element that everyone has identified as part of their successful project experience was that they had fun.

There are rarely any characteristics raised about tools or technology, and nothing about lifecycles or methodologies. These are necessary, but insufficient. Success is primarily dependent on the relationship between the team members. This does not simply go without saying. It is absolutely critical, and needs to be consciously managed. As a team member, it IS all about you.

There will be relationships between team members whether or not these are consciously observed or managed. Regardless of our selected approach for developing software, people will have to interact with one another. Each of these interactions will be successful or not based on the trust, respect and shared vision of the participants. If all of this is left to chance, how much of a consistent impact can one selected approach have over another?

We need to understand *OUR* roles

It is all too predictable. Whenever my son and daughter are playing together, it is only a matter of time before I hear raised voices. One of them, if not both, will come to me complaining about something the other one did. I've settled into a response that usually works quite well: to ask them to consider their contribution to the situation. What is most fascinating about this is how often a similar dance occurs on project teams.

Whenever a relationship goes sour, even a little bit, there are two sides of the story. It is human nature to pay most attention to the details as they are seen from our perspective. This often exists to the point where we are blind to the idea that there even can be another perspective at all. From our perspective, we usually see that we have been wronged, that our actions were innocuous. Unfortunately, that is rarely

the case. Here are a couple of interesting examples, based on working with clients using our diagnostic product over the years.

The first case comes from a group that performs better than most of the teams I have worked with. While not always performing to that high level, they have dramatically improved over the years. They are at the point where they are a strong example of the effectiveness of conscious focus on practical implementation of reasonable practices and continuous improvement. They are also one of the few companies that has been successful at leveraging outsourced resources, by carefully managing subcontractor performance. Indeed, they are able to do so only because they are one of the few companies that has a quantified handle on their own performance.

Running the diagnostic with this group, we split the responses so that we could compare the development team to the test team, and both to the outsourced team offshore. The question we always start out with is to get people to identify, from a list, their major sources of pain. In this case, the strongest response from the development and test teams was that their subcontractors were causing the most pain. Interestingly, for the group offshore, it was the customer that caused the greatest spike in responses. For a group with one of the strongest outsourcing relationships around, there were certainly some remaining challenges to deal with: both sides perceived that it was the other side that was the challenge.

The second case is with another company located only a short distance away from the first group physically, but almost diametrically opposed to them in terms of performance. This group had been struggling for some time to deliver quality product, and had literally been spending the lion's share of their time dealing with client-reported issues. Here, when asked about their greatest issues, the interesting responses came from a selection called 'other', where respondents could answer in freeform text. While the technical people indicated there were issues that they could deal with, such as a need to do more testing or a lack of understanding of the user's needs, management responses fell into a different category. All their responses indicated that the problem was outside of their sphere of influence: the war in the Middle East, the weakening economy, the weakening US dollar (this, mind you, was several years ago). With that attitude, all they thought they could do was throw up their arms in despair.

This was unfortunate as there really was quite a bit they could have done to better manage their circumstances.

We could be refining our already strong performance and dealing with minor tweaks, or battling stronger demons in a fight for survival. In either case, it is in our best interests to be able to step back when we look at issues to include our own contributions as part of the mix. If we look at all the respondents to the diagnostic to date, 38% have indicated that their customers or contractors were a significant issue they had to deal with. In how many of these could they have stepped back and considered that there were things they could do to better manage that relationship?

In any challenged relationship, there is always something that we could have done differently. The fact that we are in the midst of such a struggle should give us pause to consider our own behaviors. Indeed, one of the most powerful tools I can bring to work with clients is an objective mirror.

We need complete, open communication

Failure to communicate is at the root of almost all challenges we face in software development. Process and procedures are a means of ensuring that we do the right things at the right time. Analysis models help us communicate different aspects of our product in more precise forms than the English language can convey.

Beyond these engineering solutions to communication issues, we also have to deal with the nuances of relationships and teamwork for a complete solution.

Understanding individual motives and honing our skills in active listening and conflict resolution make it still easier to communicate, but there is still more: we need to engage others with candor.

Communicating with candor is more than merely telling the truth. Candor involves full disclosure of all information that you are aware of (and indeed, disclosure of the areas where you don't have all the information), both positive and negative. In order to do this, everyone on the team needs to feel safe in their environment. If there is any lack of trust among the team, or any issues that have not been completely dealt with in the past, it can become too easy to hold back. Candor and hidden agendas are like water and chocolate (if not in that order).

Some people have the self-confidence to be able to walk into almost any situation and speak with complete candor, but it is a very rare team where everyone exhibits this trait. In the stages of team development (forming, storming, norming and performing), it is only when a team has achieved the performing stage that we have built the infrastructure of trust, of openness and caring that supports candor. For teams that have been there, most know that it is all too easy to backslide. It takes effort to get to the performing stage, and effort to stay there.

Why is this important? It is at this stage where the team is firing on all cylinders, working together effectively as a group, and having a good time in the process. With complete candor, the shared memory of the team is more complete, and more consistent. Fewer balls are dropped. Improved efficiency opens the door for the team to be more creative.

Agile principles suggest the most efficient and effective method of communication for a development team is face-to-face conversation, where candor is a must. If you are not consciously maturing your team dynamics to the point where candor is possible, even your agile projects will be at risk.

We are a core part of a larger machine

As our team size grows, we compartmentalize ourselves into specific roles: project manager, scrum master, developer, tester, and a wide range of others. With this often comes the assumption that each role has the responsibility to produce specific products or artifacts: a project schedule, a specification, some code, or something similar. Problems arise when we take this to mean that we are the sole proprietors of the products of our roles: that we have the responsibility of doing it ourselves.

The experience of producing one component of a larger system can range from harrowing to extremely rewarding, or anywhere in between. If we have good information about what that component should be, supported by previous experience, it can be deceptively easy to get the job done. Work breakdown structures help us remember everything we need to consider when building out a project schedule. Document templates give us an outline that allows us to simply fill in the blanks. Even the defined structure of a daily scrum meeting gives us guidance on who needs to participate (everyone) and what needs to be covered.

Unfortunately, when we are doing these things on our own, it is easy to fall into the trap of following the letter rather than the intent. For collaborative events such as scrums, we can still think of this as a DIY activity if only the scrum master considers whether this collaboration was achieved. With one person thinking about whether something is done, it is easy to satisfy our own preconceived notions of completion. When working to some deadline, the task often becomes an exercise of making sure that all the fields are filled in with something within the time constraints. There might be time for polishing if you are lucky enough and there aren't any other pressing tasks to deal with.

We then check off the completion box for that task, but are we really done? As others are in the same boat, review is often superficial. Deep issues only arise much further downstream. For most tasks on a project, if you were to hand them out to be completed independently by different people, you would find that most would complete the task and believe they had done a good job, but provide wildly different solutions. A great example of this is a vision statement: a simple expression of who the key client is and what problem is being solved, along with an identification of the key competitor and the primary differentiation. Ask each person on the team to develop one independently, and you are guaranteed to get very different results, each one being complete. Which is the correct one?

This is a problem.

If done in a collaborative fashion, the dynamics are completely reversed. The differences of perspective are exposed early, highlighting the need for further discussion and clarification. The team works together to come to a common understanding, rather than the team inheriting one viewpoint and having to live with it, as usually happens. Done this way, these discussions occur with the right people at the table, at the right time in the project. Compare this to the late-in-the-project discovery that there was no alignment in vision, that most of us have experienced.

As we split into more detailed roles on a project then, we need to think of our tasks as things we are responsible for ensuring they are done, rather than things we need to do ourselves. In this way, we make sure that different perspectives are appropriately balanced. With broader participation, everyone has a stake in the production of the artifact. With that stake comes a stronger feeling of ownership: it becomes 'our plan' rather than 'his plan', or 'our tests' rather than 'his tests'. Big difference. It helps break down the silos between departments, the handing-over-the-wall of artifacts, and the carrying down from the mount of the stone tablets that drive our work.

The whole group becomes responsible for a successful project. If one person has the lead in ensuring something gets done, that person should never be alone in that activity. Nobody likes to be told what they should be doing, or what they have to do. It is far better to be part of the decision-making process, to have the opportunity to see what is needed and step up and volunteer for the task. The stake and ownership itself will tend to provide success far more often.

The counter-argument of not having enough time to collaborate on these things just doesn't fly. I'm not suggesting that everyone participate together for every component of every task, but anyone affected by a decision should be involved in the decision-making process. If we do things on our own, the collisions downstream because we haven't involved others will cost us far more in the long run. It is these costs that we never account for in our original planning.

Regardless of your role, don't get caught in the trap of thinking that you are solely responsible for the products of that role. A better approach is to be sure that the decisions you are responsible for actually get done, but get done and agreed upon taking different perspectives into account. The perspectives of others can only be clearly understood if they participate: don't do things yourself.

We need to respect the effort behind change

We all deal with change in our lives, generally doing so by avoiding it at all costs. Change is not trivial topic to deal with, which is likely one of the reasons it is so intimidating to all of us. While I often discuss change in the context of the learning curve for training, I find it more appropriate to think of change outside the context of software development. I have been struck by the applicability of the Satir Change Model to software teams.

Virginia Satir (a family therapist, but the analogy is apt, yes?) suggested "Familiarity is more powerful than comfort". That seems to be the case for the software industry as well. There is some overwhelming inertia holding the industry back from realizing its potential, despite our not being in the comfort zone.

One of the best introductions to the Satir model comes from Steven Smith (1), one of the keynotes at this year's conference. The model is simple and clear, but most importantly makes sense because we can instantly see ourselves in the model. We naturally have an affinity for the status quo. For myriad reasons, change is something we detest. Quitting smoking, traveling to exotic places, developing

software in a way that differs from the past (in some places, that 'past' can span decades). There's the unknown, the anticipated chaos, the displacement from our current way of doing things. It is easier to just stay where we are, even if we can envision a better place at the end. As a physicist I know how to express the strength of inertia through formulae. From the perspective of change, we all feel the inertia keeps us in our place of familiarity, whether or not it is comfortable.

Change is far more than simply deciding to do things in a different way. The greater we appreciate and understand all the complexities and nuances, the more effective we can be at fostering effective change at home and in the workplace.

Most critical of these issues is the recognition that change is embodied in a clear sequence of stages. For meaningful change to stick, you can expect to go through all these stages in turn. Deep change will drop us out of our comfort zone, our status quo, and carry us through a stage of disruption and chaos. Quite often we will see this as bad, and leap back to our old ways, back to that soothing place of familiarity. It takes significant effort to continue on our path, to find some mechanism or transforming idea that allows us to recognize the value of seeing the change through. If we have done well, we will reach a point where we have been successful in changing. This can be a tough journey, and at each stage of change there are new reasons to fall back to where we started.

Often, though, we attack change in ways that make the journey tougher than it needs to be. At one conference, it was interesting to note that several people saw their role as one of a change 'inflictor', and indeed this is the way many people foster change as consultants. "You guys need to adopt Scrum..." or "I can help you get to CMMI Level 3...". Recognizing the sequential nature of change is important for accepting that it's not as trivial pointing someone in a certain direction. There are other ways to simplify things, to make change easier to swallow.

A critical element of driving successful change is controlling the magnitude of change. Often, in software development or in life, we take on ludicrously large elements to change. We'll go from a totally chaotic ad-hoc development style (often in the form of every participant using their own preferred approaches from previous jobs), to a complete single way of developing software. While great in theory, this is extremely disruptive in its implementation. Whether agile or not, this complete system is made up of a collection of individual elements, each with their own change profile. Some of these may have little chaos and huge return, while others may be extremely chaotic, and actually have a net negative impact. Throw them all together, and you will have washed out much of the value of the best elements, and imposed an overall change that can be quite daunting. The team can easily find the status quo quite attractive in comparison, and slip back into old habits.

We need to find those atomic elements that will provide us the most value with the least disruption. If we select them correctly, we will find far less pushback, and faster adoption. We also benefit from demonstrating to the team that this change stuff doesn't have to be so painful after all.

The other side of simplifying or making change more attractive is cranking up the engagement from the participants. We have to be careful to present the potential change in a way that is clearly understood in terms that people understand. In addition to this, we need to avoid simply handing this change over as a task for them to do. While we may not see these elements as disruptive, this is because we have already made the required journey. We have already internalized the value and overcome the disruption.

Any change will be easier if the participant is carefully supported, given the time to acclimatize to the different way of doing things, reminded of the value of the change in making their life easier. Participants need to understand how the change brings them more closely into alignment with their own value system. They need to see what is in it for them, and this attraction will vary dramatically across the team.

In the minds of most people, change is frightening and diametrically opposed to remaining in the status quo. If we are careful in how we present change, selecting the low-hanging fruit and fostering the change through our teams, we can dramatically reduce the barriers we often face. Indeed, we can get to the point where change becomes an attractive ongoing way of doing business. We can become a true learning organization, and remove the apparent dichotomy between change and the status quo.

We all need to see the value from our perspectives

Some things sell themselves. Tell an eleven year old that Apple has a new iPod on the market, and she'll save her allowance money for months. The same goes for seven-year-old boys and Pokémon cards. If you have kids around that age, you will certainly understand where I am coming from.

The same principle holds true in the software industry. Tell a young group of developers that you are going to give Scrum a try, and you are not likely to see a great deal of resistance. They are all over it like...well, pick your favorite attraction metaphor here. Get a bigger group of more established (read: older) practitioners and wave the CMMI in front of them, you will often see a similar response. Give almost anyone in the industry a software tool as a means of solving a problem they are dealing with, and…you get the picture.

The problem is that it is not quite as easy to sell these things to everyone on the team. Simply waving the brand around is insufficient to capture the entire audience. You need to engage that whole audience in order for many practices to truly work. There is still hope for bringing good practices into your project, but you need to step away from the hype.

Senior management and many marketing people aren't interested in agile approaches. In fact, from what they have heard they are probably afraid they will have less control and visibility into what is being built than before. Unfortunately, these rumors are likely to have come from weak implementations of good practices in the past. On the other hand, if you talk about giving them a continuous stream of

new features, keeping the product running so there is no big-bang integration, and giving them a voice in the prioritization of these new features, you will hook them quickly.

Talk to an end user about a use case workshop, and their eyes will likely glaze over. Ask them (without jargon) about the different kinds of users for the product, and what each of these users will want to achieve, and you will have an engaged crowd. Dive into discussions about the steps that make the most sense for them to interact with the system to achieve their goals, rather than the normal flow. You can even discuss alternative ways they might achieve their goals, what they might expect when for some reason they can't get to where they want to be, and what state the system should be in before you get started and when you are done, and you have pretty much covered the breadth of use case issues. More importantly, you haven't scared them off with the terminology.

Tell a development team you are seeking a CMMI rating or ISO certification, and you will have a bunch of disillusioned people expecting to waste much of their precious time building needless documents. Again, perceptions based on experience can be strong. Talk about getting together to agree on an approach that will improve predictability on your projects, simplify planning, ease maintenance grief and shorten schedules to boot, and you might get them to sit up and take notice.

If you have a conflict with a co-worker and suggest that the two of you step through a five-step approach for conflict resolution, and their opinion of you will likely drop lower than it already is. Head into the discussion ready to learn something about the other person's perspective and get them to acknowledge that you really do see where they are coming from before proposing solutions to the challenge, and you'll be amazed by the connection you make.

Whenever we work with others, we need to sell reasonable approaches to solving problems first by understanding our audience. There is a strong chance that they are not motivated in the same ways you are, and it is critical that you know WIIFT – what's in it for them.

Once you know that you are well on the way. Pitch to their needs; show them how these approaches address their needs, and the pushback that often occurs with change will dissolve. As an added bonus, this approach will help keep you honest about doing the things that are right for the team as a whole. It will prevent you from being swept up in the hype surrounding trends that might happen to be in vogue today. When you are selling, you need to play to the needs of the audience.

It really is all about you

Most change is driven by selection of new tools or techniques, and is offered up as a point event to the team, with the expectation that there will be immediate improvement within the group. While some of these efforts can have a positive impact, we are not controlling all the variables in this equation.

We need to take the time to focus on the team dynamics, and manage these dynamics such that the group interacts more cohesively. With this approach, any of the usual changes we apply have a much higher likelihood of succeeding and providing even more value than before. Indeed, with the team working together effectively, we can find dramatic improvement without the addition or change of tools or techniques, and we can select changes that are more aligned with the group's needs.

References:

(1) http://www.stevenmsmith.com/my-articles/article/the-satir-change-model.html