

Team Development and Pair Programming – tasks and challenges of the XP coach

Dr. Kamran Sharifabdi
TietoEnator Consulting
P.B.233-Okern
N-0510 Oslo
Norway
+47 – 480 99 053
kamran@tietoenator.com

Dipl. Psych. Claudia Grot
Organizational Psychologist
Manglerud V.51
N-0678 Oslo
Norway
+49 – 170 2859 849
claudia.grot@c2i.net

ABSTRACT

This paper presents different ways an XP project manager can deal with resistance within the team. The XP project was a project with the Norwegian Telecom Industry the team was a mixed team (employees and external consultants) their experience ranging from 3 to 30 years with an average between 5-10 years. The team had never done projects with XP before and especially methods like pair programming and developing the test code before starting to program were met with resistance. This paper describes the different obstacles that were met and focuses on tips and tricks that can be useful for an XP project coach or project-manager(PM). Pair programming is discussed before the broader topic of team development as this is seen to be a great contributing factor to the project success. Working with the team and coaching the pairs took a much greater amount of time than was planned from the project managers (PM) side. We came to the conclusion that a PM who introduces XP needs to be very experienced in people skills: forming a team, overcoming resistance, coaching, solving conflicts and introducing methods very flexible. XP “by the book” seems to be only possible in rare circumstances.

Keywords

Pair programming, team building, role of the project manager, obstacles to pair programming.

1 INTRODUCTION

When you take over or introduce an XP project in a more traditional organisation you can expect that teamwork is rather preached than practiced and you will hardly ever or in very lucky circumstances meet the independent, mature and self running teams that a XP project need in order to produce a successful project result. The task of introducing

XP methods like pair programming falls onto the XP project manager (PM). In this paper we will focus on the PM's role as a facilitator for team development and pair programming and based on our experience as XP coaches generate ideas on how to deal with the obstacles that a PM will in one or another form meet. Pair programming and team development are interlinked with each other. Team building is the more general tasks and pair programming can be seen as a key element of teambuilding. Introducing pair programming will help along the team as has been reported anecdotally for the simple reason as performing together, learning from each other, sharing information and relying on each other is the essence of teamwork which is encouraged in pair programming and spreads out into the whole team as the pairs are shifted and a sense of achievement is reached. Making the pairs work will help making the team work.

2 OBSTACLES MET IN THE NORWEGIAN TELECOM TEAM

Apart from convincing the management to implement a 100% pair programming, working with someone else meets resistance from some programmers.

As in any team it also takes time to develop into a good working pair and some pairs alas do not make it to that stage or only after an enormous amount of coaching, conflict mediation and interventions. So in some pairs we have really wondered about the sense to let them work together and preferred to split them up.

The main obstacles that we have encountered in the Norwegian XP projects were:

- a) the resistance to work together as a pair and
- b) the resistance to write the test code before you code as this was rejected as something unheard of and unfamiliar.

3 TEAM DEVELOPMENT

Team development or team building is the process of transforming a group of individuals with different interests, backgrounds and expertise into an integrated and effective

work unit. It can also be seen as a process of change. The fact of teambuilding is simply stated:

“To help people who work together to function more effectively in teams and to assist the team itself to work more effectively as a whole.”

Effects and questions of team building

Effective teambuilding is concerned with the following functions:

- Improving performance and results
- Making greater use of both individual and team strength – not just concentrating on weaknesses
- Resolving problems about which something can and must be done, and which are within the responsibilities of the team.

The basic questions that an XP team –like all other teams - has to find answers for are:

- What are we here to do?
- How shall we organize ourselves?
- Who is in charge?
- What are our roles, responsibilities and relationships?
- Who in ore outside the organization cares about our success?
- How do we resolve problems and conflicts?
- How will our performance be measured and by whom?
- How are the awards decided?
- How do we fit in with other groups?
- What benefits/support do team members need from the team?

The PM should provide a clear focus on the goal and the XP process and should spend some time to explain it to the team – depending of course on their level of expertise. He should also focus on his role and the rights of developers and customers. He or she also has to work out beforehand how to deal with the issue of bonuses or incentives. If you work with a bonus system it should be a team bonus rewarding the team for special unforeseen efforts. An individual bonus system will undermine the idea of shared responsibility and pair programming. The only exceptions could be individual outstanding efforts that are also acknowledged by the team..

4 PAIR PROGRAMMING – TASKS FOR THE PM

In the XP methodology the work is done by pairs. The pairs should discuss the programming approach to the user story and work out the unit test solution and the test data for the function test.

What to expect when pairs are working well together

So if you start with pair programming which positive phenomena should you expect if it goes well?

- Reduction of defects, better design quality, better defect removal rates
- A sense of satisfaction and reliance on each other

- Learning and information that is exchanged through the pairs (watch out for changed behaviour: people taking over habits and ideas from their colleagues and start working in a different way)
- Improved cooperation within the team; sharing ideas and helping each other
- Roles are taken in turns: people take turns in being the teacher and the taught

In a good pair you will also find a positive form of peer pressure, a sense of focus as the other person's time is not wasted. Working pairs set a higher pace; neither person feels they can slack off. You'll find a focus on finding better solutions and an intense problem solving as they can explore more angles as they would on their own. A good working pair you can even recognize by their body language: eye contact, initiative comes from both and is supported by the other, sometimes silent understanding, sometimes heated exchange, both are focussed on the task and take turns.

What to do when programmers do not write the test code before programming

In the Norwegian Telecom project a clear task assignment was given and the PM took time to explain the advantages of testing before coding. Nevertheless this turned out to be one of the hardest requirements. Writing the test codes was either not done before the coding or it was done with the minimum amount of effort (minimum test procedures required) or some of the guys blatantly try to cheat and write the code first nevertheless.. The PM introduced random checks “you were assigned this task, in the stand up meeting you said you were in the middle of coding – may I see the unit test you have written.” The test was checked and discussed and the pair normally had to be asked to expand it. In the weekly “extended” stand- up meetings the issue of unit testing, why it was required, experiences and improvements was taken up continuously. As time went by the team tried to do a better job.

The job of the PM was it to monitor the progress constantly, take up the issue and help the people to draw their own conclusions.

What to do as a PM when your team resists pair programming -a soft introduction to pair programming

The most common argument against pair programming ran along the lines of: “We have been programming for the last 10 years and always managed to deliver what we needed to deliver without sharing our knowledge”. People on the team were used to having their own way of doing things.

In order to enable them to slowly lower their shields the PM decided to introduce pair programming “the soft way”: First the programmers were asked to come together and discuss the tasks in pairs in order to get a better

understanding of the tasks, its requirements and possible problems that could occur.

They were asked to

1st understand the tasks

2nd to define and detail the approach to the task

3rd to define the unit test (or approach to the unit test) and possible test data - here again with this approach the possibility that both of them would jump right into the programming was much lower.

At this stage each pair was using roughly 30% of the time for these activities.

As this was working quite well, the teams were then asked to sit together and programme. Refactoring was then done with a different partner.

Even this “soft” introduction was met with enough resistance

What to do as a PM when your team resists pair programming – how to deal with common obstacles

As a PM you will have to keep an eye on the following factors because they are obstacles to pair programming:

- i. Someone takes comments as personal criticism or as a general sign of mistrust
- ii. Someone thinks he or she is always right and is not willing to give up their solutions
- iii. Someone always agrees with his or her partner (especially in senior/junior pairs)
- iv. Someone is insecure or anxious about his or her own skill or doesn't want someone else to know that he/she doesn't have all the answers
- v. Someone keeps to himself and doesn't share ideas
- vi. The pair is in a continuous deadlock and cannot make a decision

In case of obstacle 1 (Someone takes comments as personal criticism or as a general sign of mistrust) : when you introduce your team guidelines stress the importance of mutual feedback. If necessary introduce feedback rules on how to give feedback in a fair manner. If necessary come in as a third wheel, listen in or even participate. Is feedback given in a manner that is personally unfair or is the recipient of feedback taking it too personally. If necessary have a talk with the pair to find out what kind of feedback can be accepted and what they want to try out.

In case of obstacle 2 (Someone thinks he or she is always right and is not willing to give up their solutions): Give him or her a personal feedback on what you have observed. Stress that pair programming is a mutual effort and can only work if both are flexible. Have an open discussion with the pair in order to find out why he or she is so unwilling to change. Negotiate the conditions under which

he is prepared to try out something or let the other have his way. If someone insists on always defining the “highway” without giving any space to his partner either find something where he can be useful to the project and work on his own or be prepared to find a replacement. Is he or she inflexible no matter with whom he is working or does he work better with someone else? If so “re-pair”.

In case of obstacle 3, 4 and 5 (Someone always agrees with his or her partner -especially in senior/junior pairs; Someone is insecure or anxious about his or her own skill or doesn't want someone else to know that he/she doesn't have all the answers; Someone keeps to himself and doesn't share ideas): Someone who always agrees or is to shy to give an opinion or doesn't contribute. Again coming in as a third wheel could be an answer, as this will help you to get a feeling for the situation and formulate a feedback based on your own observations. Encourage someone actively to voice an opinion or to criticise something. Encourage him or her and make it clear that even someone with less programming experience will be a useful team member if he or she asks, challenges keeps an eye on other alternatives etc. If someone cannot be useful in one way find out what he can come up with instead. If someone doesn't contribute find out why. Again as in case with the other obstacles negotiate the conditions for contributions. Again make it clear that a contribution is required if he or she wants to be part of the team.

In case of obstacle 6 (The pair is in a continuous deadlock and cannot make a decision): This calls for a talk between the two. Be prepared to come in as a decision maker or find out who they accept as a decision maker or come in as a conflict mediator – because lack of mutual trust tends to have a history.

The conflict needs to be solved before they can resume working again. An important part of the conflict mediation will be to agree on trust building measures. Give it a try before you think about splitting them up. If you have to split them up observe whether the same happens when they work in other teams. Are you dealing with someone who finds it generally hard to trust someone else? If so you are back to the negotiating table. This has typically happened when the two partners are equally strong (and equally stubborn). What is important here is to differentiate between a friendly testing the waters where programmers engage in battles of arguments in order to test the strength of each other and a real deadlock. Is this something that only happens at the beginning of their partnership or a continuous deadlock?

5 TIPS AND TRICKS FOR THE PM

Make working in pairs mandatory. If people have to do something they will bicker but they will give it a try especially if you try to convince them of the benefits for themselves and the project. What is reported is that people give it a careful try and that we could observe as well.

Programmers first code individually and then review changes with their partner. After a time they discover that it is in fact speedier to do it together in the first place. If you are a PM let it happen – they need to feel secure before moving onto the next stage. In your team you will always have pairs who hit it off earlier than others and can serve as good examples.

- In the beginning you will have to come in as a third wheel and join the pairs at their work to get an idea what is going on and to coach them individually.
- Encourage programmers to think aloud is usually a good first step to start working in pairs.
- Encourage the rotation of partners.
- Stress that both pair members are equal partners and equally responsible for the quality of the result. Acknowledge that there will be gaps on knowledge and experience but even a beginner can bring a fresh perspective to the task, ask the right questions and look out for faults.
- Intervene immediately if you observe blaming. Blaming each other is a waste of time. If defects are there they are equally responsible and equally responsible for solving any problems.
- Keep reminding your team there are no “mine” – but “ours”
- Focus on the quality of the outcome – the quality of ideas. Challenge ideas yourself in the spirit of a friendly competition. Encourage people to be outspoken and be outspoken yourself as to set a good example.
- If a team is working very intensely (and you have succeeded in introducing pair programming) sometimes it's a good idea to step in and make them take a break. Overworked programmers are not the ones with excellent ideas and a break can help to get a fresh perspective.
- In general create a climate of achievement: encourage new ideas and give people the freedom and opportunity to try them out while at the same time being very clear on the results you expect i.e. put pressure on them to come up with good working results.

Trust your team to come up with these results. Trusting in someone creates a challenge and works as a motivator at the same time. But trust of course is not blind trust. Your

role as a PM will always require to follow up and control the project process. Creating the opportunities and the right environment is your work package.

6 INFORMATION AND QUESTIONS

For more information, contact: claudia.grot@2i.net or kamran@tietoenator.com

References

- 1-Williams, L.A. The Collaborative Software Process. Dissertation, the University of Utah, August 2000
- 2-The Skilled Facilitator : Practical Wisdom for Developing Effective Groups (The Jossey-Bass Management)by Roger M. Schwarz ISBN 1555426387
- 3-Extreme Programming Explained: Embrace Change by Kent Beck ISBN 0201616416
- 4-Extreme Programming Examined by Giancarlo Succi & Michele Marchesi ISBN 0201710404
- 5- www.extremeprogramming.org
- 6 <http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap>
- 7- <http://xp123.com/xplor>

-