Introducing XP in a start-up

Roberto Deias
Fst s.r.l.
Sesta Strada |Ovest
Z. I. Macchiareddu
09010 Uta, Italy
+39 070 24663030
roberto.deias@fst.it

Giampiero Mugheddu
Fst s.r.l.
Sesta Strada |Ovest
Z. I. Macchiareddu
09010 Uta, Italy
+39 070 2466 3522
giampiero.mugheddu@fst.it

Orlando Murru
Fst s.r.l.
Sesta Strada |Ovest
Macchiareddu, Uta
09010 Uta, Italy
+39 070 2466 3088
orlando.murru@fst.it

Abstract

This paper relates on the on-going process of introducing XP in an Internet Company. After a brief description of our experiences with XP we discuss the issues raised by the introduction of XP in three key areas of our firm: customer relationships, project management, and ISO9001 quality assurance.

Keywords

XP, Extreme Programming, ISO9001, ISO12207, Risk Management.

INTRODUCTION

Fst is a small firm (160 employees), active in designing, building and out-sourcing Internet services. Its specific areas of expertise are building complex portals with multi-channel access, strict security requirements, support for digital signatures and various kinds of financial settlements. Fst is not a software house and aims at building a partnership with its customers, providing solutions to specific needs.

Fst has made large investments in Java technology for Internet development. Our project management process is based on the Rational Unified Process, and we use extensively the Rational tools (Rose, Requisite Pro, SODA etc.). The firm is in the process of obtaining an ISO9001 certification on its development process. Its quality system comprises 19 procedures and guidelines, plus over 30 templates for managing the process of software development, so it may be considered quite lean by ISO9001 standards.

In the past two years we have studied extensively XP, as an improvement on our current practices. The points that we hoped that XP could address were bloated documentation and software quality. In the process we learnt that XP is a way of planning and managing a project, and a new philosophy about risk management.

The rest of this paper briefly reports on the lessons we have learned while experimenting with XP, and then discusses the issues raised in favor or against the introduction of XP by people from three areas: customer relationships, project management, personnel management. Finally we report on the work we are doing to incorporate XP within our quality system.

A final note: this paper assumes that the reader is knowledgeable about XP. Key concepts will be used without any explanation. [1] and [2] are the key references to the XP terms introduced below.

OUR EXPERIENCE WITH XP

Fst research laboratory has been using XP since February 2001, completing two pilot projects. In the first project we developed a set of cryptographic components in C++. We used a pared down XP process leaving out the planning game and coding standards.

The results confirmed the 20-80 rule proposed by Beck, namely that if you follow 80% of the process you get just 20% of the benefits. The problem was especially the lack of control and coordination of the overall process.

Therefore in the second project we decided to apply XP by the book, with 2-week iterations and half hour tracking unit. We tried to release something meaningful every two iterations. The project objective was to develop a demo of an e-procurement portal supporting digital signatures and time stamp functionality according to Italian Law. As a necessary complement we also developed a time stamp server compliant with RFC3161 and a few clients for requesting and managing time stamps. The goal was to demonstrate the use of our legally compliant crypto components in a real world scenario that our customers could understand.

Both projects lasted approx. 3 months and employed 6/7 people. The customer role was played by the director of the R&D lab. The experience of the team was low, with 5 people having less than 2 years of programming experience.

Currently we have about 15 people working using XP, half of them for external customers, and there is a lot of interest in the firm for this new methodology. Our experience taught us a few lessons, which we are passing on:

You need to be cautious when tailoring XP. It is all right to use some of the XP practices inside another process. Chances are that these practices are already present in your process, and you only have to stress them. In the first pilot, the group was eager to pick up those practices which were in line with their beliefs on software devel-

opment. This is perfecly ok, provided that the aim is just to improve on the traditional methodologies, not to adopt XP.

Our experience confirms that it is the synergic interplay between all practices that allows the XP practitioner to give up key activities essential in established methodologies such as upfront requirement analysis and planning and design exercises. In the first pilot we used traditional planning, (GANTT charts, RUP iterations), but no written architecture or design specifications, and the project ended up in a middle ground where we did not benefit from either XP or the traditional methodology.

We found it difficult to develop and maintain an overall architecture for the project. Kent Beck in [1] says that the "metaphor" should play much of the role played by architecture in traditional methodologies. We could not make this metaphor concept work, maybe because we did not fully understand it. We used a metaphor for the whole system to discuss and decide which features should be implemented or added.

After a few iterations we found that we lacked a clear overall vision of the system. This could very well derive from the relative inexperience of the team members. So we decided to have posters around with sketches of the architecture or checklists of important points to remember. These posters were drawn in design sessions that usually were held at the end of each iteration, as a preparation for the next planning session. We found that it is better to keep iteration planning sessions focused on planning and hold separate design sessions. We also scheduled in advance the design sessions because inexperienced programmers are likely to be late in realizing that some design is called for. We do not know if this is a sensible thing to do, but there seems to be no clear recipe in XP for developing an architecture (see [5]).

Refactoring is hard. *Refactoring seems to rely on an aesthetic awareness which can only be developed with time. The group must have enough senior programmers so that most of the time each pair has the experience to do the necessary refactoring. Moreover, the team must share a basic agreement about software "quality". Inexperience often implies the lack of this common culture, and this translates in a lot of open-ended discussions of limited worth.*

The systematic and synergic nature of the XP process facilitates the adoption of each practice. *Most programmers in the pilot projects, and some of the most experienced ones, had some trouble accepting some XP practices like simple design, coding standards or test-and-code. When adopting XP by the book, the focus of the group shifts to implementing the process as a whole, and everyone is more inclined to be a little less critic on aspects of the process which they find objectionable. In particular, pair programming is a very powerful way to discipline programmers by averaging individual idiosyncrasies.*

Programmers reported an enhanced awareness of the state of their activities with respect to the agreed plan of the project. *This may seem obvious since XP is all about maximizing communication. We just want to report that people with RUP experiences said that the planning game gave them a stronger feeling of being in control than traditional planning: each programmer knew where the project was going, if there was any delay, how good was the code and this improved his/her motivation.*

You need to actively foster involvement in the planning game. *In our experience people new to XP assume a rather passive attitude during iteration planning sessions. Task cards and user cards do not travel around, but are traded between the two or three more senior programmers, who write most of the stuff and are too ready to advance estimates and suggest possible solutions. Other members of the team do not actively sign on tasks, but often limit themselves to agreeing on proposed estimates and accepting a task. Sometimes the situation is not so clear cut as described above, so the coach must make sure that everybody feels that he/she is part of the decision process, and is really the owner of the task description and estimates.*

INTRODUCING XP IN THE FIRM

In order to evaluate the feasibility of adopting XP for most of our software development projects we conducted a series of meetings between XP practitioners and interviews with people from marketing, project management, software engineer and quality assurance.

The interviews were conducted informally and involved 9 people, each with some prior knowledge about XP, although none of them participated in a XP project. The interviewers had a good prior knowledge of the people interviewed and of their problems and were able to establish good communication.

Each interview was organized in three sections: first we asked about the main problems facing the inteviewees. Then the interviewers recalled or explained those XP practices that could be relevant to the issues raised. Finally, the objections and observations of interviewees were noted and discussed. The actual questions made depended on the role of the interviewee in the firm.

It is apparent that the main objections to using XP do not come from persons actively involved in the software building process, but from marketing people and personnel managers. In the following we discuss the issues raised in the interviews. These issues reflect without any doubts the particular situations of an Italian firm based in Sardinia, and may not be representative of any other reality.

Customers are not ready to accept the assumption of the unpredictability of requirements. *The people that deal directly with the customers and whose objective is to have contracts signed, feel very strongly that they would have a hard time proposing contracts without a formal sign off of requirements, fixed time and cost provisions, and penalties in case some of the terms of the contract are not honored by the supplier.*

As known (cfr. [2]), XP requires the constant guide of the customer or an empowered proxy to lead software development. The customer does this by working very closely with the development team, selecting priorities, clarifying and redefining the project scope, or, as a last resort, by extending deadlines. On the contrary, if the customer requires the definition of the scope, cost and time frame of the project at the very beginning, then he/she is implicitly rejecting the adaptive nature of the XP process.

Our customers do not like contractual provisions that address the risk of changing requirements. Most of our customers regard software as just one of the goods that they acquire, and want to buy it using the same kind of contractual agreements used for buying other products.

On the contrary, XP, and the agile methodologies in general, claim that the process of software development is inherently unpredictable (see [1], [2], [3]), mainly because of the unpredictability of requirements. However, we believe that the case proposed in support of this view is not very strong. An agreement on this point can be reached among people that basically agree on the nature of software development. Customers with little or no experience with software and software projects will not be moved by sentences like "*in software development requirement changes are the norm"* [3], especially if they have just allocated considerable resources to requirement analysis.

More fundamentally however, the problem seen by our marketing people is that a lot of our customers do not seem to care a lot about risk management, which obviously is one of XP main driving points and advantages. Maybe this has to do with the peculiarities of our target market: our customers are large corporations or public administrations and Fst is often a subcontractor or a (smallish) partner in a group of large firms that won a bid. The persons in charge of the project whom we speak to are rarely the *owners* of the requirements. Often they have to play a difficult political part to gather requirements in face of unclear legislation and competing requests by their superiors. Moreover our customers often lack an emotional involvement in the project and their desire to reach the project goals at optimal costs is overshadowed by a constant preoccupation not to make mistakes which could be blamed directly on them. If this is coupled with a incomplete grasp of the technological issues, it is understandable that our customers adopt a very conservative approach to software engineering: they are not eager to assign priorities and do not like speaking about "plan B's" and adjustments to the project. Instead, they like a lot the kind of written "promises" that K. Beck criticizes in [1]: requirement sign-offs, detailed GANTT charts, contracts that heap all the risk on the supplier, who must deliver the "complete" system at a hard deadline for a fixed cost.

Lately the similarity between the values at the basis of XP and the values that led to the development of such well established disciplines as Supply Chain Management or Total Quality Management has been noticed by various writers (cfr. [3], [6]). We found their ideas really useful to discuss lightweight methodologies with non-programmers. These similarities provide an escape route from discussions based on that kind of examples from other fields of engineering (mainly civil engineering) that so many non-technical people seem so fond of. Moreover, these ideas help to lend credibility to people-centric and agile processes. We think that unpredictability of software requirements is a much harder concept to sell, not least because it can be seen as an excuse for sloppiness, lack of vision or plain whining.

XP gives too much visibility on the inner workings of a firm. *The fear of our business people is that by opening up the development process to our customers, we will end up having customers going around shopping for specific people, trying to build the team of their choice for their projects. This is happening to a certain extent even now in our firm. Moreover, it is not totally unheard of that a firm oversells its ability in some specific areas, taking some risks but confiding that its programmers will be able to catch up or that it will be possible to find timely, competent third party help. The customer on-site would make this more difficult or more risky.*

XP is a process that allows growing and maintaining the core abilities of programmers. *The problem is that in our software process there is a strong distinction between programmers and analysts. This distinction has historical roots and Fst shares it with a lot of software firms, at least in Italy and maybe in continental Europe. There is a monotone increase in prestige, salary and visibility going from programmer, to analyst and then to manager. Only programmers actually program. The analyst, (who is usually a team manager), spends all of his time coordinating, planning, coaching, designing, supervising, reviewing and interfacing with other areas of the firm. These activities make the figure of the analyst one of the busiest in the firm. The problem with this organizational model is that it is not sustainable in the long run.*

In fact there is a truth accepted by most programmers and by many non-programmers: you cannot improve your competences in software engineering by just thinking, speaking or reading. You have to compile something and make the thing run. And yet, by isolating the analyst from the menial task of programming, we are guaranteeing the rapid obsolescence of his technical competence that is the raison d'etre of his position. At present our firm is mainly using server side Java programming on application servers on Unix platforms. Suppose that in a couple of years a significant part of our customers start requiring .NET on WIN64 servers. How is the present crop of analysts going to adapt to change? In an interview a manager explained to us some ideas for flattening the team hierarchy and involving the analyst more closely in the workings of the team he or she is managing. It turns out that all of his ideas are perfectly compatible with XP, which addresses these issues in a much more systematic and thought-out manner, by means of concepts like pair programming, collective code ownership and planning game.

XP can make a big improvement on a ISO 9001 quality system. *XP gives very precise guidelines on project management. In fact we think that XP is as much about man-*

agement as it is about software engineering. For example, XP is very specific about roles and responsibilities in customer/developer relationship. Similarly for planning: XP describes in detail the characteristics of the required input artifacts to the planning game (tracking data, prioritized user stories, exploratory prototypes), suggests procedures to conduct the planning sessions and a practical method to check progress. Moreover XP is agile: it concentrates on a few key aspects of the process and leaves all the freedom to customize the rest, and so it can be adapted to different needs of formal validation, configuration management etc, which is a big plus when one wants to adapt a true XP process to a ISO 9001 framework. We think that this is a step forward from other software development methodologies, which are very clear about the artifacts that must be produced, provide templates for all sorts of documents, but are often unsatisfactory when it comes to explain the steps required to produce the documentation and the quality characteristics of the artifacts.

CONCLUSIONS

We are enthusiastic about XP, and it is difficult for us to imagine a software project where we should not try to use XP, at least in the domain of Internet development. XP is both a methodology and a novel approach to software development. We found that it is not a good idea to "customize" the core practices of XP. However, nothing in XP prevents the practitioner from integrating XP with everything that can be useful for the project: verification teams, design documentation, configuration management etc. In a sense XP requires these add-ons because it is so focused on the core practices that leaves a lot of necessary processes and tools out of the spotlight.

Our experience shows that XP is no magic either. If the team lacks the necessary programming experience results will be at best marginally better than what one would expect from any other methodology. This for two reasons: 1) the inexperienced team will be poor at applying the XP process; 2) XP relies on the team checking itself constantly for simple design, code quality and rate of progress, and this requires experience. However, we got to the conclusion that XP is a good choice even in this situation, mainly because XP is a robust and flexible methodology. In a project currently under way, staffed with people with 1 year of XP practice and 2 years of programming experience, the coach has to travel a lot and is available only for iteration planning, but the team is still functioning effectively, producing at the expected rate. If we had adopted the normal RUP-based life cycle, the project would have been stalled by the inability of the lead to work on the start-up documents (vision, architecture etc.), and would have degenerated in an unstructured effort. On the contrary, even without supervision and coaching, the XP programmers tend to stick to the rules, which are easy to follow, as they do not require anything that does not have an immediate, perceptible value for the programmers.

As discussed above, we found that the most problematic feature of the XP methodology is the requirements on the on-site customer. Frankly, we would never worry about a project where the customer is in charge, knowledgeable, flexible, available and risk conscious as required by XP. Give just some competence to the developing team and you will have a project bound to success, regardless of the methodology. At least this is true in our market, where projects are quite simple, ranging from a few weeks to 6 months elapses and up to an effort of 15 man-year. As it is, we are often practicing a split process, managing the development process with XP, but with limited direct interaction with the customer. The coach acts as a customer proxy to the team, relying requirements, priorities and deadline at the best of its knowledge, and interacting with the customer in usual ways, through documents and plans. At the same time we are making it clear that we are using XP internally and we are trying to educate our partners, but it looks like a long way to go.

REFERENCES

1. Beck, K. Extreme Programming Explained: Embrace Change, *Addison-Wesley*, 2000; ISBN 0201616416.
2. Beck, K. Fowler, M. Planning Extreme Programming, *Addison-Wesley*, 2001; ISBN 0201710919.
3. Fowler, M. The New Methodology; http://martinfowler.com/articles/newMethodology.html.
4. Fowler, M. Is Design Dead?; http://martinfowler.com/articles/designDead.html.
5. ISO/IEC 12207:1995: Information tecnology – Software life cycle process.
6. EN ISO 9000-3:1997, Quality Management Guidelines for Sofware Development
7. Poppendieck, M. Lean Programming, *Software Development Magazine ,* May-June 2001 (http://www.poppendieck.com/lean.htm)