

Certifying for CMM Level 2 and ISO9001 with XP@Scrum

Christ Vriens

Philips Research - Software Engineering Services (SES)

christ.vriens@philips.com

Abstract

This experience report describes the followed road of getting certified for both CMM Level 2 and ISO9001:2000 on a time scale of 2 years by using agile methodologies. We discuss why we selected the combination of eXtreme Programming (XP) and Scrum as the base for our software development process and which “ceremony” we had to add in order to satisfy the requirements of CMM L2 and ISO9001. Also our major challenges at the moment are described, and the way we try to solve them. Furthermore we want to share a number of issues and experiences.

1. Introduction

The department Software Engineering Services (SES) is part of the Philips Research organisation in The Netherlands. It consists currently of 26 people and is growing with about 10 persons per year. SES carries out software projects in research domains like ambient intelligence, storage, networks, copy protection and robotics. The deliverables of these projects are both prototypes to be demonstrated in exhibitions throughout the world, and products based on current Research results which are made ready to commercialise them. These projects are characterized by:

- Time of delivery is fixed, e.g. the starting date of an exhibition
- Vague and minimum requirements which also change often during development
- Most often there does not exist a customer for the product yet
- Technical challenges
- Large percentage of contractors which requires special attention to prevent loss of problem- and solution domain knowledge after project closure
- Short available time frame of 2-6 months

On the base of these characteristics, we identified the following critical success factors for our projects in order of decreasing priority:

- Delivery on time; we simply can't miss the start of the exhibition
- Quality; as agreed, and if possible quantified, with the customer at the start of the project
- And scope of functionality

The number of team members mainly determines costs. Project duration is always fixed, which in principal means that scope is the only variable we can increase or decrease during development.

In order to clearly position our department as a professional software development organisation and to guide our quality activities (keep them focussed and progressing), we decided to try to qualify for both ISO9001:2000 and CMM Level 2. We identified the following starting points for maximizing our critical success factors and reaching our ISO and CMM goals:

- Minimum bureaucracy (quantified by thinness of the resulting quality manual) in order to create - as much as possible - room for creativity; programming must be fun!
- No “reinvention of the wheel”; usage of practices that have proved themselves under comparable circumstances
- State of the art software development process, accompanied by ditto tooling, in order to transfer it to other Philips organisations in a later stage

2. eXtreme Programming

After studying a number of methodologies (XP, RUP, Octopus, DSDM, FDD) we decided to take XP as our point of departure. As in many other organisations XP had already entered our organisation bottom up, compared to the introduction of CMM and ISO that is most often top down as demanded by management. A number of the 12 key practices of XP were recognized by our engineers as highly familiar already. These include simple design, continuous integration, on-site customer, small releases, coding standards, 40-hour week. While others were still on varying levels of unclearness, like system metaphor, pair programming, test first design. By following XP's three levels of maturity [1]:

- Do everything as written
- After having done that, experiment with variations in the rules
- Eventually, don't care if you are doing XP or not (we have not yet reached this level)

we studied the then available books [2], [3], [4] and started using XP in daily practice. The resulting satisfaction of both programmers and customers were high, although the latter did not know we were doing XP at first.

From this period we would like to share the following experiences with you:

- Although we understood the concept and examples of the system metaphor, we could not come up with good metaphors for our current projects. In this sense we certainly appreciated the keynote speech of Kent Beck at OOPSLA 2002, titled "The Methaphor Methaphor" or unofficially "The last time that Kent was going to explain the metaphor practice"
- As expected, our customers rejected pair programming at first, because of the belief that costs would increase by a factor of two. After getting used to it and seeing the benefits in the form of higher defect prevention/removal and decreased cycle time [9], customers now have no problem with it, although the discussion pops up again with every new customer
- We were uncertain about when to apply pair programming and when it was acceptable to program solo. We currently use the following rules of thumb:
 - In principal all production code - including related tests - shall be written in pairs
 - Each new task shall start with a small design session. If the conclusion of this session is that the required code is relatively simple or routine, it is allowed to program solo
 - Problem reports are always resolved in a pair as in most cases the problems occur on the interfaces of two or more modules
 - Documentation may be written by one person and it will always be reviewed using a Fagan inspection
- Pair programming is a high-energy consuming activity. We try to limit it to at most 6 hours a day. But it gives a confident feeling at the end of the day, regarding the fact that peers have approved the code. Pair programming is also highly appreciated as a way to prevent Repetitive Strain Injury
- Both customers and programmers appreciate working iteratively making small increments. The customer values the early delivery of business value, and the programmers value the learning by experience. Both sense the decrease of stress and increase in confidence as the requested functionality is delivered

incrementally, in the order defined by the customer and in collaboration with the programmers

- We experienced that following the XP practices required a lot of discipline and we valued the pressure of our colleagues to stay on the "agile" track
- Daily stand-up meetings (we only use the name as we always remain seated) gradually replaced the normal status reporting meetings, which were considered as taking too long and visited by too many people who were in principal outside the direct scope of the project. It also removed the "policeman" feeling of the project leader, since he gets all the info he needs each morning and no longer "has to do his round" in order to get that info. The info he gets includes, what is accomplished yesterday, what will be done today, and what are the issues the project leader has to work on quickly
- We liked the direct way in which XP expressed the responsibilities of programmers and customer: the customer decides what is going to be delivered and in what order after hearing the feedback of the programmers and facing and accepting the consequences, and the programmers decide on how this is being developed, although we definitely appreciate the input from some of our customers here

3. XP@Scrum

Issues regarding using eXtreme Programming.

Although we are in general highly satisfied applying XP, we struggled with the following:

1. XP does not give you much help regarding documentation, modeling, and the use of UML and design patterns. For these issues, the content of the books [6] and [1] perfectly complements XP
2. Newly hired software engineers had to be instructed where the architecture is addressed in XP. The term is only briefly mentioned in the 12 practices. Our current belief is that a good introductory course regarding XP is required for each new employee. They must be instructed that writing tests first has everything to do with architecture and design: it forces to look at the code from the viewpoint of it's user which is at a higher abstraction level. And that it offers a practical way to obtain the principle of low coupling and high cohesion. This is because tests must be fully self-contained. Furthermore refactoring, as explained by Martin Fowler [7], allows to not only address architecture at the beginning of the project (as in the waterfall model) but offers the opportunity to evolve the software architecture in a cost-effective sense. Certainly, the principle design-as-you-go compared to code-and-fix must be explained from a practical context

3. XP did not help us regarding introduction of XP in an organisation, how to optimise or enhance the way of working, and how to interact with your management. Also as with most managers, the word extreme scared them off. Maybe we may propose the word excellent here. We are aware of course, that XP would not be what it is now, if XP would not have such a provocative name
4. We could not convince all our customers to provide us with unambiguous requirements in the form of acceptance tests. The best we could get from them were scenario's to execute, with some related sentences regarding the expected outcome. Defining automated tests for all requirements at the user level in the multimedia domain though, is a challenge on it's own. We now schedule meetings at the end of the iteration in which the team demonstrates the currently implemented functionality to the customer and discusses together whether it is accepted or not
5. An iteration length of two weeks worked out to be too short for our small teams (1-5 in size) in order to be able to add significant functionality. We are now using iteration lengths of one calendar month, which also introduces a nice rhythm in our projects: at the beginning of the month new requirements have to be identified and at the end of the month these requirements have to be accepted. This is easily remembered even without inspecting our calendars
6. We felt the need to not only enter functional requirements in the list of user stories but also non-functional ones, like moving from one tool to another, a major refactoring or a particular required document. After all, it is the customer that pays for them and we want them to become visible and approved. We now also add problem reports and change requests to this list

And then came Scrum. After a period of one year of only working with XP, a book about Scrum [8] came to our attention. We immediately recognized the issues numbered 3 through 6, and the solutions offered by Scrum were comparable with the ones we introduced ourselves. We decided to merge Scrum and XP, which went very smoothly, because XP focuses on engineering practices, and Scrum on managerial and organisational aspects. They overlap with the XP Planning Game, called Sprint Planning Meeting in Scrum, which is not a surprise as Kent Beck has reused elements from Scrum in XP. Later we became aware that the combination of XP and Scrum is used that often, that it was even given a name, namely XP@Scrum. At this moment we not only use the Sprint Review Meeting as the acceptance meeting with the customer, but we use the reflections, which are part of it, as a first step towards a learning organization and a way to communicate best practices between the various programmer teams.

4. ISO 9001 and CMM L2

In parallel to the introduction of XP and Scrum in our organization, we addressed certification for ISO9001 and Capability Maturity Model (CMM) Level 2. Please note that certification was not, and shall never be, a goal on itself! We use it in order to measure the progress of our quality activities using so-called guided self-assessments, called Interim Maturity Evaluations (IME) in CMM, and to be able to benchmark our software development process with other organisations.

What surprised us when comparing ISO9001 and CMM was that ISO (certainly in the new 2000 version) explicitly addresses customer satisfaction, while CMM assumes that adhering to the requirements of the various key process areas implicitly leads to higher customer satisfaction.

ISO9001 has a higher abstraction level and a broader scope (hardware, software, processing materials, services, ...) compared to CMM, which focuses strictly on software. This meant that we had to discuss how to interpret certain ISO requirements for our organisation. To assist us in this discussion and to make clear what degree of sophistication is required to satisfy a CMM and ISO auditor, we hired a consultant for each of the two quality management systems. The most important selection criterion for these consultants was their willingness to interpret the CMM and ISO requirements from an agile point of view. This freedom in interpretation exists, as only is described WHAT organisations should do and not HOW they should do it!

CMM. As CMM focuses primarily on large projects and large organizations a translation had to be made to our own particular environment, for which we used just plain common sense. We started by defining one procedure for each applicable level 2 key process area (Requirements Management, Project Management, Project Tracking and Oversight, Configuration Management, and Quality Assurance) with at most two pages. Only after explicit demand from our consultant and often-heavy debate, a certain procedure was extended beyond these two pages.

These procedures contain instructions or guidance on what and how it should be done. Now still most procedures are two pages and the largest is four pages.

Only for the configuration management and quality assurance procedures, there was no direct and adequate support (in CMM and ISO terms) in XP and Scrum. Adding a quality officer in order to provide management with appropriate visibility seemed to be even perpendicular to our agile viewpoint. We will address this topic further in forthcoming sections.

Besides these few procedures, a number of small templates and checklists were defined, in order to increase the uniformity of the various project deliverables and the effectiveness of our way of working. Defining is one thing though, deploying another. We will describe some

challenges in the next section, which are still not solved, at the moment of writing this paper.

ISO9001. With respect to ISO9001 we defined only one process description in a pre-defined format, which described on an abstract level our used combination of XP and Scrum. Where more detail was required, we referred to the books [2] and [8]. Furthermore, we included a number of references to the already defined procedures, templates, and checklists. All other required processes by ISO900, being document handling, internal audits, corrective and preventive handling, control of non-compliances and quality registrations, were re-used from the quality system of our overall Research organisation.

Metrics. Adhering to ISO9001 and CMM L2 also requires introduction of metrics, as a basis for better control. Currently we gather the following:

- Every 6 months the satisfaction of both customers and programmers. In general we score between 7 and 8 on a scale of 0-10
- Velocity of the various programming teams for each iteration
- Number of lines of both test and functionality code added in each iteration
- The resulting test coverage
- Every three months, the scores from a guided IME
- Number of major and minor non-compliances per project for every one month iteration

5. Challenges

Senior management involvement. As from one side, the involvement of management decreases as teams become more self-organized by using Scrum practices. More involvement is required though to satisfy the following ISO and CMM requirements:

- Management commitment to the defined software development process
- Setting up and staffing an independent quality assurance group
- Reviewing and approving all commitments and changes in commitments made to external parties
- Resolving non-compliance issues reported from quality assurance checks, that escalate beyond the project's scope
- Reviewing the implementation of the various defined procedures
- Checking the time schedule for the various projects on a periodic basis

We are still discussing with our senior management how to address these requirements in an agile manner.

Quality assurance. Both ISO9001 and CMM require the existence of a, preferably independent, quality assurance group to provide management with appropriate visibility into the process being used. Peer pressure from pair programming can be extraordinarily effective to

achieve a high level of assurance regarding conformance to standards, though it does not necessarily give management visibility into non-conformance issues. We are currently deploying quality assurance checks per project at the end of each iteration, but we still have to define how to escalate non-compliances to management, which could not be solved adequately within the team itself.

Agile training and mentoring in Europe. As our organisation is growing steadily (10 persons per year) and is heavily depending on a large percentage of "volatile" contractors (around 80 %), there is a continuous need for proper training and coaching. This combined with the shortage of skilled instructors in Europe and the high costs in order to fly them in from the States, currently resulted in projects operating at different levels of XP@Scrum maturity.

Proper risk assessment. Currently, only risk identification takes place in our projects, which is in CMM's Project Management. For Project Tracking and Oversight, we are studying how to improve the assessment of risks, with e.g. impact analysis, in relation to the pragmatic approach of XP not to try to predict or control the future. XP states that one can better invest the energy in keeping a system as simple as possible.

6. Experiences

Only hire the best people. Good engineers must be your top priority. No process, either rigorous or agile, can substitute talent and skills. As a side effect our innovative, XP@Scrum based, way of developing software has proven itself as a way to interest highly qualified engineers for our organisation.

Allow room for customisation. Different projects have different process needs because of criticality, problem domain, technology used, size and geographical distribution of the team.

Quality Officer must be a coach instead of a policeman. The best way a quality officer can operate in an agile environment, is by working on a regular base with the project team and coaching them in the defined way of working. Non-compliances must be considered as an opportunity to improve the way of working of a project team. Acting as a policeman by only checking long lists of quality issues in a one-way style, works counter-productive. Currently we use a team leader of another team as the quality officer for a particular team.

Proper use of time boxing. Engineers and customers must be taught that time boxing is not pressing engineers to work in overtime at the end of each iteration, but to force customers to make hard trade off decisions in order to maintain progress: what will be implemented in this iteration and what not.

Reflection workshops. Reflection workshops, as defined by Scrum, at the end of an iteration are an agile

way to increase the maturity and effectiveness of your programmer teams.

Budget. A rule of thumb we often hear is to reserve 15% of your resource capacity for a period of two years in order to raise your organisation from CMM Level 1 to 2. Ultimately, this could probably become 10% in practice, because of various pressures, which are familiar to all project organisations. By re-using existing methodologies, process descriptions, procedures, templates and checklists from other organisations (all within Philips or freely available), and by persisting on an agile way of working we managed to keep the budget spent to around 5%.

On-site customer. We experienced that it was far simpler to arrange adequate space in the proximity of the customer, then convincing the customer to sit with or to visit often the programming team. This resulted in programming teams scattered over our campus with, as a consequence, less sharing of technical knowledge. We organize lunch meetings now, where all members of the various project teams are present, to explicitly facilitate sharing of knowledge via technical lectures.

XP is not for everyone. XP and Scrum are highly disciplined methodologies, emphasizing verbal over written communication. Not every engineer or customer appreciates this way of working and some simply will not adhere to it. You should respect this. Consequence is, that some engineers could leave your organisation and some customers could “better be served” by a different supplier.

7. Conclusion

In May 2002 we certified with our software development process based on XP@Scrum for ISO9001:2000 and a quick scan in December 2002 revealed that we are on a good track for CMM Level 2 certification in 2003. But as expressed in the challenges section, there are still a number of major stumbling blocks to overcome in the meantime though.

8. References

- [1] Alistair Cockburn, *Agile Software Development*, Addison-Wesley, ISBN 0201699699, December 2001
- [2] Kent Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, ISBN 0201616416, October 1999
- [3] Kent Beck and Martin Fowler, *Planning Extreme Programming*, Addison-Wesley, ISBN 0201710919, October 2000
- [4] Ron Jeffries, Ann Anderson, and Chet Hendrickson, *Extreme Programming Installed*, Addison-Wesley, ISBN 0201708426, October 2000
- [5] Mark C. Paulk, *XP from a CMM Perspective*, November/December 2001 issue of IEEE Software

[6] Scott Ambler, *Agile Modeling*, Wiley Computer Publishing, ISBN 0471202827, March 2002

[7] Martin Fowler, *Refactoring – Improving the Design of Existing Code*, Addison-Wesley, ISBN 0201485672, June 1999

[8] Ken Schwaber and Mike Beedle, *Agile Software Development with Scrum*, Prentice Hall, ISBN 0130676349, October 2001

[9] Laurie Williams, *Pair Programming Illuminated*, Addison-Wesley, ISBN 0201745763, June 2002