



Certified Scrum Practitioner Application/Renewal

Applicant Information

Personal Information

Full Name: Minudel Luca
Last First M.I.

E-mail Address: luca.minudel@gmail.com

Current Certification: CSM

Certification Date: 8-9 Jan-2008 City, State and Country of Residence: Stockholm Sweden, Conegliano(TV) ITALY

Is English your primary language? Yes ☐ No ☒

Project-Related Questions

1. By answering the questions below, please **describe one project** during which you have used Scrum in the past year. Answer all that apply or enter NOT APPLICABLE for those that do not apply to your project.

- 1.1. What was the **purpose** of the project? What business goal was the project intended to deliver?

The project purpose was to get the best possible advantage over competitors for a Formula1 team in order win the 2006 world championship.

The goals of the project were

- the adaptation of software applications to the regulation changes introduced by FIA for the 2006 championship (*)
- the maintenance and evolution of software applications used to calculate and decide the best race strategy

(*) Here is the list of the changes relevant for the race strategy:

- limitations to the number of tire sets available to a single race event
- the qualifying session was split into 3 periods (Q1, Q2, and Q3) with the elimination of the slowest drivers at every period
- refueling forbidden after the last qualifying session
- changes to the F1 Timing System used for the Race events
- changes to the F1 Timing System for the official Test events

1.2. What was the **duration** of the project?

The project was time boxed with a time box of 12 months.

The project length and deadlines was determined by the FIA season calendar of F1:

- the project started after the last race of the 2005 championship on Sunday October 16 2005 at the Shanghai International Circuit in China

- the project ended after the last race of the 2006 championship on Sunday 22 October 2006 at the Interlagos Circuit in São Paulo, Brazil.

After 5 months the project had its first deadline (for the adaptation of software applications to the regulation changes for the 2006 season), which was the first race on Sunday 12 March 2006 at Manama in Bahrain.

In the next 7 months there is was deadline for every race in calendar, that is every 1, 2 or 3 weeks.

1.3. What was the **cost** of the project? How did budgeted costs compare to actual costs?

The costs of the project are confidential information, but they included:

- the salary for software engineers

- hardware for the development and the tests

- the use of the applications

- software licenses

- home-factory and on-track support

- travel and relocation to the circuits for tests events and races when necessary.

The project had a fixed budget decided at the beginning of the season (before January 2006).

Since the project was time-boxed and the budget was fixed, only the scope of the project was able to change, to shrink or expand based on the velocity of the team.

So the project was easily completed on budget the actual costs were equals to the 2006 budget.

1.4. Explain the **value** of the project. How did projected benefits compare to actual benefits?

As expected:

- the software applications were successfully adapted to FIA regulation changes in time for the first Grand Prix (e.g. a new application was developed to define a strategy for the qualifying session and to monitor its execution, the software applications was updated to deal with changes in the timing systems)
- the software applications used to calculate and decide the best race strategy were improved (e.g. the race simulation model was improved in order to be able to also simulate a non-linear tyre degradation function)
- the most valuable requirements (based on the team car's performance, the opponent's condition, the current classification and the next race to run) that emerged while the championship proceeded were implemented and released at every race event (e.g the simulation model was improved to also work well with counter-clockwise circuits as for the GP of San Marino, the GP of Turkey and the GP of Brazil)

Furthermore:

- the reliability of the software applications was improved (because our team developed unit tests and acceptance tests that revealed errors and bugs in the strategy calculations)
- the occurrence of human errors was reduced because new software version deploy and rollback procedures were automated
- the velocity of the sw dev team increased because lots of duplications in the source code were removed and because of improvements in the sw engineering practices used by the team

1.5. Describe the **size** of the project. How many people were on each of the project teams? How were they organized into teams?

The people involved in the project were:

- 3 software engineers: 2 coworkers and me
- the Team Principal responsible for the Race Engineers dipartiment
- 2 software engineers responsible for DB Administration
- 3 system administrators working at the home-factory
- 5 system administrators working at the circuities during races and tests events

The 2 coworkers and I composed the team dedicated to software development for the Race Engineers.

The race Team Principal was the main stakeholder for the project and filled the role of Product Owner.

The 2 DB Administrators were part of other sw dev teams. They dedicated 1 day of the Sprint to DBA activities. There were 2 DBAs in order to have always at least 1 DBA available.

The 3 home-factory system administrators were organized in one team.

The 5 race and test system administrators were organized in one team.

In addition to the sw dev team I was part of, there were other 5 sw dev teams in the IT department. The DBA and the system administrators supported all these teams.

- 1.6. Describe the project's **teams**. Were the teams cross-functional and self-organizing? Were the teams collocated in an open space? Were the teams physically separated within one location, or located in more than one physical location?

The 3 of us of in the team dedicated to software development for the Race Engineers were located together in the same open-space.

All of us had different skills, different backgrounds, and knowledge of different technologies. There were no fixed roles as Architect, Coder, Tester, QA: all of us took part in all the activities from the Sprint planning meeting to the software release. At the end of almost every Sprint we had a Sprint Retrospective meeting where we discussed what worked well and what problems we had during the last Sprint and how to improve.

The teams of system administrators and the DBA were also located in the same open-space with us and with the other 5 sw dev teams of the IT department.

We had the Scrum meetings together with all the sw dev teams of the IT department, and beside our team Sprint Retrospectives periodically we had a meeting with all the 6 sw dev teams to share experiences, what we had learned, our major obstacles, and to discuss actions to improve (e.g. the adoption of a common coding convention or pushing the practice of unit testing or asking for courses).

- 1.7. Describe the project's **initiation**. How was the project initiated? How were the teams trained to use the Scrum framework?

The decision to adopt agile methods was taken before 2006 for all the sw dev teams of the IT department by the dev manager of the IT department that had a sw dev background and an interest in Lean Software Development.

The whole team had the initial training from Francesco Cirillo (XP expert and pioneer and inventor of the Pomodoro Technique), and more training on OO design and patterns and on TDD.

Also the whole IT department attended a course together to improve communication and collaboration between all the teams of the IT department.

To further increase my knowledge about Scrum, I attended a CSM training with Joseph Pelrine.

After that the race Team Principal that wanted to take the role of Product Owner were initially informed about the process, about the planning meetings and about the responsibility of the P.O..

The feature requests and the reported bugs were review and the initial Product Backlog were setup, the first planning meeting followed.

Periodically the dev manager of the IT department met the P.O. to get his feedback and to guide him through the role and responsibilities of the product owner.

1.8. Describe project **reporting**. How did you report progress to management and to customers?

The dev manager of the IT department and the Race Team Principal as Product Owner were reported about the planned Product Backlog, the planned Sprint Backlog and the Release plan.

The Team Principal and all the users that requested a feature or a bug fix included in the Sprint Backlog were notified at the beginning of the Sprint. Also, they were informed as soon as the requested bug fix and features were released in beta, in order to accept or reject them.

The dev manager of the IT department and the Race Team Principal also were alerted (personally and via mail) every time the Sprint got behind the schedule. This was an additional transparency measure so that the dev manager could help remove impediments and the Race Team Principal could decide to shrink the scope or change the priorities in order to maximize the delivered value.

The main reports that the dev manager of the IT department got were the report from the track events:

- At the Tests events the new beta releases are used and final users can accept the new versions or report bugs and any other problems. So, the Test report contained the list of new beta versions delivered, the accepted versions, the rejected versions and new bugs if any, while

- At the Race events, the reports contained the list of all the official versions used and the list of bugs/problems reported during the event if any.

The dev manager of the IT department and the involved users also got a follow-up for every problem/bug reported after an event.

The dev manager of the IT department also received reports about team velocity and the number of new bugs reported. Historical data of released feature/software was available to the dev manager of the IT department and to the Race Team Principal.

- 1.9. How was **change** handled? What difficulties surfaced by using the Scrum framework that had to be resolved? How were these difficulties resolved?

We managed change by putting in place feedback loops, inspections of the collected feedback and the ability to react and adapt our work properly.

- After every Race and Test event, the priorities of the user stories in the Product Backlog could dramatically change, e.g. because of a new user story with very high business value (that is competitive advantage over competitors) or because of show-stopper bugs that would have a serious impact on the next race circuit. After every Race and Test event we looked at the report from every track event (see point 1.8) providing a follow-up with possible actions to address issues and also we managed the changed priority for the next Race and Test event in the Sprint Planning Meeting. In order to do this we had to use very short iterations with a time box of 1 week so we were able to have a new release at every Test and Race event (that is every 1, 2 or 3 weeks). We also found it effective to split user stories into very short user stories (1 to max 5 days of work) in order to reduce the consequences of a failure (a user story not finished in time or not accepted) and to give more options to the P.O. when setting or changing priorities.

- At the end of almost every Sprint, we had a Sprint Retrospective meeting where we inspected the way we work, the unexpected events that we faced during the Sprint, the problems that we had, the new things that we tried, the feedback from the users. We also tried to reach an understanding of how things were working, and to reach an agreement on actions to do to adapt and improve. We used metrics like velocity, number of new bugs, number of unplanned tasks that we had to deal with during the Sprint, number of issues reported in the track event report (see point 1.8) to have a general understanding of how the team was going and why. We also collected some basic code metrics like Total LOC, number of new unit tests written, the code coverage, etc. Some of the metrics were collected automatically and continuously, others manually and only periodically.

- Every morning we had a Daily Scrum meeting, and obstacles and delays if any were promptly reported to give the team the opportunity to react quickly when necessary.

- Every 2 weeks I had a meeting with the dev manager of the IT department and the software dev Responsible to evaluate the trend of the adoption of agile practices by the team I was part of and see what worked that could be applied also to the others dev teams.

Scrum surfaced for us

- the need for us to change and improve our engineering practices and our coding practices and skills (look at point 1.11) because former practices were contributing to increase technical debt and so slowing down team velocity

- that the value of confidentiality that was a top priority for the company was preventing the dev team from better learning and understanding the F1 application domain internals, and so was preventing dev from keeping focused on business value while coding a user story and to contribute to maximize the business value delivered. This became clear to the management too so I was permitted to organize some training with the help of the Race Engineers

- that when a release involved activity for a DBA, a home-factory system administrator or a circuit system administrator then they should be involved from the Sprint Planning meeting on in order to draft and estimate together the solution and to coordinate the beta release (at a Test event) and the official release (at a Race event)

- the company value of competition and the culture of blame was a threat to the value of collaboration and transparency fundamental to making the empirical approach (inspect-adapt) work. The dev manager of the IT department shielded the dev teams, building a safer environment and supporting people to share information, discuss problems and technical decisions together and to reach consensus on team decisions

1.10. Describe **management** of the project. What was the previous role of the ScrumMaster? Who took on the role of Product Owner? To what degree were they successful in fulfilling their roles?

As senior software engineer of the team I had the responsibilities of the Scrum Master for the team I was part of. The main focus was on

- ensuring that the process and the practices were followed

- ensuring that the team was fully functional and productive

- shielding the team from external interferences

- calling the Sprint review and planning meetings

Everyone in the team was very good at cooperating closely across all roles, functions and teams so I had an easy work with this responsibility

The dev manager of the IT department together with the software dev Responsible together had the role of ScrumMaster for all the 6 dev teams:

For the team I was part of, the main focus of the software dev Responsible was

- to invite people to the Daily Scrum meeting and to run the meeting and to invite people to the Sprint Planning meeting

- to monitor the Product Backlog.

For the team I was part of, the main focus of the dev manager of the IT department was

- to shields the team from external VIP interferences

- to facilitate close cooperation across intra-department roles and functions and removes intra-department barriers

Every 2 weeks I had a meeting with the dev manager of the IT department and the software dev Responsible to evaluate the trend of the adoption of agile practices by the team I was part of and see what worked that could be applied also to the others dev teams.

The Product Owner role was filled by the race Team Principal who was the main stakeholder of the project. He

- defined the new features (e.g. evolution and changes to the simulation math model, new graphs, new variables in the simulation,...)

- prioritized the features and bugs in order to maximize the competitive advantage and defined the release content with the dev team (based on estimations)

- accepted/rejected the work result after every release

- after a Test and Race event (that is after a Sprint because the Sprint was time boxed and in sync with the test and race events) time to time he updated/changed priorities

During all 2006 all these roles were fulfilled successfully:

- the goals of the project were met and the Team Principal was happy with the results,
- the team velocity grew while the new bugs decreased and the team was happy and very motivated because the success and the improvement in the adoption of Scrum and the agile XP practices

1.11. Describe **engineering**. What environmental factors or software engineering practices had to be changed?

We had to put in place new engineering practices, improve the existing ones and improve our skills in order to put in place feedback loops, to speed up existing feedback loops so that cause and effect stay close together, and to be able to react and adapt promptly and easily based on the feedback.

REQUIREMENTS ENGINEERING

- We changed the requirements collection and analysis. Since there were many changes and new feature requests but only few of them got implemented (because the quickly changing environment of the F1), and sometime details were unknown (because usages of brand new technology unknown or continuously changing) it was a waste to detail and estimate all the requests. We detailed requirements just enough to implement the user story for a release. And we made a first draft estimate in order to let P.O. define priority and release content (during the "what?" part of the planning meeting), then we did a better estimate in the second half of the Scrum Planning Meeting (the "how?" part) for the user stories in the Sprint. We also put effort in breaking every user story down into as small as possible user stories in order to get as soon as possible feedback from the users and improve understanding of the real requirements.

- The bugs reported by users were missing important information in order to reproduce the bugs. This was a big source of waste especially because the bugs could also be related to specific factors (like the network and hardware set-up in a specific circuit, like the external timing and telemetry systems or a specific event like the driver passing the new Lap trigger or a specific server/net workload condition) so we cooperated with the system administrators working at the circuit to collect useful information about a bug as soon as reported by the user during a Test or a Race event.

CONFIGURATION MANAGEMENT

- We started creating an automatic build on the Continuous Integration server for the existing software applications for the race strategy in order to verify and have quick feedback at every check-in that the applications builds on a standard machine (not only on the PC of a developer) and that all tests pass

- When refactoring from one project/application code (e.g. a rename of a public method or a responsibility moved in another place) on assembly/dll referenced by multiple projects, the CI server reported the broken builds or tests of the other referring applications that need to be changed too because the refactoring. To make this loop faster we put the applications in the same Solution so that the refactoring tool was able to automatically do all the changes to all the involved applications on the developers PC, and also the developers were able to build and run test of all affected applications immediately on their PC.

- Source code repository and versioning control was already in place. To reduce the need to merge changes from different developers to the same source code file and to reduce interferences when 2 teams were working on different applications but with common references (e.g. telemetry and timing features are common to many different applications) we adopted the practices to check-in as often as possible (from every day to every 2 hours) only unit tested and working potentially shippable code.

- Also we make it easy to react when the P.O. rejected a feature. We implemented automated application versioning (beside the bin/dll file versioning) in order to have a version number easy to reference with users and P.O., and we implemented in the setup-kit of the applications the Rollback to the former official version (the race engineers use the same PC for Test and for Race events so when a beta version was rejected they had to rollback to the former official version). This also worked well during Race events to quickly rollback an official version to the former official because of a show-stopper bug.

SYSTEM MODELING AND DESIGN

- Instead of doing all the design before starting coding, or not doing design, at all we drafted the design of a feature implementation in the second part of the Sprint Planning Meeting (the "how?" part) looking at the source code and using CRC cards. Then we continuously did design driven by the unit tests we wrote and by the code refactoring (and sometime with the CRC cards again)

- To make the code easy to read and understand we did pair programming

- To make the code easy to read and understand we had to improve our skills in continuous refactoring, unit testing, state-based unit testing, interaction based unit testing, mocking, TDD.

VERIFICATION AND VALIDATION

- The software applications' legacy code-base made it hard and error-prone to understand and change and add new features. In order to refactor the code without breaking existing functionality, we wrote acceptance/integration tests for a feature before refactoring related code, and we also wrote unit tests on the new and changed code.

- Since not all features in the legacy code base were covered by tests, and the setup kit had to be verified too (e.g. to check that when an external dependency is added to a dll, that the dependency is added to the setup-kit too) so we added a check-list of smoke tests to be done before every release on a clean PC/Server

- To catch bugs and defects as soon as possible we did pair programming

- To catch bugs as soon as possible we had to improve our skills on continuous refactoring, unit testing, state-based unit testing, interaction based unit testing, mocking, TDD.

SYSTEM OPERATION

- One goal was to ensure our software applications were able to run on all different circuits, where a complex computer network must be set up with computers and software and software updates and telecommunications and connections to external systems in few hours in a unique and different environment every time. To do this, we set up a feedback loop made of an application monitoring the servers apps and alerting the system administrators for problems and configuration errors. We also added detailed applicative logs to all the applications. This information helped us to diagnose the faults reported from the circuit; it also helped us to make more visible and to monitor the interactions between complex systems (e.g. hw, sw, net, tc) and the chain of events and the dynamics that caused a failure.

OTHERS

- Since there was no more code ownership, but a shared code base with collective code ownership, we put in place coding and naming standards agreed on by all the 6 dev teams.

- 1.12.Explain **stabilization**. For how long did the software have to be stabilized before it could be released? How did you structure this stabilization process?

F1 is a fast business, as are F1 cars: new features are released for every Race event, on the average every week and a half. So for all software applications except for the critical ones (the ones that can cause injuries to the drivers following a different stabilization process), in order to make stabilization as quick as possible a great effort was put into:

- detecting bugs as soon as possible, during the coding activity
- making it easy to react to show-stopper bugs during a Race with the capability to easily rollback the software application to the former official version.

Since all the teams work in time boxing with the same time-box in sync with the same release date it is easy to always release from the head revision of the source code repository.

The set-up kit is automatically created from the build server only after all unit tests and all acceptance tests passes. Then, every developer manually executes a check-list of smoke tests and the tests on the new features.

After that, the beta version is released to all users during the next Test event, and it is tried for at least 3 full days. After the Test event, when no bugs have been reported in the new version, the P.O. can accept the new version so it becomes Official and it will be used in the next Race event.

- 1.13.Describe the **success** of the project. To what degree was the project successful? To what degree was Scrum instrumental in the project's success?

The purposes of the project had been reached:

- the adaptation of software applications to the regulation changes introduced by FIA for the 2006 championship had been released in time for the first Grand Prix of the championship and ran smoothly through the whole race
- software applications used to calculate and decide the best race strategy were successfully maintained and evolved throughout the 2006 championship:
 - the software applications used to calculate and decide the best race strategy were improved (e.g. the race simulation model was improved in order to be able to also simulate non-linear tire degradation functions)
 - the most valuable requirements (based on the team cars' performance, the opponents' condition, the current classification and the next race to run) that emerged while the championship proceeded were implemented and released at every race event (e.g simulation models were improved to also

work well with counter-clockwise circuits as for the GP of San Marino, the GP of Turkey and the GP of Brazil)

- the reliability of the software applications was improved (because our team wrote unit tests and acceptance tests that revealed errors and bugs in the strategy calculations that then were fixed) and the number of new bugs decreased

- the frequency of human errors occurring was reduced because new software version deploy and rollback procedures were automated and because the smoke tests check-list was implemented

- the velocity of the sw dev team increased because a lot of coupling and duplication in the source code was removed, and because of improvements in the sw engineering practices used by the team

Furthermore:

- the main stakeholder of the project, the race Team Principal, was happy with the results

- the team was happy and very motivated, because of the success and the improvement in the adoption of Scrum and agile XP practices

- the management was happy and I was asked for the 2007 season to work in pair and help software dev Responsible to spread the adoption of Scrum to all 6 dev teams

The skills of the software engineer group I was part of, together with the Scrum process, were instrumental to the project's success.

The Scrum development framework was instrumental for many reasons:

- Because of the very short product life cycle of F1 (a new version every 1-2 weeks), Scrum was instrumental in enabling the team to work and keep the pace with this fast rhythm

- it helped us to reduce the scope instead of the quality, so as to have a steady team velocity at every Sprint

- it surfaced the real needs of engineering practices, only what was needed only as much as needed only where it was really needed, and avoiding the unnecessary work/infrastructure/ceremony

- it promoted transparency and collaborations with other teams and with the users. This saved a lot of effort spent in negotiations and in addressing misunderstandings and working on issues that had no real business value

- Because the chaotic environment of the F1 with frequent changes, unpredictable events, usages of new technology unknown and still under development, and the complexity of the systems involved (e.g. the car, the external systems from FIA, the large computers network of the Team at the circuit that get mounted in few hours every time on a different circuit, etc), the unprecedented use of software applications in new domains, Scrum was instrumental in dealing with this complexity

- it helped us to proceed in small incremental steps and enabled us to react quickly to changes, reducing risks and keeping options open

- it pushed us to setup feedback loops, to collect and inspect information at every incremental step, so it enabled us to learn from experiences and practices and to understand things that wouldn't be understood otherwise

- it pushed the dev team I was part of to become self-confident and responsible, capable of reacting quickly, and capable of cooperating with other teams in order to get all the info and all the capabilities necessary to solve problems

1.14. Describe how the **Scrum** framework was implemented on this project. To what degree was the Scrum framework implemented "out of the box?" To what degree did you have to modify the Scrum framework for this project? For each modification, how did you formulate it so that the basic inspect/adapt mechanisms continued to function? Which parts of Scrum could not be implemented or failed, and why?

The Scrum development framework was implemented "out of the box" with 2 minor changes:

- The Burndown Chart: the Sprint time-box was of 1 week, a Release would normally come after 2 Sprints (and very seldom 1 or 3 Sprints), the size of the team was small (3 software engineers), so the number of user stories for a single release was small. As soon as the team got late, faced an obstacle or anything that could make the release fail, they reacted immediately (e.g. brainstorming about the issue, asking for help, talking to the P.O. to reduce the scope).

We used the Burndown Chart and after 3 releases we noticed that the team reacted even before data was available to update the Burndown Chart, so based on this observation we stopped using it.

- At the Sprint Review and Demo of some Releases, the P.O. was not available because he was at the Circuit, at a Test or Race event, or traveling from one circuit to another (e.g. between the American circuits or the Asian circuits) so we had to find alternative ways to demo the features: when possible we showed the draft of the feature to the P.O. before he left the home-factory. When this was not possible or not enough we got in touch with the P.O. at the circuit (via instant messenger, phone, mail and also via the software application logs transmitted back to the home-factory) when he was

installing and trying the new features. Because the P.O. at the circuit was very busy, this did not always work, so we tried to collaborate with race and test system administrator to help us with this "remote demo", supporting the P.O. on site and giving feedback to us anytime we could not get in touch directly to the P.O. (e.g. when the Car was out in the circuit doing a test lap). And this worked fine.

We evaluated and formulated these modifications during the Sprint Retrospective meeting and also during the periodic (every 2 weeks) meeting with the dev manager of the IT department and the software dev Responsible. We tried the modifications, adopting them once we got satisfied with the result.

General Questions

2. Answer all of the general questions about Scrum below:

2.1. How do you improve the accuracy of Product Backlog estimates? To what degree does their accuracy matter?

The accuracy of the estimate is strongly related to the knowledge of the application domain and the source code base and the technology in general. This is what we've observed during the project.

Is possible to improve estimate accuracy by:

- having devs with knowledge of the app domains and the code base and the technology at the Sprint Planning Meeting or training the devs in the team

- having the P.O. available to clarify doubts and misunderstandings about the required features, and doing spikes in order to clarify doubts on technology and about the code base

- using planning poker (that is a way to apply the "expert judgement" estimation technique) in order to sum the knowledge of all devs doing planning

- having more small user stories instead of fewer bigger user stories makes it more likely that positive and negative estimation errors cancel each others

The accuracy matters to the product owner because P.O. uses the estimates to identify the contents of the Sprint which maximize the business value shippable.

When all the effort estimations for a Sprint have the same proportional error, it does not cause problems to the P.O. trying to maximize the value of the Sprint so it doesn't matter.

When different estimations have very different errors this can cause errors to the P.O. trying to maximize the value.

The iterative and incremental approach and the inspect/adapt mechanism help iteration after iteration to maximize the value delivered more than can be done trying to have estimates without errors.

2.2. How do you ensure that what a team commits to for a sprint is what the team actually delivers?

Team velocity can be used to help the team not to over-commit.

By experience I can tell that it is useful also:

- to keep some slack time in the Sprint used to eventually reduce some old technical debt in the code involved by the User Stories
- to talk with Product Owner and users to reduce scope when the Sprint is late
- to ask other teams with less pressure for help

2.3. Which metrics do you use to track the development process? Which metrics have been changed, removed, or newly implemented as a result of using Scrum?

The measure of number of bugs was already in place.

The measure of the average velocity of the team was introduced with Scrum.

Also, statistics from the report from the track events were put in place

- for the number of issues reported per gravity
- for the number of features accepted and released
- for the number of features rejected

From the logs of the applications, the usages of the applications (number of distinct users for one application per month) and the number of run-time errors (number of run-time errors for one application per month) were monitored.

We also introduced metrics for the code as Total LOC, total number of unit tests, number of builds succeeded, number of builds failed.

Other metrics were periodically inspected manually on the dev PC: coupling, CC, Code Coverage, Code Coverage of Classes with higher Ranking (Ranking calculated with Google rank algorithm based on references)

2.4. What type of training, resources, or tools would best help you successfully implement Scrum in the future?

- a scrum training of the whole team together before starting the project

- training also for the Product Owners

- an ongoing training of coding practices

- an ongoing training of team dynamics with role-games (they bringing actions near primary consequences, together under same the "learning horizon", so the team can learn quickly through direct trial and error)

- consulting on team social dynamics

2.5. Describe the largest impediments you have encountered in a project using Scrum and how you resolved them (or did not resolve them).

A large impediment came from the company culture and values:

- the value of blame over learning from errors and surfacing problems

- the value of competition over collaboration

- the value of confidentiality over transparency

The dev manager of the IT department during 2006 shielded the 6 dev teams from blame and competition, creating a more safe environment. I encouraged the team I was part of to explore, take risks, make errors to learn and use feedback loops to correct and improve in order to avoid fatal mistakes.

Product Owners of the 6 teams noticed that domain knowledge helps a more effective and fast communication with developers and also the management recognized this so during 2007-2008 training about application domain were done.

A large impediment came from missing skills from software engineers:

- lack of knowledge of OOP and OOD by old developers

- lack of knowledge of Pair Programming, TDD with state based unit testing, Refactoring

- lack of knowledge of TDD with state based unit testing and lack of experience with the retrospective and with the "How?" part of the Sprint Planning

Those impediments were mostly removed with courses and doing training on the job.

A large impediment came from some valuable people who found themselves not compatible with some agile practices and values:

- some valuable dev (working on the same shared code base) that preferred a command-and-control leadership and was unable to be part of a self-organizing team

- some valuable dev (working on the same shared code base) that preferred up-front design, manual testing and debugging, cowboy coding and was unable to do emergent design, continuous refactoring, TDD

- one valuable department responsible and his department (working for the F1 team) that preferred fixed plans and were unable to deal with Sprint Backlog and changing priorities

We had tried in different ways to remove these impediments with some drawback (e.g. effort spent because strong disagreement, more technical debt because bad coding practices, misunderstandings and loss of business value) without finding the definitive solution to deal with this impedance

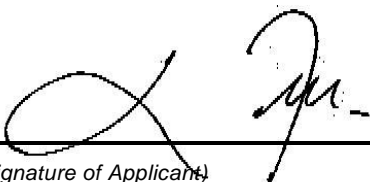
2.6. Describe how you have worked with other ScrumMasters to advance the use of Scrum within an organization or within a community.

- Every 2 weeks I had a meeting with the dev manager of the IT department and the software dev Responsible that filled the role of ScrumMaster for the others 5 dev teams to evaluate the trend of the adoption of agile practices by the team I was part of and see what worked that could be applied also to the others dev teams.

- Once a week team members, especially ones passionate about Agile software development, had a beer sharing ideas about Scrum and XP and talking about how to improve our work, our practices, our experience on Agile methods.
- Once every 2-3 week team members organized a workshop after working hours named "Pillole di conoscenza" (Knowledge Pills) where everyone could share his knowledge of some topic with coworkers and discuss together
- Actually, in the company I work for, every 2 weeks I organize a coding dojo with coding exercises and role-playing
- I'm an active member of the Italian Agile Community since 2003, and I taking part in the annual Italian conference AgileDay dedicated to the Agile Software Development as attendee, as speaker and as open space facilitator
- I'm an active member of the Italian user group named the Agile Software Development XPUG-IT
- I supported and promoted the born of the Italian ALT.NET community UGIALT.NET
- I'm a Blogger and I wrote post about Agile Software Developments: Scrum, XP, people and team dynamics, practices and coding, leadership, complexity, lean software development, experience reports, new ideas, free thoughts

Applicant Statement

With my signature below, I certify that the responses submitted in this application are my own work, ideas, and experiences using the Scrum framework and that work from other sources were appropriately cited.



(Signature of Applicant)

(NOTE: A signature is required. If necessary, print, sign and scan your completed application form for submission.)

Submission Process

- Submit your completed application form via email to: practicingcertification@scrumalliance.org.

Thank you for your interest in renewing/applying for Certified Scrum Practitioner Certification.