Agile Way Process Improvement

Chaudhari, Gajanan

Agile Way Process improvement

Business needs for process improvement projects are changing. Organizations expect faster results from their investments; they want their improvement projects to adapt to and follow changing business needs and be more engrained with the organizational way of working. The agile way of working, used more and more in software development, contains several mechanism that support these business needs. So the question is: Could a process improvement project be performed in an agile way and what would be the benefits?

In this paper I start by looking back to my first software development project. I managed that project in a way that would now be called agile, to be able to meet the needs of my customer and of the organization. Next I'll give a brief description of process improvement, and of agile; just the basics needed to understand how they can be melted into an agile process improvement approach. Then I'll go into the reasons to do it process improvement in an agile way, and the benefits that can be expected from it. I will discuss a distributed process improvement project that has been managed in an agile way, to share the learnings and benefits. Finally I'll describe some "golden rules" that help to improve agile working along the way, and to become even more effective in it.

My first project: Working Iteratively

The first project that was assigned to me was managed iteratively by me. This was not demanded by the line manager he just wanted me to finish the project as soon as possible, and deliver a product; it was up to me how to do it, and to come up with a plan. The product to be developed was software for a CNC milling machine, for milling of a so called pocket. A pocket is defined by a mathematical formula that specifies the outer boundaries (often called contour) of the area that is to be cleared with the CNC milling machine. The product to be developed is a complex mathematical embedded software solution that calculates tool paths. It has to be integrated in the complete system (hardware and software) that is used to program and control a CNC milling machine. The product was developed and maintained by an organization of +/- 60 professionals, organized in product teams that each were responsible for a part of the product. The organization combined development and maintenance work, with projects for developing new functionality, and problem reports for handling maintenance and smaller product updates. Previous projects that had tried to develop the new software for pocket milling failed to deliver a stable working product, or were stopped since they weren't able to deliver. So the customer had lost its trust in the product, and we were aiming to regain it by developing and delivering in iterations. Working iteratively would also help us break down the complex technical problem, and to deal with the technical risks early in the project.

We learned from the previous projects that the biggest difficulty was the huge set of different types of pockets that the software should be able to handle. Different algorithms had been implemented, but none of them was able to solve all the possible pockets. We decided that the purpose of the first iteration was to implement a basic version of the algorithm for calculating tool paths. Purpose of this

delivery was twofold: First to see if we could at least provide a solution for the "easy" pockets, and second to regain trust of the customer and get feedback from him. The software delivered was not running on the target (embedded) platform but on a PC, and it only had a very basic user interface. Also it could only handle a very limited set of possible pockets that could be milled. We demonstrated the software to our customer when he visited the development site, and he took the software back home and used it on his PC to test several pockets. His response was that the software actually worked with most of the basic pockets, but he found some situations where it didn't function correctly. Since he could clearly describe when it failed and how, we were able to reproduces the problems, and solve them within a couple of days.

We had a next iteration where we extended the algorithm, and delivered a second version of the product which supported several new pockets. The delivery also included solutions to the problems that were found on the first version. Again we gave a demo to our customer, where he actually tried some of the pockets that failed in the first version, and saw that they were handled correctly now (talking about an interactive demo, it was!). Again he tried the software on his own PC, and confirmed that it already provided a solution for a lot of the pockets that it was required to solve. In a discussion with the customer we were able to define which pockets should be implemented next, and what would be the priority to deliver solutions for those pockets. We also identified several pockets that were possible in theory, but that would never be used by the end customers of our application, so we did not need to provide solutions for them. Some of those pockets would have been very difficult to solve (and would have cost a lot of money to develop and test), so by clarifying our requirements with our customer we saved a lot of time! We continued to deliver iterations, until the customer decided that the available functionality was sufficient to include in the next release of our product. The project was evaluated and finished, and our customer was pleased to see that the functionality that he had been waiting for years was there. Our project took about a year, less then any of the earlier projects, and it actually delivered value to our customer.

So what where the advantages of working iteratively? First, we regained our customers trust, simply because we managed to deliver. We got lots of useful feedback, from our customer and from fellow development groups. The feedback mainly helped us to clarify our requirements and get a better understanding of what our customer needed, and what he needed first, and what he didn't need at all. It also helped us to improve the quality of the software.

One small thing that I didn't mention yet, the project with iterative development that I am talking about was done in 2000. There weren't that many books at that time on IT Project Management, let alone on iterative development, demonstrations and frequent deliveries to customers (which are now practices of agile methods). There was hardly anything published on starting with a first set of requirements, and clarifying and adapting your requirements during the project, preventing to spend time on requirements that turn out to be wrong or unneeded (which Lean calls waste nowadays).

Iterative methods like RUP and Scrum weren't invented yet; the only iterative method that I was aware of at that time was Evolutionary Delivery. I considered working iteratively as the logical thing to do. Though working iteratively was new to the organization, it made sense to our customer, the management and the team that I worked with, so they allowed me to try it out.

Distributed teams

After having delivered some first iterations of the product that focused on the mathematical algorithm, the need came to develop the user interface for the software. There were some experienced software developers who could assist the project; however they were living at the other side of the Netherlands. We defined a way to collaborative develop the software, while working on 2 locations. First we agreed upon the architecture, and the main interfaces between the algorithm software and the user interface. We also agreed upon some technical things, like the operating system and programming language to be used, the tools used to compile and build the software, and some basic configuration management aspects like file names, version numbers and how the files at the various development sites would be synchronized. Finally we drew up a short list of design and programming rules, mainly because we wanted to be able to read and understand each other's code.

The teams had one fixed day every week when they worked at the same location. At that day, code was integrated, rebuild, and jointly tested. Also there were technical discussions, and members of the team peer reviewed each other's code. The other days the team members worked on two sites, where they occasionally contacted each other when any issues or questions came up. Since they knew each other quite good, such contacts were brief, to the point and very effective.

What was the advantage of working with distributed teams? The biggest benefit was that we were able to have professionals on our team with the right expertise (though working remote most of their time), instead of having professionals that were working locally but didn't have the right knowledge and skills. This saved us a lot of time and money, and increased the quality of the product that we delivered. Having a fixed day where the whole team worked on one location helped us to communicate and synchronize and assure that we had working software every week. Remember this is 1991, practices like continuous integration did not exist, in fact the only connection that we had between the sites was a dial up modem (there was no publicly available internet) which was much to slow to synchronize all source code, and there were no tools to support synchronization. Also our programming rules turned out to be helpful when we peer reviewed each other's code, the discussions where mostly about how the problem was solved in software, and less about layout and coding styles, which made the review very effective.

What did I learn from my first project?

In my first project I clearly saw the benefits of working iteratively. I learned that we were able to deliver quicker to our customer, and deliver only what was needed; not wasting time on things that were not required. Also the early customer feedback helped to deliver the right quality. My team members liked to work this way, and (also important since this was my first project) my manager was happy. This all was confirmation for me that this was an effective and efficient way to develop software. My main

learning of working with distributed teams was that is was actually possible to do it. You need to arrange things to be able to work smoothly on multiple sites, but then you can work together. It is important though that the team members know each other, so initially people have to work on one site, or they need to travel between the two sites to work together and build up a relationship.

As far as I can recall, all of my software development projects have been iterative, and lots of them had distributed or dispersed teams in some way. Even when there was no possibility to do early deliveries to customers, we develop and tested the software in iterations to have early internal feedback and to have working software early and often in the project. This way of working has a lot of similarity with what is nowadays called agile.

So when I started to work as a quality manager, and later as process improvement manager, working iteratively was a logical choice for me. And I was confident to work with distributed or dispersed teams, knowing the benefits it could bring and being aware of the preconditions that make it possible. But before I describe how we applied this iterative and agile approach to process improvement, let me first describe what process improvement is, and give you some insight on the agile principles that were used in our approach to process improvement.

Process Improvement

Given that software development is still a young and immature profession, there is a need to continuously improve the development and management of software. Software Process Improvement (SPI) is one way to arrange this. SPI is based on the assumption that there is a process that describes how the software is developed, and how this development is managed. By improving the underlying process, the quality of the software that is the result of it will improve. It also assumes that the process, which governs how the software development is managed, can be improved to deliver software on time and within budget. Since the principles describes in SPI can and have also been used in projects that deliver hardware, or presentation and documents that can be used to improve an organization, I use the more general term process improvement in this paper.

Many organizations use the Capability Maturity Model – Integration ® (CMMI) [CMMI 2006], which a collection of best practices. The practices are arranged into process areas. Each process area has defined goals that need to be satisfied, and a set of practices that can be implemented to reach the goals. There are two representations of the CMMI model: Staged and Continuous. The staged representation assumes that process improvement is implemented using 5 maturity levels, where each level (except level 1) has a set of process areas. Process areas of higher level assume that the process areas of a lower level are in place; staged implies that the organization is improved in steps from maturity level 1 to 5.

The continuous representation provides an organization to pick those process areas that would bring the biggest benefits, and implement the practices from those process areas to improve capabilities of the organization. This implies however that an organization will have to have some insight in the process areas and the potential benefits, to be able to make a good choice of process areas to implement. The CMMI suggests doing a zero-assessment to determine the current capability level of a process area, and then define the wanted capability level of each process area and implement the changes to bridge the

capability gap. Many organizations starting with process improvement do not have sufficient insight and skills to take this approach. To help them to get started with the continuous representation, an additional approach was developed, called CMMI Roadmaps.

CMMI Roadmaps are a goal driven approach to select process areas from the CMMI model. There are 5 pre-defined roadmaps of CMMI process areas. A roadmap consists of 4 to 6 related process areas. CMMI Roadmaps help organizations to focus upon a specific solution area. The roadmap approach combines the strength of the staged and continuous approach, by helping an organization to get started with CMMI and reap benefits, while selecting those process areas that will most probably give the biggest benefits.

Visibility is crucial for reaching results with process improvement. Process improvement is a learning process, where professionals change their way of working and reflect to see if this leads to improved performance. By making the aims and the results of process improvement visible, professionals can exchange experiences and learn from each other. Visibility enables alignment between process improvement and the business goals, which is vital to gain and keep commitment in the organization. Visibility also shows success, and since success breeds success this helps to increase adoption of improvements. Later in this paper I will go into how the agile approach to process improvement supports visibility, thereby increasing the performance improvement of the organization.

Agile

The term agile was coined by group of people experienced in software development, when they gathered in 2001 to define the manifesto for agile software development [Agile Manifesto]. As stated in the manifesto, they are "uncovering better ways of developing software by doing it and helping others do it".

Agile is not a specific method, but includes a broad set of methods and techniques that subscribe to the values and principles of agile. Well-known agile methods are XP and Scrum. But also a method like Evolutionary Development or EVO is agile, even though it was there long before the term agile was invented.

The agile manifesto contains values, one of them is "to value individuals and interactions over processes and tools". Process improvement as we saw earlier puts much emphasis on the process as the base for developing software. At first sight, agile and process improve might look incompatible, but that is certainly not the case. As the agile manifesto states, it is not that that they do not value the right statement, but they value the left more. So as long as processes help professionals to interact effectively and efficiently, they can certainly be agile.

Agile Process Improvement

Let’s go back to the question if we can do process improvement in an agile way? First I’ll describe the reasons why you want to become more agile, and the benefits that it can deliver. Next I’ll go into detail how agile process improvement was actually done. I’ll finish by looking back if the agile approach for

process improvement has paid off, and will show of "golden rules" that were used to deploy the agile approach and to further improve along the way.

Why Agile Process Improvement?

There are a couple of reasons to become more agile when doing process improvement. From a business point of view it is more and more important to be able to adopt process improvement programs to the changing business needs, and to be able to deliver more value to the business. This included making the value visible, which supports discussions on what value has been brought already, and what value can be expected from process improvement. Another reason to become more agile was to engrain process improvement in the way of working within the organization. Many organizations have or are adopting agile development methods, and use techniques like retrospectives to continuously improve on a team level; it feels natural to adopt similar mechanisms for process improvement programs. Finally, since there are almost always limitations in lead-time and money (available hours of the team and of the organization that need to adopt the changes), you have to work as efficiently as possible. Agile helps the improvement team to deliver quickly, and deliver maximum value with a limited budget. So altogether there are sufficient reasons, from a business point of view to adopt an agile way of working for process improvement.

Additional there are several benefits that an agile approach to process improvement can deliver. Firstly, agile can improve the collaboration between the process improvement team and the stakeholders. Agile iterations enable frequent interactions between those involved in process improvement, involving the customers of process improvement much more into the change process as the "traditional" process improvement approach. The agile approach also helps to focus on deployment of the processes in stead of making large process documentation, based upon the second value of the agile manifesto: "working software over comprehensive documentation". Finally by working agile themselves, the improvement team gets a good understanding of agile which can be used to better support to teams in the organization that are also working agile, or making the transition to agile. Basically it's eating your own dog food, which helps you to get insight into the consequences and the obstacles of changing towards agile, while at the same time getting the benefits of it. The improved collaboration, focus on deployment, and personal experience with agile of the process improvement team leads to a bigger impact in the organization, and therefore to increased organizational performance and business results.

Project: Improving Process Management

The agile approach has been used in two process improvement projects that I presented in organization conference in 2010. One project concerned the improvement of process management activities within a business unit of a multinational company. This project was done with a dispersed team, consisting of operational development professionals from R&D centers around the world. The other project was executed in one of the R&D centers. Purpose of the second project was to do selective and quick process improvements, to continuously improve the performance and results. This paper goes into the first mentioned project, and describes how we have used the agile approach, and what the benefits have been for the project, team members and the organization.

The purpose of the process management project was to align the various development and delivery processes that were used by different R&D centers within a business unit. Expected benefits were that by using the aligned processes, the business unit would be able to develop and release products quicker, thereby being faster to the market with new products with high quality. Cost was also considered important, but by making the flow leaner and remove unnecessary activities, the cost would automatically go down. The product was ordered by the global business unit, and involved several R&D centers spread around the world.

The project used Scrum as method to coordinate and collaborate [Scrum Alliance]. Though Scrum is mostly used by agile software development teams, it looked suitable for our process improvement project. Scrum recognizes 3 roles, the Product Owner, the Scrum Master, and the Team. The manager responsible for processes within the business unit acted as Product Owner, his prime responsibility was to define and prioritize the user stories. The user stories described the services and products that had to be delivered by the Team. Given that this was a different kind of project, the products produced were mostly presentations, slide sets, and documents containing policies and texts for the Intranet or newsletters. Next to products there were also tasks that were done, like assuring that issues were discussed in the appropriate groups and meetings and those decisions were taken and communicated, and activities like arranging meetings and workshops to discuss and review materials that the team had produced.

The project manager took the role of Scrum Master, and adopted the project plan to reflect the agile way of working. A major step was to publish the project information on the wiki instead of a word document, thus facilitating the discussions of project scope and approach using this wiki. Actually our wiki, which started as a project space grew to being one of our main communication tools, planning board, delivery platform and even demo space; we continuously evaluated our use of the wiki and found new ways to use it as a team working space.

The team consisted of professionals involved in operational development at the R&D centers. They were very experienced in process management, but most of them had never worked in an agile team before. Based upon my experience with agile I got the role as agile coach, next to my role as team member in the agile team from an R&D center. In that role I helped my fellow team members on how to fill in the different Scrum roles, by coaching them on activities like the planning game, stand up meetings, and how to use the wiki. I also organized the retrospectives at the end of every iteration, and supported the team members to implement the improvements that were suggested in the retrospective into the iterations. All team members worked part time on the project, where the weekly availability varied from just a couple of hours to some days per week. Occasionally team members were not available, due to local R&D center activities or travel. Since the team was aware that they were depending on each other to deliver quickly, we kept each other up to date on the wiki regarding our availability.

Having all Scrum roles assigned we started to use the Scrum principles to organize our work. As mentioned earlier, a backlog was set up with user stories from our product owner. We started with a limited set of user stories, which were extended whenever new products or services that the team needed to deliver where identified. We had a first planning game at one of the R&D sites, where all

team members gathered with the Scrum master and the product owner to discuss and define a first sprint. We agreed upon the user stories to be included, and how we would do our "stand up" meetings, use the wiki to collaborate, and do the demo for our customers and the retrospective to learn and improve as a team. We decided to work in sprints of 3 weeks. Each sprint started with a planning game, and ended with a demo and retrospective, in which all team members participated. The demo was given to the product owner, and we also invited relevant members for our organizations to attend and give us feedback.

The sprints were done while the team members were working at their own R&D centre, so we worked most of the time as a dispersed team. At several occasions, members of the team met and worked together at one of the R&D sites. For instance, when a team member was visiting another R&D, they arranged to meet and work jointly on tasks. But most of the collaboration was done using the wiki and telephone conferences, which turned out to be a very efficient and effective way of working together.

The wiki was our task board, and the communication tool both within the team, and from the team (including the product owner) towards the organization. The project plan was a wiki page, which included the goal of the project, the way of working (our process), and contact information of all team members and stakeholders. Also the backlog was on the wiki. This created visibility for the improvement project; both in the communication within the team and from the team to the stakeholders. In emails and in newsletter, references were made to relevant pages on the wiki, we saw that many people clicked on the link and read the information on the wiki. This is different from project plans in for instance Word, which many people do not read when sent as an attachment.

For every iteration, a separate wiki page was made. Initially this page contained the availability of the team members for this iteration. Planning games were done by telephone conference, where the Scrum Master continuously updated the wiki during the meeting, based upon the task breakdown and estimates from the team members. The user stories and engineering tasks were added, prioritized, and checked by the team which at the end of planning committed them to deliver the required functionality. The wiki was also our planning board, so when a team member started to work on a task, he marked this task on the wiki and added his/her name to it. Two times a week there was a "stand-up meeting", were all team members dialed in and again used the wiki to discuss what they have done, what they will do next, and if there are any obstacles. Intermediate versions of products were posted on the wiki, and were reviewed by other team members which put their comments on the wiki. The last days before the demo, all materials were gathered from the wiki into a demo presentation. The demo consisted of a phone conference with presentation. Team members and the stakeholders either joined in from their workstation, or (if multiple people were at 1 development site) arranged for a conference room with a beamer and hands-free telephone connection.

It took a couple of iterations to tune our "definition of done". Having a presentation or document made was not enough, so we decided that a task could only be done if the products had been reviewed and commented upon by at least one other team member, and the product was updated based upon the review comments. Of course all documents and presentations have to use the official templates and

comply with the corporate standards, but this was something all team members were already familiar with.

Using the wiki has really given the team a boost. It was not uncommon that a document was updated and posted several times a day, using the comments that team members shared on the wiki. If possible team members responded within hours when a document was offered for review, also documents and presentations were shared so where this was possible team members updated the documents directly. If a team member saw the need to discuss a comment either with 1 or multiple team members, a conference call was setup. Since we also mentioned shortly on the wiki how a comment has been handled, it was also possible for team members to catch up (remember, we all worked part-time on this project), or to trace back how we had come to certain decisions.

Finally since out wiki task board always reflected the latest status of our tasks, there was no need for burn down charts. Working in this very disciplined way felt natural for the team members; it was their solution to interact and collaborate in this dispersed team. It was also a very efficient and effective way of working, enabling the team members to combine the projects tasks with the other activities that they had to do.

Benefits of agile process improvement

Looking back there were several benefits from the agile approach as we used it in our global process management project. The main benefits that we saw were:

- Being able to deliver the right product with high quality, using frequent feedback
- Understanding the strengths & weaknesses of our processes, and the business value
- Alignment and streamlining of processes between several R&D centers
- Efficient ways for professionals to work together in a dispersed team

We were able to produce many documents, newsletter, slides sets and supporting material that the local R&D centers could use to align and improve their development processes. Though we have no hard figures, our impression is that it took us less time than usual to make them. An even bigger benefit was that the quality of the products delivered by the project was very high, and there was a strong commitment for the changes proposed by the project, due to the frequent planning game and demo meeting where we aligned with our customer and the stakeholders. Several R&D centers started to use the material while it was still under development to do their local process improvements. This helped them to start quickly, and come to results faster, and it also helped our process improvement project since we got feedback from the local improvement initiatives and were able to update our products along the way. All together the commitment for the process improvement initiative was high, both at a business unit level and at the local R&D centers.

Working intensively together with the business managers (our product owner and the stakeholders) helped the team to get a better understanding of the strengths and weaknesses of our processes from a business point of view. Together we developed an overall development & delivery flow, which identified the main decision and synchronization points in product development. At a lower level, local processes

were linked into this flow, enabling sites to continue to use and further improve their way of working. An important result was how we managed to combine both traditional and agile development methods in the overall flow. Several R&D centers had migrated to agile; some were considering it; for those development sites it was very important to see how they fitted into the overall flow. Vice versa it helped functions on a business unit level, like product portfolio management and product release to work with the different R&D centers in a similar way regardless of the processes used, thereby saving money and being able to deliver earlier to our customers.

The process discussions also helped the team and the stakeholders to get a better insight into the value of our processes, and how much processes were actually needed? In some cases, processes were phases out since they were no longer to be used in the organization. This could be because there were similar processes from other R&D centers, and we decided to combine the multiple processes into one description which could be used throughout the business unit. In some other cases we found processes that were so detailed that it was almost impossible for people to understand and use them. The proposed solution was to streamline process documentation into two levels. At the high level, process documentation consisted of flows which identified the main decision and synchronization points. Purpose of those processes is to govern development & delivery of products, and to enable different R&D centers and central functions to work together. At the low level we proposed to focus upon training, and develop supporting material like templates, examples and checklists. The use of extensive process or procedure documents was discouraged, since they were often not used and thus brought no value for the organization.

In the retrospectives that we did after every 3 week iteration we also discussed how the team member had experienced the agile way of working. It took some time initially to develop an understanding of the roles, and how to combine the team role with the local responsibilities of the team members. This happened because the team member had two roles in the team. First they represented their local R&D center and acted as a linking pin to local process support teams, and the local management. Second they brought in their experiences and skills with process management. The aim of the team was to enable team members to contribute as much as possible from their experience to the result, increasing the effectiveness of the team as a whole. This resulted in team members picking up different tasks, depending on where they saw the biggest possible contribution. If we saw that higher priority tasks were not picked up, we discussed this in our stand-up, ensuring that they would be done in time before the iteration finished. This task approach could be conflicting with the representation role that team member had, e.g. in situation where certain process documents were to be made by a team member which would not be of use for his own organization. Where there were potential conflicts, we discussed them in our stand-up meeting, and we always managed to come to a solution. That could either be that another team member would pick up the task, or that the team member would make a first version which would then be extended by other team members based upon their experience. Though being dispersed, we managed to work together in pair where this benefited the team. All team members were very positive about the agile approach, which enabled them to contribute to the end result. Also they valued the support of the other team members, which helped them to learn from each other, and to further develop their process and change management skills.

Golden rules

The team developed and used a set of "Golden Rules". These rules helped them to better understand the agile approach, and to work together in a smooth and positive way. These golden rules were formulated based upon principles from the agile manifesto [AgileManifesto], open space technology [OpenSpace], and retrospectives [Retrospectives]. At several moments in the project, we checked if the rules were still valid and refined them to get maximum benefit out of them. Below the set of rules that we used until the end of the project:

- Dare to share–As early as possible and frequently
- The result depends on the team –Not the individual members
- The one who checks out a task is not necessarily the one who has to finish it
- The one's working on a task are the right people
- You may critique anything, but you may never criticize anyone

This simple set of rules was used throughout the project as a reminder on how we wanted to work together, they were our team values. They helped the team members to discuss and deploy the agile approach, in a very practical way. So let me go into the golden rules a little bit more.

Dare to share–As early as possible and frequently: Team members often worked in short chunks of just a couple of hours, where they produced and updated slides and documents, and reviewed the work of other team members. By sharing early we were able to continuously add value to our products, enabling delivery in short iterations.

The result depends on the team –Not the individual members: Team members frequently asked other team members to support them, or to contribute their experience or results from their own R&D centre to the project. This rule helped the team members reminding that we all brought value to the team, at different times and in different ways. Since we were all also representing our local R&D centre, this was an important value which helped the team, and the stakeholders to focus upon the contribution to the business unit result. We weren't competitors but co-workers, and the way we collaborated was beneficial for all.

The one who checks out a task is not necessarily the one who has to finish it: Team members supported each other, and collaborated where possible. It was ok for a team member to contribute just a little bit to a product, and release it for others to work on. If somebody wanted to contribute to a product that was being updated, then (s)he waited until it became available (or checked with the one working on it when it would become available), and then added his/her contribution.

The one's working on a task are the right people: We saw that when team members had the time, and the energy to work on a certain task, then they added real value to the product or service that they were working on. Team members did not wait for others to pick up tasks, but contributed when they had the possibility to do it.

You may critique anything, but you may never criticize anyone: This reminded the team to always focus on the products and services that were developed. Often it was just a matter of wording, how team members expressed their critique, but that didn't make it less important to be aware how they did it. We always assumed that people were doing the best they could, based upon what they knew and were able to do at that point in time.

These golden rules are something that many of the team members have learned from the project and are still using in their current and future work. For them it is a way to collaborate effectively and efficiently in a team.

Agile process improvement and certification

It is of course always possible to do an (class A CMMI) assessment, to get certified, also when you have taking an agile approach to process improvement. Before doing the assessment I would suggest to do some kind of gap assessment (class C or B) to identify where you are not meeting the goals of the CMMI process areas. If it is important enough for your company to be certified, then it should be easy to make a business case to implement any remaining processes or other changes that are needed to be compliant. So the agile approach to process improvement is fully possible if you want to reach a maturity level.

I actually worked with a company that was assessed to be on maturity level 3, but decided after the assessment to let their process improvement be fully driven by business needs in stead of maturity levels. They accepted that chance that they would no longer be fully level 3 compliant. My statement has always been that, if there was a need at any time to be certified again at level 3, an improvement program with a maximum lead time of 6 months should be enough to assure that all the needed processes are in place. This was based upon the capability of that organization to define and deploy processes, which could be used to introduce any process that is needed, in an efficient and quick way.

Outcome of the article.

This blog described how we have applied our experience from developing and delivering a product in iterations (which is nowadays called agile) into process improvement projects. Scrum was used as a method to iteratively deliver products and services for a process improvement project, where a wiki and telephone conferences were the main tools to manage the work and deliver results. The agile approach has shortened the lead time in our project, gave us a better understanding of what our customer needed, and increased the commitment for the changes that were needed in the organization. The team member appreciated this way of working since it helped them to continuously contribute value, and to develop their knowledge and skills. A set of golden rules helped the team to stay aware of the way we collaborated, and work in a disciplined and effective way.