Progressive Integration of Method Components: A Case of Agile IS Development Methods

Rebecca Deneckere, Elena Kornyshova, Adrian Iacovelli

Progressive Integration of Method Components: A Case of Agile IS Development Methods

Rébecca Deneckère
CRI, University Paris 1 - Panthéon Sorbonne
90, rue de Tolbiac, 75013 Paris, France
Rebecca.Deneckere@univ-paris1.fr

Elena Kornyshova
CEDRIC, Conservatoire National des Arts et Métiers
2, rue Conté, 75003, Paris, France
elena.kornyshova@cnam.fr

Adrian Iacovelli
CRI, University Paris 1 - Panthéon Sorbonne
90, rue de Tolbiac, 75013 Paris, France
adrian.iacovelli@gmail.com

***Abstract*—Situational Method Engineering aims at constructing methods adapted to a given situation, either by a construction from a set of predefined method components or by a customization of an existing method using different techniques: configuration, extension, reduction, and so on. However, these techniques are still limited in practice, as considered complicated and heavy to implement. In this paper, we describe a practitioner experience of a gradual integration of different method components (issued from agile methods of software development). In a real case of a development company, we have practiced and observed the progressive introduction of agile method components instead of the construction or customization of methods in one go. We discuss the lessons learned from this experience and define different research perspectives.**

Keywords—Situational Method Engineering; Method Component; Agile Method; Progressive Integration; Experience Report

I. INTRODUCTION

Situational Method Engineering (SME) argues that a method to be used for the development of a system must be aligned with the context of the project because the situation of each project is different and requires a different methodological support. For this purpose, SME promotes the situation-specific method construction on the fly by reusing parts of existing methods generally designed as autonomous components and stored in method repositories. Whereas many different SME approaches exist, their implementation in practice is difficult as companies are slow to adopt these approaches and techniques even though they acknowledge the significance of the role that methods play in their engineering activities.

A way to use SME in a smoother way should be to implement the components progressively, one at a time, and to wait for the users to be accustomed to the first changes before going to another one. We propose in this work the result of an experiment we performed in a company already using some project management features but wishing to improve the software development processes eventually by introducing an agile method of software development. This company didn't want a too rapid change by implementing an agile method right away and it was then proposed to implement the features one by one, on an incremental and progressive way. The goal of this work was to verify if a progressive integration of method components was possible and if it induced a better acceptance by practitioners.

We first studied the method used by the company in order to identify the missing project management features. We then selected a set of method components corresponding to these features. An integration plan was defined on a long term (two years long) to fit the acceptance/reluctance of change in the company. The method engineer responsible for the experiment was part of the company and helped the integration along the way. Good results were obtained as all the method components had been nicely integrated in the original method.

Section 2 states the context of the company used in our experience while section 3 explains the theoretical background of the work. The case experience is explained in Section 4 and Section 5 states the lessons learned from it. Section 6 concludes this experience and proposes future works.

II. ORGANIZATIONAL CONTEXT

The company capitalizes more than 10 years of research and development in Cloud computing and Big Data. It is specialized in the development of complex information systems, with a particular emphasis on healthcare IT, e-health and biomedical research.

The company was working on several projects at the same time and was using a project development method

mostly based on a classical development method. The team had a weekly meeting to discuss the tasks to be done during the current and next weeks. New tasks could be added and the members of the team discussed their feasibility. The team has used a google doc to save and to share the minutes of weekly meetings. A new version of the googledoc was created each week. The tool RedMine[1] was used to manage the project but its usage was limited to the definition of the high-level tasks and the decomposition of these tasks into sub-tasks. The duration of task realization was also recorded in RedMine.

This organization of project management had several problems. The first problem was the bad definition of tasks. The identified tasks were of very high level and not detailed enough. In addition, their formulation was quite informal (no specific formalism used, fuzzy formulations which conducted sometimes to misunderstandings).

The follow up of the project constituted the second problem. The time used for a given task was specified in the googledoc. Thus, the project progress was observed. However, it was done only at the high level of the tasks hierarchy and the data were not always up-to-date.

The third problem was the lack of a specific tool to support the project management activities. In addition to the difficulties to obtain the follow up statistics, the problems of the googledoc were: it was not possible to know why a task runs slowly and is not finished at time; it was not possible to measure the team productiveness.

Another problem concerned the customers' new requirements and feedbacks. Their management was not structured around a specific procedure. No meeting dedicated to the relationships with customers was set up. Each new customer requirement/feedback was treated in real-time without managing the priorities (as a result leading to the delay of other tasks) or put aside for an undefined period. The follow up of feedbacks to the customers was also complicated.

All these problems were related to the lack of specific project management method and/or tool. In addition to these internal problems, it was necessary to have a formal project management approach to have a better image in the face of customers. In fact, the customers are more confident to work with companies using a well-known method/tool and could more easily renew contracts.

The above-mentioned problems implied discussions inside the team about introducing an agile method of software development. However, the team members were reluctant to the adoption of such a method. In fact, the current solution surely was not perfect, but it was functioning and gave results. The team members were against to take a step forward as it required too much effort and time for a weak chance of success. It was then that we intervene and proposed a progressive way to introduce the changes in the method by integrating small method components one by one in the method, until all the desired agile features were integrated.

[1] http://www.redmine.org/

III. THEORETICAL BACKGROUND

This section proposes a rapid overview of the works on agile methods, Situational method engineering and the use of SME for agile methods.

Agile ISD methods. The concept of agile methods appeared at the beginning of the millennium with the launching of the agile manifesto in 2001 [1]. As stated in [2], while the publication of the manifesto did not start the move to agile methods, which had been going on for some time, it did signal industry acceptance of agile philosophy. Several agile methods have been defined and worldly communicated since then, as Lean Software Kanban [3], Extreme Programming [4], Scrum [5], Crystal [6], DSDM [7] among others. A 2013 survey [8] states that 57% of its respondents work in companies where there are 5+ teams practicing agile and 38% have 10+ teams. These figures indicate that the agile momentum has taken off and that its successes are being embraced at the enterprise level.

Situational Method Engineering and Method components. Situational Method Engineering (SME) was introduced in early nineties [9]. This field of researches argues that the method to be used for the development of an information system must be aligned with the situation of the project. As a matter of fact, the engineering situation of each project - its context - is different and requires a different methodological support. For this purpose, SME promotes a construction of situation-specific method by reusing parts of existing methods generally designed as autonomous components and stored in method repositories. Today, many different SME approaches exist (e.g. assembly-based [10]), extension-based [11], configuration-based [12], process tailoring [13], model driven engineering [14] and service-oriented [15] [16].

SME for Agile Development Methods. Agile methods are usually defined as a set of best practices and behaviors. Since the agile manifesto, lots of books and documents got out to explain these best practices – for instance [17] [11] or [18] - but all these documents lack a clear explanation of the processes to apply, on the statement that you have to be agile to do agile. But the fact that there isn't a formal process doesn't mean that the agile developments aren't structured, it is just that there is a transition away from a completely rigid and formalized process. In [19], Bertrand Mayer states "the typical agile book is a succession of alternating general observations and personal anecdotes of project rescues and project failures. These anecdotes are usually entertaining and sometimes enlightening, but a case study is only a case study, and we never know how much we can generalize". In this jungle of tasks, principles, advices and recommendations, new users of agile methods are sometimes lost in all the possibilities that are offered to them, with no clear understanding of the argumentation to choose one alternative over the others. For instance, [19] mentions that "every agile team in the field makes up its own cocktail of agile practices, rejecting the ones that do not fit - until now,

however, each organization and project has had to repeat for itself the process of sorting out the gems from the gravel." This behavior is common to other fields and the use of other kind of methods (development methods, deployment methods, design methods, etc.) and the SME domain can help to solve this issue.

Agile methods have already been studied in several works in the SME domain. [20] showed in a case study that a situational method engineering approach together with an agile software solution framework (ASSF) can be used to create a feasible and usable hybrid software development method. This can be done by combining agile and formal practices and for a specific situation in large software development organizations. In [21], the Method for Method Configuration has been proposed as a method engineering approach to tailor a specific agile method: eXtreme Programming. One of the conclusions was that XP does not initially provide an extensive coverage of different project paths so it was a bit difficult to obtain contributing ideas from the developers. Moreover, this study used only one agile method, which makes it difficult to generalize. [22] [23] provides a basis for the application of assembly-based SME to the development of agile methodologies. A method base is proposed that contains the necessary agile method fragments derived from prominent agile methods, conform to the Software Process Engineering Metamodel (SPEM 2.0), and can be plugged into CAME tools which implement this metamodel, including the Eclipse Process Framework Composer (EPFC).

We have elaborated a set of method components dealing with the high level representation of the agile methods and a detailed view of components corresponding to the launch phase of the following agile methods: Scrum [5], XP [4], DSDM [7] and Crystal Clear (2004). These components are integrated in the method family called Agile Project Launch (APL) method family. The agile method components were firstly described in [16] and then developed in [25].

IV. Case Description

During the period covered by this study, five components of agile methods were integrated progressively. Thus the whole integration process includes five steps (or stages). The agile components identified by the method engineer as corresponding to the method requirements are: Plan the project with Sprint planning meeting, Plan the project with Planning game, Plan the project with Product Backlog, Manage the project with Daily meeting and Manage the project with Burndown chart. The established integration plan is presented at Figure 1.

The predefined order of the integrated components was defined according to two criteria:

- The requirement emergency for a given functionality and
- The acceptance rate by the team members.

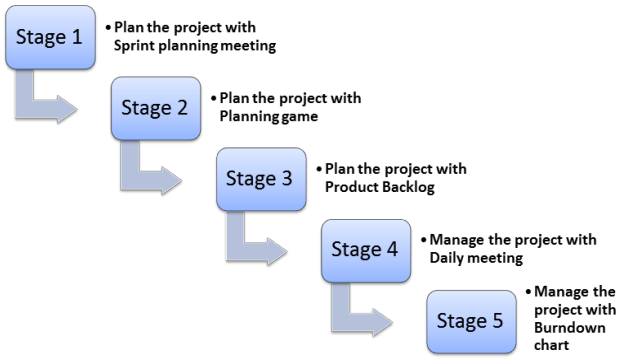

Fig. 1. Plan of the Agile Method Components Integration.

In the next sub-sections, we describe the integration of these five components using the following template: Context, Component, Process, Feedback.

Context: description of the situation in the company and of the motivation for the usage of a given component.

Component: description of the integrated component at the given stage. Each of five agile method components is illustrated on a summary figure.

Integration process: description of how the component was integrated, specifying the particularities of the method component usage and, if relevant, option of the selected component.

Feedback: description of the users feedback on the new features.

Between the five integrated components, four are atomic (representing the most detailed activities of the agile method functionality) and one (Plan project with Product Backlog) is composite as it is composed of three other agile method sub-components.

A. Component 1: Plan the project with Sprint planning meeting

Context: The team was using a google document to plan all the project tasks. However, even if all the project team members shared the document, this process was a bit fastidious to handle as it was filled in a hectic manner after the meetings. The goal here was then to find a tool supporting this task on a better way.

Component: The component 1 (illustrated on the figure 2) explains how to use the new features associated with sprint meetings. Each sprint meeting, the set of the identified project tasks is studied and a state is affected to each of them. Manual or electronic colored post-its are used to visually improve the differentiation between the different states. New tasks can be defined from the initial user stories and added to the set with a duration estimation of their integration into the project.

Integration process: A Redmine add-on allows to handle numeric post-its with the managing of a table of post-it corresponding to the Redmine tasks. The tool also allows to link the stories to the high level tasks identified at the

beginning of the project. This add-on had been tested by the method engineer and integrated in the project process. No other integration has been made until this change has been completely accepted by all the members of the team (3-4 months duration).

Feedback: The planning came from the management of a googledoc to the filling of story tasks during the meetings. One interesting finding was the fact that the meetings, far from being longer, due to the task filling usually made at another time, became a bit shorter. The new feature was proved very efficient to collect all the information. Moreover, it was also proved more efficient as it handled the tasks time estimation, which was not taken into account in the original Google document.

Component 1 – < {Requirements}, Plan the project with Sprint planning meeting>
Description: The goal of this component is to define the tasks within each user story and to provide the follow up of the tasks execution and of the global advancement of user stories of the project. The goal is to help the project manager to handle the tasks by having a clear vision of their life cycle.
Source situation: Planning of tasks **Target situation:** Planning of tasks
Process for each sprint planning: 1. Quick review of the user stories delivered • Make demonstration of the users stories achieved during the sprint. • Update the state of each task: Ready, Assigned, Terminated, Expired, Forwarded, Finished, Failed. • Make retrospective (analysis of the continuous improvement). • Update the status of the user stories "Assigned" using different colors of post-it. 2. Review of the user stories to deliver. • Description of the user stories to implement. The product owner describes what he wants for the next sprint. The team banter back and forth with the product owner, asking clarifying questions and driving away ambiguity • Identify a sprint goal: a one-sentence description of the overall outcome of the sprint. If a work does not directly tie to the sprint goal, then it is not done during the sprint. 3. The team decides how the work will be built. • Plan tasks for the new user stories with the estimation of the tasks duration. • Describe the sprint.

Fig. 2. Component 1: Plan the project with Sprint planning

B. Component 2: Plan the project by planning game

Context: This component was required since the beginning by the team members as it is one of the technics quickly identified as 'agile'. It was then the second component to be integrated. The use of a tool was also required as there was a member of the team who was working on a distant way and it was then necessary to work on digital cards instead of physical real cards.

Component: The aim of this component (illustrated on the figure 3) is to integrate a game play in the planning of tasks with the use of what is called a 'poker game'. All the requirements are written on cards and the game helps to estimate the duration of their development. Each team member has the opportunity to speak and to express himself until the team agrees on the estimation. The requirements are then regroup into deliverables.

Integration process: RedMine already had a planning tool called 'Planning Pocker' so it was just a matter of introduce it to the team, explain the process of the prioritization and supervise the first uses in the project.

Feedback: The acceptation of the team of this new way of planning was quite quick as it was already one of their main requirements. The team was eager to use planning poker and all the team members were very satisfied of this new way to plan and prioritize the tasks.

C. Component 3: Plan the project with Product Backlog

Context: The team wanted to have the possibility to handle the tasks on a more long term. The stories were created week after week with no specific meeting for the tasks estimation. High level tasks were created at the beginning of the project but there was missing a level of granularity between the high level and the low level tasks. The missing functionality of project management was a correct and a complete use of the backlog.

Component: This component (illustrated on the figure 4) explains how to use a backlog to plan the project. The backlog contains the list of all the necessary project functionalities. It is initiated at the beginning of the project but its elaboration can be pursued through all the development. The backlog can be considered as the team referential on the matter of the project requirements. As

Component 2 – < {Requirements}, Plan the project with planning game >
Description: The goal is to help the project manager to estimate the time needed to implement users' requirements and to establish the first schedule of deliverables. This component uses the Poker Game technique originally introduced in the XP agile method and used later in other agile methods.
Source situation: Requirements list **Target situation:** Identified and scheduled deliverables.
Process: 1. Write each user requirement on a card. Usual cards have numbers on them following the Fibonacci sequence including a zero: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89; other decks use similar progressions. The reason for using the Fibonacci sequence is to reflect the inherent uncertainty in estimating larger items, which means that the numbers are to account for the fact that the longer an estimate is, the more uncertainty it contains. 2. Evaluate the ideal time (ideal time in weeks that the project team would need to implement the given requirement) to implement each requirement using the cards. A Moderator, who will not play, chairs the process. The Product Manager provides a short overview. The team is given an opportunity to ask questions and discuss to clarify assumptions and risks. Each individual lays a card face down representing his or her estimation (units used vary - they can be days duration, ideal days or story points). During discussion, numbers must not be mentioned at all in relation to feature size to avoid anchoring. People with high estimates and low estimates are given a "soap box" to offer their justification for their estimate and then discussion continues. Repeat the estimation process until a consensus is reached. The developer who was likely to own the deliverable has a large portion of the "consensus vote", although the Moderator can negotiate. 3. Group the cards on the table to constitute deliverables (each deliverable is a group of requirements to be delivered to the customer), without ordering requirements within a deliverable. 4. Calculate the delivery dates in function of the estimations of time for requirements for each deliverable.

Fig. 3. Component 2: Plan the project with Planning game

Component 3 – < {Project requirements}, Plan the project with Product Backlog>
Description: The component objective is to help the project manager and the stakeholders to identify and manage the software requirements. The backlog is then used through the project to have a good view on the tasks developments. Sub-components: • Plan the project with an Estimating meeting • Plan the project with continuous Backlog input • Plan the sprints with Backlog
Source situation: Project requirements **Target situation:** Product Backlog
General process to use the backlog: 1. Identify high level requirements at the beginning of the project and put it in the backlog. 2. Put the stories in the backlog. 3. On an everyday basis, if a requirement is identified as not necessary, delete it from the backlog. On the contrary, if a new necessary requirement is identified, add it to the backlog. 4. Use the backlog to regularly plan the sprints.

Fig. 4. Component 3: Plan the project with Product Backlog

mentioned above, this component includes three more detailed sub-components detailed in the corresponding sub-sections.

Integration process: The used tool had a backlog functionality that wasn't used by the team and that has been integrated in the process. The unfinished tasks are now put in the backlog, as well as the stories not planned for the incoming week. The backlog is filled at the beginning directly with the client in a specific meeting dedicated to this process step and updated during the development.

The integration of this component was decomposed in three phases (corresponding to the three sub-components): first, the backlog was only used for the high level requirements defined at the beginning of the project (for at least 6 months). Then, the stories were put in the backlog on a regular basis when they were coming up, on an informal way (6 months). Finally, the backlog was used to plan the sprints (2 months).

Feedback: After more than a year, the backlog is now used on a complete way. There are several types of meeting that had been integrated in the project development process:

- Estimating meeting (with the client) to create the stories (with the poker planning technique), report on the project velocity and make some demonstrations.
- Start of sprint meeting to prioritize the stories with the client.
- End of sprint meeting to evaluate the work done and make some demonstrations to the client.

1) Sub-component 3.1: Plan the project with an Estimating meeting

Context: High level tasks were created at the beginning of the project but there was just put on a Googledoc with some informal description. As the objective was to get rid of the googledoc, it was necessary to use another way to store the requirements.

Component: This component (illustrated on the figure 5) explains how to use a backlog to initially plan the project. The backlog contains the list of all the necessary project functionalities and is initiated at the beginning of the project with the client. The backlog can be considered as the team referential on the matter of the project requirements.

Integration process: The used tool had a backlog functionality that wasn't used by the team and that has been integrated in the process. The backlog is filled at the beginning directly with the client in a specific meeting dedicated to this process step.

The integration of this component was long (6 months) as the team was quite reluctant to use the backlog. It was then necessary to integrate it carefully and without too many changes in the process.

Feedback: An estimating meeting with the client) is now used in the projects development process to create the stories (with the poker planning technique).

2) Sub-component 3.2: Plan the project with continuous Backlog input

Context: The team only used the backlog at the beginning of the project to store the high level requirements. They wanted to have the possibility to handle the tasks on a more long term. High level tasks were created at the beginning of the project but there was missing a level of granularity between the high level and the low level tasks. The project manager needed a more detailed use of the backlog.

Component: This component (illustrated on the figure 6) explains how to use the backlog through the project. The backlog contains the list of all the necessary project functionalities and its elaboration can be pursued through all the development. The backlog can really be considered as the team referential on the matter of the project requirements through all the project development.

Integration process: The tool had a backlog functionality used at the beginning of the project. The stories are now put in the backlog on a regular basis when they are coming up, on an informal way. Again the integration of this component was quite long. Even with an acceptance of the backlog functionality at the beginning of the project, the team was quite reluctant to use the backlog on an everyday basis. It was necessary to accustom them to this functionality for 6 months before going further on.

Feedback: This functionality helped a lot the team to handle the stories that were coming up through the development. The backlog is now used not only for the high level requirement but also for the low level ones and the link between them is much clearer to all the team members.

3) Sub-component 3.3: Plan the sprints with Backlog

Context: The team filled the backlog on a regularly basis but they didn't used it completely to plan the sprint on a regular way.

Component: This component (illustrated on the figure 7) explains how to use a backlog to plan the sprints. Two meetings have to be implemented for each sprint in order to compare what has been done to what is yet to be done.

Integration process: The backlog was already used to put all the requirements (high and low level). The unfinished tasks are now put in the backlog, as well as the stories not planned for the incoming week. It took two months to integrate this functionality.

Feedback: After the integration of three different sub-components (3.1, 3.2, 3.3), the backlog is now used on a complete way. Two meetings were integrated in the process: Start of sprint meeting (sprint planning meeting) and End of sprint meeting (review sprint meeting). The sprints are now of two weeks on a regular basis and the client has been integrated on a better way in the meetings.

Sub-component 3.1 – < {Project requirements}, Plan the project with an Estimating meeting>
Description: The component objective is to help the project manager and the stakeholders to identify the high level software requirements. Looking at the problem description, the team identifies some high level requirements that will be refined later on in the process. There is no predefined formalism for the backlog; it can be a table or a text document, a database or even a set of post-its. Each item has to be described on an atomic way (only one requirement for each item). The Product Backlog contains features, bugs, technical work and knowledge acquisition.
Source situation: Project requirements **Target situation:** Product Backlog
Process to initiate the backlog: 1. Identify high level requirements at the beginning of the project and put it in the backlog. 2. Put the stories in the backlog.

Fig. 5. Component 3.1: Plan the project with an iniital Product Backlog

Sub-component 3.2 – < {Project requirements}, Plan the project with Product Backlog>
Description: The component objective is to help the project manager and the stakeholders to manage the software requirements in real-time. After its initial definition, the backlog is updated on a day-to-day basis. Once the backlog gets larger, it may be necessary to group the backlog into near-term and long-term items.
Source situation: Project requirements, Product backlog **Target situation:** Product Backlog
Process to use the backlog: 1. On an everyday basis, if a requirement is identified as not necessary, delete it from the backlog. 2. On the contrary, if a new necessary requirement is identified, add it to the backlog.

Fig. 6. Component 3.2: Plan the project with Product Backlog

Sub-component 3.3 – < {Project requirements}, Plan the sprints with Backlog>
Description: The component objective is to help the project manager to correctly plan the sprints. The product backlog is used to handle the requirements and to indicate which tasks are under way, finished or to be done. Two meetings are introduced at the beginning and at the end of each sprint to evaluate, with the product owner, which stories have to be developed for this sprint and what has been done already.
Source situation: Product backlog **Target situation:** product backlog, Sprint
Process to plan the sprints with the backlog: 1. At the beginning of the sprint, make a sprint planning meeting to prioritize the stories with the product owner. This meeting helps to define the sprint goal: a short sentence describing what the team plans to achieve during the sprint. The team and the product owner write it collaboratively. The sprint backlog is the other output of the sprint planning. It is a list of the product backlog items the team commits to deliver and the list of the necessary tasks. 2. At the end of the sprint, make a sprint review meeting to evaluate the work done and make some demonstrations to the product owner. The success of the sprint will later be assessed during the sprint review meeting against the sprint goal, rather than against each specific item selected from the product backlog.

Fig. 7. Component 3.3: Plan the project with Product Backlog

D. Component 4: Manage the project with daily meeting

Context: The project team was quite small so its members usually met and talked around the coffee machine. However, as there was a team member working remotely, it was difficult to have a real efficient communication between all the members. The tool used helped to have information about the advancement state of all the tasks but some crucial details were missing, only evocated in an informal way outside the official meetings.

Component: This component (illustrated on the figure 8) explains how a daily meeting (or stand-up meeting) can be put in place. The team meets once a day, always at the same time to share the development advance. This meeting is usually scheduled for short time. If a point seems to take too long, the team put it away to be discussed with only a sub-part of the team. This meeting allows sharing all the important information about the development. More than team-building exercises, regularly communicating, working, and helping each other build effective teams. This is also linked with team members helping each other with shared obstacles.

Integration process: Meetings were planned regularly to improve the communication between all the team members, with a direct internet access for the team member working remotely.

Feedback: the meetings improved a lot the communication between the team members, essentially concerning the developer who was working remotely and who is now completely aware of all the information about the project.

Component 4 – < {Task planning}, Manage the project with Daily meeting >
Description: The goal is to help the project manager to handle the project by organizing a short meeting (less than 15 minutes) everyday. The whole team should attend the meetings. All the important things about the development can be shared (pitfalls, new requirements). The daily commitments allow team members to know about potential challenges as well as to coordinate efforts to resolve difficult and/or time-consuming issues.
Source situation: Sprint **Target situation:** Sprint
Process: The daily meeting should not excess 15 minutes. If something should take longer, then plan another meeting to discuss about it. The meeting can take place with participants standing up to remind people to keep the meeting short and to-the-point. This meeting should take place at the same time everyday. All team members are asked to attend, but the meetings are not postponed if some of them are not present. 1. All the team members share their advance on the development. They talk about progress since the last day, the anticipated work until the next one and any impediments, taking the opportunity to ask for help or collaborate. 2. To help, the team members can ask themselves three questions: What did I accomplish yesterday? What will I do today? What obstacles are impeding my progress? It may not be always practical to limit all discussion to these three questions but the goal is to stick as closely as possible to them. 3. Obstacles are written on a board, which is publicly visible, identifies raised obstacles and tracks the progress of their resolution. This board can be updated outside of the daily meetings and serves as a more immediate and less confronting way to initially raise obstacles.

Fig. 8. Component 4: Manage the project with Daily meeting

E. Component 5: Manage the project with Burndown chart

Context: It was difficult for the project team to have a long-term view on the incoming sprints. The information was there but sometime disseminated when the same story was in several sprints so it was difficult to have a good time evaluation on the different sprints. Progress on a project can be tracked by means of a burndown chart, updated at the end of each sprint.

Component: The component (illustrated in the figure 9) shows how to use a burn down chart to have a better view of the incoming works. It results in up-to-date project status being, it encourages the team to confront any difficulties sooner and more decisively.

Integration process: Redmine had this functionality in its toolbox. However, the team didn't use it as the planning wasn't correctly implemented at the beginning of this project. Now that the planning information is completely filled on the tool, the functionality can be used on a more efficient way. The daily meeting helps the team members to enter correct information on a regular way.

Feedback: This component has been integrated quite recently but the first feedbacks are good. The team has a better estimation of the remaining works and is able to have a better planning of the tasks/sprints.

Component 5 – < {Task planning}, Manage the project with Burndown chart >
Description: The goal is to help the project manager to handle the project with a better view of the incoming works. A burndown chart is a graphical representation of work left to do versus time. The burndown chart is an essential part of any agile project and is a way for the team to clearly see what is happening and how progress is being made during each sprint.
Source situation: Task Planning, Backlog **Target situation:** Burndown Chart
Process: 1. Construct the burndown chart: The burndown chart is a graphic with two axis. The horizontal axis represents the sprints. The vertical axis corresponds to the amount of work remaining at the start of each sprint (in whatever unit the team prefer – story points, ideal days, team days, etc.). a. The project start point is the farthest point to the left of the chart and occurs at day 0 of the iteration. b. The project end point is the point that is farthest to the right of the chart and occurs on the predicted last day of the project/iteration c. The ideal work remaining line is a straight line that connects the start point to the end point. At the start point, the ideal line shows the sum of the estimates for all the tasks (work) that needs to be completed. At the end point, the ideal line intercepts the x-axis showing that there is no work left to complete. d. The actual work remaining line shows the actual work remaining. At the start point, the actual work remaining is the same as the ideal work remaining but as time progresses; the actual work line fluctuates above and below the ideal line depending on this disparity between estimates and how effective the team is. In general, a new point is added to this line each day of the project. Each day, the sum of the time or story point estimates for work that was recently completed is subtracted from the last point in the line to determine the next point. 2. Interpret the burndown chart a. If the actual work line is above the ideal work line, then there is more work left to do than originally predicted and the project is behind schedule. b. If the actual work line is below the ideal work line, then there is less work left than originally predicted and the project is ahead of schedule.

Fig. 9. Component 5: Manage the project with Burndown chart

During the period covered by this study, the planned agile method components were integrated progressively. Thus the integration included five steps as the integration of method components was done consecutively. The integration of components did not follow their logical order from the initial agile method. We can say that all selected method components come from SCRUM essentially, but some of them are also present in other agile methods. The logical sequence of SCRUM and other agile methods would suggest the integration of these components at the same time.

V. LESSONS LEARNED

A. 5.1. Positive results

Progressive integration of methods components possible. The integration of the method components in the original method succeeded. The team now uses an enriched method containing the new required features.

Method components acceptance. It seems that a gradual integration leads to a better acceptance of new methods. A smooth change, continuous on a long period of time, allows to introduce the changes one little step after one little step, until the desired amount of change is reached (or if there is a complete opposition to a change, which will have to be handle correctly). At the end of this specific experience, all of the missing SCRUM techniques have been introduced to the project team.

Better consideration of the team priorities. We studied the features required by the changes needed in the method, but we also studied the change reluctance of the project team. We then prioritized the components by taking into account the importance of each feature and the easiness of its acceptation.

B. 5.2. Limitations

Method Engineer. A method engineer who was also part of the development team conducted the integration. It was then easier for us to forecast the potential problems and to realize a smoother integration, adapted to the development team. It will certainly be more difficult for a method

engineer outside the development team to convince the team to accept the changes.

Common ground. This kind of integration is only possible if there is only one project at the same time or if all the development teams agree to modify the development method at the same time. There are sprints for each project and it would be very difficult to use different methods at the same time for the same people. A common ground is necessary to handle correctly the developments.

VI. CONCLUSION AND FUTURE WORKS

This experience showed that a progressive integration of new method components in a development method is possible and can be nicely accepted by the development team.

In this experiment, the integration leaded to an increasing efficiency. The company can now claim, in total honesty, that they are using the SCRUM methodology in their projects. The team is happy to work now with a more formalized process, they have improved their meeting duration, their task estimations, and their marketing relationships. The clients have an access to a part of the tool information so they are much more involved in the development process.

The integration has been made on two distinctive ways. First, the method engineer integrated the planning and estimations tasks on a slowly and very gradual way. The change improved the week meetings as they came from during 3 hours to 2 hours then 1h30 (for a team of 5-8 people). This improvement was mainly due to the fact that there were no more discussions about things already everybody agreed on. It was a matter of formalizing in the tool what already existed in the Google document. Second, the method engineer integrated the problematic of the backlog and burndown chart. As the first changes had been happily accepted by the team, it was quite easier to introduce new techniques into the development process. These changes improved the middle-term estimation of the tasks, a closer relationship with the client, and a quicker and more efficient sharing of the information.

Our future works will be to handle new experiments in other companies to see if we reach the same results and if we can improve the method components. New experiments would allow to understand for which kind of projects the progressive integration is appropriate and which are the factors for a successful integration of method components. We intend to organize new experiments using a formalized evaluation protocol in order to gather the information about efficiency of the suggested approach.

VII. REFERENCES

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, Manifesto for Agile Software Development. Agile Alliance, 2001.

[2] Serena Software, “An Introduction to Agile Software Development”, Inc. Serena, Mariner, TeamTrack, http://www.serena.com (accessed in march 2015), 2007.

[3] D. Anderson, “Agile management for software engineering: Applying the theory of constraints for business results,” 2003.

[4] K. Beck, “Extreme programming explained: embrace change,” Addison-Wesley, UK, 2000.

[5] K. Schwaber and M. Beedle, “Agile Software Development with Scrum”, Prentice Hall PTR, Australia, 2002.

[6] A. Cockburn, “Agile software development”, Addison-Wesley, London, UK, 2002.

[7] J. Stapleton, “DSDM −Dynamic system development method.” Addison-Wesley, UK, 1995

[8] VersionOne, “8th annual state of agile development survey”, http://www.versionone.com (accessed in Nov. 2014), 2013.

[9] S. Brinkkemper, and R.J. Welke, “Methodology Engineering: A Proposal for Situation Specific Methodology Construction”. In: Challenges and Strategies for Research in Systems Development, Cotterman, W. and J. Senn (eds.), J. Wiley, Chichester, UK, 1992, pp. 257-266.

[10] J. Ralyté, and C. Rolland, “An Assembly Process Model for Method Engineering”. In: Proceedings of CAISE 2001, LNCS 2068, Springer, Berlin, 2001, pp. 267-283.

[11] R. Deneckere, “Approche d'extension de méthodes fondée sur l'utilisation de composants génériques”. PhD thesis. University of Paris 1 Panthéon-Sorbonne, 2001

[12] F. Karlsson, P. J. Ågerfalk, “Method Configuration: The eXtreme Programming Case”, in XP 2008, Limerick, Ireland, pp 32-41, 2008.

[13] M. Rossi, B. Ramesh, K. Lyytinen, and J.-P. Tolvanen, “Managing evolutionary method engineering by method rationale”. Journal of the AIS, vol. 5(9), 2004, pp. 356-391.

[14] M. Cervera, M. Albert , V. Torres , and V. Pelechano, “A Model-Driven Approach for the Design and Implementation of Software Development Methods”. International Journal of Information System Modeling and Design (IJISMD), vol. 3(4), 2012, pp. 86-103.

[15] G. Guzélian, and C. Cauvet, “SO2M: Towards a Service-Oriented Approach for Method Engineering”. In: Proceedings of IKE'07, Las Vegas, Nevada, USA 2007.

[16] A. Iacovelli. Approche orientée service pour la configuration de méthodes outillées. PhD thesis. University Paris 1 Panthéon-Sorbonne, 2012.

[17] J. Shore, “The art of agile development”, O'Reilly Media Editors, 2007.

[18] M. Cohn, “Agile estimating and planning”, Prentice Hall editors, 2005.

[19] B. Meyer, “Agile!: The Good, the Hype and the Ugly”, Springer Publishing, 2014.

[20] A. Qumer, B. Henderson-Sellers, “Construction of an Agile Software Product-Enhancement Process by Using an Agile Software Solution Framework (ASSF) and Situational Method Engineering”, In COMPSAC 2007, Beijing, China, 2007, pp 539-542

[21] F. Karlsson, and P.J. Ågerfalk , “Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets”. Inf. and Soft. Technology, vol. 46(9), 2004, pp. 619–633.

[22] Z. Abad, M. Sadi, and R. Ramsin, “Towards tool support for situational engineering of agile methodologies” In APSEC 2010, Asia Pacific, 2010.

[23] Z. Abad, A. Alipour, R. Ramsin, “Enhancing Tool Support for Situational Engineering of Agile Methodologies in Eclipse”, Studies in Computational Intelligence, Vol 430, pp 141-152, 2012.

[24] K. Beck, Extreme programming explained: Embrace change. Addison-Wesley, UK, 1999.

[25] R. Deneckere, E. Kornyshova, J. Ralyté. Famille de méthodes: la flexibilité au coeur du processus de construction de méthodes. ISI Journal, 19(1): 67-95, 2014.