# Reflections on Reflection in Agile Software Development

David Talby[1], Orit Hazzan[2], Yael Dubinsky[3] and Arie Keren[1]
*[1] MAMDAS – Software Development Unit*
*Air Force, IDF, Israel*
*davidt@cs.huji.ac.il*
*ariekk@netvision.net.il*
*[2] Department of Education in Technology and Science*
*Technion – Israel Institute of Technology*
*oritha@techunix.technion.ac.il*
*[3] Department of Computer Science*
*Technion – Israel Institute of Technology*
*yael@cs.technion.ac.il*

## Abstract

*This paper analyzes the reflections of an agile team, developing a large-scale project in an industry setting. The team uses an Iteration Summary Meeting practice, which includes four elements: The customer's summary, a formal presentation of the system, review of metrics and a reflection. The technique for the entire meeting and for the reflection element in particular is described, and empirical evidence is given to show that it is assessed as highly effective, achieving its intended goals, and increasing team satisfaction. Further, the proposed practice supports tracking past decisions. This practice is shown to be valuable to stabilizing a new project as well as a continuous improvement forum for a stable project. It also incurs a lower overhead than existing alternative reflection practices.*

## 1. Introduction

Reflection and continuous process improvement are a fundamental aspect of agile software development. The agile manifesto [3] contains a principle stating that an agile team should regularly reflect on how to become more effective, and tune its behavior accordingly. Cockburn [4] asserts that "each situation calls for a different methodology", and entails that a key agile practice should be conducting regular post-iteration workshops aimed at reflection and process tune-up.

Several systematic approaches have been suggested on how to conduct effective reflection in agile teams. The most widely known are post-iteration workshops [4,15] and the postmortem review technique [5,13]. Empirical studies on the effectiveness of these methods have been conducted by Salo et al. [14,15,16], and a few experience reports have been published as well [12]. This paper complements and extends the current body of knowledge in this area, in three primary respects.

First, this paper presents empirical evidence taken from a real, large-scale agile software project developed in the Israeli Air Force. In contrast, to the best of our knowledge the only empirical data published on this issue to date is based on a series of short-term projects done in a research setting. This is important since in a long-term, industrial setting, team members may act very differently towards process improvement practices (such as reflection), since they have a longer and more profound effect on their daily lives. Our data confirms that reflection is a useful way to improve an agile team's effectiveness and team satisfaction in a real-world setting.

Second, this paper proposes and analyzes an iteration summary practice that is different from those studied to date [4,5]. Specifically, the meeting's protocol contains several retrospective elements, only one of which is team reflection. The reflection phase itself follows slightly different ground rules than commonly suggested [4,5]. The end result is a more light-weight process, which takes only 2.1% of the team's time, in contrast to 3.7% and 4.7% reported in

[15] and [5] respectively. We present data that shows the effectiveness of the new elements in the iteration summary in the team's eyes, and analyze data pertaining to the modified aspects in the reflection technique.

Third, our quantitative and qualitative data about the method's effectiveness uses a different research method than applied before. While the work of Salo et al. [14,15,16] is mainly based on action research [1], we use team members' perception on the process – i.e. their reflections on reflection – as the main data source. As the two research methods each have their own merits and weaknesses, this work complements previous work to draw a more complete joint picture.

This paper is structured as follows. Section 2 presents background and related work. Section 3 describes our research setting and method. Section 4 presents the Iteration Summary Meeting practice, including the team's perceptions about it. Section 5 presents the "classic" reflection element of the iteration summary, and analyzes both the technique used and its effectiveness. Section 6 compares our Iteration Summary Meeting practice and its reflection in particular to other proposed methods. In section 7 we conclude.

## 2. Background and Related Work

### 2.1. The Notion of Reflection

Reflection is the process according to which an individual (or a group) examines his/her/its actions during the accomplishment of the action or after it. Though reflection is not a new concept, its common practice has been boosted after Schön had published his two books *The Reflective Practitioner* [17] and *Educating the Reflective Practitioner* [18], which advocated the idea that a person who keeps reflecting becomes a Reflective Practitioner, a position which enables him or her to keep improving his or her professional skills. While the first book presents professions for which reflective thinking is (or should be) inherent, such as architecture and management, the second book focuses on how *to educate* students of such professions to be reflective practitioners. The working assumption in all cases is that such a reflection improves both proficiency and performance within such professions.

In the two books mentioned above, Schön analyses the added advantages one may obtain from continuously examining one's practice and one's thinking about his/her practice. With respect to science and engineering, Schön says that "[b]etween 1963 and 1982 … [i]ncreasingly we have become aware of the importance to actual practice of phenomena – complexity, uncertainty, instability, uniqueness, and value-conflict" ([17], p. 39). At that time, the Computer Science community observed a similar phenomenon with respect to developing software systems (Cf. the "Software Crisis" terminology introduced in 1968 at the NATO Conference in Garmish, Germany). Many at the conference recognized that software development should be guided by a professional-systematic approach. The mental complexity involved in developing software projects was acknowledged, and, as a result, there was tremendous awareness of the impossibility of managing software systems without systematic (engineering oriented) methods. However, though the complex nature of the profession of software development was known at the time when Schön wrote his books, he did not discuss the application of the reflective practitioner perspective with respect to software engineering.

The work presented in this paper follows other publications that emphasize the importance of reflection and retrospective in the context of software development in general and with respect to agile methods [4] in particular. In general, Schön [19] discussed this application. Other examples are Hazzan [7], who describes the relevance of the reflective practice perspective to software engineering based on a systematic analysis of Schön's book, and Kerth [11] who specifically applies the retrospective perspective to software development process.

### 2.2. Reflection in Extreme Programming

Hazzan and Tomayko [8,9] suggested adding a reflective practice (RP) perspective to agile software development processes in general and to Extreme Programming (XP) [2] in particular. Specifically, based on Schön's work mentioned above, it was suggested that as a reflective practitioner one may improve the performance of the XP practices. Analysis of the field of Software Engineering (SE) and the kind of work that software engineers usually accomplish in general and the XP practices in particular, support the adoption of the RP perspective to SE in general (as did Hazzan in [7]) and to XP in particular. Specifically, in Hazzan and Tomayko [8] it is suggested that a reflective mode of thinking may improve the application of some of the XP practices, as follows.

It seems that a RP approach fits very well to XP, since XP emphasizes learning through reflection processes. For example, the estimation of the team's velocity is improved from project to project based on a reflective process; when a pair is engaged in a pair programming session, the navigator reflects on the drivers' coding. Thus, it seems that one of the implicit XP guidelines is reflection. Still, as far as we know, it is not outlined inherently in the practices themselves. Similarly to some of the XP practices, RP is not explicitly directed to code production but in the long term it may improve code production and quality. As XP incorporates activities that are not directly oriented to code production, yet may improve code development processes, we suggest that the RP perspective may be integrated naturally in XP.

### 2.3. Reflection in Other Agile Methodologies

Some agile methods other than XP have already integrated some form of reflection into their practices. Kähkönen's recent model for deploying an agile method in an organization [10] includes a post-iteration workshop and analysis of metrics as the central continuous improvement tool of the model. The project analyzed in this paper, which has independently used a very similar practice, provides an empirical case study of this model's effectiveness.

Cockburn's Crystal family of methodologies [4] regards the need to tune the development process to the specific needs of each team and project as a key issue, and refers to "the mystery of how to construct a methodology for each situation without spending so much time designing the methodology" ([4], p. 184). The proposed solution is a reflection workshop, conducted regularly during the mid- or post-iteration events. Such a workshop begins with gathering issues that need to be discussed, and ends with a list of tasks and decisions about concrete changes in team behavior.

Dingsøyr and Hanssen [5] have proposed postmortem reviews – a "lightweight" version of the similar "heavyweight" highly recommended practice for large projects [13]. They also suggest performing the review once per iteration, involving the whole team. They propose a technique called "the KJ method", named after the Japanese ethnologist Jiro Kawakita [20]. In this method, each participant lists 3-5 issues about the development process on post-it notes. The notes are then grouped into positive and negative issue, and the negative issues are prioritized. The high-priority issues are then discussed by the team, and turned into action items for changing the team's behavior. The authors report that the method was effectively used in several XP projects.

Salo et al. [14,15,16] provide an empirical evaluation of the above two methods, based on a series of research projects at the VTT Technical Research Center of Finland. Each of four reported projects had 4-6 developers, and was six weeks long. A combination of post-iteration workshops and postmortem reviews was used, but all issues were considered at each workshop (instead of only high-priority ones). The number of positive and negative issues found at each workshop, the total overhead of the workshops, the number of resulting action items and the number of implemented action items were all measured. Since this is the main source of empirical evidence about agile reflection to date, most of our comparison later on will relate to it.

## 3. Research Framework

### 3.1. Research Setting

The subject of research of this paper is a large-scale agile software project developed at the Israeli Air Force. The project is a business-critical enterprise information system, considered to be highly complex and intended for a large and varied user population. The agile software development method used in the project is based on Extreme Programming [2], with a few adaptations in line with the agile approach that were dictated by the project's size and the organization.

Since the project is both large-scale, and the first one of its scope to be developed in the organization using an XP-based method, it was considered risky and, consequently, was scrutinized by many different stakeholders. In order to manage risks and provide full and timely information about the project's progress, a set of metrics was developed [6], data was consistently collected and analyzed, and regular reflection meetings and formal iteration summary meetings were conducted. These were the main stabilization and (later on) continuous improvement forums for the team, and they are the ones analyzed in this paper. The project team's research efforts were complemented by external consulting researchers.

This paper presents data based on the first four releases (eight months) of the project. The project's development team averaged 15 developers during this period; this is an average since the team experienced several personnel changes. According to the Whole Team practice, the development team includes a mix of programmers, business analysts, testers and managers. Note that although 15 people filled the questionnaires used as our main data source, not all of them were in the team throughout the entire period. Still, the answers include at least two people of each of the above roles.

### 3.2. Research Method

The main research method used in this paper is personal reflection of team members on the reflection process, done via written questionnaires several months after the reflections in question took place. This is inspired by Hazzan and Tomayko's ladder of reflection [8] used to illustrate the potential contribution of adding a reflective practice to XP. The concept of the ladder of reflection is described in ([18], p. 114):

> *We can begin with a straightforward map of interventions and responses, a vertical dimension according to which higher levels of activity are "meta" to those below. To move "up", in this sense, is to move from an activity to reflection on that activity; to move "down" is to move from reflection to an action that enacts reflection. The levels of action and reflection on action can be seen as the rungs of a ladder. Climbing up the ladder, one makes what has happened at the rung below an object of reflection.*

In a reflection workshop of an agile software team, team members reflect on their day-to-day activities. When asked to fill a detailed questionnaire, for the purposes of this research, regarding their attitudes towards the reflection practice, team members are performing a reflection on the reflection process – one level up the ladder. This paper can be viewed as a reflection on their responses – another level up.

In contrast, the research method in Salo et al.'s papers [14,15,16] is action research [1], which focuses more on what practitioners do rather than what they say they do [1]. For example, [14] concludes that reflection raises team members' satisfaction, since the number of negative issues found in reflection workshops decreases along time. In contrast, we simply ask team members' (in several ways) whether they are satisfied with reflection as practiced in their team.

The reason why a ladder of reflections was chosen as the main research method for this paper was because action research requires intervention [21], which was impossible since the analyzed project is not a research project. However, this provides an unintended benefit since our data well complements Salo's work, as each research method has its strengths and weaknesses. For example, regarding participant satisfaction, the ladder of reflection approach seems more appropriate, since satisfaction is by nature a matter of individual perception. On the other hand, directly measuring the number of action items that were actually implemented by the team (as done in action research) seems better than asking team members how often this happened.

## 4. The Iteration Summary Meeting

This section describes the researched project's iteration summary practice and the team's reflections about it. The meeting had the following strict protocol, which was first used to summarize the first iteration, and remained virtually unchanged:

| | |
|---|---|
| 9:00-9:10 | Customer's summary of the iteration |
| 9:10-9:25 | Formal presentation of the system |
| 9:25-9:50 | Review of iteration's metrics |
| 9:50-10:45 | Reflection (see section 5 below) |
| 10:45-11:00 | Break before planning game |

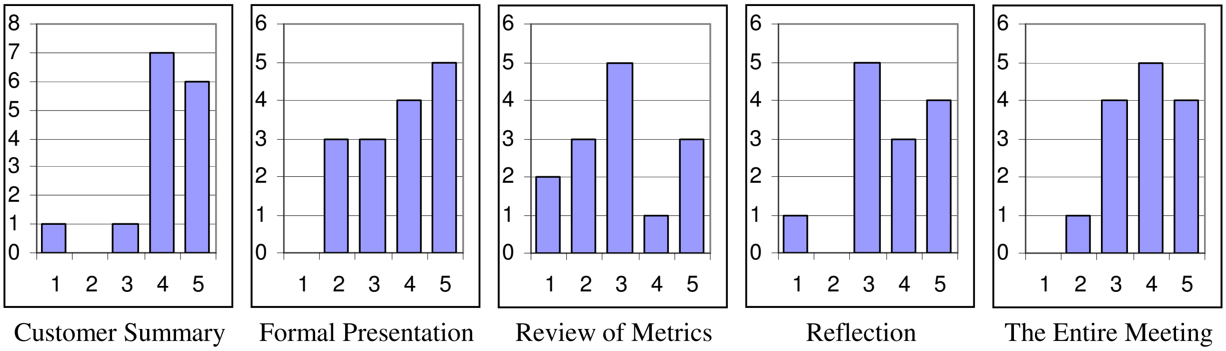
**Listing 1**. Iteration Summary Meeting Protocol

The planning game for the next (beginning) iteration started immediately after this meeting. All team members were required to participate, and when the project was regrouped into three smaller teams in the third release, this meeting was still shared, and became the only formal project-wide gathering. Only the planning games were separated. The following lists the content and intended goal of each meeting element.

**The Customer's summary** was a short, informal verbal summary of the iteration, given by the customer. This was direct feedback, usually focused on the product rather than the process. When designing the iteration summary, it was important for the team's management to begin the iteration summary with the customer's message, to signal his importance in the team. It also helped in focusing people on the product as an end goal, rather than their own specific tasks or the development process by itself.

**The Formal Presentation** of the system was a demo of the main new features of the ending iteration, run on the actual, integrated system. This did not replace a separate meeting with the customer to present the iteration's product, done towards the end of each iteration, which normally took two to four hours of a task-by-task demonstration and intense free-hand use of the system by the customer. The formal presentation had two different goals. First was to make sure that the system is indeed fully integrated, and that the team is able to deploy it at will (this was non-trivial at first since multiple servers were involved). Second was to make sure that everyone knew all of the system's features, from a user's (in contrast to a programmer's) point of view. The second reason became the prominent one as the team grew, while the deployment scheme stabilized.

**The Review of Metrics** was a presentation and analysis of the ending iteration's metrics. As described in detail in [6], four metrics were presented from the beginning of the project: The product metric (amount



Customer Summary Formal Presentation Review of Metrics Reflection The Entire Meeting

**Figure 1.** Perceived Importance of Elements of the Iteration Summary Meeting

of written and passed tests), the pulse metric (measuring continuous integration), the burndown metric (estimating convergence to release goals), and defect metrics (number of new and open defects). Starting from the middle of the third release, task-by-task estimated versus actual time was also analyzed, as well as the time reported for overhead activities.

The goal of this element of the iteration summary was twofold. First was to present the data to the entire team, replacing individual perception (for example, about product quality, time lost to overhead, etc.) by facts. Second was to openly discuss the reasons behind the metrics, since metrics cannot be analyzed regardless of context. For example, a decrease in the number of new defects does not necessarily stem from improved product quality: It may be the case that less testing was performed in this iteration (and thus fewer bugs were found), or that people don't report all defects into the common defects table (for example if they fixed them at once and consider the report redundant).

**The Reflection** is intended to discuss a specific problem in the development process, and change it as necessary. It is described in detail in the next section.

Figure 1 summarizes the team's answers to the following question: "Indicate the importance of each element of the iteration summary meeting". Possible answers were on a scale from 1 (unimportant) to 5 (very important). Team members were also given the option to explain their choices in free writing.

The results indicate that as a whole, team members consider the iteration summary meeting to be of high value – its average importance is 3.9. The most important element of the meeting according to this team is the customer's summary, with an average of 4.1. Note that this is a simple ten-minute element at the beginning of the meeting. It seems that team members place very high value on this direct form of feedback; as one wrote: "It's hard to explain why, but it's good to know what he thinks".

The formal presentation of the system as well as the reflection elements both received an average of 3.7. Respondents' explanations of their choices were usually in line with the intended goals of these elements. The review of metrics element received an average importance of 3.0, and the written comments support the impression that team members are divided in their opinion regarding its importance. All managers and team leaders view it as highly important, while some of the novice developers wrote that "it is mainly of interest to managers". We believe these results reflect an inherent difficulty in balancing an informative workplace against some programmers' general dislike of "management".

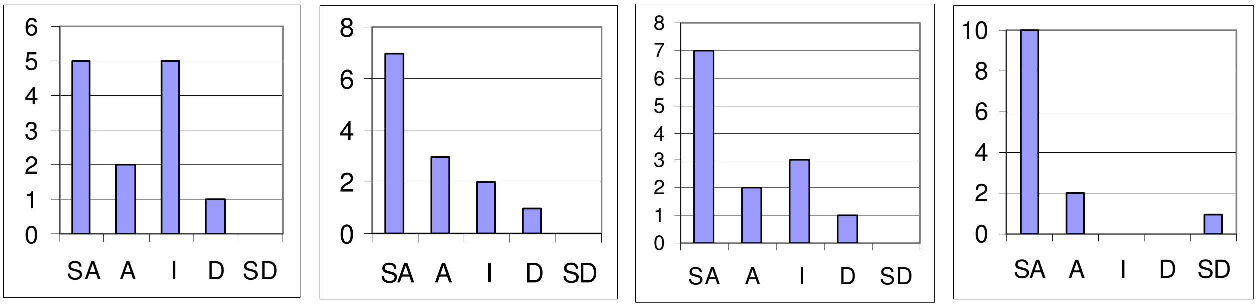
When asked the open question: "Which elements of the meeting should be modified (extended, reduced or cancelled)?" The most popular answer was that reflections can be extended, when their subject is very important. There were no suggestions to cancel any element, and (this was asked in a separate question) no offers for new elements or additional continuous improvement practices.

## 5. The Reflection Practice

### 5.1. Technique

In line with the guidelines used in [4,5,15], this team's reflection practice was designed to be "agile": Enable small incremental improvements in the development process, achieved by a very simple process which encourages high communication and feedback. Listing 2 summarizes the details of the reflection technique used in this team; judging from this research's results, we recommend it as a recipe for adopting reflection as an XP or agile practice in other teams as well.

The technique employs one special role – the reflection's moderator. It was usually carried out by the team leader, and this was exploited to extract the subject selection as well the exact phrasing of action items out of the reflection meeting itself.



"Decisions reached in reflection meetings were usually implemented"

"Reflection is an efficient way to solve tensions and conflicts"

"One of the goals of reflection is to let people 'blow off steam' on an issue"

"In a reflection anyone can speak their mind fully, even if it is obviously unaccepted"

**Figure 2.** Reflections on the goals of Reflection Meetings

- Only one specific problem is discussed at each reflection meeting.
- The discussed problem should relate to the development process, not the developed product.
- The subject is chosen in advance by the moderator (after informal consultation with other team members), and presented at the beginning of the reflection meeting.
- The reflection cannot exceed one hour.
- The whole team is required to attend the reflection.
- The reflection may be an open discussion, or use some structured problem solving technique.
- Everyone is proactively encouraged to speak, but is not required to do so.
- Team members are encouraged to speak their own opinions, as bluntly as they see fit.
- The moderator records important insights and proposed action items that surface during the meeting.
- The moderator summarizes the meeting by reading to the team the decided action items.
- The decided action items are effective immediately. They are actual changes in day-to-day team operations that should reduce the debated problem.
- The moderator publishes the main insights and action items to the teams soon after the reflection. Emails and newsgroup posting were the common format for these messages.

**Listing 2.** Reflection Technique

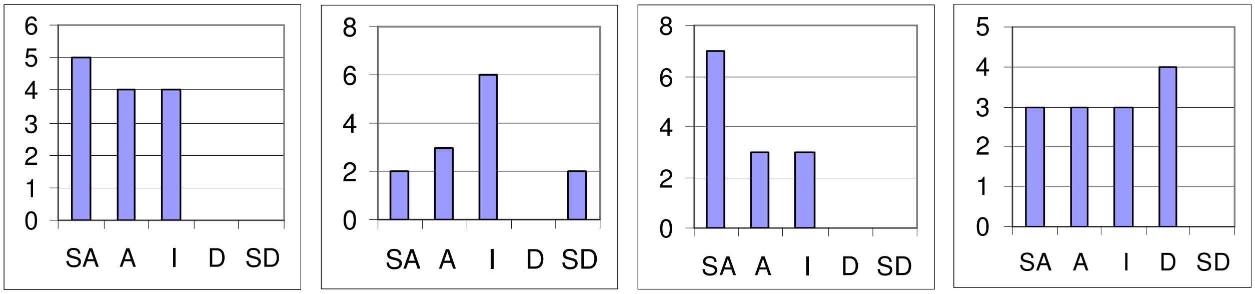
## 5.2. The Goals of Reflection

Figure 2 summarizes team members' reflections about the goals of the reflection process. They were asked about their agreement with given statements, and possible answers were strongly disagree (SD), disagree (D), indifferent (I), agree (A) or strongly agree (SA).

In addition to the initial purpose of process improvement, reflection has two additional social goals: To resolve conflicts in a business-like rather than emotional manner; and to enable people to 'blow off steam' on disturbing issues, thus clearing away negative feelings. An open question about the value of reflection did not raise additional goals. These goals are in line with other reports [12].

As Figure 2 shows, team members generally assess the reflection technique as carried out in this project to be achieving these goals. Questionnaire respondents agree with an average of 3.8 that decisions reached in reflection meetings are usually implemented (where SD is counted as 1 and SA as 5). Note that almost half of the team did not "agree" or "strongly agree" to this claim; written comments on this issue stated that the realization of some decisions was not tracked well enough. We will return to this issue in section 6.1.

Reflection is viewed as an excellent tool to resolve conflicts (average agreement of 4.2) and to vent negative feelings on an issue (average 4.2 as well). The main cause for this was the fact that team members were able to express their true opinions and feelings about the debated subject, even if it was obviously unaccepted by the rest of the team. This statement had an average agreement of 4.5, with only one disagreement who wrote that personal insults should not be allowed. Blunt and sarcastic statements were a natural part of reflections, and the results suggest that their positive effects, of creating an open and honest atmosphere, outweigh their negative effects.



"The subjects of reflection meetings are usually relevant to day-to-day work"

"A structured reflection is preferable to free conversation"

"It's important to publish the reflection meeting's decisions immediately after it ends"

"Limiting the reflection's time sometimes impeded the ability to get to the root of the problem"

**Figure 3**. Reflections on the technique of the reflection meeting

### 5.3. The Reflection Technique

The questionnaire filled by team members included several questions intended to measure several specific aspects of the reflection technique. We have chosen to focus to four aspects, summarized in Figure 3.

First, team members assess the subjects of reflection meetings to be relevant on their ongoing work (average of 4.1, no one disagreed). This is of importance since in contrast to other suggested reflection techniques, such as post-iteration workshops [4,15] and post-mortem reviews [5], in this technique the subject is chosen in advance by the moderator (which was usually the team leader in this project). This has the advantage of not spending time to find and decide on the subject during the reflection meeting itself, thus lowering the overall time it requires. The risk of selecting "wrong" subjects did not seem to materialize in this project, probably because the moderator is the team leader, who is a member of the development team and is thus intimately familiar with its day-to-day problems.

The questionnaire also included two open questions, asking which subjects were the best to handle in a reflection, and which subjects were the worst. The results were very consistent among different team members, and are summarized in Listing 3. They should be regarded as strong guidelines for future teams.

Reflection subjects **should** be …

✓ Relevant to the entire team
✓ Organizational issues
✓ Issues not everyone agrees on

Reflection subjects **should not** be …

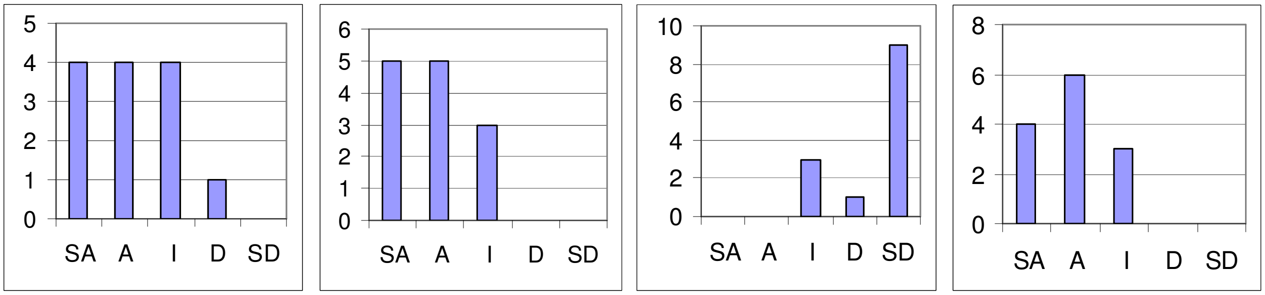
× Personal quarrels and accusations
× Technical problems
× High (external) management initiatives

**Listing 3**. Reflection Subjects Do's and Don'ts

Second, only five team members agreed that structured reflections were preferable to unstructured ones (average of 3.2). Only four reflection meetings were structured throughout the examined period. It may be the case that the right problem solving technique was not introduced to this team. In any case, open discussions were sufficient to achieve the bottom-line positive results presented in the next section.

Third, publication of a written summary of each reflection meeting, even if done in an informal forum such as email, was perceived by the team as highly important (4.3 average, no one disagreed). In free writing comments, people explained that this was important mainly to prevent arguments on whether something was agreed upon and in what exact way, particularly as the project's history became longer and the risk of forgetfulness rose. An additional benefit was enabling newcomers to the team to catch up quickly with the list of process adjustments and general insights that the team applied to the textbook XP practices.

Fourth, the team had mixed opinions on whether the strict limit on the reflection's time frame impeded its effectiveness. The average agreement to this statement was 3.4, and written comments on the issue where mostly of two flavors: Commenting that more time was required only in several difficult reflections; and commenting that in some reflections much more time could have been spent, but this would not have led to any improved results. It may be the case that when trying to balance a fruitful discussion, a minimal overhead and a thorough investigation of the given problem, maintaining a strict one-hour limit goes against the last of these goals. However, since the team is undecided on this issue, it is impossible to make any specific recommendations.



"When I'll be a team leader, I'll conduct ongoing collection and tracking of metrics"

"I'll be glad if Reflection will be used in my next team"

"I don't understand at all the purpose of reflections"

"Raising a problem in a reflection meeting is better than having a decision made by the team leaders"

**Figure 4**. Reflections on the Perceived Effectiveness of Metrics and Reflections

### 5.4. Effectiveness of Reflection

Having discussed the technique and the goals of the Reflection practice, we now turn to the most important question: It is effective? In this paper we analyze the *perceived* effectiveness as viewed by the team, which is summarized in Figure 4. As a short summary, all results are very positive.

Team members highly agree that raising a problem in a reflection meeting is better than having a decision made by the team leaders alone (4.1 average, no one disagreed), and that they'll be glad if reflections will be used in their next team (4.2 average, no one disagreed). No one agreed with the statement "I don't understand at all the purpose of reflection" (1.5 average). It seems that regardless of eventual outcome, people are much more satisfied when they take part of the decision process, and their opinions are seriously heard.

As the team managers testify, this social effect also greatly eases making difficult behavioral changes in the team. Introducing such changes by presenting the problem being addressed in a reflection meeting reduces their perception as 'dictated by management', and substantially reduces resistance to change. In contrast, changes that were forced on the team from above encountered great resistance, which resulted in reduced success rates. Presenting the team with a problem rather than a solution is the key factor success here.

Regarding metrics, team members state that they will collect and track metrics when they become team leaders (3.9 average). The most common comment people wrote next to this question was that they may not use the exact same set of metrics analyzed in this project. This was commented both by people who agreed to this statement and those who didn't, and reflects the fact that not all metrics had a (perceived) equal contribution to ongoing development.

## 6. Discussion and Comparison

The previous sections presented the team's practice and reflections about this practice, regarding the Iteration Summary Meeting, the reflection technique, the reflection's goals and its effectiveness. This section discusses three additional aspects of the proposed practice, required to complete its presentation as a continuous process improvement framework: How to verify that decisions are implemented, how much overhead the process requires, and how it evolves over a long period of time. The results from the researched project are compared to previous publications.

### 6.1. Verifying Implementation of Decisions

A systematic method to verify that decisions are carried out is required in any process improvement scheme. Reflection is no exception, even though many of the decisions are implemented naturally since they are decided by the team, and their rationale is well known to team members.

Salo [16] proposes such a systematic method, in which the action items resulting from a post-iteration workshop are grouped in a table with the following columns: Finding, Action Point, Actor, Validation Plan and Validation. The first four columns describe the problem, what the action to correct it will be, who is responsible for overseeing it, and how the results will be validated. The fifth column is filled during the following post-iteration workshop, when the table is reviewed to verify that all actions were taken, and that the desired results were achieved.

Such a method is a major improvement over having no method at all, but it does not handle one central issue: Most action items resulting from reflection meetings are changes to continuous activities. For

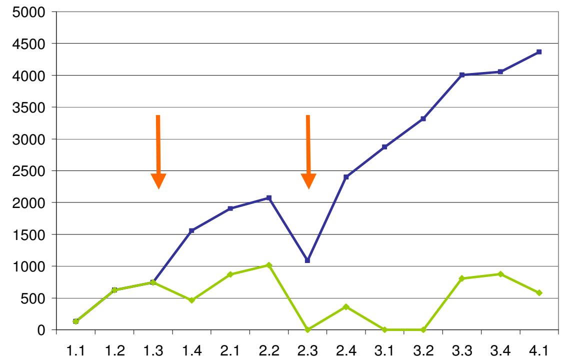


**Figure 5.** Product Size per iteration

🟦 by Entire Team 🟩 by Professional Testers

example, action items such as "stand-up meetings should be limited to ten minutes, and not evolve into technical discussions", or "task durations in a planning game should be between 5 and 25 hours" should be applied continuously. Reviewing them only in the first iteration after they were decided does not ensure their continued application. Action items that are one-time tasks – for example "upgrade version of night build tool" – are covered by this method, but this is redundant: In the researched project, such tasks were simply added as development tasks in the planning game, and were thus tracked (including estimated versus actual time to completion) like any other task.

The most effective way to validate that a given problem is solved and remains solved is to track it using the project's metrics. Figure 5, replicated from [22], illustrates this approach using the following story.

The project started with one professional tester on the team, who took responsibility for the entire acceptance testing effort in the first three iterations. However, this came at the cost of overtime, and by the third iteration it was clear that one tester cannot deal with the product's growth rate. Therefore, the reflection meeting of iteration 1.3 (First release, third iteration) was devoted to this subject, and it was decided that all team members will begin writing and running acceptance tests, and will be trained to do so. The team also decided to start measuring the product size metric generated by its professional tester(s) and by the entire team (including the testers) separately. As Figure 5 indicates, what followed is a rapid increase in product size, while the tester's contribution to it remained at the same level for the next three iterations.

The issue resurfaced again in iteration 2.3, in which the tester was transferred in the middle of the iteration to another project which had an emergency. This was the only iteration ever in which the team failed to increase the previous iteration's product, since not all regression tests were run. As the metrics indicate, only the tester's contribution was missing in that iteration – the rest of the team contributed as before. This was once again the topic of that iteration's reflection, in which the team discussed how other developers can take over the tester's test suites, and prevent such an occasion from recurring in the future. As the metrics show, the problem was solved in the next iteration, and growth in product size was stable from then on, even though testers' contribution often fluctuated.

Note that not every action item can be tracked by metrics, since the complete set of metrics must remain small – otherwise the collection and analysis of all metrics every iteration would be impractical. However, tracking metrics seems like the only successful formal practice that can be effectively used to monitor a problem or a project risk over a long period of time.

## 6.2. Overhead of the Process

All previous works on the subject of reflection relate to its overhead, since this is often an obstacle to its practical use. For example, investing half a day every iteration on reflecting equals to 5% of the entire team's time (four hours out of every two forty-hour weeks). A high fixed overhead may outweigh the benefits of reflection, as effective as it may be.

Dingsøyr et al. [5] report that post-mortem reviews require an average of 4.7% of the team's time, or 1.4 hours per person in absolute terms (calculated from their data). Cockburn [4] suggests that post-iteration workshops should take a minimum of two to four hours. Salo et al. [15] report an average required effort of 3.7% or 1.6 hours per person; this number changes significantly during the four iterations reported there. However, the post-iteration workshops as practiced in [15] considered all negative issues raised by team members, and (for research purposes) did not prioritize them as commonly recommended.

Our proposed Iteration Summary Meeting practice requires 1.75 hours per person per iteration, which is 2.1% of the team's total time (a forty-hour workweek was used). The reflection element by itself requires at most one hour (1.25%), and this is in fact the number that can be compared to the other methods, since all of them are based on reflection alone. The reduced overhead in our reflection technique stems from two facts: That the subject is chosen in advance by the moderator, and that the meeting has a strict one-hour limit, which helps in focusing the discussion. As the previous sections show, these issues do not stand in the way of making our reflection practice highly effective and satisfactory to the team. In addition, we recommend that future teams adopt the entire Iteration Summary

**Table 1**. Properties of Reflection Meetings Over Time

| Month | *Jan* | | | *March* | | *April* | | *May* | | *June* | | *July* | | August | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Iteration | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Emergency? | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Technical? | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |

Meeting practice, and not only the reflection element, since as the results show, it is highly effective and still requires lower overhead than other proposed practices (which only contain the reflective component).

### 6.3. Evolution of Reflections Along Time

In [14] Salo reports that the duration of the post-iteration workshop in two studied projects is high in the first iteration (5.5% and 7.8% of its total time), but decreases rapidly, and is halved in both projects between the 2nd and 4th iterations. Similar results are reported in [15]. Since all projects analyzed in this paper had four iterations and were six weeks long, it seems reasonable to conclude that an XP team's process is expected to stabilize roughly within six weeks. Kähkönen [10] also suggests that once an agile team stabilizes, the time between reflection meetings can be increased, and the need for an external coach may be reconsidered as well.

Since this paper is the first to report empirical data from a long-term, industry-based XP project, we can provide new data regarding these claims. The first and most obvious finding is that the need for reflection meetings does not decrease over time – there was no lack of subjects at any time during the four releases investigated in this paper. In addition, note that the questionnaires on which our effectiveness and satisfaction are based were done several months after these four releases, so if reflections would lose their perceived value as the development progressed, this would have been revealed in the results.

What does happen is that reflections evolve and change their role over time, from a stabilization role to a continuous improvement role. To investigate this, we asked the team leader to state whether each of the fifteen reflections this paper investigates was an 'emergency' reflection – meaning that it dealt with a problem that seriously endangered the team's ability to reach the next iteration's goals – and whether it dealt with a technical or an organizational issue.

The results are presented in Table 1. As they show, the first three months of development, and to a lesser extent the first five months, consisted mostly of 'Emergency reflections'. Reflections had a key role in stabilizing the team's development process in this period.

In addition, most team members were new to XP, and another key role of reflection was to communicate the rationale behind key practices used in the team, and to enable everyone to speak their mind.

After five months of development, the role of reflection evolved to be a continuous improvement forum. Some reflections were triggers by challenges that were not emergencies, but that the team hasn't faced before, such as how to accept four new team members who joined the team at once, or how to best split the team when it became too big. Other reflection meetings at this stage of development were pure improvement offers, and discussed pair programming, the defect management process and so forth.

Technical issues were rarely discussed in reflection meetings: With the exception of the first reflection (about the project's immature integration environment), all emergency reflections were about organizational and people-focused issues. Two additional technically-oriented reflections were improvement offers (a new unit testing utility, and a review of common coding mistakes) were done, but were not perceived by the team as highly successful. The team did have plenty of technical challenges to deal with, but it seems generally agreed that reflections were not the best forum to do so. Many team members have little to contribute on such technical problems (testers, business analysts and novice programmers), and often feel their time wasted in such reflections. In addition, the adoption of technical solutions usually does not face a social resistance to change, so the reflection's counter-effect to this is redundant here.

Based on these findings, we recommend that the Iteration Summary Meeting practice and reflections in particular be conducted in every iteration in long projects as well. They should also be focused on organizational issues: Technical problems seem to be best tackled by a small group of 2-3 top developers, who can later present their solution to the team.

## 7. Conclusions

This paper analyses the reflection practices of an agile team in a large-scale, long-term software project in an industry setting. The results suggest that the proposed reflective practice is highly valuable for this kind of projects. Reflection is perceived as effective in stabilizing a new agile project, fostering continuous improvement, and resolving team conflicts. In addition, the satisfaction of team members increases.

The proposed Iteration Summary Meeting practice extends current approaches to include the customer's summary and formal presentation of the system – which were assessed as more important than the reflection itself – as well as the review of metrics, which enables ongoing tracking of high-risk issues and past decisions. The proposed practice achieves these additional goals even though it incurs a lower overhead than current alternative practices. Concrete guidelines for implementing the proposed practice are provided.

## References

[1] Avison, D.; Lau, F.; Myers, M. and Neilsen, P.A., "Action Research". *Communications of the ACM*, vol. 42, 1999, pp. 94-97.

[2] Beck, K., *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000 ($2^{nd}$ Edition in 2005 – with Andres).

[3] Beck, K. et al., "Principles Behind the Agile Menifesto". Available on World Wide Web at: http://agilemanifesto.org/principles.htm, 2001.

[4] Cockburn, A., *Agile Software Development*, Addison-Wesley, 2001.

[5] Dingsøyr, T. and Hanssen, G. K., "Extending Agile Methods: Postmortem Reviews as Extended Feedback". In *4th Intl. Workshop on Advances in Learning Software Organizations (*LSO 2002), LNCS 2640, Springer 2003, pp. 4-12.

[6] Dubinsky, Y.; Talby, D.; Hazzan, O. and Keren, A., "Agile Metrics at the Israeli Air Force", Proceedings of *Agile 2005*, Colorado, 2005.

[7] Hazzan, O., "*The reflective practitioner perspective in software engineering education*", *The Journal of Systems and Software 63(3)*, 2002, pp. 161-171.

[8] Hazzan, O. and Tomayko, J., "*The reflective practitioner perspective in eXtreme Programming*". Proceedings of *XP Agile Universe 2003*, New Orleans, Louisiana, USA, 2003, pp. 51-61.

[9] Hazzan, O. and Tomayko, J., "The reflective practitioner perspective in software engineering". Proceedings of *CHI 2004 Workshop on Designing for Reflective Practitioners*, ISR Technical Report #UCI-ISR-04-2, 2004, pp. 75-78.

[10] Kähkönen, T., "Life Cycle Model for Software Process Improvement Project Deploying an Agile Method". In *ICAM 2005 – Intl. Conf. on Agility*, Helsinki, Finland, July 2005.

[11] Kerth, N. L., *Project Retrospectives: A Handbook for Team Reviews*. Dorset House, 2001.

[12] Lamoreux, M., "Improving Agile Team Learning by Improving Team Reflections". Proceedings of *Agile 2005*, Colorado, 2005.

[13] Myllyaho, M., Salo, O.; Kääriäinen, J.; Hyysalo, J. and Koskela, J., "Analysis of Small and Large Post-mortem Review Methods". Proceedings of ICSSEA *2004: 17th Intl. Conf. on Software & Systems Engineering and their Applications*, Paris, France, December 2004.

[14] Salo, O., "Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies". In *EUROMICRO 2004*. IEEE Computer Society Press, Rennes, France, 2004.

[15] Salo, O.; Kolehmainen, K.; Kyllönen, P.; Löthman, J.; Salmijärvi, S. and Abrahamsson, P., "Self-Adaptability of Agile Software Processes: A Case Study on Post-iteration Workshops". Proceedings of *XP 2004*, Germany, 2004, pp. 184-193.

[16] Salo, O., "Systematical Validation of Learning in Agile Software Development Environment". *7th International Workshop on Learning Software Organizations (LSO 2005)*, Germany, April 2005.

[17] Schön, D. A., *The Reflective Practitioner*, BasicBooks, 1983.

[18] Schön, D. A., *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession*. Jossey-Bass, San Francisco, 1987.

[19] Schön, D. A. interviewed by John Bennent. "Reflective conversation with materials". Terry Winograd, *Bringing Design to Software*, Addison-Wesley, 1996, pp. 171-184.

[20] Scupin, R., "The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology". *Human Organization*, vol. 56, 1997, pp. 233-237.

[21] Susman, G. I. and Evered, R. D., "An assessment of the Scientific Merits of Action Research". *Administrative Science Quarterly*, vol. 23, 1978, pp. 582-603.

[22] Talby, D.; Hazzan, O.; Dubinsky, Y. and Keren, A., "Agile Software Testing in a Large-Scale Project". *To appear in IEEE Software, Special issue on Software Testing - Jul/Aug 2006.*