The Reflective Practitioner Perspective in Software Engineering

Position Paper
CHI 2004 One Day Workshop - Designing for Reflective Practitioners

Orit Hazzan[1] and Jim Tomayko[2]

[1]Department of Education in Technology and Science, Technion - IIT, Haifa 32000, Israel
oritha@tx.technion.ac.il

[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
jet@cs.cmu.edu

This position paper is based on our articles:

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

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

1. Introduction

This position paper focuses on the application of the Reflective Practitioner (RP) perspective to the profession of Software Engineering (SE). The RP perspective, introduced by Donald Schön (1983, 1987), guides professional people (architects, managers, musicians and others) to rethink and examine their professional creations during and after the accomplishment of the creation process. The working assumption is that such a reflection improves the proficiency and performance within such professions. Analysis of the field of SE and of the kind of work that software engineers usually accomplish, supports the adoption of the RP perspective to SE.

Specifically, this position paper focuses on the construction of ladders of reflection that may serve as a means that supports one's thinking in terms of different levels of abstraction. Indeed, one message that is conveyed in this position paper is that the transition between levels of abstraction is an important skill for software developers. It is proposed that developers' experience in the construction of ladders of reflection may improve their performance in the process of software development.

2. Software Engineering as Reflective Practice

The importance of reflection as a habit-of-mind in the context of SE is derived mainly from two factors: The first factor is the complexity involved in developing software systems, regardless of whether one examines this complexity from an engineering, social or cognitive point of view; the second factor is the crucial role of communication among teammates for the success of developing a software system. The first factor suggests that one must improve one’s understanding about one’s own mental processes. One way to do this is by adopting a reflective mode of thinking. The second factor implies that in order to improve the communication within software development teams, one must increase one’s awareness of one’s own mental processes as well as of the mental processes of others.

3. Ladders of Reflection

This position paper suggests adopting the reflective practice perspective as a cognitive tool which may help software engineers in developing software systems. For this aim we propose to use what Schön (1987) describes by the term ladder of reflection:

> *We can […] introduce another dimension of analysis* [for the chain of reciprocal actions and reflections that make up the dialogue of student and coach]. *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.* (Schön, 1987, p.114)

The ladder of reflection described in Schön's quote refers to the architecture studio in which a coach guides his or her students. In what follows we attempt to explore how such a ladder of reflection may be associated with software engineering processes. Section 4 presents two ladders of reflection.

4. The Reflective Practice Perspective in Extreme Programming

It seems that an RP approach fits very well to Extreme Programming (XP) (Beck, 2000), 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, reflection is not outlined in the XP practices themselves.

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. In the description that follows, readers' familiarity with the XP practices is not necessary.

Tables 1 and 2 present two ladders of reflection[1]. The first one (Table 1) presents a ladder of reflection that is constructed during a pair programming[2] session. The second ladder (Table 2) is constructed during a Planning Game session.[3] In the first case the idea is to illustrate how a pair of programmers may increase the abstraction level of their thinking when reflection is interwoven within the process of software development. In the second case we illustrate how the fact that the customer

---

[1] Tables 1 and 2 illustrate the participants' thinking/discourse at each reflection-rung.

[2] The practice of pair programming specifies that any piece of code should be written by two developers, each of whom has a different role: the one with the keyboard and the mouse thinks about the best way to implement a specific task; the other partner thinks more strategically. As the two individuals in the pair think at different levels of abstraction, the same task is thought about at two different levels of abstraction *at the same time.*

[3] The planning game defines a process in which the customer, together with the developers, defines his/her requirements and priorities. One of the significant advantages of the planning game is that both the customer and the entire team participate in it, and thus all know the development process. Furthermore, guidelines that lead to decisions with respect to a specific release or iteration are clear to all.

and the team define together the next release/iteration makes it possible to introduce a reflective mode of thinking.

Table 1: A ladder of reflection: The case of pair programming

Ladder rungs	***Pair dialogue***
Designing *[a process of reflection-in-action]*	A: Did we consider all the exceptions?
Description of designing *[it takes the form of description with: appreciations, advice, criticism, etc.]*	B: Good question. Let's think about the best way to search for exceptions. I'm trying to understand what to think about when I'm looking for potential exceptions.
Reflection on description of designing *[reflection on the meaning the other has constructed for a description he or she has given]*	A: I think that this is not such a simple task. I have never thought about such systematic ways to look for exceptions. OK. Let's give it some thought. [*Working on formulating a systematic way for finding exceptions*]
Reflection on reflection on description of designing *[the parties to the dialogue reflect on the dialogue itself]*	B: Now that we have developed a systematic way for finding exceptions, I think we must analyze these strategies and reflect on the path that led us to finding these guidelines. A: Yes, this may improve our ability to solve problems of a similar nature in the future.

Table 2: A ladder of reflection: A Planning Game session

Ladder rungs	**A conversation during a planning game session**
Designing *[a process of reflection-in-action]*	Customer: In fact, I want this feature to behave this way [*moves her hands to illustrate*]. Developer 1: Can you think about a similar feature you needed before?
Description of designing *[it takes the form of description with: appreciations, advice, criticism, etc.]*	Customer: What do you mean? Would you like me to think about a similar case in the past in which I wanted a similar feature? Interesting. I have never been asked to do something like this before. But yes, I can think about a situation in the past when we needed a new system for our inventory management. I wanted the application to have this feature and only when we received the system I realized that, in fact, what we need is something else, more … [*illustrates with her hands*]. Let's call it B. Wow! Does that mean that we should not have at all the feature I described before?
Reflection on description of designing *[reflection on the meaning the other has constructed for a description he or she has given]*	Developer 2: We do not know. We can check the two options. But, can you please recall, what, in the case you just mentioned, led you at the end to realize that what you need is B, and why you didn't (or couldn't) realize this before, I mean, before you got the system and started working with it. Customer: Truly, the problem was that we did not consider the full setting in which the system would work. I think that we should consider the same issue now, before I make the final decision. [*The customer and the developers think about the way the application will be used, focusing on the specific considerations that were neglected in the customer's previous experience. At the end they decided about a third option that should be applied for these specific circumstances.*]

Reflection on reflection on description of designing *[the parties to the dialogue reflect on the dialogue itself]*	Customer: It's amazing. I must trace with you the full path we went through together. [*The customer and developers dedicate the next 15 minutes for this purpose*]. Customer: I do not want even to think about the catastrophe that could have happened if you develop one of the first two options we talked about. I must learn the lesson. First of all, I'd like to apologize for my resistance to take part in the planning game. I must confess that only now I understand how I should manage the all business with the new application. Tracker: I think we also learnt something from this experience. First, we should not be afraid to ask our customers difficult questions and to insist on getting answers. Second, the specific circumstances you introduced us to may be useful in our future projects. Finally, we should remember that before making final decisions and moving on, sometimes it is worth checking whether we consider all options. I believe that eventually, even if we stay with the first option, this would not be considered a waste of time.

Looking at the various rows of Tables 1 and 2, one may find that the subjects of reflection on each rung are objects of different levels of abstraction: While detailed elements are the focus on the first rung, ways of thinking and heuristics are at the center of attention on the fourth rung. As can be observed, the participants improve their understanding throughout the scenarios described in Tables 1 and 2.

5. Conclusion

This position paper describes a framework for adopting an RP perspective in general and the construction of ladders of reflection in particular into software engineering processes. Specifically, we illustrate how the awareness to the potential contribution of ladders of reflection to software development processes may improve developers as well as customers' understanding of processes they are engaged in.

We propose to discuss with the workshop participants the following questions:

- How to identify situations in software engineering in which a reflective mode of thinking in general and a construction of ladders of reflection in particular may be suitable.
- How to assimilate a reflective mode of thinking into these situations for which it is identified that a reflective mode of thinking may contribute.
- How to educate software developers to efficiently construct ladders of reflections.

6. References

Beck, K.(2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Schön, D. A. (1983). *The Reflective Practitioner*. BasicBooks,

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