Agile Software Development: A Case Study in a Software Company

L. Ashkenazi*, A. Shmilovici**, I. Ben-Gal*

*Dept. of Industrial Engineering, Tel-Aviv University
**Dept. of Information Systems Engineering, Ben-Gurion University

In this case study we accompanied a software development project which was performed using the Agile SCRUM method for software development. Subjective parameters of the design and code quality were collected and analyzed and a comparison was performed to an equivalent non-Agile project. We analyzed the software development process from the following aspects: Initial Effort Estimation process compared to the Effort Estimation given for the project by the COCOMO-II model; Control and management process compared to the one suggested by the PERT/CPM and Critical Chain methods.

Results and conclusions: The COCOMO II model showed the smallest deviation in its effort estimation from the real required effort; The PERT/CPM method is not recommended as in most cases the real result was out of the PERT/CPM allowed variance.

1. Introduction and Goals

Software development is thought from a project management point of view: too many software development projects overrun their schedules, their budgets, and deliver a low quality product. Scrum is an iterative incremental process of software development that focuses on a subset of project management and requirements management. It is one of the new Agile methods for software development [1, 2]. In this case study we accompanied a software development project which was performed using the Scrum method. Subjective and objective parameters of the design and code quality were collected and analyzed and a comparison was performed to an equivalent non-agile project. We also analyzed the same software development project using other known project management methods:

- COCOMO II is a method for estimating the effort required for a software project [3].
- PERT/CPM - CPM (Critical Path Method) and PERT (Program Evaluation Review Technique) are project management techniques for scheduling the project activities.
- Critical Chain Project Management (CCPM) - is method of planning and managing projects that puts the main emphasis on the resources required to execute project tasks [4].

The purpose of this case study was to examine a real implementation of Scrum in a large scale software company and analyze its success rate and contribution to the project quality and customer satisfaction. In this case study we also examine the complimentary benefits of different project management methods in the context of a real implementation of SCRUM.

2. Methods

Data was collected from a software company that specializes in applications for telephone companies. The company decided that it wants to test the Scrum methodology for


software development in a project developing a “Call Completion service for the mobile operator customers” software. The first author, who worked at a different department in that company, decided to accompany the Scrum project from its start to its completion. She joined the weekly Scrum meetings, interviewed the project leader and developers, and collected the data recorded in the systems that accompany each project (e.g., project budgeting and control, and bug reports). Five persons were responsible for developing the product, a team leader and five developers. Following the scrum methodology, the project contained 5 semi-independent development phases, each phase lasting about six weeks. The authors interrogated the project leader before each project phase for the data required to implement the COCOMO II method for Effort Estimation (in man days); and the activities required for the completion of each phase, as required to implement the PERT/CPM and the CCPM methodologies. By the completion of each phase, the authors compared the numbers predicted in the design phase (time and effort) with the actual numbers (time and effort) required for the completion of that phase.

Subjective measures of design quality and code quality: the project team leader graded on a scale of 1-5, his subjective evaluation of the design quality and code quality for each one of the four developers involved in the different phases of the project.

Objective measures about the code quality: defects can be detected by anyone in the project cycle, naturally most defects exposed by the Quality Assurance department and reported back to the software development team. We collected the timeline of the number of opened defects, and the timeline of the number of closed defects (defects corrected). Furthermore, the defects were classified in four categories: {Critical, Major, Minor, Warning}. For comparison purposes, the same data was collected for an equivalent project that was conducted by the same project team using a non-Scrum methodology.

3. Results

Table 1 presents % deviation (actual/planned) in man days for three methods. As we can see, COCOMO II has the smallest mean absolute deviation while Scrum has the largest one. However, note that only COCOMO II is an effort estimation methodology. Scrum and PERT/CPM are methodologies for project planning and control. Furthermore, there was an unexpected problem in Phase3 (a key programmer had to take an unexpected leave-of absence from personal reasons) which delayed the completion of phase 3, without a significant increase in man-days.
Table 1: Comparing the methods by the deviation (in man days) from the design target

<table>
<thead>
<tr>
<th>% Deviation</th>
<th>COCOMO II</th>
<th>PERT/CPM</th>
<th>SCRUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>-4.00%</td>
<td>3.45%</td>
<td>17.93%</td>
</tr>
<tr>
<td>Phase 2</td>
<td>-1.70%</td>
<td>-25.00%</td>
<td>-2.38%</td>
</tr>
<tr>
<td>Phase 3</td>
<td>17.00%</td>
<td>24.00%</td>
<td>76.67%</td>
</tr>
<tr>
<td>Phase 4</td>
<td>-15.70%</td>
<td>-28.57%</td>
<td>2.94%</td>
</tr>
<tr>
<td>Phase 5</td>
<td>-3.40%</td>
<td>0.00%</td>
<td>21.43%</td>
</tr>
<tr>
<td>Mean Absolute Deviation</td>
<td>8.36%</td>
<td>16.20%</td>
<td>24.27%</td>
</tr>
</tbody>
</table>

For the Scrum project, the defects timeline (Figure 1) demonstrates control of the project's quality since the defects are closed and problems are being resolved almost at the same rate that they are being opened and detected. There is a backlog of defects only at the middle of the project which is resolved at its end. For the equivalent non-Scrum project, the timeline demonstrates a fairly constant backlog of defects which is not resolved at the project end (reported by the customer). Comparing the two projects by the severity of the defects shows us that though the total number of open defects is quite similar in both projects, in the Agile project most of the detected defects were of severity “Major”, while the non-Scrum project shows considerable defects of severity Minor and Warning. This can indicate that in the Agile project most of the defects detected are really of acute problem in the software capability and that are captured by the pre-written test cases (as required by the Scrum methodology) and that the testers understand the tested requirements. Minor defects are mostly problems that are not very acute to the customer, yet, they indicate that the development team did not fully understand the customers’ requirements and that the development phase was not totally customer oriented.

Figure 1: Timeline of open and closed defects for the Scrum project.
The design quality and code quality grades demonstrate an increasing trend for the less experienced developers while, being quite stable for the more experienced developers. This is quite expected by Scrum, that the less experienced developer will improve with experience.

4. Conclusions

The COCOMO II method is a relatively simple method for effort estimation that claims to err on the order of 7%. [7]. Our results validated the usefulness of this method.

Scrum proved to deliver its main premise: improved software quality: As Figure 1 indicates, there were no defects by the end of the project.

The PERT/CPM method had some benefits in the design of each project phase – detecting its critical path and paying attention to its progress. The CCPM had some advantage in allocating the resources to the different tasks. However, those methodologies do not seem to be necessary for Scrum, as Scrum methodology endorses the partitioning of software projects into semi-independent and short sub-projects, small teams of developers, and flexibility in the development. In such configuration, the project leader is very aware of the critical path and the resources even without the use of a formal methodology.

References