

## **ABSTRACT**

Kylmäkoski, Roope

RaPiD7: A Collaborative Method for the Planning Activities in Software Engineering - Industrial Experiments

ISBN 952-15-1517-1

ISSN 1459-2045

Keywords: human interaction, planning activities, documentation, inspections, quality assurance, workshop

Software engineering is not only about writing code. Software engineering requires different kinds of competences, among them interpersonal skills. This is evident when the activities aimed at planning software implementation are considered. Software engineering activities such as system analysis, architecture design and interface design require extensive human interaction. However, the support for human interaction is currently not sufficient to address the interpersonal nature of software engineering.

Furthermore, the traditional approach for the planning activities has not reached its goals. Creating understanding efficiently and sharing information is time consuming and prone to errors. Actors mostly work alone and quality assurance occurs late in the development process in the form of inspections.

To address the issues referred to above, a method called RaPiD7 (rapid production of documentation, 7 steps) was developed at Nokia. RaPiD7 describes how human interaction is planned in software projects and how documents are to be created in facilitated workshops.

The method is evaluated using the results from industrial experiments at Nokia Networks. In the presented use survey data RaPiD7 is shown to speed up the planning work, improve the quality of the results and to enable more efficient information sharing. Moreover, metrics from a case study show how RaPiD7 has reduced significantly the number of fatal findings in inspections.

To conclude, this study provides a method that amends the planning activities in software engineering. The method can be utilized systematically in software engineering projects and the experiences show evidence of the improvements.

## **PREFACE**

I started the journey with this study half-accidentally. Earlier I was more interested in object-oriented methods, but I could not avoid the issues related to this study in the software projects I was involved with. I saw how inspections did not serve their purpose and how documentation sometimes took months to get finished. Inspections were often specification meetings, people did not show up to the inspections or if they did, they often came unprepared. Personally, there was a lot of frustration even involved and I am sure I was not alone in my feelings.

This was all back in 1997-1998. To compensate the frustration, I came up with an improved model for specification work, which I later found to include six steps. These steps were not about steps in a workshop, but rather something that explained the whole process that is now covered in RaPiD7 layers. During the next year I sent the ideas forward in our quality organization and elaborated them further with my brother, senior quality manager, Riku Kylmäkoski.

In 2000, I became responsible for the software process development in the organization developing network management systems in Nokia. That time gave me an opportunity to try out my ideas in practice. Soon the first real drafts of the method were sketched and the name RaPiD7 saw the light of day. During 2000 the first teams were introduced to the method and the first trials took place.

The first real deployment project for the method started on year 2001. At this time quality manager Oula Heikkinen strengthened our team of two and helped us to develop RaPiD7 even further. Oula brought into RaPiD7 the idea of writing the actual documents in the workshops. We trained tens of change agents for the method, and the results from the individual cases we became aware of, were often really encouraging. Already back then the idea of writing my doctoral dissertation on RaPiD7 came up. This realized more concretely in Portland USA in ICSE 2001 in discussion about the method with Professor Kai Koskimies, my supervisor-to-be.

In 2002, we started a huge endeavor to take RaPiD7 into use in Nokia Networks with the support of Sari Baldauf, the then Executive President of Nokia Networks. In this scope there were several thousand potential users. This provided me with a good chance to get material for my study as well as develop the method even further. Naturally, this delayed writing the dissertation, but the experiences were well worth the delay. At this time, senior quality engineer Hanna Turunen, and senior quality engineer Elina Ahonen, came to the rescue, supporting our limited resources. The deployment project lasted two years and we ended up having change agents throughout our organization, in China, Hungary, Germany and Finland.

At ICSE 2003 I published my first paper on RaPiD7. There I also got to know Dr. Pekka Abrahamsson. He quickly accepted the idea of RaPiD7, and

he has since then been supporting me in the journey's later important steps. Meeting Pekka had a great influence in getting my research included in an ITEA labeled project called ITEA AGILE and this way disseminating the messages further and outside Nokia.

The last part of the journey has been putting together all the issues learned during these years. There are more experiences that I can write about and many of them are not necessarily fulfilling the standards needed for a dissertation. However, from my personal perspective these experiences and the fact the method is still in use are probably the greatest reward. I hope this dissertation provides enough information that others can take on similar journeys with lesser efforts. I do not believe RaPiD7 is a tool for every situation, but the relative simplicity of it combined with the possibilities, can provide significant help in software projects. This is what the dissertation sets out to prove.

## ACKNOWLEDGEMENTS

As the journey has lasted several years, I have met and worked with numerous people who have all made valuable contribution towards writing this dissertation. I would like to express my gratitude to some of these people.

In the University of Technology in Tampere, two people have supported me throughout my journey. My supervisor, professor Kai Koskimies has read several versions of my writings and helped me in shaping the final version of the dissertation. Professor Ilkka Haikala helped me in writing my first publication about RaPiD7 for ICSE 2003 and has supported me continuously.

In Nokia there are numerous colleagues to thank for making this journey possible. My brother, Riku Kylmäkoski, has supported me from the very beginning. Oula Heikkinen has been one of the key supporters as well. In fact, the first version of RaPiD7 was sketched with Riku and Oula in a pub called “Kahdet kasvot” on one March evening in the year 2000. Since then we have been developing the method, creating training materials and training people together for many years. Other people that have helped me in Nokia are Timo Takamäki, Katherine Rose, Hanna Turunen, Elina Ahonen, Pentti Kolari, Kaustabh Debbarmann, Eila Himanen, Tuomas Lamminpää, Harri Klemetti, Raija Oksanen, Juhani Törnqvist, Juhos Botond, Gabor Ponyi, Tuomo Kähkönen and Kari Känsälä.

Dr. Pekka Abrahamsson has been encouraging and helping me since I met him in ICSE 2003 in Portland USA. He is one of the persons to thank for making the writing of the dissertation possible as part of ITEA AGILE project. Furthermore, I would like to thank Ko Dooms from Philips, who had the interest to try out RaPiD7 in another environment outside Nokia and provided me further confidence in our results at Nokia. I would also like to thank the Nokia Foundation for providing me an award and in this way enabling writing the dissertation.

In my personal life there are numerous people I would like to thank as well. My without-question-better-half, Maiju Ojamies, has not only given the typical support that is expected, but she has challenged my research methods as well as for example, provided important information on communication studies. Additionally, I would like to thank Dr. Sanna Hilden, who has been a cornerstone in giving me belief in my skills as a researcher and giving concrete guidance on research work. Furthermore, my parents have supported me all the way through and I would also like to thank Ukko Kylmäkoski, for making me take enough breaks during writing the dissertation.

It's been a long journey, but certainly worth all the efforts.

Pirkkala 20.12.2005 Roope Kylmäkoski

## LIST OF FIGURES

Figure 1 Value added-analysis in BPI [Harrington 1991] (reproduced).....	1
Figure 2 Division to value adding and non-value software engineering .....	2
Figure 3 Objects of software product development (adapted from [Bevan 1999]) .....	9
Figure 4 Modeling the objects of software product development .....	10
Figure 5 The planning activities in software engineering.....	11
Figure 6 The traditional approach for the planning activities in software engineering.....	13
Figure 7 Information sharing in the traditional approach .....	15
Figure 8 The collaborative approach visualized .....	34
Figure 9 Efforts for individual responsibilities in the collaborative approach .....	41
Figure 10 Efforts for team support in the collaborative approach .....	41
Figure 11 The three layers of RaPiD7.....	49
Figure 12 Planning RaPiD7 workshops in projects .....	51
Figure 13 Creating a project plan in RaPiD7 workshops.....	54
Figure 14 RaPiD7 workshop invitation template.....	55
Figure 15 Identifying the workshop participants from the V-model .....	56
Figure 16 Steps of RaPiD7 [Kylmäkoski 2003] .....	58
Figure 17 Workshop agenda creation process .....	67
Figure 18 The relation of agenda creation process and RaPiD7 steps.....	68
Figure 19 ATAM phases and steps (reproduced from [Clements et. al 2002]).....	70
Figure 20 Results of the early assessment .....	83
Figure 21 Using the elements of RaPiD7.....	85
Figure 22 Calendar time efficiency.....	87
Figure 23 Team cooperation .....	88
Figure 24 Information sharing and communication.....	89
Figure 25 Common understanding.....	89
Figure 26 Commitment to project work.....	90
Figure 27 Time used inspections.....	91
Figure 28 Inspection efficiency.....	92
Figure 29 Defects found in inspections.....	92
Figure 30 Quality of documents.....	93
Figure 31 Efforts used in producing documents .....	94
Figure 32 Types of documents produced.....	95
Figure 33 Relevance for reviewing the documents.....	96
Figure 34 Challenges in using RaPiD7 .....	97
Figure 35 Division of efforts into specification and implementation and testing efforts .....	101
Figure 36 Division of efforts in specification work.....	102
Figure 37 Findings per hour.....	104
Figure 38 Fatal findings per inspection.....	104
Figure 39 Average number of findings per finding category (medium and neutral).....	105
Figure 40 Rework hours per finding .....	106
Figure 41 Rework hours per page .....	106
Figure 42 Total inspection efforts per finding .....	107

## LIST OF TABLES

Table 1 Action research cycles and research methods.....	6
Table 2 Different problem types ([Steiner 1972] reproduced) .....	26
Table 3 How JAD and AM conform to the key principles of the collaborative approach .....	39
Table 4 Goals and typical duration of RaPiD7 steps .....	60
Table 5 Pattern for requirements workshop .....	69
Table 6 Pattern for an initial architecture workshop.....	71
Table 7 Pattern for architecture workshop (components and interfaces) .....	71
Table 8 The project layer comparison.....	73
Table 9 The case layer comparison.....	75
Table 10 The workshop layer comparison .....	76
Table 11 Analysis of other elements .....	78

## TABLE OF CONTENTS

<b>PREFACE.....</b>	<b>II</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>IV</b>
<b>LIST OF FIGURES .....</b>	<b>V</b>
<b>LIST OF TABLES .....</b>	<b>VI</b>
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 MOTIVATION .....	1
1.2 CONTEXT OF THE STUDY.....	4
1.3 THE RESEARCH PROBLEM AND THE RESEARCH METHODS .....	5
1.4 CONTRIBUTIONS .....	6
1.5 ORGANIZATION OF THIS DISSERTATION.....	7
<b>2 THE PLANNING ACTIVITIES IN SOFTWARE ENGINEERING 9</b>	
2.1 “OBJECTS” AND MODELS OF SOFTWARE PRODUCT DEVELOPMENT .....	9
2.2 GOALS FOR THE PLANNING ACTIVITIES IN SOFTWARE ENGINEERING.....	11
2.3 TRADITIONAL APPROACH FOR THE PLANNING ACTIVITIES IN SOFTWARE ENGINEERING .....	12
2.3.1 <i>Definition of the traditional approach.....</i>	<i>12</i>
2.3.2 <i>How the traditional approach fails in reaching its goals?.....</i>	<i>13</i>
2.4 QUALITY ASSURANCE FOR THE TRADITIONAL APPROACH .....	15
2.4.1 <i>Definition of inspections .....</i>	<i>16</i>
2.4.2 <i>Problems in the inspection practices .....</i>	<i>16</i>
<b>3 GROUP PROCESSES IN SOFTWARE ENGINEERING .....</b>	<b>19</b>
3.1 TEAMS VERSUS GROUPS.....	19
3.2 TEAMWORK VERSUS GROUP WORK.....	20
3.3 ELEMENTARY AND STRUCTURAL ASPECTS OF GROUPS.....	21
3.3.1 <i>Becoming a member of a group .....</i>	<i>21</i>
3.3.2 <i>Group norms .....</i>	<i>21</i>
3.3.3 <i>How becoming a member of a group and norms relate to software development teams? .....</i>	<i>22</i>
3.3.4 <i>Roles in groups.....</i>	<i>22</i>
3.3.5 <i>How group roles relate to software development teams?.....</i>	<i>22</i>
3.4 SOCIAL INFLUENCE IN GROUPS .....	23
3.4.1 <i>Influence by majority.....</i>	<i>23</i>
3.4.2 <i>Influence by minority.....</i>	<i>23</i>
3.4.3 <i>The relation between social influence and software development teams .....</i>	<i>24</i>
3.5 PROBLEM TYPES IN SOFTWARE ENGINEERING .....	25
3.5.1 <i>Definition of problem types.....</i>	<i>25</i>
3.5.2 <i>Problem types in software engineering.....</i>	<i>25</i>
3.6 EFFICIENCY OF TEAMWORK .....	27
3.7 GROUP DECISION MAKING .....	29
3.7.1 <i>Group polarization and groupthink.....</i>	<i>30</i>
3.7.2 <i>How decision-making relates to software development teams? .....</i>	<i>30</i>
3.8 SUMMARY .....	31
<b>4 DEVELOPING A COLLABORATIVE APPROACH .....</b>	<b>33</b>
4.1 THE KEY PRINCIPLES OF THE COLLABORATIVE APPROACH.....	33
4.2 USING WORKSHOPS FOR REALIZING THE KEY PRINCIPLES .....	34

4.3	EXISTING RELATED APPROACHES .....	34
4.3.1	<i>JAD (Joint Application Development)</i> .....	35
4.3.2	<i>AM (Agile Modeling)</i> .....	36
4.3.3	<i>Comparison of the approaches</i> .....	38
4.3.4	<i>Summary</i> .....	39
4.4	COMPARING THE COLLABORATIVE APPROACH WITH THE TRADITIONAL APPROACH .	39
4.4.1	<i>The key differences between the approaches</i> .....	39
4.4.2	<i>Identifying the tasks of the approaches</i> .....	40
4.4.3	<i>Comparison of the approaches</i> .....	43
4.4.4	<i>Other issues</i> .....	46
4.4.5	<i>Conclusions</i> .....	46
4.5	APPLICABILITY OF THE COLLABORATIVE APPROACH .....	47
4.6	EXPECTED BENEFITS OF THE COLLABORATIVE APPROACH .....	47
<b>5</b>	<b>IMPLEMENTING THE COLLABORATIVE APPROACH IN NOKIA – THE RAPID7 METHOD.....</b>	<b>49</b>
5.1	LAYERS OF RAPID7.....	49
5.2	IMPLEMENTATION OF PROJECT LAYER.....	50
5.2.1	<i>Roles in the project layer</i> .....	51
5.2.2	<i>Key activities</i> .....	52
5.2.3	<i>Tools in the project layer</i> .....	52
5.2.4	<i>Best practices for the project layer</i> .....	52
5.3	IMPLEMENTATION OF THE CASE LAYER .....	53
5.3.1	<i>Roles and teams in the case layer</i> .....	54
5.3.2	<i>Key activities</i> .....	54
5.3.3	<i>Tools in the case layer</i> .....	55
5.3.4	<i>Best practices for the case layer</i> .....	56
5.4	IMPLEMENTATION OF WORKSHOP LAYER .....	58
5.4.1	<i>Roles and teams in the workshop layer</i> .....	60
5.4.2	<i>Key activities</i> .....	62
5.4.3	<i>Tools and techniques for the workshop layer</i> .....	62
5.4.4	<i>Best practices for workshop layer</i> .....	65
<b>6</b>	<b>COMPARING THE IMPLEMENTATIONS OF RELATED APPROACHES .....</b>	<b>73</b>
6.1	ELEMENTS OF THE PROJECT LAYER .....	73
6.2	ELEMENTS OF THE CASE LAYER .....	74
6.3	ELEMENTS OF THE WORKSHOP LAYER .....	76
6.4	OTHER ELEMENTS.....	77
6.5	SUMMARY OF THE COMPARISON.....	78
<b>7</b>	<b>EMPIRICAL EVALUATION - RAPID7 EXPERIENCES IN NOKIA .....</b>	<b>80</b>
7.1	THE ENVIRONMENT .....	80
7.2	EARLY DEPLOYMENT – UNIT OF 1000 PEOPLE.....	81
7.2.1	<i>What was measured and how?</i> .....	82
7.2.2	<i>Assessment results</i> .....	82
7.2.3	<i>Analysis of the results</i> .....	83
7.3	LARGE SCALE DEPLOYMENT .....	84
7.3.1	<i>What was measured and how?</i> .....	84
7.3.2	<i>Which elements of the method have been used?</i> .....	84
7.3.3	<i>Benefits perceived</i> .....	86
7.3.4	<i>The types of documents authored using RaPiD7</i> .....	94
7.3.5	<i>How are the inspections perceived?</i> .....	96
7.3.6	<i>Challenges in use</i> .....	96
7.4	METRICS FROM A SOFTWARE PROJECT .....	97

7.4.1	<i>Case description and environment</i> .....	97
7.4.2	<i>What was measured and how?</i> .....	98
7.4.3	<i>Status of the project</i> .....	100
7.4.4	<i>Calendar time and effort metrics for the second iteration</i> .....	101
7.4.5	<i>Workshop and document metrics for the second iteration</i> .....	101
7.4.6	<i>Elements of RaPiD7 used in the project</i> .....	102
7.4.7	<i>Results from the inspections</i> .....	103
7.4.8	<i>Analysis of the results</i> .....	107
7.5	CONCLUSIONS .....	108
<b>8</b>	<b>CONCLUSIONS</b> .....	<b>111</b>
8.1	REVIEWING THE RESEARCH PROBLEMS .....	111
8.2	REVIEWING THE CONTRIBUTIONS .....	111
8.3	LIMITATIONS OF THE STUDY .....	113
8.4	CONCLUDING REMARKS .....	115
	<b>REFERENCES</b> .....	<b>116</b>
	<b>APPENDIX</b> .....	<b>I</b>
	USE SURVEY QUESTIONS (LARGE SCALE DEPLOYMENT) USED IN THE STUDY .....	I

# 1 INTRODUCTION

## 1.1 Motivation

In engineering disciplines, immature in their nature, it is often required to test the results of earlier work in order to guarantee their quality (for example [Grunenwald 1989]). In software engineering testing and inspections [Fagan 1976] can be seen as a result of this. Indeed, it is commonly recommended to inspect the documents in software projects (for example [Sommerville 1996 p.484-488]). Moreover, inspections, for example, have been regarded in the literature as a key element in achieving document quality and productivity increases [Gilb and Graham 1993, p.24].

The cost of change, however, is often said to be dramatically higher later in the project development stream (for example [Jalote 1984 p.16]) indicating that early quality assurance is essential. On the other hand, there are claims that the cost of change could be made constant rather than linear or logarithmic using the right kind of approach for software development [Beck 2000 p.21-25]. In any case, without going to the argumentation of whether the cost of change is logarithmic, linear or even constant, there is always a cost for change [Vonderembse et al. 1996 p.82].

Business Process Improvement (hereafter BPI) (for example [Andersen 1999] or [Harrington 1991]) is one of the approaches that address this issue. BPI includes practices for process improvement work that aim at improving the quality of work. There are streamlining activities, for example, that go by the names bureaucracy elimination, redundancy elimination, value-added analysis and process cycle time reduction [Andersen 1999]. The value added analysis activity is presented in Figure 1.

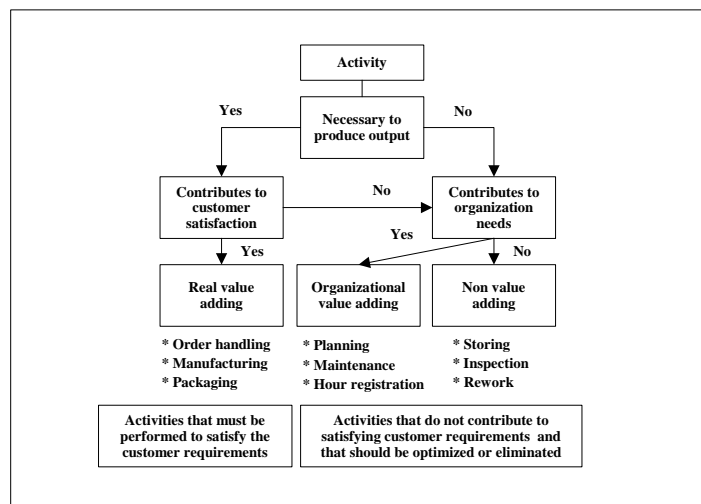


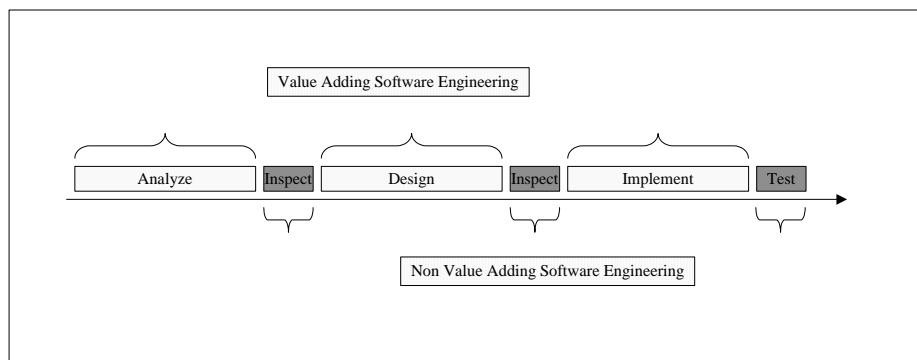
Figure 1 Value added-analysis in BPI [Harrington 1991] (reproduced)

Although the terms in the figure are not tailored for software engineering the elements of software development can be identified from the figure<sup>1</sup>. In fact, a similar approach to BPI is also applied in Lean Software Development [Poppendieck 2003] by discussing reducing waste from the process.

First, according to BPI, planning a software system can be seen as an organizational value-adding activity. Planning should be understood here as a wider concept than what is typical in software engineering. In other words, planning is not only about planning projects, but also about all activities that aim at defining the way implementation is carried out (to plan: to make preparation for something that is expected to happen [Oxford dictionary 1995 p.881]). Therefore, in this context planning is used as a common name for project planning, analysis and design activities.

Secondly, software implementation is clearly a real value-adding activity, and on the other hand the inspections can be seen as non value-adding activities according to BPI. Consequently, all activities before implementation should be either optimized or even eliminated. At least the focus should be directed at the organizational value-adding and real value adding activities that aim at planning or implementing the product rather than verifying that something has been done correctly. A simplified example of this division is illustrated in Figure 2. In the figure value-adding work covers the planning activities such as analysis and design and the implementation itself.

Hence, according to the above discussion, the quality assurance in the planning activities in software engineering should be addressed even earlier than in the inspections.



**Figure 2 Division to value adding and non-value software engineering**

<sup>1</sup> The analysis here should be seen independent and not be confused with Boehm's Value Based Software Engineering [Boehm and Jain 2005] despite the apparent similarity in their names.

Automation could be one possibility to optimize the process. However, despite all the efforts for automating software engineering, the goal of automation has not been reached to the level long desired and promised (for example [Gregory 1977]). Software engineering still requires substantial amount of individual efforts and in large-scale projects, tens or even hundreds of people are involved in the work. This results in substantial amounts of human interaction, too (for example [Cockburn 2002]). Hence, it is justified to say software engineering is craft and interpersonal activity. Logically, this would suggest that software engineering processes, methods, conventions, techniques and tools address these issues comprehensively. The following analyzes briefly the existing conventions, methods and processes when it comes to the help they provide for software engineers.

First, let us look at the support individuals carrying out various tasks in software engineering receive from the existing conventions, methods and processes. For analysis and design on different levels, whether it be architecture design or lower level design, numerous books are written in the field (for example [Fowler 1997], [Barclay and Savage 2004], [Bass et al. 1998], [Dennis et al. 2002] and [George et al. 2003] to mention a few). Similarly, there are numerous books written about implementation (for example [Stroustrup 2002], [Horstmann 1997] and [Schildt 2002]). Even with a very brief look, it appears that individuals carrying out different tasks in software engineering are getting aid in various forms from the literature.

Secondly, let us consider the support from the human interaction point of view. There are a few well-known existing practices described for collective activities within the software development teams that involve human interaction, for example, inspections [Gilb and Graham 1993], quality reviews [Sommerville 1996, p.616-619] and periodic status assessments [Royce 1998, p.133-134]. However, using the principles of value added analysis none of these practices are addressing value adding software engineering, but are rather ways to verify that something has been done correctly.

On the other hand, agile methods [Agile Manifesto 2004] have addressed the field of human interaction in recent years by providing approaches that value communication and individuals over process descriptions or tools. They provide methods for collective ownership of the code [Beck 2000], or focus on keeping the development team members in proximity to each other [Cockburn 2002 p.77-99], to mention a few practices. Furthermore, Team Software Process (TSP) described by Humphrey [Humphrey 1999] realizes that software development is a team-based activity. Individual approaches exist also to tackle the collaborative nature of software engineering (for example [Coughlan and Macredie 2002], [Wood and Silver 1995], [Macaulay 1999] and [Carmel et al. 1993]).

The lack of support for human interaction has been noticed in recent studies as well. Ochs and Van Solingen state as conclusions in their study on

meeting efficiency the following: “Software engineering is more than just writing code. It is a multi-disciplinary job, largely depending on peoples' social skills and social interaction. We emphasize the need for more thorough research on how software engineers interact, and how this interaction can be made more effective” [Ochs and Van Solingen 2004].

Apparently the need for having more support for interpersonal planning activities in software engineering has been recognized. Hence, as a conclusion, the lack of support for human interaction in software engineering should be addressed and the quality assurance in the planning activities should be focused in earlier phases than inspections.

## ***1.2 Context of the study***

This study emerges from the findings in Nokia’s software engineering processes that have roots in the considerations mentioned in the previous section. In fact, the findings were often related to inspections. People were not prepared well enough for the inspections or they did not even participate in them. In addition, the inspections often turned into specification meetings, because documents were either incomplete or people simply had opinions and knowledge that should have been taken into account when writing the documents. In number of cases these problems were tracked down to missing interaction between software designers. All this sometimes resulted in less than ideal compromises or in substantial rework, and hence delayed the projects. In some cases, the calendar time used to write a document was substantial, but the quality was not guaranteed even in these cases. Furthermore, people reported that they simply did not have enough time to study the work of others resulting in a lack of common understanding.

Before any study was put forward in Nokia, the problems were tackled by developing a collaborative approach during 1999-2000 to address the challenges in documentation work. The new approach developed is called RaPiD7. RaPiD7 name is an abbreviation of the words ‘rapid production of documentation, 7 steps’. Although the problems realizing in inspections were the initiative for developing RaPiD7, the aim was to understand the root causes of the problems and address these. The method was developed to enable better and earlier creation of understanding, to share the understanding effectively and to speed up all this work.

RaPiD7 was deployed by two consecutive projects at Nokia Networks during 2001-2004. Currently the method is used by hundreds of people in Nokia, and thousands are influenced by the use of the method. What is characteristic for the method is the environment where it is being used, in other words, in large-scale projects (hundreds of people) developing extremely complicated systems (several millions of lines of code). On the other hand, there are small independent teams utilizing the method as well.

The method has also been taken into use in other organizations recently, such as Philips and Hantro as part of a EU project called ITEA AGILE [ITEA AGILE 2005]. However, the results from these later developments outside Nokia are not under scrutiny in this dissertation.

### ***1.3 The research problem and the research methods***

The research problem of the study has its roots in the lack of support for human interaction in software engineering and in late quality assurance for the planning work. The research problem is presented in the following and further elaborated by opening up the research problem for three viewpoints.

**The research problem.** *How does emphasis on human interaction based methods improve the planning activities in software engineering?*

The three viewpoints for the research problem are:

- a) How can the process of creating understanding be improved in terms of speed and quality?
- b) How can the perceived quality of information sharing and communication be improved?
- c) How can the perceived quality of co-operation be improved?

**Research methods.** The study has been organized in a form of an *action research* [Lewin 1946] as the study has been part of an improvement project and a person belonging to the organization in question has carried out the study (for example [Avison 2002] or [Kock 2003]).

According to Susman and Evered the action research approach includes the following steps: *diagnosing*, *action planning*, *action taking*, *evaluating* and *specifying learning* [Susman and Evered 1978]. The *diagnosing* step means identifying or defining the problem. In this study the problem was defined as lack of support for human interaction and as late quality assurance for the planning work. The *action planning* phase covers the considerations for an alternative approach. For this study this means the collaborative method developed at Nokia for the planning activities. *Action taking* then means selecting the course of an action, and for this study it means deploying the method at Nokia. According to Susman and Evered, in action research *Evaluating* is used to study the consequences of the actions. In this study this refers to evaluating the deployment results. *Specifying learning* analyzes what was learned in the *evaluation* step. Here this refers to understanding how the method improves the planning activities in software engineering based on the results.

**Table 1 Action research cycles and research methods**

	Early deployment	Large scale deployment	
		Use survey	Detailed project study
Target	To get initial feedback from the users of the method about the possible benefits of the method	To see the benefits of the method based on users' evaluation	To measure the benefits of the method
Research method	Survey	Survey	Case study

The cycle defined by Susman and Evered was carried out in this study twice by two separate deployment projects. In these cycles the evaluation steps have been carried out by surveys and by a case study (see Table 1). The results of these are presented and analyzed in Chapter 7.

The role of the first early deployment project was to get initial feedback on the benefits of the method. The second deployment project covers more of the benefits of the method and is carried out in a larger unit. The surveys are used to capture the opinions of people [Hirsjärvi et al. 2000 p.184] that were using the method in Nokia. The results are compared against the expected benefits for the method. These results among discussions with the users have been used in developing the method further. However, the main driver has been in understanding how human interaction based methods improve the planning activities.

Finally, the role of the case study in the second deployment project was to analyze a software project using RaPiD7 in order to see how the benefits appear in a software project. The case study includes also experimental research elements in form of metrics to make the evaluation of the benefits explicit.

Neither the surveys nor the case study were addressing whether the method itself and its structure appears valid. Indeed, the aim of the study has been in evaluating the possible benefits of the method rather than validating the structure of the method itself. This approach was selected as the method was implemented in the industry to answer recognized problems in the planning activities.

#### **1.4 Contributions**

The contributions of this study are twofold and are summarized as follows. First, the study identifies problems in the planning activities in software engineering, provides a solution for the problems and implementation of the solution in form of a method. Second contribution is the evaluation of the method against the industrial experiments and showing evidence of the expected benefits of the method. The contributions are elaborated in the following.

**The method.** RaPiD7 addresses the lack of support for human interaction in software engineering. Although similar approaches exist the focus of these methods has not been in institutionalizing the practices into software engineering. RaPiD7 is a method intended to be integrated into software processes and projects. It can be applied in both large-scale and small-scale projects as a systematic approach to ensure the speed and quality of the planning activities in software engineering. As a conclusion, the contributions from the method point of view are:

- Addressing the lack of human interaction by providing a method that can be
  - institutionalized in the software processes and
  - used in both small and large scale.
- Providing an approach to enable continuous quality assurance for the planning activities in software engineering.
- Providing a way to speed up as well as improve the quality of the planning activities in software engineering.

**Evaluation of the method against the industrial experiments.** The study provides results to validate the expected benefits of the method and to give better understanding of the possibilities human interaction based methods provide for software engineering. To make explicit, the contribution from the industrial experiments is:

- Providing evaluation of industrial experiments that support the expected benefits of the RaPiD7 approach for the planning activities in software engineering.

### ***1.5 Organization of this dissertation***

The following presents briefly each of the chapters in the dissertation to help the reader in understanding how the dissertation is constructed, why certain issues are covered and finally to help in selective reading of this dissertation.

Chapter 1 (this chapter) introduces the whole study for the reader with a short discussion of the problems and theories in the background. The lack of support for interpersonal activities and the late quality assurance in the planning activities are mentioned as key problems. The research problems, research methods and contributions of the dissertation are presented.

Chapter 2 discusses more deeply the planning activities in software engineering. This is done in order to understand the goals of the planning activities and to understand the purpose the collaborative approach was developed for. The traditional approach for the planning activities is

presented, and the problems of the traditional approach for the planning activities are covered, too.

Chapter 3 presents group processes from the field of social psychology and analyzes their relation to software engineering and to software development teams. This chapter identifies possible challenges in human interaction based software methods related to group processes, and provides suggestions to overcome these risks. These are for example, related to the risks in decision-making in groups and to the efficiency of groups. However, no new theories, terms or results are presented from the field of social psychology.

Chapter 4 presents a collaborative approach for the planning activities in software engineering based on the discussions in the first three chapters of the dissertation. It also presents other similar approaches to support the collaborative approach introduced and also explains how they differ from each other. A comparison to the traditional approach for the planning activities is also covered.

Chapter 5 introduces the implementation of the collaborative approach realizing as RaPiD7 method at Nokia. The implementation of the method based on three layers is presented with key activities, roles and best practices identified by the users in Nokia.

Chapter 6 compares the implementations of the related approaches with RaPiD7. The comparison is done based on the structure of RaPiD7.

Chapter 7 presents the results gained from the industrial experiments in Nokia. The results are analyzed and discussed, and mapped to the expected benefits of the method. First, experiences from an early experiment are covered. Then experiences from a larger scale experiments are discussed and analyzed, and finally a detailed analysis of a software project is presented using selected metrics.

Chapter 8 concludes the study by explaining how the research problems are answered in form of the contributions of the study. The limitations of the study are also discussed, and finally implications, future work and concluding remarks are presented.

## 2 THE PLANNING ACTIVITIES IN SOFTWARE ENGINEERING

This chapter provides explanations why the lack of support for human interaction and late quality assurance realizes as problems in the planning activities. This chapter has its roots in [Kylmäkoski 2003] and develops the ideas in it further. This is done by elaborating on the origins of the study discussed in Chapter 1. First, a connection between software product development and the planning activities is developed and the goals for this work are identified. The traditional approach for the planning activities (presented also originally in [Kylmäkoski 2003]) is developed further and problems in it are discussed. Moreover, inspection practices are presented as part of the traditional approach and the limitations of these practices are discussed.

### 2.1 “Objects” and models of software product development

To better understand what are the planning activities in software engineering, it is important to understand the objects<sup>2</sup> of software product development. While the end goal in software product development is naturally the software itself, an important part of software product development is the process with its intermediate outputs, which contribute to achieving the end goal. What are these objects of software product development? A view of the different objects is presented in Figure 3 by adopting the approach in [Bevan 1999]. These objects are *context of use*, *external product view* and *internal product view*.

Context of use means the complete environment where the software product will be used. This includes other software systems, other non-software systems, people performing the tasks, the goals of the people and the social environment.

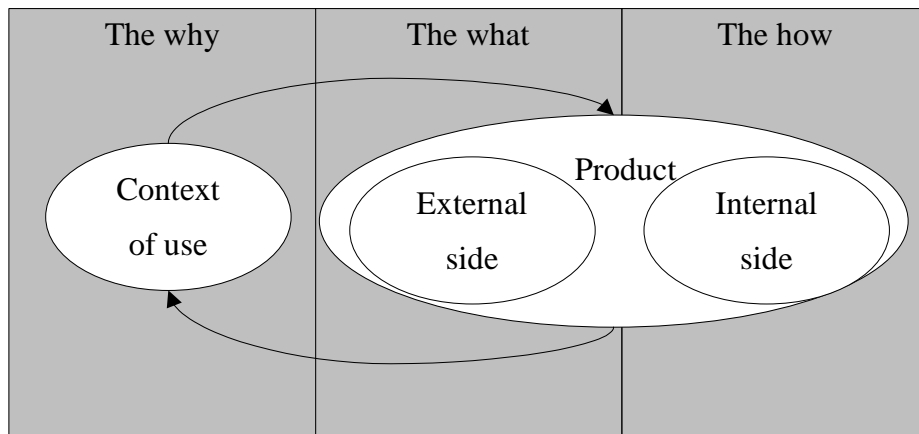


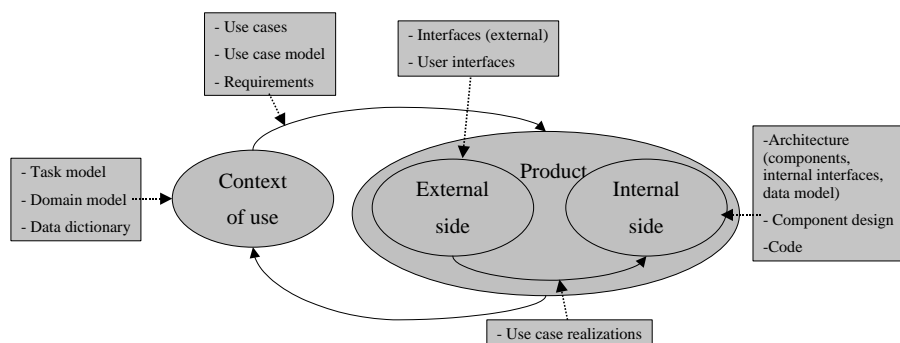
Figure 3 Objects of software product development (adapted from [Bevan 1999])

<sup>2</sup> Objects here do not refer to objects as they are used in object-oriented programming, but refer to the use of the word as the target of an action.

In short, the context of use tells us *why* we produce the software in the first place and hence how it should meet the user requirements. In practice, this means how the use of the software will be integrated with the other software or non-software systems in the user environment, and what the value of the software will be for its users as part of their big picture.

The external side of the product tells us what functionality the software will provide, and for example what the software will look like. It also tells us how the system relates to other systems. The internal side of the product tells us what software constructs will implement the functionality, in other words how one can do the actual coding. The code itself is part of the internal side of the product as well.

How are these different aspects or objects of software product development then approached? The needed understanding of the different objects can be visualized using modeling languages such as UML [UML 2005] and then stored into documents. Different software processes have their own views about how to model and what should be modeled (for example [Jacobson et. al 1999] or [Aalto et al. 1999 ]). An example of the different kinds of models that could be used in modeling the objects of software development is presented in Figure 4. The process or task model explains how the software integrates into the environment where it is being used, while the domain model [Scott 2002 p.22] explains the concepts used in the target environment. Use cases [Jacobson et. al 1992 p.128-132] explain through stories how the reasons for developing the software can be fulfilled with software functionality. The use cases are accompanied with requirement statements expressing aspects the use cases cannot cover (for example non-functional requirements [Aalto et al. 1999 p.9]).



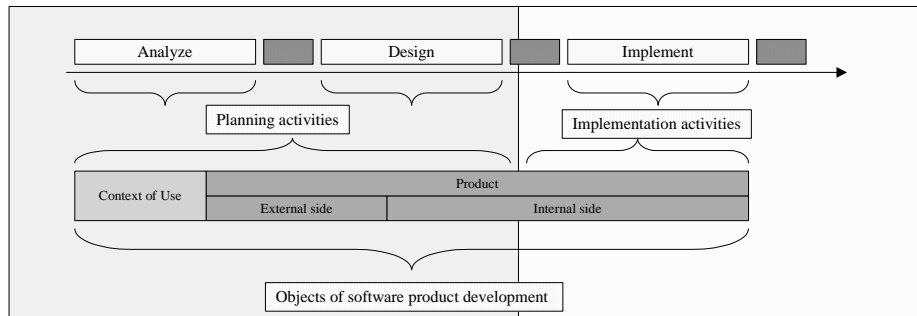
**Figure 4 Modeling the objects of software product development**

The external side of the product is described with user interface specifications and interface specifications towards other systems (for example [Jaaksi 1997]). The internal side of the product is modeled by realizing the software components from the use cases. The components will be further modeled and finally the coding can start.

To conclude, the planning activities in software engineering concern the different objects of software product development and use different models to depict the results of this work. The planning activities can be understood to cover also software project management related issues as well.

## 2.2 Goals for the planning activities in software engineering

To address the possible improvements for the planning activities in software engineering correctly, it is best to first identify the goals for this work. The relation between the planning activities in software engineering and the objects of software product development is illustrated in Figure 5.



**Figure 5 The planning activities in software engineering**

The objects of software product development help us in identifying the goals. First, the planning activities are carried out to be able to take the next steps in work [Van Deursen and Kuipers 1999 p.40], in other words, we need to understand the existing elements of a system and at the same time we need to create new understanding for the new yet undeveloped elements of a system. On the other hand, this information needs to be shared with the development team [Cockburn 2002 p.31] and stored to enable later use of the information. Hence, the goals for the planning activities in software engineering can be defined as follows:

1. creating understanding,
2. sharing understanding, and
3. storing the created understanding.

Taking a concrete example of planning software architecture, for example, these three goals could mean the following:

1. Creating understanding of the architecture for the system to be developed (what is the general architectural approach, what are the components of the system, interfaces between the components and so forth).
2. Sharing the knowledge about the architecture with all the related stakeholders.
3. Storing the decisions about the architecture somewhere (normally in an architecture specification document).

To verify that the goals are complete, let us consider different scenarios. Firstly, it is relatively easy to create some understanding, even quite quickly, but it is not necessarily at the same time easy to create flawless understanding with common agreement while also ensuring effective information sharing. Similarly, it is relatively easy to come up with, at least close to, flawless decisions given enough time and good sharing of information with all the stakeholders. However, obviously this approach does not guarantee quick results. This allows us to realize that the activity goals need to be accompanied with supporting quality attribute goals:

1. flawless decisions,
2. timely decisions, and
3. common agreement on decisions.

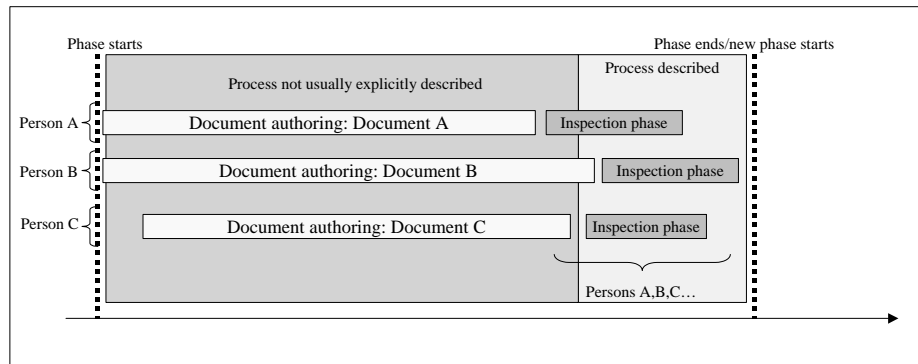
Despite the relative simplicity behind the goals, ensuring these goals are reached does not appear to be straightforward in software projects. In fact, there are almost always challenges in reaching each of the goals.

### ***2.3 Traditional approach for the planning activities in software engineering***

How then are the planning activities in software engineering traditionally carried out? The following elaborates on this by developing the concept of the “traditional approach” presented in [Kylmäkoski 2003] for planning activities in software engineering.

#### ***2.3.1 Definition of the traditional approach***

In the traditional approach creating understanding is mostly the task of a single individual. At an early point, the responsibilities are divided among the project members, for example, based on architecture structure. This is illustrated in Figure 6. Furthermore, information sharing takes place mostly by reading the documents of others albeit discussions and meetings take place on ad-hoc basis.



**Figure 6 The traditional approach for the planning activities in software engineering**

The individuals having their own responsibility area ask questions from others when needed, but this is not done according to any plan nor is it systematic. Although the process for creating understanding jointly and sharing the understanding is not described, the applied software process may describe the kind of documents required.

Therefore, the characteristics of the traditional approach are:

- Each single individual is usually responsible for document authoring and creating understanding of their own work within the project organization.
- Understanding of other's work is mostly achieved by reading the documents authored by others.
- The process for creating understanding and authoring the related documents is not a described process and is not repeatable.
- Quality assurance and acceptance of other's work is achieved by inspecting the documents.

Using these characteristics as the starting point, we can continue by looking at how they may realize as problems in the planning activities in software engineering.

### 2.3.2 *How the traditional approach fails in reaching its goals?*

Let us first consider why the traditional approach does not necessarily succeed in creating understanding. Sometimes the context of use is not known well enough, and this may result in problems when defining the external side of the product as well. At other times, even if the reasons for developing the software are known, there are challenges in implementation decisions due to new technologies or entering new domains without experiences of similar systems. A third challenge in creating the understanding correctly is the continuously changing environment be it from the context of use or internal product development point of view. Yet

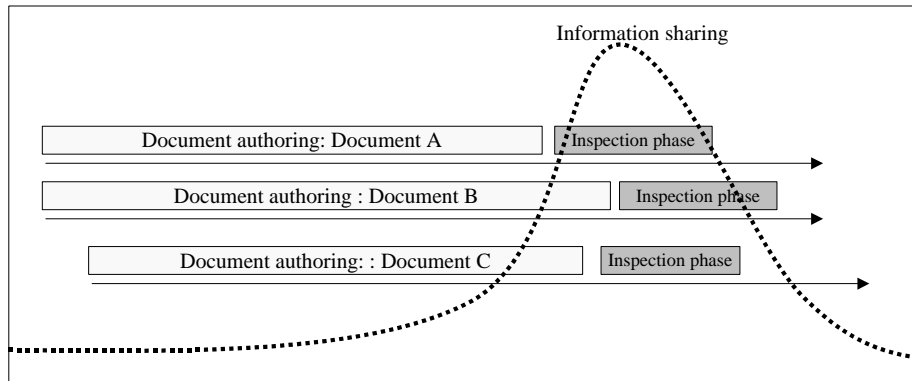
another challenge for creating proper understanding comes from not knowing what others know or what decisions they have made. [Curtis et. al 1988]

To elaborate and make explicit, the typical challenges in creating understanding are:

- Not having enough domain knowledge, which results in defective decisions and lengthens the time needed for the work.
- Not knowing the decisions, other team members make, which results in defective decisions and lengthens the time needed for the work. Reaching common agreement is also difficult.
- Not knowing if the created understanding is correct, which results in defective decisions and lengthens the time needed for the work.
- Not knowing when all the needed understanding is reached delaying the work unnecessarily.

What are the challenges in sharing the understanding then? The traditional approach is not enforcing and not always even explicitly encouraging human interaction. Information sharing happens often with delay, as the information possessed by others is typically only available when inspections take place. This is illustrated in Figure 7. Moreover, reading the documents of others is the usual method of information sharing. However, documentation is often regarded as an ineffective way to ensure communication. One of the reasons for communicational inefficiency of documentation is the lack of nonverbal clues [Redmond 1995 p.6], and thus documentation and computer mediated communication should not be regarded as the main form of communication [Hirokawa and Poole 1996 p.437] in the planning activities. In fact, emphasizing human interaction is, for example, one of the key messages in the agile approaches [Agile Manifesto 2004]. Furthermore, people often talk about documentation, and for example, design separately as if the two were unrelated.

Documentation is the task that makes sure the decisions are stored somewhere. However, the distinction between documentation and the related primary activity can also lead to the misunderstanding that documentation is a self-purposeful task that is needed without tasks such as architectural design, test planning and so on [Ambler 2002 p.90]. This happens when documentation is carried out only after the related work is already carried out. Of course, documentation can also be carried out for business reasons or simply to help to make explicit the brainwork needed in planning, and thus helping individuals in their own tasks [Ambler 2002 p.144-145]. In any case, documentation should not be considered the best nor the only way to ensure efficient planning activities in software engineering.



**Figure 7 Information sharing in the traditional approach**

As a conclusion, the typical challenges in information sharing in the traditional approach can be listed as follows:

- Information sharing is mostly dependent on reading documents of others and this requires substantial amount of reading in a short time. This is time consuming and common agreement on decisions is difficult to reach.
- The information sharing takes place briefly, late during the development process. Information is also often incomplete or imperfect because all of the relevant information holders do not or are unable to contribute. These result in defective decisions and lengthen the time needed for the work.

The third goal, storing the created understanding, is easier to reach, at least in principle. In fact, in the traditional approach the focus has often been on writing the documents, in other words, storing the created understanding. However, if people fail in creating the understanding, storing it does not help too much and may actually even be harmful as the information sharing takes place by reading defect-injected documents.

To conclude, it appears that in the traditional approach the goals for the planning activities in software engineering are not reached.

#### **2.4 *Quality assurance for the traditional approach***

In the traditional approach the quality assurance for the planning activities is left to individuals. Therefore, to understand the quality assurance in the traditional approach, the verification activities need to be considered as well. In practice the quality assurance is addressed by the inspections. The inspections, their relation to the traditional approach and the challenges of inspections are covered in this section.

#### *2.4.1 Definition of inspections*

Manual defect detection activities go by various names such as Fagan inspection [Fagan 1976], inspections [Gilb and Graham 1993], and peer reviews (for example [Patton 2001]), for example. In general, all these practices refer to an approach where individuals other than the author of the work product try to identify possible defects in it. While there are differences in the approaches, in-depth analysis of the differences is not carried out here. Inspections are the general term used here for all of these different manual defect detection practices, which cover a phase where reviewers of the work product study the material, and a meeting to record the findings is held. Rework carried out to correct the found defects is not as such part of the inspections practice, but a natural consequence of it, and hence of interest here.

#### *2.4.2 Problems in the inspection practices*

Document authoring is almost solely dependent on the skills and competence of people. There is no automated tool to be used in inspections expressing whether a document is or is not ready [Parnas 2003]. This does not change from one approach to another, be it the traditional approach or something else. Verifying the quality of a document is simply up to people, and this verification takes place during inspections in the traditional approach.

Although inspections clearly help in finding defects in documents (for example [Gilb 1988 p.214-215]), there are problems in the inspections, too. The traditional approach enforces the involvement of the people, and thus the quality assurance, late in the process. In fact, the quality assurance takes place only when everything should be ready. This leads to realizing that some aspects of inspections are contradictory.

First, it is illogical that inspections should be focusing on major faults while the documents to be inspected should be ready without any major issues in them [Gilb and Graham 1993 p.63]. How can a document that is almost ready still have major findings in it? This is most likely because the document is not in fact “almost ready”, although the author believes it to be so.

Secondly, an inspection should only be focusing on logging the possible defects, not defining the possible answers for the findings [Gilb and Graham 1993 p.81-83]. If a person has been trying his or her best to come up with a faultless document, and nevertheless has not been successful in this, how is this person going to be able to solve the defects later? Is this simply because this person was told about the defects? If this was the case, the defects found could hardly be classified as major findings but rather as trivia. Consider, for example, the following case:

*An architect is writing an architectural plan and then sends the document for an inspection. Others then inspect the document, and the architecture is found to have a major problem in it. No discussion takes place about the nature of the problem in detail nor does any discussion take place about the possible improved solution, as this is not the intention of inspections. The architect simply corrects the major finding based on the inspection log.*

Should the solution be easy to recognize for the architect and at the same time be a major issue, the architect had probably realized the defect even before the inspection. On the other hand, if the defect indeed was a major one, it is very unlikely to have a simple solution that can just be swapped to replace the original solution. However, the case just described is not truly realistic. The following example describes the case in a more realistic form:

*An architect is writing an architectural plan and then sends the document for an inspection. Others then inspect the document, and the architecture is found to cause a lot of discussion. It appears the architect has not covered all the required aspects in the architecture. Discussion does not appear to conclude quickly enough and the moderator decides that a finding should be recorded; a finding about the very essence of the document. No solution or way to find a solution to the problem is defined. However, a follow-up inspection is agreed on with the same people.*

Someone defending inspections could say that the document was not ready for the inspection in the first place as it had a major fault in it. However, how was anyone to realize this before all the related people sat together for the first time? It is possible that the architect had discussed with all the relevant people about the architecture. However, the conversations would probably have been held separately, and although the architect thought he understood everyone's viewpoints, this was obviously not the case. Similarly, the people the architect was in contact with thought their ideas would end up in the document the way they had intended, which is also not always the case. Naturally, someone may suggest in the inspection meeting that a workshop should be organized about the matter. If this were a regular occurrence, would it not make sense to agree on such a workshop earlier in the development cycle and even make it a standard practice?

The help that inspection practices can bring to the aforementioned problems is limited. Inspections themselves do not help to get the documents produced any faster. In fact, they tend to make the process of authoring documentation last longer, although often for a proper reason. In addition, inspections neither help in forming common understanding or common agreement on the contents. As common understanding and agreement is missing, it is probable that commitment will be missing, too.

Based on the discussion and reasoning above, the key problems identified in the inspection practices were:

- There is a contradiction in the whole concept of inspections; documents entering inspections should be nearly ready, but major findings are expected.
- Inspection meetings do not provide help to solve the problems identified in them.
- The late involvement of people results in a poor ability to make decisions and to reach common understanding.

Based on the experiences in Nokia, other problems are found in inspection practices, too. These are more related to how well the inspection practices are carried out: First, people do not often prepare well enough or do not even attend inspections. They do not have the time or do not see the benefits of the inspections. Moreover, inspections tend to have negative nature, as they are for finding defects, and inspections may hinder document acceptance as people try to polish documents to perfection before inspections.

### **3 GROUP PROCESSES IN SOFTWARE ENGINEERING**

Software development is an interpersonal activity as recent approaches for software engineering have also recognized. Therefore, any improvement work especially on the joint efforts in software engineering cannot neglect the field studies about the behavior of individuals in teams and the behavior of teams as a whole. The efficiency of teamwork has caused a lot of discussion over time in both business environment and in the research. In business environment teamwork appears to be often seen in a positive light (for example in [Wood and Silver 1995 p.176] and [Chesla 2000]) whereas research provides more criticism about the efficiency of teamwork (for example, [Brown 1988], [Frey et al.1999], [Helkama et al. 1998], [Hewstone and Stroebe (ed) 1996]). In fact, the different studies even appear to conflict on the results, and the results seem to be dependent on the research methods used. Even the analyses of the same results appear to vary depending on the viewpoints.

This chapter presents related theories from social psychology (more precisely from group processes), and discusses how they relate to software engineering and the interpersonal nature of software engineering. The objective is not to cover all aspects from social psychology. Neither is the idea to cover all the studies and literature in the pertinent field. It has to be realized that the field studies and the literature in the domain of group process are diverse, and therefore it is not even possible to cover briefly all the possible interpretations of these studies. Hence, the aim is rather to cover the commonly identified and agreed aspects from group process that clearly relate to the joint work in software engineering and provide suggestions for software engineering practices based on the related findings. Few sources ([Brown 1988], [Hewstone and Stroebe (ed) 1996]) are used specifically for identifying the common aspects of group processes, although similar issues and results are also discussed in other sources (for example [Burr 1988], [Collier et al. 1991] and [Helkama et al. 1998]).

#### ***3.1 Teams versus groups***

Social psychology literature discusses often groups (for example in [Collier et al. 1991]) rather than teams. It appears that the term team is more often used in business environment (for example marketing team, business team, development team and so forth), and the term group is used in more generalized form. Groups can be defined in business environment in the following way: “Having profession quite often implies working with others. For these groups the common denominator is interaction and this interaction is assumed to give certain outcomes for each of the group members” [Hewstone and Stroebe (ed) 1996 p.440]. Another approach is to define when a person is acting as a member of a group. Three aspects can verify this [Brown 1988 p.5]. First, the presence or absence of two clearly identifiable social categories (for example, man and woman and worker and employer) needs to be identified. In other words, these two social categories

exist in groups. In practice, software developers that work in a certain company form a group from this perspective. Secondly, it needs to be identified if there is low or high variability between persons in their attitudes or behavior. In groups, the variability in attitudes is typically lower. The third aspect is the variability in person's attitude and behavior towards other group members, that is, does the person react differently to different members of the group? In a group, a person's behavior towards other group members is similar.

It is easy to see software development teams as groups using the first definition of groups by Hewstone and Stroebe. The definition in [Brown 1988] requires more analysis. First, the social categories can be identified rather easily in software development teams. The second aspect is more difficult to evaluate and actually results in understanding that some software development teams do not hold necessarily similar attitudes or even behavior making them actually less group like. Similar problems are of concern with the third aspect as well. However, it appears that at least ideally a typical software development team member should be able to be defined as belonging to a group.

### ***3.2 Teamwork versus group work***

If software development teams can be defined as groups, the next important question is how to define teamwork. Intuitively teamwork means activities such as different kinds of meetings or workshops. Studies organized about group efficiency also suggest that this is the case [Brown 1988]. In other words, these studies observe groups that have been working together temporally and physically, and have been trying to reach a certain goal by interacting with each other. Hence, the term teamwork can be defined using the following statements:

- The team is working towards a common goal with defined schedule.
- The team members interact with each other.
- The team is physically located in the same place (such as meeting room) or is possible to communicate with each other by technological solutions (such as voice conference tools).

However, social psychology often also refers to groups that do not conform to all the previous statements. These groups may only clearly conform to the first statement. These groups can be called nominal groups where the efforts of individuals are added together after a certain period of individual work. These summed up results of individuals form the nominal group. As the results of the studies present comparisons between nominal groups and real groups, the differences in the results between these different groups are also discussed in this chapter. This is especially interesting as nominal groups are closer to the way the planning activities in software engineering are

carried out in their traditional form, in other words, the writing of a document individually and then inspecting the documents.

### ***3.3 Elementary and structural aspects of groups***

Having some of the basic terminology defined, we can have a look at some elementary and structural aspects of groups. The elementary and structural aspects of groups can be seen as the foundations that will influence how the group dynamics can and will evolve, and on the other hand, these are also something the processes in a group will change.

#### ***3.3.1 Becoming a member of a group***

First, let us consider how becoming a member of a group affects the new member and the group itself. In this process there are a few related phenomena. First, the person becoming a member of a group will go through an influential process that will define how a person sees himself or herself, and this has consequences on the self-esteem of the person. On the other hand, the group will undergo a process when accepting a new member for a group influencing how the group and thus its individuals see themselves as well. A strong person will most likely have an effect on the group as a whole while someone else may be more influenced by the group than vice versa. In any case, the new members will most likely undergo some sort of “a ritual” before they can be accepted as part of the group. These rituals vary in different organizations and groups, and these rituals may be anything from unpleasant to warm welcome. [Brown 1988 p.20-29]

#### ***3.3.2 Group norms***

Groups form rules that everyone is expected to conform to without explicit notification. These rules are referred to as group norms. The group does not explicitly define the norms, but these are automatically formed in a group. (for example [Brown 1988 p.42-48] or [Hewstone and Stroebe (ed) 1996 p.478])

The norms are based on the commonly accepted behavior of individuals in a team, and hence the kind of people forming the group and the cultural backgrounds of people, have an influence on the norms. The norms are restricting the behavior of individuals, but on the other hand the norms give us the feeling of comfortableness because they give us the behavioral frame of reference [Burr 1988 p.42].

Furthermore, studies have shown that in more complex tasks it is not only crucial to agree on the goals of the group, but also on the way reaching the goals [Brown 1988 p.98]. This implies that in groups that work with complex issues, it is not enough to rely on the “automatically” defined norms, but to make these norms explicit.

### 3.3.3 *How becoming a member of a group and norms relate to software development teams?*

Becoming a member of a new group is something that will be faced frequently in an organization. As the process of becoming a member of a group influences the way a person sees himself or herself, this should be considered in the software development teams.

In a case when a new member joins an existing team, “the rituals” influence how welcomed a new person joining the team will feel. On the other hand, the norms that a team possesses should be made clear to the new member. The process of becoming a team member and the time of uncertainty will probably be shorter if the new member can adopt the team norms quickly.

Furthermore, while a new member of a team may hold new valuable ideas, the team may not accept these from a new member, unless this is a commonly accepted procedure.

### 3.3.4 *Roles in groups*

In addition to the norms, the group ranks its members to different hierarchical statuses. These statuses can be also referred to as *roles*. Again, this ranking happens automatically and is a result of various factors. The different roles, as the word ranking suggests, hold different value in the group. As the group members hold different statuses by their roles, they define the way we see ourselves in the group. Hence, the roles need to be clearly defined in order to keep people motivated. ([Brown 1988 p.51-52] and [Burr 1988 p.66-69])

The roles do not only define the way we see ourselves, but they may give us authority within group. The people that are leaders by nature may acquire authority by their behavior in the group. These people tend to be verbally talented, and gain the authority in the group by their verbal skills. [Brown 1988 p.52-59]

### 3.3.5 *How group roles relate to software development teams?*

An evident finding of structural aspects in groups in their relation to software development teams is that roles are clearly important. This is because the people in the groups are all affected by their views of their own identity in the different roles within the groups. In addition, people have expectations of the different roles. The team leader is expected to give directions for the team and an expert is expected to give answers for technical problems.

Roles are evident in the groups even if they are not explicitly defined and the different roles hold different values. Therefore, it is only natural that the motivation of people in software development teams is related to how well their roles are expressed. The expectations people have may be wrong

unless the roles are explicitly assigned. Consider, for example, if someone has been acting in the role of an expert in a previous project. People may automatically assign the same role for the person in the forthcoming project unless this issue is addressed. This indicates that even in a relatively familiar team the definition of roles of all participants should be addressed.

Furthermore, the roles should emphasize the knowledge people possess more than their organizational status. In fact, it may be a good idea to exclude all discussion related to the organizational roles if these are not the subject of the work. In teams where the members are familiar with one and other, there will evidently be knowledge of the organizational statuses and the previous approach may not help. In these situations, the team leader should try to empower all members of the team with equal influence irrespective of the organizational position.

### ***3.4 Social influence in groups***

People do not behave in groups the same way as they would individually. Social factors influence how people behave in the presence of others. At least two categories for social influence can be identified; influence by majority and influence by minority. These aspects can have an effect in software development teams, too.

#### *3.4.1 Influence by majority*

Studies show that people in groups tend to be swayed towards the opinions of the majority [Brown 1988 p.90]. Brown discusses studies that have shown that a coherent majority can affect the opinions of people even in the cases when individually the solution would be obvious to find out. It is said that people conform to majority and the larger the majority the easier we conform to it.

Why do people conform then? As was explained before, the group we belong to, defines something of how we see ourselves, therefore our opinions also reflect how the group sees us. It has been shown that before the communication in a group starts the opinions show greater variety than after the communication has begun in the group. Again, studies have shown that deviants from majority are often less liked than the ones agreeing with the majority. This alone explains greatly the way majority influences on our own opinions. It is often seen as more important to be socially accepted than to hold to our own opinions. In fact, the deviates are often neglected in groups or even expelled from the group. ([Brown 1988 91-106] and [Helkama et al. 1998 p.279-282])

#### *3.4.2 Influence by minority*

The possibility of the minority to influence in groups has been an interest of later studies (for example [Brown 1988 p.106] or [Hewstone and Stroebe (ed) 1996 p.498]). It was first thought that only the majority could have an

influence in a group. However, there is evidence in real life and in the studies showing that minority can influence, too. The way the minority influences is by being consistent, convincing and persistent. This way the minority can persuade some people to change their opinion. This sort of behavior may change the norms in the group finally changing the opinions of majority as well.

### *3.4.3 The relation between social influence and software development teams*

The influence the majority and minority may exert can be seen as a risk or as a positive aspect in software development teams. This depends on the case. These advantages and risks should be recognized and considered in planning the work.

In some teams a strong manager may be a risk. In other teams, a coherent majority favoring a new technology without proper reasoning, for example, may be the risk in form of influence by majority. On the other hand, a team can benefit from the cohesiveness the conformance offers. In other words, the project team may have better commitment to the decisions made. However, this scenario can also present risks. This is where the influence by minority can help. Someone may possess crucial information, for example, on a technical solution that the majority is overlooking. Whether the message of the minority gets through is highly dependent on the characteristics of the persons in the minority.

How can the influence of majority be made less risky? First, it would be beneficial to enable the team to express their opinions individually and silently first. With this approach, the ideas and opinions of individuals are expressed before they are influenced by the social factors. The influence of the majority can also take place later and therefore the team leader should take care that all the ideas presented earlier are analyzed carefully and equally against similar criteria.

The possible influence by minority is not necessarily that great risk in a software team, although a strong person with a great influence (high rank in status) may very well influence the majority and the whole group during the decision-making process. Consequently, this sort of behavior should be recognized by the team leader in order to prevent the opinions of one person overruling the decisions. However, it could also be that some team members are introverts or simply hold lower status in the group, but still possess innovative or crucial information. Again the team leader should identify the experts on each problem and ensure all the viewpoints are recognized adequately.

### **3.5 Problem types in software engineering**

Groups face different types of problems. The different problem types have an influence on various aspects of group work and team efficiency. Therefore, understanding the different problem types in general and the specific problem types in software engineering, provides basis for analyzing the influence of different problem types people face in the joint work in software engineering.

#### *3.5.1 Definition of problem types*

Steiner has defined the different problem types that a group can face and which group then needs to solve [Steiner 1972]. These problem types are frequently referred to in social psychology studies in the field of group processes (for example [Helkama et al. 1998]). These are listed in Table 2. In principle, there are three different questions to be asked to identify a problem type. These are *can the problem be divided into subtasks*, *does quality or quantity matter* and *what is the relation of individual efforts to the results of the team*. The original division by Steiner has been often adjusted according to needs, but the basic division usually remains rather intact.

#### *3.5.2 Problem types in software engineering*

The problem types put forward by Steiner are used here to be able to understand the nature of problems in software engineering and especially in software engineering activities. This is important in being able to understand the kind of problem types that appear in joint software engineering work.

The software engineering tasks are clearly divisible, that is, the work can be divided into subtasks by dividing the system into subsystems with defined interfaces. This relates to the specification of the system as well as to the implementation of the system. With implementation tasks, the divisibility is even more apparent. However, in the analysis of the system and in the very high-level technical specification of the system, the divisibility is not that clear. Certainly, subtasks can be identified even before dividing the system into subsystems. Certain people can concentrate on hardware constructs while others concentrate on software structures, for example. On the other hand, the evident connections these tasks have make them not so easy to partition. However, overall, the software engineering tasks generally appear to be divisible rather than non-divisible. After all, software engineering is not a tug-of-war kind of task where all the efforts are needed simultaneously.

**Table 2 Different problem types ([Steiner 1972] reproduced)**

<b>Question</b>	<b>Answer</b>	<b>Type</b>	<b>Examples</b>
Can the problem be divided into subtasks?	Yes	Divisible	Basketball, organizing a party
	No	Non-divisible	Tug-of-war
Which matters most, quantity or quality?	Quantity	Maximizing	Rowing-race, brainstorming
	Quality	Optimizing	Accounting, working on thesis
What is the relation of individual efforts to the results of the team?	Individual efforts are summed up	Additive	Tug-of-war
	The team selects from suggestions of individuals	Disjunctive	Mathematical problems, quizzes
	All members affect the results	Conjunctive	Mountaineering, relay race
	The team can decide how individual efforts affect team results	Discretionary	Voting the result, leader decides

Are software engineering tasks then maximizing or optimizing? It appears the answer depends very much on the situation. Sometimes it may be that new features are needed quickly to penetrate a market and the quality can be compromised. However, probably a more common situation is to select the features carefully so that they can be implemented in the given time with good quality. Moreover, software engineering, whether it be planning or implementation, is by nature more about getting issues right than maximizing the results in volume. Therefore, software engineering tasks can be said to be typically optimizing.

Furthermore, software engineering tasks like implementation are clearly conjunctive, that is, the software is a result of all the developers working on the whole system and the system is not complete until all the members have finished their work. On the other hand, when people are working on specifications the answer is not as simple. The work can be described as disjunctive, conjunctive and even discretionary, but perhaps not additive. If people are specifying certain parts of the system individually and the results are put together as the system specification, it can be concluded that these tasks are conjunctive. This is evident when software designers are specifying different parts of a system and the whole system does not “work” unless all the elements work, that is, the weakest developer defines the speed of the whole group. However, depending on the abstraction level at

which the system is looked at, this is not necessarily the case. In the case of a system that has different functions working independently the separate parts of the system can be ready even if the whole system is not. It can be argued that such a system does not meet its desired requirements and thus even this case could be understood as conjunctive. On other hand, when the team is working to define the overall architecture of the system or an architect is individually carrying out the work, the work can be defined as disjunctive, that is, the team selects the best possible solution from the candidates represented. The previous case can be discretionary, too. This can be the case, if the architect decides the result or the decision is voted somehow.

To conclude, typical software engineering tasks appear to have elements of almost all kinds of problem types. On the other hand, the problem types are presented at such a general level that probably almost any engineering discipline would cover all the different types when considered as whole. Therefore, the key is in understanding the kinds of problems that need to be faced in order to decide the best approach; should one use team working at all or what is the right approach when working in teams? It also has to be realized that even the organizational culture can define how a problem is solved despite its nature. In some organizations, the team selects from the suggestions of individuals and in other organizations the leader may decide.

### 3.6 *Efficiency of teamwork*

Social psychologists have been studying the efficiency of individuals compared to the efficiency of groups for some time. One of the approaches has been to first identify the potential the group members hold individually and then compare this to the actual group efficiency. Often it is said that the result of a team is more than the sum of the results of the individual members. However, the studies have shown that the groups are often not as efficient as the individual members and their potential would indicate (for example [Hewstone and Stroebe (ed) 1996 p.447-467]). For example, a famous study on a tug-of-war competition resulted in lower on average pulls for same people in groups than they were capable of individually [Brown 1988 p.125-130]. These results have lead to defining the possible reasons for the losses in potential that the groups are facing. Among others the following reasons have been identified:

1. **Loss of motivation.** In a group its member may feel that his or her efforts are not clearly visible, and hence this person may be “guilty” of *social loafing*. On the other hand, the members may be guilty of *free riding* which is a result of feeling that their efforts are not meaningful enough to the results [Helkama et al. 1998 p.258-259].
2. **Loss of coordination.** Evidently the group loses some of its potential in the process of putting the efforts together. In a tug-of-

war competition, someone may be pulling a little bit too early or may be stepping into the toes of someone and so forth.

These losses have led to the equation of group work efficiency ([Steiner 1972 p.9]):

$$\text{Group efficiency} = \text{potential} - \text{loss of motivation} - \text{loss of coordination}$$

By looking at this equation it appears clear that no group could ever exceed the potential of its individual members, and thus groups could never be more than the sum of its individuals. This would be the case in reality if there were only potential losses in the equation. However, there are also studies showing that process gains can be identified. Furthermore, the validity of how the studies that showed only process losses were performed has been challenged. Brown, for example, criticizes the studies for including only groups that have comprised solely of randomly selected people [Brown 1988 p.139-142]. This is not the typical case, for example, in business environment. Certainly, new teams are established occasionally, but other teams have been in place for years and teams are generally not put together randomly. These teams or groups benefit from what is called social labouring [Brown 1988 p.142]. They have refined their ways of working, the roles have been clarified, and thus the teams have been able to minimize the loss of coordination. Moreover, the loss of motivation is reduced by clarified role definition and feeling of responsibility by social bonding. In fact, it is claimed that the loss of motivation is not only reduced, but the motivation is actually improved in comparison to that of individuals. In addition, Brown criticizes some studies due to their limitations of involving limited, if any, group facilitation and claims that facilitation would also lead to improvements in the results of the groups [Brown 1988 p.140]. These effects, increased motivation and facilitation efficiency improvements are called process gains. Therefore, the original equation for group efficiency can be reformulated as:

$$\text{Group efficiency} = \text{potential} -/+ \text{loss or improvement of motivation} - \text{loss of coordination} + \text{gains by facilitation}$$

Overall, the results on the efficiency of group work are numerous, and it appears the results depend on who is analyzing group work. On one hand, the results appear to be very negative with respect to group working, and on the other hand, they appear to show possible gains compared to the individual approaches. Thus it appears probable, that both the process losses and gains exist, but their significance may very well be case dependent.

How do the studies of group efficiency then reflect on software development teams? In the light of the negative results, the use of joint work initially do not appear justified. Indeed, this is something to consider and leads to the realization that the use of teamwork is not always a definite improvement over individual approaches.

However, the problem types and their relation to the efficiency of teamwork should be considered when the applicability of joint work is analyzed. As was mentioned the approach in social psychology has often been to measure the efficiency of group work by looking at the combined strength of the group or number of ideas generated. These indicate that the problem types have either been non-divisible or divisible and maximizing. Other types of studies have been carried out as well, and since the software development tasks were recognized as often being divisible and optimizing, the results of these studies should be given the greatest value.

One concrete example of how the problem types play an important role is concretized in [Helkama et al. 1998 p.264-265]. The so called eureka tasks (optimizing and disjunctive and having one correct answer) may require the whole team's input in order to be able to come up with the correct result, as no one in the group can solve the problem alone. The group members can solve pieces of the total solution and in this way they may provide the total solution. In designing software, the tasks can rarely be defined as eureka tasks, in other words having only one correct solution (although these do exist, too), but the same logic applies for many design issues. The whole team can come up with a close to ideal solution, as the different members are able to solve different pieces of the problem.

In addition, in tasks that are more difficult, it has been shown that it is important for teams not only to identify the goals, but also the way of working [Brown 1988 p.98]. This speaks on behalf of structured approach for the joint work. Furthermore, the social groups are clearly more effective if the team is really involved in working together.

### ***3.7 Group decision making***

Group decision making has been addressed by several studies in the field of social psychology. The very early assumptions and results of the early studies in the 20<sup>th</sup> century stated that the teams make more conservative decisions than individuals (for example discussed in Brown 1988 p.142-143). Intuitively, this makes sense as teams can discuss various viewpoints and then select the best alternative with lesser risk to choose a defective solution. However, later studies showed that teams would actually make more risky decisions than individuals do (for example discussed in [Brown 1988 p.142]). Again, these results can be justified relatively easily. In groups no one has to take responsibility: "the team decided; not me", and on the other hand, no one needs to face accusations alone. Even these results have been challenged, in the more recent studies. These studies indicate that the groups decide on whatever the majority of the group initially favored [Helkama et al. 1998 p.283-284]. This phenomenon is known as group polarization (for example [Brown 1988 p.144] or [Hewstone and Stroebe (ed) 1996 p.513-515]).

### 3.7.1 *Group polarization and groupthink*

Group polarization is a realization of the dominance of majority's viewpoint and clearly has its roots in the influence by majority that was discussed earlier in this chapter. This again is not difficult to understand as people may easily feel the peer pressure of the majority. Contradicting with the majority could lead into being disliked by the majority and this is something people are rarely aiming for. In fact, the influence of the majority has been studied and the results indicate that even with an easily identifiable correct answer the majority would be able to influence the results of the minority [Brown 1988 p.142-147].

In its extreme manifestation, group polarization may lead to what is called groupthink. In groupthink, the group is cohesive and originally biased to a certain direction. Therefore, the group can come up with a decision following the patterns of group polarization that could be even easily proven wrong by someone outside the team. However, the cohesiveness of the team does not provide, enough chances for anyone in the team to intervene with the decisions and the decisions made this way can be catastrophic. There are a few well-known cases of groupthink, for example, the explosion of space shuttle Challenger in 1986 due to making the obviously wrong decision to launch the shuttle [Hewstone and Stroebe (ed) 1996].

Studies have been carried out to identify possibilities to avoid groupthink. Brown discusses these and lists five key elements that can potentially lead to groupthink that are identified in different studies [Brown 1988 p.158]. These are:

1. very cohesive group,
2. isolation from information outside the group,
3. no systematic search for alternatives in decision making,
4. stress caused by urgency to reach the decision, and
5. domination by a very directive leader.

Brown concludes that the domination of a very directive leader is the most important of these whereas cohesiveness is not necessarily a key factor. Nevertheless, this list provides an important checklist for any decision making body and hence for joint software engineering work, too.

### 3.7.2 *How decision-making relates to software development teams?*

Group polarization and groupthink may have a similar influence in software development teams as well. That is, the team originally being relatively biased easily comes up with a decision, and being satisfied with their efficiency, simply accepts the decision. How decision making in teams is

carried out can have an influence on how big a role group polarization, for example, can have in software development teams.

There are at least three ways to address these problems. The first relates to selecting the team. The team should not be selected so that it only contains members that are known to work well together at least when the reason behind this is that they appear to agree on everything. Secondly, the team should identify beforehand the decision-making criteria that the decisions should meet. It should be also ensured this criterion is met. Thirdly, the leader or facilitator of the team should be selected so that he or she is not a member of a potentially biased group and is therefore able to act as an independent referee.

### **3.8 Summary**

The first conclusion is that group processes do not provide any dramatic single finding in favor or oppose for joint work in software development teams. However, social psychology provides a number of observations that are either something that should be considered when working in teams or considered to identify when teamwork is not the best approach at all. The following lists the key findings put forward from the evaluation in this chapter:

1. The factors related to becoming a member of a group and to conforming to the group norms should be considered in the software development teams.
2. The roles are important in groups and thus in the software development teams, too. The roles in the software development teams and the different values that they hold are important elements of the interpersonal activities. Everyone in the team should be able to feel important due to their given role, although some roles by nature may have higher value in people's mind.
3. The possible negative influence of majority should be minimized in software development teams at least before all the options are understood.
4. The group process losses are evident in software development teams, and therefore one has to minimize the process losses and focus on activities that aim at producing process gains. These are:
  - a. adequate planning of activities minimizing the coordination and the motivation losses,
  - b. enabling motivation gains by enabling social groups to evolve,

- c. enabling motivation gains by using proper techniques for higher activation levels,
  - d. using facilitation, and
  - e. agreeing the ways of working.
5. Decision-making in software development teams could experience polarization and even groupthink. Therefore, the risks for groupthink should be minimized by:
- a. not enforcing decisions by the (organizational) manager of the group, but rather letting the group to make the decision,
  - b. ensuring, for example by effective idea gathering and idea analysis, that all of the different solution candidates are considered, and
  - c. by careful selection of the team reducing the risks of being separated from opinions outside the team, and hence having negative cohesiveness in the team.

Overall, although social psychology does not appear to provide unarguable facts, it provides an interesting and important viewpoint for teamwork-based software engineering approaches that the implementation of such should not overlook.

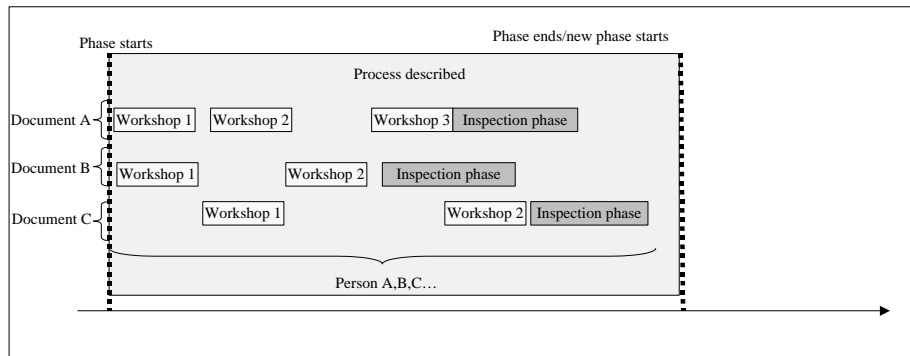
## **4 DEVELOPING A COLLABORATIVE APPROACH**

This chapter explains how the interpersonal nature of software engineering and the problems of the traditional approach can be addressed. This is done by developing key principles for a collaborative approach for the planning activities. The key principles are derived from the discussions in the first three chapters of the dissertation. These principles are then explained briefly in practical terms. However, the actual implementation of the collaborative approach is presented in Chapter 5. The essential existing related approaches are then covered to support the ideas presented here. This is followed by a comparison between the existing related approaches and the key principles of the collaborative approach. This is done to show the contribution of the collaborative approach presented here. The traditional approach for the planning activities and the collaborative approach are compared to identify their key differences. This provides better understanding on the possible benefits of the collaborative approach. The expected benefits are presented at the end of this chapter.

### ***4.1 The key principles of the collaborative approach***

Based on the discussions in Chapters 1 and 2, the collaborative approach should address the lack of support for human interaction and address human interaction and quality assurance earlier than in the inspections. In addition, the collaborative approach needs to address the goals for the planning activities in software engineering. The work should also be planned to minimize motivation and process losses. These issues give us the key principles of the collaborative approach:

1. Creating understanding is done principally as a joint activity in order to reduce the need for inspections.
2. Information sharing is done continuously in order to address the goal of information sharing of the planning activities.
3. Document writing is done as a joint activity as far as feasible in order to reduce the need for inspections.
4. Quality assurance in the planning activities in software engineering is addressed continuously by the project team in the project life cycle.
5. The joint activities are planned, as any other project activities are planned in order to reduce the process and motivation losses and in order to ensure common rules and defined roles for the joint work.



**Figure 8 The collaborative approach visualized**

#### **4.2 Using workshops for realizing the key principles**

Based on the above listed key principles, the collaborative approach can be seen as an approach for enabling planning the human interaction and joint creation of understanding in software projects. The joint creation of understanding has been often approached in software engineering in the form of facilitated workshops (for example [Macaulay 1999] and [Gottesdiener 2002]). Similarly the joint efforts in the collaborative approach presented here can be organized around workshops. The facilitated workshops can be used to define solutions and to come up with decisions for various issues in the planning activities. The approach based on the key principles and workshops is illustrated in a simplified form in Figure 8 (see Figure 6, page 13, for differences with the traditional approach). Instead of the individual document writing the work culminates around the workshops where relevant people (here as examples A, B and C) participate and the documents (here as examples A, B and C) are partially already written in the workshops. Inspections will take place as before if deemed necessary. Quality assurance and information sharing are integral elements of these workshops. Furthermore, the workshops are planned as part of the project planning activities to ensure their effectiveness.

#### **4.3 Existing related approaches**

The key ideas of the collaborative approach are relatively straightforward, for example, bringing people together as early as possible to effectively communicate and to design various issues. Therefore, understandably similar approaches already exist. There are, for example, complete software development methods, such as DSDM (Dynamic Systems Development Method) [DSDM 2004], that include facilitated workshops. The scope of DSDM is much wider covering almost the whole lifecycle of software development. On the other hand, there are approaches to facilitated workshops for software related issues. The scope of these approaches is usually about very specific cases (for example [Gottesdiener 2002], [Schalken et. al 2004] and [Coughlan and Macredie 2002] for requirements work) or their scope is generic fitting to almost any type of workshops or

collaborative way of working outside the scope of software engineering (for example [Klatt 1999], [Facilitation 2004] and [Taxen 2004]).

The approaches discussed here in detail do not provide a complete view of all the related approaches existing, but are the ones that appear the most similar to the key ideas of the collaborative approach presented here. These two approaches are *Joint Application Development* (hereafter JAD) [Wood and Silver 1995] and *Agile Modeling* (hereafter AM) [Ambler 2002]. JAD is probably the best-known method in the field of facilitated workshops for software related issues. The other approach, AM is a realization of the latest movement in software engineering, namely agile processes and methodologies. This section first briefly presents JAD and AM and then provides a comparison to the key principles of the collaborative approach to point out the key differences of the approaches.

#### 4.3.1 JAD (*Joint Application Development*)

JAD is defined in [McConnell 1996] as a requirements-definition and user-interface design methodology. As this was the original aim of JAD, it has been used to address the efficient production of the early documents in software projects, such as product definition related matters. However users of JAD have later realized the potential in the approach and widened the scope of the method. In fact, it is claimed that thousands of JAD meetings have been organized [Wood and Silver 1995 p.16].

In JAD, the idea is to have a set of workshops for a specific and selected matter rather than plan the workshops in a project to cover all the planning activities. In fact, JAD use appears to be a project itself. Moreover, JAD workshops always appear to be somewhat formal. JAD covers roles for the workshops such as facilitator and scribe. In addition, brainstorming and other techniques are used and facilitator's role is clearly crucial. JAD was not originally based on phases but in [Wood and Silver 1995] five steps were introduced. The phases are listed in the following:

**JAD project definition.** In this phase, the whole case is planned including who belongs to the JAD team, and the actual session is scheduled.

**Research.** This phase covers the research work that needs to be done in order to have all the material in place for the session.

**Preparation.** In the preparation phase all the details of the workshops are planned such as agenda, meeting rooms and other meeting facilities.

**The session.** This is the phase when the workshop or session is actually run.

**The final document.** In the last phase the final document is being inspected and distributed to the different people involved.

#### 4.3.2 AM (Agile Modeling)

AM is more like an ideology or methodology than a method. Ambler, the developer of AM, introduces AM as a practice based process in [Ambler 2002] and in [Agile modeling 2005] he refers to AM as a methodology.

AM provides a set of values, principles and practices describing how to perform effective modeling and documentation of software-based systems. Although AM provides a set of practices, it leaves a great freedom for the practitioner.

The applicability of AM is also explicitly restricted. Ambler mentions in his book *Agile Modeling* [Ambler 2002] that AM does not work best in large companies such as telecommunications firms, nor in environments where, what he calls, descriptive processes, such as RUP [Rational Corporation 2002], are used.

In addition, AM does not provide ways to plan the modeling sessions in software projects. On the other hand, AM provides a number of pragmatic ideas how to perform AM sessions in order to produce certain kind of models. The values, principles and practices of AM will be discussed in the following to give an outline of AM.

##### *AM Values:*

AM values define the basic ideology, or foundation as Ambler states [Ambler 2002], behind the methodology. These are listed in the following.

**Communication.** Communication with the different stakeholders is seen important in AM, as is the case with agile approaches in general.

**Simplicity.** AM is, as the name suggests, modeling focused approach, and discusses a lot about the models being just accurate enough – not more.

**Feedback.** The direct feedback about one's work is seen important in AM.

**Courage.** AM states that people doing agile modeling have to have the courage to work together.

**Humility.** Humility is similar value to courage in the sense that it has to do with people's characteristics. In practice, this means according to Ambler that developers realize they do not know everything [Ambler 2002 p.25-26].

##### *AM Core Principles:*

AM core principles define the basic rule set for applying the methodology. These are not really that different to the values, but according to Ambler the principles are more concrete than the values. The principles along with the values should give a playground for the practitioners of AM. The core principles are listed in a similar manner to the values in the following. In

addition to the core principles, there are also supplementary principles provided in AM. These are not covered here.

**Software is your primary goal.** Agile approaches in general have the principle of software being the primary goal. This principle is also included in AM, although AM itself is not about writing software.

**Enabling the next effort is your secondary goal.** In AM it is seen as important to inform the practitioners that one important reason to model is to be able to take the next step in the work.

**Travel light.** In AM it is seen that you are ready when you do not have anything to remove from your models. This means traveling light.

**Embrace change.** As in agile approaches in general, it is seen as important in AM to allow and even expect changes even late in development.

**Incremental change.** This is a principle closely related to embracing change. When the work is done in increments then changes can be taken into account. Incremental work also allows rapid feedback.

**Model with a purpose.** In AM it is noted that one should always be aware of the reason the modeling is performed.

**Multiple models.** AM states that one should do multiple models to cover the different aspects of whatever is being modeled.

**Assume simplicity.** This principle is a derivative of the value ‘simplicity’, that is, it does only whatever enables the next effort – nothing more.

**Quality work.** To explain the principle *quality work* Ambler writes “Agile developers understand that they should invest effort to make permanent artifacts, such as source code, user documentation, and technical system documentation of sufficient quality” [Ambler 2002].

**Rapid feedback.** The feedback of ideas and development decisions is intended to be immediate in AM.

#### *AM Core Practices*

While the values and principles of AM are somewhat abstract, the practices are said to be the heart of AM: “It’s the practices that you will actually apply on your projects, practices that are guided by AM’s values and principles” [Ambler 2002]. The practices are derived from the principles; similarly as many of the principles were derivatives of the values. AM core practices are categorized and listed in the following.

**Artifacts.** Apply the right artifacts and iterate to another artifact.

**Modeling.** Model in small increments, depict models simply, display models publicly and create simple content.

**Interpersonal aspects.** Have active stakeholder participation, model with others and have collective ownership.

**Tools.** Use the simplest tool possible.

**Implementation and testing.** Consider testability and prove it with code.

In brief, the practices emphasize working together in an incremental mode with all the relevant stakeholders and focuses on appropriate simplicity. Furthermore, the practices emphasize that the models are created for coding purposes and need to be able to be tested as well.

#### *4.3.3 Comparison of the approaches*

In the following the key principles of the collaborative approach are compared to JAD and AM (summary presented in Table 3) in order to understand the contribution of the new approach.

**Creating understanding.** Clearly all the approaches promote the idea of joint creation of understanding. However, the scope of creating understanding is different. JAD is most limited in this respect, at least originally, and focuses on the early phases in software projects. AM excludes project level planning issues out, although gives freedom for the applicability of the approach.

In addition, AM does not address the problems of teamwork explicitly. JAD discusses them on a practical level, but addresses the theory in rather a limited way.

**Information sharing.** Information sharing is an important matter in the case of both JAD and AM. AM even states communication is one of its values. However, the continuity of the matter is only guaranteed in the collaborative approach presented here by the planned and continuous workshops. JAD typically covers the first phases of a software project and in AM this is left up to the practitioners.

**Document writing.** In JAD and in AM the idea is not to write anything of the actual final document like it is in the collaborative approach presented here.

**Quality assurance.** In JAD quality assurance is implemented to a certain extent for requirements work, however, JAD has no clearly recognizable impact on reducing the need for inspections. In AM there are no explicitly stated intentions in this area.

**Table 3 How JAD and AM conform to the key principles of the collaborative approach**

Key element of the collaborative approach	JAD	AM
Creating understanding is done principally as a joint activity	Partially	Partially
Information sharing is done continuously	Partially	Can be
Document writing is done as a joint activity as far as feasible	No	No
Quality assurance in the planning activities in software engineering is addressed continuously by the project team from early on in the project life cycle	Partially	Partially
The joint activities are planned, as any other project activities are planned.	Partially	No

**The joint activities are planned.** JAD covers only this type of planning for requirements work and AM explicitly excludes this type of approach as was already mentioned.

#### 4.3.4 Summary

Clearly all the approaches are addressing the same issues, in other words all the approaches are trying to address the planning activities in software engineering. On the other hand, their starting points are different.

The key difference in the presented collaborative approach to the existing approaches is in their comprehensiveness for software engineering. The collaborative approach covers all the planning activities and addresses planning of these activities to ensure continuity for information sharing and quality assurance. This approach is also believed to have an impact on the needed rework and inspection efforts. The other two approaches leave more room for their application and are not aimed to be institutionalized into software engineering.

#### 4.4 Comparing the collaborative approach with the traditional approach

This section elaborates on the differences between the traditional approach, as defined in section 2.3, and the collaborative approach. This is done by first looking at the key differences of the approaches and by then identifying the tasks of the two approaches. The tasks are then used in recognizing the differences of the two approaches.

##### 4.4.1 The key differences between the approaches

**Planned versus non-described way of working jointly.** In the collaborative approach the joint effort for the planning activities in software engineering are planned. In the traditional approach, the way of working jointly is up to the responsible persons albeit the software process followed may describe the needed artifacts with their contents.

**Creating understanding jointly in workshops versus individually.** In the collaborative approach creating understanding and making the majority of

decisions is done collectively in the workshops. In the traditional approach, individuals make the majority of the decisions.

**Understanding other's work by learning in the workshops versus reading documents.** In the collaborative approach reaching common understanding happens by participating in the workshops and in this way learning about the decisions outside one's primal responsibility area. In the traditional approach, reading the documents of others is the primary way of achieving this.

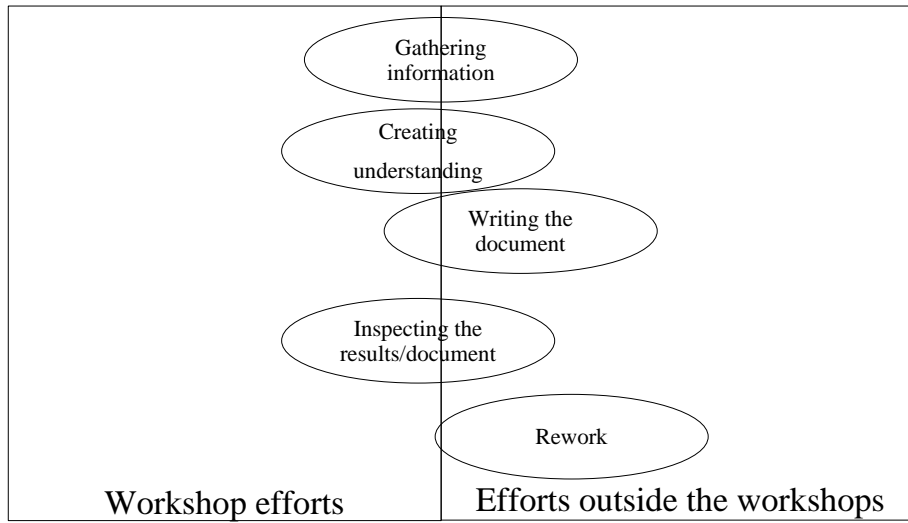
**Early sharing versus late sharing of information.** In the collaborative approach the information is shared while it is being created. In the traditional approach, this takes place mostly by reading documents of others by preparing for inspections and during the inspections themselves.

**Quality assurance built in to the way of working versus achieved by inspecting the documents.** The traditional approach relies mostly on the inspections to ensure the quality of the documentation. In the collaborative approach, the comparable quality assurance is built into the process of writing the documents, although inspections can be used, too.

#### *4.4.2 Identifying the tasks of the approaches*

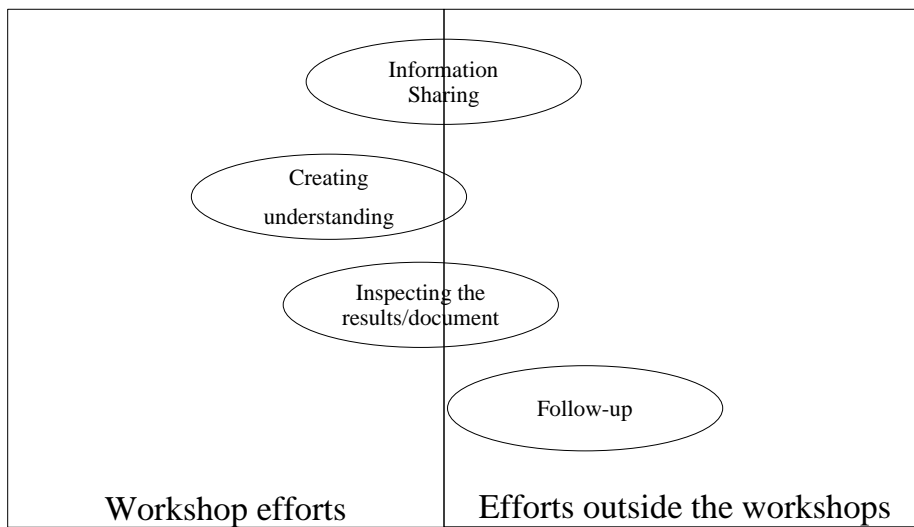
In order to compare the two approaches one has to first understand what are the planning activities. These can be derived from the goals for the planning activities presented in section 2.2. One needs to gather the required information, create new understanding based on the information and the store the information. The results are inspected and possible rework or follow-up work is carried out. Once the activities are identified, one has to identify which activities remain the same despite which approach is used. Similarly, one must also identify the activities that are not shared by both techniques.

**Common tasks.** Let us first identify the tasks that are universal for both of the approaches. In Figure 9 the universal tasks for an individual working on his or her primal responsibility area are listed. As both approaches share these activities, these tasks may result in efforts in the workshops or outside the workshops. Figure 9 illustrates how the tasks may be divided between the individual and workshop tasks when the collaborative approach is applied. In other words, majority of the efforts for creating understanding is carried out in the workshops whereas writing the document still takes place partially outside the workshops, as was suggested by the key principles of the collaborative approach. The division is case dependent and therefore the division presented is only directive. When ad-hoc planning meetings are not considered all the tasks would be outside the workshops in the traditional approach.



**Figure 9 Efforts for individual responsibilities in the collaborative approach**

Similarly, Figure 10 illustrates the universal tasks that an individual carries out to support his or her team. Again, an estimate on the division of the efforts between workshops and individual work is given. In practice, this means that the team members support each other by sharing information in the workshops, jointly create understanding and even inspect the documents in the workshops. The support team members give for each other is assumed to mostly take place in the workshops. The follow-up work for inspections, however, is carried out outside the workshops. The following describes the universal tasks in more detail and gives rationales for the illustrations.



**Figure 10 Efforts for team support in the collaborative approach**

*Gathering information/Information sharing:* No matter whether the collaborative approach is used or not, the relevant facts need to be acquired. Such information can be, for example, standards that need to be understood before the work can continue or information coming from outside the team. While this kind of work can be included in the workshops, it is not the intention in the collaborative approach to do the research type of work in teams. On the other hand, other members of a team may possess the required information for someone else's responsibility area. In addition, there is typically a need to understand the areas of the design that other people are working on.

*Creating understanding:* Unless new understanding is created the planning activities have not much value. In an experienced team that is working with familiar subjects, creating understanding plays a less important role, but even in these cases the possessed knowledge needs to be applied to the situation at hand. Creating understanding in the traditional approach is mostly left to individuals whereas in the collaborative approach the majority of this work can be done together.

*Writing a document:* If a document needs to be written, it has to be written in one way or another. This does not change from one approach to the other. The difference between the two approaches is, however, the amount of writing done individually. In the collaborative approach, part of the writing is carried out in the workshops while in the traditional approach the person responsible for the document at hand performs all the writing alone.

*Inspections:* Here inspections are understood widely to cover both preparation work and the actual meetings. Inspections are something that could be left out from both of the practices in principle, but in the traditional approach inspections are basically the only quality assurance activity, and are hence more important.

*Rework and Follow-up:* Rework after the inspections is typically needed. The task is universal similarly to inspections. Follow-up activities are a consequence of rework.

**Tasks specific to the traditional approach.** If we then consider the tasks specific to the traditional approach, it turns out that all traditional approach tasks are universal to both approaches. Consequently, the collaborative approach actually only adds new tasks to the way of working.

**Tasks specific to the collaborative approach.** The efforts that are put into running a workshop are about gathering information, creating understanding and inspecting the created understanding. Hence, workshop implementation itself is not a new task, it is just another way of performing the same tasks as before. In the workshop approach, however, the team can waste or save time compared to the traditional approach.

*Workshop planning and preparation:* All the activities that are required to run a successful workshop are new tasks related to the collaborative approach. Naturally, the scheduling of the workshops is coordinately part of this.

#### 4.4.3 *Comparison of the approaches*

The comparison of the two approaches is a multidimensional matter and modeling perfectly how people perform the activities is relatively difficult. Therefore the following analysis makes a few assumptions.

Firstly, the people are expected to work only for one project, and secondly their efforts are considered to only consist of the ones described just earlier. This means that no one has activities outside the planning activities. In practice, this means that for a certain period of time the work is going to result in exactly the same efforts in both of the approaches. For example, if the planning activities and inspections take two months and 10 people are working in the project, both of the approaches result in 20 man-months worth of efforts. These assumptions, although not exactly true in real life, actually reveal two key principles of the comparison.

First, the calendar time efficiency of the work is important as this defines also the efforts. This means that if people are not participating the workshops they are still gathering information or creating understanding individually. Hence, joining a workshop does not yet save any time, as the project members use this time anyway, but the real differences come from the effectiveness of the approaches.

Secondly, and quite self-evidently, the quality of the work is essential as this defines the rework required in the later phases.

The following analyses the tasks one at the time and introduces the possible differences that can be identified between the two approaches.

**Gathering information.** Let us first consider gathering information. From the individual point of view this means reading documents that were previously written, asking questions from the project members (or someone outside the project) and in the workshop approach asking questions in the workshops. The more the person knows the less information gathering efforts are needed. For an individual there appears to be no differences between the individual approach and the collaborative approach. The person needs to gather the same information in both of the approaches. The more knowledge acquired individually the less needs to be acquired in the workshops and vice versa. Naturally, this view is limited as only the efforts of single individual are considered, and thus the efforts of the team are discussed later. It also assumed that the person knows whom to ask. If people are well aware from whom to ask, there are no extra efforts required. On the other hand, if people do not know from whom to ask, workshops

provide a better chance of reaching the right people and may result in lesser efforts. This gives us the first finding:

1. The less people know the expertise of other people the more efforts can be potentially saved with the workshop approach.

The finding indicates, but does not ensure, that a project with people new to each other may benefit more from using the collaborative approach.

Additionally, it is expected that people really do ask. In other words, people are not introverts in this sense or ignorant of the knowledge of other people. However, it is not so clear how the social aspects could influence these efforts, and whether asking individually or in workshops is preferred.

In addition, the former does not take into account which approach is more efficient in information gathering, reading or asking. This may cause some differences between the two approaches as in the traditional approach reading plays a typically more important role than in the collaborative approach.

**Sharing information.** Information sharing can be planned or it can happen on ad hoc basis. The workshops can be used for information sharing purposes, although sometimes a specific information sharing session is needed to cover wider audiences.

If we then consider information sharing as a support function, answering the questions people are asking from other members in teams, take almost the same time required for asking the questions. In workshops, the same applies, however, now the same issues need not be asked several times. Hence, the more overlapping issues there are, the less time is required for answering the questions. This aspect gives us the second finding:

2. The more people need the same information the more time and effort it is possible to save in information sharing through the workshop approach.

Naturally, this works vice versa. If people generally do not require the same information and workshops are used for information sharing purposes, this means extra efforts for all the people that the information does not concern. This, in fact, indicates that selecting the right participants and subjects for the workshops is crucial.

**Creating understanding.** Let us then consider creating understanding. Both individually and in teams the complexity of a problem is a factor for influencing the efforts required. Additionally, the efforts are dependent on the competences of people. In both of the approaches there is a chance that the problem is too complex resulting in principle in infinite efforts. However, in a business environment, this does not really occur. A decision is made anyway, although it is not necessarily correct. It may also be that the original problem is avoided with an alternative approach.

Nevertheless, the differences for the individual approach and the workshop approach are based on the competence levels of people and complexity levels of the problems. When individuals are highly skilled there is a chance that workshop approach provides no added value. The team may actually require more time to come up with a decision, as the coordination of the team requires time. This gives us the third finding:

3. The more skilled the individuals are, the less benefit is to be gained from the workshops.

On the other hand, there is a chance that even highly skilled individuals are still only experts in their own areas. This gives us the fourth finding:

4. In tasks that require several fields of expertise the workshop approach can provide answers for issues that individually are extremely difficult or even impossible to solve.

Moreover, creating understanding can also serve the purpose of information sharing in the workshop approach. This gives us the fifth finding:

5. The more creating understanding serves the purpose of information sharing the more time it is possible to save from information sharing tasks (see also finding 2).

**Writing the documents.** For writing the documents there are no clear differences between the two approaches. The documents need to be written in one way or another. However, if the document is written in a workshop, efforts are potentially lost as several people may be waiting for someone to write the decisions. The size of the team is consequently a factor as well. On the other hand, part of this work can be seen as quality assurance work in form of inspections. Hence, there is a trade off between the waiting around while the document is written and the quality benefit that is added by the participation of the additional people during the writing process. Therefore, we get the findings:

6. The more efficient writing the document is in the workshop the less efforts are wasted.

7. The larger the workshop team the more efforts are possibly wasted in writing the documents.

8. The more the results are inspected during the workshops the less time is required in preparing for inspections and in the inspections themselves.

**Inspections, rework and follow-up.** Inspections are a quality assurance activity that takes place after the work is finished. Preparation efforts are the same in both of the approaches in principle, but in the workshop approach some of the preparation efforts already take place as part of the workshops. Actual inspection meeting efforts are dependent on the size of the material

to be inspected and the quantity of defects found. Rework and follow-up efforts are consequences of the defects and thus the efforts reflect the number of defects found. These aspects give us the next finding:

9. The inspection, rework and follow-up efforts are dependent on the quantity of defects and their severity. If the use of workshops reduces the number of defects found the related efforts are reduced accordingly.

**Workshop efforts.** The workshop related efforts that cannot be defined as universal tasks are additional to the traditional approach. These are all the tasks required for planning the workshops. Hence, we get the last finding:

10. The workshop planning, preparation and implementation tasks are additional efforts to the traditional approach. Hence, these efforts are deducted from the possible benefits.

Therefore, there is a threshold for the amount of effort that should be invested in planning the workshops beyond which there is no additional benefit. This is when the planning efforts outweigh the potential gains they bring. In practice identifying this threshold is often not straightforward and this is a common problem in any planning activity.

#### *4.4.4 Other issues*

For a whole software project to really benefit from the use of the collaborative approach there must be savings gained throughout the whole project. The savings may be gained in two phases: first they may be gained in the actual phase when the collaborative approach is used, during the planning activities, or they can be gained later during the implementation, the testing or the maintenance phases, for example. Therefore, analyzing only the planning activities for software engineering is not sufficient as the defects may be manifesting in the implementation phase or testing phase. Hence, one should be looking at issues such as the timetable of the whole project and specification faults identified in implementation and testing, to really be able to compare the two approaches.

#### *4.4.5 Conclusions*

The findings presented above are based on simplified views of reality. Furthermore, as mentioned earlier a thorough analysis would also require examining the time spent in implementation. Additionally the specification defects realized in implementation and the findings in testing phase should be analyzed. Nevertheless, the analysis and findings provided in this section already point out some crucial factors differentiating the two approaches.

First of all, the calendar time efficiency for the whole planning period is an important factor as it plays an important role in saving efforts. Secondly and quite self-evidently, the quality of the work is a crucial factor, and this may result in savings in later phases of development also.

In addition, an important matter is the divisibility of the problem (see section 3.5.1 for the definition of problem types). This could mean for example, if a system can be divided easily into subsystems without major dependencies between the subsystems. The less divisible the problem is the more is potentially gained with the joint efforts. This is at least true for findings 2 and 5.

To conclude, the comparison does not provide an answer if one approach is better or worse from the perspective of effort or defects. It merely identifies some factors influencing the differences between the efforts required for the two approaches. The real life approach to the planning activities is a relatively complex and multidimensional process, and hence the simplifications provided here do not provide complete enough answers to fully analyze the differences.

#### ***4.5 Applicability of the collaborative approach***

The collaborative approach as presented does not explicitly state restrictions for its use. Hence, the applicability in software engineering work is possibly wide too. However, at least the way the approach is applied varies from one artifact or document to another. Probably the key factor in the differences in using the approach is in the purpose of the artifacts. Some artifacts are products themselves, others are internal artifacts supporting the later development phases and some documents are only back-of-the-napkin sketches supporting a relatively small team with very low requirements for the visual form of the artifact. The difference is usually about how much of the finished artifact can be achieved in the workshops. Product artifacts are, for example, customer documents, agreement documents and interface definitions for wide audiences. These artifacts most likely cannot be created in the workshops. Non-product artifacts, and the more typical in software projects, are documents like subsystem specifications, user interface specifications and requirements specifications. When a small team (2-3) is making their internal decisions, which the process does not require to be documented, we can talk about back-of-the-napkin sketches. In these cases there is hardly any documentation work left outside the workshops.

#### ***4.6 Expected benefits of the collaborative approach***

The following discusses the expected benefits of the method. Reaching these expected benefits is discussed in the empirical evaluation section of the dissertation (in Chapter 7). In short, all the mentioned benefits seem to have some sort of support from the experiment results.

**Reducing calendar time used for document authoring.** The collaborative approach is expected to improve the calendar time efficiency for document authoring. This can only happen if all the needed information is at hand. However, even if the needed information is not at hand using the collaborative approach is expected to speed up the process of acquiring it.

**Improving team co-operation.** By providing a systematic practice for human interaction in software projects the approach is also believed to help building team co-operation.

**Enabling more efficient information sharing and communication.** By providing a continuous approach for creating understanding jointly as well as sharing it, the information sharing is expected to be improved. The overall communication of the project is also expected to improve with this approach.

**Improving common understanding.** With the improved information sharing and the early involvement in decision-making, the approach is believed to improve and balance the general understanding level of the stakeholders.

**Ensuring better commitment to project work.** With the improved common agreement and earlier involvement in decision making the approach is believed to improve the commitment of different stakeholders.

**Decreasing inspection time and effort.** By getting people to familiarize themselves earlier with material in question the approach is believed in general to decrease the time needed for inspections. The risk that inspections turn out to be specification meetings is expected to be reduced by the earlier involvement of participants.

**Improving inspection efficiency.** Similarly to the previous benefit, by getting people to familiarize themselves earlier with the different matters of concern, the inspections are believed to be more efficient in terms of finding defects.

**Reducing rework resulting from document authoring.** Assuming the process of using the collaborative approach reduces the number of problems or their severity identified in documents during inspections (or later), the related rework should also be lower.

**Improving the overall quality of documentation.** This goal is quite similar to the goal of reducing rework. If there are less defects or the defects are of less significance, the overall quality has probably improved too. Furthermore, the quality perceived by the different stakeholders is expected to improve, that is, by participating more in the decision making process people are generally happier with the results.

**Reducing the efforts used in producing documents.** The belief is that by improving the information sharing, the communication and the inspections while at the same time reducing the rework, the efforts used in producing documents would lower too.

## 5 IMPLEMENTING THE COLLABORATIVE APPROACH IN NOKIA – THE RaPiD7 METHOD

This chapter elaborates on the key principles of the collaborative approach presented in Chapter 4 and explains its realization in Nokia as a method called RaPiD7 (rapid production of documentation, 7 steps). Layers for the method as defined in Nokia are presented and then covered in detail with related key activities, roles, tools and best practices.

### 5.1 Layers of RaPiD7

RaPiD7 has been described in Nokia through layers that have different concerns. These are visualized in Figure 11. The reasoning for the layers is twofold. First, the layers give the approach structure that helps the implementation by separating the viewpoints of the method. This is important since the method is trying to be a pragmatic approach for the identified problem. Secondly, the layers separate the different activities for the different users of the method.

The first layer describes how the method is applied in software projects using the software processes applied in Nokia. Hence the project layer is targeted to ensure that as a part of project planning all the needed human interaction and joint decision-making is planned, too.

The next layer describes how the method is applied for a single case, such as one document. This is because the different artifacts are tangible elements of software development. For this layer the method provides help on deciding how many workshops are needed for a case, and who to involve in the planning work.

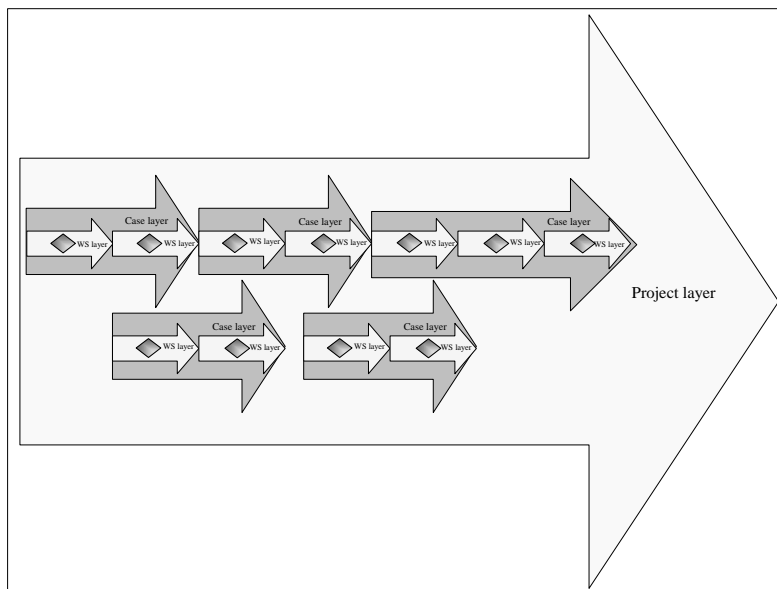


Figure 11 The three layers of RaPiD7

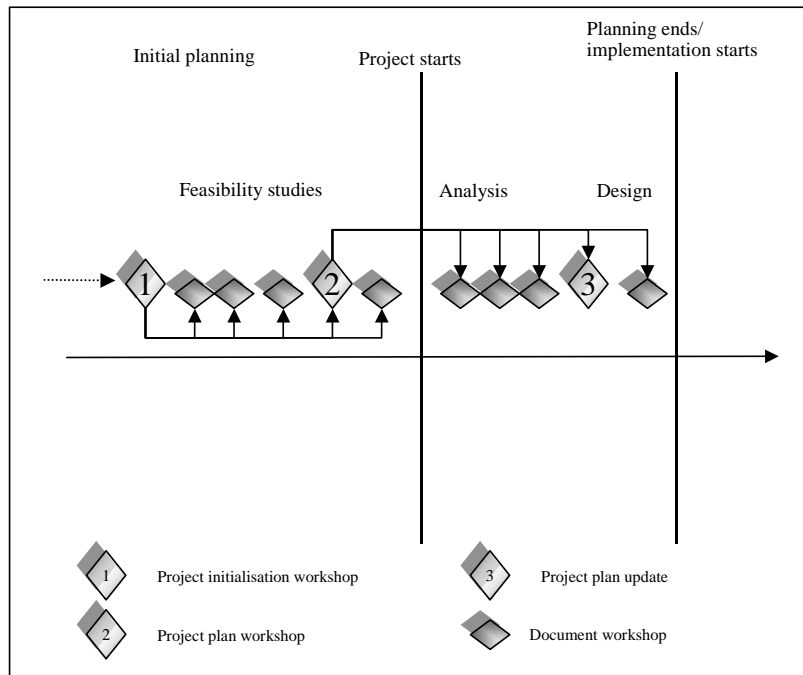
Finally the last layer describes how to have workshops and how to answer the challenges in the workshops in software engineering context. In the workshop layer, a single workshop is planned and carried out according to the ideas of RaPiD7. This is where the steps, as suggested by the name, appear in the method implementation.

## ***5.2 Implementation of project layer***

Systematic benefits are expected if the method is used in software projects instead of using it for individual cases only. However, before the project level can be addressed the used software process needs to be analyzed. Software processes have received an increasingly important role as process focus has been emphasized during the 1990s [Zahran 1998]. Many organizations have tailored their own processes (an example of such in the target organization of this study is OMT++ [Jaaksi 1997] or [Aalto et al. 1999 ]). As was described in Chapter 2 the different processes define the models or artifacts produced in developing the objects of software development. Hence, the used process needs to be taken into account when using RaPiD7. The following defines steps of integrating RaPiD7 into a general software process and thus giving the starting point for project layer work:

- a. Identify the planning activities.
- b. Identify the activities that require and benefit from the joint decision-making and information sharing.
- c. Identify the outputs (artifacts) of the activities benefiting from the joint decision-making and information sharing.
- d. Identify the typical number of workshops required for a document, people participating in the workshops and issues for the workshops.
- e. Identify workshop approaches for the most common workshop types.

In brief, the key matter in integrating RaPiD7 into a software process is in identifying how the different software processes are addressing the planning activities. The planning activities need then to be addressed in the different layers of RaPiD7 and initially in the project layer. The expected benefits of integrating RaPiD7 with software processes are enabling systematic and efficient use of the method in the projects, providing information on the required workshops, enabling shorter planning times for workshops, and enabling more efficient workshops due to usage of known viable approaches.



**Figure 12 Planning RaPiD7 workshops in projects**

Once the process integration is done the planning can be done on the project level. This means in practice that the whole project is initiated by a workshop to initiate the workshop planning process, and from this point on the project comprises of workshops concentrating on different aspects of the project. An example of this is illustrated in Figure 12.

In practice, the whole idea is about moving from individual specification work towards working in teams for most of the decision-making and document authoring activities within the project. This does not mean that there would not be any individual tasks left. Some study work is better done alone and some detailed technical solutions are still left for individual designers. However, the idea is to realize that software development is really a joint activity right from the beginning as was discussed already in section 1.1.

### 5.2.1 Roles in the project layer

**Project manager.** The project manager is the person typically responsible for making sure the first initialization workshop takes place. This is one of the new tasks for the project manager when using RaPiD7. When the first workshop is held the people owning the cases should take the responsibility for the related workshops. However, the project manager can utilize his workshop participation to track the progress of the project. Often, project meetings are the way progress is tracked, and the progress evaluated by analyzing the feedback from the different members of the project. The

workshop approach allows the project manager to track the progress of the project more easily than in the traditional approach. Furthermore, the project managers need to react to the possible problems or changes in the initial workshop plan and to make sure adequate workshop planning sessions are planned for the future, too (unless the first session covers the whole project).

**Case owners.** Once the initiation workshop has been carried out successfully, the case owners should take the responsibility for the different cases. The audience for the first workshop should be wide, as several case owners will be identified in the first workshop. The actual planning of the cases is then carried out at the case layer.

### 5.2.2 *Key activities*

The key activities identified for the project layer are:

- organizing project planning workshop (project manager),
- (re-)scheduling the workshops (project manager with case owners), and
- initial planning of the cases (case owners).

### 5.2.3 *Tools in the project layer*

**Artifact list.** The roots of this tool are in the process descriptions as the candidates for the issues to be solved can be identified from these. The artifact list includes the cases, the related workshops with tentative goals, case owners, case-planning teams and the timetable for the cases. These issues are then assigned to certain workshops and this list acts as a plan for solving them. The project manager can use this tool when planning the workshops. In practice the artifact list can be implemented by a spreadsheet application, for example.

### 5.2.4 *Best practices for the project layer*

**Planning the workshop schedule.** In the first workshop the overall layout is planned, in other words, roughly how many workshops will be needed (or can be held), when the workshops are going to be held and who should participate in the workshops. The idea is not to plan the whole set of workshops in detail early on, as changes may happen. The idea is rather to allow people to reserve time for the workshops, to get the initial commitment, and to roughly understand the number of workshops needed. After the initialization workshop the workshops continue according to the plan, but changes are made as needed. There can be certain checkpoints when the plan is subject to a closer study in order to see whether the original plan needs to be changed more dramatically (see in Figure 12 the workshops two and three). Another approach for the project layer is to agree on certain

workshop dates every week similar to regular project meetings. The risk in this approach is to have the planning of the workshops overlooked.

**Organizing inspections.** In principle, inspections hold the same value as before. However, inspections are not necessarily needed when applying RaPiD7 rigorously as the feedback from the different stakeholders is received several times. On the other hand, there are a few clear exceptions to this:

- The document has not been completely authored in the workshops for example, because of its product like nature. It is possible that the workshop would just be out of time and then the document would be finished by individuals later and would thus require an inspection.
- The document is meant for wider audience than the people in the workshops and thus the need for input outside the team is required.
- The process requires that inspections are held.

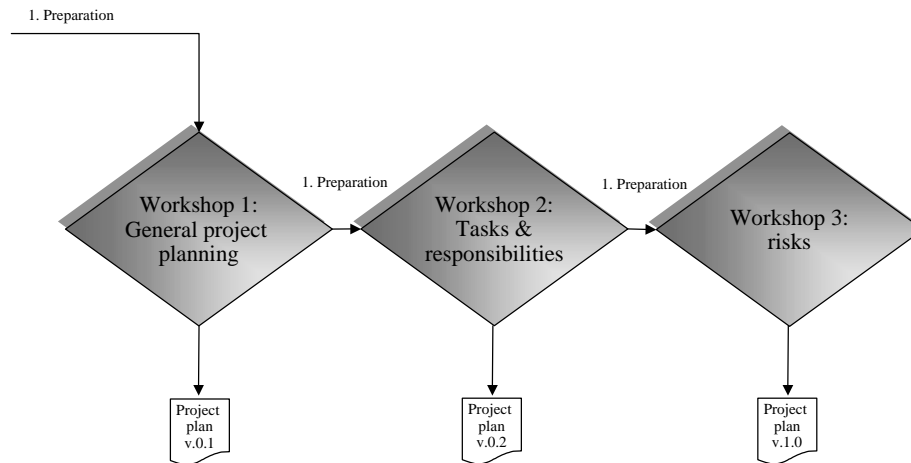
Naturally, if inspections are not held and the documents need to be formally accepted, the acceptance takes place in the last workshop. This may require changes in the list of the participants for this workshop. Additionally, a cohesive workshop team may overlook some solution candidates, and for this reason inspecting the documents by someone outside the team may be required.

### ***5.3 Implementation of the case layer***

The case layer ensures that a case is planned and carried out efficiently by organizing the needed amount of consecutive workshops. Figure 13 illustrates a simplified example of how consecutive RaPiD7 workshops are used, for example, in order to create a project plan (or rather some parts of it).

In the first workshop illustrated in Figure 13, some general decisions are made related to the project. These general decisions can be, for example, the scope and goals of the project. Then in the second workshop the tasks, their relations, timetables and responsibilities are discussed. In the final workshop, the risks are analyzed and all the previous work is put together.

The idea of producing documents in consecutive workshops, as explained above, is one of the key principles of RaPiD7. If there is only one workshop for some entity the separation of the case layer and the workshop layer is somewhat immaterial.



**Figure 13** Creating a project plan in RaPiD7 workshops

### 5.3.1 Roles and teams in the case layer

**Case owner.** Case owners continue their work from the project-planning phase (the project layer) by initiating the planning of the workshops. They also make sure the workshops for the cases are really carried out and the goals of the workshops are met. The case owners report both the successes and failures of the workshops to the project manager and inform him about the possible changes in schedules.

**Case-planning team.** The case-planning teams can be initially decided in the project-planning workshop, but the case owners should ensure these teams really assemble. It is often beneficial to plan the workshops with a team instead by single individual in order to ensure wide enough coverage in the workshops. Hence these teams are important. This team decides the roles for the workshops and decides how the individual workshops are planned.

### 5.3.2 Key activities

The key activities identified for the case layer are:

- Planning the case (case owner with case-planning team) by
  - a) (re)-estimating the number of workshops,
  - b) deciding the team for the workshop (or teams for workshops), and
  - c) organizing the workshop facilities.
- Reporting to project manager (case owner).

### 5.3.3 Tools in the case layer

**Invitation templates.** The invitation template is a blank version of the real invitation to be sent to the people participating in a workshop and lists the relevant issues to be remembered in an invitation. Thus, the invitation template is a kind of checklist for the invitation. The invitation template used in Nokia is presented in Figure 14. Typically, a single invitation is sent per case even if several workshops are held.

**Document templates.** If the software process followed provides document templates these are also possible candidates to be used in the workshops. However, a document template is not always useful in a workshop as such. The template may not provide a good structure for the workshop and filling the template may be difficult in the workshop. Therefore, it may be more practical to develop a new version of the template for the workshop by capturing the relevant issues from the original template and creating a workshop template of the document.

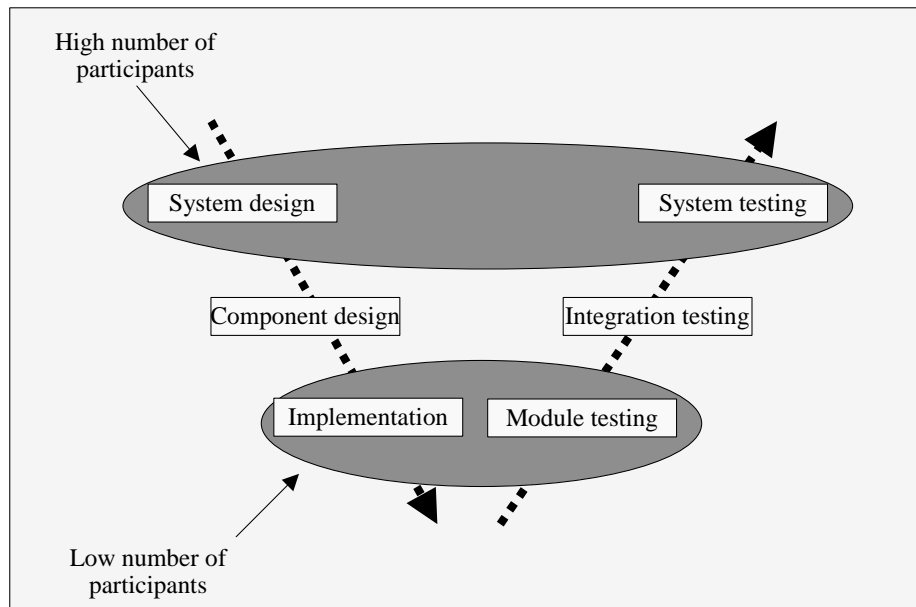
<p><b>Project:</b> &lt;project name&gt;</p> <p><b>System component affected:</b> &lt;system component name&gt;</p> <p><b>Feature:</b> &lt;feature name&gt;</p> <p><b>Goal:</b> &lt;goal of the workshop, for example "writing the technical specification X"&gt;</p> <p><b>Scope:</b> &lt;in the case of multiple workshops limit the scope of this workshop, for example, "the first RaPiD7 workshop shall clarify the scope of X and find out how far existing Z can be used."&gt;</p> <p><b>Time:</b> &lt;date(s) and time&gt;</p> <p><b>Venue:</b> &lt;meeting room name, address, room number&gt;</p> <p><b>Roles:</b> &lt;name, role, needed full time/needed part time (when?)&gt; &lt;name, role, needed full time/needed part time (when?)&gt;</p> <p><b>Facilitator:</b> &lt;name&gt; <b>Secretary:</b> &lt;name&gt;</p> <p><b>Preliminary agenda(s):</b> &lt;timetable roughly and agenda items, for example the following way: 14:00 - 14:30 Introduction to X 14:30 - 15:00 Discussion: Scope of the Y 15:00 - 15:30 Discussion: How far can the existing technical specification "Z" be used. 15:30 - 16:00 Effort Estimation &gt;</p> <p><b>Commitment by:</b> &lt;date, time&gt;</p> <p><b>Commitment to:</b> &lt;name&gt;</p> <p><b>Location for preliminary material:</b> &lt;location of the material, for example, a web page, network drive&gt;</p>
--

Figure 14 RaPiD7 workshop invitation template

### 5.3.4 Best practices for the case layer

**Selecting RaPiD7 workshop participants.** To succeed in the workshops forming a proper workshop team is crucial. The right team is important because it provides all the information needed in the workshop. A team possibly provides a wider scope for planning, and therefore the suggestion in RaPiD7 is to plan the workshops in teams too, although the case owner initiates this work. Having a team planning the workshop helps in achieving an agenda with better coverage, and in addition, it minimizes the problems related to different opinions about how a workshop should be organized.

Selecting the workshop participants for the first time in a new project also possibly with new personnel is not straightforward. It requires enough understanding of the problem or document at hand as well as knowledge of who knows what in order to be able to invite the right people. This may also be true when trying out the method for the first time. The planning team does not necessarily know the project members well enough to enable them to make the right choices. In addition, the way the workshops will change the former ways of working cannot be foreseen necessarily. However, the planning gets easier after the first trials. The team will understand better who knows what and who is missing from a workshop. One approach for identifying the workshop participants, and thus identifying the kind of roles required in the workshop, involves looking at the V-model [Rook 1986] often used to describe the relationship between value adding and verifying software engineering activities. This is presented in Figure 15.



**Figure 15 Identifying the workshop participants from the V-model**

Evidently, the number of people required in the workshops in the earlier phases is higher than later in the project. Architects responsible for the various parts of the system try to create the understanding of the overall system, testing engineers want to understand the big picture to be able to start planning testing early enough, and software designers should be involved in the early phases as well to better understand what they will be developing. Quite logically, the number of people needed to participate in the workshops gets lower later in the project, and actually can go as low as two to three people. It is questionable if meetings of two people can really be referred to as workshops. However, if the need for communication and discussion between two people can be identified, it does not mean that the work related could not be planned and followed up. The level of formality in these small “workshops” may however, be substantially lower than in a “regular” workshop.

The following summarizes the key issues in forming a RaPiD7 workshop team and in planning the workshop with the team:

- The case owner initiates the planning work.
- A team plans the workshop together and the team consists of selected people participating in the actual workshop.

The number and roles of the people participating the workshop are dependent on issues the workshop addresses.

**Considerations on the number of workshops.** Practitioners of the method often ask what is the ideal number of workshops for a certain case. Unfortunately, there is no correct answer to this question. The correct answer no one really likes to hear is “it depends on the case”. However, the number of workshops does not need to be guessed either. First, probably the most intuitive approach is to simply learn from the past and look at the number of workshops needed for similar cases before. The workshop agenda planning process can also help in estimating the number of workshops needed. This process gives a good estimate of how much work can be put into a single workshop. The process for planning a workshop agenda is described in detail in section 5.4.4 as a part of the workshop layer description.

**Formal versus informal workshops.** As RaPiD7 is described in a quite detailed manner, the natural question that comes to ones mind is that will the method be too heavy for smaller scale use. Hence, the formality or precision in using the method is an issue that needs to be addressed. Here formality means how rigorously the different layers, practices and ideas of the method are applied for different cases. This level of formality in RaPiD7 workshops depends on several issues. The greatest difference is in the number of stakeholders participating in a workshop or workshops. The more

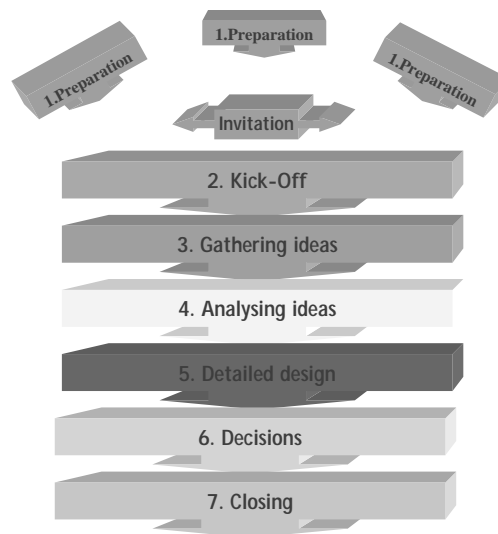
stakeholders are participating in a workshop the more formality is typically needed.

The reason for the need for higher formality due to higher participant numbers is simply due to greater risk of ending up in chaos with a larger number of participants. This is because all of the team members will bring their own opinions on what should be done and how. However, there is always a limit in how much effort should be put into planning the workshops. At a certain point, more careful planning cannot be justified by improved results as the planning efforts are jeopardizing the resulting efficiency gains in the workshops.

In general, the first workshops require more formality and as the project progresses less formality is needed. Typically, later in the project people start to work with their familiar colleagues. Lowered formality in these cases is often even a necessity to ensure consistency in using the method as the planning efforts may outweigh the benefits gained or planning may simply feel too heavy. Too high formality in such an environment may result in reluctance in using the method. Practitioners report having ad-hoc workshops consisting of two to three people and they see these workshops as beneficial, too.

#### **5.4 Implementation of workshop layer**

The workshop layer describes how the method is used in a single workshop. The steps of the method appear in the workshop layer. There are also a few unique roles only existing in this layer.



**Figure 16 Steps of RaPiD7 [Kylmäkoski 2003]**

The following explains the steps, explains how workshops agendas can be created and provides other best practices for the workshop layer implementation. RaPiD7 steps are illustrated in Figure 16 as defined in Nokia. These steps act as the starting point of a single workshop.

**1 Preparation.** Preparation is the only step performed before the actual workshop. It covers tasks that should be performed in order to guarantee a successful workshop. These tasks include, for example, ensuring the commitment from the participants, planning the workshop in detail, reserving the required facilities for the workshop and inviting the people to the workshop. In Figure 16 the different arrows for preparation illustrate that this step may take place distributed.

**2 Kick-off.** Kick-off is the first step in a RaPiD7 workshop. In the kick-off, the workshop team ensures that everyone is aiming at the same goal in the workshop, and that everyone understands and agrees on how to proceed in the workshop. The kick-off step should not be overlooked for the sake of starting “the actual work” in the workshop. Rushing forward too early typically leads to aiming at different goals, and disagreement about the way the workshop should be run. The kick-off takes typically anything from 20-60 minutes.

**3 Gathering ideas.** The gathering ideas step is typically performed few times in a single workshop. The gathering ideas step serves couple of purposes. First, it is meant for, as the name suggests, coming up with an idea asset to start the discussions and analysis. Secondly, it is about controllable way of handling the idea asset at hand. Therefore, it is not always necessary to perform the step in order to come up with new brilliant ideas, but rather to introduce different people to the same idea asset and thus to be able to start the analysis. The gathering ideas step is typically short and takes about 5-20 minutes. However, it often saves even more time in the analysis phase.

**4 Analysing ideas.** After gathering the ideas typically the most laborious task in the workshop starts – analyzing the ideas. In this phase, the listed ideas are covered and elaborated further one by one. This phase demands probably the most careful planning work and good facilitation. Analyzing ideas typically takes from 30 to even 90 minutes. To utilize efficiently a 90 minutes period, this part of the workshop must be well structured, and use techniques planned well beforehand.

**5 Detailed design.** Detailed design is something that often merges with the analyzing ideas step. This is because when the analysis is made, decisions are also made, and these decisions tend to be written down or at least they should be written down. Therefore, detailed design is typically more of an element in a workshop than a step, in other words, you can identify later that the detailed design was being made, but in what order it took place is not as clear.

**6 Decisions.** Similarly to detailed design, the decisions step is often merged into the analysis as well, as it is difficult to see whether one is doing analysis, making decisions or detailed design. However, sometimes after a long period of analysis (and detailed design) the group may need to make a decision about the suggested ways to go. On these occasions, one of a selection of decision-making techniques can be used. In such a case the step is independent from analyzing ideas and detailed design steps. As an individual step, the decisions step should not take too long (5-15 minutes) if the analysis has been done properly.

**7 Closing.** Closing in RaPiD7 is a similar activity to the activity of closing any meeting. In RaPiD7 workshops, the closing covers a review of the initially set goals versus reached goals, open issues, next steps, and finally feedback about the workshop itself. Closing can take 10-20 minutes.

The steps, their goals and typical duration are summarized in Table 4.

**Table 4 Goals and typical duration of RaPiD7 steps**

Step	Goal	Typical duration
1. Preparation	To be prepared enough for an efficient workshop that has a defined goal and needed people with the right information	from days to couple of weeks
2. Kick-off	To gain common understanding about the goal and terminology of the workshop	20-60 minutes
3. Gathering ideas	To find and create possible ideas for a specific part of the artifact to be produced	5-20 minutes
4. Analysing ideas	To select the initial set of ideas that will be further developed and specified in the workshop	30-90 minutes
5. Detailed design	To further develop, design and document the ideas to the desired level	(integrated with 4. Step)
6. Decisions	To select the most feasible solutions, and also to check the consistency between different solutions	5-15 minutes (or integrated with 4. Step)
7. Closing	To decide if the desired goal is achieved or to make a decision to have an additional workshop, and to decide what to do before the next workshop	10-20 minutes

#### 5.4.1 Roles and teams in the workshop layer

There are two particular roles defined in RaPiD7 for the workshop layer. These roles are *facilitator* and *secretary*. In addition, there is the workshop planning team as a direct continuation from the case layer.

**Workshop planning team.** Often, if not always, the *case-planning team* is the same as the workshop planning team. Sometimes when the case is large and covers several different competence areas the workshop planning teams can be different to case-planning teams. In any case, the workshop planning team makes sure an individual workshop is planned and carried out efficiently.

**Facilitator.** The role of the facilitator is the most essential in the workshop layer. A facilitator initiates the set of workshops (or single workshop), takes care of the initial planning, invites people, organizes the workshops and runs the workshops. Therefore, it is crucial for the success of the workshop that the facilitators are skilled in using the method. There can be several

facilitators in a single workshop and there can also be more than one facilitator at a time. However, in such cases a clear definition of the responsibilities for the facilitators should exist. In addition, a head facilitator should always be named having the overall responsibility about the issue at hand.

Sometimes the facilitator can be solely to blame for an unsuccessful workshop, and while a good facilitator cannot always save a workshop, at least the facilitator makes a big difference in the case of a successful workshop, too.

One of the typical misconceptions is that the facilitator should be the owner of the subject of the workshop. In an ideal case, the facilitator should be someone with good knowledge of the domain of the workshop. However, it would also be good if the facilitator does not have any personal interest in the contents of the outcome. The facilitator takes care of the process of getting to the results, but not so much for the results themselves. The facilitator can do some sanity checks on the results, but even here the facilitator can and should utilize the team.

Additionally, the facilitator must have good interpersonal skills. The facilitator should be able to express his or her own opinions clearly as well as being able to reproduce the ideas of others. This in turn reduces the need for all the participants to have good interpersonal skills. In addition, the facilitator is not the “manager” in the workshop. The facilitator should be as invisible as possible and yet provide steering when needed. The facilitator is like an active safety instrument in a car; not noticeable before needed, subtle but noticeable in handling minor incidents and takes heavy control in order to save the day when needed. In addition, the facilitator should always stay as objective as possible and separate the ideas from the people.

There are number of books and articles written about the field of facilitation (for example [Wood and Silver 1995] Chapter 12, [Macaulay 1999] and [Facilitation 2004]) and thus all the elements of facilitation are not discussed here.

**Secretary.** Secretary’s role is essential, too. Secretary takes care of recording decisions and the results in the workshops. Often constant discussions are going on, and hence it is essential that the secretary is skilled in listening, capturing the gist of the discussions and documenting the decisions. Originally, the role of secretary was not considered crucial and hence it was believed to be easy to fulfill. However, practice has shown quite the opposite. Firstly, the decisions made in the workshop do not help much if they are not written down. Ensuring this is of course not only the task of the secretary, but also a task for the whole workshop team. Being successful in the role of the secretary is greatly dependent on the good planning of the workshop. The communication between the facilitator and the secretary is crucial as well. People often comment that writing even

parts of the document in the workshop is a waste of time. On the other hand, people also comment that the decisions in the workshop are not written down at all or are not written in great enough detail. Therefore, the role of the secretary should not be overlooked, as the person in this role must enable efficient capture of the decisions if the workshop is to be successful.

Other roles exist in the workshops, too. However, these are case dependent. These roles are for example product manager (or customer representative) architect, software designer, testing engineer and project manager.

#### 5.4.2 *Key activities*

The following lists the key activities in the workshop layer:

- planning the workshop (facilitator, secretary, workshop planning team),
- running the workshop, and
- finalizing the work not finished in the workshop.

#### 5.4.3 *Tools and techniques for the workshop layer*

**Agenda templates (workshop patterns).** Agendas from previous projects or from previous phases of the same project can be reused if found functional and feasible. Repeating successful patterns in the workshops can reduce the effort required for planning the workshops.

**CASE-tools.** CASE-tools are defined as computer aided software engineering [Sommerville 1996 Chapters 25-27]. Therefore, almost any software can be described as a CASE-tool although the applicability is very context related. Hence, there are a number of tools to choose from when using RaPiD7 workshops. Despite this, tools such as word processors and spreadsheet applications are the most typical ones used.

As one of the key ideas in RaPiD7 workshops is to write as much as possible of the final document in the workshops, it is tempting to use the tools and the templates that would be traditionally used for these tasks, too. For normal document writing, this often means a word processor application, but other applications may also be utilized. These other tools can be modeling tools, for example. Overall, RaPiD7 does not restrict the tools used, but the person (usually secretary) using the tool should be familiar with it to avoid inefficiency.

**Brainstorming and other idea gathering techniques in RaPiD7.** The literature describes several different techniques used to gather ideas, and the names of these techniques appear to refer to different approaches depending on the author. One of the perhaps best-known technique, namely brainstorming (originally suggested by [Osborn 1957]), is in social

psychology related literature often referred to as an approach to enable a team to come up with ideas on selected issues in a given timetable by presenting the ideas aloud (for example [Hewstone and Stroebe (ed) 1996] p.454). Techniques that involve an individual phase before the analysis of the ideas have been referred to as Nominal Group Techniques (NGT) (for example in [Brown 1988]). Furthermore, techniques involving textual form of recording of the ideas initially have been referred to as Brainwriting (for example in [Frey et al.1999]). In more business-oriented literature, brainstorming appears to be understood more widely perhaps due to not knowing the definitions made in the fields studying these techniques. In any case, the main purpose of brainstorming, thus, is to be able to use the whole set of ideas that a team can produce.

In RaPiD7, brainstorming is understood as any technique, involving or not involving an individual phase, aiming at generating an idea asset. The communication takes place by writing or discussing aloud. Moreover, a few brainstorming techniques, as they are applied in RaPiD7, already incorporate some analysis work. Brainstorming used in RaPiD7 have their roots in some existing brainstorming methods [Oddo (ed) 1995], but they have been tailored for RaPiD7 use. Brainstorming techniques in RaPiD7 go by names *Individual idea gathering*, *Idea walk* and *Paper Rotation*, to name a few.

**Analysis techniques.** Some of the brainstorming techniques already include a bit of analysis, and on the other hand, many of the decision-making practices require analysis work, too. The techniques needed in the analysis phase itself are not explicitly described for RaPiD7. There are many methods appropriate for use in this phase like *affinity diagram* [Oddo (ed) 1995 p.11] for grouping the generated ideas, *fishbone diagrams* [Oddo (ed) 1995 p.27] and *5-why diagram* [Eckes 2001 p.139-140] for root cause analysis and process mapping or flow charts [Oddo (ed) 1995 p.63] to create a visual presentation of an activity. Roughly, the analysis techniques can be put to four categories. These are:

- techniques for grouping the ideas,
- techniques for process mapping,
- techniques for prioritizing the ideas, and
- context related analysis techniques.

In the software engineering context one approach is to use the aid that the different software processes may provide for the analysis. If object modeling is performed, for example, the process may describe how to perform this activity. Once the candidates for the objects are listed in idea gathering, the steps the process provides for analyzing the candidates can be utilized. UML [UML 2005] can be utilized in the work, too.

**Detailed design techniques.** Detailed design means in practice that the ideas selected are concretized to the level desired so that they can be commonly understood even weeks after a workshop. Therefore, detailed design is not such a technique-oriented step, but is more concerned with human interaction aspects as well as the tools used in recording the decisions. In practice, this means an active secretary role and efficient communication between facilitator and secretary. In RaPiD7, there are no recommended techniques or tools described for this phase. However, writing the results into a selected template is strongly recommended. The CASE-tools can have an important role in this work as well.

**Decision making techniques.** There are number of decision-making practices that can be utilized in RaPiD7, such as *Authority decides*, *Compromise*, *Voting* and *Consensus building* (for example [Facilitation 2004] and [Brillhart and Galanes 1998]). The decision making techniques try to address, for example, the possible problems that relate to group decision making discussed in Chapter 3.

**Unified Modeling Language (UML).** Unified modeling language is the most recognized language for modeling software systems. It is described as “*a general-purpose visual modeling language that is used to specify, visualize, construct and document the artifacts of a software system*” [Rumbaugh et. al 1999].

The relation of RaPiD7 and UML is relatively simple and straightforward. That is, UML is simply a language that can be utilized in RaPiD7 workshops to record the decisions. Actually, UML holds the same value in RaPiD7 workshops that it holds in general. Sometimes English (or whatever language is used) is best for describing the artifacts of software systems and other times UML provides a standardized way for doing this. The benefit of UML in RaPiD7 workshops is that with a certain UML diagram participants can describe issues rather explicitly and efficiently. In natural language, the same could require hundreds of words. This can save greatly the efforts needed in capturing the decisions made in the workshops.

Use of UML as part of RaPiD7 is mostly depending on the existence of UML culture in the organization in general. The workshops cannot be used for learning UML. In addition, all the participants should be able to “talk” UML and ideally, the software systems should in general be specified and visualized by UML.

**Communication technology.** Improving communication is one of the key principles in RaPiD7, and thus ensuring good communication in the workshops is essential in implementing RaPiD7. Technology comes into play mainly if the participants of the workshop are physically in different locations. Depending on the technology used, different aspects of natural face-to-face communication are lost.

There are roughly three categories for communication tools used in RaPiD7 workshops. These are:

1. Video communication tools
2. Audio communication tools
3. Presentation communication tools

Experiences with RaPiD7 have shown that while video communication provides the possibilities for showing non-verbal clues, the quality of these is not always sufficient to justify the use of videoconference systems. Often setting up the systems is laborious, use of the systems has been relatively expensive and the quality of the communication leaves room for improvement. Other reasons are probably the simplicity of audio conferencing compared to videoconferencing and the lack of videoconference equipment. What is lost with videoconferencing are the possible ways of interacting, no matter how good quality the system is. This limits, for example, the way brainstorming, such as *idea walk*, can be performed. The same limitations apply for audio conferencing, too. Hence, a common practice with RaPiD7 is to use phone conference facilities along with the systems allowing sharing of information in electronic format.

**Other technical aids.** Technical aids beyond videoconferencing and audio conferencing equipment and facilities can be used as well. In concrete terms, these technical solutions providing help may be digital cameras, multiple data projectors and so forth. The arena of tools is rather unlimited and really depending on what is applicable and helpful.

#### 5.4.4 *Best practices for workshop layer*

**Planning a workshop agenda.** One of the key problems in creating an agenda is quite similar to estimating schedules in software projects. That is, how much effort is needed for a certain task. Typically, people tend to create agendas by first writing the start time and end time for their workshop. Then they continue by adding the issues to be handled in the desired order. The overall schedule is fixed early on and still all the desired issues find their way to the agenda. While this might work in some cases, the approach tends to have too much focus only on the overall schedule and issues to be handled. This is quite similar to software projects where the schedules and features to be developed are given by management, and designers try to come up with the kind of effort estimates that would fulfill the management's unrealistic schedules.

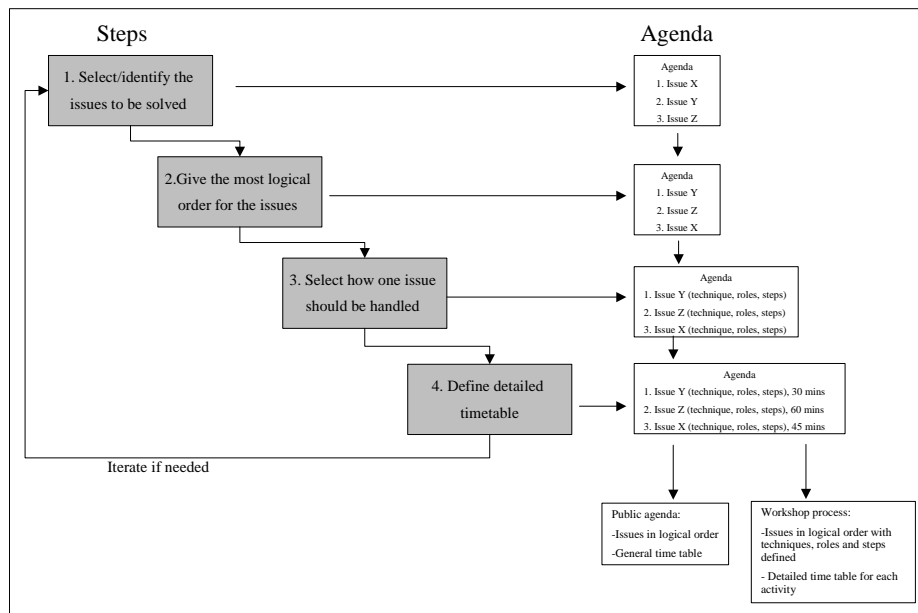
Therefore, first defining the ultimate goal to be reached in the workshop, and then defining the work products to be created on the way to the ultimate goal, can improve the planning. At this point, it is quite common to realize that the ultimate goal cannot be reached in a single workshop and in the schedule people had originally had in mind. There is usually simply too

much to discuss and decide on. Furthermore, there is another reason why schedules tend to fail. It is not always easy to determine the correct schedule by looking at the issues on a high level of abstraction without understanding the details. These problems have led to a workshop planning process in RaPiD7 resulting with two agenda types.

First, there is the actual agenda that is presented in the beginning of a workshop. This is called *the public agenda* because it is presented to everyone in the workshop. The other agenda describes quite thoroughly the issues covered and the way they are to be covered. Originally, this agenda has been referred to as *detailed agenda* in RaPiD7. Another, and perhaps more apt name for this agenda, is *workshop process*. The detailed agenda works as a guideline for the facilitator, explaining in a detailed manner *how* the workshop is carried out. The workshop process needs to cover the issues handled, but these are also recorded into the public agenda. Thus, the focus and goal of the two agendas are different and could be stated the following way:

- The public agenda explains for every participant present in the workshop what issues are to be covered and shows the timetable (start time, breaks, end time).
- The detailed agenda or workshop process describes for the facilitator how the workshop is to be carried out with details about the way certain issues are to be handled, what techniques are to be used and how much time is available for each phase.

The process of creating these different types of RaPiD7 agendas is presented in Figure 17. First, the issues to be handled are selected. After all the issues to be covered are identified, their right logical order is analyzed. There is probably no single correct order, but there are definitely better and worse orders of doing this. What is looked for is the order that appears best in the light of the possessed knowledge. In all probability, several iterations will be required. This does not become a major problem if the need for iterations has been realized early on. Therefore, the result is an estimation of the most logical order to handle the issues while jumping from one issue to another is still possible. Once the second step in the process, illustrated in Figure 17, is completed, the RaPiD7 steps from 2 to 7 can be utilized. The steps provide input for the beginning of the workshop (kick-off), the closing of the workshop and for how to handle the issues identified in the workshop itself. This is illustrated in Figure 18.



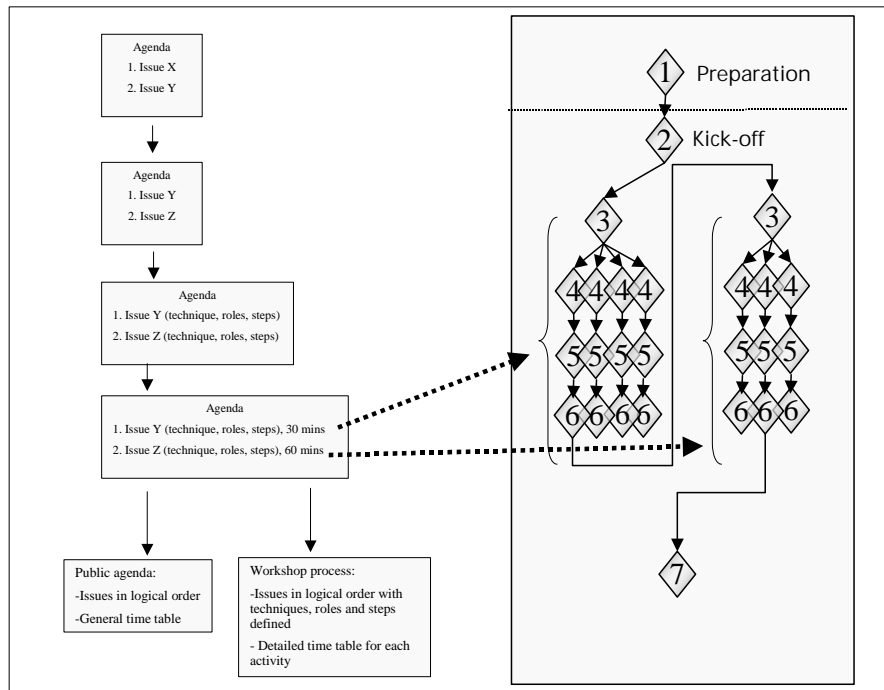
**Figure 17 Workshop agenda creation process**

Once a workshop is planned to the level of detail where even minutes are of concern, it is easier to see where things may go wrong in the schedule. In reality, the workshop is most unlikely to run according to the minute based schedule, but the workshop is less likely to go wrong by hours.

Therefore, the time used in planning is probably worthwhile. After all, when having a workshop the time wasted is multiplied by the number of participants. In addition, the possible motivation loss should not be underestimated either.

There is yet another way of trying to ensure an efficient agenda; whenever it appears difficult to foresee the outcome of a certain phase “a dummy run” can help. For example, the techniques can be tested by a dummy run with examples of real life cases. With this approach, the potential amount of ideas can be estimated. The participants can also be interviewed before the workshop. Double-checking the agenda items with the participants gives confidence in the agenda, too.

An experienced RaPiD7 practitioner does not necessarily follow the shown process of creating an agenda rigorously, at least not consciously. However, even an experienced user benefits from the workshop planning process when the workshops are either handling unfamiliar issues or the environment is somehow new (new people, new team, new organization, for example).



**Figure 18 The relation of agenda creation process and RaPiD7 steps**

**Organizing requirements workshops.** Requirements workshops can be crucially beneficial in developing early understanding within the development team. The workshops help in creating understanding of the most important features by discussing with the customer or customer representative. In Table 5 a generic workshop pattern for a requirements workshop is presented.

The kick-off in a requirements workshop covers the items that are typical to any kick-off in a RaPiD7 workshop. In addition to the general items of a kick-off, a briefing from the business point of view covering the context of use should be given. Therefore, before the requirements development can really start the context of use for the product needs to be understood. The actual items of a requirements workshop are *the actors*, *use cases* and *use case model*. Before starting to discuss the use cases in detail at least the initial list of the actors should be identified. When the actors have been identified and defined in detail, the use cases can be defined. Most likely, there are some candidates for use cases existing already from the business presentation in the kick-off. New ones (if applicable) are “invented” and existing ones are validated so that everyone understands the reasoning of the use cases. Smaller teams can concentrate on the actual details of the use cases to save time. The workshop is closed as any RaPiD7 workshop is closed.

**Table 5 Pattern for requirements workshop**

Issue	Step	Item	Approach (techniques etc.)	Input	Notes
Kick-off	2	Goal	Facilitated discussion	Preparation, customer negotiations, context of use	
		Roles of participants	Facilitated discussion	Process model, organization, project responsibilities	
		Business presentation (context of use)	Presentation & facilitated discussion	Customer negotiations, business analysis	
Actors	3	Gathering ideas for actors	Brainstorming & facilitated discussion	Context of use	
	4	Analyzing actor ideas	Presentation of ideas & facilitated discussion	Brainstorming (previous step)	
	5,6	Detailing actors and decisions	Jointly agreeing on the definitions of actors and recording the decisions	Analysis	
Use cases/use case model	3	Gathering ideas for use cases	Brainstorming & facilitated discussions	Customer negotiations, business presentation, actors	Brainstorming is not necessarily needed to come up with new use cases, but to ensure everything is covered.
	4	Analyzing use case ideas	Grouping of uses cases based on the functionality types and/or actors, removing use cases found inapplicable	Brainstorming (previous step)	
	5,6	Detailing use cases and accepting the use cases	Facilitated discussion (in sub teams), recording details of use cases	Analysis (use case groups)	Depending on the size of the workshop team and number of use cases, the detailing is performed together or in sub teams. The acceptance is always performed together.
Closing	7	Checking if the goals were reached	Facilitated discussion	The workshop	
		Checking open issues	Facilitated discussion		
		Agreeing on the continuation	Facilitated discussion		
		Analysis of the workshop and process improvement discussion	Brainstorming & facilitated discussion		

**Organizing architecture workshops.** Architecture can be explained the following way: “The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution” [IEEE Std 1471-2000]. This definition gives a starting point in identifying the items to be covered in an architecture workshop.

Another approach for identifying the issues for an architecture workshop and developing its structure is to use the existing approaches for architecture work. For, example ATAM (Architecture Tradeoff Analysis Method) provides a clear structure for the initial architecture evaluation [Clements et. al 2002]. ATAM is a structured, and thus repeatable, phase and step based architecture evaluation method. Figure 19 shows the phases and steps of ATAM. While ATAM and RaPiD7 share commonality in their structure, as well as the aim to improve the architecture, they have one clear difference in their approaches. ATAM is an evaluation method for architecture. Nevertheless, ATAM provides for the initial architecture workshop a feasible structure that can be utilized even without an existing architecture.

<p><b>Phase 1: Presentation</b></p> <p><i>Step 1: Present the ATAM:</i> In the first step, the stakeholders get a presentation of ATAM.</p> <p><i>Step 2: Present the business drivers:</i> In the second step the business drivers for the software are presented giving thus the starting point for the architecture drivers, too.</p> <p><i>Step 3: Present the architecture:</i> An architect gives a presentation about the current existing (realized) architecture or architecture plans.</p> <p><b>Phase 2: Investigation and Analysis</b></p> <p><i>Step 4: Identify the architectural approaches:</i> In this step the architect presents the possible architecture approaches, but also the evaluation group captures the possible approaches they have identified in the presentation phase.</p> <p><i>Step 5: Generate the quality attribute utility tree:</i> In step 5, based on the business drivers, the quality attributes for the architecture are elaborated. These are prioritized and scenarios for the given quality attributes are defined. These scenarios give the quality attributes a concrete context where all related stimuli and expected responses can be given. These are also prioritized.</p> <p><i>Step 6: Analyze the architectural approaches:</i> The identified architecture approaches are analyzed in their capability to fulfill the quality attributes identified from the previous step.</p> <p><b>Phase 3: Testing</b></p> <p><i>Step 7: Brainstorm and prioritize scenarios:</i> A set of scenarios is elicited from the stakeholders, and these are prioritized.</p> <p><i>Step 8: Analyze the architectural approaches:</i> This step repeats the activities from step 6, but now with prioritized scenarios from step 7 to test the analysis performed so far. New architectural approaches, risks, non-risks, and so forth, are looked for.</p> <p><i>Step 9: Present the results:</i> The findings of the previous steps are presented to the whole evaluation team including all the stakeholders.</p>
--

**Figure 19 ATAM phases and steps (reproduced from [Clements et. al 2002])**

Table 6 shows an example of an initial architecture workshop derived from ATAM steps. The RaPiD7 workshop is structured so as to follow quite closely the steps of ATAM. Naturally, the approach needs to be adjusted slightly as the work is not about evaluating something, but rather creating something new. This means that for example, the iterations are used for different reasons. Generally, architecture work is an iterative type of work. Sketching of an architecture needs to be started with a certain concrete issue such as the general architectural approach, but defining the quality attributes, for example, can impose changes in the general architectural approach and so on. In addition to architectural approaches and the attribute utility tree, as was mentioned in [IEEE Std 1471-2000], the architecture is about components (or subsystems) and relationships between them. When the general approach to architecture and the constraints are identified, the architecture work can continue in a workshop focusing on defining the components and interfaces. The pattern for this type of workshop is presented in Table 7.

**Table 6 Pattern for an initial architecture workshop**

Issue	Step	Item	Approach (techniques etc.)	Input	Notes
Kick-off	2	Goal	Discussion	Preparation, customer negotiations, context of use	
		Roles of participants	Discussion	Process model, organization, project responsibilities	
		Business and requirements presentation	Presentation & discussion	Customer negotiations, business analysis, requirements workshop	
Architectural approaches	3, 4, 5,6	Gathering ideas for architectural approaches	Brainstorming & facilitated discussion	Business and requirements presentation	
		Analyzing architectural approaches	Presentation of ideas & facilitated discussion	Brainstorming (previous step)	
		Detailing architectural approaches	Recording the details architectural approaches	Analysis	
Attribute utility tree	3, 4, 5,6	Gathering ideas for attribute utility tree	Brainstorming & facilitated discussions	Customer negotiations, business presentation, actors	
		Analyzing utility tree	Categorizing the quality attributes, excluding inapplicable ones	Brainstorming (previous step)	
		Detailing utility tree and accepting the utility tree	Depending on the size of the group and number of quality attributes this is performed together or in sub teams. Acceptance is performed jointly.	Facilitated discussion	Another iteration of architectural approach may be required. Detailing may not be required if the quality attributes can be agreed on as part of the analysis and grouping.
Scenarios	3, 4, 5,6	Gathering ideas for scenarios	Brainstorming & facilitated discussions	Customer negotiations, business presentation, actors	
		Analyzing scenarios	Facilitated discussion, prioritization of the scenarios	Brainstorming (previous step)	
		Detailing scenarios and accepting scenarios	Depending on the size of the group and number of scenarios, this is performed together or in sub teams	Facilitated discussion	Another iteration of architectural approach may be required.
Closing	7	Checking if the goals were reached	Facilitated discussion	The workshop	
		Checking open issues	Facilitated discussion		
		Agreeing on the continuation	Facilitated discussion		
		Analysis of the workshop and process improvement discussion	Brainstorming & facilitated discussion		

Again, the workshop is initiated with a kick-off. The key issues in the kick-off are the use cases, their realizations, and the output from the initial architecture workshop. The iteration concentrates on identifying the interfaces.

**Table 7 Pattern for architecture workshop (components and interfaces)**

Issue	Step	Item	Approach (techniques etc.)	Input	Notes
Kick-off	2	Goal	Discussion	Preparation, requirements workshop, first architecture workshop	
		Roles of participants	Discussion	Process model, organization, project responsibilities	
		Presentation of use cases, use case realizations and architecture utility tree and scenarios	Presentation & discussion	Requirements workshop, first architecture workshops	The first architecture workshop will give this workshop the constraints.
Interfaces	3, 4-5, 6	Gathering ideas for interfaces	Brainstorming in subgroups	Use cases, use case realizations	
		Analysing and detailing interfaces	Groups identify from use case functionality groups the interfaces (using use cases or use case realizations)	Brainstorming (previous step)	
		Accepting the interfaces	Presentations of interfaces and acceptance of use cases with facilitated discussions	Analysis (previous step)	
(High level) components	3, 4, 5,6	Gathering ideas for components	Brainstorming in subgroups	Interfaces, use cases, use case realizations	The work can start either from the interfaces or components, but the interfaces provide sometimes a more natural beginning point.
		Analyzing and detailing components	Groups identify from interfaces the logical components and detail the component definitions	Brainstorming (previous step)	
		Accepting components	Jointly agreeing on the definitions of components	Analysis (previous step)	
Overall architecture	3, 4-5, 6	Gathering ideas for connections between components	Brainstorming & facilitated discussion	Components, use cases, use case realizations	This step can be performed jointly if the number of use cases is limited (<15).
		Analyzing and detailing connections	Presentation of ideas & facilitated discussion	Brainstorming (previous step), use cases, use case realizations	
		Decisions on connections	Jointly agreeing on the definitions of components	Analysis (previous step)	
Closing	7	Checking if the goals were reached	Facilitated discussion	The workshop	
		Checking open issues	Facilitated discussion		
		Agreeing on the continuation	Facilitated discussion		
		Analysis of the workshop and process improvement discussion	Brainstorming & facilitated discussion		

The components can be investigated after the interfaces have been identified. Identifying logical groups of interfaces, for example, can be used in forming the components. Design patterns [Gamma et al. 1995] or simply architectural competence can be utilized when identifying the components. Once the components and interfaces are identified, the work can concentrate on defining which of the components utilize the services of others. This can be achieved by using the use cases and their realizations as input. If needed, the previous steps can be performed iteratively. Finally, the workshop is closed as usual.

**Organizing design workshops.** Design workshops can follow largely the same pattern that was identified for the architecture workshop in the previous section. The scope is now on the lower level components (or subsystems) their interfaces and interaction through their interfaces. Furthermore, depending on the size of a component it may be required that the work is initially addressed with a similar paradigm to the overall architecture with tailored approach from ATAM, for example.

## 6 COMPARING THE IMPLEMENTATIONS OF RELATED APPROACHES

This chapter compares the related approaches identified in Chapter 4, namely JAD and AM, with the implementation of the collaborative approach as defined in Nokia. In Chapter 4 the comparison was done only on the level of key principles. Here the comparison is done on the level of implementation in order to understand how the approaches differ in pragmatic terms. This provides understanding on the contribution of RaPiD7. The comparison uses the RaPiD7 layer structure, but covers also other identified elements for comparison.

### 6.1 Elements of the project layer

There is actually only one key activity in RaPiD7 in the project layer, scheduling the workshops. Most of the other activities can be understood as derivatives of this activity. The roles that are meaningful in the project layer are project manager and case owner roles. In Table 8 the comparison of the different approaches, and how they address these issues in the project layer are presented.

**Table 8 The project layer comparison**

Category	Item	RaPiD7	JAD	AM
Key activity/element	Explains how to plan the use as part of software projects	Yes	Partially	No
Roles	There are roles defined for project layer activities	Yes	Partially	No

**Key activities/element.** In RaPiD7 the project aspects are addressed explicitly, because this is believed to improve both the possibilities for using RaPiD7 and the efficiency in the actual use.

Neither JAD nor AM addresses the project layer extensively. For JAD this is obvious, as the initial idea of the method was only to cover requirements work. Naturally, some type of project level work can be identified in requirements work as well. Despite the fact that JAD has been used since for purposes other than requirements work, the structure provided by JAD in [Wood and Silver 1995] suggests that using JAD is, in fact, a project in itself. The first phase is called “JAD project definition”. This sounds like a somewhat heavier approach compared to that of AM or RaPiD7. Furthermore, Hughes and Cotterell, for example, mention that the workshops take three to five days conducted away from the normal environment [Hughes and Cotterell 1999, p.64]. This alone indicates something about the formality of JAD. Additionally, JAD briefly discusses handling large projects [Wood and Silver 1995 p.275-282], but the workshop scheduling process, for example, is not described.

The project layer is slightly different in the case of AM, because according to Ambler the project issues are not in the scope of AM [Ambler 2002 p.72].

However, the project layer is closely related to integrating the use of a method into software processes, and in AM, the integration to Extreme Programming (hereafter XP) [Beck 2000] and Unified Process (hereafter UP; for example [Scott 2002]) is described relatively well. This indicates, despite what is suggested by Ambler, that the project layer is described in AM to certain extent. However, the project layer and process concerns are not exactly the same matter. AM can be followed in an UP based software project without having any project planning activities related to AM, for example. This is actually more due to the nature of AM, that is to say, the approach is used when needed, and not according to any plan. Ambler mentions the following “Developers only apply AM practices minute-by-minute; you wouldn’t see tasks such as ‘Create Simple Content’ or ‘Model to Understand’ on project schedule.”

This is a clear difference to RaPiD7. While RaPiD7 is integrated into the software processes in Nokia, RaPiD7 also incorporates the project planning work. In RaPiD7, the belief is that the “use per need” is not sufficient. RaPiD7 is trying to encourage the systematic use of the method as a software engineering practice among other practices instead of an ad-hoc approach. This approach in RaPiD7 has its roots in moving quality assurance activities earlier in the product life cycle and away from inspections and even in trying to make inspections obsolete. To enable this the use of the method needs to be consistent. In fact, it is also believed that planning the possibilities for communication and information sharing is better than leave it to chance. This does not mean, however, that ad-hoc RaPiD7 workshops could not be held in addition to the ones planned. Practice has shown that scheduling meetings or workshops in a slightly larger team without planning can be difficult. This points out another difference between the approaches, that is, AM is clearly targeted for smaller teams.

**Roles.** The roles are expected to clarify the responsibilities in RaPiD7. The roles in project layer are project manager and case owner roles. Neither JAD nor AM address the roles of this layer explicitly, which is only natural as neither addresses the project layer comprehensively.

## ***6.2 Elements of the case layer***

The important elements in the case layer in RaPiD7 are the roles and the key activities. There are also a few supporting activities in the case layer. Table 9 provides the comparison of how the different approaches address the issues in the case layer.

**Table 9 The case layer comparison**

Category	Item	RaPiD7	JAD	AM
Key activity/element	Provides guidance on planning different kind of cases	Yes	Mainly to requirements work	Yes
	Provides guidance on selecting the workshop team	Yes	Yes	Partially
Supporting activity/element	Provides guidance on the number/length of the workshops	Yes	Yes	Yes
	Provides guidelines for workshop environment and facilities	Yes	Yes	Yes
	Formality of workshops varies	Yes	No	Yes
Roles	There are roles defined for case layer activities	Yes	Yes	No

**Key activities/elements.** RaPiD7 explains the different types of cases<sup>3</sup> in order to reduce the work needed per case and to improve the efficiency of planning and organizing each type of case. This enables the systematic use of the method. Selecting the right team for a case is a related issue, and works towards the same goals.

JAD, being originally a requirements oriented method, covers the case layer for the requirements work extremely well, but other types of cases are not discussed thoroughly. However, the applicability of the JAD method for other cases is mentioned [Wood and Silver 1995]. The limitation in the applicability is also visible in the structure of JAD. The phases *research* and *session* in the method cover issues such as *documenting data requirements* and *documenting business processes* [Wood and Silver 1995 Chapter 5]. On the other hand, the project approach defined by Wood and Silver gives a case structure for any work to be carried out according to JAD, providing the approach is tailored appropriately.

AM, on the other hand, is similarly to RaPiD7 more versatile when it comes to the kind of cases they provide support for. AM again uses UP and XP as examples to define the cases. RaPiD7 provides, for example, approaches for different types of workshops.

**Supportive activities/elements.** Although the number of workshops per case cannot be known absolutely surely, RaPiD7 gives guidance on estimating the number of workshops. This helps in optimizing the time used. Similarly, the guidance on workshop facilities helps in running successful workshops (these are not covered in this study when presenting RaPiD7 in Chapter 5). In RaPiD7, the workshops may vary in formality to enable use in projects of different sizes.

JAD workshops appear to always include some formality, whereas the use of AM can be either formal or ad-hoc as is the case with RaPiD7. In fact, Ambler suggests that the formal versions of AM workshops can be carried out according to JAD rules [Ambler 2002 p.140-142].

---

<sup>3</sup> These are not covered comprehensively in this dissertation.

JAD provides help in selecting the participants for the workshops and gives guidelines on the number of workshops needed. The requirements focus is visible here, too. AM provides guidelines for selecting the workshop teams, but these are not usually concrete and are such as “recruit a few good developers” [Ambler 2002 p.124] or “Recognize there is no ‘I’ in Agile” [Ambler 2002 p.126].

All the approaches address the workshop environment and facilities by describing the important aspects to consider when organizing the workshops.

**Roles.** Like in the RaPiD7 project layer, the roles are expected to clarify the responsibilities in the case layer. There are also roles described in JAD, but their responsibilities overlap with the roles defined for the case and the workshop layer in RaPiD7. In AM the case layer roles are not covered.

### 6.3 Elements of the workshop layer

The workshop layer is the layer that is described in most detail in RaPiD7 and in JAD. AM is less explicit about the workshop layer. The comparison is shown in Table 10.

**Table 10 The workshop layer comparison**

Category	Item	RaPiD7	JAD	AM
Key activity/element	Provides a workshop planning process	Yes	No	No
	Provides step based approach for the workshops	Yes	No	No
Supporting activity/element	Integrates techniques and tools for the workshops	Yes	Yes	Partially
	Provides guidelines for facilitation	Partially	Yes	No
	Provides guidelines on how to model in the sessions	Partially	Partially	Yes
	Provides guidelines on what technical aids to use and how to use them	Yes	Yes	Partially
Roles	Provides guidelines for secretary (scribe)	Yes	Yes	Partially

**Key activities/elements.** In RaPiD7 it is seen as important to provide guidance on running effective workshops. These skills are not necessarily something software engineers automatically possess.

RaPiD7 explains in more detail than JAD or AM how to plan an agenda for a workshop. This is done in RaPiD7 by providing a process for this work and utilizing the steps. In JAD the process for creating *the session agenda* is covered relatively briefly. Ambler states that in AM the practitioner can select whichever approach appears best for the workshop layer. Furthermore, neither JAD nor AM have the step approach for the

workshops, as is the case with RaPiD7. However, the structure of the workshops with these three approaches can still be quite alike.

**Supportive activities/elements.** The techniques used in the workshops, guidance on facilitation and technical aids can all improve RaPiD7 workshops. Additionally, if people are capable in modeling they are most likely more efficient in the workshop. However modeling skills are not the essence of RaPiD7, as it is believed that software engineers should learn modeling skills anyway.

JAD incorporates more facilitation aspects into the method than RaPiD7 does. AM does not appear to consider facilitation important enough to cover it in detail. JAD explains the techniques and tools for the workshops in a similar way to RaPiD7. AM covers the tools in principle by explaining that “use the simplest tool possible”, but workshop techniques are mostly left out from the method.

AM appears to be the only approach of the three that does not really cover the typical aspects of the workshop layer issues. This observation is interesting as AM covers only the case layer well, and this indicates that the focus of AM is outside the layers described by RaPiD7. By looking at the description of AM it is evident that AM is defined by its values, principles and practices. This actually makes the analysis of agile methods difficult, as the methods are very loosely defined.

**Roles.** In RaPiD7 roles are important in the workshop layer, as they clarify the responsibilities and improve the motivation of people. All the approaches describe the role of the facilitator and the secretary (scribe in JAD and AM).

#### **6.4 Other elements**

There are certain elements in all of the approaches that do not fit well the layer approach used for the analysis. These general elements are derived from the Chapters 4 and 5. These are listed in Table 11.

**Documents are inspected.** The role of inspections is clearly different in RaPiD7 compared with the other approaches. This is because the problems observed in inspections were one of the original reasons for the development of RaPiD7. Additionally, as discussed already in Chapters 1 and 2, the traditional approach for authoring specifications including inspections was only seen as a step towards a more mature document authoring approach, but not the end goal itself. JAD does not address this issue explicitly at all, but rather includes inspections as a relevant part of the approach. In AM this issue is not addressed clearly either. Ambler only states that you can review the models if you wish [Ambler 2002 p.326].

**Table 11 Analysis of other elements**

Item	RaPiD7	JAD	AM
Documents are inspected	If needed	Yes	Not in scope
Project applicability	Large to small scale projects	Large to small scale projects	Small scale projects
Document applicability	All types (also other than software related)	All types (also other than software related), focus has been on requirements	Focuses on software issues
Integration to software process is described	Yes, mostly at generic level	No	Yes, especially for UP and XP

**Project applicability.** In JAD the optimum size of the project is not explicitly mentioned. It appears, however, that the focus is on small-scale use in large-scale projects. Ambler, on the other hand, states that you are likely to run into trouble if your organization utilizes descriptive processes, is a large established firm, or if the team is large and distributed [Ambler 2002 p.314-315]. These statements clearly limit the applicability of AM. However, RaPiD7 has been used in a large company using descriptive processes in both small and large projects, and occasionally in distributed teams. This indicates RaPiD7 has been designed to be more versatile than JAD or AM in terms of project applicability.

**Document applicability.** JAD was originally developed for a relatively limited scope, but it has widened its scope later when practitioners realized the potential of the method. RaPiD7 was originally targeted for larger scope use than JAD. AM is again following the typical agile approach “use whenever feasible”. However, none of the approaches really restricts the applicability when it comes to what can be developed by the approaches.

**Integration to software processes.** Integration to software processes is most concretely described for AM with the use of the examples of UP and XP. RaPiD7 presents a generic approach for the integration, but not to the level of AM. JAD hardly recognizes that software processes even exist.

### **6.5 Summary of the comparison**

JAD being the oldest of the approaches (developed in IBM in the 1970s) was originally used to address requirements related work for the most part. JAD is especially similar to RaPiD7 in the workshop layer, but lacks some of the elements of RaPiD7 in the other two layers.

AM, on the other hand, is clearly a manifestation of the agile front and approaches the description of the methodology in the same way as agile approaches in general, that is, by the values, principles and practices. AM is in some ways the most loosely defined of the approaches. What is curious though is that the methodology is covered by a few hundreds of pages,

making it somewhat descriptive and well documented. The nature of expression is often “do whatever works” leaving great freedom for the practitioner of the methodology. The clear differences between AM when compared to RaPiD7 are in the applicability of the approaches, the lack of project integration in AM and also the lack of the workshop layer in AM.

For both JAD and AM, there is one key difference compared to RaPiD7 and this actually leads to one of the contributions of RaPiD7; a key element in RaPiD7 is to improve the software process systematically having all the planning activities in software engineering as the focus area.

## **7 EMPIRICAL EVALUATION - RaPiD7 EXPERIENCES IN NOKIA**

This chapter presents and analyzes RaPiD7 use results in Nokia. The results are viewed from two different angles. First, two deployment projects are presented and analyzed. The early deployment is a previously published case [Kylmäkoski 2003] from a unit of 1000 people and is covered only briefly. The second case is a large-scale deployment project covering a unit of approximately 5000 people. The unit sizes do not indicate the number of RaPiD7 users, but rather the sizes of the organizations the method deployment has affected. For these two projects, the results are based on surveys, and hence are opinions of people. Secondly, a software project and its RaPiD7 related results are analyzed with the help of selected metrics. The results are compared against the expected goals for the method presented in Chapter 4. The combined results from the surveys and the project metrics highlight the potential in the method.

### **7.1 *The environment***

Nokia Networks is one of the world's leading suppliers of mobile networks with net sales of 6,367 Billion Euro in 2004 and having over 16 000 employees at the end of year 2004. RaPiD7 has been used in different parts of Nokia's organization, but the focus has been in Nokia Networks and the presented results are almost solely from Nokia Networks (there are a few individuals taking part in the surveys outside Nokia Networks). The survey results have been gathered from organizations in China, Denmark, Finland, Germany and Hungary.

The processes used in the organizations are all based on the same framework provided by Nokia Networks. However, due to different types of products and market situations, the different organizations have tailored the original framework. In the majority of these cases processes followed are either a variant of the waterfall [Royce 1970] or incremental approach (for example [Scott 2002 p.7]) to software development.

The especially significant aspect for this study is the way the planning activities are addressed in Nokia Networks. Before RaPiD7 deployment, the planning activities could best be described as the traditional approach presented in Chapter 2. The traditional approach has been described in a generic form, and hence it does not provide an exact fit for any of the target organizations. For example, in some organizations workshops had been organized even before RaPiD7 deployment. However, the workshops had not usually been carefully planned and especially not planned as part of the project planning. Furthermore, in the workshops no generic format, such as the steps in RaPiD7, had been utilized. Therefore, the baseline situation was not in most of the cases as bleak as the traditional approach depicts, but the difference compared to RaPiD7 is still evident.

The environment in the early deployment faced a number of typical challenges that any deployment project would face in a business environment. The software projects have tight schedules, changes in both projects and organizations are common, and the resources for the work are limited. However, there were also positive aspects in the early deployment. Having a dedicated organization for the software process improvement was important as the deployment of RaPiD7 was being one of the key goals of the organization. Additionally, the software projects in the target organization also had the use of RaPiD7 in their incentives. As a result, the key people were trained with the mandate of the highest management in several parts of the organization.

For the larger scale deployment and for the studied software project the environment can be characterized as very challenging; there were hundreds of people and tens of teams taking the method into use in a relatively short time frame, and only a few of them got any external support aside having a few team members trained. The teams starting to use the method had the same typical challenges described earlier; including heavy time pressures in their software projects and in some cases teams also had other methods, tools or processes to be taken into use at the same time. Furthermore, in this wider deployment project the use of the method was rarely included in incentive programs, and the method usage was also not enforced in the majority of cases. In fact, the decision to take the method into use was left completely for the projects. Although this may influence positively the perceived value of the method, it made the deployment more challenging. In addition, the deployment of the method took place at the same time as Nokia Networks and the entire mobile network industry was facing an extremely challenging business situation. Hence, Nokia Networks was reducing the number of employees by several hundreds of people during the time of deploying RaPiD7. This alone had an impact on the deployment results.

The positive aspects of the larger scale deployment were that the deployment was supported from the highest possible authority in Nokia Networks and other management levels were committed to support the method use. Had the deployment been carried out, say 5 years earlier, the challenges would not have been as severe as they were. Overall, the environment that this study focuses on can be described as more challenging than typical business environment for software projects.

## ***7.2 Early deployment – unit of 1000 people***

The RaPiD7 method was developed in Nokia during 1999-2000 originally based on the feedback from software inspections. The first teams took the method into use during fall 2000. The results from the initial workshops were encouraging and the method was then taken into use in several additional teams. The focus was on R&D organizations. The results from

the early deployment have already been covered in [Kylmäkoski 2003], and therefore the following only highlights some findings from the early study.

### *7.2.1 What was measured and how?*

The very first results from the use of RaPiD7 were derived from a relatively simple assessment. People were randomly selected from all the different product lines within the organization and were asked to participate in the assessment. In the assessment, information was gathered about various elements in the software process used in the target organization. The assessment itself was a simple questionnaire with statements about the software process (see the statements for RaPiD7 in Figure 20). The statements were then answered by telling how well respondents felt the statements reflected reality in a given category. The categories were the following:

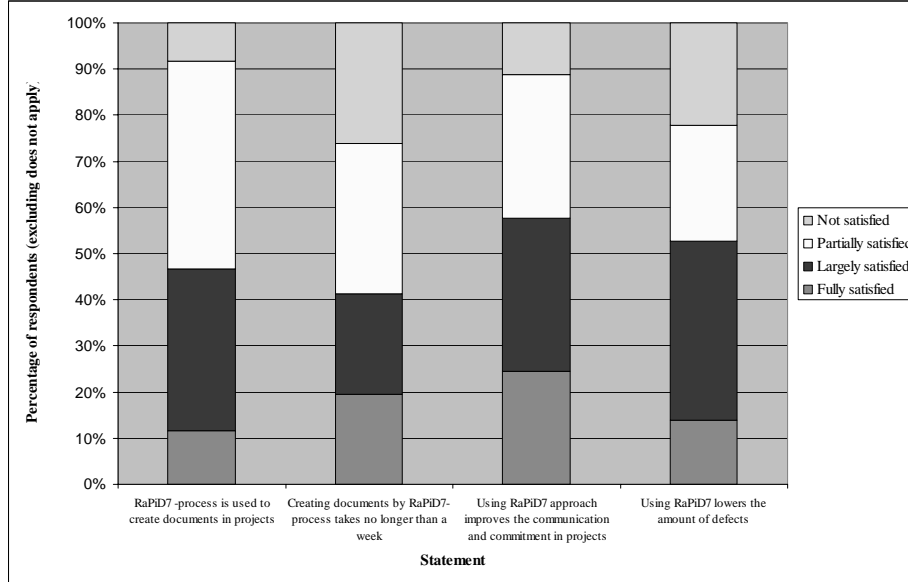
1. Fully satisfied
2. Largely satisfied
3. Partially satisfied
4. Not satisfied
5. Does not apply

### *7.2.2 Assessment results*

The results according to a software development process assessment held in 2001 are presented in Figure 20 (excluding does not apply answers)<sup>4</sup>. In statements 3 (using RaPiD7 approach improves the communication and commitment in projects) and 4 (using RaPiD7 lowers the amount of defects) the majority of respondents either fully or largely agreed with the statement. Almost everyone reports using the method to some extent, although majority report only partially agreeing with statement 1. 20% feel documents are always produced in a week, 22% in majority of the cases and 33% believe this is the case sometimes. 26% of the people believe this is never the case. The early results expressed in terms of opinions of people show very radical changes in calendar times used for document authoring. The assessment showed that 44% of people felt the documents were produced within one week when before authoring a document was possibly taking several months and typically at least several weeks.

---

<sup>4</sup> The statements are in the actual form they were used in the use survey held in Nokia. Therefore, there are different naming conventions appearing in the statements, for example, RaPiD7-process and RaPiD7 approach.



**Figure 20 Results of the early assessment**

The one-week target was originally set by the quality organization to “wake people up” to the idea that these kinds of clear improvements could be achieved. When the results from the document authoring cases were compared against the typical times used, the results showed an improvement of 15-96% depending on the case [Kylmäkoski 2003]. The calendar timesaving was considered important in the early deployment of the method, and therefore measured explicitly.

### 7.2.3 Analysis of the results

The results of the early trial indicate quite wide use and relatively significant reductions in calendar time. The results also show that people feel that communication is improved, and that the method reduces the amount of defects. However, of the early results only the calendar time efficiency was backed-up by concrete metrics from projects. Measuring calendar time efficiency is relatively easy, and indeed is probably the reason why calendar time efficiency was measured. However, calendar time efficiency is not always the most important aspect. For example, calendar time results alone tell nothing about the quality of the produced outputs. Additionally, even if the calendar time efficiency is improved in a certain case, the efficiency can be reduced elsewhere, for example. Hence, the focus shifted in the later deployment of RaPiD7 from measuring the calendar time efficiency towards measuring the total efforts and the quality aspects. Nevertheless, the early results provide a good view on the opinions of people and how they perceive the use of the method in their daily work.

### 7.3 *Large scale deployment*

After the first relatively successful deployment, the method was deployed further in Nokia Networks organization. The focus continued to be in R&D organizations although no restrictions were made on other organizational teams who also wished to take the method into use. Some of the characteristics of the results are similar to the first trial; however, more data was obtained from the second deployment stage as is presented in the following.

#### 7.3.1 *What was measured and how?*

For the large-scale deployment project a couple of ways to measure the progress and the results were established. The measures used were:

- number of people trained (and feedback from the training sessions), and
- results of use surveys.

The use surveys were organized three times during the deployment. The survey examined in this dissertation was the last one that was sent to everyone who was trained (slightly over 300 people). The last survey was selected because of its best coverage and because for the average participant a longer time had elapsed since the training. The better coverage is expected to saturate the fluctuations in the perceived results, and the longer times after the training are expected to minimize negative or positive results of the first pilot cases. The response rate was slightly over 27% for the last survey. Although better coverage was hoped for, the results present the views of almost 100 users of the method.

Some of the questions in the survey covered issues about the success of the deployment, and hence are not discussed here (for example, questions like “has the training supported the use?”). The survey questions used for the analysis are attached in the appendix. Some of them follow the same pattern that was described in section 7.2.1. In other words, people were asked if they fully agree, largely agree, partially agree or do not agree with a certain statement. This question setting was used especially when the possible benefits of the method were looked for. For the larger scale deployment, the results are presented by showing the percentages of answers given for a certain category. Furthermore, open questions and specific questions on the method elements were used as well.

#### 7.3.2 *Which elements of the method have been used?*

People were asked which elements of the method they are using. This was asked to verify how widely people use the different elements of the method, and how the method use actually realizes for the people. Therefore, these results have an impact in understanding, for example, how the benefits,

should be interpreted. The elements being heavily used could be seen as a key to certain benefits, or similarly some elements not used could be the reason for not being successful in reaching certain goals. On the other hand, using some element heavily or having poor balance between the different elements may also lead to unsatisfying results.

The survey did not emphasize on what layers the different elements asked were. The elements focused on in the survey and the corresponding results are shown in Figure 21. In the results three of the elements are well above the others. First, the steps for the workshop layer appear to be the most used element in the method. Brainstorming is used widely. In addition, the guidelines on how to plan the workshops also appear to be widely utilized. This is partially an issue of the case layer. Two other items in the workshop layer were *the writing of the document in the workshop* and the *roles in the workshop layer*. Slightly less than half of the people report utilizing these elements. Finally, the only element related to the project layer, scheduling the workshops, gets the lowest use rate. This is actually natural, as only few need to be active in the project layer. Keeping this in mind, the score (36% of people) is actually quite high.

Overall, the results show that the method means slightly different things for different people. According to the results, approximately 10% of the people report using all the asked elements of the method, 47% use four or more elements and 81% use three or more elements of the method.

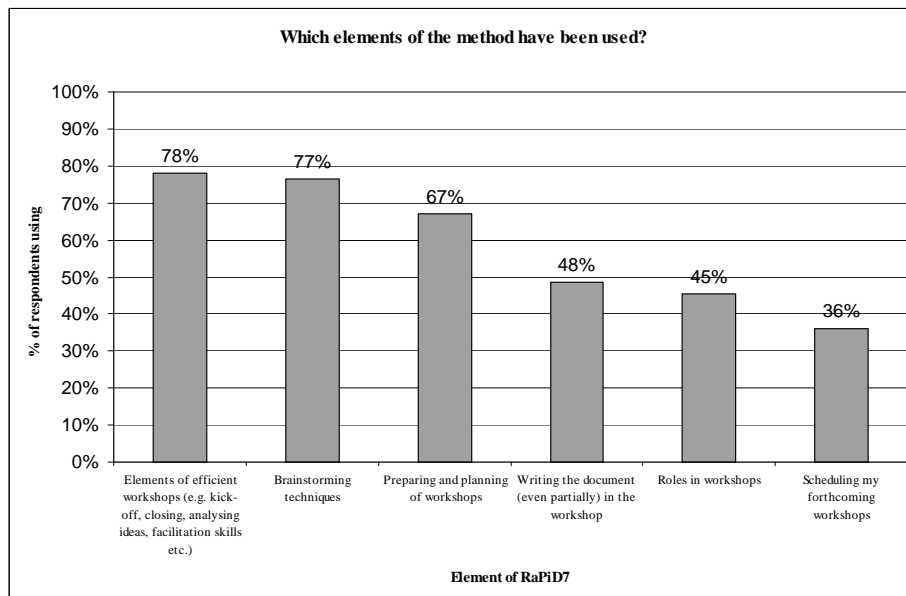


Figure 21 Using the elements of RaPiD7

One interesting issue in the light of the results is to define when someone is really using RaPiD7. Unquestionably, if only brainstorming is used, one is not using RaPiD7. However, for a project manager just scheduling the workshops may be enough. This leads to the realization that the analysis should be done on layer basis by looking at the key activities in the layers.

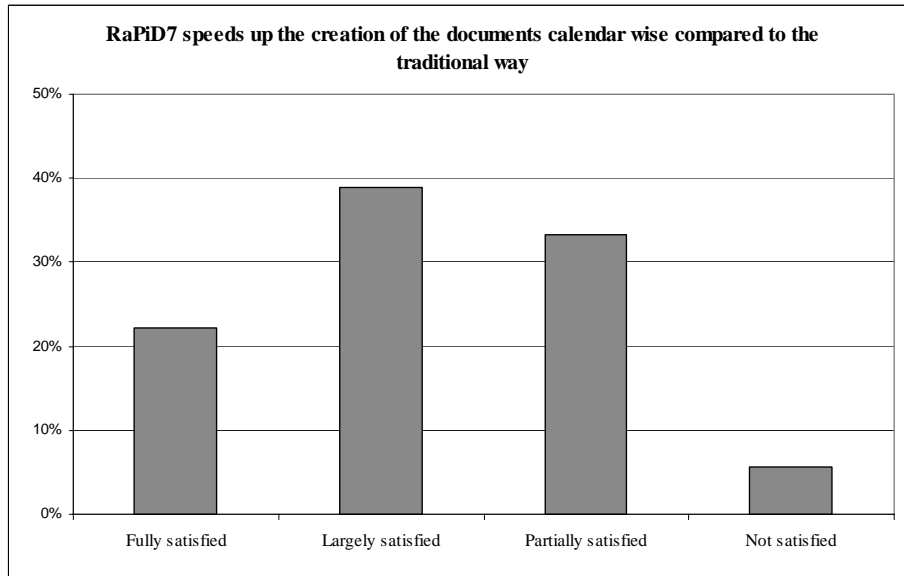
For the project layer the case is evident, as was mentioned earlier; scheduling the workshops is enough. However, approximately 5% of the respondents reporting to schedule the workshops are project managers. Almost 67% of the people using *scheduling of workshops* are people with technical roles, and the rest are people with line management (about 24%) duties or quality related roles (about 4%). This indicates either that the people with technical roles hold project management roles as well, or that the project managers have not institutionalized the use of the method as much the technical people have. The former sounds more likely as the majority of the people trained were people with technical roles.

For the case layer, preparing and planning the workshops is similarly enough to be able to say one is using RaPiD7. For the workshop layer, the roles or the elements of the efficient workshop may be a sufficient level of use, but brainstorming techniques or writing the document in the workshops are not. Furthermore, the flexibility in applying the method causes reasons for not utilizing some elements of the method. For example, in workshops of less than five people it is not always necessary to utilize the workshop roles explicitly.

To conclude, all the different layers of the method appear to be rather well utilized. The project layer is getting lower scores as there are fewer people interacting with the layer. The case layer elements are frequently used, and a clear majority uses at least two of the workshop layer elements. However, more focus should be put on utilizing the roles such as the facilitator in the workshop layer. This is important as was already pointed out in the group process analysis in Chapter 3.

### 7.3.3 *Benefits perceived*

The following statements in the survey are based on the analysis of the expected benefits of the method presented in Chapter 4. These expected benefits were also tested in the use survey. The following presents the results accompanied by a brief analysis. All the results presented use the same question format described in section 7.2.1.



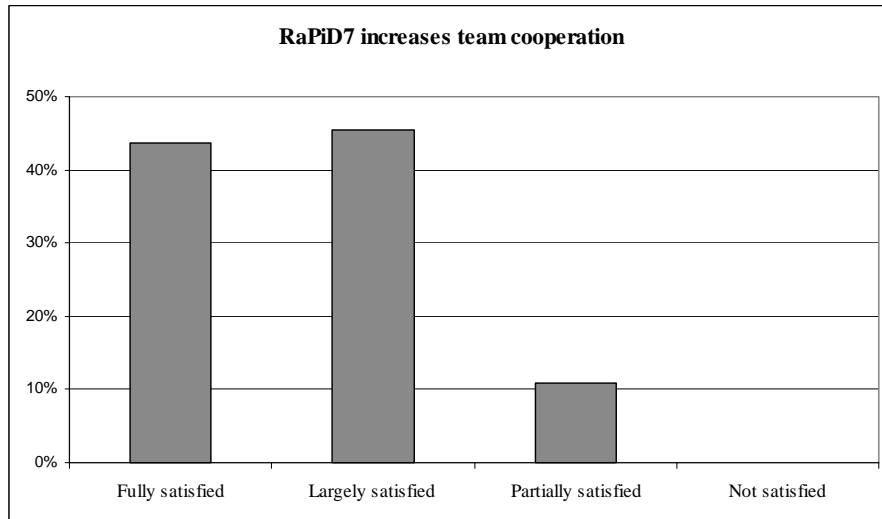
**Figure 22 Calendar time efficiency**

***1. RaPiD7 speeds up the creation of the documents calendar wise compared to the traditional way***

Clearly, the majority feels that the method shortens the calendar time needed to produce documents (94%). This is illustrated in Figure 22. However, the percentage of respondents feeling this is only partially true is significant, too (33%). Nevertheless, people that either fully or largely agree with the statement are in the majority again (61%). The result does not provide an answer to why some people felt the statement is partially true or not true. For example, the environments and documents authored may have had an influence on the results. Furthermore, this type of analysis does not provide any evidence on the skill levels of the people in general or in using RaPiD7. These may have an impact too. One clear difference to the early deployment was that the idea of getting documents ready in one week was not pushed anymore. Therefore, people may have not even considered reducing calendar time that important.

***2. RaPiD7 increases team cooperation***

For this statement, the opinions of respondents are mostly in favor of the benefit. 100% of the people feel this is at least partially true, only 11% believe this is only partially true, 45% believe the statement is largely true and 44% agree fully with the statement (Figure 23).



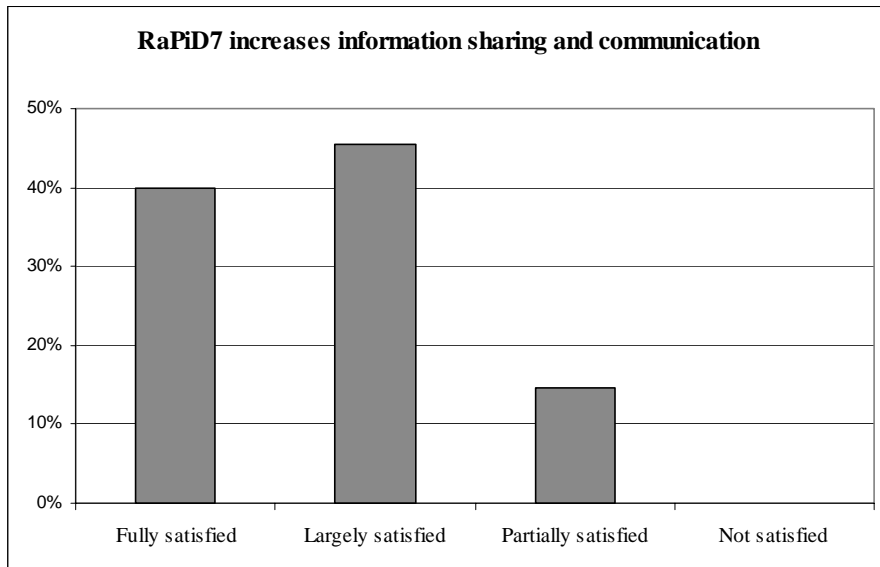
**Figure 23 Team cooperation**

Of course, this finding is expected, as RaPiD7 brings people together more than the traditional approach. Nevertheless, despite the relative obviousness of the result, the increased team cooperation holds distinct value. Organizations often use significant amounts of resources to increase team cooperation and the tasks related are not necessarily continuous everyday tasks. In addition, as was already mentioned in Chapter 3, the teams exploiting the positive process gains are often facilitated or well trained to work together. Hence, the increased team cooperation increases the efficiency of teams.

### ***3.RaPiD7 increases information sharing and communication***

The results of the information sharing and communication are very similar to the increased team cooperation answer. Firstly, 100% of people feel this statement is at least partially true, 15% feel that it is only partially true and 85% feel that it is either largely or fully true (45% largely, 40% fully) (Figure 24).

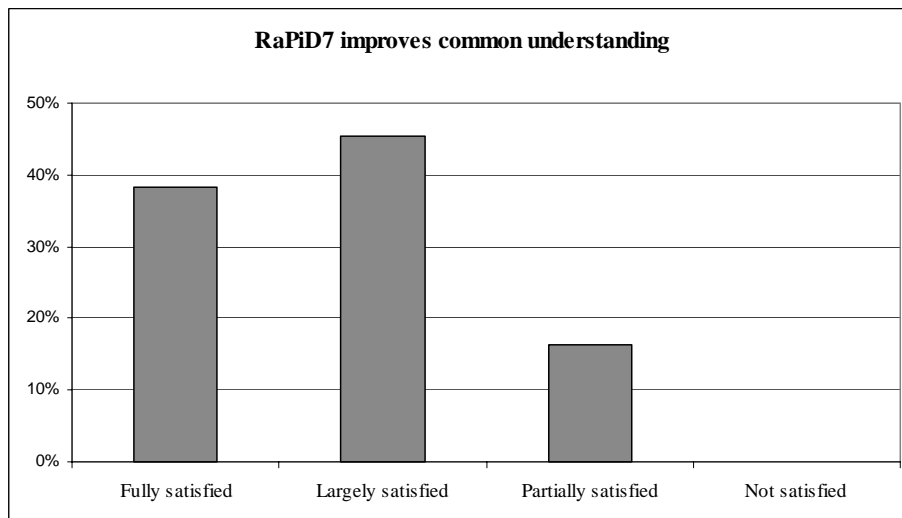
The respondents were inclined to agree with the statement and the variance in this case is insignificant. This answer speaks clearly on behalf of the increased information sharing and communication gained through the use of RaPiD7. Information sharing was one of the goals for the planning activities in software engineering that were already identified in Chapter 2. If 100% of the users of the method feel this is at least partially true, certainly something is gained for the planning activities. This is only the perceived benefit. On the other hand, while increased information sharing and communication could have been measured by metrics as well, the way people perceive this issue is also of significant importance.



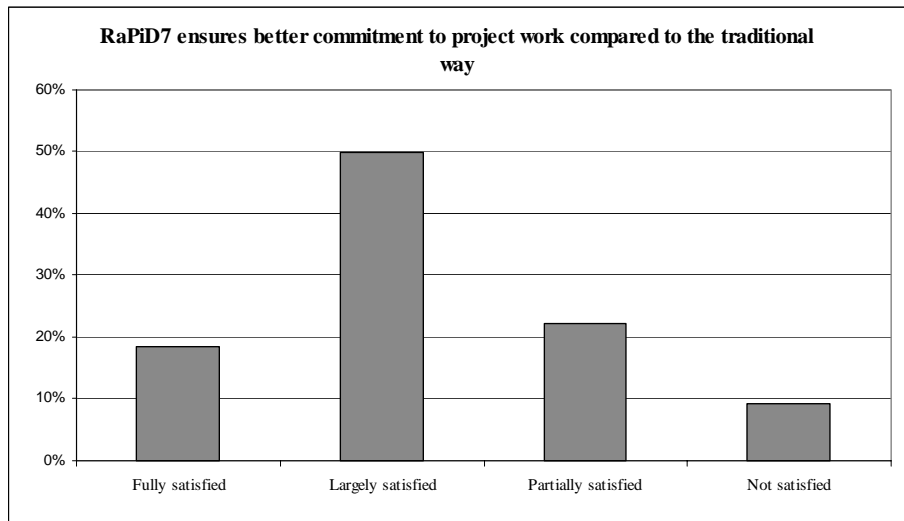
**Figure 24 Information sharing and communication**

**4. RaPiD7 improves common understanding**

The results of *improving common understanding* follow the same pattern put forward for the two previous statements. Firstly, 100% of people feel this statement is at least partially true, 16% feel it is only partially true and 83% feel it is either largely or fully true (45% largely, 38% partially) (Figure 25). This statement was looking for an answer if RaPiD7 influences reaching common agreement in the planning activities. Through better common understanding, common agreement is probably easier to reach.



**Figure 25 Common understanding**



**Figure 26 Commitment to project work**

The improved communication is contributing to improved common understanding as well. The improved common understanding could be interpreted as improved quality of information sharing.

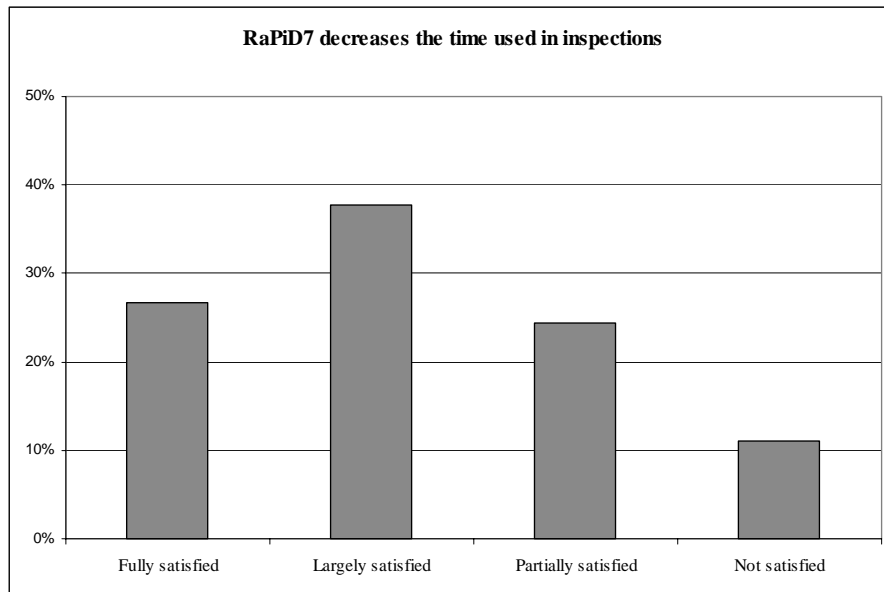
***5. RaPiD7 ensures better commitment to project work compared to the traditional way***

91% of the respondents feel RaPiD7 ensures better commitment to project work. 22% of the people feel this is only partially true while 69% of the people feel it is either largely (50%) or fully (19%) true (Figure 26).

The result is not as conclusive as it was in the case of improving information sharing or common understanding. Nevertheless, again the answers are in favor of the statement. How people perceive commitment is an important aspect. However, even when people express high commitment levels in the survey there can be other means for measuring the commitment of project members. These results could be different compared to the opinions of people.

***6. RaPiD7 decreases the time used in inspections***

89% of the respondents feel RaPiD7 decreases the time used in inspections. 24% of the people feel this is only partially true and 65% of the people feel it is either largely (38%) or fully (27%) true (Figure 27).

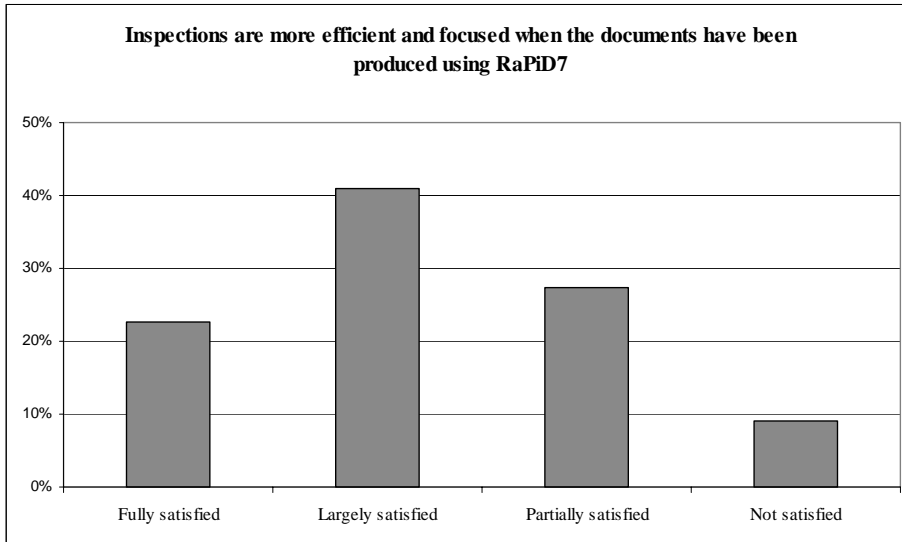


**Figure 27 Time used inspections**

The results show that the majority of survey participants feel the time used in inspections is shortened. However, the reduction is not very significant in respondents' mind. The time used in inspections was believed to reduce due to more efficient meetings, by better preparation and due to less defects being found and discussed in the inspection meetings. There are probably a few reasons why some people feel this is not the case. Firstly, the inspections may still be planned to have the same duration as in the past, and this may influence both the actual length of the inspections and how people perceive the length of the inspections. Furthermore, people are not necessarily aware which documents have been produced using RaPiD7 and which not. Hence, answering the statement may not be that easy.

***7. Inspections are more efficient and focused when the documents have been produced using RaPiD7***

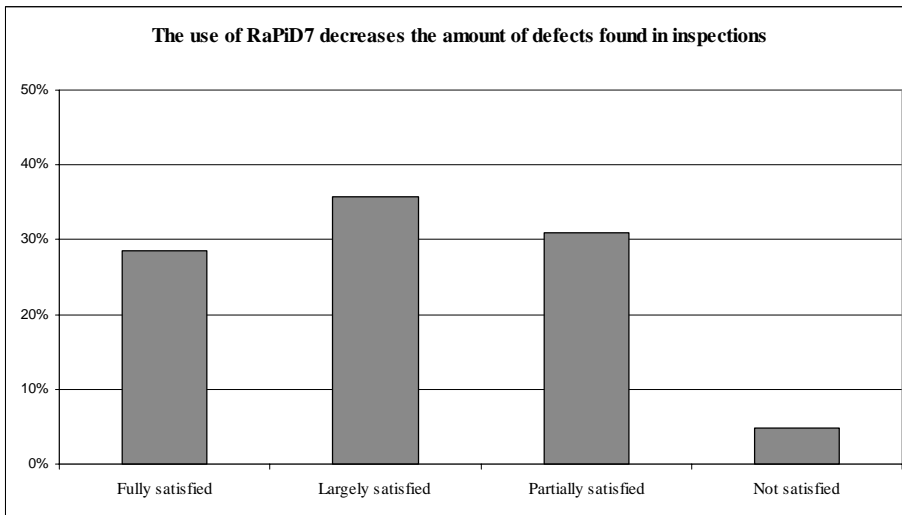
91% of the respondents feel RaPiD7 is making inspections more efficient and focused. 27% of the people feel this is only partially true while 64% of the people feel it is either largely (41%) or fully (23%) true (Figure 28). The idea behind the statement was to see if the potentially improved common understanding would also result in more efficient and focused inspections. Overall, people appear to agree with the statement. However, this is probably very difficult for people to evaluate effectively, because the connection to RaPiD7 use may not be clear. Therefore the responses are fairly evenly distributed across the field. One reason for results being more on agreeing side is that people may want to believe this statement is true, although the real life findings are not necessarily supporting this.



**Figure 28 Inspection efficiency**

**8. The use of RaPiD7 decreases the amount of defects found in inspections**

95% of the respondents feel RaPiD7 is decreasing the number of defects found in inspections. 31% of the people feel this is only partially true and 65% of the people feel it is either largely (36%) or fully (29%) true (Figure 29). The reason for the statement was to see if the quality of the planning activities is improved by reducing the number of defects found in inspections.

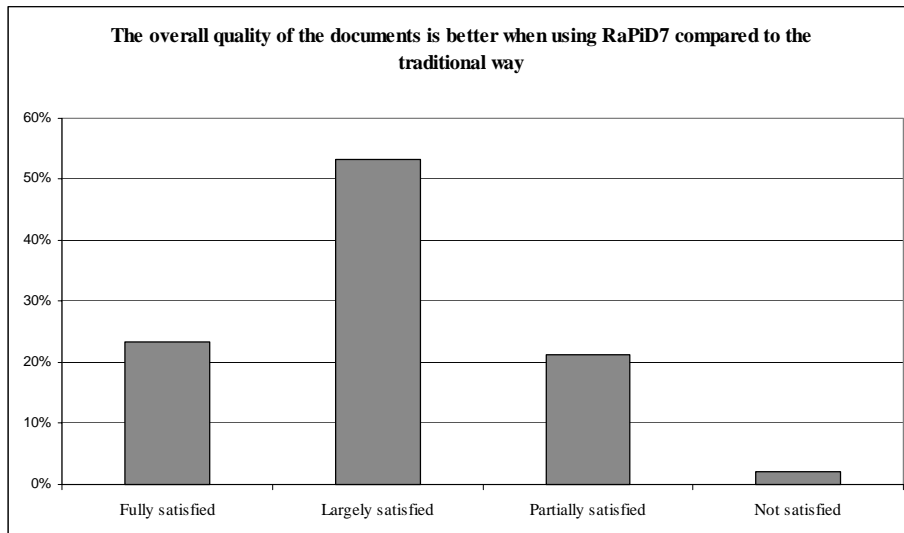


**Figure 29 Defects found in inspections**

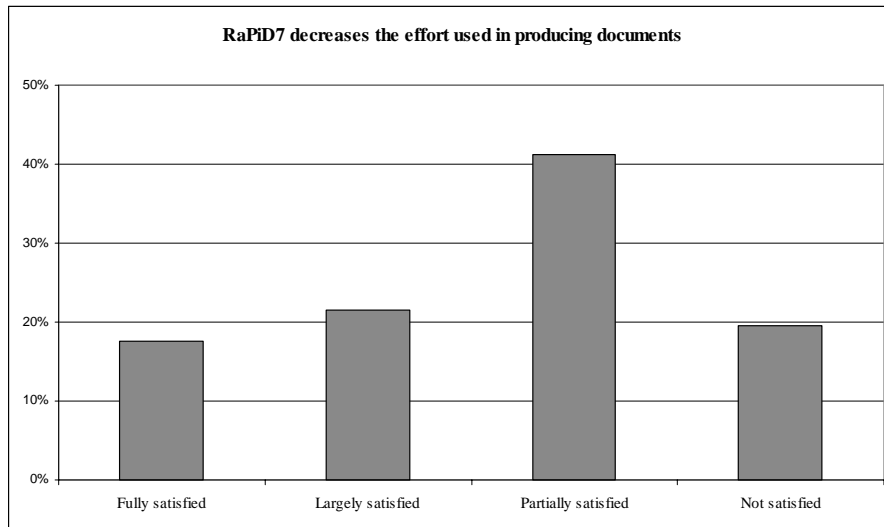
Clearly, the majority feels this is the case, however, the division in the answers tells us that not all respondents are sure the reduction is significant. This result would however indicate reduced rework time. On the other hand, the reduction in the amount of defects is not necessarily directly realized as reduced rework time. The severity of defects plays a more important role in this aspect. These results are analyzed further by the metrics results later in this chapter.

**9. The overall quality of the documents is better when using RaPiD7 compared to the traditional way**

98% of the respondents feel RaPiD7 is improving the quality of documents. 21% of the people feel this is only partially true and 86% of the people feel it is either largely (53%) or fully (28%) true (Figure 30). The quality of a document is difficult to measure, as different people understand the term quality differently. For some the quality is simply related to the number of defects found, for others it is the completeness of the document and so on. Despite the different viewpoints of quality that the respondents may have, evaluated by their personal views, the quality is clearly improving. There are probably several factors behind this. One may be that people are able to affect the document creation better during the joint activities, and hence feel that they are taking an active part in improving the document quality. Whether overall quality is really improved, in other words, the quality the customer of the software will perceive, is an issue this statement cannot give any answers for. In any case, the perceived quality of documents is important as the whole process of creating documents relies on the competence and opinions of people.



**Figure 30 Quality of documents**



**Figure 31 Efforts used in producing documents**

***10. RaPiD7 decreases the effort used in producing documents***

80% of the respondents feel RaPiD7 is decreasing the efforts in producing documents. 41% of the people feel this is only partially true while 40% of the people feel it is either largely (22%) or fully (18%) true (Figure 31). While majority also agrees with this statement, the people are far more conservative and actually majority falls into the category of partially agreeing with the statement.

Again, there are most likely several factors why this is the perceived reality. First, the matter is quite a complex to be evaluated without any facts to support the feelings. Secondly, the use of RaPiD7 is not even expected to make specification work easier as such, and actually the workshops can be perceived as a more challenging approach than the traditional approach. However, why this question was asked, and expected to some extent to be true, was because of the belief that the later the changes are made the more costly they are. Therefore, if considering the planning activities in software engineering as a whole, using RaPiD7 would reveal the needs for changes earlier and thus reduce costs, which often impact the effort, needed in the project. However, several project managers, have reported in informal discussions that using the method has actually revealed unrealistic schedules for specification work with the traditional approach and hence RaPiD7 has not reduced efforts in comparison with the unrealistic plans.

***7.3.4 The types of documents authored using RaPiD7***

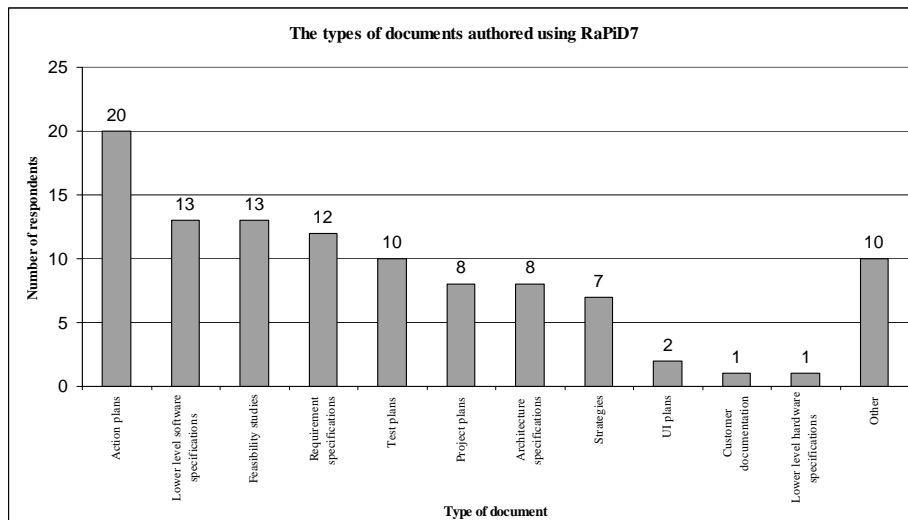
The people were asked what kind of documents they have been producing using the method (Figure 32). As people were given a chance to just check off all the possible document types they had been applying the method for, the results show the number of references to a certain document type and

not the actual number of documents produced. In other words, small number of references does not necessarily mean that the actual number of documents produced is lower than some other types of documents getting higher number of references.

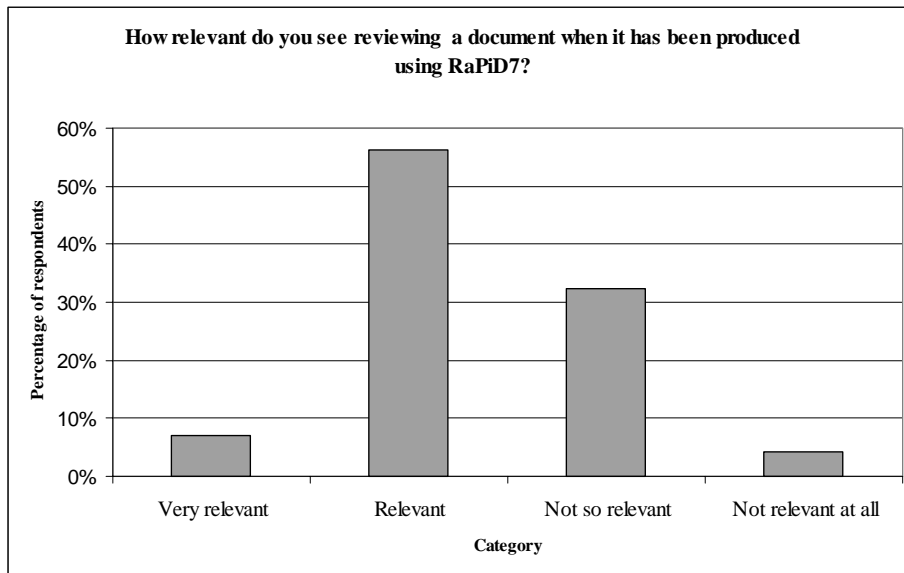
The highest number of references is for action plan documents. When considering the more typical software project documents, the results show four document types getting 10-13 references. These are lower level software specifications, feasibility studies, requirements specifications and test plans. These document types include both high level documents and low level documents. In fact, when the same results are studied from the earlier surveys held in the deployment project, these results reveal that the phases of a project appear to be a key factor in defining the types of documents created using RaPiD7.

Other documents, getting close to the same amount of references to the top four, are project plans, architecture specification and strategies. Strategy documents again are not software project documents, but rather organization related documents. Considering the relatively low number of references overall, the differences between all the mentioned document types are not significant. The ones getting lower number of references can be explained by fewer numbers of people working with these specific types of documents. Hardware specifications are an exception to this, but this most likely reflects the lower number of hardware people trained.

The conclusions from these results are scarce, but perhaps the most important finding is that there exists great range of types of documents that can be produced using the RaPiD7 method, and no clearly identifiable common denominator for these types can be found.



**Figure 32 Types of documents produced**



**Figure 33 Relevance for reviewing the documents**

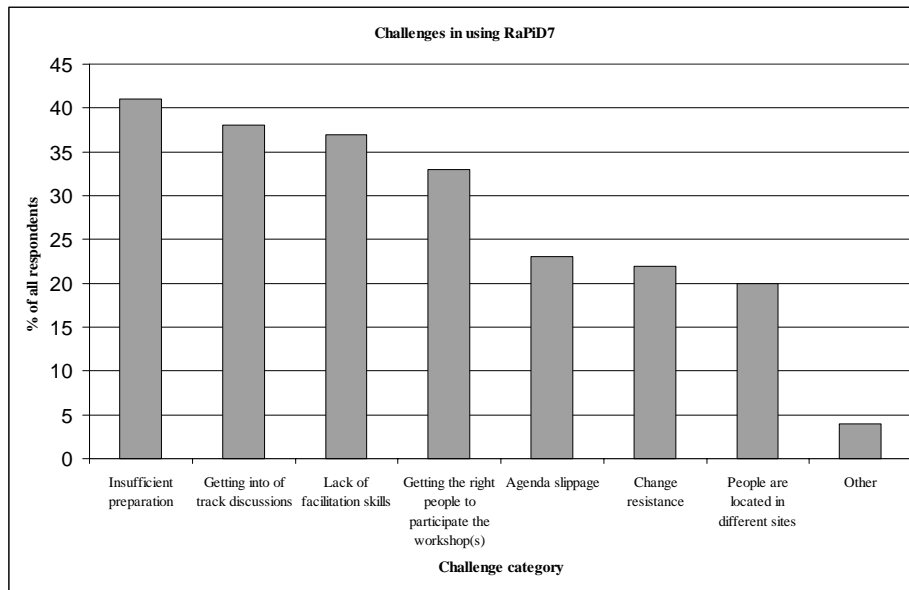
#### 7.3.5 *How are the inspections perceived?*

An initial goal for RaPiD7 was to significantly reduce the need for inspections and move the quality assurance higher up in the development stream. The results show how people saw the need for reviewing the documents when applying RaPiD7. These are illustrated in Figure 33.

Achieving the goal of making inspections obsolete does not appear to have been reached according to the respondent’s opinions, although several teams report that, the importance of inspections has dropped. A clear majority (64%) sees that reviewing documents still holds value and is needed. On the other hand, a significant minority (36%) sees the need for reviewing documents not so relevant or not relevant at all.

#### 7.3.6 *Challenges in use*

People were asked in the survey to list the major challenges in using the method. Figure 34 visualizes the answers. None of the categories gets crucially higher number of references from the respondents than others. However, insufficient preparation, getting into off-track discussions, lack of facilitation skills, and getting the right people to participate appear to be the top four challenges. Others, getting slightly lower references are *agenda slippage*, *change resistance* and *people are located in different sites*.



**Figure 34 Challenges in using RaPiD7**

These results highlight the issues anticipated or earlier identified as possible challenges in the method. The major challenges can be roughly put to two categories. These are insufficient planning and inefficient workshop implementation. These problems are interdependent. Insufficient planning may result in an inefficient workshop implementation. On the other hand, even with proper planning the lack of facilitation skills may still cause workshop problems. In fact, many of the problems mentioned in the results could be avoided by better facilitation. These are for example, *getting into off-track discussions* and *agenda slippage*.

The challenges reported by people mainly emphasize what was already expected, as similar problems are typical for any method requiring human involvement. Proper planning can never be taken for granted and people need be trained to be able to implement the workshops efficiently. On the other hand, when considering the challenges, it has to be understood that the results are not based on open questions, and hence the selection of the possible challenges may color the results.

#### **7.4 Metrics from a software project**

To have another view to the results besides opinions of people, metrics from a software project were collected, too. The project in question was a part of the large-scale deployment described earlier.

##### **7.4.1 Case description and environment**

The project team was relatively independent in their work, and thus is a good candidate for the result evaluation. The results have been gathered over a time span of approximately one year. During this time, the team was

developing a new application from scratch. Furthermore, the technology used in developing the application was new to the team, and similarly RaPiD7 itself was new to the team. Only a few trials of the method had been carried out before the project started. The project team consisted in the early phases of approximately ten people and later in the project, there were a little over 20 people working in the product development. The project team was placed in Finland in one location. The evaluation covers the time from the initiation of the project to the first customer release of the system. The project was not over at the time of writing the dissertation, but working on its fourth iteration.

Developing something from scratch, like in this case, is not necessarily the typical case in today's software engineering as software development is increasingly carried out on top of existing systems. On the other hand, when new software is being developed more understanding needs to be created, and hence more information needs to be shared, too. This provides better understanding of how the method acts in an environment when the goals for the planning activities in software engineering are essential. In addition, creating a new application from scratch is riskier in the sense that no one assuredly knows what the system should be like and therefore, for example, the effort estimation can be less accurate. In fact, the overall starting point for the project did not look that promising when the typical risks in software engineering are considered; new application was to be created with vague customer requirements, new technology was used in creating the application and new ways of working (incremental development and RaPiD7) were used in the development work.

#### 7.4.2 *What was measured and how?*

The results are approached from two different angles. First, the whole project is studied when it comes to schedules and overall success of the project. In addition, the inspection data for the whole period of the project is available.

In summary, the information collected from the whole project were as follows:

- Deviations to the original plans (schedules, implementation and testing results),
- efforts for inspections, and
- inspection findings (number of, types).

Another angle is to look at a complete iteration in the project. The iteration results provide detailed data, such as division of efforts into different types of tasks and number and types of documents produced. Again, the data from the inspections is available as well. The approach to analyze certain iteration more carefully was partially done due to practical reasons, as not all of the

metric collection practices were established early on. This approach also provides results for a certain period in the project with a clear beginning and ending. Some of the results from this iteration can be compared with the overall results. Furthermore, the use of the method was no longer new for the project team during this iteration resulting in lesser learning problems. The iteration studied is the second iteration of the project.

In summary, the information collected from the second iteration were as follows:

- efforts for all activities,
- efforts specifically for workshops (preparation, workshop, finalization),
- workshop data (number of planned, number of implemented, estimated success level)
- efforts for inspections,
- documents: types and length, and
- inspection findings (number of and types).

These results from the inspections are compared against baseline data. The baseline data has been collected from inspections over the course of a few months from the inspection tool used in Nokia Networks. The inspections in the baseline data were organized just before RaPiD7 was deployed in Nokia to exclude cases when RaPiD7 has been used in conjunction with the inspections. The only selection made when identifying the inspections is the efforts used in inspection preparations, that is, the efforts used by the RaPiD7 team and other teams for preparing to inspections are almost equal (baseline preparation efforts  $-10\%$  to  $+10\%$  range in comparison with the studied data). This selection was done in order to have comparable results from the inspections themselves.

It would be relatively natural to get different findings from the inspections if the preparation times were significantly different. Of course, it would be interesting to see if preparation efforts needed were lower when using RaPiD7, but to make this comparison possible it would require the assumption that teams always perform the preparations to the level where no further preparation would benefit the inspection meetings. As this is not typically the case, the previous selection is justified. On the other hand, the size of the projects or the complexity of the documents may vary and be different from that of the project under study. This is simply because it would have been impossible to generate a set of baseline data where these factors would have been considered as well, as this type of information was not available in the inspection tool.

In summary, the baseline data types collected were as follows:

- documents (type and length),
- efforts for inspections, and
- inspection findings (number of, types and efforts).

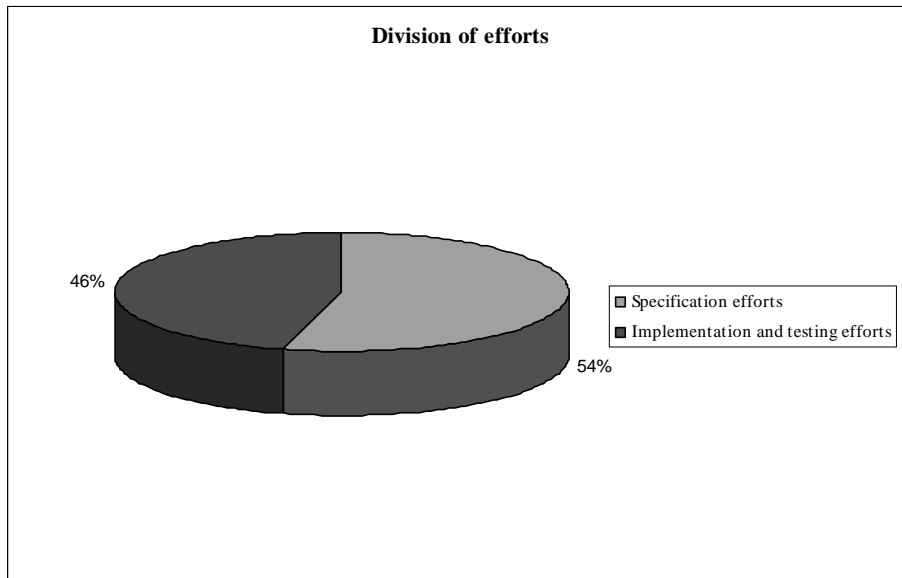
The project team has been reporting its efforts for the different activities, as is the usual procedure in Nokia. The measuring has been based on the team members' understanding of the hours used and on the defects found. In other words, no external team has been observing the team and collecting the results based on observations. As the collection of the metrics is everyone's task, the whole team has known that the metrics were collected.

#### *7.4.3 Status of the project*

At the time of writing this dissertation, the project was approximately 10 weeks delayed (excluding the influence of holidays) from the original plan, and the project had just exited from its third iteration. The delay means approximately a 15% deviation from the original plan. While the delay from the original plan clearly exists, its meaningfulness can be questioned. One of the main reasons for questioning the original timetable was that an end date for the third iteration was not originally explicitly planned. The end date was rather estimated when the original plan was made. Furthermore, as the project has been carried out in an iterative manner the functionality developed in the different iterations has changed during time. Hence, it is better to look at the results of the individual iterations with their updated timetable than the whole timetable based on the original plan.

In fact, the second iteration, the one studied in detail here, exited almost within the timetable of the original plan, and when compared to the re-planned schedule, the project was only delayed by two days. The first iteration was not delayed at all and the third iteration was delayed approximately by one month.

The most significant delays when compared to the original plan realized before entering the second iteration, and while implementing the third iteration. Before the second iteration, the delay was caused by work on the customer requirements that were not understood comprehensively and correctly. The third iteration was a customer release. Hence, it may be that unexpected issues in ensuring the quality for the customer release have caused the later delays. Discussions with the project personnel have also revealed that the future iterations may be more challenging than the early ones. In fact, the whole architecture has been questioned by the project personnel and is about to be re-analyzed in the next release of the system.



**Figure 35 Division of efforts into specification and implementation and testing efforts**

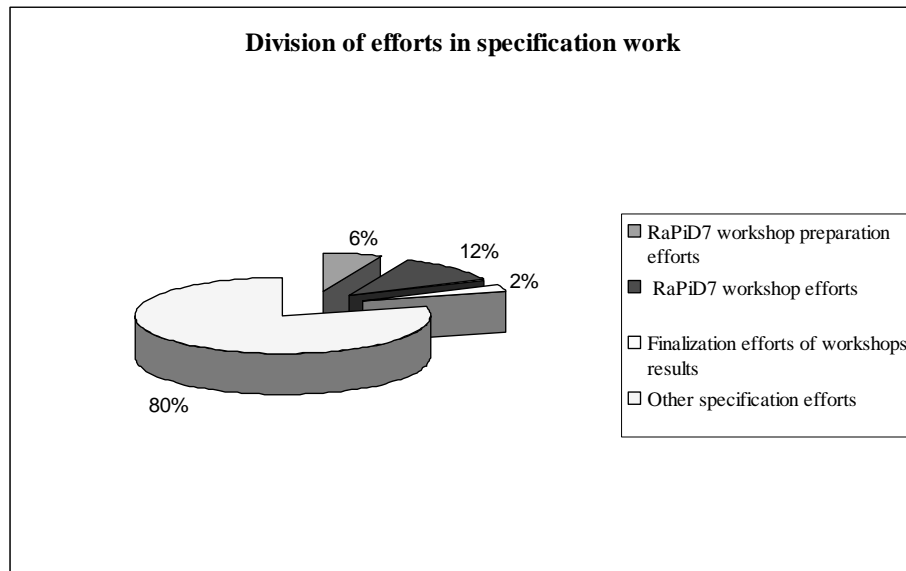
#### 7.4.4 Calendar time and effort metrics for the second iteration

The second iteration was planned to last a few days less than three and half months. This timetable was exceeded by two days. The total efforts for the iteration are 4371 hours. Figure 35 shows in percentages how these efforts have been divided into specification work, and implementation and testing efforts. The specification efforts take roughly half of the time in the iteration, and half of the efforts consist of implementation and testing tasks.

#### 7.4.5 Workshop and document metrics for the second iteration

Altogether 32 workshops were planned and 23 of these (approximately 72%) were held. The cancelled workshops were mostly back-up workshops. However, in reality more than 23 workshops were held as these figures only cover the ones officially planned and then kept. Ad-hoc workshops were held too and it is not actually certain if people have reported these as RaPiD7 use or not. Based on the discussions with the project people the practices vary. Out of the 23 planned workshops 13 were reported to have met their targets fully (52%), 6 workshops met the targets largely (24%), one did not meet the targets (4%) and for 5 workshops the results were not recorded.

Figure 36 presents the division of all specification efforts into RaPiD7 work and other specification work. The use of RaPiD7 is visible in the results, but not taking majority of time available. Use of RaPiD7 has taken approximately 20% of the time used for specification work.



**Figure 36 Division of efforts in specification work**

The number of documents produced in total in the second iteration was 14 and the different types of documents produced were interface specifications (6), subsystem specifications (5), user interface specifications (2) and a database definition (1). The total number of pages was 378.

#### 7.4.6 Elements of RaPiD7 used in the project

The elements of RaPiD7 used in the project were studied by discussing with the project members and by participating in the planning of the workshops and in the workshops themselves.

The project layer ideas were utilized comprehensively. The project layer planning was organized by collecting project level issues and subsystem level issues in separate spreadsheets and listing all the needed workshops in these spreadsheets. The spreadsheets can be understood as *artifact lists* as described for the project layer tools. The spreadsheets also covered the case layer issues such as the initial planning of the workshops and the workshop layer issues such as workshop agendas. Thus, the planning elements of all three different layers were utilized extensively.

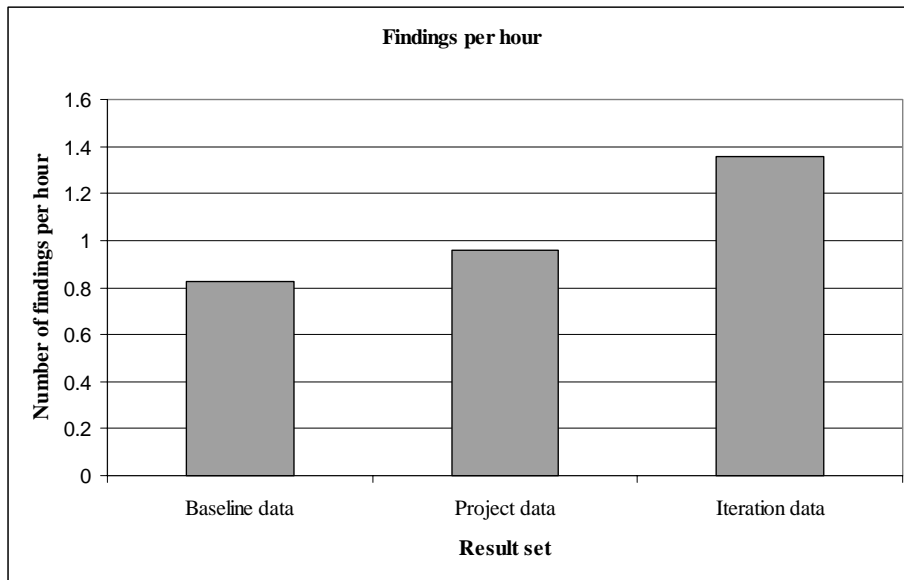
On the other hand, people often mentioned that more extensive planning of the workshops would have been helpful, although the explicit steps of RaPiD7 were not always evident in the workshops. Sometimes the roles were not utilized and brainstorming was also seldom used. Although writing the actual documents rarely happened in the workshops, relatively good decision records were kept. These were attached to the spreadsheets next to the corresponding agendas.

Overall, it appears that the project utilized the project layer issues as extensively as possible. The case layer activities were also utilized quite comprehensively although in some cases more planning of the cases would have been helpful. The workshop layer was the least utilized in terms of support from RaPiD7. Several workshops were held, but the workshop layer elements were not always utilized. There appears to be a few reasons behind this. First, the planning of the workshops is laborious, and hence is often overlooked. The team actually utilized a sort of workshop pattern in some of their workshops, but this was not consistent and therefore did not support all the workshops.

Another reason for not utilizing all the workshop layer elements is the relatively small number of participants in the workshops. As the method suggests, the formality can be and even should be, significantly lower in cases of small numbers of participants and people known to each other, thus enabling consistency in use. Thus, not utilizing all the elements in the workshop layer is justified. In fact, initially people were in the workshops without clear reasons, and therefore later the participants were more carefully selected. Some project members even questioned whether RaPiD7 was used at all, as several workshop elements did not seem to be used. This was natural as the layer structure did not explicitly exist at the time and people generally may have felt the workshop layer was all there was in the method.

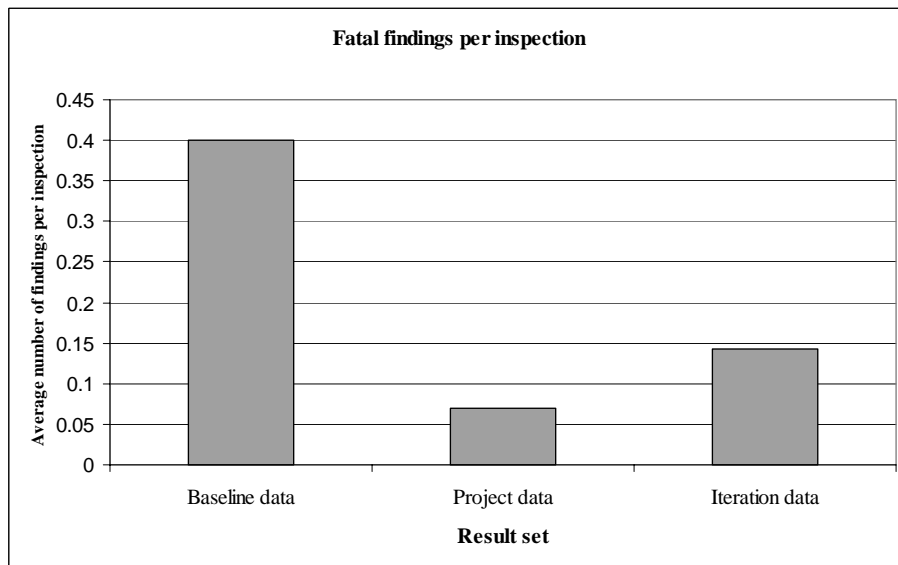
#### *7.4.7 Results from the inspections*

The results from the project inspections are compared with the baseline data (see section 7.4.2 for definition) in the following section. Both the second iteration results and the results of the whole project are presented in the figures. The rate of finding defects per hour in the project (whole project and iteration) with comparison to the baseline data is presented in Figure 37. The results indicate approximately 15% increase in finding efficiency on hourly basis for the whole project. The finding efficiency in the iteration studied is even greater (over 50%). This indicates that the use of RaPiD7 has increased the ability to find the defects in documentation. One reason behind this may be better understanding of the work of others. However, this result indicates that the number of defects is not necessarily reduced as such.

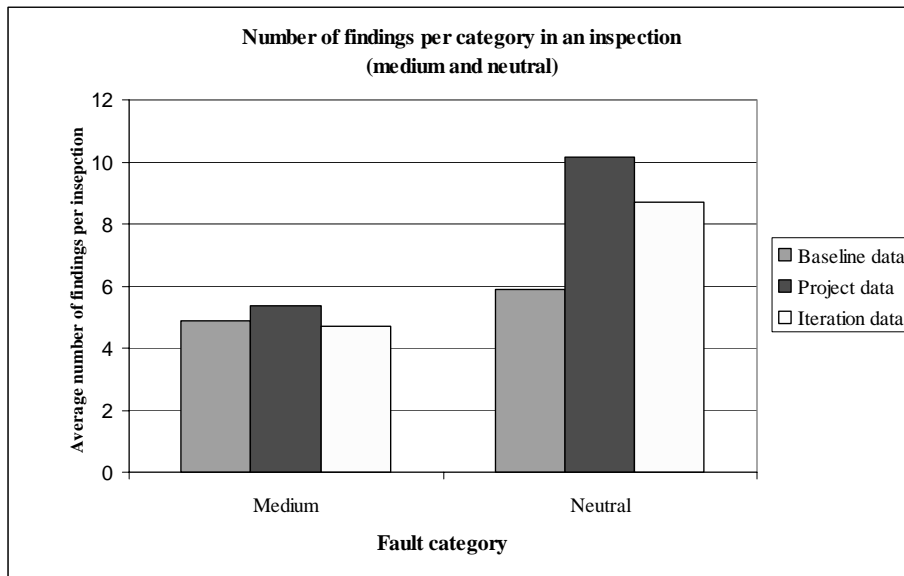


**Figure 37 Findings per hour**

Figure 38 then presents the number of fatal findings from the inspections. The total number of fatal findings in an inspection overall is not a prominent figure. The baseline data shows results of 0.4 fatal findings per inspected artifact. Hence, not all the inspections find fatal findings, although finding them has been seen as the major driver for inspections [Gilb and Graham 1993 p.75]. However, the difference to the number of fatal findings per inspection in project data and iteration data is clearly higher.



**Figure 38 Fatal findings per inspection**

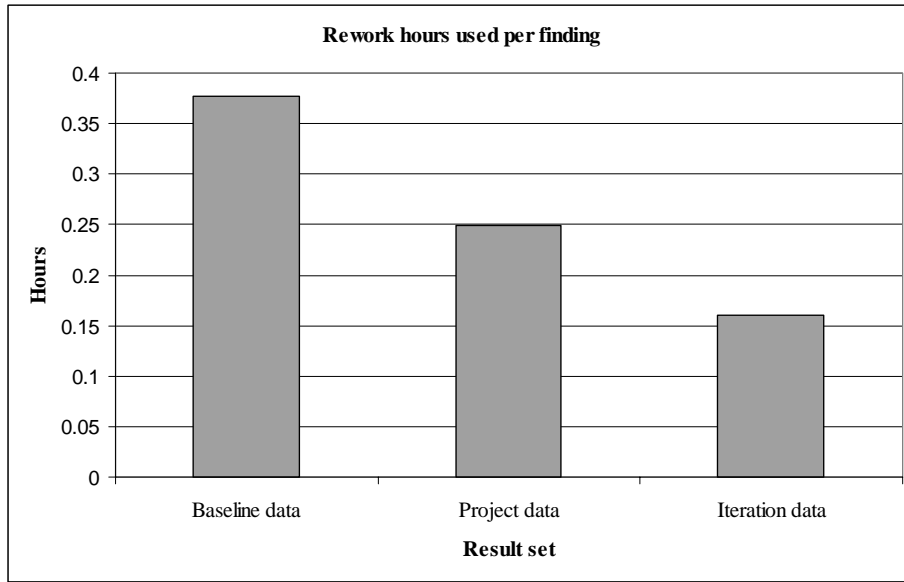


**Figure 39 Average number of findings per finding category (medium and neutral)**

There were over 6 times more fatal findings in the baseline than in the studied project as whole and almost 4 times more when compared against the studied iteration. This indicates that although the numbers of findings are not reduced, the use of RaPiD7 has essentially reduced the numbers of fatal findings.

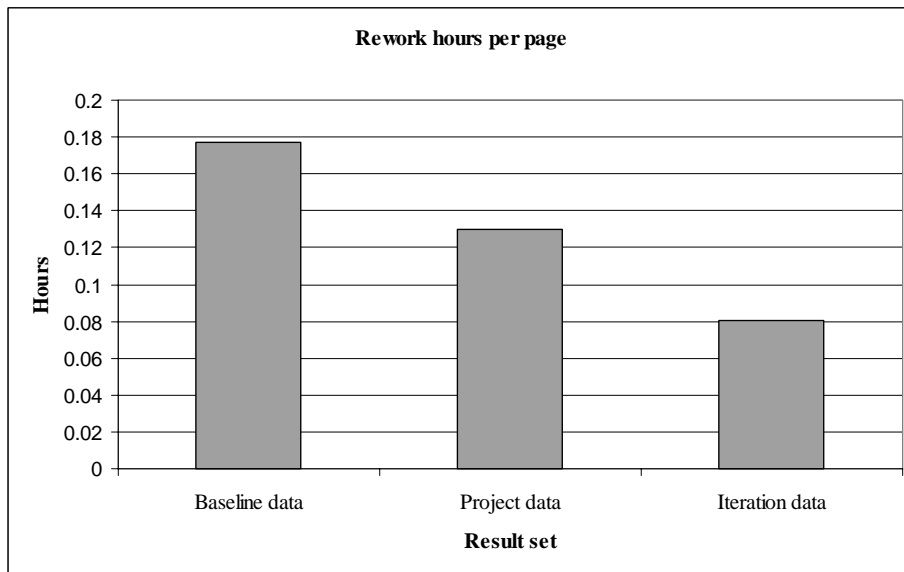
The other finding categories, medium and neutral, are illustrated in Figure 39. Interestingly the number of findings grows slightly or remains at the same level in the project using RaPiD7 when compared to the baseline data. This all suggests that the medium and the neutral findings are found in a shorter time when using RaPiD7 as the defect finding efficiency is higher. One reason might also be that the workshop approach leaves a higher number of medium and neutral errors to be found; things that perhaps one individual would polish out of his or her own document.

It is also interesting to see how the rework needed per finding changes from one approach to the other, as the changes in rework per finding will make the severity of the findings explicit. This is presented in Figure 40. The results indicate approximately 34% decrease (compared with the project data) in efforts per finding compared to the baseline data. This result matches well with the results of the finding types and the their numbers; although there are more medium and neutral findings found in the project using RaPiD7, people need less time for correcting the defects.

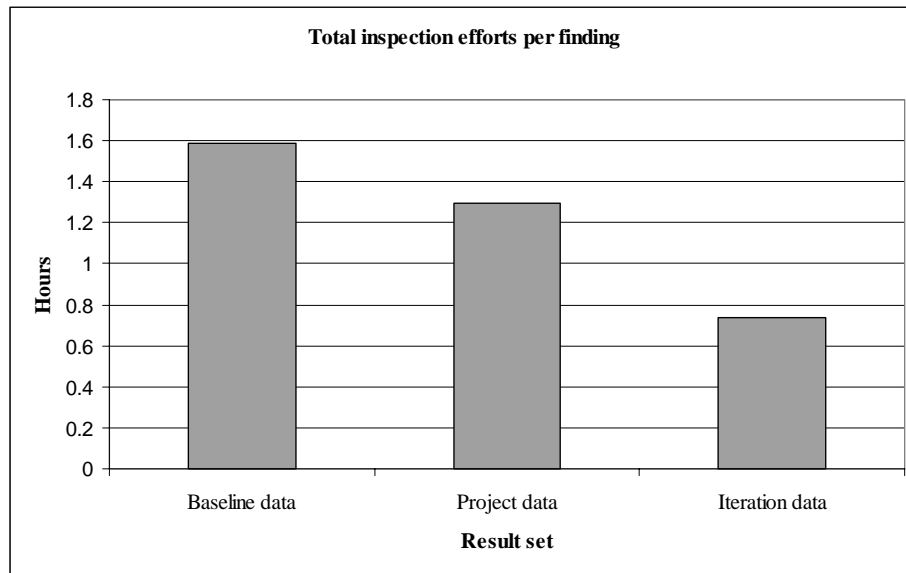


**Figure 40 Rework hours per finding**

This means that the fatal findings play an important role in the total efforts of rework. This is a naturally anticipated result. However, the findings can still be of different nature, and therefore it is interesting to see how the rework hours per page change. Figure 41 presents this result. Putting together the improved defect finding efficiency, lower number of fatal findings, higher number of findings in total and reduced rework time used per finding, it appears that use of RaPiD7 has a positive influence on the need for rework.



**Figure 41 Rework hours per page**



**Figure 42 Total inspection efforts per finding**

Figure 42 includes all the inspection efforts per finding in comparison with the baseline data. The project results indicate approximately an 18% decrease in efforts per finding compared to the baseline data. The effort decrease is over 50% when the baseline data is compared to the iteration data.

#### 7.4.8 Analysis of the results

The results cover the overall efforts for the planning activities in software engineering. These are the efforts used for specification work in general and the workshop efforts. Inspections efforts are covered also in the analysis. The workshop time, inspection preparation time, inspection meeting time and rework time were measured. The inspection related data was compared with the baseline data. The focus has been on the inspection related results. The analysis of the other elements, due to the lack of appropriate baseline data, has been an independent analysis of the results rather than a comparison.

When the overall efforts are analyzed, it appears that RaPiD7 does not make up even half of the efforts in specification work, even though RaPiD7 is used quite comprehensively. It would be interesting to see if adding the RaPiD7 related efforts were to increase the perceived benefits of RaPiD7 or vice versa. In addition, it would be interesting to see if a certain optimum could be obtained between RaPiD7 and other specification work. The effort division between planning, workshop participation and finalization show that the typical non-planning symptom is not the greatest challenge in this project, although the inadequate planning was mentioned in the interviews. However, it is not possible to tell from the data available if more planning would have resulted in better workshop results.

The results from the inspections are interesting due to being able to see the differences between studied project and the baseline data. The individual elements measured were related to *inspections preparation*, *inspections meetings* and *rework* efforts. Firstly, it is obvious that efficiency in finding the defects is better when using RaPiD7. This alone was not necessarily the result hoped for, but when the rework times per findings and per page are clearly reduced at the same time; the benefits in using RaPiD7 are evident.

An interesting finding is noted when looking at the differences in defect categories. There are clearly more defects found per inspection in the studied project. However, there are almost seven times less fatal errors found per document when using RaPiD7. Although the number of fatal findings is less than one defect per inspection in all cases, the difference between the different data sets is substantial. This is most likely why the difference in rework times can be seen so clearly. On the other hand, when the rework times are compared to the total efforts, the rework does not appear to play that big a role in the project efforts. This indicates that the reductions in rework are not as substantial as the pure percentages suggest. However, reducing the fatal findings in the inspections can have a positive influence on the number of fatal findings slipping through to implementation phase.

As mentioned, other metrics were collected, too. These metrics are for example, the overall project schedules and efforts. Overall, the results of the project are excellent keeping in mind the difficult starting point that it had. Almost anyone outside the team would have bet in favor of not keeping the schedules as well as they have been kept. In addition, significant problems in implementation and in testing could have easily been foreseen. There are numbers of factors why the project achieved these results, and while RaPiD7 was a factor in all this, it is most likely only one element leading to success. First, the incremental development has allowed the evolution of the application in the later iterations when the understanding of the requirements has improved. RaPiD7 plays a role in making the communication efficient so that making changes has been easier. When common understanding in teams is at a higher level the possibility to understand the effects of the changes is significantly better, too. In addition, the relatively limited number of members in the team and being located on one site has enabled cohesiveness in the team. Nevertheless, the evident changes in inspection findings point out quite well the contribution of RaPiD7 while other contributions are not as easily attributed to certain elements in the project environment.

## **7.5 Conclusions**

The results covered in this chapter are of two types. There are opinions of people and metrics from a project. While the metrics may feel intuitively more reliable, the issue is not that straightforward. The metrics are also often, at least to some extent, dependent on people's opinion, and hence the

metrics are always a somewhat distorted view of the actual events. This is because the metrics collection is not automated, but the metrics are based on how people report, for example, their efforts. Opinions of people are similarly likely to be biased personal views on the actual events, but there is also some value in the way people perceive the results. For example, a method may work well on paper and even provide actual results, but if people do not see the benefit of the results, the long-term use may be jeopardized. Therefore, a good approach to evaluating a method such as RaPiD7 is a combination of these two approaches when possible. If the results from these two approaches support each other, this indicates higher reliability.

There are a few aspects measured with both approaches. These are the calendar time efficiency, inspection efficiency and rework times. Let us analyze the results of these one by one.

First, the calendar time efficiency improvement perceived by people is evident. However, calendar time efficiency is not one of the major benefits perceived by people, in other words, there are other benefits people see as more important. On the other hand, the metrics show very clear improvement in the calendar time efficiency. This implies that while there is potential in RaPiD7 to get calendar time efficiency improved, people are not exploiting this possibility consistently. Another, possibility is that there is either no reason to improve calendar time efficiency as much as it could be, or there are external limitations, such as delays in related projects, in reaching the calendar time efficiency. Nevertheless, people do see that calendar time efficiency is improved and the metrics support this too.

Improving inspection efficiency was one of the anticipated secondary goals. The belief was that the use of the method would make inspections more focused. It was believed that the discussions taking place in the workshops would decrease the amount of discussion needed during the inspections. Furthermore, the assumption was that people would be more aware of the work of others, and hence be able to better contribute to the inspections. This was investigated in the survey using the statement “*Inspections are more efficient and focused when the documents have been produced using RaPiD7*”. Clearly the majority of people believe this to be either largely or fully true, although the benefit is not the most important perceived by people. This result is corroborated by the metric results, too. The metrics indicate a higher rate of defect identification in the inspections, in other words, less time is required to find a defect.

The changes in rework were not explored directly in the survey. However, there were statements examining if the number of defects is getting lower and if the overall quality of documents has improved. The metrics provide information on the rework changes, although the baseline data affects how metrics results appear. Nevertheless, all the results speak on the behalf of reduced rework. People see the reduction in the number of defects, the

overall quality improving and the metrics even show dramatic improvements in the rework times. What is interesting, however, is that the metrics actually show higher number of defects per page figure and this is in contradiction how people perceive the issue. On the other hand, the time used per defect is significantly lower and the time used per page is clearly lower, too. This goes hand in hand with the perceived quality improvement of documentation. In fact, the reason why people may believe the number of defects is getting lower is because the number of significant defects is clearly getting lower.

When considering the importance of the results from the surveys it has to be realized that the survey statements from both the early and later surveys show the method in positive light. In other words, the statements themselves indicate something of the possible benefits. Furthermore, the given categories for the possible answers are already biased towards positive feedback. In other words, three out of four categories are in favor of the statement (fully, largely and partially).

The approach in the surveys was selected because the same setting was also used in other surveys within Nokia. Although the approach is probably not scientifically the most reliable, this setting can also be justified from the process improvement point of view. After all, the surveys were part of a process improvement program and it is only natural that improvements were expected. However, using a different kind of survey, with more balanced categories would have most likely given more reliable results. Therefore, the statements achieving the majority of answers in fully satisfied and largely satisfied categories are most reliable when considering the perceived benefits of the method.

Even if the survey setting can be criticized, nearly all the expected benefits get majority of people responding to either fully satisfied or largely satisfied categories (reducing efforts in specification work is an exception to this). Therefore, even with the criticism applied, the results speak on the behalf of perceived improvements.

As a conclusion, the following lists the key findings from the results:

- The anticipated goals of the method are supported by people's opinions and supported by metrics too where available.
- The most important benefits of the method that people perceived are increased team cooperation, increased communication and information sharing and improved common understanding.
- The most important benefits in the light of the metrics are lowered defect identification efforts and lowered rework time.

These results are put together in the next chapter with the limitations of this study, goals of the method and contributions of the study.

## 8 CONCLUSIONS

In this chapter the elements of the study are brought to together by first reviewing the research problems and then describing how the contributions of the study provide answers for these. The limitations of the results are discussed and final words are presented.

### *8.1 Reviewing the research problems*

This study highlights the need of support for human interaction for the planning activities in software engineering. This study also claims that the goals for the planning activities in software engineering have not been reached. The activity goals were defined as creating understanding, sharing understanding and storing the created understanding. These were accompanied with quality attribute goals and were defined as: flawless decisions, timely decisions, and common agreement on decisions. Furthermore, the late quality assurance in the traditional approach for the planning activities in software engineering was identified as problematic.

These issues, the lack of support for human interaction, not reaching the goals for the planning activities and late quality assurance, have formed the research problem of the study. Thus, the research problem was formulated the following way:

*How does emphasis on human interaction based methods improve the planning activities in software engineering?*

The research problem was then opened up for three viewpoints:

- a) How can the process of creating understanding be improved in terms of speed and quality?
- b) How can the perceived quality of information sharing and communication be improved?
- c) How can the perceived quality of co-operation be improved?

The answers to the research problems and the different viewpoints are made explicit by the contributions of the study. These are reviewed in the following section.

### *8.2 Reviewing the contributions*

The lack of support for human interaction is tackled by providing a method addressing the planning activities in software engineering. The method is called RaPiD7. RaPiD7 can be used systematically and its applicability is not limited to certain project types. RaPiD7 enables continuous information sharing and joint decision making through the use of facilitated workshops. The workshops also allow the opportunity for continuous quality assurance activities and enable documents to be produced in a comparatively shorter

calendar time. These aspects give us the contributions of the study from the method point of view:

- Addressing the lack of human interaction by providing a method that can be
  - institutionalized in the software processes and
  - used in both small and large scale.
- Providing an approach to enable continuous quality assurance for the planning activities in software engineering.
- Providing a way to speed up as well as improve the quality of the planning activities in software engineering.

Theories are only that before they are implemented, and therefore this study has provided answers to the research problems by evaluating the experiences of industrial trials.

Looking at the goals for the planning activities in software engineering and explaining how the expected benefits of the method support these, provide understanding of the significance of the experiment results. This also provides answers to the research problem of the study. Therefore, let us analyze the goals of the planning activities one at a time in the light of the results and goals of RaPiD7.

The first of the activity goals is to create understanding. While creating understanding as such is needed, it becomes more meaningful as a goal when looked at together with the attribute goals, in other words, making timely decisions, making flawless decisions or creating understanding that everyone agrees on. If we look at the ability to create the right kind of understanding the results imply this ability has improved (reduced rework, improved overall quality of documentation, improved inspection efficiency). On the other hand, being able to make timely decisions have improved as well (reduced calendar time needed for document authoring, decreased inspection efforts, reduced rework). Finally, creating understanding that people agree on appears successful in the light of the survey results (improved common understanding, improved commitment).

Let us then continue by looking at the goal of sharing the understanding. This goal can be understood without the quality attributes. Again, it appears that sharing understanding has improved in the light of the survey results (increased information sharing and communication).

Finally, let us consider storing the created understanding. This was already believed to be addressed relatively well by the traditional approach and hence metrics were not directed to show the ability to store the created understanding. The improvement of overall quality of documentation

indicates certain aspects of this goal, but not directly. Thus, it would have been worthwhile to measure directly how well the goal of storing the created understanding was reached. This approach would have been important, because RaPiD7 changes the way the documents are created and this may have positive or negative impacts on the results.

Furthermore, the way people perceive the RaPiD7 way of working was measured indirectly. This was measured by asking how people perceived the method changed the level of team cooperation. Again, the results indicate improved team cooperation. On the other hand, having questions that would have been more directly looking at this aspect could have been helpful in analyzing the results.

To summarize the analysis, the following observations from the use of RaPiD7 in the target organization can be obtained:

1. RaPiD7 improves creating understanding in terms of quality and speed.
2. RaPiD7 improves creating understanding by enabling better common understanding.
3. RaPiD7 increases sharing of the created understanding.
4. RaPiD7 improves the way of carrying out the planning activities in software engineering by improved team cooperation.

This analysis gives us the contribution of the study based on the experiences:

- Providing evaluation of industrial experiments that support the expected benefits of the RaPiD7 approach for the planning activities in software engineering.

### **8.3 *Limitations of the study***

The limitations of the study give a better understanding of the level of generalizations that can be made based on the results. There are various issues suggesting that generalizations should be made with caution. Similarly, the research methods cause some limitations. On the other hand, there, are also factors that give more credibility to the results.

**Environment.** First, the results are from a single organization, Nokia Networks. This limits the credibility of the possible results in other environments. On the other hand, even if the results are from only one organization, it has to be understood that Nokia Networks is a very large organization including several different software cultures and environments inside one company. Although there are unified ways of working inside Nokia, there are variations in the ways of working as well. The people are located in different countries and different continents with different cultural

backgrounds and different teams are developing different kinds of products. Therefore, the results can be seen to some extent as results from several small companies (inside a large company).

Secondly, the metrics are mostly from one project. Thus, the results from the metrics should not be seen as universal or absolute truth about the benefits of RaPiD7. Several other factors influencing the results of the project could exist and this study may have not been able to capture all of them. On the other hand, while having results from one project only is a clear limitation, the opinions supporting the metrics give some credibility for the results. In addition, the early results from the companies participating in the ITEA AGILE project [ITEA AGILE 2005] have been similar to Nokia's results, and although these have not been analyzed in this dissertation, they provide additional confidence to Nokia's results (see [Dooms and Kylmäkoski 2005]).

Finally, the majority of the results are based on opinions. Opinions as such do not provide any concrete figures to proof the benefits associated with RaPiD7 and thus these should be addressed with caution. On the other hand, opinions of people cannot be neglected; a method valuable in theory may proof worthless in practice if the practitioners are not satisfied with the results.

**Research methods.** The expected and measured benefits of RaPiD7 are based on the comparison between the traditional approach presented in Chapter 2 and the collaborative approach presented in Chapter 4. The traditional approach presented here is not a complete view of reality in software engineering for the planning activities. There are variations to how the planning activities are carried out, and hence this limits the generalization of the results. However, the traditional approach presents a view that has its roots in several real life approaches for the planning activities. Therefore, the traditional approach does not present a view of single software engineering culture either.

Although the survey results are mostly positive, the survey statements are not balanced for giving objective feedback. The surveys are positively biased. This may also influence the results positively. Furthermore, it would have been clearer to identify similar issues from different elements for all the layers. These could have been the roles and tools, for example. Moreover, the survey could have been focusing on the different typical tasks in the different layers as well. This distinction would have given a better understanding of which layers in the method are used and to what extent. The way the method has been developed influenced the way the survey was defined. The workshop layer issues were the starting point of developing the method, and the case and the project layers have been natural steps forward. However, the division of the items into the different layers was a very late development. On the other hand, even when considering this limitation, the survey results are in the majority of the cases clearly positive.

Moreover, the author of this study developed the original ideas behind RaPiD7 and has been one of the key persons in its further development. In addition, the researcher has been responsible for the process improvement activities related to the method. These aspects may influence the objectivity of the researcher, and have to be understood when the results are considered.

**The method.** The method has been developed in the industry under time and funding constraints and these cause limitations on the method implementation. There are benefits in this respect as well. The fact that it has been possible to try out the ideas in practice makes the development of the method easier.

Overall, it appears justified to say at least that the method benefits can realize in similar environments with similar implementation. Credibility is also gained from the experiences of hundreds of users with different kinds of software engineering cultures.

#### **8.4 Concluding remarks**

It has been shown that RaPiD7 can provide help for the planning activities in software engineering. There are most likely challenges in effective implementation of the method as well maintaining the results. Moreover, there are new skills required from the users of the method.

The further improvement of the method would probably benefit from studies addressing the social psychology aspects of the method. In addition, more project metrics and metrics outside Nokia would be beneficial to give additional credibility to the results. In fact, some experiences have been reported already in [Dooms and Kylmäkoski 2005].

To conclude, even in its current form and with the current results RaPiD7 contributes to software engineering by amending the traditional practices for the planning work.

## REFERENCES

- Aalto et al. 1999 , Aalto Juha-Markus, Aalto Ari, Jaaksi Ari and Vättö Kimmo, *Tried & True Object Development – Industry-Proven Approaches with UML*. Cambridge University Press, 1999.
- Agile Manifesto 2004, *Agile Manifesto web page*. <http://agilemanifesto.org/>, Last visited March 2005.
- Agile modeling 2005, *Agile modeling web page*. <http://www.agilemodeling.com/>, last visited March 2005.
- Andersen 1999, Andersen Björn, *Business Process Improvement Toolbox*. ASQ Quality Press, 1999.
- Ambler 2002, Ambler Scott, *Agile modeling*. John Wiley & Sons, Inc., 2002.
- Avison 2002 Avison David, *Action Research: A Research Approach for Cooperative Work*. Conference on Computer Systems for Cooperative Work, Rio de Janeiro, September, 2002.
- Barclay and Savage 2004, Barclay K., Savage J., *Object-oriented Design with UML and Java*. Elsevier, 2004.
- Bass et al. 1998, Bass Len, Clements Paul, Kazman Rick, *Software Architecture in Practice*. Addison-Wesley Longman Inc., 1998.
- Beck 2000, Beck Kent, *Extreme programming explained: Embrace change*. Addison-Wesley publishing company, 2000.
- Bevan 1999, Bevan Nigel, *Quality in Use: Meeting User Needs for Quality*, Journal of System and Software, 49(1), p.89-96, 1999.
- Boehm and Jain 2005, Boehm B., Jain A., *An Initial Theory of Value-Based Software Engineering*, USC-CSE Technical Report 2005-505, March 2005.
- Booch et al. 1999, Booch Grady, Rumbaugh James, Jacobson Ivar, *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.
- Brillhart and Galanes 1998, Brillhart John K., Galanes Gloria J., *Effective Group Discussion*. McGraw Hill, Ninth edition, 1998.
- Brown 1988, Brown Rupert, *Group Processes – Dynamics within and between Groups*. Blackwell Publishers Inc., 1988.

- Burr 1988, Burr Vivien, *Sosiaalipsykologisia ihmiskäsityksiä*. Gummerrus Kirjapaino, 2004 (translated from The person in Social Psychology, Burr Vivien, 2002).
- Carmel et al. 1993, Carmel Erran, Whitaker Randall D., George Joey F., *PD and Joint Application Design: a transatlantic comparison*. Communications of the ACM, Volume 36, Issue 6, June 1993.
- Chesla 2000, Chesla Erik, *Successful Teamwork*. Learning Express, Inc, 2000.
- Clements et. al 2002, Clements Paul, Kaztman Rick, Klein Mark, *Evaluating Software Architectures – Methods and Case Studies*. Addison-Wesley, 2002.
- Cockburn 2002, Cockburn Alistair, *Agile Software Development*, Addison-Wesley, 2002.
- Collier et al. 1991, Collier Gary, Minton Henry L., Reynolds Graham, *Currents of Thought in American Social Psychology*. Oxford University Press, New York, 1991.
- Coughlan and Macredie 2002, Coughlan Jane, Macredie Robert D., *Effective Communication in Requirements Elicitation: A comparison of Methodologies*. Springer-Verlag London Limited, 2002.
- Curtis et. al 1988, Curtis Bill, Krasner Herb, Iscoe Neil, *A field study of the software design process for large systems*. Communications of the ACM archive, volume 31 , Issue 11, November 1988, p.1268 – 1287, 1988.
- Dennis et al. 2002, Dennis Alan, Haley Barbara Wixom, Tegarden David, 2002, *Systems Analysis and Design: An Object Oriented Approach with UML*. Wiley, 2002.
- Dooms and Kylmäkoski 2005, Dooms Ko, Kylmäkoski Roope, *Comprehensive Documentation Made Agile - Experiments with RaPiD7 in Philips*. PROFES 2005, p.224-233, 2005.
- DSDM 2004, *DSDM web pages*, <http://www.dsdm.org/>, last visited October 2004.
- Eckes 2001, Eckes George, *Six Sigma Revolution*. John Wiley, 2001.

- Facilitation 2004, *Facilitation at glance*. Goal QPC 2004.
- Fagan 1976, Fagan M.E., *Design and Code Inspections to Reduce Errors in Program Development*. IBM Systems Journal, 15, p.182-211, 1976.
- Fowler 1997, Fowler Martin, *Analysis Patterns. Reusable Object Models* Addison-Wesley, 1997.
- Frey et al.1999, Frey Lawrence R., Gouran Dennis S., Poole Marshall Scott, *The Handbook of Group Communication Theory & Research*. Sage Publications, 1999.
- Gamma et al. 1995, Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John, *Design Patterns*. Addison-Wesley, 1995.
- George et al. 2003, George Joey F., Batra Dinesh, Valacich Joseph S., Hoffer Jeffrey A , *Object-Oriented System Analysis and Design*. Prentice Hall, 2003.
- Gilb 1988, Gilb Tom, *Principles of Software engineering management*, Addison-Wesley, 1988.
- Gilb and Graham 1993, Gilb Tom and Graham Dorothy, *Software inspection*. Addison-Wesley publishing company, 1993.
- Gottesdiener 2002, Gottesdiener Ellen, *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley, 2002.
- Gregory 1977, Gregory Ruth R., *Automatic programming: Automating the software system development process*. Proceedings of the 1977 annual ACM conference. ACM press, 1977.
- Grunenwald 1989, Grunenwald William J., *Cost of quality as a baseline for Total Quality Management (TQM) implementation, Aerospace and Electronics Conference*, Proceedings of the IEEE 1989 National, p.1611 - 1613, 1989.
- Harrington 1991, Harrington James H., *Business Process Improvement – The breakthrough strategy for total quality, productivity, and competitiveness*. McGraw-Hil Inc., 1991.
- Helkama et al. 1998, Helkama Klaus, Myllyniemi Rauni, Liebkind Karmela. *Johdatus Sosiaalipsykologiaan*. Edita, 1998.

- Hewstone and Stroebe (ed) 1996, Hewstone Miles, Stroebe Wolfgang (ed), *Introduction to Social Psychology: A European Perspective*. Blackwell 1996.
- Hirokawa and Poole 1996, Hirokawa Randy Y., Poole Marshall Scott, *Communication and Group Decision Making*. Sage Publications Inc., 1996.
- Hirsjärvi et al. 2000, Hirsjärvi Sirkka, Remes Pirkko, Sajavaara Paula, *Tutki ja kirjoita*. Tammer-Paino Oy, Tampere, 5th edition, 2000.
- Horstmann 1997, Horstmann Cay, *Practical Object-Oriented Development in C++ and Java*. Wiley, 1st edition, 1997.
- Hughes and Cotterell 1999, Hughes Bob, Cotterell Mike, *Software Project Management*. McGraw-Hill, 1999.
- Humphrey 1999, Humphrey Watts S., *Introduction to the Team Software Process*. Addison-Wesley Professional, 1999.
- IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE standard 1471-2000, Approved 2000.
- ITEA AGILE 2005, *ITEA-AGILE project website*, [www.agile-itea.org](http://www.agile-itea.org), last visited January 2005.
- Jaaksi 1997, Jaaksi Ari, *Object Oriented Development of Interactive Systems*. Tampere University of Technology Publications, 1997.
- Jacobson et. al 1999, Jacobson Ivar, Booch Grady, Rumbaugh James, *The unified Software Development Process*. Addison-Wesley, 1999.
- Jacobson et. al 1992, Jacobson Ivar, Chirsterson Magnus, Jonsson Patrik, Övergaard Gunnar, *Object Oriented Software Engineering – A Use Case Driven Approach*. Addison-Wesley, 1992.
- Jalote 1984, Jalote Pankaj, *An Integrated Approach to Software Engineering*, Springer-Verlag, 1984.
- Klatt 1999, Klatt Bruce, *The Ultimate Training Workshop Handbook: A Comprehensive Guide to Leading Successful Workshops and Training Programs*. McGraw-Hill, 1999.

- Kock 2003, Kock Ned, *Action Research: Lessons Learned From a Multi-Iteration Study of Computer-Mediated Communication in Groups*, IEEE Transactions on professional communication, vol. 46, no. 2, 2003.
- Kylmäkoski 2003, Kylmäkoski Roope, *Efficient Authoring of Software Documentation Using RaPiD7*. 25th International Conference on Software Engineering - Proceedings, IEEE, 2003.
- Lewin 1946, Lewin, Kurt, "Action research and minority problems" in *Resolving Social Conflicts*, New York: Harper & Row, p.201–216, 1946.
- Macaulay 1999, Macaulay Linda A., *Seven-Layer model of the role of the facilitator in requirements engineering*. Requirements Engineering; Issue: Volume 4, Number 1, p. 38–59, Springer-Verlag London Ltd, 1999.
- McConnell 1996, McConnell Steve, *Rapid development*. Microsoft Press, 1999.
- Ochs and Van Solingen 2004, Ochs Michael, Van Solingen Rini, *Making Meetings Work*. CrossTalk, The Journal of Defense Software Engineering, 2004.
- Oddo (ed) 1995, Oddo Francine (ed), *Coaches guide to the memory jogger*. Goal QPC, 1995.
- Osborn 1957, Osborn Alex F., *Applied Imagination*. Scribner's, 1957.
- Oxford dictionary 1995, *Oxford dictionary*. Oxford University press, 1995.
- Parnas 2003, Parnas David L., *Inspection's Role in Software Quality Assurance*, IEEE Transactions on software engineering, vol. 29, no. 8, 2003.
- Patton 2001, Patton Ron, *Software Testing*. Sams Publishing, 2000.
- Poppendieck 2003, Poppendieck Mary, *Lean Software Development*, C++ Magazine Methodology Issue, 2003.
- Rational Corporation 2002, *Rational unified process home page*. [www.rational.com/products/rup/index.jsp](http://www.rational.com/products/rup/index.jsp), last visited October 2004.
- Redmond 1995, Redmond Mark V. *Interpersonal Communication – Readings in Theory and Research*. Harcourt Brace College Publisher, 1995.

- Rook 1986, Rook Paul E., *Controlling software projects*. IEEE Software Engineering Journal 1, 1, p.7-16, 1986.
- Royce 1970, Royce W. W., *Managing the Development of Large Software Systems*. Proceedings of the IEEE WESCON, p.1-9, 1970.
- Royce 1998, Royce Walker, *Software Project Management - a Unified Framework*. Addison-Wesley Longman Inc., 1998.
- Rumbaugh et. al 1999, Rumbaugh James, Jacobson Ivar, Booch Grady, *The Unified Modeling Reference Manual*. Addison Wesley Longman, Inc., 1999.
- Schalken et. al 2004, Schalken Joost, Brinkkemper Sjaak, van Vliet Hans, *Assessing the Effects of Facilitated Workshops in Requirements Engineering*. EASE 2004 Proceedings, 2004.
- Schildt 2002, Schildt Herbert, *Java 2: The Complete Reference*. McGraw-Hill Osborne Media, fifth edition, 2002.
- Scott 2002, Scott Kendall, *The Unified Process Explained*. Pearson Education Inc., 2002.
- Sommerville 1996, Sommerville Ian, *Software Engineering*. Addison-Wesley, 1996.
- Steiner 1972, Steiner Ivan, *Group processes and productivity*. New York: Academic Press, 1972.
- Stroustrup 2002, Stroustrup Bjarne, *The C++ Programming Language*. Addison-Wesley Professional, 3rd edition, 2002.
- Susman and Evered 1978, Susman G.I. and Evered R.D., *An assessment of the scientific merits of action research*. Administrative Science Quarterly, 1978.
- Taxen 2004, Taxen Gustav, *Introducing participatory design in museums*. Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices -Volume 1, 2004
- UML 2005, *UML webpages*. <http://www.uml.org/>, last visited 2005 March.
- Van Deursen and Kuipers 1999, Van Deursen, Arien, Kuipers, Tobias, *Building documentation generators*. Software Maintenance, (ICSM '99) Proceedings. IEEE International Conference, p. 40 – 49, 1999.

- Vonderembse et al. 1996, Vonderembse Mark A., White Gregory P., *Operations Management*. West, 1996.
- Wood and Silver 1995, Wood Jane, Silver Denise, *Joint Application Development*. John Wiley & Sons Inc., 1999.
- Zahran 1998, Zahran Sami, *Software Process Improvement – Practical Guidelines for Business Success*. Addison-Wesley, 1998.

## APPENDIX

### *Use survey questions (large scale deployment) used in the study*

**What parts of RaPiD7 have you used?**

- Brainstorming techniques
- Elements of efficient workshops (e.g. kick-off, closing, analysing ideas, facilitation skills etc.)
- Preparing and planning of workshops
- Roles in workshops
- Scheduling my forthcoming workshops
- Writing the document (even partially) in the workshop

**How relevant do you see reviewing a document when it has been produced using RaPiD7?**

- Very relevant
- Relevant
- Not so relevant
- Not relevant at all

**What type of documents have you created using RaPiD7?**

- Action plans
- Strategies
- Project plans
- Requirement specifications
- UI plans
- Customer documentation
- Architecture specifications
- Lower level hardware specifications
- Lower level software specifications
- Test plans
- Feasibility studies
- Other

**What kind of problems have you faced when using RaPiD7? Select THREE most important ones.**

- Insufficient preparation
- Getting the right people to participate the workshop(s)
- Getting into off track discussions
- Agenda slippage
- Lack of facilitation skills
- Change resistance
- People are located in different sites
- Other

1. RaPiD7 speeds up the creation of the documents calendar wise compared to the traditional way
2. RaPiD7 increases team cooperation
3. RaPiD7 increases information sharing and communication
4. RaPiD7 improves common understanding
5. RaPiD7 ensures better commitment to project work compared to the traditional way
6. RaPiD7 decreases the time used in inspections
7. Inspections are more efficient and focused when the documents have been produced using RaPiD7
8. The use of RaPiD7 decreases the amount of defects found in inspections
9. The overall quality of the documents is better when using RaPiD7 compared to the traditional way
10. RaPiD7 decreases the effort used in producing documents