Encouraging SME software companies to adopt software reviews in their entirety

Tervonen Ilkka and Harjumaa Lasse
Department of Information Processing Science
University of Oulu, P.O. Box 3000, FIN-90014, Oulun yliopisto, Finland
Tel: +358 8 553 1908, Fax: +358 8 553 1890,
e-mail: ilkka.tervonen@oulu.fi, lasse.harjumaa@oulu.fi

Ilkka Tervonen is a professor of software engineering at the University of Oulu specialized in software quality and quality techniques. He received his PhD in software engineering from the same university in 1994. His current research interests include software quality, software inspection, process improvement and open source software, and he heads the research team i3GO (improved inspection initiative Group in Oulu).

Lasse Harjumaa is a lecturer and PhD student at the University of Oulu. His PhD work focuses on inspection process improvement by means of process patterns. His current research interests include software inspection, agile methods and process improvement, and he is a member of the i3GO research team.

Abstract

Small and medium-sized enterprises (SME) in the field of software have recognized quality as an asset, but they tend to have rather a limited view of possible techniques. They are typically familiar with different types of testing, but are often lacking in knowledge of software reviews and inspections. This paper introduces the results of a long-term research programme conducted with SME software companies which has largely been a learning process, as we walked through the three phases from simple review process tailoring through process improvement to the analysis of motivational factors and obstacles to reviews.

We learned that although the members of the quality group may be interested in review process improvement, this does not guarantee that they can install improvements in projects. This is understandable, because project schedules are very tight and do not allow for process improvement experiments. We came to the same conclusion in our analysis of motivation factors and obstacles. Although interviewees understand that review projects can enable them to find defects earlier, faster and in greater numbers, shortage of time is the main reason which prevents them from using reviews.

1. Introduction

Small and medium-sized enterprises (SME) in the field of software have recognized quality as an asset and a necessity and are increasingly emphasizing the use of software quality techniques. They tend to have rather a limited view of the possible techniques, however. They are typically familiar with different types of testing, but often lacking a knowledge of software reviews and inspections. Even the terminology may cause confusion, and it is often reasonable to start discussions with company staff using the term review and to focus on inspections later, if needed.

Our research group, i3GO <http://www.tol.oulu.fi/i3/>, has participated since 1998 in various local projects and national initiatives focused on software reviews and inspections. In this paper we report converging results on our experiments, which can be classified into three groups, (1) tailoring the review process to more agile cases, (2) improving the review process by means of capability analysis and process patterns, and (3) analysing review techniques used in SME companies.

Our research has consisted of a learning process in which we have walked through the three phases from review process tailoring through more disciplined process improvement to the analysis of motivation factors and obstacles to reviews. We started with simple tailoring cases because companies were interested in light-weight processes and appropriate tools. Since then we have developed a number of supporting tools for inspection, evaluating their usability in a classroom context. We searched for alternative, light-weight inspection processes that could be carried out on the web, so that, as the inspection meeting was typically considered burdensome in small companies, this could be replaced with tool support. After some experiments in 1998-2000 we realized that the saying “a fool with a tool is still a fool” is very true, in the sense that introducing an inspection tool within a company that does not have an established development process will probably fail.

During the next period, from 2001 until now, we have been looking for review improvements in six companies, four of them SME companies and two similar-sized departments of nationwide companies. The work was based on a capability evaluation matrix in which capability was evaluated by means of twelve review/inspection-related activities. A very rough, generalized estimate of the results is that the review process in the SME companies has been typically at a rather low level, because they have recently started work on improving the process. In one company we introduced a set of patterns to direct the improvement tasks. The idea was that companies that desire to improve their processes often lack adequate knowledge to implement such an improvement efficiently. In our experiment the patterns worked as a concrete tool for improvement, although some examples were also needed in order to exploit the patterns fully.

A tentative analysis of twelve small and medium-sized software development departments in the Oulu area based on a short interview process has just started, the main idea being to find out the most important motivation factors and obstacles affecting the adoption if software reviews in their entirety. Another objective of the interview is to find potential companies for further review process improvement measures.

This paper introduces three approaches to the question of stimulating software companies to adopt software reviews. We first report our results in the tool development field, then our experiences with review process improvements, and finally the first results of our analysis of motivation factors and obstacles when adopting software reviews in companies.

2. Tool support for web-based inspection

Distributed software engineering projects cannot make use of the same traditional methods as do co-located teams, although their communication and quality assurance needs are the same. This is especially apparent in software inspections that have a heavy accent on teamwork. Although tools are enablers for distributed team working, they are also identified as one of the main problems in development [10]. Inspection as a form of teamwork can be a method for sharing ideas, knowledge and work between team members in different locations. The collaborative aspects of inspection are facilitated by web technology, because the web is global and work is not limited to working hours. This not only introduces flexibility into the inspection meetings (in the form of asynchronism and geographical diversity), but it also enables easy, manageable distribution of the artefacts for inspection, including the document to be inspected, checklists, or other related documents.

Even the early review/inspection tools such as CSI (Collaborative Software Inspection) [13] or CSRS (Collaborative Software Review System) [9] supported the major phases of a web-inspection process. Likewise, web-based tools and prototypes such as CoReview and its successor Group Review [5], developed at the University of Arizona, WiP (Web inspection Prototype) [6], developed at the University of Oulu, HyperCode [15], developed in Lucent Technologies, and ReviewPro [16], a commercial tool, support the inspection of text documents with WWW browsers, and the on-line commenting facilities which they provide are convenient and easy to use. These tools are similar in many aspects, as they support inspection by means of checklists, collect defect data, provide links from code lines to these data and allow readable summary reports to be produced.

Based on experiences with WiP, more comprehensive inspection tools called WiT (Web inspection Tool) and InWin (Inspection Window) [17] have been developed. The main feature of these is their ability to inspect any HTML document. The WiT approach is document-oriented and the web page itself is under inspection. The main objectives include automation of the distribution of the inspection material and the enabling of on-line viewing. The difference between the InWin tool and the WiT tool is that the individual and public inspection phases are merged in InWin. All three tools were tested by computer science students taking part in a Reviewing and Testing course at the University of Oulu in 1998-2000. Approximately 50 people used each tool in a distributed manner (typically from home or company sites) to inspect a requirements specification document. All of them had previously performed a traditional inspection with a logging meeting on a similar document. One rather surprising result of the InWin experiment was that students did not like to start the inspection of a product document with marked issues, but preferred starting without other people's markings.

A brief period of experimentation with the InWin tool has been arranged in a company developing Internet applications, where development takes place at a fast pace, the teams are working extremely dynamically and independently of each other in a hectic environment and there is not necessarily time to carry out the whole inspection cycle. Thus InWin was a more suitable option. The focus of the experiment was on the role of the inspection moderator.

The organisation in which the tool was tested did not have software inspections mentioned at all in its process model before the introduction of the InWin system, nor had it ever done any inspections. The potential of the method had been noticed, however, and the staff were willing to implement inspections on condition that the process was tailored to be more light-weight and flexible than traditional software inspections with face-to-face meetings.

While a one-step inspection is very simple to prepare and manage, it still strengthens communication between developers, thus making the development more transparent and public among the team members. Furthermore, it is easier to make use of outside expertise in the case of web inspections, even requesting comments from people who may be working in other branches of the organisation. The features of the tool that were most readily appreciated in this experiment, were its adaptability to a variety of operating system platforms and its attachment to a web browser. We also realised, however, that the job of moderator may become quite burdensome, as happened in our experiment.

This experiment in the real world led our development work in a new direction, and we understood that we should place more emphasis on rich functionality, flexibility and integration. Rich functionality in this context means functions that typically support the needs of the inspection, such as distribution of the material electronically, facilities for recording and attaching comments (suspended defects) directly in the document, classification of defects according to their severity and category, checklist management and reporting of inspection efficiency. Flexibility, a characteristic already implemented in earlier tools, means independence of time and place combined with tailorability. Integration of the inspection tool also required more concern, because all supporting tools, such as existing data repositories, authoring tools, version management systems, collaborative systems and project management tools, should work together. To demonstrate and evaluate the interoperability aspects in particular, we have implemented a new supporting tool XATI (XML Annotation Tool for Inspection) [8], which supports the inspection of artefacts with varying structures. Suspended defects are attached directly to the structures concerned and recorded in the database to await further handling.

We recognized that it is fairly difficult to introduce companies to a new tool. This is understandable, because companies strive to keep their development process alive and try to avoid all disturbing factors. This was the reason for experimenting with Adobe Acrobat [7], an off-the-shelf product that supports PDF (Portable Document Format) and may thus be of interest from a company viewpoint. It also allows annotations to be shared between team members and thus enables asynchronous discussion of the issues detected. This is useful if the details of the issue (its severity, for example) have not been agreed on during the first reviewing iteration. Both XATI and Adobe Acrobat have been tested with students in the same way as our earlier tools.

All in all, even though the functionality and ease of use of the tools has improved during the development cycle, it is a somewhat challenging task to induce SME companies to make real use of them. It is more plausible that companies will attempt to use existing tools and tailor them to support reviews as well as possible. This means utilizing the characteristics of the Microsoft Office toolset and Adobe Acrobat, for example.

3. Review process improvement

There are many accepted process capability determination models in the area of software engineering, including the Capability Maturity Model (CMM) [14], BOOTSTRAP [12] and SPICE [2]. In addition, there are also testing-tailored models such as the Testing Maturity Model (TMM) [1] and Test Process Improvement Model (TPI) [11]. From a review/inspection process improvement viewpoint none of the general or testing-tailored models is sufficiently oriented towrds inspection evaluation, although BOOTSTRAP and SPICE allow focused process assessment and improvement, i.e. the possibility to choose the process for improvement (e.g. review/inspection).

3.1 Capability evaluation in SME companies

We introduced a capability model tailored especially to review/inspection process evaluation that looks for weak points in a review/inspection through indicators and activities. The indicators can be enabling or verifying ones. The idea is that, if we find indicators of an activity, we then have justifications for its existence. If some activities are missing or at a low level, they should be set as targets for inspection process improvement. The improvement activities are usually based on discussions with company staff. We have also generated a preliminary set of improvement patterns by means of which companies can find improvement activities more easily.

Inspection is traditionally defined in terms of steps such as entry, planning, kick-off meeting, individual inspection, logging (inspection) meeting, edit, follow up, exit and release [3, 4]. The structure of the ideal inspection process, which we use as a reference model in evaluation, is based on these steps. Inspection activities are classified into three sets (cf. Figure 1): supporting activities (at the bottom), which help in carrying out an instance of the inspection process, the core set of activities (the middle bar), which are the essence of the inspection process implementation (defined in inspection books, e.g. [4]), and organisational activities, which ensure continuous improvement and efficient organisation of the inspection process.

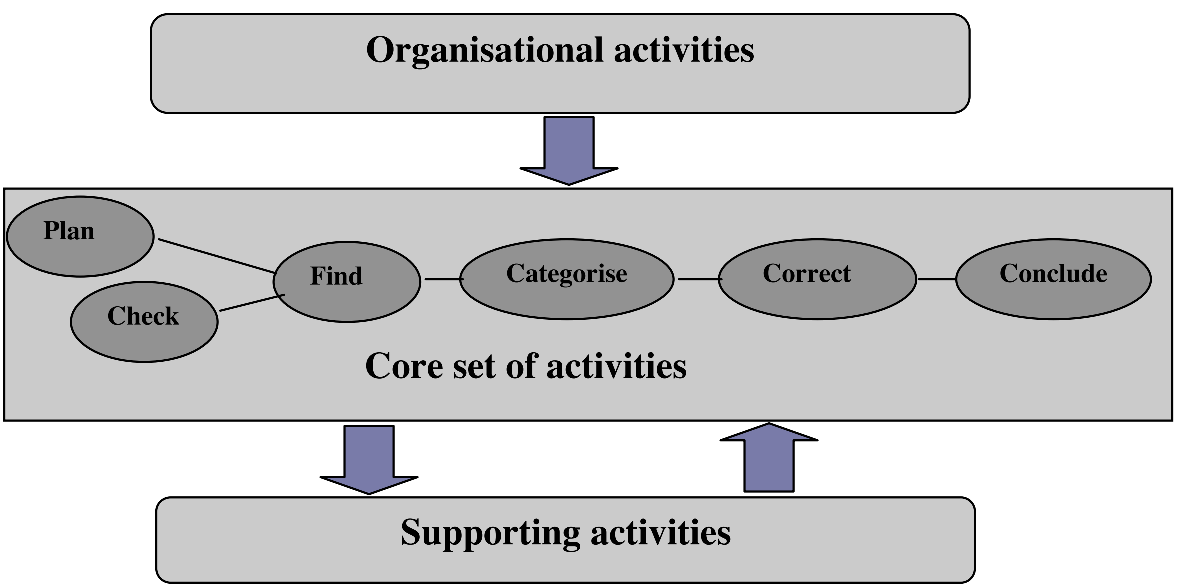
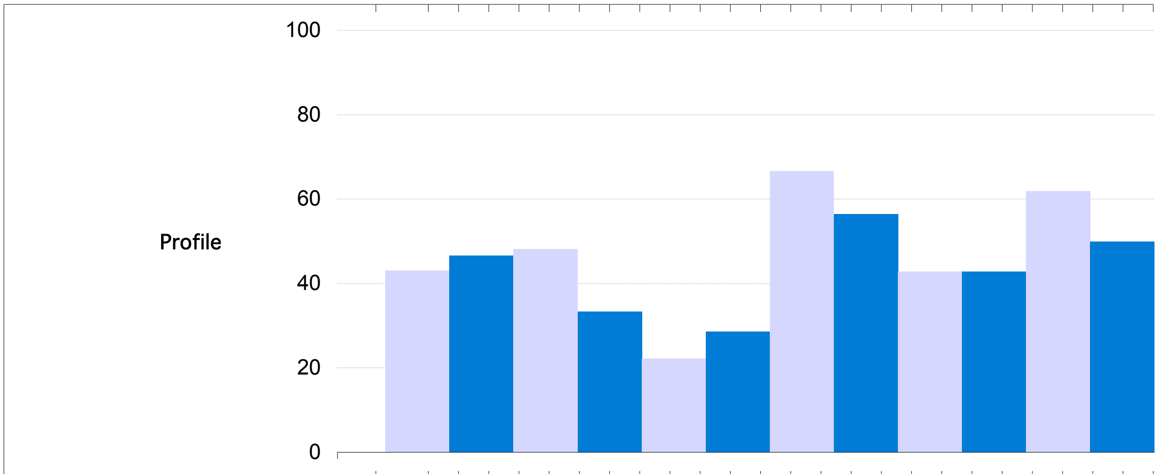


Figure 1. The i3GO capability model for software inspections

The supporting activities are “A.4 Support with computer tools", " A.5 Maintain rules and checklists" and "A.6 Refine information”. The core set of activities includes "A.7 **Check** the preconditions for inspection", " A.8 **Plan** the inspection", " A.9 **Find** issues in the artefact", "A.10 **Categorise** defects", "A.11 Make **correct**ions" and "A.12 **Conclude** the inspection", while the organisational set consists of the rest of the activities, such as “A.1 Establish and improve the inspection process “, "A.2 Organise the inspection" and “A.3 Train the participants". The A.N numbering refers to the numbers of the activities in Figure 2.

The result of a capability evaluation performed in an IT company is also presented in Figure 2. The diagram at the top of the matrix shows an estimate of each activity in graphical form, and the estimate is represented in numerical form at the bottom of the matrix. The company concerned has about 70 employees in Finland and our assessment revealed weak points in the inspection process at the whole company level. Capability determination in the i3GO model is based on the checking of 29 indicators. As we can see from the matrix (Figure 2), one indicator can affect a number of activities. An enabling indicator (E) confirms that the preconditions for a specific activity are met, and the absence of such an enabling indicator strongly suggests that the corresponding activity does not exist. A verifying indicator (V) confirms that a specific activity has produced appropriate and adequate results, and its existence suggests that the corresponding activity may exist, but does not guarantee this. The existence of an indicator is evaluated on a five-grade scale: na = not relevant to the organisation, 0 = not in use at all or only rarely, 1 = partially, exists to some degree, 2 = largely, exists fairly well, and 3 = fully, always in existence.

In the evaluation session we walk through all the indicators, grouping them according to core, supporting and organisational activities, the first 10 indicators applying mainly to the core set of activities, indicators 11-20 being related to all groups of activities and indicators 21-29 being mainly for organisational activities. This order of indicators is justified because it is easier to start the assessment session by evaluating core activities and to go on to supporting and finally organisational activities. In Figure 2 we can see the result of the evaluation in the column score and the multiplied values (score * indicator weight) in the cells. The weights of the indicators vary, an enabling indicator having the weight 1 and a verifying indicator the weight 2. Thus the indicator Entry decision, for example, has the value 4 (2*2) for activity 7 and the value 2 (2*1) for activity 9. The percentage denotes how well the activity is controlled in the company, calculated in terms of the sum value as a proportion of the maximum value, see activity A.12 (100 * 15/ 30 = 50 %)



Indicator	Score	A.1	A.2	A.3	A.4	A.5	A.6	A.7	A.8	A.9	A.10	A.11	A.12
	%	43	47	48	33	22	29	67	56	43	43	62	50
1) Entry decision	2							V 4		E 2			
2) Entry criteria	1	V 2						E 1					
3) Schedule	3								V 6	E 3			
4) Inspection resource allocations	1		V 2						V 2	E 1			
5) Request for rework on another artefact	1											V 2	
6) List of corrections	3						E 3					V 6	E 3
7) Acceptance of corrections	1												V 2
8) Request for recorrection	0												V 0
9) Exit decision	3												V 6
10) Exit criteria	1	V 2											E 1
11) Forms	2	V 4	EV 6		EV 6					E 2	E 2		
12) Tutorials	2	V 4		E 2							E 2		
13) Courses	1	V 2		V 2						E 1			
14) Collaborative tools	1		V 2		V 1				E 1	E 1			
15) Data gathering & anal. tools	0		V 0		EV 0	E 0	E 0			V 0	E 0		
16) Rules	2	V 4				EV 6			E 2	E 2	E 2	E 2	
17) Role descriptions	1	V 2		E 1		E 1			EV 3				
18) Checklists	0	V 0				EV 0			E 0	E 0			
19) Issue list	3						E 3			V 6	E 3	E 3	E 3
20) Categorisation	0					E 0	E 0				V 0	E 0	E 0
21) Pool of inspectors	2		V 4	V 4					E 2				
22) Selected inspection leader	3		V 6					E 3	E 3				
23) Request for training	1		V 2	E 1									
24) Quality resource allocations	1	V 2	V 2										
25) Reports	0	E 0	E 0	E 0	V 0	E 0	V 0						
26) Measured process data	0					E 0	E 0			V 0			
27) Improvement suggestions	1	E 1											
28) Process change requests	1	V 2	E 1			E 1							
29) Inspection process definition	3	V 6	E 3	E 3	E 3				E 3				
	sum	31	28	13	10	8	6	8	22	18	9	13	15
	max	72	60	27	30	36	21	12	39	42	21	21	30

Figure 2. Capability determination matrix.

3.2 Process improvement with patterns

After the assessment, each improvement has to be planned and carried out. Inspection improvement patterns are pre-defined guides and procedures for upgrading an inspection process. Seven patterns defined in our catalogue are listed in Table 1.

Table 1. Inspection improvement patterns

Pattern name	Main goal
Greed	Aims at finding more defects during inspections.
Early bird	Aims at finding defects at earlier stages in development.
Substance	Aims at finding more serious defects in inspections.
Comfort	Aims at making the inspection process easier to run.
Promotion	Aims at promoting the process so that it is carried out more often and in a larger number of projects.
Wisdom	Aims at a more understandable, transparent and effective inspection process.
Precision	Aims at making the process more rigorous, and thus more effective.

Each pattern defines an unambiguous goal and a list of actions for making the reviewing process more efficient. The purpose of the pattern catalogue is to aid the assessor and the company representatives in focusing their improvement activities on the most crucial parts of the process and to provide practical guidance for conducting the improvement. The patterns provided in this table address some of the most common deficiencies concerning inspections, and the list is not exhaustive. Additional patterns can be created as needed, or several patterns can be combined to achieve the desired improvement. The Greed pattern is presented as an example in the following:

Pattern name
Greed
Intention
The intention is to upgrade the inspection process in order to uncover more defects during inspections.
Symptoms
Symptoms that suggest using the Greed pattern include the following:

- Only a few errors are revealed in inspections.
- Only cosmetic errors are found in inspections.
- A lot of time is used to achieve poor results.

Problem
Even though inspections are carried out frequently, they can produce poor results. Despite visible investment in the method, its advantages remain unclear. Inspectors may consider the whole process as a waste of time, and eventually the management will think the same way if no observable gains are achieved.

A certain rigor is necessary to get the most out of the process. If this discipline is overlooked, or inspectors are not provided with adequate support for doing their job, inspections will probably reduce productivity and motivation. Reviewing is meaningful and efficient only if it reveals adequate numbers of defects, and if these defects are not trivial.

The reasons for poor performance may vary. First, the artefact to be inspected or material related to the artefact may be unavailable or hard to obtain for the inspectors or there may be shortcomings in the workflow of the process. Second, there should be an adequate number of inspectors to read through the document and they should be given enough time for performing the review. Third, the inspectors must have an up-to-date checklist and tools to support them in the process. As illustrated in Figure 3, insufficient resources or support can dilute the outcome of an inspection.

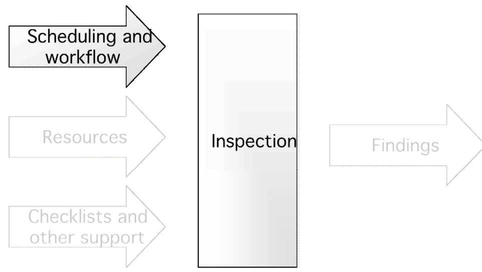


Figure 3. Inadequate resources and support produce poor results

Use of the Greed pattern should balance the process in order to get better results from the inspection.

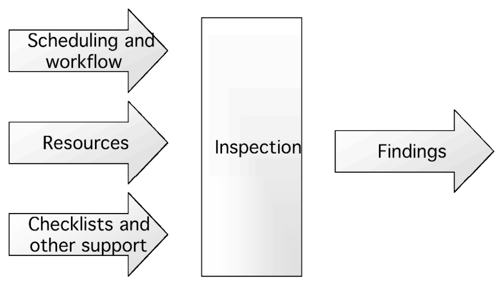


Figure 4. Improvements gained by use of the Greed pattern

Context

The Greed pattern should be used when the inspection process has recently been installed or skills in the use of inspections have diminished, and the process does not perform efficiently enough. Inspection tasks may be seen as extra work and the benefits as vague, or else inspections may be a normal procedure, but arranged and controlled carelessly by the QA group. If inspectors feel that their work is not supported or appreciated adequately, they will probably fail to discover real defects.

Solution

To approach the inspection process with the Greed pattern and upgrade it as follows:

1. Name a person or persons to be responsible for the inspection process. Such persons should pay special attention to the reaching of a common agreement (especially with the management) that inspections are planned, scheduled and resourced in every project. Consider adjusting the project plan document template so that inspections cannot be omitted.
2. Checklists have to be created or updated. Either the person responsible for the process has to take responsibility for maintaining the checklist, or else another person has to be named for the task. It is handy to derive one checklist from one document template – a requirements checklist from a requirements document template, for example. For code,

use the language-specific coding conventions and related guides to create initial checklists. If possible, use historical data (search through a bug database, interview experienced developers) to find typical errors and write them into the checklists.
3. Make artefact distribution more efficient. Study the feasibility of using e-mail or other software tools for this. Ensure that the inspectors really use the checklists and source documents. Observe their work and arrange training if feasible. Ensure that checklists and source documents are distributed along with the artefact, and that the inspectors know how to use them.
4. Consider establishing specialist roles – inspectors who focus on usability, interoperability or code reusability aspects, for example. Create a separate set of checklists for these specialist viewpoints. Train the participants if needed.

Consequences

Implementation of the actions listed in this pattern will affect activities A.5 Maintain rules and checklists, and A.9 Find issues in the artefact, in particular. Organisationally these improvements aim at making the inspection process more visible and establishing responsibility for it. The actions defined in this pattern will strengthen the very essential practices of the process, and may cause modifications to the company's quality handbook, if it does not already adequately address the planning and scheduling of inspection tasks.

4. Analysis of motivation factors and obstacles

The interview was conducted in spring 2004 in twelve software development departments varying in size between 160 and 5 persons. Seven of the companies had a quality group, and five of them maintained that it worked actively, regularly and visibly. Six of the companies possessed a quality handbook, and the interviewees were familiar with it. Among the other questions, we asked about motivation for reviews (Table 2) and obstacles to them (Table 3). The grade of importance is depicted in the tables as follows, VI = very important, I = important, R = rather important, S = some importance, N = not important.

Table 2. How companies are motivated towards reviews

Motivation	VI	I	R	S	N
Sharing expertise	1	7	1	2	1
Sharing information	2	4	5	1	0
Teaching novices	4	3	2	2	1
Controlling project progress	0	4	5	2	1
Process improvement	3	2	3	4	0
Finding more defects	8	3	0	1	0
Finding defects earlier	9	2	0	1	0
Finding defects faster	7	3	0	1	1

As highlighted in Table 2, the interviewees prioritised the "finding defects" tasks as the most important motivation issues, while "sharing expertise" and "sharing information" rose fairly high as motivation issues, and "teaching novices" and "controlling project progress" also received relatively high scores. "Process improvement" was evaluated as a very important source of motivation in three companies, but the majority rated it as rather important or of some importance.

We also studied the reasons why companies do not adopt reviews in their entirety (Table 3). "Shortage of time" and "shortage of staff" were the most important obstacles, and reasons such as "not required", "lack of expertise" and "laborious" were also mentioned.

Table 3. Reasons for avoiding reviews

Obstacles	**VI**	**I**	**R**	**S**	**N**
Shortage of time	8	4	0	0	0
Shortage of staff	4	5	1	1	1
Not required	0	2	2	1	7
Costs	0	0	1	3	8
Lack of expertise	0	1	3	0	8
Laborious	0	3	2	3	4
Geography	1	0	0	1	10
No need	0	0	1	1	10

It is also worthwhile noting that a significant number of companies did not see obstacles such as "not required", "costs", "geography" and "no need" as being of any importance.

5. Conclusions

This paper reports three approaches to stimulating the adoption of reviews by software companies. We started with simple tailoring cases, because companies were interested in light-weight processes and appropriate tools. Even though the functionality and ease of use of the tools had improved during the development cycle, it is still a challenging task to induce SME companies to really use them. It is more plausible that companies will attempt to use existing tools and tailor them to support reviews as well as possible.

We also looked for review improvements in six companies, in an analysis based on a capability evaluation matrix. A very rough generalized estimate of the results was that the review process in SME companies was typically at a rather low level, because they had recently started improvement of the process. We also learned that although the members of the quality group might be interested in review process improvement, this did not guarantee that they would install improvements in their projects. This is understandable, because project schedules are very tight and do not allow for process improvement experiments.

Our analysis of motivation factors and obstacles gives the same explanation for why SME companies do not use reviews. Although the interviewees understood that reviews would enable defects to be found earlier, faster and in greater numbers, shortage of time was the main reason which prevented them from using reviews.

In conclusion, we can say that research should proceed in the reverse order from ours. It should start with (1) analysing the review techniques used in companies (including motivation factors and obstacles), then (2) search for weak points in the review process as used in the company (based on capability analysis and process patterns), and finally (3) develop and tailor appropriate supporting tools. Our research will continue in that order. As a result of the analysis of review techniques, we will start process analysis in some companies and move on to improvement by means of existing process patterns. In the light of our earlier experience, we realise that the achieving of improvements through projects is a challenging task and requires support from the management, a quality group who are keen on improvements and the proper allocation of resources – but it is possible.

6. References

[1] Burnstein I., Practical Software Testing: a Process-Oriented Approach, Springer Verlag, 2003

[2] El Emam K., Drouin J. and Melo W. (eds), SPICE: The Theory and Practice of Software Process Improvement and Capability Determination, Wiley, 1997

[3] Fagan M.E., Design and Code Inspection to Reduce Errors in Program Development, IBM Systems Journal, vol 15, no 3, 1976, pp.182-211

[4] Gilb T., and Graham D.: Software Inspection, Addison-Wesley, Wokingham, England, 1993

[5] Group Review, The URL of the Group Review info is http://www.cmi.arizona.edu/collaboration/Group%20Review.html

[6] Harjumaa L., and Tervonen I., A WWW-based Tool for Software Inspection, Proceedings of the 31st HICSS Conference, vol III, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 379-388

[7] Harjumaa L., Distributed Software Inspections – an Experiment with Adobe Acrobat, in Proceedings of the IASTED International Conference on Computer Science and Technology (CST 2003), 2003, pp. 26-31

[8] Hedberg H., Hajautetun ohjelmistotarkastuksen tukeminen XML-teknologioilla, Phil. Lic. Thesis, University of Oulu, 2003

[9] Johnson P.M., and Tjahjono D., Improving Software Quality through Computer Supported Collaborative Review, in De Michelis G., Simone C., and Schmidt K. (eds.), Proceedings of the Third European Conference on Computer Supported Cooperative Work, Kluwer Academic Publishers, Dortrecht, Netherlands, 1993, pp. 61-76

[10] Komi-Sirviö S., and Tihinen M., Great Challenges and Opportunities of Distributed Software Development – An Industrial Survey, Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, 2003

[11] Koomen, T., and Pol M., Test Process Improvement: A practical step-by-step guide to structured testing, Addison-Wesley, 1999

[12] Kuvaja P. et al., Software Process Assessment and Improvement, The BOOTSTRAP Approach, Oxford, Blackwell, 1994

[13] Mashayekhi V., Drake J.M., Tsai W-T., and Riedl J., Distributed, Collaborative Software Inspection, IEEE Software, September 1993, pp. 66-75

[14] Paulk M. et al., The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, 1995

[15] Perpich J., Perry D., Porter A., Votta L., and Wade M., Anywhere, Anytime Code Inspections: Using the Web to Remove Inspection Bottlenecks in Large-Scale Software Development, Proceedings of the ICSE-1997 Conference, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 14-21

[16] ReviewPro, The URL of the ReviewPro info is http://www.sdtcorp.com/reviewpro.html

[17] Tervonen I., Harjumaa L., and Iisakka J., The Virtual Logging Meeting: a web-based solution to resource problems in software inspection, Proceedings of the 6th European Conference on Software Quality, ADV Handelsgesellschaft m.b.H., 1999, pp. 342-351

[18] Tervonen I., Iisakka J., Harjumaa L., A Tailored Capability Model for Inspection Process Improvement, Proceedings of the Second Asia-Pacific Conference on Software Quality, 2001, pp. 275-282