Transactions on Information and Communications Technologies vol 9, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

A tool and a set of metrics to support technical reviews

J.B. Iniesta

Quality Group 4000, Telefónica Investigación y Desarrollo, 28043 Madrid, Spain

ABSTRACT

What is not measured is not controlled. What is not tracked is not done. If both adage are true, it is obvious that we need to measure and track the process of doing technical reviews of software projects.

In this paper I intend to describe how we keep track and measure this process in our Company. We have found that a tool to support these activities is a necessity.

INTRODUCTION

In late 1.992 our Quality Group was instructed by Management to set up technical reviews in all the projects of our Division. At first 16 projects were planned to be reviewed. Since each project would generate at least 4 to 10 documents, and each document would be reviewed by some 4 to 6 people, it became obvious from the start that we had in our hands a big management problem.

THE PROJECT DEVELOPMENT METHODOLOGY (MDP)

In October 1.992 our Company adopted officially a methodology for software project development: **Metodología de Desarrollo y Gestión de Proyectos (MDP).** It is the work of an internal committee that was formed in March 1.992.

The MDP specifies every phase of a project lifecycle, including the activities to be performed, intermediate work products, and the technical reviews that must be performed.

The documents that must be reviewed are the following: *Product Requirements Specification (PRS), User's Manual (USM), Architectural Design (ARD), Software Component Specifications (SCS), and System Test Specifications (STS).* The MDP specifies the minimum content this documents

must have, and also what kind of people are responsible for writing and reviewing them.

The MDP does not call for the writing and reviewing of detailed design documents. The SCS go to the level of black box specification of the processes, including message sequence charts, interface descriptions and state transition diagrams. When the Company finally adopts an Object Oriented Design Methodology, sometime in the years 1.994-5, the corresponding work products will be reviewed.

Code inspections are not covered in this paper.

TRADITIONAL TECHNICAL REVIEWS.

Our early experience with technical reviews was very disappointing.

The process was as follows:
The **Project Leader** would contact with our Quality Group to select a **Moderator** for the review. Moderator and Project Leader would collect a list of documents to be reviewed.

For each document, the Moderator asked the Project Leader for the date the review should be finished and the **Author**'s name and other data such as his or her telephone number, mail address, etc. The Moderator also asked the Project Leader for the name of at least one and if possible two experts from the project group, not involved in the writing of the document, to serve as **Reviewers**.

Another reviewer would come from the **Product Quality Division** of the Company. This Division is responsible for auditing the products of all projects. They participate in some technical reviews: requirements, user manuals and system test plans. They do not participate in design reviews.

Another reviewer would come from the **System Tests Division.** This Division is responsible for performing the final acceptance test of a product. They are interested in getting an early understanding of the product requirements and user manuals, so as to be able to plan these testing activities. So they take part in some reviews.

Sometimes we would ask an expert from another project to take part in these reviews. The practice later was discontinued, as we found that the results were poor.

The Moderator would distribute a paper copy of the material to each Reviewer, who would make annotations on the same copy and then send them back to the Moderator, who would then correlate all these annotated copies to collect a single list of improvement proposals for the Author.

The Moderator would arrange a review meeting. The list of improvement proposals was distributed to Reviewers and Authors before the meeting. During this meeting, each improvement proposal was discussed and a decision was reached on its acceptance or not, until the list was finished. The Author would edit the material, and the process was done.

Problems with the traditional process:

1. The Moderator spent quite a long time collecting and writing improvement proposals from the Reviewers into a single list. Sometimes the number of improvement proposals generated was very considerable, as many as 100 or more. In any case, this was a redundant work, since the Reviewers had already written them.

   The Moderator sometimes had to interpret what the Reviewer wanted to say, since the annotations were not wholly self-explanatory, relying on the context of the document.

   For the reasons seen above, the compilation of a single list of improvement proposals was a bottleneck for the process.

2. The Moderator had nothing to help him in the task of tracking the whole process. When the number of documents to be reviewed increased, the Moderator lived in a state of growing insecurity. Have the Reviewers written down their improvement proposals so as to collect them? Has the Author accepted or rejected the improvement proposals and has he or she already modified the document? What documents are currently under review? Records had to be kept on paper and had to be updated by way of telephone calls to the growing number of people involved in the process. All of this made life uncomfortable for the Moderator.

REQUIREMENTS FOR A TECHNICAL REVIEW SUPPORT TOOL

The experience acquired in doing technical reviews during 1.992 demonstrated the convenience of having a tool to support the Moderator's job. We felt that otherwise we would not be able to guarantee success for this activity. The requirements for this tool have been evolving all the time (surprise, surprise) so I am going to present them in some kind of chronological order. This is a natural way of presenting the problems as they came up and the solutions that we implemented in the tool.

EARLY SOLUTION: JANUARY-JUNE 1.993

The first requirement: collecting improvement proposals from the Reviewers:
The idea is very simple: we don't want to do things twice, and we don't want one person doing the job of many others. So it was decided that the Reviewers would write their improvement proposals not on the paper copies of the material to be reviewed, but on simple, ASCII files using the UNIX file editor. These files would be automatically collected by the tool, sorted by page number, and then printed into a single list.

In Vahid [1.] there is a very inspiring presentation of a distributed enviroment for performing these tasks.

Now I have to say that the technology used in the tool has been the UNIX Korn shell and the awk utility processing simple ASCII data files. It has proved to be adequate for some time, but we have plans to move into a relational database and Motif user interfaces. I will try, however, to make this presentation independent of the technology used and concentrate on the problems and their solutions.

The benefits from this automatic collection of improvement proposals were the following:

1. The Moderator collected all the improvement proposals from the Reviewers executing a command. The resulting list was sorted by page number and presented in a nice tabular format, including an automatic numbering of the improvement proposals and a form for the Author to accept or reject them.
2. The Reviewers were forced to write self-explanatory improvement proposals, since they didn't have the context of the document to make do for any imprecision.
3. The Moderator now had a very easy way to determine whether a particular Reviewer had written his or her improvement proposals without bothering telephone calls: if that Reviewer's file existed or not.

The second requirement:
organising and keeping track of the reviews of many documents.

Since we were using UNIX files to store the Reviewer's improvement proposals, it was only natural to organize things in directories. A tree structure was adopted, with a node representing the project, and each branch a different document reviewed in that project. As the number of projects reviewed grew, it was necessary to add one and later two new hierarchical levels, to organize projects into Divisions and other groups.

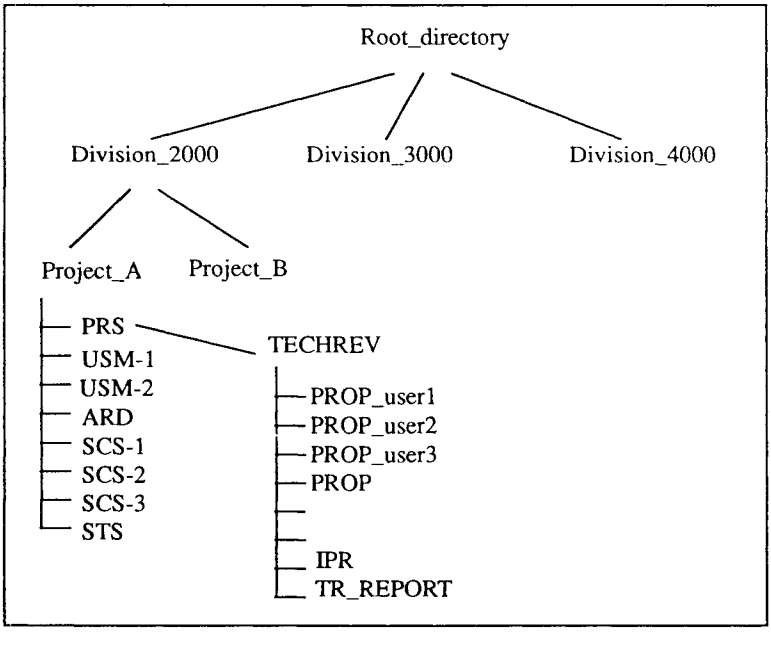


Figure 1. Organizing technical review work products.

When we move into the relational database system, the organization of the data will be along these lines, using other mechanisms. For the sake of clarity, I am now going to describe the present solution, using UNIX files and directories.

The tool allows easy access to each document of any project under review, by way of menus that select projects and documents. The Reviewers can log into the tool from any terminal in the Company, and they can use these menus to select the document when they want to introduce improvement proposals.

The Moderator uses programs to scan all the projects and automatically generate status reports of the reviews under way.

Lets suppose ***user1*** wants to introduce improvement proposals against the *Product Requirements Specification (PRS)* of Project_A. This ***user1*** would write file ***PROP_user1***, and the tool would store it in the directory
`<installation_dir>/Division_2000/Project_A/PRS/TECHREV`
shown in Figure 1, "Organizing technical review work products.," on page 4. The tool will store all data about this particular technical review into this directory, including improvement proposals files from other Reviewers, reports and data files.

The Moderator can generate a progress report on this document review (TR_REPORT), simply by automatically comparing the list of ***PROP_userX*** files available with the list of reviewers, which is also stored in the tool in a configuration data file.

When the Moderator sees that all the reviewers have written their improvement proposal files he or she generates automatically a single file ***PROP*** containing all the proposals, sorted by page number of the document were the improvement proposals apply. If an improvement proposal affects all the document, it appears at the beginning of the file. The tool stores the name of the reviewer together with the improvement proposals he or she generates, so as to be able to later evaluate his or her performance. More of this latter.

Having the single ***PROP*** file, the tool now generates an **Improvement Proposals Report (IPR)** for the Author with the list of all the improvement proposals affecting his or her document.

Table 1: Example of an Improvement Proposal Report to the Author.

N#	Paragraph Page	Improvement Proposal	Decision
12	**6.1.1** 6-08	"RES-ALARM" command, it is said "depending on the parameters, the system will show..." but this command doesn't have any parameters. **Author's comments:**	Accepted O Rejected O
16	**6.2.1** 6-12	"VIS-MEASURES" command. Range definition should include as many zeros as necessary to fill in the number of chars required. For instance 001 to 127. **Author's comments:**	Accepted O Rejected O

This report includes a form for each improvement proposal, so that the Author is able to easily specify if it is accepted or rejected. The Author sends the report back to the Moderator, with the forms all ticked up, and the Moderator decides whether a review meeting is necessary or not.

The Author handwrites in the report why he or she rejects any improvement proposal that deserves so, and may ask the Moderator to organize a review meeting if he or she needs further discussion of any improvement proposals.

Third requirement: control the changes made on a document as a result of a technical review.

A Review Meeting is expensive, and also it is a synchronous activity (Vahid [1.]) that requires every participant to be available, so arranging it may generate delays in the review process. We feel that Review Meetings should be held only when there are reasons to justify them: when the Moderator or a Reviewer thinks the Author has rejected improvement proposals that should be accepted, or when the Author needs more information on a particular improvement proposal.

For this reason, the Moderator introduces manually into the tool the Author's acceptance or rejection of each improvement proposals and automatically generates a Change Agreement Report (CHAR) that he or she distributes to all the Reviewers, so that they have a chance to negotiate with the Author the acceptance of a previously rejected improvement proposal.

Table 2: Example of a Change Agreement Report.

N#	Paragraph Page	Improvement Proposal	Decision
12	**6.1.1** 6-08	"RES-ALARM" command, it is said "depending on the parameters, the system will show..." but this command doesn't have any parameters. **Author's comments:**	**Accepted X** Rejected ---
16	**6.2.1** 6-12	"VIS-MEASURES" command. Range definition should include as many zeros as necessary to fill in the number of chars required. For instance 001 to 127. **Author's comments:** The system accepts "1", "01", "001", etc.	Accepted --- **Rejected X**

This CHAR is useful also because it is a great help for the Author when the time comes to modify a document. The Author has a list of all the improvement proposals, which are accepted and which are rejected, so his or her work is clearly mapped.

This CHAR is the equivalent of an Action List in Vahid [1.]

Fourth requirement: measure the quality of the review itself.

It was felt it was necessary to have a way to measure if technical reviews were useful for the projects under review.

The solution was to make use of the information stored into the tool about the decision agreed on each improvement proposal, so that it was possible to measure the percentage of acceptance of the Reviewers' work.

This measure proved to be insufficient, as we shall see presently.

Fifth requirement: measure the quality of the document.

Management wanted a metric of the quality of the documents reviewed. Some work was done in this area, including a set of questions adapted to each type of document, which the Moderator would answer after the review was done. The questions had a system of weights, and the document could satisfy them in a partial, complete or null way. As a result of this questions, the tool calculated a number measuring the quality of the document. We have found that this number is not very useful.

PROBLEMS WITH OUR EARLY SOLUTION

Problem: too many improvement proposals, very few interesting ones.

Since the tool made it easy for the Reviewers to introduce improvement proposals, and the measure of their performance was to be how many proposals were accepted, it is easy to understand why the Authors began complaining that they were overwhelmed with lots of what they thought were uninteresting and often trivial improvement proposals, such as typos, style, etc. These improvement proposals would be accepted, but they contributed very little to the quality of the documents reviewed.

Now I think it is necessary to state that our Methodology doesn't ask Reviewers to look for defects in the documents. In this subject we follow Philbin [2.] The idea is that if you say something is a defect, you must convince the Author that it is so. It is much easier just saying that something will be better if it is done some other way. The Author may agree or not, but he or she will feel less under attack in this way, and it will be less of a problem for him or her to accept what is a just an improvement proposal, not a defect on his or her work.

It is also much easier for the Reviewers to write improvement proposals than to find defects. This is not to say that Reviewers do not find defects. It is obvious that if a defect is found, the Reviewer will write an improvement proposal to have it corrected. But certainly the risk exist that the Reviewers may introduce too many improvement proposals of little interest, if they are not limited to the reporting of obvious defects.

This problem was serious. Some time in June there was a state of opinion that technical reviews were very expensive "Spelling Checkers". It was said that Quality people were only interested in correcting typos and punctuation, so the whole thing was a costly way to delay the delivering of documentation to the client.

Problem: project managers complain their personnel spend time doing worthless reviews.

This problem comes as a consequence of the previous one. Some project managers would resist their personnel taking part in this process because they were not convinced that technical reviews were worth the time and effort invested in them.

Of course, technical reviews were made mandatory by higher management, so project leaders had little choice but to comply with the procedure. Still their concern was justified, and the process should demonstrate that it was useful for the projects.

METRICS TO UNDERSTAND THE PROBLEMS.

What we wanted was a measurable way to evaluate the quality and productivity of the technical reviews that were being carried out.

1. We think quality means that the Authors benefit from the review process.
2. As for productivity, that is a subject we have not addressed yet. We have concentrated in the quality problem first, and by the time this paper is written the productivity metrics remains an objective for the near future. We have data recorded, but we have made no effort yet to interpret it.

Data gathering.

Our first attempt with metrics consisted in recording the number of improvement proposals made by the Reviewers, and how many were accepted or rejected by the Authors. We also recorded the number of reviewers, how much time each one of them devoted to this activity, number of pages in each document reviewed and the type of document: PRS, ARD, USM, etc.

With this data we wanted to answer questions such us the following:

1. How much effort does a Reviewer need to devote to a technical review? This is the question most managers would ask before allowing his or her personnel to take part in a technical review.
2. Are technical reviews worthwhile? This is the question all managers would ask before allowing documents from his or her project to be reviewed.

So we see the need for two sets of metrics about technical reviews: quality metrics and productivity metrics. Quality metrics should say whether the process is useful for the projects reviewed. Productivity metrics should say whether the effort invested in doing reviews is worthwhile.

Quality metrics.

As we have said before, the number of improvement proposals and the percentage of accepted and rejected is not a good indicator of the quality of reviews. We wanted to know whether the Authors thought the quality of their documents benefited from the improvement proposals. The obvious solution, of course, consisted in asking the Authors whether they found each improvement proposal made by the Reviewers very interesting or not interesting at all, or something in between.

The Improvement Proposals Report (IPR) was duly modified to make room for a simple form that allowed Authors to easily specify the interest of each improvement proposal. The Moderator would load this information into the tool when he or she introduced the accepted/rejected status of each improvement proposal. The tool would then compute a number to evaluate the average

interest of the whole review. This information would also appear in the Change Agreement Report, as it is shown in Table 3.

Table 3: Example of a Change Agreement Report including an Interest Evaluation Form.

N#	Paragraph Page	Improvement Proposal	Decision
12	**6.1.1** 6-08	"RES-ALARM" command, it is said "depending on the parameters, the system will show..." but this command doesn't have any parameters. **Author's comments:**	**Accepted X** Rejected --- The proposal is interesting? **Very Much X** Normal -- Spelling -- Little -- Not at all --
16	**6.2.1** 6-12	"VIS-MEASURES" command. Range definition should include as many zeros as necessary to fill in the number of chars required. For instance 001 to 127. **Author's comments:** The system accepts "1", "01", "001", etc.	Accepted --- **Rejected X** The proposal is interesting? Very Much -- **Normal X** Spelling -- Little -- Not at all --

Authors can say whether they find each improvement proposal very interesting, normal, of little interest, or not interesting at all. Later a fifth category was added: spelling, syntax, style, etc were considered in a separate class, because these improvement proposals are acceptable but they do not contribute to the information content of the document.

The tool assigns a weight of 10 points to very interesting improvement proposals, 3 points if they are interesting, 1 point if they are of little interest, and 0 points if not interesting at all. Spelling etc gets no points and do not contribute to the mean calculation. The formula (1) for the Technical Review Interest (TRI) indicator is as follows:

$$TRI = \frac{1}{NIP}(10V + 3I + L) \qquad (1)$$

TRI: Technical Review Interest indicator

NIP= V+I+L+N: Number of Improvement Proposals made by the Reviewers

V: Number of very interesting Improvement Proposals

I: Number of interesting Improvement Proposals
L: Number of little interesting Improvement Proposals
N: Number of not interesting Improvement Proposals

METRIC RESULTS.

Lets present some results now. Up to the date of writing this paper, we had a database with records from 78 technical reviews corresponding to 17 projects, carried out between March and November, 1.993.

Records from the first 10 documents reviewed do not include improvement proposals acceptance data, so they will not appear in the graphs. From the 68 documents remaining, the first 21 were reviewed before we introduced review quality evaluation. These will be represented to the left of a dotted line in the following graphs.

Before March 1.993 there were no records of any kind about the reviews performed.

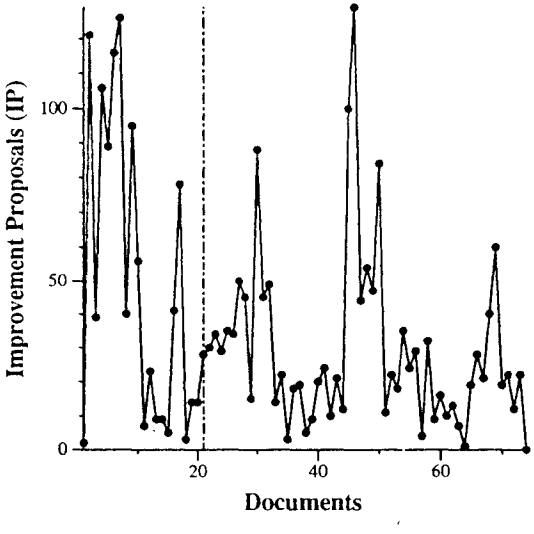


Figure 2. Number of Improvement Proposals per Document Reviewed between March and November, 1.993.

In Figure 2 we see quite a wild distribution of statistical data. Many factors contribute to this strong variations:

1. Many different Reviewers performed in different ways, some of them would introduce many improvement proposals, some of them would introduce very few or none at all.

2. Reviewers took part in a reduced number of reviews. Only Quality people took part in a significant number so as to be able to show a regular performance.
3. Even then, the same Reviewer would have different performance with documents from different projects, sometimes even with documents from the same project.
4. Obviously, a document with many pages may have more improvement proposals than another one with fewer pages, though this is not by any means a general rule.
5. Also it is possible that different kinds of documents have different behaviour in the gathering of improvement proposals, meaning that PRSs may have more improvement proposals than USMs, for instance. We have yet to check this one out in the future.

Calculating the mean and standard deviation of the accumulated data helps to show the behaviour of the process. We wanted to validate with these metrics that Reviewers were making a better job, introducing fewer improvement proposals but with a more interesting content, specially after the introduction of the Technical Review Quality evaluation metrics.

Lets say that our metric function f(x) consist of the set of points $(1,x_1)$, $(2,x_2)$, etc, so that $f(i)=x_i$ For instance, we can define the function IP(x) as the number of improvement proposals made against document number x.

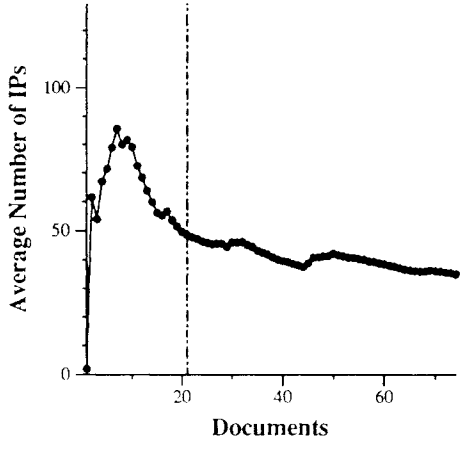
The formula (2) for the accumulated mean is as follows:

$$Mean(x) = \frac{1}{x}\sum_{i=1}^{x} f(i) \qquad (2)$$

The formula (3) for the accumulated standard deviation is as follows:

$$Deviation(x) = \frac{1}{x}\sum_{i=1}^{x} \sqrt{(f(i) - Mean(i))^2} \qquad (3)$$

After computing the accumulated averages and the standard deviation, we can see in Figure 3, "Accumulated Average Number of Improvement Proposals per Document Reviewed between March and November, 1.993.," on page 12 that before review quality metrics were introduced, the accumulated average number of improvement proposals was considerable. After they were introduced, the accumulated average tends towards a conservative 35 improvement proposals per document, with a small standard deviation of 3.8.



Average= 34.9189 Standar Deviation= 3.76257

Figure 3. Accumulated Average Number of Improvement Proposals per Document Reviewed between March and November, 1.993.

The metrics are telling us that the Reviewers were making less improvement proposals. Were these more acceptable to the Authors?

We can see in Figure 4 the percentage of improvement proposals accepted, which once again have a very wild distribution of data, and in Figure 5 the accumulated averages, which again show a very remarkable improvement just to the right of the 21 documents point, after the quality metrics were introduced.

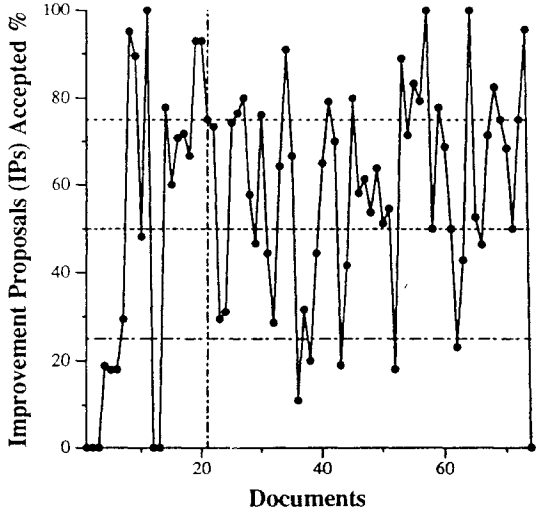
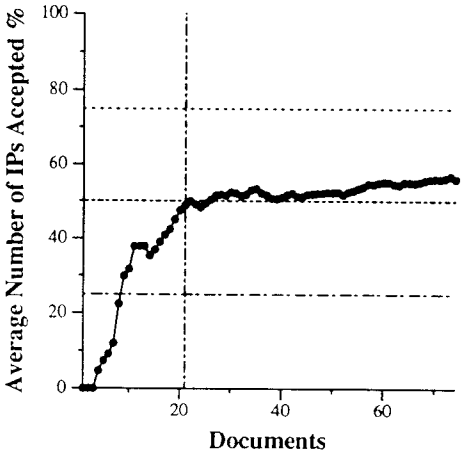


Figure 4. Percentage of Improvement Proposals accepted by the Authors per Document Reviewed between March and November, 1.993.



Average= 55.9453 Standar Deviation= 3.27775

Figure 5. Accumulated Average of Improvement Proposals accepted by the Authors per Document Reviewed between March and November, 1.993.

The metrics are telling us that the Reviewers were making less improvement proposals and that these were more acceptable for the Authors. But did the Authors find these improvement proposals interesting?

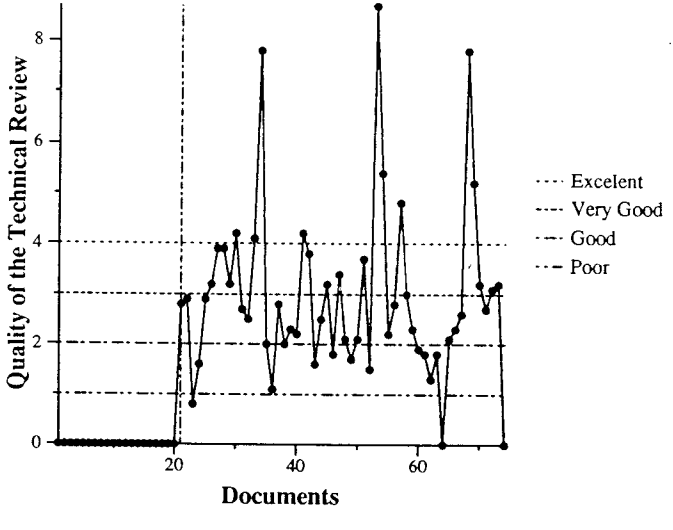
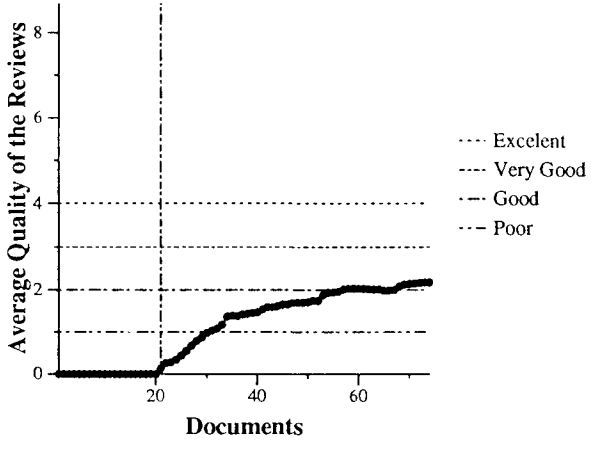


Figure 6. Quality of the Technical Reviews performed between March and November, 1.993.

In Figure 6 we see how the Authors evaluated the interest of the improvement proposals, using the Technical Review Interest (TRI) indicator presented in (1). If the TRI is between 0 and 1 we say the review was Insufficient, between 1 and 2 it was Poor, between 2 and 3 it was Good, between 3 and 4 it was Very Good, and above 4 it was Excellent. The first 21 documents are not evaluated since we do not have data.

In Figure 7 we see the accumulated averages, which confirm our impression that a very remarkable improvement in the quality of our reviews took place after the quality metrics were introduced in July 1.993.



Average= 2.17397 Standar Deviation= 0.224677

Figure 7. Accumulated Average Quality of the Technical Reviews per Document Reviewed between March and November, 1.993.

It is said that statistics can say whatever you want them to say. In our case, these metrics are confirming a feeling that technical reviews are a successful activity in our Company. There are many factors besides the metrics that support this view:

1. Project leaders now regard technical reviews as a very normal activity. In January 1.993 there was a lot of resistance to carry them out. Now the project leaders come to our Quality Group to ask for their documents to be reviewed.
2. In 1.992 it was hard work to get people to take part as Reviewers in this activity. Negotiations with project leaders and a measure of arm twisting was normal procedure. Now it is quite straightforward to collect a Review team, participants have a good morale and they use the tool with efficiency.
3. Authors respond very quickly to IPRs, now that they have the chance of evaluating the interest content of the Reviewers' improvement proposals. They feel more in control of the situation. We were surprised by this fast reaction

time, because during the months before this metric was introduced Authors were very slow in responding to IPRs, and then they would sometimes complain about the lack of interest of the proposals.

EVALUATION OF THE INDIVIDUAL REVIEWERS

We have data that allows us to evaluate the performance of individual Reviewers. We know the date the reviews took place, the number of improvement proposals generated, percentage of accepted and rejected, interest content of the improvement proposals, time dedicated to the activity, and number of documents and number of pages reviewed.

This information is included in the Change Agreement Report, so that Reviewers have an opportunity to discuss with the Author the evaluation of their improvement proposals. We have found that this simple mechanism makes it more interesting for the Reviewers the job of checking whether the Author accepts or rejects their improvement proposals.

In Table 4 we see an example of a Reviewer Evaluation Report for one particular document. This report includes an entry for each Reviewer and a TOTAL row for the whole document.

Table 4: Example of Reviewer Evaluation Report.

Reviewer		Time	Time per Improvement Proposal (m)	Number of Improvement Proposals	Percentage Accepted	Interest	Evaluation of the Review
1	user1	3:56	3.9	53+ 8	42/19: 68%	1:14: 8:31: 7	1.6 Poor
2	user2	2:00	3.5	34+ 0	17/17: 50%	1: 7: 0:26: 0	1.7 Poor
3	user3	3:01	13.9	13+ 0	8/ 5: 61%	0: 5: 0: 8: 0	1.8 Poor
4	user4	4:54	22.6	12+ 1	4/ 9: 30%	0: 4: 1: 8: 0	1.7 Poor
5	user5	1:45	26.2	4+ 0	3/ 1: 75%	2: 1: 0: 1: 0	6.0 Excellent
6	user6	4:00	80.0	3+ 0	1/ 2: 33%	0: 2: 0: 1: 0	2.3 Good
TOTAL		19:36	25.0	119+ 9	75/53: 58%	4:33: 9:75: 7	1.8 Poor

It is interesting to note that we separate clearly spelling, syntax, style etc improvement proposals from those that really modify the information content of the document. In the report you can see the column labelled "Number of Improvement Proposals" with two numbers separated by the "+" sign. The number to the left is equal to the number of improvement proposals that had some information content, the number to the right is equal to the number of syntax, spelling, etc proposals.

In the column labelled "Interest" you can see five numbers. From left to right, they represent the number of improvement proposals that were very interesting, normally interesting, spelling etc, of little interest and not interesting at all.

We see that the best performing reviewer, user5, only spent 1:45 in the review and made 2 very interesting improvement proposals, 1 interesting and 1 of little interest, with no spelling or style proposals.

In contrast, user1 spent 3:56 in this activity, made 53 improvement proposals and 8 spelling and style proposals, 1 was very interesting, 14 interesting, 31 were of little interest and 7 were not interesting at all. This user1 had a poor performance, since he or she loaded the Author with a lot of proposals that were of little interest, but in any case contributed with 15 interesting improvement proposals.

The final evaluation for the review as a whole is 1.8 Poor, because there were too many improvement proposals of little interest. The logical thing to do now is to help Reviewers to make fewer and better proposals. As it happens, user5 and user6 were experienced reviewers, while the others were new to this activity.

This information makes it possible to choose the most effective Reviewers for a particular project or a type of document.

FUTURE WORK

Lack of space forbids presenting any more material on this subject. Nevertheless, I want to finish this paper with a list of standing issues where we expect to continue our work:

1. We need a productivity metric. This metric will have to take into account some form of evaluation of the cost of defects found. This means that some form of correlation between improvement proposals and defects must be established. We already have data necessary for this metric, such us time spent in the review, number of documents and pages reviewed, etc.
2. We want to avoid as much paper shuffling and manual data entry activities as possible. Much of this will be achieved with the future Motif user interface of the tool, which will allow Authors to receive and respond to interactive IPRs instead of paper ones.
3. Our Company has recently moved from ASCII text processors to workstation-based, WYSIWYG documentation tools. We want to distribute documents electronically so that Reviewers can make hypertext annotations on them. This is in line with Vahid [1.]
4. We want more automatic management tracking support from the tool. We are using the mail system to notify Moderators when the Reviewers have written their improvement proposals, and now it is possible to query the tool for the completion status of reviews under way, and possible actions needed.
5. We want to collect checklists to help people review different types of documents.

REFERENCES

1. Vahid Mashayekhi, Janet M. Drake, Wei-Tek Tsai and John Riedl, 'Distributed, Collaborative Software Inspection' *IEEE Software,* pp. 66-75, September 1.993.
2. Alice I. Philbin, Ph.D. and John W. Presley, Ph.D., *Technical Writing: Method, Application and Management,* Delmar Publishers Inc., 1989.