



Automatic metrics to measure the efficiency of technical reviews

J.B. Iniesta

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

ABSTRACT

In SQM-94 in Edinburgh I presented a paper whose title was *A Tool and a Set of Metrics to Support Technical Reviews* [1.] That paper had two distinct parts, the tool to automate the technical review process and the metrics to measure and control the task.

Lack of time during the presentation forbade me to explain satisfactorily the metrics part of the paper. Furthermore, since then we have made quite a lot of technical reviews and we have devised some new metrics, so now I feel we have quite a lot of new data to support our ideas about how metrics can foster and improve technical reviews. The aim of this paper is therefore to concentrate on the metrics part of technical reviews.

INTRODUCTION

In late 1992 our Quality Group was instructed by Management to set up technical reviews in all the projects of our Division. During 1993 we began using a home-developed tool that automated many tasks related to technical reviews. One of these tasks was metrics collection.

The metrics allowed us to measure the progress and the efficiency of all the technical reviews we had under way. This automation helped us to review 89 documents during 1993, and an impressive 264 documents during 1994.

WHAT DOCUMENTS DO WE REVIEW?

In October 1992 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 1992.

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 our methodology says must be reviewed are the following: *Product Requirements Specification (PRS)*, *User's Manuals (USM)*, *Architectural*

184 Software Quality Management

Design (ARD), Software Component Specifications (SCS), and System Test Specifications (STS). The MDP specifies the minimum content these 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 processes, including message sequence charts, interface descriptions and state transition diagrams.

Code inspections are not covered in this paper.

METRICS OBJECTIVES

We want to be able to answer some basic questions:

1. How long does it take to make a technical review?
2. How much effort do we have to invest in a technical review?
3. Technical Reviews: are they worth all the trouble?

Questions 1 and 2 are simple to answer. We must keep track of how many calendar days it takes to get a document reviewed, and how many hours or minutes the reviewers have spent at the job. But the last question is a little more complex.

ATTRIBUTES OF DEFECTS FOUND DURING TECHNICAL REVIEWS

Testing and reviewing are not the same thing

Traditional technical reviews had the stated objective of finding defects in software documents.

These defects could be classified regarding their severity: fatal, major or minor for example. A return on investment analysis could then be performed based on the number and severity of defects found and the amount of time devoted to the technical review.

These analysis usually would compare the efficiency of technical reviews with that of testing. The comparison usually would be very positive for technical reviews, since testing requires a lot of effort and fixing defects in that stage of development is expensive.

We have found this approach unworkable. The fundamental reason is that when testing it is possible to observe system failures, and while debugging it is possible to fix defects that were the cause of these failures. That is, we are dealing with defects that provoke system failures that can be fatal, major or minor.

But when reading a document it is possible to find defects that do not have an obvious connection to system failures. So it may not be easy to say whether a defect found during a technical review was fatal, major or minor. We need a new classification method.

But first let us consider a very basic question. What is a defect?

What Is A Defect?

When testing and code inspecting the answer is obvious: a fatal or major defect could be anything that may provoke a system failure, while a minor defect could be

something we do not like very much, but seems more or less innocuous, for instance, a deviation from a programming standard.

This definition implies that the work product that is under verification is going to be executed, so that it is possible to predict whether the defect will provoke a failure or not.

But when reading a document it may be possible to find things that we do not like, and we may propose them to be changed, while not being able to link these changes to system failures at all.

The reason is that documents are not executable.

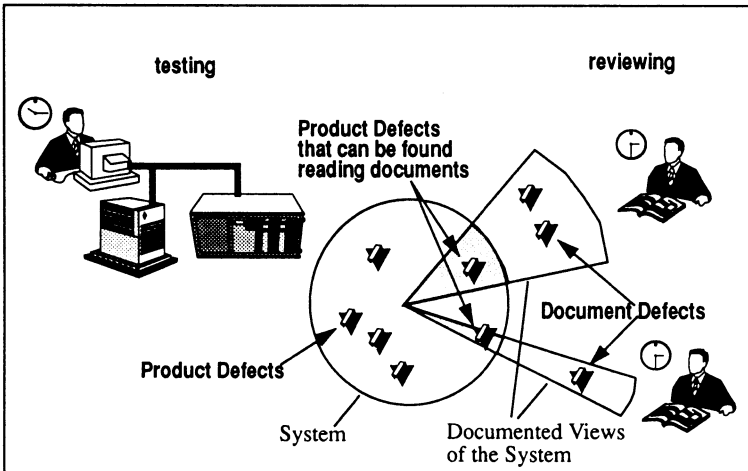


Figure 1. Testing and reviewing find different kinds of defects.

In Figure 1 we represent the idea that when testing the software engineer has access to all the system and can find defects that may provoke system failures, while when reviewing the software engineer has access only to documented, partial views of the system. He or she can only find defects that are in the documentation, and these defects may be responsible of system failures or may be just documentation defects.

Attributes Of Defects Found Reading Documents

Imagine a reviewer wants something changed in a document. First thing the author may say, is that the reviewer is wrong: the change is not necessary. It may be that the reviewer got the idea wrong, or what he or she proposes is not right in the eyes of the author. This situation is typical of technical reviews: our metrics say authors accept about 63% of the reviewers' change proposals. The remaining 37% are rejected and the documents are not modified.

This is quite different from testing: change request originated during system testing or integration testing are rejected very rarely.

186 Software Quality Management

So we have our first attribute of a defect found in a technical review: acceptability by the author.

One could imagine a metric of the efficiency of technical reviews as a measure of both the number of change proposals and the percentage of acceptance of them. But this is not enough. There is another very important attribute we must consider, which is the interest that each change proposal has for the author.

Since a document is not executable, we have no obvious way to tell in advance whether our change proposal is uncovering a serious defect, or whether it is of no great significance to the quality of the system. Our best way to evaluate the impact of our change proposal is through the collaboration of the author. He or she is in best position to evaluate the importance of our contribution to improve the quality of the work product under review. We have divided the range of importance of a reviewer's change proposal into four categories, from very interesting to not interesting at all.

Change proposals that have to do with spelling, syntax, style, format etc are a fifth category and they are not taken into account when measuring the efficiency of the technical review.

So here we have the second attribute of a defect found in a technical review: interest to the author.

AUTOMATIC METRICS COLLECTION AND EVALUATION OF THE EFFICIENCY OF TECHNICAL REVIEWS

We can now explain our procedure to automatically measure the efficiency of technical reviews.

Our process for performing technical reviews requires reviewers to write change proposals¹ into a database for each document reviewed. The author has access to this database, so that he or she can read the change proposals, can accept or reject them, and can evaluate their interest.

After the author has processed all the change proposals, the tool sends electronic mail to all reviewers informing them of the author decisions. The reviewers may accept these decisions or they may wish to contact the author to further discuss some points.

Review meetings are avoided if possible, and all communication is done in writing.

Reviewers also introduce into the database information on how much time they have spent in this activity, including time spent reading documents, and time spent writing change proposals. In this way we can measure the effort spent in this activity.

Since we have a database with all the change proposals, the authors' evaluations of them, dates the technical review began and ended, and the effort spent by reviewers, it is possible now to measure the efficiency of technical reviews. All data used is collected automatically by the tool so it is highly reliable and the overhead is minimum.

1. Change proposals and improvement proposals are synonyms in this paper.

METRIC RESULTS

How many technical reviews have we done?

We began using the technical review support tool in March 1993, and we introduced the efficiency metrics in June of the same year. This fact is shown in some of the graphs that follow in the form of a dotted vertical line separating data from technical reviews performed before and after June 21st, 1993. We can thus see how the introduction of this metric had a considerable and certainly positive impact on the whole process.

When somebody is asked to perform a technical review of a document, we say he or she is assigned a review job. Several review jobs of the same document make a technical review of the document. So we have data about 354 technical reviews performed by 148 reviewers who were assigned 907 review jobs.

Validation of data

Of these 907 review jobs, 150 had zero change proposals made. The reasons could be the reviewer had no time to do the job, or he or she read the document but could not find anything to change. It may happen that one reviewer finds a lot more change proposals than another in the same document. Actually we have found the choosing of the right people to be reviewers as the main factor that contributes to the efficiency of technical reviews. All other factors are second in importance, including number of pages of the document, time spent in the review, etc.

There were 25 review jobs with a reported effort of less than 10 minutes. Some of these were because the reviewer just browsed along the document, but most of them, actually 20, were because one particular user would not report the time spent reading the document.

This is one of the reasons why our tool had to introduce sanity checks of the data before any metrication could take place. The criteria adopted is that a review job is valid if the reviewer reports at least 10 minutes of time devoted to the job, and has found at least one change proposal that is not spelling, syntax, format, style etc. So out of 907 review jobs, we chose not to include in the final analysis the data from 175 (19%) review jobs. Since a technical review is valid if at least one of its review jobs is valid, this gave us 307 technical reviews that were valid and 47 (13%) that were invalid.

Number of technical reviews per month

This is a good indicator of the penetration of our methodology into the everyday life of our projects. As we can see in Figure 2, there has been a steady rate of technical reviews performed with our tool since its introduction in March, 1993. Particularly during 1994, when the process was well understood and widely

188 Software Quality Management

adopted by many projects, the rate has been above 13 documents reviewed per month, except during August with 8 and December with 9 documents reviewed, due in part to holidays.

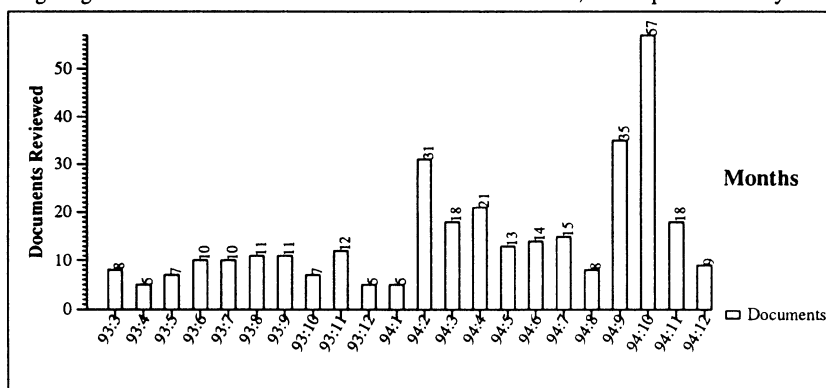


Figure 2. Number of documents reviewed per month, March 1.993 to December 1.994.

Time spent by reviewers

Our 148 reviewers have reported an average effort of 2:13 per document, with a standard deviation of 6 minutes (4.65%). This effort includes time to read the document and time to write all necessary change proposals.

If you consider that the average review meeting may take about that time to complete, it is very clear the advantage of using this method for performing technical reviews. We have found written change proposals and electronic mail very satisfactory for communication among reviewers and authors. Making technical review meetings mandatory would double the average effort spent in this activity. It could even make the whole thing unworkable: imagine 57 technical review meetings in one month?

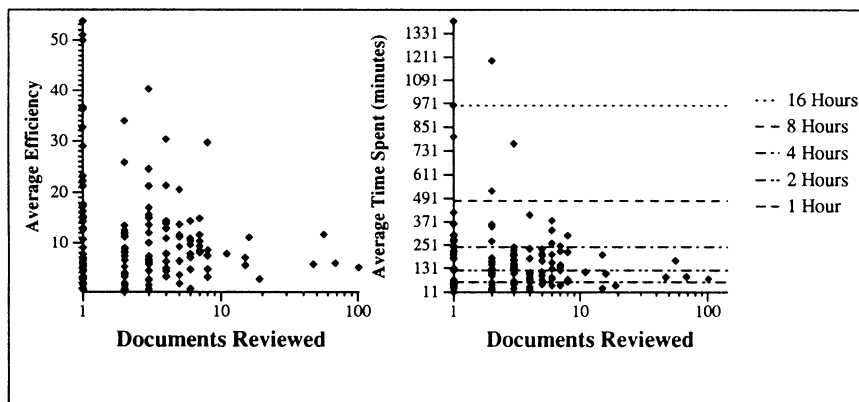


Figure 3. Reviewers' average efficiency and effort per document.

In Figure 3 we see two graphs. The one on the right shows the average effort spent by each of our 148 reviewers as a function of the number of review jobs assigned to him or her.

It is very clear from the data available that people spend a lot of effort only if they review a few documents that have a lot to do with their own project. All points above the 4 hour limit belong to reviewers who had less than 10 review jobs assigned to them, and the three points above the 16 hour limit had respectively 1, 1 and 2 review job assignments. On the contrary, those points to the right of the 10 job assignments reported an average effort of about 2 hours per document. These points correspond to quality assurance people, who read a lot of documents from many projects.

The graph on the left of Figure 3 is telling us a similar story: it is only reasonable to expect a high average efficiency if people review a few documents from their own project. Quality assurance people get less than 10 points after reviewing many documents, as many as 40 or 60 over a period of two years. Local experts from the projects get very high average efficiency when reviewing a few documents of their own project.

By the way, I have not defined efficiency yet. We will see how this metric is calculated very soon. But before leaving this subject I want to highlight an important lesson we learned through experience: the most effective reviewers are experts from the project under review.

Experts from other projects did have very low results. One reason may have been that they did not give enough importance to reviewing documents from other projects. Other reason may be that they did not have enough common knowledge of the project to be able to really understand the documents. This point is important, because when we began this activity we gave a lot of importance to this idea of having experts from outside the project reviewing the work products, but now we do not even ask anybody from outside the project to do the job, except quality assurance people who have to read documents from many projects.

Duration of technical reviews

Once the process was well understood and usage of the support tool improved communication among reviewers and authors, so that review meeting were largely unnecessary, the duration of the whole process ceased to be a problem.

We have performed technical reviews in as little as 24 hours, but the average duration has been 12 natural days, with a standard deviation of 0.5 days.

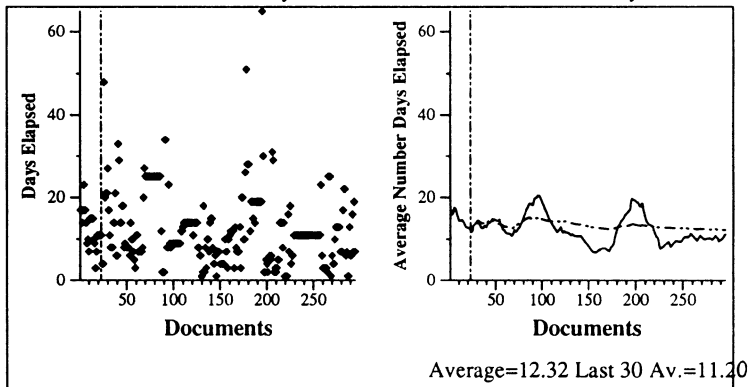


Figure 4. Duration of each technical review and evolution of the average.

190 Software Quality Management

In Figure 4 we see two graphs. To the left we represent the value of the metric for each technical review performed. The vertical dotted line shows when the efficiency metric began to be used. As we can see it did not affect the technical reviews duration.

The graph on the right shows the evolution of the average value for the metric, and we can see two curves. The dotted line represents the evolution of the total average, while the solid line shows the evolution of the average value of the metric for the last 30 technical reviews performed. We decided to use this evolution to see more clearly oscillations around the total average, which is affected very much by the history of the metric.

In the rest of the paper we shall again use this kind of graphs.

Number of improvement proposals per document

This metric was used initially as a measure of the efficiency of a technical review, together with the percentage of them accepted by the authors of the documents reviewed. Very early we detected that the result was counterproductive, since authors complained that reviewers were writing too many improvement proposals of very little interest. As a consequence we introduced the interest metric, allowing authors to defend themselves from what they considered ill-conceived or uninteresting change proposals.

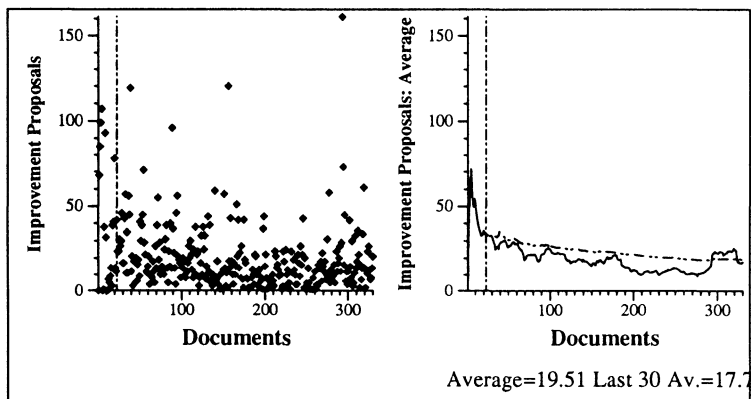


Figure 5. Number of improvement proposals per document and evolution of the average.

The procedure called for authors to accept or reject each change proposal made by the reviewers, while at the same time evaluating the interest of each change proposal. The authors then were empowered to evaluate the quality of the technical review. Only if they found the change proposals acceptable and interesting would the technical review be effective and of good quality.

In Figure 5 we see again a double graph, the one to the left showing the value of the metric for each technical review performed, the one to the right showing the evolution of the total and 30 last averages. The dotted vertical line separates technical reviews performed before the interest metric was adopted.

In this case we do see quite a change after the interest metric is introduced. The number of improvement proposals written by reviewers before the interest metric was used was very high, with the average above 40 improvement proposals per document. After the interest metric was introduced, the average has dropped to about 20 improvements proposals per document, while the last 30 average has been below that figure for some months.

So we have the reviewers making less improvement proposals to the authors. But do the authors find these improvement proposals acceptable?

Percentage of improvement proposals accepted per document

Once again in Figure 6 we see the value of the metric for each technical review to the left, and the evolution of the total average and last 30 averages to the right. And once again we can see how the introduction of the interest metric had quite an effect on the percentage of improvement proposals accepted. Before this metric was used, the percentage was very low, with an average value below 50%, while after its introduction the average has improved dramatically, with a total average of 63% and a last 30 average topping 75% in some cases.

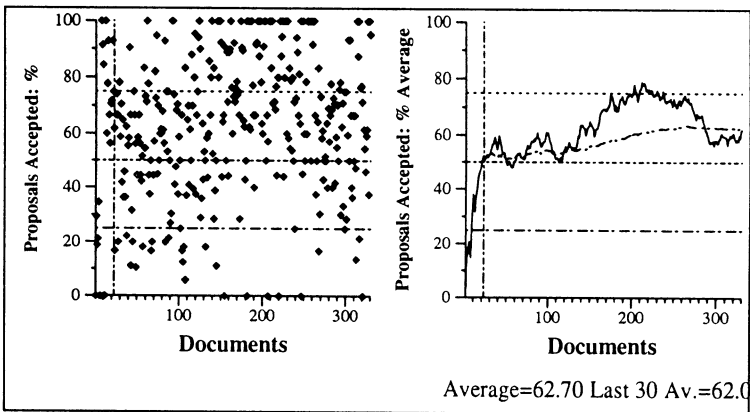


Figure 6. Percentage of improvement proposals accepted per document and evolution of the average.

These metrics are telling us that reviewers are making less improvement proposals and that these are more acceptable to the authors. But do the authors find these improvement proposals useful and interesting?

Interest and efficiency

We distinguish between the interest and the efficiency of a reviewers' job. It will be more interesting the more the author finds the change proposals useful. It will be more efficient the more the document is modified as a result of these change proposals.

The document will be changed the most if the reviewer writes many improvement proposals, many of them are accepted, and many of these are interesting or very interesting to the author. Then the efficiency metric should have a big value.

192 Software Quality Management

If the reviewer writes a lot of improvement proposals which are acceptable but of very little interest, the document will not benefit from the technical review, so the efficiency should be low.

If the reviewer writes a few improvement proposals which are acceptable but very interesting, the document may benefit from the technical review, so the efficiency should be average.

The solution adopted is to have two metrics: interest and efficiency. The first measures the usefulness of each improvement proposal, while the second measures the impact these improvement proposals have on the document. So it is possible to have a high value in one and a low value in the other, though usually the correlation between both metrics will be linear.

As we have said before, authors can say whether they find each improvement proposal very interesting, normal, of little interest, not interesting at all, or spelling-syntax-style-format-etc.

Our technical review support 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 calculation. The formula (1) for the Technical Review Interest (TRI) indicator is as follows:

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

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

As you may see, in this formula we do not take into account the acceptability of improvement proposals. This TRI indicator is telling us only if the reviewer writes interesting change proposals. It may happen that the author rejects change proposals that are interesting or even very interesting, for some reasons that he or she should explain in the e-mail response to the reviewers.

The formula (2) for the Technical Review Efficiency (TRE) indicator is as follows:

$$TRE = \frac{1}{3} \times IPA \times TRI \quad (2)$$

TRE: Technical Review Efficiency indicator

TRI: Technical Review Interest indicator

IPA: Number of Improvement Proposals made by the reviewers and Accepted by the authors.

In this formula we take into account both how many improvement proposals we are accepting and also how interesting they are. Note that improvement proposals rejected do contribute to this formula through the TRI

indicator. This is a way to reward reviewers who make interesting or very interesting change proposals even though they are not accepted for some reason by the author.

The reason for using the 1/3 scaling factor is because we gave a weight of 3 to interesting improvement proposals. So one possible way to look at this TRE indicator is as the **equivalent number of interesting improvement proposals accepted in a technical review**.

So if we have a technical review with a TRE of 30, it means the document has got the equivalent of 30 interesting changes. This is a simple way to look at the result of a technical review, taking everything into account: interest of the improvement proposals, how many they were and how many have been accepted. So the TRE indicator is our favourite metric, since it gives us a clear message about how much the document has changed as a result of the technical review performed.

The metrics we have for TRI are as follows:

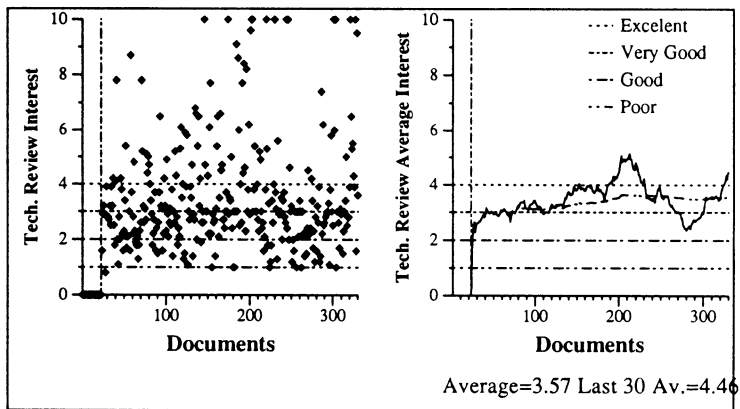


Figure 7. Interest of improvement proposals per document and evolution of the average.

It is obvious that before this metric was introduced, there was no data available, so that no points are shown to the left of the vertical dotted line we see on both graphs.

Though no data is available before June 21st 1993, it is possible to see that this metric improved very fast during the first technical reviews after its introduction. The average climbed from less than 2 to more than 3 and has stayed there for most of the time, climbing above 5 around the 200 technical review point. We can see the last 30 average dropped below 3 sometime around technical review number 270, but it has come back to above 4. The total average now is 3.57 and the last 30 average is 4.4.

Since an interesting change proposal gets 3 points, we have arbitrarily chosen the following sectors for this metric: above 4 it is excellent, between 3 and 4 is very good, between 2 and 3 is good, between 1 and 2 is regular, below 1 is poor. This means that if TRI for a given technical review is good, it means many improvement proposals were interesting (3 points each) and some were

194 Software Quality Management

less than interesting (1 or 0 points), so that the average was something between 2 and 3. If the TRI is excellent, it means that some improvement proposals were very interesting (10 points each), and that pushed the metric above the 4 level.

The metrics we have for TRE are as follows:.

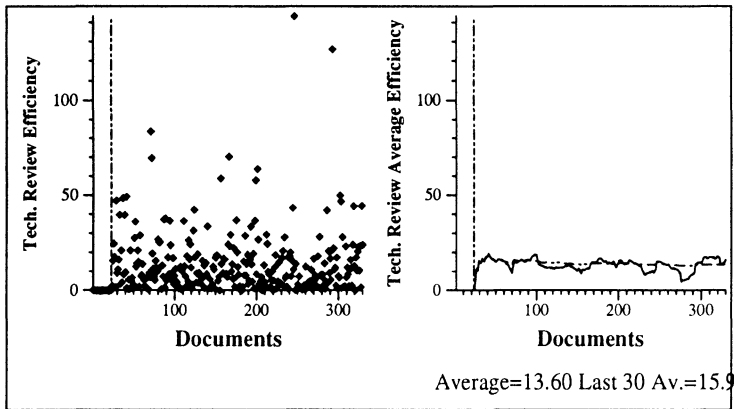


Figure 8. Efficiency of improvement proposals per document and evolution of the average.

It is interesting to see that the average TRE indicator has remained fairly constant all along the history of our technical reviews, with a total average of about 13. Since we know that the average number of change proposals has dropped, it means that the percentage of acceptance and the interest of these has increased to compensate this drop. So we may say that in our organization, we may expect 13 interesting changes when a document goes through a technical review.

EVALUATION OF INDIVIDUAL REVIEWERS

Since our tool keeps data about who wrote each improvement proposal, we can evaluate the performance of each reviewer. We have found this information to be very valuable to understand why some technical reviews are effective and why some of them are not.

The main factor that has contributed to the efficiency of technical reviews in our organization has been the capability and disposition of our reviewers. All other factors: type of document (user manual, test specification, design etc), number of pages, duration of the review, time spent etc have been second in importance. We must be careful how we use this information, but we think it is useful to inform all participants in a technical review how they have performed. It rewards the hard working and it may help those who did not get good metrics to improve in the future.

REFERENCES

1. Javier Borrajo, 'A Tool and a Set of Metrics to Support Technical Reviews' Software Quality Management II Vol. 2, pp 579-594, Computational Mechanics Publications 1994.