Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

What do developers really think about software quality?

T. Hall
School of Computer Science, University of Westminster, 115 New Cavendish Street, London, W1M 8JS, UK
Email: hallt@westminster.ac.uk

Abstract

For an organisation to get the biggest increase in software quality that it can from introducing a more formal and quality-focused process of software development, the attitudes that developers and managers have to any new working practices must be understood, listen to and accommodated. It is only then that software quality will be improved and sustained in the long-term.

This paper presents the results of a recent wide-ranging study which explores the attitudes that developers and managers have to software quality generally, and to the introduction and use of quality-oriented working practices specifically. The study is based on the analysis of over two hundred questionnaire responses from software developers and managers in a number of organisations. All of these organisations either had, or were in the process of introducing, a formal quality management system.

The study found that developers and managers often had markedly different perspectives on the usefulness and scope of mechanisms for software quality. Sometimes managers were more enthusiastic about a particular mechanism - for example the collection and use of software metrics. Whilst at other times developers were more keen on a particular mechanism - for example the use of standards.

The study also revealed some interesting goal variations between developers and managers. For instance developers rated software reliability as a much more important organisational goal than managers did. Conversely, and probably more predictably, managers rated low costs as a much more important organisational goal than developers did. Clearly such goal incongruence would seem a potential source of quality problems.

1 Introduction

The introduction of any new tool or technique into a work environment will inevitably change that work environment. Unfortunately the resulting change will not always be the change that was expected, planned or desired. Indeed the introduction of a new innovation, unless skilfully and sensitively managed, may actually change an environment for the worse. It is, therefore, important that organisations consider the full implications such a change is likely to have. In particular it is important that organisations understand the full impact on staff. Though it is equally important that organisations appreciate the effect that staff can have on the success or otherwise of a new innovation. This argument applies as much to the introduction of quality mechanisms into software development as it does to any other working practice innovation. An organisation introducing software quality control mechanisms must understand how those quality mechanisms are being received by the developers using those mechanisms.

Furthermore, for optimal quality improvement it is necessary to customise and calibrate quality mechanisms to particular development environments. In order to do this calibration managers need to understand their developers. Without this understanding calibration, and therefore improvement, is not controlled. Software developers, however, are not a homogeneous group, and developers will have different thoughts on, and reactions to, the use of formal quality mechanisms. Indeed developers will view "quality" *per se* differently. To implement effective quality control mechanisms managers need to understand these different developer attitudes.

This paper presents an overview of the preliminary results from an on-going empirical study of software practitioner attitudes. The study has sought to examine how the introduction and use of a variety of quality-related mechanisms has affected practitioners in a number of targeted organisations. The paper discusses the views practitioners have about software quality generally and the use of quality mechanisms specifically. It also presents a rationalisation of why different sets of practitioners are more positive about different quality control mechanisms. Finally the paper tries to quantify the use and effectiveness of particular quality mechanisms.

2 The Study

The study was based on a questionnaire that was sent to staff in four British organisations (see figure 1 for more details of those organisations). The study itself was completely independent of any of the organisations involved. All of the targeted organisations used quality control mechanisms, although the maturity of their quality improvement programmes varied from very ad hoc to quite mature.

Organisation	Size	Type	Software Developed	Target Sample
A	Very large (200++)	Public body	Information systems	Sample from computing function
B	Large (200+)	Engineering Co	Safety critical	The whole of one small SE team
C	Large (200+)	Technical Co	A range of applications	A sample from one large SE project
D	Quite small (50-100)	Software house	Information systems	All SE staff

Figure 1: The Organisations in the Study

A mixture of development and management staff were targeted within each organisation and, although the response rate varied between the organisations, the overall response rate at 65% was much better than usual for a questionnaire-based survey. A total of 182 responses were received.

3 Quality Control Activities

The study showed that quality control mechanisms (inspections, reviews, standards, procedures, manuals etc.) were in common and structural use in the four organisations in the study. Indeed, only 13% of people said that they personally never spent any time on quality-related activities. The use of standards was considerably more prevalent than the use of inspections and reviews (10% of practitioners said that they never used any standards and 23% said that they were never personally involved in any reviews or inspections). It was also the case that, on the whole, the right people were using the right mechanisms. For example 99% of programmers said that they used programming standards and 74% of them were involved in code reviews or inspections. The overall most commonly used quality control mechanism was documentation standards - almost everyone said that they used documentation standards.

There was also a surprising amount of other quality-related tools in place. For example 62% of people said that software metrics data was collected, 95% said that they used a Quality Manual and 91% that a configuration management system was in place.

4 Attitudes to Quality

Overall practitioners were very positive about the use of most quality-related mechanisms. Many were also genuinely interested in the tools and procedures that could help in the improvement of software quality. For example 70% of all respondents thought that the collection of software metrics was a useful

activity. There was also a surprising amount of consensus on how working within an increasingly quality-orientated framework allowed work to be undertaken effectively (67%), efficiently (71%) and flexibly (76%). Indeed within some of the organisations in the study developers were surprisingly enthusiastic to see the use of more standards and procedures.

Practitioners were also candid about the quality levels that they were attaining (see figure 2).

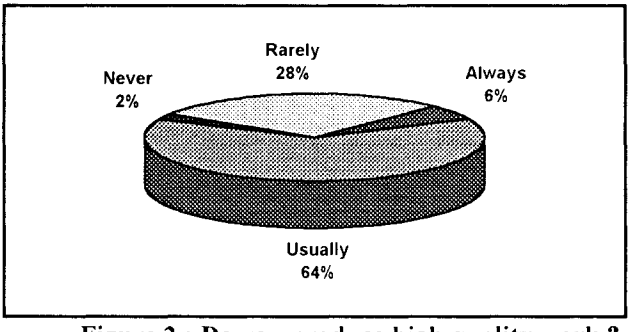

Figure 2 : Do you produce high quality work ?

Though managers were over-represented in the small number of respondents who claimed to 'always' produce high quality work. This is interesting in the light of the recent Capers Jones [Jones94] study which suggests that technical staff are performing much more effectively than managers, as it was the managers in this study who thought that they were performing better.

5 The Effects of Quality Control Mechanisms

The study suggested that the everyday use of quality control mechanisms had a powerfully positive effect on the attitudes that practitioners had to software quality. The regular use of standards was a particularly significant factor. 70% of practitioners who claimed to "always" produce high quality work also said that they "frequently" used standards of some sort. Furthermore, all of the practitioners who claimed to "always" produce high quality work did at least "occasionally" use standards i.e. there were no people in this staff group who "never" used any standards (see figure 3).

The regular use of reviews or inspections also had a positive effect on the perceptions that practitioners had about the quality of their work. Again, there was a higher proportion of practitioners who claimed "always" to produce high quality work, who were also "frequently" involved in reviews or inspections. Here however there was also a proportion of practitioners with high perceptions of the quality of their work who were "never" involved in reviews or inspections (see figure 4). This suggests that the use of standards has more impact than the use of reviews and inspections on perceptions of work quality.

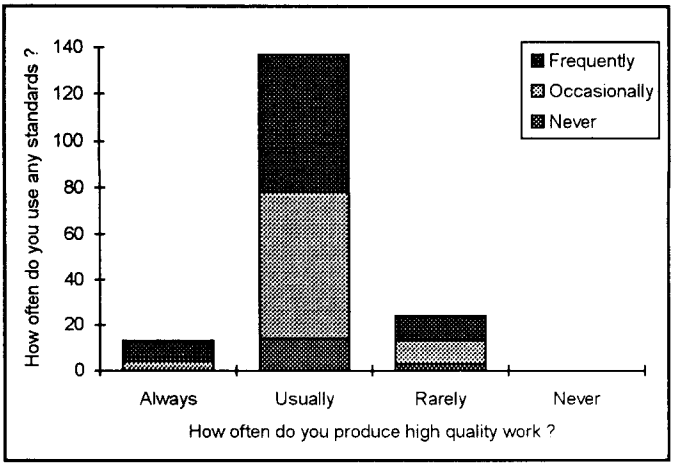

Figure 3 : The effect of standards on perceptions of quality

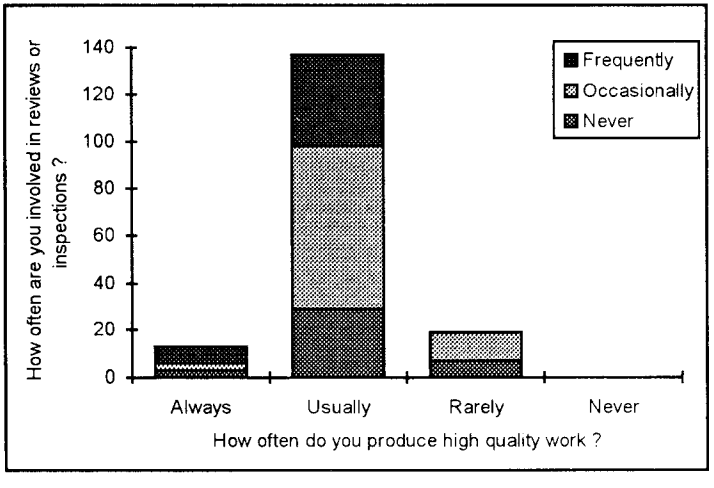

Figure 4 : The effect of reviews and inspections on perceptions of quality

Conventional wisdom suggests that, on the whole, developers do not like to use things like standards and inspections. This study suggests that developers actually find the use of such mechanisms useful. It seems that if developers are given a standard to work to and a mechanism to check that work has met that standard people feel good about the quality of their work. Clearly people feeling good about the quality of their work is good for the organisation. People seem to like formal aims (i.e. standards) and formal appraisal of having met those aims (i.e. inspections). Without this formalism people seem more unsure about how good their work is.

Lack of information emerged as an issue in connection with the use of standards, reviews and inspections. When asked whether the data that was collected from inspections was actually being used, although 35% of practitioners said that it was, 56% did not know whether it was or not. This means that about a third of the people who were personally involved in inspections and reviews did not know what happened to the data that was generated at these events. This can either mean that there is really no improvement process in place; it is not transparent to those involved; the communication channel is ineffective. Whatever the reason for this lack of knowledge, effective quality improvement cannot be achieved without a clearer understanding of the process of quality improvement than was demonstrated.

6 Quality Goals

In order to improve quality it is necessary for an organisation to have quality improvement goals. In turn practitioners must have a clear understanding of those goals. Ideally they should have also played an active part in setting those goals. To actually attain those goals it is also crucial that organisations and practitioners have a proper commitment to those goals. In the light of the importance of quality goals the study revealed some worrying goal problems.

On a most basic level 40% of practitioners said that their organisation's goals were not made clear to them. However, this problem was compounded when respondents were asked to rank 5 software development goals according to what they thought their organisation's goals were. Figure 5 shows the confusion that people have about what they think their organisation's goals are.

Obviously practitioners are unclear about what their organisation is aiming for. Practitioners themselves were, however, much clearer about their own goals, see figure 6.

Goal	Mean Score	Ranking
Speed	2.8	1+
User Satisfaction	2.8	1+
Low Costs	3.1	3+
Reliability	3.1	3+
Conformance to Requirements	3.1	3+

Figure 5 : Perceived Organisational Goals

Goal	Mean Score	Ranking
User Satisfaction	1.9	1
Reliability	2.2	2
Conformance to Requirements	2.7	3
Speed	3.8	4
Low Costs	4.4	5

Figure 6 : Practitioner's Own Goals

Further analysis of these results reveals no particular difference in personal goals between developers and managers. But a significant difference in the goals that developers and managers think that the organisation has. For example managers rate reliability as a lower organisational priority than developers. There is no difference, however, in how managers and developers personally rate the importance of reliability (they both rate reliability as more important than they think that they organisation does). This suggests some interesting dilemmas for managers. Since it seemed that it was the managers who were setting the goals in the first place, are they setting organisational goals that even they do not believe in ?

Given these differences in projected organisational goals and practitioners' own goals it seems difficult to believe that quality improvement initiatives are going to be as effective as they might have been.

7 Attitudes of Managers to Quality

Managers and developers generally had different preferred quality mechanisms (though this did vary slightly according to the particular organisation). In general managers were keener on mechanisms that could be used for assessment, while developers were keener on aids to software quality.

For example managers were generally more positive than developers about software measurement (compare the pie charts in figure 7). Indeed the more senior the managers the keener on measurement they seemed to be. Managers were also much less likely than developers to perceive data accuracy and measurement design as problems. For example 34% of managers said that the data being collected was accurate, compared to just 10% of development staff. Although middle managers (and presumably the people most directly involved in measurement) were more likely to say negative things about measurement.

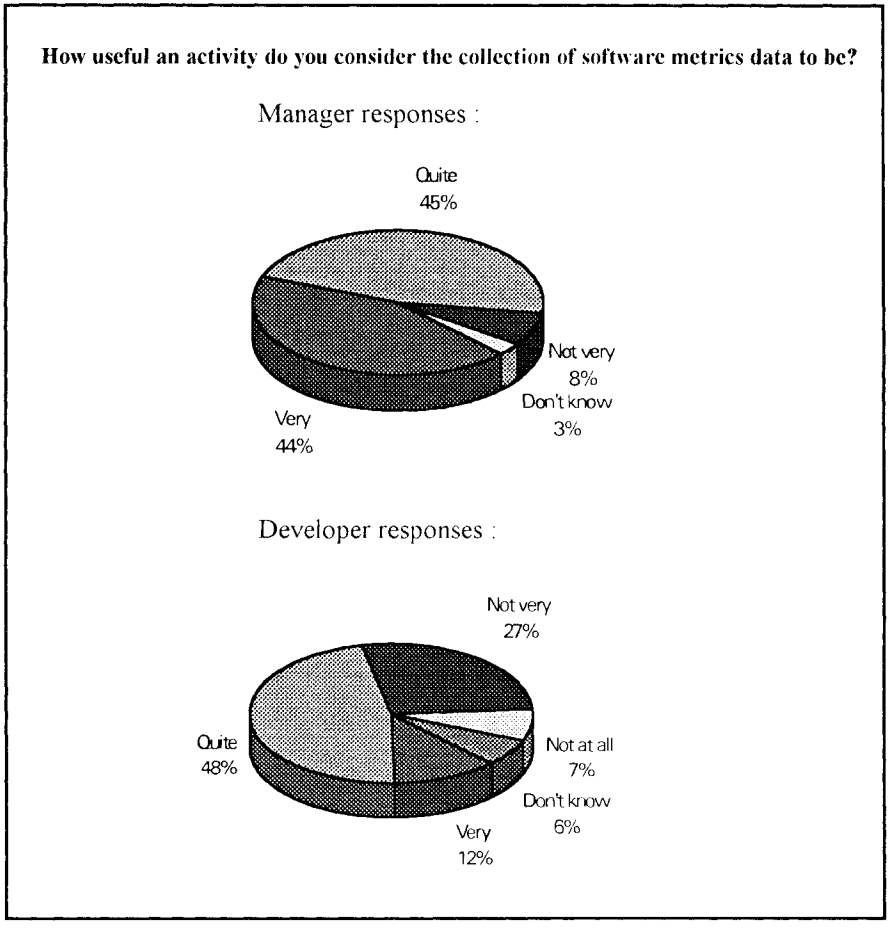

Figure 7 : Developer and Manager views of Software Metrics

8 Feedback and Participation

The study revealed some worrying and persistent trends regarding the failure of organisations to communicate effectively with practitioners about quality. Figure 8 shows that practitioners thought that quality performance feedback was missing at all levels.

For a quality improvement initiative to be successful, people must have information on quality performance.

The survey also revealed a shortfall in developer participation in constructing and managing quality improvement programmes. For example, only 28% of practitioners said that they thought developers had any input at all in determining the quality measures that had been implemented.

Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

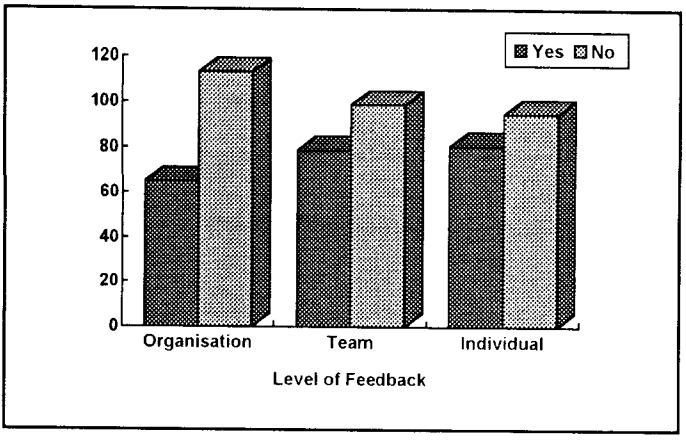

Figure 8 : Do you receive enough quality feedback ?

Direct participation in the measurement activity, however, was seen to improve practitioners attitudes to quality measurement. The more time practitioners spent working with measurement, the more positive they were about measurement. For example, only 17% of the practitioners who spent less than 1 hour per week on measurement felt that measurement was a "very useful" activity, compared to 40% of practitioners who spent more than one hour per week on measurement.

9 Summary and Conclusions

Overall, the practitioners in the study were generally optimistic about the use of quality control mechanisms and realistic about the current state of quality. However the survey suggested that the way in which these quality mechanisms were being managed could be significantly improved. Basic implementation errors were all too obvious. For example, organisations did not seem to be involving developers enough during the implementation process, nor did they seem to be providing enough feedback to developers. These errors could have almost certainly been avoided had the advice contained within the existing quality literature been heeded.

The study also revealed a worrying level of quality goal incongruence. This was not only present between developers and managers, but also between managers, developers and 'the' organisation.

The organisations in this study were actively seeking to improve quality and were trying hard to implement mechanisms that would allow them to do this. Predictably they had problems in getting the quality framework right. However they were keen to find out what their staff would say to an independent researcher and act upon the results presented to them. But how many organisations do not ask their staff what they think about how things are going and whether what should be happening is actually happening ? Such

organisation are all too common, and these organisations will have great difficulty in improving quality effectively.

References

[Jones 94] Jones C, "Software Management : The Weakest Link in the Software Engineering Chain" *IEEE Computer* May 1994