University of Hertfordshire
Computer Science Technical Report Nº 367 (Feb 2002)

Persuading developers to 'buy into' software process improvement: an exploratory analysis

Austen Rainer, Tracy Hall, Nathan Baddoo

Centre for Empirical Software Process Research
University of Hertfordshire
Department of Computer Science
Hatfield Campus, College Lane
Hertfordshire, AL10 9AB
England

a.w.rainer@herts.ac.uk

In order to investigate practitioners' opinions of software process and software process improvement, we have collected information from 13 companies, in a variety of ways i.e. the use of Repertory Grid Technique, survey and focus group discussions. Both the Repertory Grid Technique and the focus group discussions (43 discussions occurred, in total) produced a large volume of qualitative data. At the same time, other researchers have reported investigations of practitioners, and we are interested in how their reports may relate to our own. Thus, other research publications can also be treated as a form of qualitative data. In this paper, we review advice on a method, content analysis, that is used to analyse qualitative data. Content analysis is a method for identifying and classifying words and phrases used in ordinary language. We use content analysis to describe and analyse discussions on software process and software process improvement. We report preliminary findings from an analysis of both the focus group evidence and some publications. Our main finding is that there is an apparent contradiction between developers saying that they want evidence for software process improvement, and what developers will accept as evidence. This presents a serious problem for research: even if researchers could demonstrate a strong, reliable relationship between software process improvement and improved organisational performance, there would still be the problem of convincing practitioners that the evidence applies to their particular situation.

Keywords: empirical study, case study, content analysis, software process, software process improvement, opinions

1 Introduction

There is a growing body of research, some of it empirical, that reports on the effects of software process improvement (SPI) programmes. Some of this research considers the benefits of SPI programmes on organisations at both lower-levels [1] and higher-levels [2-4] of process maturity. Such benefits include increases in productivity, reductions in cost, reductions in duration, increases in product quality, and improvements in process stability. Some other research, however, suggests possible negative effects of SPI. For example, Kuilboer and Ashrafi's [5] survey of developers *suggests* that companies conducting SPI for a longer period of time showed an overall increase in development cost and project duration. Gray and Smith [6] criticise process assessment and improvement on theoretical grounds. Their most fundamental criticism is that the software research community still only has a poor understanding of the software process. This criticism is similar to previous observations made by Abdel-Hamid and Madnick [7] and Remenyi and Williams [8]. Over a decade ago, Abdel-Hamid and Madnick observed that we still lack a fundamental understanding of the software development process, and used this as a motivation for developing system dynamic models of software projects. More recently, Remenyi and Williams [8] observed that we lack an established theory of software development, and proceeded to argue for a grounded-theory approach (e.g. [9, 10]) to investigating the software process.

One important aspect of process engineering is implementing a new, or modified, process. While the research community and industry needs to better understand process, so the research community and industry also needs to better understand the implementation of process. As part of the Practitioners,

Processes and Products (PPP) project, we are investigating practitioners' opinions of software process and software process improvement. Our focus is on understanding the difficulties experienced by practitioners during the implementation of SPI programmes, with the intention that this understanding may lead to improvements in programme implementation. The PPP project emerged from previous investigations that we have conducted on the relationships between human factors in software development and software quality (e.g. [11-13]).

In order to investigate practitioners' opinions, we have collected information from practitioners at 13 companies, and collected such information in a variety of ways i.e. through the application of the Repertory Grid Technique, a survey and focus group discussions. Both the Repertory Grid Technique and the focus group discussions (43 discussions occurred, in total) have resulted in a large volume of qualitative data. (The questionnaire has collected quantitative data.) We are also interested in investigating findings published by other researchers. Such publications may also be treated as a form of qualitative data.

This paper reports our investigation of an appropriate method, content analysis, for analysing 'ordinary language'. The paper also presents results of some initial analyses. We have already reported findings from an analyses of the data collected through the Repertory Grid Technique [14, 15].

Content analysis is an unusual method for software engineering research. Also, we acknowledge the arguments and advice of Fenton, Pfleeger, Kitchenham and Glass (e.g. [16-20]) to document and improve our methods of analyses. For these reasons, we direct a substantial amount of attention at discussing the method. This discussion emphasises:

- That the investigation of ordinary language offers considerable potential for gaining insights into practitioners' and researchers' opinions.
- That the analysis of ordinary language must address potentially significant difficulties.
- That content analysis, as used here, is a method for identifying and classifying words and phrases used in ordinary written language.
- That content analysis, as used here, is treated as an initial (although substantial) investigatory phase, producing classifications that are subsequently analysed by other means.
- That content analysis is one method in a multi-method approach being used by the PPP project.

Two sets of analyses were conducted. In the first set of analyses, we analysed a transcription of a group discussion about SPI between developers within Company 2. In the second set of analyses, we analysed four published research papers on software process improvement. This second set of analyses is analyses of secondary data originally collected and analysed by other researchers. Overall, analysing two different types of communication allows us greater insight into the feasibility and desirability of using the content analysis of language to understand people's opinions of the software process. It may also act as a form of cross-validation, in that similar insights may be drawn from different types of data.

2 Ordinary language and content analysis

Because the content analysis of ordinary language is a novel approach to investigating the software process, we have looked outside of the software engineering research literature to gather advice on this approach. The main sources that we have drawn from are: Bromley's account of analysing ordinary language descriptions of personality [21]; Holsti's guide to content analysis as an approach to documentary research [22]; Strauss's handbook for qualitative analysis for social science [10]; and Miles and Huberman's sourcebook of qualitative data analysis [23]. While each of these texts has its own particular focus, they all contribute important advice for analysing language. Additional work, such as that of Reddy [24] and Weber [25] would also be relevant were one to conduct a more exhaustive review of the literature.

2.1 Ordinary language

Bromley [21] defines the term *ordinary language* as:

> "… natural ways of speaking and writing in everyday life, as contrasted with specially contrived notations, displays and terminologies." ([21], p. ix)

This definition is fairly easily applied to software practitioners within industry recognising, however, that these practitioners will develop and use their own idioms, such as using terminology (e.g. three letter acronyms) to refer to the technical substance of their work. For these practitioners, their language is 'ordinary' in that it is used in *their* everyday work. (One may argue that focus group discussions are not an ordinary activity for practitioners. Practitioners do, however, have group discussions as part of their everyday work e.g. design meetings, reviews and inspections.) The definition of ordinary language may also be applied to researchers: while their language may be unusual compared to other professionals or lay people, for people who practise software engineering research their language is ordinary because, again, it is used in their everyday work. One significant exception, however, may be the fact that researchers carefully draft their publications.

Because of the complexity and richness of language, and thus its ability to express ideas, the investigation of ordinary language offers considerable potential for gaining insights into practitioners' and researchers' opinions; specifically their opinions about software process and software process improvement. Such insights may help industry and academia to better understand why successful software process improvement programmes are so difficult e.g. the difficulties caused by practitioners' resistance to change.

There are, however, potentially significant difficulties in analysing ordinary language. The meaning of many, perhaps most, words and phrases are modified, subtly or grossly, by the context [10, 21]. Also, a text may have both 'surface' meaning(s) and deeper meaning(s). As examples, consider metaphors and puns. Finally, transcriptions introduce additional problems because they do not represent much of the verbal and non-verbal information that is present in spoken language e.g. stresses, pauses, facial expressions.

Strauss [10], amongst others, addresses these potential difficulties. He argues that although an analyst may misinterpret any particular phrase, and may not even settle on a particular interpretation, the analysis is still useful because it enriches the inquiry; it generates conjectures and ideas that can be refined later in the analysis. Strauss also argues that subsequent analysis may be used to test the validity of the previously generated conjectures (*cf.* Yin's [26] discussion of the replication of case studies and experiments). Similarly, Remenyi and Williams [8] would argue that the value of analysing ordinary language is that it produces concepts that are more or less *useful* (for developing our understanding) rather than more or less true. These issues are considered in more depth in a later sub-section.

2.2 Content analysis

Holsti [22] reviews several definitions of the term *content analysis*, commenting that there has been a marked tendency toward viewing content analysis as a basic research tool which may be useful in various disciplines and for many classes of research problem. Holsti recognises that some researchers treat content analysis as the *quantitative analysis* of texts, for example counting the frequency of occurrence of particular words (Weber [25] emphasises this approach.) This is not a position taken by Holsti, however, who argues that content analysis also includes the qualitative analysis of texts. Holsti identifies the need for content analysis to be objective, systematic and theoretically relevant, states that these three requirements are necessary conditions for all scientific inquiry, and from these concludes that content analysis is the application of scientific method to documentary evidence.

Bromley provides comments that complement Holsti, but within the context of investigating personality:

> "For our purpose the term 'content analysis' refers to a method for identifying and classifying words and phrases used in ordinary written language to describe and analyse personality." ([21], p. 37)

Clearly, we have a different subject for the analysis i.e.

> For the purpose of the PPP project, content analysis refers to a method for identifying and classifying words and phrases used in ordinary (written) language to describe and analyse software process and software process improvement.

Note the presence of four types of inquiry: identifying, classifying, describing and analysing. Note also an implied sequence to these types, and an implied boundary to the focus of content analysis i.e.

> For the purpose of the PPP project, content analysis refers to a method for identifying and classifying words and phrases used in ordinary written language *in order to subsequently* describe and analyse software process and software process improvement.

This suggests that content analysis may be treated as an initial, although substantial, investigatory phase producing classifications that are subsequently analysed (or interpreted) by other means. For example, a quantitative content analysis that produces a count of the frequency of occurrence of particular words subsequently requires an interpretation of what that frequency means.

2.3 The 'ordinary reading' of 'ordinary language'

One may argue that because much information is lost during the transcription process, or because of the difficulties in determining the exact meaning of the text, one should identify general themes expressed in the text, rather than attempting to identify and define detailed issues. Phrased another way (and perhaps simplifying) one should read through the text (perhaps several times) and get a 'feel' for the main themes being expressed there.

Holsti cautions against relying solely on this 'ordinary reading' of texts, and employing what he describes as "a sort of sixth sense that will alert you to tell-tale signs." He writes:

> "The difficulty with such advice is not that it is wrong, but rather that it may be insufficient. Intuition, insight, or a brilliant flash [of inspiration], borne of experience, thorough knowledge of one's data, imagination, or luck are perhaps always present in creative research. The 'folk wisdom' that 'the facts speak for themselves' is decidely not true. Hence there is always a place in research for such intangible qualities as intuition and imagination. But the same idiosyncratic qualities of intuition which render it important in some stages of research, especially in originally formulating the problem and in drawing inferences from the data, makes it less useful in others. Intuition is not a substitute for objectivity, for making one's assumptions and operations with data explicit where they are open to critical purview. Nor is it a substitute for evidence." ([22], p. 19)

Strauss adopts a similar position to Holsti. Strauss recognises that a contrasting approach to a minute analysis of texts is to read through the data quickly, yielding an "impressionistic cluster of categories". Strauss does not recommend this contrasting approach, however, stating that it produces "… conceptually thin and often poorly integrated theory." ([10], p. 31). (There is, of course, the assumption here that one wants to produce theory. One may be interested in only describing a phenomenon, prior to attempting to explain it.)

To summarise this issue of the 'ordinary reading' of 'ordinary language': if one is analysing ordinary language then one should use a method that encourages a systematic approach; an approach that makes one's assumptions and operations with the data explicit and available for public inspection. An 'ordinary reading' of 'ordinary language' is insufficient for scientific inquiry. In addition, however, all methods have their limitations and a general strategy for dealing with the limitations of any particular method is to employ contrasting methods. So, for example, the PPP project has combined survey research, Repertory Grid Technique and focus group discussions. Different methods for analysing different datasets, where these datasets are collected in different ways, helps to compensate for limitations. Additionally, one should also compare one's findings with literature, in an attempt to identify confirmatory and dis-confirmatory evidence [27].

3 Method

Our review of the work of Bromley, Holsti, Strauss, and Miles and Huberman have informed our development of a method for analysing the transcriptions and publications. As indicated in the introduction, we conducted two sets of exploratory analyses. In this section, we first discuss the general method we used and then consider issues specific to the transcript and the publications.

3.1 Applying the method

We use the following method to analyse the qualitative data:

1. Select the texts to analyse.
   We chose the developers' transcription from Company 2 because we considered that the issues raised in the company (from our experience of collecting the evidence) were not too complex, so that we would have a fairly 'simple' text to analyse. The selection of papers was more serendipitous, and is discussed in more detail later in this paper.

2. Identify units of text.
   Units of text may be single statements, or paragraphs of text. The statements from the transcription were easily identified. This is partly because the transcription was a simplification of the discussion. Statements from the papers were harder to identify, because it is not always clear how much of a statement is sufficient: what counts as a statement depends on what kind of thing we are interested in. Having identified a unit of text in one paper (or the transcription), we sought similar and dissimilar units from the same paper (or the transcription), and from the other papers being analysed.

3. Identify key words from each unit of text.
   Again, this is partially influenced by the kind of thing we are interested in, and what we are looking for. But again, thinking about one key word in one unit can suggest contrasting key words in other units. It is also important to identify key words in several sessions of analysis. This is because the analyst may come to a new session, with a different perspective, and this will help to identify new key words.

4. Think about each key word. Ask the following kinds of questions:
   - What are the different key words?
   - What ideas is each key word expressing?
   - What ideas could each key word be expressing?
   - How does the use of this key word, in this unit of text, compare with the use of the same, and different, key words in other units of text?
   - How do the ideas being expressed with this key word, in this unit of text, compare with ideas being expressed with other key words in other units of text?
   - How do the ideas being expressed with this key word, in this unit of text, compare with ideas expressed in other people's work? Cite the other work explicitly.
   - Are the key words expressing specific ideas for which there are more general ideas?

   Some of these questions focus on the identification of words taken directly from the text. Other questions focus on what these words may mean. Both foci are important for the analysis because they make the analysis more explicit.

3.2 Analysing the ordinary language of developers

As already noted, we have collected a variety of evidence from practitioners at 13 companies. Practitioners were grouped into senior management, project management, and developers. For each group of practitioners, we conducted focus group discussions. These sessions were attended by between three and six members of a respective group. (In some companies, we were able to conduct more than one session for a particular type of group.) In each session, the practitioners were asked to answer and discuss several questions. For this analysis we have focused on the discussion of the following question:

> *What are the potential motivators to software process improvement in your company?*

A second question was also used, as a prompt:

> *What will make it [i.e. software process improvement] happen?*

Table 1 presents the transcription of the developers' discussion.

Table 1 Transcription of the developers' discussion

#	Text
1	If we could see it work
2	If we have evidence of benefits
3	If it allows you transparency into the current processes
4	If it is imposed. Make it a "got to do it"
5	If it is introduced via phasing. And introduced into a small area and people can see the benefits then […]
6	[…] they will buy in.
7	If it improves the configuration management aspect of our development
8	If we can all work in a standard way

As the table indicates, the transcription is actually quite short, particularly for a group discussion. This is due, in part, to the fact that this question was only one of several questions being asked of the developers. Consequently, developers were not expected to spend too long discussing the question being asked. Also, the transcription has been 'tidied up'. From a pragmatic perspective, a small transcription is easier to analyse. As discussed earlier, the analysis of the four publications is considerably more demanding, due to the large volume of text that needs to be considered.

3.3 Analysing the ordinary language of researchers

Table 2 Papers reviewed in this report

Author	Method	Logic	Sample	Country
Sharp *et al*. [28]	ethnography	inductive	mixed	Unknown (probably UK)
Laporte and Trudel [29]	case study	historical	one	America
Moitra [30]	anecdotal	historical	unknown	India
Stelzer and Mellis [31]	formal literature review	inductive-deductive	56	Europe & America

Table 2 provides a summary of the four papers that have been analysed. As the table indicates, there are a mixture of research methods, logic of analysis, samples sizes, and sources of the samples. This mixture is desirable because the papers complement each other in various, different ways.

Laporte and Trudel [29] report on the process improvement activities that occurred at a defence contractor, Oerlikon Aerospace, over several years. In particular, they focus on the 'people issues' of process improvement.

Stelzer and Mellis [31] conducted a two-stage study. In the first stage, they proceeded inductively, exploring literature on factors that affect organizational change, interviewing managers from German software companies that had implemented ISO-based software process improvement, and analysing experience reports and case studies from European software companies that had implemented ISO-based quality systems. Through these investigations they compiled a list of ten factors that seemed to influence the success of organisational change in software process improvement efforts. In the second stage of the study, the researchers proceeded deductively, analysing published experience reports and case studies. The experience reports and case studies were organised into two sets: one set consisting of reports and studies relating to ISO-based certification; the second set relating to CMM-based improvement efforts. For each report or case study, the researchers examined whether each factor was reported in that report or case study (with a binary scale of reported or not reported).

Sharp *et al*. [28] report on three studies that they have conducted: the analysis of videotaped presentations and discussions at a conference, a discourse analysis of archival data (e.g. trade magazines, journals and conference proceedings), and the analysis of evidence (for example, collected through interviews) from five companies.

Moitra [30] provides a pragmatic approach to managing change in software process improvement efforts, based on her many years of experience designing and implementing improvement programmes in many high-tech organisations in India.

The selection of papers occurred serendipitously in that they were part of a larger group of papers, relating to organisational change and software process improvement, that we were compiling. It became clear that the differences in these four papers (e.g. different research methods, sample sizes) meant that an analysis of these four papers might produce some interesting and useful insights; insights that could complement or contrast those drawn from the analysis of the developers' discussion. Due to the intensive nature of the analysis, the analysis of a larger number of papers was impractical. A quantitative content analysis of a larger sample of papers may be useful, and stands as one opportunity for developing this research.

The language used by researchers is more technical and formal than the language used by practitioners. This is not a comment about the relative competence of practitioners and researchers, but rather a comment on the process of communication. Researchers often choose to communicate in writing, as this allows the development of a more abstract and complex argument. Verbal communication typically does not allow the development of arguments with comparable complexity. Written communication may present separate difficulties for analysis compared to transcriptions of verbal communication.

4 Summary of the analysis

Table 3 summarises the main 'opinions' identified in the analysis, the source of those opinions, and some examples of the statements that express those opinions.

Table 3 Summary of opinions identified during the content analysis

		Focus	Publications				
	Opinion	**group**	**[29]**	**[30]**	**[28]**	**[31]**	**Example statements**
1	Developers want evidence of the benefits of SPI	Yes					See lines 1,2 & 5 of Table 1.
2	Most developers are sceptical about process improvement			Yes		Yes	"I have found that the resistance for (sic) change is mainly because of a perception of: (i) uncertainty and skepticism about the effectiveness of the new processes and the possible benefits from them…" ([30], p. 201)
3	Developers are passionately committed to the excellence of what they do				Yes		(See comments on opinion #4.)
4	Developers believe that they can achieve very high standards				Yes		
5	Prominence of the individual			Yes	Yes		"The firm belief in their own abilities indicates the prominence of the individual that we found in all companies, and which at times was dramatic. In one company, we found a local guru whose technical judgement was always deferred to…" ([28], p. 46)
6	Preference for local expertise				Yes	Yes	"They (opinion leaders) often act as advisors, advocates and communication liaisons." ([31], p. 238)
7	Discount empirical evidence in favour of local opinion				Yes		
8	Advocation of an incremental approach to SPI	Yes	Yes			Yes	See lines 5 & 6 of Table 1. "… a prime source of ideas should come from those people who are working, on a daily basis, with the processes…" ([29], p. 195) "Staff members should be involved in the improvement initiatives because they have detailed knowledge and first hand experience of strengths and weaknesses of the current processes." ([31], p. 236)
9	Developers focus on the 'doing' of the process	Yes					See lines 3,7 & 8 of Table 1.

Given that four papers are reviewed there are actually a surprisingly small number of opinions identified in Table 3. This is a reflection of the fact that the analysis of the papers was focused by the issues identified from the transcription. A further point of interest is that the publication that expressed the most ideas, Sharp *et al*. [31], is the publication that is most similar, methodologically, to the current investigation.

4.1 Evidence, opinion and the credibility of knowledge

As indicated in Table 3, the developers claim that evidence of the benefits of process improvement is a potential motivator for process improvement in their company. But Sharp *et al*. [28] found that practitioners prefer local expertise and *discount* empirical evidence in favour of (personal) opinion. Sharp *et al*.'s additional findings, that developers are committed to the excellence of what they do and believe that they can achieve very high standards, underpin (and perhaps explain) their preference for local expertise. Stelzer and Mellis [31] and Moitra [30] both claim that developers are sceptical. These claims can be taken as support for both the claims of the developers (i.e. that they want evidence) and the claims of Sharp *et al*. (i.e. that at least some types of evidence are not acceptable) Thus, there is an apparent contradiction between the developers saying that they want evidence, and what the developers will accept as evidence.

Table 4 Credibility of knowledge

Source of knowledge	Type of knowledge	
	Opinion	**Empirical**
Local	1 (most)	2
Remote	3	4 (least)

There is some suggestion, then, for a hierarchy of knowledge, such as that presented in Table 4. In such a hierarchy, local opinion may be the most credible type of knowledge and remote empirical evidence the least credible. Such a hierarchy appears to contrast with the type of knowledge typically valued by academics. It would seem logical for academics to place a high value on empirical evidence and to place a low value on opinion. But against that, and considering the sociology of science, an individual researcher may evaluate empirical evidence against, or with, their personal opinions and values, and not necessarily evaluate empirical evidence against other empirical evidence.

McCroskey's investigations (e.g. [32], but see also [33-35]) into persuasive communication provides an example that supports the suggestion of a hierarchy of knowledge. McCroskey argues that a speaker should first draw upon the opinions, values and attitudes already held by the audience; that the speaker should then draw on their own opinions, values and attitudes; and only when these two strategies fail (or, as a complement to either of these two strategies) the speaker should draw on third-party facts and opinion.

The issue of the credibility of knowledge, and the preference for local opinion, presents a serious implication for empirical research on software process improvement. Even if researchers could demonstrate a strong, reliable relationship between software process improvement and organisational performance, there would still be the problem of convincing practitioners that the evidence applies to their particular situation. Phrased another way, there would still be the need to 'transform' the empirical evidence into local opinion. The recognition of the need to tailor process models and the recognition of the need to calibrate estimation models (e.g. [36, 37]) both support the argument that each organisation is distinct, and both undermine any assumption that a set of findings regarding software process improvement would *ipso facto* apply to another organisation.

4.2 Local experts

Local experts are, presumably, valuable for at least two reasons. First, the person is an expert in that they possess technical knowledge of the application being developed, and the methods being used to develop that application. Second, the fact that the person is local allows colleagues to become familiar, over time, with the skills and knowledge of the expert. (The expert demonstrates their competence over time.) There may also be a third value, one of leadership. It may not just be that the local expert has an opinion but that they are an opinion *leader*.

4.3 Incremental software process improvement

The issue of familiarity may help to explain the advocation, by some developers and some researchers in the data analysed, of an incremental approach to software process improvement. Developers are already familiar with the strengths and weaknesses of the current process. It may be that developers want to become familiar with the changes that are being proposed: familiar with the benefits and drawbacks that these changes bring. In describing techniques for bottom-up process improvement, Jakobsen [38] writes of 'rhythm's power': "We feel safe with the everyday rhythm of our lives..." ([38], p. 66). Jakobsen goes on to describe how the change, in his company, from process-driven to time-driven activities can changes people's habits: "After two weeks, *people got into the habit*..." ([38], p. 66; emphasis added).

4.4 The 'doing' of the process

Developers appear to focus on the benefits relating to the *doing* of the process. For example, no references were made to quality, productivity, cost or duration (see Table 1). Instead, developers referred to configuration management control, transparency of the process and standard ways of working.

Cost, quality, duration and productivity are all issues that would interest managers. The differing interests of developers and managers are consistent with their differing roles. Managers are not so interested in the detail of actually doing development (although perhaps they should be), but are interested in the inputs and outputs of that development. Developers, by contrast, would obviously be interested in the doing of the process. One implication of this difference is that developers may place different value(s) or expectations on software process improvement to that of managers; and a consequence is that attempts to gain developer 'buy in' must address issues different to those valued by management. This clearly relates back to the issues of scepticism and what counts as evidence of benefits. Developers may be sceptical because they are not being provided with information on the benefits to the doing of the process. Conversely, addressing developers' concerns about how SPI will improve the doing of the process may help to persuade developers that SPI is worthwhile.

Through publications, managers see that other companies have reduced costs, improved productivity etc. Managers see the benefits that they are looking for. But reports from other companies (whether they are research publications, company case studies, opinions from gurus' or opinion leaders) provide little information on how the process changed. Consequently, developers are not provided with information on the 'doing' of the process. Also, it may be that the process will be different for different companies, so again, developers may find it harder to relate to these companies, and may remain sceptical of the improvements.

5 Discussion

The content analysis of one transcription and four publications has produced some interesting findings. These findings are interesting because they suggest reasons for difficulties in successfully implementing SPI programmes e.g. that developers want evidence of benefits relating to the 'doing' of the process, and that developers seem to favour local opinion over independent empirical evidence. The findings are also interesting because they suggest further questions e.g. What is the value of local empirical evidence? How does one improve the value of independent empirical evidence?

Given the small sample size, it is necessary to conduct further analysis using additional focus groups to validate these findings. As noted earlier, we have 43 focus group discussions from 13 companies, and we intend to conduct further analyses. Furthermore, we have other datasets (survey data and Repertory Grid Technique data) that may also contribute to this analysis.

From a methodological viewpoint, content analysis appears to be useful for analysing ordinary language and generating interesting insights. Thus, content analysis provides a method for analysing evidence that it naturally produced by organisations and their projects. More specifically, content analysis provides a method for analysing unstructured evidence (such as meeting minutes e.g. [39]), and this method complements the automated collection and analysis of quantitative evidence naturally produced by projects (e.g. [40-42]).

As noted in the earlier sections of this paper, there are some potentially significant difficulties with this method. Our experience from using content analysis suggests:

- That content analysis is demanding in terms of time and effort. This is because it encourages a very intensive analysis. Content analysis is also rewarding, however, in the insights that it generates (or uncovers). Careful preparation, such as in one's research design, may reduce the workload. Also, the use of quantitative content analysis may help focus the qualitative analysis e.g. focus on words that occur frequently (excepting such words as 'a' and 'the').
- That there are difficulties in systematically identifying and categorising concepts or ideas expressed in the ordinary language of practitioners and researchers. This is partly due to the difficulties in understanding the 'true' meaning of a text (discussed earlier in section 2). Bromley, Holsti, Strauss, and Miles and Huberman all provide useful advice on the identification and classification of concepts and ideas.
- That there are difficulties in organising, 'compressing' and comparing categories. Earlier, we argued that two strengths of language are that language is rich and complicated (as this allows the expression of rich and complicated ideas). There is, then, an inherent problem in simplifying and structuring this complexity and richness.

6 Conclusions

This paper has reported some exploratory work on content-analysing the 'ordinary language(s)' of practitioners and researchers. The paper has reviewed advice on conducting content analysis, has presented a simple method for conducting such an analysis, has reported some preliminary findings, and has briefly reflected on the value of content analysis.

Central problems with analysing such qualitative evidence are:

- The intensive nature of content analysis, and its demands in terms of time and effort.
- Systematically identifying and categorising concepts or ideas expressed in the ordinary language of practitioners and researchers.
- Organising those concepts and ideas so that they can be managed and further analysed.

The main finding from this analysis is that there is an apparent contradiction between developers saying that they want evidence, and what developers will accept as evidence. This main finding is related to issues such as hierarchies of knowledge, the value of empirical evidence to practitioners, local expertise, an incremental approach to improvement that may develop familiarity with those improvements, and differences between developers and managers with regards to their interest in the process. A serious implication follows from the main finding: even if researchers could demonstrate a strong, reliable relationship between software process improvement and organisational performance, there would still be the problem of convincing practitioners that the evidence applies to their particular situation (that the evidence counts as evidence!).

Acknowledgements

We are sincerely grateful to all the companies and practitioners (who, for reasons of confidentiality, must remain anonymous) for their participation in the PPP project. The PPP project is funded by the UK's Engineering and Physical Science Research Council (EPSRC), under grant number GR/L91962.

References

1. Wakulczyk, M., *Success Is Not Accidental: CMM Level 2 In 2.5 Year.* CrossTalk, 1997(Sept): p. 1-5.
2. Paulk, M.C., et al., *A High-Maturity Example: Space Shuttle Onboard Software*, in *The Capability Maturity Model: Guidelines for Improving The Software Process*, M.C. Paulk, et al., Editors. 1994, Addison-Wesley: Harlow, England.
3. Paulk, M.C., D. Goldenson, and D.M. White, *The 1999 Survey of High Maturity Organizations*. 2000, Software Engineering Institute, Carnegie Mellon University.
4. Fiztgerald, B. and T. O'Kane, *A Longitudinal Study of Software Process Improvement.* IEEE Software, 1999. **16**(3): p. 37-45.
5. Kuilboer, J.P. and N. Ashrafi, *Software Process And Product Improvement: An Empirical Assessment.* Information and Software Technology, 2000. **42**: p. 27-34.
6. Gray, E.M. and W.L. Smith, *On The Limitations of Software Process Assessment And The Recognition of A Required Re-orientation For Global Process Improvement.* Software Quality Journal, 1998. **7**: p. 21-34.

7. Abdel-Hamid, T.K. and S.E. Madnick, *Lessons learned from modeling the dynamics of software development.* Communications of the ACM, 1989. **32**(12): p. 1426-1438.
8. Remenyi, D. and B. Williams, *Some aspects of methodology for research in information systems.* Journal of Information Technology, 1995. **10**(3): p. 191-201.
9. Glaser, B.G. and A.L. Strauss, *The Discovery of Grounded Theory: Strategies For Qualitative Research*. 1967, New York: Aldine de Gruyter.
10. Strauss, A.L., *Qualitative Analysis For Social Scientists*. 1987, Cambridge: Cambridge University Press.
11. Hall, T. and N. Fenton, *Software Quality Programmes: A Snapshot of Theory Versus Reality.* Software Quality Journal, 1996. **5**(4): p. 235-242.
12. Hall, T. and D.N. Wilson, *Views of Software Quality: A Field Report.* IEE Proceedings on Software Engineering, 1997. **144**(2): p. 111-118.
13. Hall, T. and D.N. Wilson, *The Real State of Software Quality - Practitioners' Experiences*, in *Software Quality Management V: The Quality Challenge*, C. Hawkins, M. Ross, and H.C. Sharp, Editors. 1997, Mechanical Engineering Publications Ltd: London UK. p. 111-118.
14. Hall, T., D.N. Wilson, and N. Baddoo. *Towards Implementing Successful Software Inspections*. in *International Conference on Software Methods and Tools (IEEE Computer Society)*. 2000. Wollongoing, Australia, 6th-10th November.
15. Hall, T., et al. *Optimising Software Measurement Programmes Using Practitioner Input*. in *Australian Conference on Software Measurement*. 2000. Sydney, 1st-3rd November.
16. Fenton, N., S.L. Pfleeger, and R.L. Glass, *Science and substance: A challenge to software engineers.* IEEE Software, 1994. **11**(4): p. 86-95.
17. Glass, R.L., *The software research crisis.* IEEE Software, 1994. **11**(6): p. 42-47.
18. Glass, R.L., *A Structure-based critique of contemporary computing research.* Journal of Software Systems, 1995. **28**: p. 3-7.
19. Kitchenham, B., L. Pickard, and S.L. Pfleeger, *Case Studies for Method and Tool Evaluation.* IEEE Software, 1994(July): p. 52-62.
20. Kitchenham, B.A., *Evaluating Software Engineering Methods and Tools. Part 2: Selecting an appropriate evaluation method - technical criteria.* Software Engineering Notes, 1996. **21**(2): p. 11-15.
21. Bromley, D.B., *Personality Description in Ordinary Language*. 1977, London: John Wiley & Sons.
22. Holsti, O.R., *Content Analysis For The Social Sciences And Humanities*. 1969, London: Addison-Wesley.
23. Miles, M.B. and A.M. Huberman, *Qualitative Data Analysis*. 2nd ed. 1994, London: SAGE Publications.
24. Reddy, M.J., *The Conduit Metaphor: A Case Of Frame Conflict In Our Language About Language*, in *Metaphor And Thought*, A. Ortony, Editor. 1993, Cambridge University Press: Cambridge. p. 164-201.
25. Weber, R.P., *Basic Content Analysis*. 2nd ed. Quantitative Applications In The Social Sciences, ed. M.S. Lewis-Beck. 1990, London: SAGE Publications.
26. Yin, R.K., *Case Study Research: Design and Methods*. 2nd edition ed. 1994: SAGE Publications.
27. Eisenhardt, K.M., *Building theories from case study research.* Academy of Management Review, 1989. **14**(4): p. 532-550.
28. Sharp, H., H. Robinson, and M. Woodman, *Software Engineering: Community and Culture.* IEEE Software, 2000: p. 40-47.
29. Laporte, C.Y. and S. Trudel, *Addressing the People Issues of Process Improvement Activities at Oerlikon Aerospace.* Software Process - Improvement and Practice, 1998. **4**(4): p. 187-198.
30. Moitra, D., *Managing Change for Software Process Improvement Initiatives: A Practical Experience-based Approach.* Software Process - Improvement and Practice, 1998. **4**(4): p. 199-207.
31. Stelzer, D. and W. Mellis, *Success Factors of Organizational Change in Software Process Improvement.* Software Process - Improvement and Practice, 1998. **4**(4): p. 227-250.
32. McCroskey, J.C., *A Summary Of Experimental Research On The Effects Of Evidence In Persuasive Communication.* The Quarterly Journal of Speech, 1969. **55**: p. 169-176.
33. McCroskey, J.C., *Toward An Understanding Of The Importance of "Evidence" In Persuasive Communication*. 1995.
34. McCroskey, J.C. and R.E. Dunham, *Ethos: A Confounding Element In Communication Research*. 1995.
35. McCroskey, J.C., V.P. Richmond, and J.A. Daly, *The Development Of A Measure Of Perceived Homophily In International Communication*. 1995.
36. Jeffery, D.R. and G. Low, *Calibrating estimation tools for software development.* Software Engineering Journal, 1990. **5**(4): p. 215-221.
37. Tate, G. and J. Verner, *Software Costing in Practice*, in *The Economics of Information Systems and Software*, R. Veryard, Editor. 1991, Butterworth-Heinemann: Oxford, UK. p. 101-126.
38. Jakobsen, A.B., *Bottom-Up Process Improvement Tricks.* IEEE Software, 1998: p. 64-68.
39. Rainer, A.W., *An Empirical Investigation of Software Schedule Behaviour*, in *Department of Computing*. 1999, Bournemouth University: Bournemouth UK.
40. Cook, J.E., *Process Discovery and Validation Through Event-Data Analysis*, in *Department of Computer Science*. 1996, University of Colorado: Boulder.
41. Cook, J.E. and A.L. Wolf, *Software Process Validation: Quantitatively Measuring The Correspondence Of A Process To A Model.* ACM Transactions on Software Engineering and Methodology, 1999. **8**(2): p. 147-176.
42. Wolf, A.L. and D.S. Rosenblum. *A Study In Software Process Data Capture And Analysis*. in *2nd International Conference on the Software Process*. 1993. Berlin, Germany, February 25-26.