Identifying the causes of poor progress in software projects

Austen Rainer and Tracy Hall
Centre for Empirical Software Process Research
Department of Computer Science
University of Hertfordshire
College Lane, Hatfield, Hertfordshire AL10 9AB, England
{a.w.rainer, t.hall}@herts.ac.uk

Abstract

In this paper we present data on the progress of two projects at IBM Hursley Park. Each project lasted approximately 12 months in duration. We use the data to identify the areas in the projects where poor progress was occurring and to investigate the causes of this poor progress. We find some similarities between the two projects in terms of where some poor progress is occurring i.e. with the design, code and test processes. But we also identify differences between the two projects that can be partially explained by the dependency of these two projects on other parts of IBM. We also find that limited quantitative data is reported in the projects, and that there is little explicit comparison of actual progress with planned progress. Indeed, the reporting of progress seems to be affected by factors like the 'deadline effect' and preferences for reporting certain types of progress. We conclude that these factors may threaten the valid reporting and management of the projects.

1. Introduction

In this paper we present data on the progress of two projects at IBM Hursley Park. Our data particularly highlights areas of poor progress in these projects. We relate our data on poor progress to data on waiting and overdue work in order to identify the problematic areas in software development projects.

One way in which poor progress is addressed in projects is to commit more effort to those areas where poor progress is occurring. The commitment of more effort can be achieved in a number of ways e.g. hiring additional staff, working shift work, or working overtime. Many estimation models recognise a close relationship between project duration and project effort. Consequently, committing more effort to problematic areas is likely to increase the probability that the project duration will be extended.

Ideally, *quantitative* measures of actual effort in software projects would allow researchers and practitioners to best understand where effort is being directed. Unfortunately, many software projects do not (or cannot) collect quantitative data on effort. The two projects that we report on here report little quantitative data. In this paper, we investigate the use of naturally occurring *qualitative* data to provide insights into project progress and, indirectly, into effort.

The data is based on reports of waiting, overdue work and the progress of work for two projects at IBM Hursley Park. It was collected from practitioners' comments during project status meetings over a 12 month period. The data has been collected on a week-by-week basis, and has been aggregated so that it applies at the project level (rather than a lower level within the project). We use these three sets of data to examine which functional areas of the project report most problems and which types of project work appear to be the cause of these problems.

As the data is based on reports made by practitioners, the data is subjective. Nevertheless, the context within which these reports were made i.e. during the highest-level meetings in each project, at which key project personnel were present, leads us to suggest that the data provides valuable insights into the progress of these two projects. While the results of our study may not generalise, we believe that our research strategy of collecting and analysing naturally-occurring qualitative

Proceedings of the 10th International Symposium on Software Metrics (METRICS'04)
1530-1435/04 $ 20.00 IEEE

data on progress, waiting and overdue work has wider applicability.

In section 2 we provide some background on the collection of qualitative and quantitative data. In section 3, we explain our study design. In sections 4 through 6, we present our evidence and analysis of waiting, overdue work and the progress of work. In section 7 we summarise our findings and briefly discuss their implications. We also indicate further analysis that we intend to conduct with this evidence. In section 8, we provide brief conclusions.

2. Background

One of the major difficulties with investigating software development is collecting detailed and accurate quantitative data. Survey results from the Software Engineering Institute suggest that most companies do not collect quantitative data [1]. Where companies do not collect quantitative data, it is difficult for researchers to collect their own quantitative data from projects in those companies.

However, it is often possible to 'generate' *quantitative* data from other kinds of data that are collected within a project. For example, Cook *et al*. [2] were able to 'recover' quantitative data on conformance to process from events occurring in a defect-fixing process. Bradac *et al*. [3-5] were able to use a lead developer's personal time diary to investigate the use of time by that developer over a 30-month time period. In their investigation, Bradac *et al*. focused particularly on the effects of waiting, but recognised that it was not clear whether their findings for an individual developer would scale up to the project level. (See [6] for a detailed re-analysis of Bradac *et al*.'s published data).

Bradac *et al*.'s difficulties with scaling up their findings to the project-level illustrate a more general difficulty in empirical software engineering research: collecting quantitative, *longitudinal* data at the project-level. Porter *et al*.'s [7] study of code inspections provides another illustrative example. They have used quantitative data 'naturally' collected by a project over the duration of that project. Using the data they collected, Porter *et al*. suggest that 'development phase' confounds the relationship between code inspections and faults found. Nevertheless, Porter *et al*. are focusing on one process within the project, so their study is not taking a project-level perspective.

To get some sense of what may be happening in software projects at the *project level*, we have collected qualitative data that has been naturally produced by two projects at IBM Hursley Park. We have used this qualitative data to gain insights into the overall process. Our general research question is:

RQ How does progress vary during software projects?

3. Study design

3.1 Overview

We selected two projects for longitudinal case study (see below for further information). Our main criterion for selecting a project was that the project was planned to complete within approximately 12 months. In order to know when a project was *planned* to complete, the project must be close to completing its initial requirements analysis and planning phases. This means that our investigations focused on the progress of the two projects *after* the project plans were established.

We collected a variety of data on these two projects. Our primary source of data was the minutes of each project's status meetings. These meetings occurred on a weekly or fortnightly basis, and were attended by representatives of all the functional areas of the project e.g. marketing, finance, design/code, testing. (We provide further information on functional areas below.) We also conducted interviews with project members, and collected other information e.g. project plans.

3.2 The selection of projects

Five projects were initially selected for case studies from a candidate set of 16 projects, all taken from IBM Hursley Park. Almost immediately, there were problems gaining regular access to two of these projects, and these projects were dropped as case studies and replaced by a sixth project. As the evidence collection period progressed, it became increasingly clear that it would be impractical to maintain four case studies (because of the demands of collecting and analysing evidence from four cases), so the number of cases was further reduced to two, here called Project B and Project C. [8] provides more information on the 16 candidate projects, the criteria for selecting the original five cases, and more detail on the reduction of case studies from four to two.

Table 1 Characteristics of the projects

Characteristic	Project B	Project C
Size of support team	Support team of 50 people (separate from Project B).	Support team of 12 people (part of Project C)
Size of planned development team	approx. 38 people	approx. 3 people
Size of planned management team	approx. 6 people	approx. 3 people
Assignment of work between support team and development team	Developers are either support or development (but development may support in critical situations)	Developers 'own' components and both develop and support those components.
Role(s) of Project Leader	Project Leader	Project Leader, Design/Code Manager, Support Manager
Strategic value of product	Higher; long-term	Lower; mid- to short-term
Purpose of project	New functionality	Port to new platform
Type of product	Large, mission-critical, middleware legacy system	Large, middleware legacy system
Release sizes	36 KLOC	70 KLOC
Number of features/design changes	13 features (planned) 12 design changes (unplanned)	19 features (planned) and 11 features (unplanned)
Platforms	Mainframe	Workstation
Project status meetings	Yes	No, but design/code/test status meetings
Project duration (in weeks)	57 (planned and actual)	48 (planned) 59 (actual)
Product delivery week	52 (planned and actual)	48 (planned) 59 (actual)
Determination of project duration	Project end-date driven, due to market considerations	Project end-date driven, due to resource funding constraints

3.3 Characteristics of the projects

Table 1 compares the two projects according to a number of characteristics of the projects. Four entries in Table 1 require clarification. First, the strategic value of the two products is *relative* to the two products. Although Product C has a lower strategic value this is not to say that the product is not valued by the organisation (if the product had a low value to the organisation it is unlikely it would be maintained). Second, although design changes and additional features are unplanned, this is not to say that such work is unexpected. Experienced Project Leaders recognise that the workload for a project will probably increase. Third, the KLOC sizes of the two projects might misleadingly suggest that Project C is very much more productive than Project B. Product B is, however, a mission-critical product requiring very high levels of reliability. In addition, much of the code for Product C is being ported from an existing version of the product. Fourth, a feature is the most basic unit of development for a very large software system, and represents a long-term effort [9].

3.4 Functional areas in the projects

In any large software project, personnel involved in the project tend to be organised into teams, and teams tend to be assigned specific types of work. For example, there may be one or more design teams, and these will be separate from the test team(s). Some teams are not necessarily assigned *technical* work but are, instead, assigned other types of work e.g. financial management, marketing. Each of the projects that we studied comprised a large number of different teams. To emphasise the specific roles of these teams, we have referred to them as functional areas of the projects. The functional areas are summarised in Table 2.

Table 2 Functional areas of the projects

Functional area	Explanation
Build	A team responsible for building versions of the system for testing and, later, for release
Defect screen team	A team that is established to prioritise and assign defects to defect-fixers.
Design / Code	The team(s) responsible for the high-level and low-level design, and the actual coding.
Early market support	The team responsible for marketing the product to client companies that are 'early adopters' of technology
External organisation	An organisation external to IBM Corporation with which the project must interact.
Information development	A team that develops the documentation that accompanies the product.
Organisational 'units'	Other organisational 'units' in the corporation that may not be software or hardware, research or development, projects. For example, a corporate management team would be classified under this category.
Other project (within laboratory)	Other projects within IBM Hursley Park with which the project might interact e.g. sharing human resource.
Other projects within the organisation	Other projects within IBM Corporation but outside of IBM Hursley Park.
Performance	A team responsible for ensuring that the system's performance is not unduly affected by new functionality
Project management	The project management 'team' for the respective project
Service	Provide technical support / maintenance to customers
Test	There are various testing teams e.g. System test, performance
Unknown	This category is used to signify situations where it was not possible to identify the specific functional area from the data collected.

Table 3 Types of work causing poor progress and waiting, or that were overdue work

Type of work/waiting	Explanation
Decisions	Some decision that needs to be made.
Defect/Fix	A defect or fix relating to some piece of design, code or information.
Information	Information (e.g. documentation) relating to the product.
Code	Some source code for the system.
Resource	Equipment, such as test PCs.
Other	Other types of waiting
Unknown	The type of waiting could not be identified from the available evidence.

As noted earlier, we started investigating these projects once their requirements analysis and planning phases were completed. Therefore, we are unable to investigate the requirements analysis and planning phases which is why they are not presented in Table 2.

Our investigations identified a number of different types of work that were causing poor progress, 'causing' waiting, or that were overdue work. These types are summarised in Table 3.

Table 4 Definitions

Reports of	Definition
Waiting	A functional area cannot start or continue work on a task until some other functional area provides input. It is likely that the functional area that is waiting will work on some other task while they are waiting.
Overdue work	A functional area has not completed work that they expected to have completed. This indicates a difference between planned and completed work.
Poor progress of work	The rate at which work is actually completed is lower than the expected rate. This indicates a difference between planned and actual work rate. This may be due to an over-optimistic plan.

3.5 Data collection

As our data consists of practitioners' *reports* of waiting, overdue work and poor progress there is inevitably a major *subjective* component to these reports. The subjective nature of these reports implies that practitioners are intentionally or unintentionally 'filtering' the information that they report. We are assuming that practitioners' reports emphasise the more significant instances of waiting, overdue work and poor progress i.e. that practitioners filter out the less significant instances.

Table 4 provides simple definitions of our three main sets of data.

The primary source of evidence used in the analyses was the minutes of status meetings. Project B held meetings attended by representatives of all the functional areas in the project. Project C held meetings attended mainly by representatives of the design, code and test functional areas only. For both projects, the status meetings were the highest-level meetings within the respective projects, occurred regularly (typically weekly or fortnightly), were typically attended by representatives from functional areas important to the given project, and are a naturally occurring phenomenon (so that the researcher is not intruding on the project).

Overall, the status minutes provide a broad view of the project over the duration of the project. Naturally, minutes do not record all that was discussed at a meeting, or even necessarily the most important issues (e.g. for political reasons, a discussion at the meeting may not be reported in the minutes), and such meetings are unlikely to discuss all the issues occurring within the project at the time of the meeting. Despite these simplifications, the minutes provide a large volume of 'rich' information about the project over the duration of the project, and this information appears rich enough to provide substantive, longitudinal insights into progress in software development. Furthermore, the minutes provide detail that is unlikely to be collected from other sources of data, and detail that can also indicate simple causal connections between events. For example the Performance functional area of Project B reported that there was no progress on a particular feature being developed. This was because of an overdue fix to a severe problem with the feature. This was noted in the minutes:

> "Performance: No progress on [feature F07] due to outstanding sev 1 [severity 1] problem."

Such statements allow us to make connections between functional areas, overdue work and poor progress.

3.6 Data analysis

In order to investigate the characteristics of waiting, overdue work and the progress of work the minutes of the status meetings were searched, using a text editor, for particular phrases.

Table 5 summarises the phrases used for the search, including derivatives of a phrase. The terms presented in Table 5 are not exhaustive but cover all of the terms that were used in this investigation.

Having refined the set of references, each of these references (together with their surrounding 'chunk of meaning') was then copied into a separate text file and labelled with the week number in which it occurred. Each item was then classified in various ways. See [8] for more information on the methods of analysis.

Table 5 Phrases used for searching the minutes of status meetings

Evidence	Phrase	Derivatives (examples)
Reports of waiting	wait	waiting, awaiting, await
	block	blocked, blocking
	held up	
	hold	holding (holding up)
Reports of overdue work	outstanding	
	backlog	
Reports of progress of work	progress	

Table 6 Types of progress

	Project B		Project C	
Type of progress	**Count**	**%**	**Count**	**%**
Good progress	11	16.4	24	24.2
Reasonable progress	8	11.9	2	2.0
Slow progress	19	28.7	23	23.3
No progress	13	19.4	22	22.2
Other types	16	23.6	28	28.3
Total	67	100.0	99	100.0
Poor progress	32	48.1	45	45.5

Note:
Poor progress is an aggregate of slow progress and no progress.

4. The progress of work

4.1 Reports of progress

Table 6 presents the progress reported for Projects B and C. For both projects, there is more reporting of poor progress than good or reasonable progress. However, the Project Leader for Project B explained, during an interview, that he wanted reports of poor progress but he wasn't interested in good progress. This may partially explain the tendency towards a bi-modal distribution for Project B. More generally, representatives of functional areas may be inclined towards reporting negative exceptions to planned progress. This indicates a limitation of our data. But it also suggests an area for improvement in projects i.e. if the projects are not going to report progress in terms of actual vs. planned effort, then a more comprehensive subjective summary/reporting would still be useful.

Table 6 also indicates that poor progress accounts for a substantial percentage of all reports of progress. For Project B, poor progress accounts for about 48% of the reports of progress of work. For Project C, poor progress accounts for about 45%.

4.2 Functional areas reporting poor progress

Table 7 indicates which functional areas were reporting poor progress. The table indicates some clear differences between Project B and Project C in where poor progress is occurring. For Project B, most of the poor progress is occurring in the Test functional area.

For Project C, most of the poor progress is occurring in the Design/Code functional area. The differences between Project B and Project C suggest that we should be cautious about assuming that the same problems will affect other projects, even when these projects occur in the same company. Van Genuchten *et al*. [10] also found differences in the 'projects' they studied and were also cautious about how they interpreted their findings.

Table 7 Functional areas reporting poor progress

Functional area	Project B Count	Project B %	Project C Count	Project C %
Design/ Code	4	12.5	25	55.6
Test	20	62.5	14	31.1
Unknown	8	25.0	6	13.3
Total	32	100.0	45	100.0

Table 8 Factors contributing to poor progress, for Project B

Factor	Poor progress	%
Defect/Fix	10	47.6
System reliability problems	6	28.6
Other	5	23.8
Total	21	100.0

Table 9 Types of waiting

Category	Project B Count	Project B %	Project C Count	Project C %
Decision	44	42.7	7	16.7
Defect/Fix	27	26.2	14	33.3
Code	18	17.5	3	7.1
Other	5	4.8	10	23.8
Information	4	3.9	5	11.9
Resource	3	2.9	0	0.0
Unknown	2	1.9	3	7.1
Total	103	99.9	42	99.9

4.3. Reports of the causes of poor progress

As noted in our method section, the data we collected allowed us to make some simple causal connections between some of the data we collected. We analysed the underlying causes of the poor progress reported for Project B. This analysis is shown in Table 8.

It is clear from the Table 8 that the most frequent cause of poor progress is Defects/Fixes. System reliability problems also cause significant delay. More generally, technical problems are most frequently reported as the causes of poor progress.

5. Waiting

5.1. Types of waiting

Table 9 presents results on the types and frequency of waiting for Projects B and C. (See Table 3 for an explanation of the types of waiting.)

With the exception of the Defect/Fix category of waiting, there appears to be little consistency between the two projects. The similar frequencies of waiting on Defects/Fixes, for the two projects, suggests that

Table 10 Dependent functional area

Functional area	Project B Count	Project B %	Project C Count	Project C %
Build	8	7.8	0	0.0
Design / Code	14	14.0	13	31.0
Test	36	35.0	19	45.0
Project management	24	23.0	0	0.0
Other project (within laboratory)	2	1.9	0	0.0
Early market support	3	2.9	0	0.0
Information development	10	9.7	4	9.5
Unknown	6	5.8	6	14.0
Total	103	100.0	42	100.0

Table 11 'Source' functional areas

Functional area	Project B Count	Project B %	Project C Count	Project C %
Build	7	6.8	1	2.4
Design / Code	19	18.4	20	47.6
Test	0	0.0	1	2.4
Marketing	0	0.0	2	4.8
Organisational issues	26	25.2	1	2.4
Other project(s) within the organisation	33	32.0	4	9.5
Project management	1	1.0	0	0.0
External organisation	2	1.9	2	4.8
Unknown	15	14.6	11	26.2
Total	103	100.0	42	100.0

functional areas within Projects B and C often wait on either software or fixes to software defects. This suggests that the defect process and the coding process are either problematic processes in themselves or are impacted by problematic processes. One implication is that these processes could be a focus for improvement.

5.2. Functional areas involved in waiting

The data we have collected allows us to identify which functional areas are waiting. We can also identify the type of work a functional area is waiting on and those functional areas yet to deliver work. The dependent functional area is waiting on the delivery of some type of work from the source functional area.

Table 10 shows those functional areas that are waiting on the delivery of work. The main dependent functional areas in Project B are Test, Project management and Design/Code. The presence of Test and Project Management probably reflects the project's dependency on an external project in another part of IBM, and the importance of Project B to the corporation. See [8] for more information on this issue. For Project C, the situation is somewhat simpler: the Test and Design/Code functional areas are the main dependent functional areas.

Table 11 shows those functional areas that are causing waiting by not completing work on time. Again, Project B is clearly affected by external entities, notably external projects and also senior management outside of IBM Hursley Park. For both projects, the Design/Code functional area is responsible for not completing work.

Table 9 shows the types of work that are at the 'centre' of the waiting. Source functional areas are failing to

complete these types of work, and therefore failing to pass on their outputs to the dependent functional areas. For Project B, most of the decisions being waited on were to be made outside of the project. Similarly, a number of the fixes to defects were being provided by another project.

The differences between Project B and Project C suggest that Project B would have greater difficulty managing the project and improving the project's development processes, because some of the main problematic processes are beyond the control of Project B's management.

6. Overdue work

6.1. Types of overdue work

Table 12 breaks down the types of overdue work for the two projects. For Project B, Defect/Fixes clearly dominate. In addition to Defects/Fixes, Decisions, Problems and Tests are also important types of overdue work. Problems may be yet-to-be identified Defects.

Two key points emerge from the analysis of types of overdue work. First, that for Projects B and C, the most important types of overdue work are those that relate to design/code-oriented issues and test-oriented issues (i.e. the Defects/Fixes, Design/Code, Tests and Problems types). This suggests the prominence of the design/code and test functional areas within these two software development projects. Second, that there may be a strong relationship (dependency) between the Test functional area and the Design/Code functional area. Some of our data, therefore, confirms widely held expectations of the relationship between design, code and test processes.

6.2. Functional areas reporting overdue work

For Project B, the Defect Screen Team was created at the beginning of the project to manage the allocation of defects to developers. The Defect Screen Team changed from weekly meetings to daily meetings in week 38, two weeks after the completion of the design/code phase. Typically, a Defect Screen Team would be formed later in a project at this organisation (perhaps around the time that the design/code phase completes and the test phase commences). It is not clear whether a Defect Screen Team actually existed for Project C. Overdue work by the Defect Screen Team can be related to the overdue Defects/Fixes and overdue Tests, because the Defect Screen Team decide the priority of the defect and allocate that defect to a fixer. While the Defect Screen Team may help to manage defect fixing, it also acts as a potential bottleneck in the defect fixing process.

It is clear from Table 13 that there is little similarity between the two projects in terms of the functional areas reporting overdue work. Note, however, that for both projects the Design/Code and Test functional areas report the most number of different types of overdue work. This is consistent with the Design/Code and Test functional areas reporting poor progress. For Project B, overdue Defect/Fixes seem to be reported by a relatively large number of different functional areas.

Table 12 Types of overdue work

	Project B		Project C	
Type	**Count**	**%**	**Count**	**%**
Design/Code	0	0.0	17	28.8
Decision	9	13.2	0	0.0
Defects/Fixes	37	54.4	11	18.6
Tests	7	10.3	10	17.0
Problem	8	11.8	8	13.6
Publications	2	2.9	6	10.2
Other	4	5.9	5	8.4
Unknown	1	1.5	2	3.4
Total	68	100.0	59	100.0

Table 13 Functional areas reporting overdue work

	Project B		Project C	
Functional area	**Count**	**%**	**Count**	**%**
Design / Code	15	22.1	38	64.4
Test	22	32.4	12	20.3
Build	1	1.5		0.0
Information development	6	8.8	1	1.7
Defect screen team	8	11.8		0.0
External to project	8	11.8	1	1.7
Other	7	10.3		0.0
Unknown	1	1.5	7	11.9
Total	68	100.0	59	100.0

7. Discussion

7.1. The collection and use of quantitative measures of progress

The minutes for both projects present little *quantitative* data. There is one exception, in Project B, where detailed quantitative data on the actual progress of test cases and defects is reported (see [11] for more detail). Regardless of the amount of quantitative data presented, there are few comparisons of actual progress with planned progress. For example, there are few comparisons of actual milestones with planned milestones, and there are no comparisons of any kind of planned work breakdown structure with an actual work breakdown structure.

It may be that comparisons of progress in other work (e.g. design, code) are made but not recorded. However, the first author of this paper attended two project meetings for Project B: at neither meeting were actual comparisons discussed.

Another possibility is that actual versus planned comparisons occurred outside the status meetings. However this would be very surprising because the status meetings for both projects are an explicit mechanism for reporting the progress of each functional area to the rest of the project. Also, the quantitative reports of Project B's progress on test cases suggest that where quantitative data is available it is reported at the status meetings.

In one interview, Project B's Project Leader said:

> "Everyone knows that a work breakdown structure is only valid on the day it was created."

The project leader's statement provides a small insight into why comparisons are not made: the original plan is expected to be out of date anyway. This doesn't explain, however, why data on the actual process is not collected and reported.

The lack of use of quantitative data in the two projects we studied supports our argument on the use of qualitative, subjective measures of progress. While reliable quantitative data on progress is most desirable it is clear that, for some projects at least, such data is not available. In fact, results from the SEI suggest that *most* projects are in this situation. To investigate these kinds of project the most feasible approach may be to exploit qualitative data naturally produced by the projects.

7.2. Problems with reporting progress

The *reporting* of progress, waiting and overdue work does not appear to be only a function of *actual* progress, waiting and overdue work. Specifically, reporting seems to be affected by:

- The presence of major milestones (internal or external) that may cause a Deadline Effect [12].
- The difficulty of properly assessing progress due to the lack of data on planned work and actual work.
- The preference for reporting exceptions at status meetings.
- The preference for only *recording* exceptions in the minutes of the meetings, even if other information is actually reported at status meetings.

These issues could be treated as limitations of this study. But these limitations are inherent within the naturally-produced data. Therefore, a much more important conclusion may be that these issues are severely limiting the effective management of the *projects*.

8. Conclusion

There are some similarities between the two projects in terms of the types of work that are causing problems, and in the functional areas that are experiencing the most difficulties. Some of our analysis, therefore, confirms widely held expectations of the relationship between design, code and test processes. There are also differences between Project B and Project C, and these suggest that we should be cautious about assuming that the same problems will affect other projects, even when these projects occur within the same company.

Our analysis indicates that, for these two projects, there was very little reporting of quantitative data in the projects and, related to this, there was little explicit comparison of actual progress with planned progress. In addition, the reporting of progress does not just seem to be a function of progress itself. Other factors (such as the Deadline Effect and some 'preferences' for reporting and recording certain types of information) seem to affect the accurate reporting of progress. While these other factors may threaten the validity of empirical studies, a much more serious threat is to the valid reporting and management of the projects themselves.

In terms of further research, we have collected this data on a week-by-week basis and we intend to consider the temporal aspects of this data. For example, we are interested to know whether the frequency of reports are affected by other project events e.g. an approaching deadline. We also plan to investigate the effects of Project B's dependencies on projects and 'units' that are external to the project.

Overall, while the use of qualitative, subjective data may be less desirable than quantitative data, we hope that work in this area can contribute to improved project planning and control, and may (in the longer term) encourage practitioners to move toward more developed, quantitative measures of progress.

Acknowledgements

Whilst the majority of this research was conducted whilst at Bournemouth University, the actual writing of this paper occurred during a lectureship at the University of Hertfordshire. Austen Rainer is very grateful to Professor Martin Shepperd for his support throughout the duration of his doctoral research, of which this paper is a product. Similarly, he is grateful to Paul Gibson and John Allan, at IBM Hursley Park, for their support during the doctoral research, and to the many people at IBM Hursley Park for allowing their projects to be studied (or at least candidates for study). Dr. Sarah Beecham provided useful comments on a draft of this paper. We also thank the reviewers for their helpful comments.

References

[1] SEI, "Process Maturity Profile: Software CMM - 2003 Year End Update," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. March 2004 2004.

[2] J. E. Cook, L. G. Votta, and A. L. Wolf, "Cost-Effective Analysis Of In-Place Software Processes," *IEEE Transactions on Software Engineering*, vol. 24, pp. 650-663, 1998.

[3] M. G. Bradac, D. E. Perry, and L. G. Votta, "Prototyping A Process Monitoring Experiment," *IEEE Transactions on Software Engineering*, vol. 20, pp. 774-784, 1994.

[4] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, organizations, and process improvement," *IEEE Software*, vol. 11, pp. 36-45, 1994.

[5] D. E. Perry, N. A. Staudenmayer, and J. G. Votta Jr., "Understanding and improving time usage in software development," in *Trends in software: software process*, A. Fuggetta and A. L. Wolf, Eds.: John Wiley and Sons Ltd, 1995.

[6] A. W. Rainer, "A secondary analyses of Bradac et al.'s prototype process-monitoring experiment," University of Hertfordshire, Technical Report CS-TR-383, April 2003 2003.

[7] A. A. Porter, H. P. Siy, A. Mockus, and J. G. Votta Jr., "Understanding the sources of variation in software inspections," *ACM Transactions on Software Engineering and Methodology*, vol. 7, pp. 41-79, 1998.

[8] A. W. Rainer, "An Empirical Investigation of Software Schedule Behaviour," in *Department of Computing*. Bournemouth UK: Bournemouth University, 1999.

[9] L. M. Taff, J. W. Borchering, and W. R. Hudgins Jr., "Estimeetings: Development estimates and a front-end process for a large project," *IEEE Transactions on Software Engineering*, vol. 17, pp. 839-849, 1991.

[10] M. van Genuchten, "Why is software late? An empirical study of reasons for delay in software development," *IEEE Transactions on Software Engineering*, vol. 17, pp. 582-590, 1991.
[11] A. W. Rainer, "A catalogue of 'analytic fragments' of the behaviour of a software project," Empirical Software Engineering Research Group (ESERG) Bournemouth University, Bournemouth, Technical Report ESERG-TR00-007, 2000.
[12] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, N.J.: Prentice-Hall, 1981.