A secondary analyses of Bradac *et al*.'s prototype process-monitoring experiment

Austen Rainer
Department of Computer Science
University of Hertfordshire
Hatfield Campus
College Lane
Hatfield
AL10 9AB
U.K.

a.w.rainer@herts.ac.uk
+44 (0) 1707 284763

April 2003

Computer Science Technical Report CS-TR-383

We report on the secondary analyses of some conjectures and empirical evidence presented in Bradac *et al*.'s prototype process-monitoring experiment, published previously in *IEEE Transactions on Software Engineering*. We identify 13 conjectures in the original paper, and re-analyse six of these conjectures using the original evidence. Rather than rejecting any of the original conjectures, we identify assumptions underlying those conjectures, identify alternative interpretations of the conjectures, and also propose a number of new conjectures. Bradac *et al*.'s study focused on reducing the project schedule interval. Some of our re-analysis has considered improving software quality. We note that our analyses were only possible because of the quality and quantity of evidence presented in the original paper. Reflecting on our analyses leads us to speculate about the value of 'descriptive papers' that seek to present 'empirical material' (together with an explicit statement of goals, assumptions and constraints) separate from the analyses that proceeds from that material. Such descriptive papers could improve the 'public scrutiny' of software engineering research and may respond, in part, to some researchers criticisms concerning the small amount of software engineering research that is actually evaluated. We also consider opportunities for further research, in particular opportunities for relating individual actions to project outcomes.

~ Criticism of our conjectures is of decisive importance: by bringing out our mistakes it makes us understand the difficulties of the problem which we are trying to solve.
Sir Karl Popper, *Conjectures and Refutations*, p. xi

Introduction

An important activity when conducting empirical research is to review work that has previously been conducted in the same and related areas. Certain studies may warrant a great deal of attention and a very detailed examination. Such an examination may become an analysis in itself. In the case of older papers, a detailed examination may 'breath new life' into a paper, perhaps through revealing alternative interpretations of the same empirical evidence, through revealing new conjectures and insights, or through relating 'old data' to more recent theoretical issues.

A detailed examination of a previously published paper can also fulfil another important quality of the research process: evaluating the results of the study. Glass [1] has previously argued that almost no computing research has an evaluative phase. Tichy *et al*. [2] provide quantitative evidence that complements Glass' arguments. Fenton *et al*. [3] reflect on the

quality of software engineering research, reviewing both good and poor examples, and offering advice on evaluating research.

This report presents a very detailed, secondary analyses of the evidence presented in Bradac *et al*. [4]. Although Bradac *et al*.'s paper was published almost a decade ago, the analyses we present here are a significant contribution to our understanding of actual software development. This is because:

- The study was part of a long-term, coherent 'programme' of research at Lucent Technologies that directed attention at developing a better understanding of how software is actually developed. There appear to be few such long-term 'programmes' although NASA's Software Engineering Laboratory (e.g. [5-8]) and the MCC consortium (e.g. [9-12] are two notable complements to the work of Lucent Technologies.

- The study is problem-oriented, focusing on a 'real-world' problem recognised in industry i.e. reducing development time.

- The study is based on data that is produced within an industrial context (*cf*. Potts' [13] concept of industry-as-laboratory) and so has 'ecological validity'.

- The study provides a useful complement to other studies that have investigated the planned and actual duration of formally-defined activities in projects. There may be the opportunity to investigate how studies of actual time usage can integrate with studies of formally-defined project activities.

- Bradac *et al*.'s paper provides a sufficient amount of appropriate evidence to allow secondary analyses. This is unusual for journal papers but is, in itself, an indication of the quality of Bradac *et al*.'s study.

An earlier version of Bradac *et al*.'s paper appeared as a conference paper [14] and the findings of the paper are also discussed in [15] and [16]. The paper has also informed other research (e.g. [17] and [18]) within Lucent Technologies.

The secondary analyses that we present in this paper are very detailed, and the paper is lengthy as a result. This detail reflects our general approach i.e. to 'decompose' the original analyses into constituent parts, to re-analyse those parts, and then to conduct additional analyses using the original empirical evidence. Our analyses lead to the identification of assumptions underlying some of the original conjectures, the identification of alternative interpretations of the original conjectures, and the identification of a number of new conjectures. We emphasise that our analyses are only possible because of the quality and quantity of evidence presented in the original paper.

The remainder of this report is organised as follows. Section 1 presents a detailed review of the empirical evidence and analyses presented in Bradac *et al*.'s paper, and a summary of the conjectures forwarded by Bradac *et al*. The detailed review is necessary as it forms the foundation for all of the subsequent secondary analyses. Section 2 then reports our secondary analyses of several of Bradac *et al*.'s conjectures, *using their empirical evidence*. Section 3 then presents additional analyses of Bradac *et al*.'s empirical evidence, resulting in several new conjectures. Section 4 discusses the results of the secondary analyses, relates these results to other work, and considers possible future work. Section 5 provides brief conclusions. An appendix provides a list of statements from Bradac *et al*.'s paper that can be used to complement the review provided in section 1.

1 A review of Bradac *et al*.'s arguments

1.1 An overview of Bradac et al.'s study

Bradac *et al*. report on a pilot study (they call it a 'prototype process monitoring experiment') that investigates how a developer spent their time during development. Bradac *et al*. state: "The specific goal of our process monitoring experiment is to find ways to reduce the development interval." ([4], p. 774; see the appendix for further information).

Bradac *et al*.'s study was a pilot study in preparation for a more substantial study, which has been reported elsewhere [15, 16]. The presence of a subsequent, more substantive study raises the question of why the pilot study, rather than the subsequent study, is being analysed. The short answer is data: more data, and more appropriate data, is presented in the pilot study. In addition, however, the pilot study lays foundations for the subsequent study. Consequently, it is sensible to examine those foundations.

The pilot study was based on a reconstruction of a 30-month time diary for the lead engineer of a feature composed of both hardware and software. Bradac *et al*. explain that features are often the most basic unit of development for a very large software system and represent long-term efforts spanning up to several years from inception to actual use. In their pilot study, Bradac *et al*. broadly distinguish between working the process and being prevented from working the process. Where a developer is prevented from working the process, this is because the process is blocked, primarily due to some form of waiting. As is to be expected of a pilot study, Bradac *et al*. revise their opinion on certain aspects of their study design and empirical findings. For example, they note that where a developer is not working the process, this may either be because their work is blocked or because the developer has been temporarily reassigned to another project.

1.2 Empirical evidence and analysis

Bradac *et al*. present three main sets of evidence and use these sets to develop some conjectures about the way that individual developers use their time when developing features. The three sets of evidence are:

- A summary of the amount of time spent working and being blocked over the 30-month time-period.
- A summary of the percentage of time spent waiting in each of the tasks.
- Two 75-day 'time-lines' providing more detail on the actual allocation of time to tasks.

1.2.1 An overall analysis of time spent working and blocked

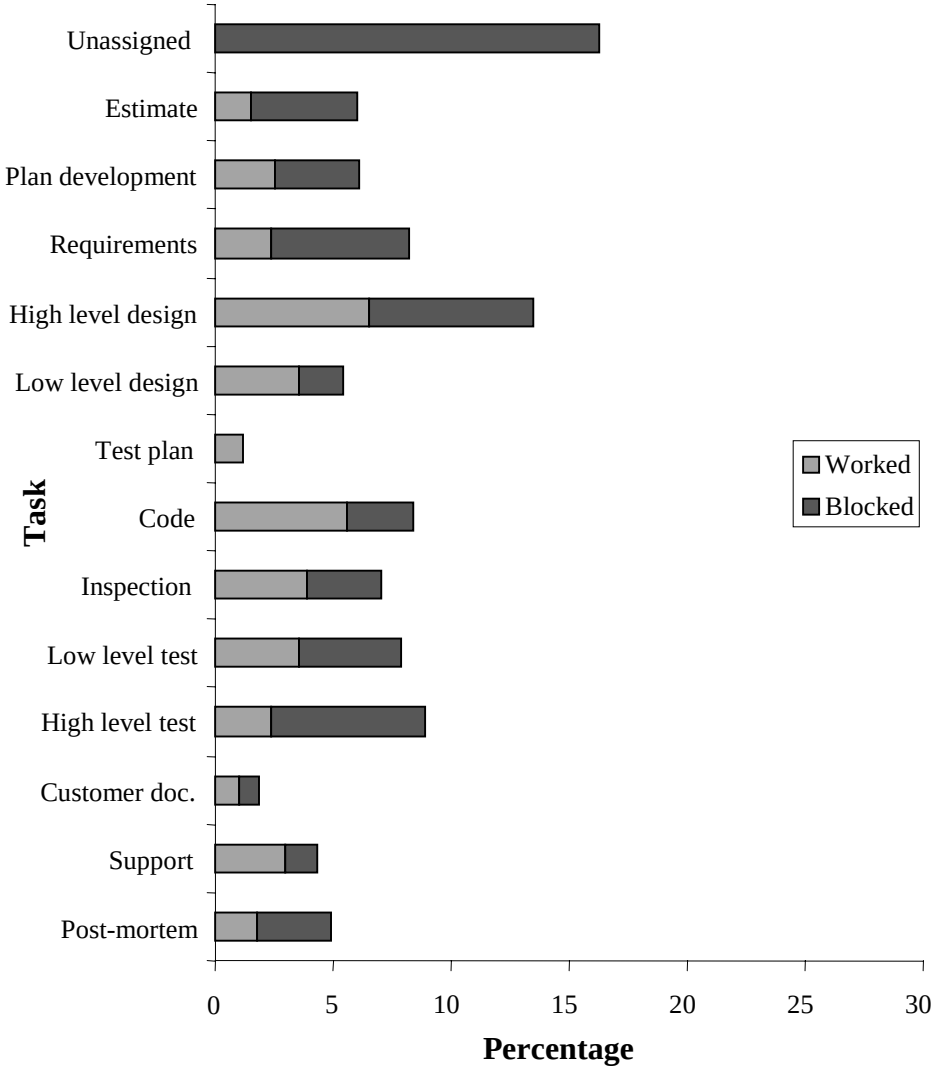

Figure 1 Percentage time spent working and blocked

(Reproduced from Bradac *et al.* [4])

Bradac *et al*. present data on the percentage of time spent working and blocked in 14 tasks. This data is reproduced here in Figure 1. (Bradac *et al*. are not explicit about the source of this data. We assume that this data is based on the complete 30-month time period.) Bradac *et al*. note that for almost every task, the time spent waiting exceeds that of productive work. Overall, 60% of the time was spent waiting rather than working. This leads Bradac *et al*. to state that:

> "… it seems clear that one important way of improving the process is to reduce significantly the number of days in blocking states." ([4], p. 782)

Bradac *et al*. add:

> "The utility of this conjecture depends on the degree of multiplexing within these processes. Clearly, if the global level of blocking is consonant with this local level, the conjecture will hold." ([4], p. 782)

1.2.2 The prevalence of blocking in the process

Bradac *et al*. also examine the percentage of waiting for each task according to the total number of days spent in that task (with total days including both productive and blocked days). Figure 2 is a reproduction of the data used by Bradac *et al*. Bradac *et al*. emphasise that the breakdown into small, medium and large numbers of days helped them to distinguish the more important tasks in the process. The figure indicates, for example, that work on *Customer documentation* was blocked for almost 50% of the time but, overall, *Customer documentation* only required a small number of days.

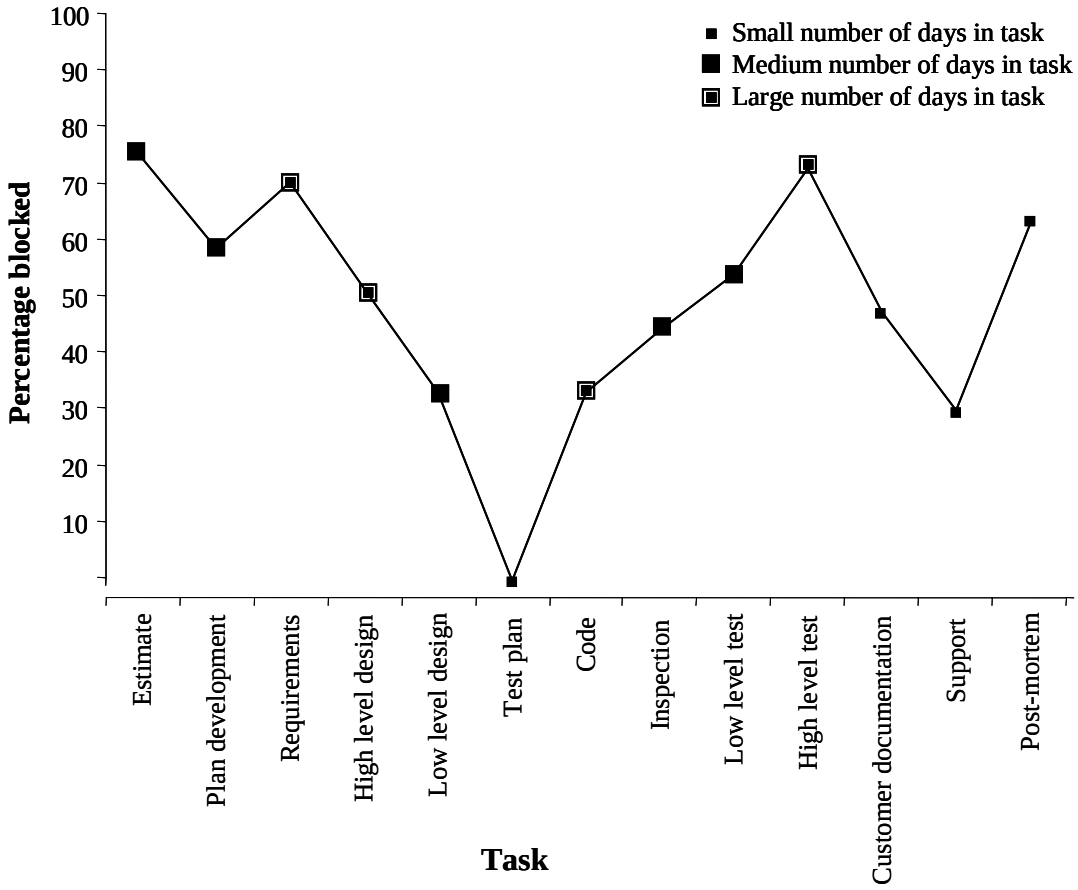

Figure 2 Percentage of time blocked per task

(Reproduced from Bradac *et al.* [4])

Based on the data presented in Figure 2, Bradac *et al*. state:

> "The other thing to note is that blocking tends to be more prevalent at the beginning and end of the process." ([4], p. 783).

With this insight, Bradac *et al*. suggest that one should attack blocking factors in the requirements, high-level design and high-level test phases of the process because these have some of the highest percentages of blocked work, together with the highest number of days spent in the tasks.

Bradac *et al*. appear to assume that the tasks occur in a broadly sequential manner i.e. there are certain tasks (such as requirements and high-level design) that occur during the earlier periods of the process, certain tasks (such as low-level design and coding) that occur during the middle periods of the process and certain tasks (such as high-level testing) that occur during the final periods of the project. (The use of a line to connect the points in Figure 2 suggests continuity between tasks and, consequently, implies an assumption that the tasks are broadly sequential.)

We also note that Bradac *et al*. do not explicitly state the criteria for distinguishing between a small, medium and large number of days, and do not provide a rationale for this distinction.

1.2.3 Bradac *et al*.'s two 'time-lines'

Figure 3 and Figure 4 reproduce the two 'time-lines' presented in Bradac *et al*.'s paper. Figure 3 presents data on the tasks and states of a developer during an earlier part of a project. By contrast, Figure 4 presents data on the tasks and states of a developer during a later part of a project. Bradac *et al*. are not specific about when exactly, during the 30-month period, these two time-lines occur. The difficulty in mapping the time-lines to the 30 months is relevant to our secondary analyses and will be returned to later in this report.

Bradac *et al*. note that the first part of the project (see Figure 3) appears to conform to the traditional waterfall process, with the developer moving sequentially from the plan development to the requirements. Bradac *et al*. also note the lengthy blocking times during this earlier part of the project, in particular the lengthy times spent waiting on reviews. These observations appear to confirm the assumption of a sequential process and also appear to confirm the conjecture that blocking is more prevalent toward the beginning of the process.

As indicated in Figure 4, a later part of the project appears to be dramatically different to the beginning of the project. The developer is alternating between a series of tasks, with tasks and states being inter-mixed. Bradac *et al*. recognise the less sequential nature of their developer's behaviour and relate that behaviour to Guindon's 'opportunistic developers' [10, 11].

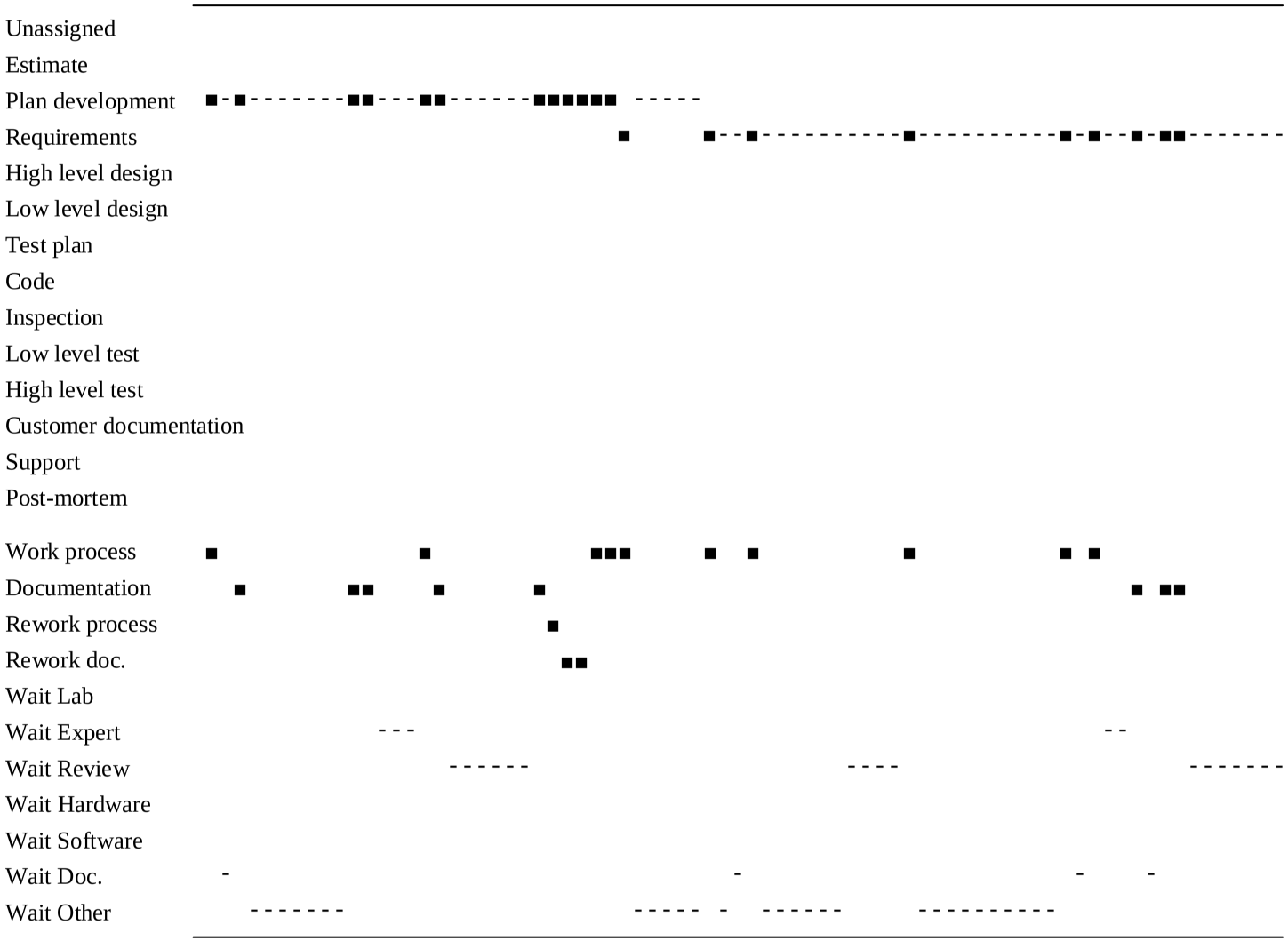

Figure 3 Tasks and states for a developer during an earlier 75-day period of a project

(Reproduced from Bradac *et al.* [4])
Key: The dashes ('-') indicate waiting. The solid squares ('■') indicate working.

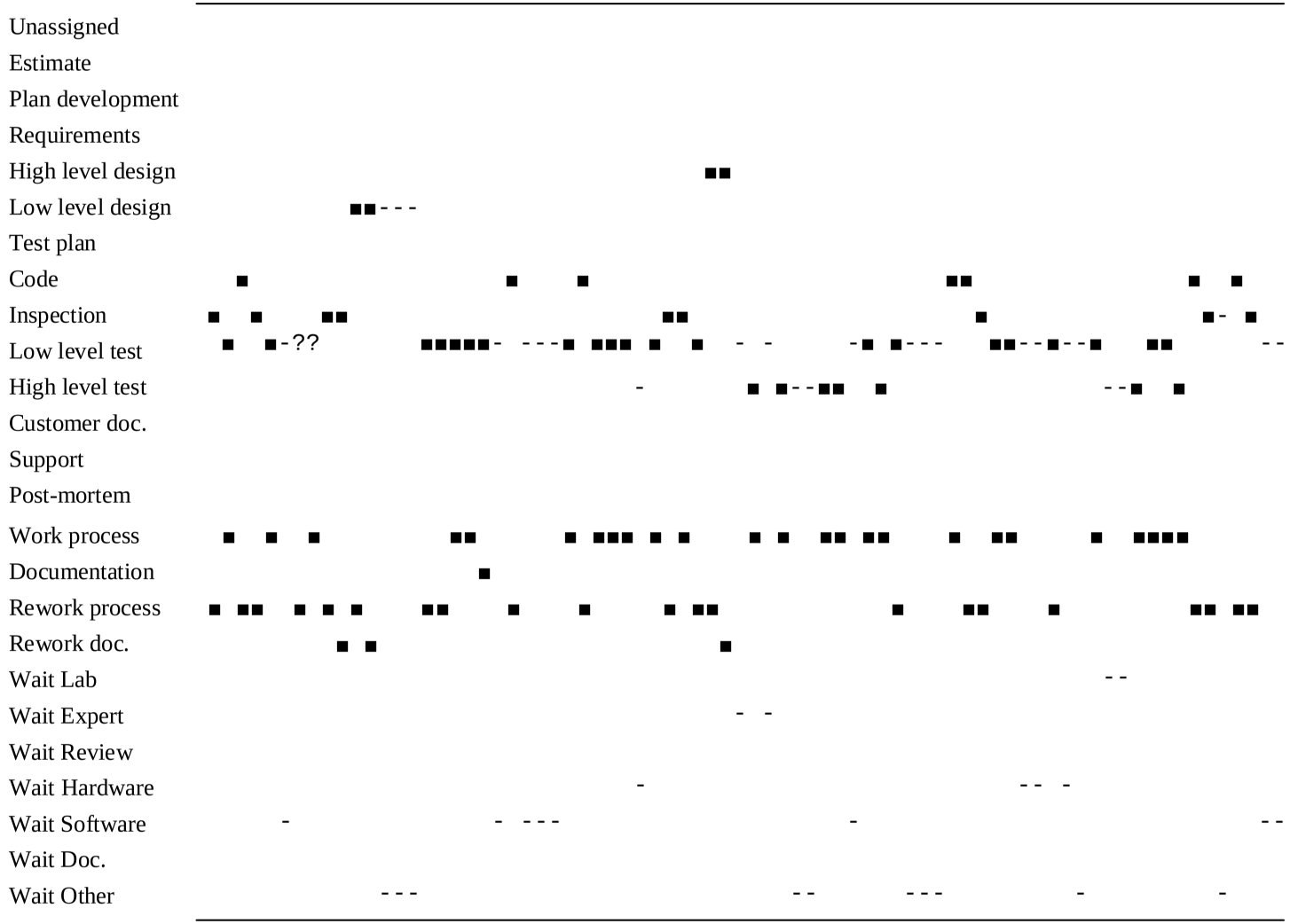

Figure 4 Tasks and states for a developer during a later 75-day period of a project

(Reproduced from Bradac *et al*. [4])

A question mark (i.e. '?') indicates a day of working, but the specific task is unknown. The dashes ('-') indicate waiting. The solid squares ('■') indicate working.

1.3 Conjectures

Overall, Bradac *et al*. propose several conjectures regarding the 'behaviour' of developers as they develop features. We present these conjectures in Table 1, and indicate which of these conjectures we have re-analysed. (Bradac *et al*. also make a number of statements regarding the theoretical, empirical and methodological aspects of their study. We have summarised these statements in the appendix.) We re-analysed those conjectures where the data made it feasible to do so. In total, we re-analysed six of the 13 conjectures.

Table 1 Bradac *et al.*'s conjectures

#	Conjecture	Re-analysed?
1	The specific goal of our process monitoring experiment is to find ways to reduce the development interval.	No
2	A process is characterised by a set of tasks and a set of states. The tasks define the various activities within the process that are of interest and that will be sampled in the experiment. The states represent either progress within a task or lack of progress (that is, where the task is blocked for some reason).	No
3	We assume that developers really do only one or two things per day per process – that is, their time is not overly fragmented.	No
4	The intent is to sample the most important aspect of the task on the previous day.	No
5	Overall, approximately 60% of the time is spent waiting or being reassigned to other projects [with 40% of the time spent working].	Yes
6	It seems clear that one important way to improving the process [to reducing development duration cf. conjecture 1] is to reduce significantly the number of days in blocking states.	Yes
7	Blocking tends to be more prevalent at the beginning and the end of the process [compared to the overall process].	Yes
8	Waiting for reviews dominated both the beginning and the end of the process.	No
9	The utility of conjecture 7 depends on the degree of multiplexing within the process. If the global level of blocking is consonant with the local level, then the conjecture should hold.	No
10	If blocking is more prevalent during the beginning and the end of the project, and this occurs at the project level, then one should attack blocking factors in the requirements, high-level design and high-level test phases of the process.	Yes
11	The first part of the process is almost a pure waterfall process moving first through the plan development task and then to the requirements.	Yes
12	The later part of the process is much more inter-mixed, consistent with Guindon's arguments that design is opportunistic.	Yes
13	A caveat: although this analysis is based on real data, they are reconstructed from one instance of the process, with some blurring of the accuracy because of retrospection.	No

2 Secondary analysis of Bradac *et al*.'s conjectures

This section reports the results of our secondary analysis of Bradac *et al*.'s conjectures.

2.1 Conjecture 5: Approximately 60% of the time is spent waiting or being reassigned

Table 2 Comparison of the earlier and later parts of the project with the overall process

State	Earlier in the project		Later in the project		Overall process	
	f	%	*f*	%	*f*	%
Working	21	27.6	50	65.8	30	39
Blocked	55	72.4	26	34.2	46	61
Total	76	100	76	100	76	100

Table 2 presents the frequency counts and percentages for the earlier and later parts of the project, and the overall process. (The data on the earlier and later parts of the process are taken from Figure 3 and Figure 4. The data on the overall process are taken from Figure 1[1].) The table suggests that there is a great deal of variation in the amount of time that can be spent blocked or working. Between a quarter and two-thirds of the developer's time is spent working, and between a third and three-quarters of the developer's time is spent waiting.

[1] In their paper, Bradac *et al*. did not provide frequency counts for the overall process, instead providing only percentages. To 'recover' the frequency counts, we have converted the percentages of the overall process to frequency counts based on the total number of days of the earlier and later parts of the project i.e. based on a total of 76 days. More specifically, as 39% of the overall process was spent working, this can be converted to approximately 30 days i.e. 39% of 76 days. Similarly, 61% of the overall process was spent waiting and this converts to approximately 46 days i.e. 61% of 76 days.

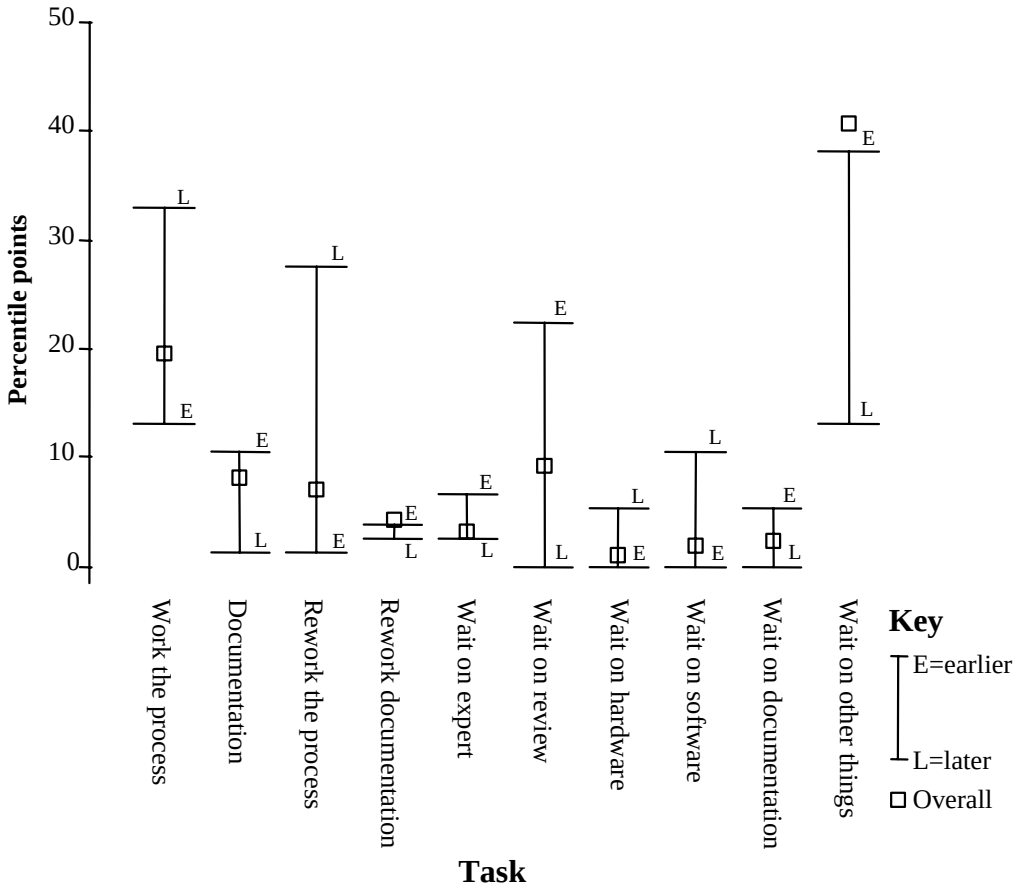

Figure 5 Variation in percentile points of blocked work

Figure 5 summarises the variation, in percentile points, between the earlier, later and overall process for all the states examined by Bradac *et al*. In almost all cases, the earlier and later parts of the process represent the extreme points[2].

Both Table 2 and Figure 5 suggest that one should be cautious about the 60:40 ratio of waiting to working suggested by Bradac *et al*. because this ratio does not 'capture' the variation that is present in the process being examined[3]. Figure 5 suggests that one should be particularly cautious about the ratio of waiting to working for the following states (because of the degree of variation observed for these states):

- Working the process
- Reworking the process
- Waiting on reviews
- Waiting on other things

Bradac *et al*. recognised that waiting on other things was actually a special category that included being reassigned to other projects. This suggests that a reasonable amount of reassignment occurs; a suggestion consistent with other studies e.g. [20-23].

[2] For only two exceptions – reworking documentation and waiting on other things – the earlier and later parts of the process do not represent the extremes. This means that during some other part(s) of the 30-month process, the amount of reworking documentation and waiting on other things was greater than that represented in the two time-lines.

[3] The presence of variation, and strategies to monitor and then reduce that variation, are the concerns of statistical process control e.g. 19. Florac, W.A., A.D. Carleton, and J.R. Barnard, *Statistical Process Control: Analyzing A Space Shuttle Onboard Software Process.* IEEE Software, 2000. **17**(4): p. 97-106..

2.2 Conjecture 7: The prevalence of waiting

An 'eyeball test' of Table 2 suggests that of the earlier and later parts of the process, it is the earlier part of the process that is more like the overall process. This 'eyeball test' appears to undermine Bradac *et al.*'s conjecture that waiting is more prevalent during the beginning and the end of the process. As a result of the 'eyeball test', we forwarded two hypotheses:

H_1 The frequency of waiting in the earlier part of the process is significantly different to the frequency of waiting in the overall process

H_2 The frequency of waiting in the later part of the process is significantly different to the frequency of waiting in the overall process

We tested these hypotheses using chi-square ($\chi 2$) tests of independence.

Table 3 Results of chi-square tests of hypotheses H_1 and H_2

		statistics				
#	**Hypotheses**	**p**	***df***	**$\chi 2$**	**α**	**β**
H_1	The frequency of waiting in the earlier part of the process is significantly different to the frequency of waiting in the overall process	0.122	1	2.39	0.05	unknown
H_2	The frequency of waiting in the later part of the process is significantly different to the frequency of waiting in the overall process	0.001	1	10.556	0.05	unknown

The results of the chi-square tests of the two hypotheses are presented in Table 3. The results indicate that:

1. The earlier part of the process is *not* significantly different to the overall process.
2. The later part of the process *is* significantly different to the overall process.

These results suggest two things. First, the results suggest a re-interpretation of Bradac *et al.*'s seventh conjecture *viz*. that waiting is more prevalent in the earlier *and later* parts of the project. The test results suggest that blocking is indeed more prevalent in an earlier part of the process, but is not so prevalent in a later part of the process. Second, the test results are further support for our re-analyses of conjecture 5, and provide stronger evidence that one should be cautious about the 60:40 ratio of waiting to working. The test results suggest that there is a significant difference in the amount of waiting between the overall process and a later part of the process. Therefore, there will be different ratios of waiting to working for these two parts of the process.

The two time-lines presented in Figure 3 and Figure 4 represent extremes of the process but it is not clear where exactly in the 30-month time period these two time-lines occur. It may be that either of the time-lines occur closer to the middle of the process and so may not be appropriate for testing the validity of Bradac *et al.*'s seventh conjecture.

A related issue is that Bradac *et al*. may make their claim about the prevalence of waiting based on more data than is presented in their paper. Being extremes, these two time-lines may not be entirely representative of other data that Bradac *et al*. possessed at the time of writing their paper.

2.3 Methods for investigating the prevalence of waiting

The results of our two hypotheses lead us to re-consider the methods that Bradac *et al*. used to identify the most important tasks in the process. Recall from section 1.2.2 that Bradac *et al*. distinguished between a small, medium and large number of days spent waiting and working the process. Recall also that Bradac *et al*. subsequently used these distinctions to identify those tasks that had both a large percentage of waiting and a large total number of days in the task.

Table 4 Thresholds for small, medium and large number of days

Threshold	Percentages of days in task
Large	$n \geq 8$ percentile points
Medium	$8 > n > 5$ percentile points
Small	$0 \leq n \leq 5$ percentile points

Note: *n* is the total percentage of days in the task

In their paper, Bradac *et al.* do not explicitly state the 'thresholds' that they used to distinguish between a small, medium and large number of days. Our estimates of Bradac *et al*.'s thresholds for distinguishing between a small, medium or large number of days waiting and working a task are presented in Table 4.

Table 5 Alternative thresholds

Threshold	Bradac *et al*	1st Alternative	2nd Alternative
(Very large)	N/A	N/A	$13.5 \leq n$
Large	$8 \leq n$	$13.5 \leq N$	$6.4 < N < 13.5$
Medium	$5 < n < 8$	$5.3 < n < 13.5$	$3.8 < n \leq 6.4$
Small	$0 \leq n \leq 5$	$0 \leq n \leq 5.3$	$0 \leq n \leq 3.8$

We considered two alternative thresholds to Bradac *et al*.'s. For the first threshold, we calculated the range of percentile points (i.e. 13.5 – 1.2 = 12.3, *cf.* Table 6) and divided the range into three equal sections (i.e. sections each with a range of 4.1) and then calculated thresholds. These thresholds are presented in Table 5 as the 1st alternative threshold. With this threshold, we recognised that the *high-level design* task distorts the calculation of the range and this then has an effect on the calculated thresholds (leaving only the *high-level design* task in the category of a large number of days).

For our second alternative, we treated the *high-level design* task as a special case. We removed the *high-level design* task from our initial data, and re-calculated our range (i.e. 8.9 – 1.2 = 7.7), our three equal sections (i.e. each section now had an approximate range of 2.6) and then the thresholds. We treated the *high-level design* task as exhibiting a very large number of days in the task. These thresholds are presented in Table 5 as the 2nd alternative threshold. The re-classification of tasks with small, medium, large and very large numbers of days is presented in Figure 6.

Depending on the thresholds chosen, different tasks emerge as the most important tasks in the process. This leads to different advice about which tasks to concentrate on in order to reduce waiting and blocked work. For example, the second alternative threshold in Table 5 and Figure 6 suggests that one should focus on all the major tasks of software development i.e. requirements, high-level design, low-level design, code, inspection, low-level test and high-level test. The problem with all three of the sets of thresholds is that they are unable to effectively discriminate between a larger number of relevant tasks and a small number of really important tasks.

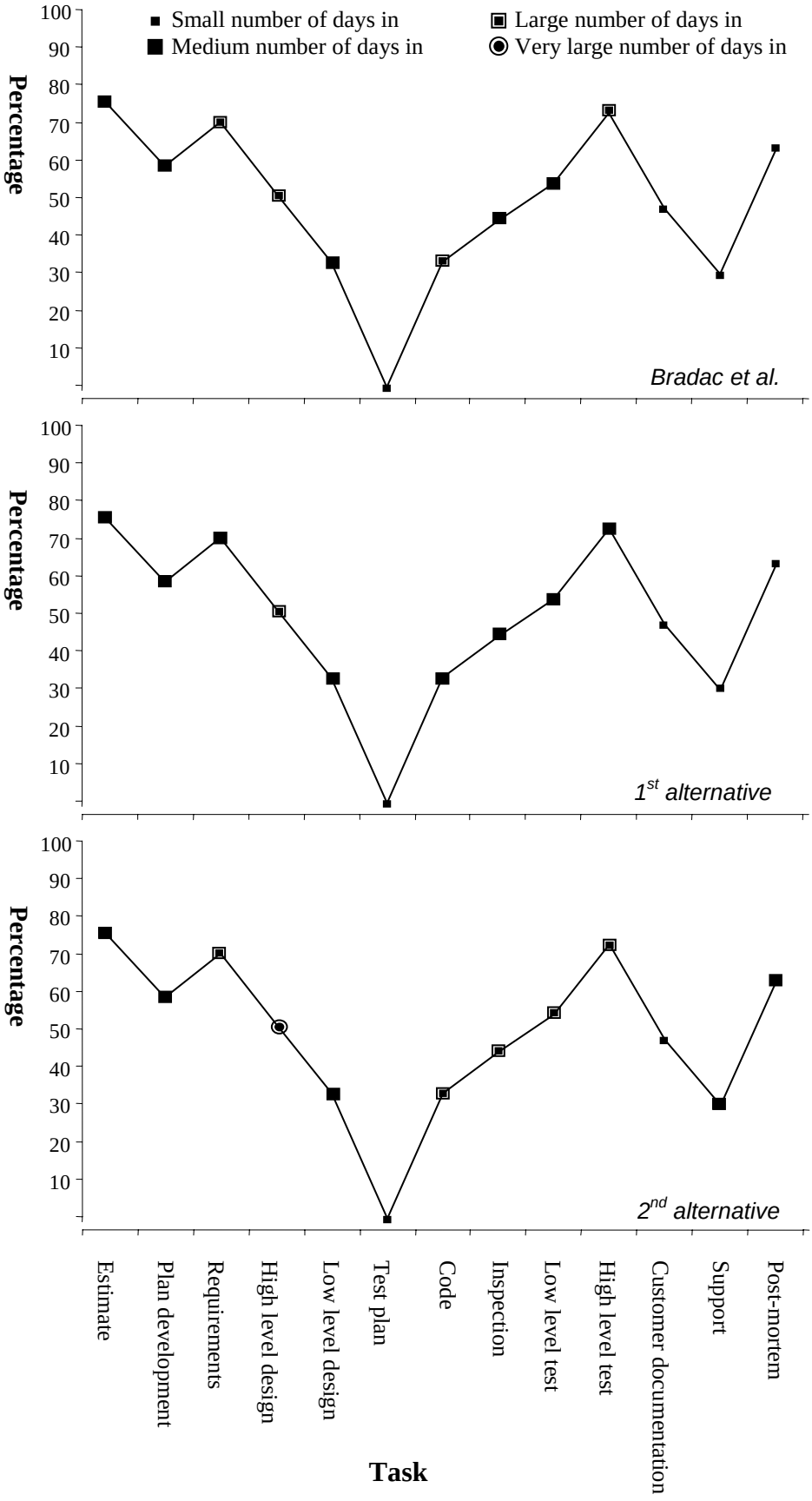

Figure 6 Breakdown of important tasks

Because of the problems of poor discrimination with the thresholds, we investigated another method of identifying the most important tasks.

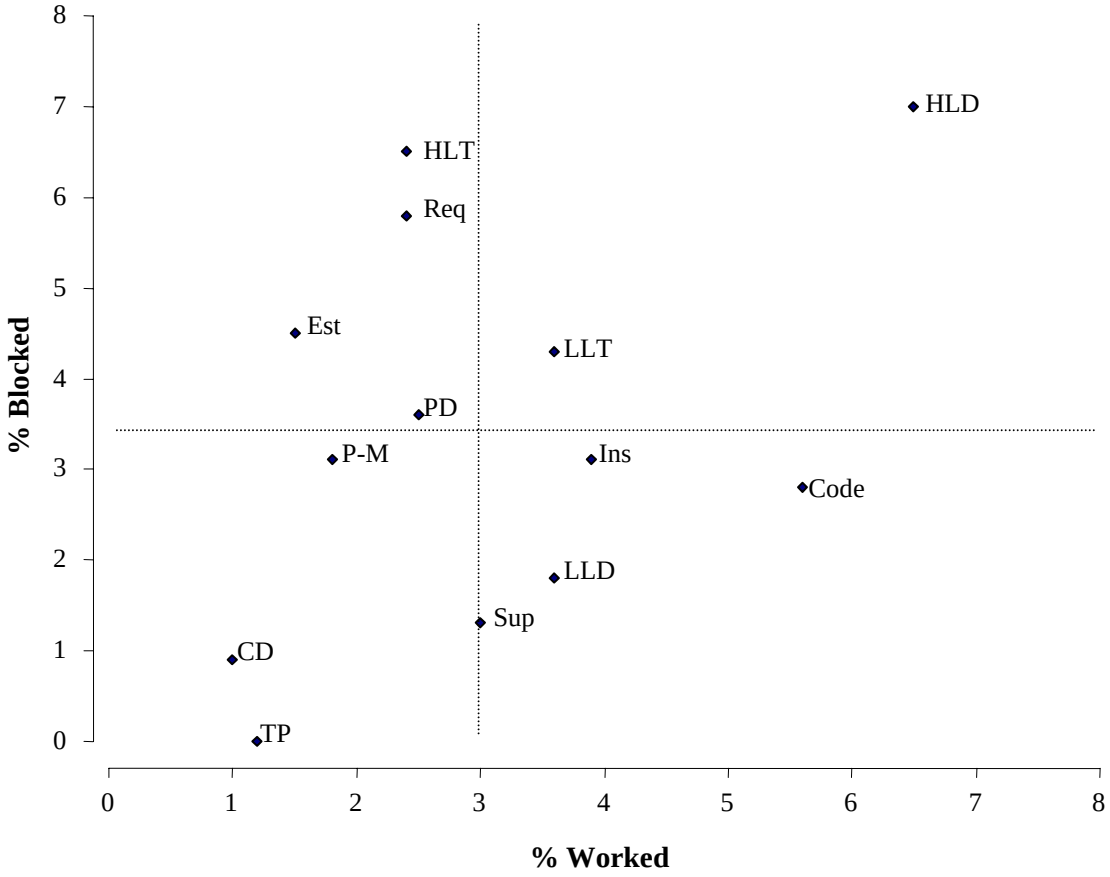

Figure 7 Scatter-plot of percentile points of blocked work (waiting) and working the process

Key: CD=Customer documentation; Code=coding; Est=Estimation; HLD=High level design; HLT=High level test; Ins=Inspections; PD=Plan development; P-M=Post-Mortem; LLD=Low level design; LLT=Low-level test; Req=Requirements; Sup=Support; TP=Test planning.

Figure 7 provides an alternative representation of the data presented in Figure 2, and plots the percentile points for work the process against waiting. This analysis suggests that the most important tasks are high-level design and low-level test. We discuss this analysis in more detail in section 2.5.

2.4 Conjecture 8: The dominance of waiting on reviews

Figure 5 indicates that reviews are not dominant during the end of the process. Recall that the figure is based on the two 75-day time-lines and, as noted earlier, there may be problems concerning the representativeness of the two time-lines.

2.5 Conjecture 10: Attacking blocking factors

Bradac *et al*. used data (presented in this report as Figure 2) to suggest that reducing blocking in the earlier and later parts of the process could be an effective method for reducing the project schedule. This is because these tasks are the most heavily 'weighted' in terms of blocking. An alternative analysis, one that focuses on the amount of time spent *working*, leads to somewhat different conclusions.

Table 6 Percentages and rankings of time spent working or blocked in each task

(Based on Figure 3 of Bradac *et al.* [4])

	Percentages			Rankings		
Task	**Working**	**Blocked**	**Total**	**Working**	**Blocked**	**Total**
Post-mortem	1.8	3.1	4.9	10	8	11
Support	3	1.3	4.3	6	12	12
Customer documentation	1	0.9	1.9	13	13	13
High level test	2.4	6.5	8.9	8	3	3
Low level test	3.6	4.3	7.9	4	6	6
Inspection	3.9	3.1	7.0	3	8	7
Code	5.6	2.8	8.4	2	10	4
Test plan	1.2	0	1.2	12	14	14
Low level design	3.6	1.8	5.4	4	11	10
High level design	6.5	7	13.5	1	2	2
Requirements	2.4	5.8	8.2	8	4	5
Plan development	2.5	3.6	6.1	7	7	8
Estimate	1.5	4.5	6.0	11	5	9
Unassigned	0	16.3	16.3	14	1	1
Total	39	61	100			

Table 6 presents the percentages, and the rankings of those percentages, of the time spent working or blocked in each task for the overall 30-month process. If one ranks the tasks based on the percentage of waiting then one finds that, indeed, the tasks of requirements, high-level design, and high-level test are the most important tasks (excluding the unassigned task). If one ranks the tasks based on the percentage of time spent *working* then one finds that high-level design, low-level design, coding, inspections and low-level tests become the most important tasks. This represents a shift from the earlier part of the process to, predominantly, the middle of the process. Phrased another way, most of the actual work appears to occur during the middle of the process[4].

With the combined perspectives of waiting and working, we can extend Bradac *et al.*'s initial advice that one should attack the earlier and later parts of the process. An initial step for reducing the development interval may be to reduce the amount of time spent waiting. As that time is reduced, however, further efforts to reduce the development interval will need to be directed elsewhere i.e. to where the actual work is being done.

Ranking tasks according to the percentage of actual work also suggests strategies for resource allocation. The data suggests that the most effort is used during the middle parts of the project. (This is consistent with findings we have made in other work e.g. [24].) The data also suggests that attempts to improve the quality of the product should be directed at the middle of the process, as this is where the most work is actually done and hence where there are the greatest opportunities to introduce defects into the product.

[4] Based on the assumption of a sequential process.

Table 7 Tasks to focus on to reduce time and improve quality

Task	Reduce time (cost)	Improve quality	Reduce time and improve quality
Customer documentation	-	-	
Code	-	Yes	
Estimation	Yes	-	
High level design	Yes	Yes	Yes
High level test	Yes	-	
Inspection	-	Yes	
Plan development	Yes	-	
Post-Mortem	-	-	
Low level design	-	Yes	
Low-level test	Yes	Yes	Yes
Requirements	Yes	-	
Support	-	Yes	
Test planning.	-	-	

Table 1 provides an alternative summary of the evidence presented in Figure 7, and indicates which tasks to focus on when trying to reduce schedule time or improve quality. This analysis appears to possess more discriminatory power, in that it clearly identifies a small number of apparently very important tasks i.e. high-level design and low-level test. (These two tasks seem an odd combination, and some further research is required here.)

2.6 Conjectures 6, 11 and 12: The nature of the process

An 'eyeball test' of Figure 5 suggests that the overall process is more similar to the earlier process than to the later process. This, in turn, would suggest that the time-line for the earlier part of the process (together with summary data of that time-line) is more representative of the overall process. One implication is that the overall process is more structured and orderly. This appears to qualify Bradac *et al.*'s claim that the later part of the project exhibits behaviour similar to Guindon's designer. One explanation may be that Guindon focused on a task requiring only one individual rather than interaction with other individuals. Also, there is an argument that Bradac *et al.*'s data has more ecological validity because it was collected 'from the field'.

2.7 Summary of the re-analysis of Bradac et al.'s conjectures

We have independently re-analysed several of Bradac *et al*.'s conjectures using Bradac *et al*.'s original empirical evidence. We have not sought to discredit Bradac *et al*.'s analyses, but rather to understand that analyses in more detail. We have made assumptions more explicit and this has helped to make the interpretation of the evidence more explicit. Our re-analysis has introduced a number of qualifiers to the original conjectures. Our main qualifiers are:

1. The ratio of 60:40 waiting to working appears to be most problematic for the tasks of working the process, reworking the process, waiting on reviews and waiting on other things. Furthermore, the evidence suggests that the ratio of 60:40 waiting to working may only apply to the earlier and middle parts of the process and not the end of the process.

2. There appears to be a tension within Bradac *et al*.'s study between modelling the process with a structured model and modelling the process with an opportunistic model. This tension has been discussed in other literature e.g. [10, 25-27].

3. As the process become more iterative, so the assumption of a sequential process is less easy to sustain. This introduces problems for the prevalence of waiting, as the conjecture on the prevalence of waiting seems to be based on an assumption of a sequential process.

4. The thresholds used by Bradac *et al*. to define tasks with a small, medium or large number of days appears to be problematic because it is not clear how the threshold were determined. This means that it is not clear which tasks in the process have the most affect on schedule interval.

   But developing alternative thresholds does not seem to be effective for identifying the most important tasks in the process, because these thresholds lack discriminatory power: they seem to identify a large number of tasks. Further analysis based on a scatter-plot of time spent waiting and time spent working does seem to be more effective and has identified two tasks which appear to affect both schedule interval and software quality. These two tasks are high-level design and low-level test.

5. Ranking the tasks according to the amount of work performed in the task (rather than the amount of waiting in the task) suggests insights for improving the quality of the product and not just for the time interval of the project.

3 Additional analyses of Bradac et al.'s evidence

3.1 The developer's states in the earlier and later parts of the process

Table 8 A detailed summary of the developer's states

State	Earlier part of project				Later part of project			
	Working		Blocked		Working		Blocked	
	f	%	*f*	%	*f*	%	*f*	%
Work process	**10**	13.2			**25**	32.9		
Documentation	**8**	10.5			**1**	1.3		
Rework process	1	1.3			**21**	27.6		
Rework documentation	2	2.6			3	3.9		
Wait Lab							2	2.6
Wait Expert			5	6.6			2	2.6
Wait Review			**17**	22.4			0	0.0
Wait Hardware			0	0.0			4	5.3
Wait Software			0	0.0			**8**	10.5
Wait Documentation			4	5.3			0	0.0
Wait Other			**29**	38.2			**10**	13.2
Sub-Total	21	27.6	55	72.4	50	65.8	26	34.2
Total (Working + blocked)			76	100.			76	100.

Key: Emboldened figures in the table denote the most interesting observations.

Table 8 provides a detailed summary of the various states of the developer in the earlier and later parts of the process.

The table suggests that more work and more rework occur in a later part of the process, that there is almost as much rework as work in the later part of the process, and that there is very little original documentation conducted later in the process, but some rework of documentation (perhaps in response to the rework of the process). These insights have implications for the prevalence of waiting and the structure of the process, which we have considered elsewhere in this report.

Regardless of the *prevalence* of waiting in the earlier and later parts of the project, we note that there appears to be a shift in the *kind* of waiting that occurs. In an earlier part of the process, the developer is waiting on reviews and on other things (perhaps being assigned temporarily to another project). In the later part of the process, there is a substantial drop in both the amount of waiting on other things and the amount of waiting on reviews. There is also some increase in waiting on software.

In other work [23], we have noted that there appears to be an increase in the urgency of a project as the project approaches its planned completion. (We understand 'urgency' to be the pressure of necessity i.e. it is *necessary* for the project to complete.) This may explain the reduction in waiting on other things. During the earlier parts of the project, the project has a lower urgency and other projects may be more urgent. Consequently, the developer may be reassigned, temporarily, to those other projects. As the current project proceeds into its later stages, so the project becomes more urgent (perhaps with other projects becoming less urgent). Consequently, the developer is not assigned to other projects. In addition, other developers may be assigned to the current project. This may explain why there is less waiting and more work and rework. Developers temporarily reassigned to the current project may get outstanding work completed, allowing this developer to get their work completed. Also, with the increasing urgency of the project, the project's managers may decide not to

hold reviews. Similarly, the developer may focus more on work and rework rather than on documenting that work and rework.

3.2 An alternative analysis of the developer's tasks

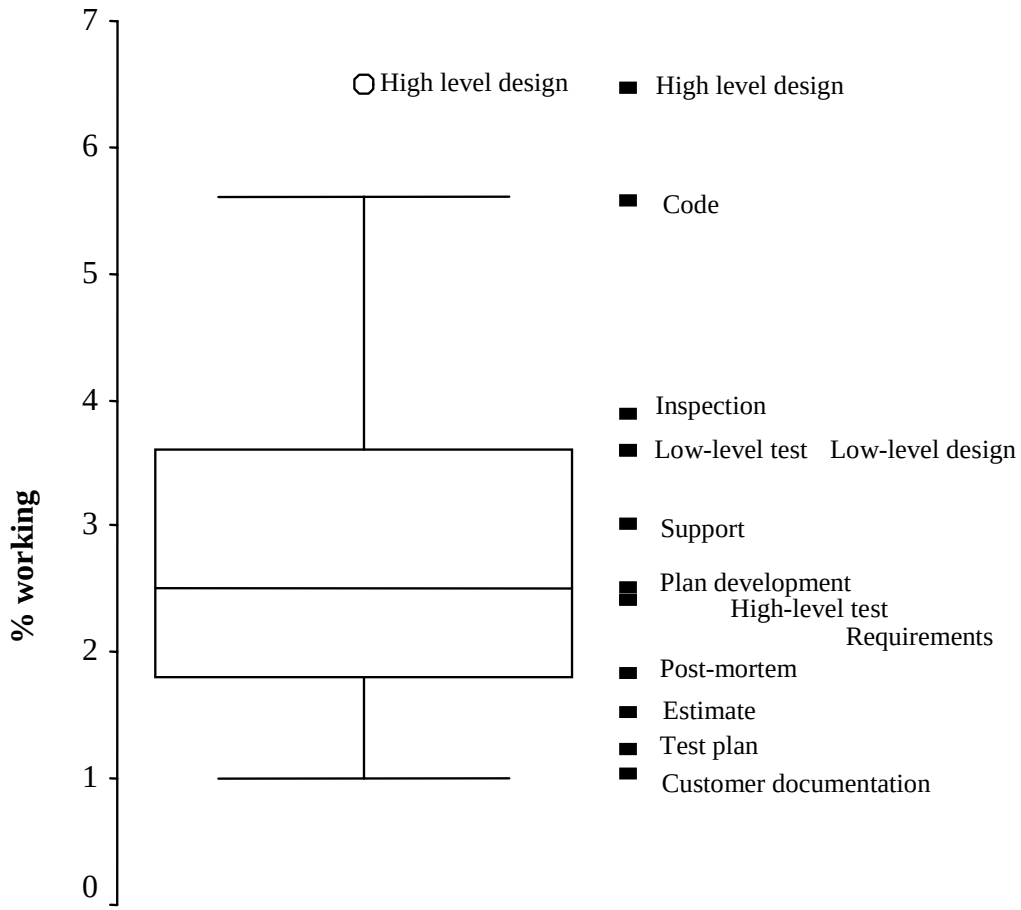

Figure 8 Combined box-plot and scatter-plot of the percentage of working for each task

In Figure 7, the task of high-level design (HLD) appears to be an outlier. Figure 8 confirms that the task of high-level design consumes an exceptional percentage of time in the process. Figure 8 also indicates that the task of coding is close to consuming an exceptional percentage of time in the process.

4 Discussion

4.1 A review and extension of the original conjectures

The secondary analyses of Bradac *et al.*'s empirical evidence have produced some interesting insights that clarify, enrich and extend the original conjectures. The analyses have also generated some new conjectures.

Table 9 summarises some of Bradac et al.'s original conjectures, together with comments and revisions from the secondary analyses. Table 10 summarises the new conjectures, together with comments on those conjectures.

Table 9 Revisions to Bradac *et al.*'s conjectures

#	Conjecture
1	The specific goal of our process monitoring experiment is to find ways to reduce the development interval. Bradac et al.'s explicit statement of the goal of their research has helped us in our secondary analyses because we can be clear about what goal (or criterion) we are evaluating their research against. At the same time, there are other goals involved in improving the process, such as product quality (see Table 10).
5	Overall, approximately 60% of the time is spent waiting or being reassigned to other projects. While 60% of the overall process may be spent waiting, the evidence indicates significant variation within the process. While the average ratio may be 40:60 (working to waiting) this ratio varies between the limits of 30:70 and 65:35. (Note that even in the best case scenario, 35% of the time is spent waiting! Only stating the average may be misleading for subsequent research that might build on Bradac *et al.*'s study. Furthermore, there appear to be particular tasks that exhibit the most variation. These are: working the process, reworking the process, waiting on reviews, and waiting on other things.
6	It seems clear that one important way to improving the process [to reducing development duration *cf.* conjecture 1] is to reduce significantly the number of days in blocking states. There is no evidence, however, to indicate what effect reducing the development interval would have on other goals, such as product quality. The lack of evidence is understandable, given the practical constraints of research and Bradac et al.'s research goal.
7	Blocking tends to be more prevalent at the beginning and the end of the process. Blocking tends to be most prevalent during the beginning of the process, *assuming* that the tasks in the process proceed in a broadly sequential manner (cf. conjectures 11 and 12). Our re-analyses found that, depending on the perspective taken, blocking occurred mainly during an earlier part of the process or, alternatively, throughout the process.
10	If blocking is more prevalent during the beginning and the end of the project, and this occurs at the project level, then one should attack blocking factors in the requirements, high-level design and high-level test phases of the process. As stated for conjecture 7, our re-analyses found that, depending on the perspective taken, blocking occurred throughout the process. Therefore, depending on one's assumptions, one should either attack blocking factors during an earlier part of the process, blocking factors during the earlier and later parts of the process, or blocking factors throughout the process.
11	The first part of the process is almost a pure waterfall process moving first through the plan development task and then to the requirements. This conjecture supports the assumption of a sequential process, which is necessary for the statement of conjectures 7 and 10. There appears to be a tension within Bradac et al.'s study and within the broader research community as to the validity of this conjecture.
12	The later part of the process is much more inter-mixed, consistent with Guindon's arguments that design is opportunistic. This conjecture seems to contradict the assumption of a sequential process, which is necessary for the statement of conjectures 7 and 10. As with conjecture 11, there appears to be a tension within Bradac et al.'s study and within the broader research community as to the validity of this conjecture.
13	A caveat: although this analysis is based on real data, they are reconstructed from one instance of the process, with some blurring of the accuracy because of retrospection. This conjecture also underpins our secondary analyses.

Table 10 Additional conjectures based on Bradac *et al.*'s evidence.

#	Conjecture
14	Waiting occurs during parts of the beginning, middle and end of the process
15	Attempts to improve the quality of the product should be directed at the middle of the process, as this is where the most work is actually done and hence where there are the greatest opportunities to introduce defects into the product. There is popular opinion in the research literature that the requirements process has the greatest impact on the success of software projects. Therefore, attention needs to be directed at how conjecture 15 relates to the opinions on requirements. It may be that most of the work is done in the middle of the process as a result of an ineffective requirements phase.
16	More work and more rework occur in a later part of the project.
17	There is as much work as rework in a later part of the project
18	There is very little documentation conducted in a later part of the project
19	There appears to be a shift in the type of waiting that occurs as the project progresses, from an emphasis on waiting on reviews and waiting on other things (notably reassignments to other projects) toward an emphasis on waiting on software.
20	The reduction in waiting on other things, the shift toward more work and more rework, and the shift toward waiting on software might be explained by an increase in the urgency of projects as they approach their planned completions.

Bradac *et al.*'s empirical evidence has been re-analysed in terms of other research objectives. The most notable of these is the objective of understanding project urgency.

4.2 The value of secondary analyses

The secondary analyses presented in this paper are, by definition, dependent on an appropriate quantity and quality of evidence being presented in a preceding publication. Given the constraints of journal and conference papers, it is unlikely that one would (or should) include empirical evidence that is surplus to the aims and objectives of a paper. (Because of the exploratory nature of the research being presented, Bradac *et al.*'s paper is a 'fortuitous' exception.) There may be value, however, in descriptive papers or reports that present 'empirical material', and that attempt to present that material with the assumptions about that material made explicit. Such descriptive papers could then be used in subsequent secondary analyses to help address a variety of different research questions, in particular research questions not considered by the researchers who collected and organised the original empirical material. For example, we have used Bradac *et al.*'s data (originally collected to investigate reducing development interval) to draw some insights about where to focus process improvements for improving software quality. Our analyses of Bradac et al.'s conjectures also highlight the importance of stating assumptions. For example, the assumption of a sequential process appears to be used to support the conjecture that waiting is more prevalent during the earlier and later parts of the process.

Connected to the issue of re-analysing evidence to investigate different research questions, different researchers may interpret the same empirical material differently for the same research question, as we have done so with the conjecture on the prevalence of waiting. Lee [28] argues that multiple, conflicting theories increase the degrees of freedom in an analysis, and thus strengthen the final claims of that analysis. Similarly, analysis of empirical material by a number of independent researchers can, in the long-term, strengthen the validity of the claims that are made from that material (although, in the short-term, the independent analyses may lead to disagreements). For example, there are studies that have re-analysed data used to build estimation models.

We have previously produced a descriptive paper [29] that presents 48 'analytical fragments' of the progress of a software development project. The analytical fragments are intended to present the empirical material separately from our interpretations of that material (which have been presented in, for example, [24]). The number of fragments is partly an indication of the 'breadth' and 'depth' of the evidence and partly an indication of our attempts to separate the data from its interpretation (recognising, of course, the profound problem of being able to truly separate data from its interpretation).

Some journals, for example *Empirical Software Engineering*, require researchers to (where appropriate) submit empirical data with their papers so that the data can be made available to others. Empirical data in itself, however, may not articulate the methods by which the data were collected, or the constraints on, and assumptions of, researchers during the collection and analysis of that data. Descriptive papers may better communicate such constraints and assumptions, and provide more information to help others in their evaluation or replication of previous empirical studies. We note that others (e.g. [30, 31]) are developing tools for sharing data whilst maintaining the integrity of that data.

Archaeological field evaluation may serve as an illustrative example of the purpose of descriptive papers. The *Institute of Field Archaeologists* define archaeological field evaluation as follows:

> "… a limited programme of non-intrusive and/or intrusive fieldwork which *determines the presence or absence* of archaeological features, structures, deposits, artefacts or ecofacts within a specified area or site on land, inter-tidal zone or underwater. If such archaeological remains are present field evaluation *defines* their character, extent, quality and preservation, and *enables an assessment of their worth* in a local, regional, national or international context as appropriate." ([32], p. 3; emphasise added)

As this definition indicates, archaeological field evaluation provides a description of a resource and an assessment of the value of that resource. (Clearly, an assessment of the value of a resource requires an awareness of how others may want to use that resource.) Note that archaeological field evaluation does not seek to explain or predict the occurrence of that resource i.e. does not investigate why or how that resource came to exist within the specified area or site. In making an assessment of the value of a resource, archaeological field evaluation would consider the potential benefits of a resource for subsequent explanations and predictions.

Another way of understanding the value of descriptive papers is in terms of critical thinking [33] (also known as informal logic [34]). With critical thinking one seeks to either produce sound arguments or to evaluate the soundness of existing arguments. Sound arguments have true premises and logical strength [33]. A logically strong argument is one where there is a robust connection between the premises and the conclusion. Therefore, a sound argument is one with both true premises and a robust connection of those premises with the conclusion. Note that there may be many 'threats' to the logical strength of an argument, such as implicit premises. Descriptive papers can be used to subsequently help critically evaluate the premises of an argument, including the identification of implicit premises.

4.3 Integrating the findings of Bradac *et al*. with other studies

Intuitively, there should be some connection between the behaviour of an individual and a project's ability to act [16]. In particular the actual use of an individual's time within a project ought to have some relationship to the actual use of time at the project level, and to eventual project outcomes. Bradac *et al*.'s study appears to be founded on the assumption that there is a connection between the individual's use of time and project outcomes.

Perry *et al*. state:

> "… there has been little research on time related behavior at the individual level (the micro level) and on the connection between the individual actions and the organization's

> ability *to act* – that is, on the relationships between the micro and macro levels." ([16], p. 2; emphasis in original)

Although there has been some work relating individual actions and a project's ability to act (most notably, Humphrey's work [35] on the Personal Software Process (PSP) and its relationship to the Capability Maturity Model (CMM), and Curtis *et al.*'s [9] classic analysis of the development of large software systems), there is a surprising lack of research in this area.

We believe that the secondary analyses presented in this report can be used as part of a wider aim to collect and analyse existing empirical evidence on the connections between individual actions and a project's ability to act. For example, Perry *et al.* [16] observed the number of times that developers were interrupted, a factor that would affect productivity. Van Solingen *et al.* [36] have further investigated interruptions.

As another example, Van Genuchten [37] collected data on 160 activities across six development projects to consider the impact of minor problems on the eventual outcomes of the projects. Van Genuchten's activities seem to loosely map to Bradac *et al.*'s tasks, and this mapping provides an opportunity for relating the micro behaviour of individual tasks to the macro behaviour of projects.

We have observed [24, 38]} that reports of waiting, outstanding work and the poor progress of work are all more prevalent during the later parts of a project than during the earlier and middle parts of the project. These reports occurred at the team level, and seem to contradict the conjectures of Bradac *et al.* at the individual level. But this contradiction may be resolved if one recognises the presence of additional factors. For example, reporting the status of a development team may not only be a function of the actual status of that team, but may also be a function of the status of the project. As a project approaches its planned completion, the urgency of the project increases and this may make reporting more important. Conversely, during the beginning of a project, there is less urgency and so team leaders may not report waiting when it occurs.

Finally, attempts to integrate findings from a number of studies suggest the need for one or more frameworks within which to position empirical studies of actual software development. Such frameworks may be particularly valuable if they attempt to integrate the empirical *findings* of the studies and not just classify the technical characteristics of those studies (such as the research strategy, or sample size, or methods of analysis). Such frameworks may come to form partial, or incomplete, theories in themselves.

Finally, as noted in the introduction to this report, Bradac *et al.*'s study was a pilot study for a more substantial subsequent study. Also, other studies have referenced the Bradac *et al.* pilot study. One logical extension to this research is to conduct secondary analyses of these subsequent papers.

5 Conclusions

We have reported on secondary analyses of the conjectures and empirical evidence presented in Bradac *et al.*'s pilot study. Our secondary analyses leads to a clarification, enrichment and extension of several of Bradac *et al.*'s original conjectures. We note that our analyses were only possible because of the quality and quantity of evidence presented in the original paper. We also note that such quality and quantity is unusual for journal and conference papers (principally because of the editorial constraints placed on such papers) and this leads us to suggest that there is value in publishing 'descriptive papers' that seek to present 'empirical material' (together with an explicit statement of goals, assumptions and constraints) separate from the analyses that proceeds from that material. Such descriptive papers can improve the 'public scrutiny' of software engineering research and may respond, in part, to Glass's [1] criticisms concerning the small amount of software engineering research that is actually evaluated.

Acknowledgements

Some of the secondary analyses presented in this paper were first conducted as part of my doctoral research whilst at Bournemouth University and IBM Hursley Park. The bulk of the analyses, however, have been conducted subsequent to the doctoral research.

A. Appendix

A.1 Detailed statements from Bradac et al.

Table 11 presents key statements made in Bradac *et al.*'s paper. We include these statements here for three reasons. First, the statements provide a detailed summary of Bradac *et al.*'s arguments. Second, being a detailed summary, the statements provide context to the empirical evidence that has been taken from Bradac *et al.*'s study and re-presented and analysed in this report. Third, the statements provide the 'raw material' from which we have 're-constructed' Bradac *et al.*'s conjectures (as presented in section 1 of this report). Where the reader is further interested in our 're-construction', these statements provide a basis from which to pursue those interests. In the table, the page number refers to the page in Bradac et al.'s original paper.

Table 11 Statements from Bradac et al.

#	Statement	Page
1	Features are often the basic unit of development for very large software systems and represent long-term effects, spanning several years from inception to actual use.	774
2	… monitoring these processes is a long-term effort as well	
3	Time, like costs, can be viewed as a unit of optimisation in improving software development processes.	774
4	The specific goal of our process monitoring experiment is to find ways to reduce the development interval.	774
5	We want to find out what people actually do when they add features to a large software system.	776
6	What people do and how they interact are of paramount importance in engineering processes.	776
7	We want to understand how people progress through their activities… and where and how they are hindered from making that progress in those activities.	776
8	With features as the unit of development, it is important to understand what people do when developing a feature, how they interact within a single feature development, and how the different development groups interact with each other in developing several features concurrently.	776
9	The purpose of this experiment is to provide an understanding of the feature development process. We then use this understanding as the basis both for accurate descriptions of, and for substantive improvements to, these processes.	776
10	Our motivation for prototyping the experimental design is also straightforward: The experiment will be a long-term effort, and we want to work out as many wrinkles as possible before committing to the actual experiment.	776
11	In general, the [the TAME project and Amadeus system] emphasis is on automating measurement and control of evolving products. This approach assumes that we know what needs to be measured to control the process at the desired level of precision within the desired cost constraints. We believe that we have not reached that level of maturity in the measurement and control of software processes.	777/ 778
12	A process is characterised by a set of tasks and a set of states. The tasks define the various activities within the process that are of interest and that will be sampled in the experiment. The states represent either progress within a task or lack of progress (that is, where the task is blocked for some reason).	778/ 779
13	The intent is to sample the most important aspect of the task on the previous day.	779
14	… people often do multiple tasks concurrently and do multiple things during the course of a workday. However, this is ameliorated by the facts that we monitor per-feature development (which will differentiate the effort of a develop working on several features concurrently) and that we have a blocking category for other assignments (which will differentiate a developer doing things other than working on this feature).	780
15	Furthermore, we assume that developers really do only one or two things per day per process – that is, their time is not overly fragmented.	780
16	We need three things to prototype the experiment: a process, a development, and a set of analyses.	780
17	For the process, we selected a well-understood one that is a standard development process.	780
18	For the development, we selected a relatively large feature development…[because]… we felt that a large, complex feature would stress the experiment in such a way that if there were inherent problems, we would see them.	780

19	For the analyses, we selected a set of basic views of the data to consider both the blocked and working states of the process.	780
20	… the reconstructed development data represents only one path through the process – that is, it is one instance of the process.	782
21	… the accuracy of the data are open to question; because of the retrospective nature of the data	782
22	We also note that 60% of the time was spent waiting rather than being productive	782
23	Although there may well be other factors to consider, it seems clear that one important way of improving the process is to reduce significantly the number of days in blocking states.	782
24	The utility of this conjecture depends on the degree of multiplexing within these processes. Clearly, if the global level of blocking is consonant with this local level, the conjecture will hold.	782
25	Approximately 27% of the total process time was spent working and reworking the process, and approximately 13% of the time was spent writing and rewriting the documentation.	783
26	About one third as much time was spent reworking the process as working it; about one-half as much time was spent rewriting documentation as was spent initially [writing] it.	783
27	The other thing to note… is that blocking tends to be more prevalent at the beginning and end of the process.	783
28	The middle of the process… tend no to be interrupted by waiting on other factors.	783
29	Clearly, one should attack the blocking factors in the requirements, high-level design, and high-level test phases of the process, because they are the more heavily weighted.	783
30	It will be interesting to see if this conjecture… holds over a wide variety of developments.	783
31	It is worth noting that the first part of the development is almost a pure waterfall process.	783
32	It will certainly be interesting if this holds over a large class of processes and developments. The important question then will be the source of this linearization.	783
33	The second thing worth noting is the lengthy blocking times in this phase of the process. In particular, note the time spent waiting on reviews…	783
34	Not surprisingly, waiting for reviews dominated both the beginning and the end of the process, but was relatively infrequent in the middle.	783
35	… a later part of the process, however, shows a completely different overall process. Various tasks are intermixed, alternating between four and five different tasks, and the various blocking factors are intermixed aswell… This reflects more the kind of process we would expect…	783
36	We reiterate our caveat about these data: Though they are real data, they are reconstructed data of only one instance of the process, with some blurring of accuracy because of retrospection. We feel, however, that there are some intriguing conjectures about our feature development processes that we hope to validate with subsequent experiments.	783
37	We conclude that just as prototyping is often a necessary auxiliary step in a large-scale, long-term development effort, so, too, is prototyping a necessary step in the development of a large-scale, long-term process monitoring experiment.	783

References

1. Glass, R.L., *The software research crisis.* IEEE Software, 1994. **11**(6): p. 42-47.
2. Tichy, W.F., et al., *Experimental Evaluation in Computer Science: A Quantitative Study.* Journal of Systems Software., 1995. **28**(1): p. 9-18.
3. 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.
4. Bradac, M.G., D.E. Perry, and L.G. Votta, *Prototyping A Process Monitoring Experiment.* IEEE Transactions on Software Engineering, 1994. **20**(10): p. 774-784.
5. Billings, C., et al., *Journey To A Mature Software Process.* IBM Systems Journal, 1994. **33**(1): p. 46-61.
6. NASA, *An Overview Of The Software Engineering Laboratory*. 1994, NASA Goddard Space Flight Center: Greenbelt, Maryland.
7. 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.
8. Pajerski, R., S. and V.R. Basili. *The SEL Adapts To Meet Changing Times*. in *22nd Software Engineering Workshop*. 1997. NASA/Goddard SEL, Greenbelt, Maryland, December 1997.
9. Curtis, B., H. Krasner, and N. Iscoe, *A field study of the software design process for large systems.* Communications of the ACM, 1988. **31**(11): p. 1268-1287.
10. Guindon, R., *Designing the design process: Exploiting opportunistic thought.* Human-Computer Interaction, 1990. **5**(2-3): p. 305-344.
11. Guindon, R., *Knowledge exploited by experts during software system design.* International Journal of Man-Machine Studies, 1990. **33**(3): p. 279-304.
12. Walz, D.B., J.J. Elam, and B. Curtis, *Inside A Software Design Team: Knowledge Acquisition, Sharing, And Integration.* Communications of the ACM, 1993. **36**(10): p. 63-77.
13. Potts, C., *Software-Engineering Research Revisited.* IEEE Software, 1993. **10**(5): p. 19-28.
14. Bradac, M.G., D.E. Perry, and L.G. Votta. *Prototyping A Process Monitoring Experiment*. in *15th International Conference on Software Engineering*. 1993: IEEE Computer Society Press.
15. Perry, D.E., N.A. Staudenmayer, and L.G. Votta, *People, organizations, and process improvement.* IEEE Software, 1994. **11**(4): p. 36-45.
16. Perry, D.E., 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, Editors. 1995, John Wiley and Sons Ltd.
17. Ballman, K. and L.G. Votta. *Organizational congestion in large-scale software development*. in *Third International Conference on Software Process*. 1994. Reston, Virginia, USA, October: IEEE Computer Society Press.
18. Dandekar, A., D.E. Perry, and L.G. Votta, *A study in process simplification.* Software Process: Improvement and Practice, 1997. **3**(2): p. 87-104.
19. Florac, W.A., A.D. Carleton, and J.R. Barnard, *Statistical Process Control: Analyzing A Space Shuttle Onboard Software Process.* IEEE Software, 2000. **17**(4): p. 97-106.
20. Waterson, P.E., C.W. Clegg, and C.M. Axtell, *The dynamics of work organisation, knowledge, and technology during software development.* International Journal of Human-Computer Studies, 1997. **46**(1): p. 79-101.
21. Sommerville, I. and S. Monk. *Supporting informality in the software process*. in *Third European Workshop on Software Process Technology (EWSPT'94)*. 1994. Villard de Lans, France: Springer-Verlag.
22. Rodden, T., et al. *Process modelling and development practice*. in *Third European Workshop on Software Process Technology (EWSPT'94)*. 1994. Villard de Lans, France: Springer-Verlag.
23. Rainer, A. and M.J. Shepperd, *An empirical investigation into waiting in software development projects.* 1998, Bournemouth University.
24. Rainer, A.W., *An Empirical Investigation of Software Schedule Behaviour*, in *Department of Computing*. 1999, Bournemouth University: Bournemouth UK.
25. Visser, W., *Organisation Of Design Activities: Opportunistic, With Hierarchical Episodes.* Interacting With Computers, 1994. **6**(3): p. 235-238.
26. Davies, S.P., *Characterizing The Program Design Activity: Neither Strictly Top-Down Nor Globally Opportunistic.* 10, 1991. **3**(173-190).
27. Suchman, L., *Office Procedures as Practical Action: Models of Work and System Design.* ACM Transactions on Office Information Systems., 1983. **1**(4): p. 320-328.

28. Lee, A.S., *A scientific methodology for MIS case studies.* MIS Quarterly, 1989. **13**(1): p. 33-50.
29. Rainer, A.W., *A catalogue of 'analytic fragments' of the behaviour of a software project*. 2000, Empirical Software Engineering Research Group (ESERG) Bournemouth University: Bournemouth.
30. Kitchenham, B., R.T. Hughes, and S.G. Linkman, *Modeling software measurement data.* IEEE Transactions on Software Engineering, 2001. **27**(9): p. 788-804.
31. Kitchenham, B., S.L. Pfleeger, and N. Fenton, *Toward a Framework for Software Measurement Validation.* IEEE Transactions on Software Engineering, 1995. **21**(12): p. 929-943.
32. Institute of Field Archaeologists, *Standard And Guidance For Archaeological Field Evaluation*. 1999, Institute of Field Archaeologists (University of Reading): Reading, UK. p. 21.
33. Hughes, W., *Critical Thinking: An Introduction To The Basic Skills*. 2nd ed. 1996, Ontario, Canada: Broadview Press Ltd.
34. Fisher, A., *The Logic of Real Arguments*. 1988: Cambridge University Press.
35. Humphrey, W., *Three process perspectives: organizations, teams, and people.* Annals of Software Engineering, 2002. **14**: p. 39-72.
36. van Solingen, R., E. Berghout, and F. van Latum, *Interrupts: Just A Minute Never Is.* IEEE Software, 1998. **15**(5): p. 97-103.
37. van Genuchten, M., *Why is software late? An empirical study of reasons for delay in software development.* IEEE Transactions on Software Engineering, 1991. **17**(6): p. 582-590.
38. Rainer, A. *Waiting in software projects: an exploratory study*. in *International Symposium on Empirical Software Engineering*. submitted. September 2003, Rome, Italy.