



Comparative measures of software quality and productivity across different software development environments

R.J. Cole & D. Woods

*Software Measurement Laboratory, Glasgow
Caledonian University, Cowcaddens Road,
Glasgow, G4 0BA, UK*

ABSTRACT

This paper presents preliminary results for comparing software development methods in the context of quality and productivity attributes. A software data collection system over a defined standard software lifecycle has been implemented across company sites. Product, process and resource data have been recorded. Different software development methods were covered by the first set of industrial case studies. The work is part of a long term project to establish quantitative measures to support quality improvement, software assessment and certification.

INTRODUCTION

This paper describes current work done under the project META (Measurement-based Experiment for Techniques Assessment, (Cole & Woods[1]) to develop software measures through the lifecycle. It reports some early results from the first set of industrial case studies on quality and productivity measures that bear on the development method used.

The long term aim of the META project is to establish comparative software *quality* and *productivity* lifecycle measures. Data is being collected throughout

402 Software Quality Management

the software lifecycle on a set of industrial projects. Evidence is being gained on the effect of adopting different development techniques on downstream product and process quality attributes.

The main components of the measurement project are:

- a software data model,
- a data collection mechanism,
- a set of case studies,
- data presentation and analysis.

An overview of the META project is given in Woods and Cole [2].

SOFTWARE LIFECYCLE MODEL

A general observation or data model was developed (Woods & Cole [3]) for implementation on different software producing sites. Four industrial software development sites were selected for the first phase of the measurement experiment.

The Software Quality Data Collection System is built around a conventional waterfall lifecycle model defining nine phases P1 to P9, bounded by ten milestones M1 to M10.

This data collection system is designed for any software project whose development path can be reasonably mapped on to this lifecycle. Naming of phases and milestones is given in Table 1 below.



Milestone No.	Milestone Descriptor	Phase No.	Phase Descriptor
M1	Start of Project		
		P1	Preliminary Investigation
M2	End of Preliminary Investigation		
		P2	Requirements Analysis
M3	End of Requirements Analysis		
		P3	System Design
M4	End of System Design		
		P4	Detailed Design
M5	End of Detailed Design		
		P5	Coding & Unit Testing
M6	End of Coding & Unit Testing		
		P6	Integration Testing
M7	End of Integration Testing		
		P7	System Testing
M8	End of System Testing		
		P8	Acceptance Testing
M9	End of Acceptance Testing		
		P9	Maintenance
M10	End of 1year Maintenance		

Table 1: Milestones and Phases of the META lifecycle model

The model provides a baseline milestone framework for the comparison of different development methods.

SOFTWARE DATA MODEL

The data model is the framework which defines the metrics to be recorded and specifies how they are to be measured. For each lifecycle phase and milestone a set of metrics has been identified. Additional information at the global project level is also defined.

404 Software Quality Management

The Goal-Question-Metric (GQM) approach (Basili & Rombach, [4]) was used to define appropriate data items for the collection of software data across organisations and projects. The AMI method (Kuntzmann-Combelles et al, [5]) was used to develop metrics for the META data model. More details of the application of the GQM approach to the META data model are given in Cole and Woods [6].

The initial data model was designed to harmonise with the design structure of a data collection tool Data Collection and Storage System (DCSS [7]). Data items are associated with one of three entities or levels:

- the project itself,
- milestones,
- components.

Data categories are summarised below.

Project level data

Information recorded includes:

- Project Description, Product Application, Failure Risk Class, Language, Functional Complexity & Function,
- Operational Characteristics, ISO 9126 Quality Attributes (ISO [8]), Project Management Tools & Quality Standards,
- Tools & Methods for Requirements Analysis, System Design, Detailed Design, Coding and Unit Testing,
- Budget & Schedule Constraints Criticality, Client Application Experience & Availability.

Milestone level data

Information recorded includes:

- Staffing Levels and Phase Effort, Phase Duration & Milestone Dates,
- Product Size & Reuse, Phase Stability.

Component level data

Information recorded includes:

- Staff Experience and Tasks, Code Modules and Documentation, Incidents and Changes.

Data Model implementation

To allow data across case studies to be aggregated and compared it was essential that data be collected in a uniform way. The META data collection system has ensured this by:

- implementing a defined data model,
- standardising the definitions of all metrics.

The META data model supports the mechanism for managing and controlling both the definitions of software attributes and their corresponding measures. Metrics definitions are set out in a data dictionary (Woods & Cole [3]).

PHASE 1 CASE STUDIES

In the first phase of the experiment, four industrial software developments were monitored as case studies. An overview of selected case study features is shown in Table 2. It may be seen that the four case studies used four different development methods. The case studies will be denoted by A1, B1, C1 and D1. The development methods will be denoted by MBY, TDP, INF and MAS respectively

	Case Study A1	Case Study B1	Case Study C1	Case Study D1
Project attributes				
Application Class	Information Management System	Knowledge Based System (with database interaction)	Information Management System	Control System
Failure Risk Class	Limited economic loss or damage	Limited economic loss or damage	Work loss only	Work loss only
Design Method(s)	Mixed - Bespoke + Yourdon	Informal Top Down (paper)	Inforem	MASCOT
[method AA]	[method MBY]	[method TDP]	[method INF]	[method MAS]
Effort (person-months)	124	6-8	33	6
Product Size-KLOC	34	12	150	3
Language(s)	INFORMIX, C	C, SQL, Crystal	Telon generated COBOL	Pascal / Assembler
Project Duration(m)	8	4-5	13	5

Table 2: Overview of Phase I Case Studies

Data Collection Mechanisms

Two tools were used to support the data collection process. Unified data collection and storage on each site was ensured by M-BASE, formerly called Data Collection and Storage System, (DCSS [7]) developed by the NCC. M-BASE, containing a set of on-screen forms, was calibrated to the META Data Model. A master system was installed at the Software Measurement Laboratory with satellites installed at each different observation site. Communication between master and satellite for data transfer and aggregation was via export and import facilities.

The QUALMS tool, QUality AnaLysis and Metrication Software, (QUALMS [9]) originally developed by South Bank University and subsequently enhanced at Glasgow Caledonian University was used to perform static code analysis. Code structure measures may be exported to M-BASE.

Collection of process and product data was considerably eased through the use of the single standard data collection and storage system.

PRELIMINARY RESULTS ON DEVELOPMENT METHODS

At this stage in the experiment, the Phase 1 Case Studies from companies A, B and C. are essentially complete. Results from Case Study D are not complete and are not included here. Four different design methods were used, corresponding to the Detailed Design Phase 4 in the META lifecycle model.

Five sets of simple measures will be used to consider methods MBY, TDP and INF in the context of results for Case Studies (or projects) A1, B1 and C1. They are concerned with:

- effort phase distribution,
- Incident origin phase distribution,
- Incident origin-detection phase lag distribution,
- Effort estimation accuracy phase distribution,
- Code productivity.



408 Software Quality Management

With such a small number of project data sets, comparisons at this stage between the design methods used must be made with caution. Nevertheless some interesting overall comparisons are already emerging.

Effort phase distribution.

In Figure 1 the development effort distribution is shown over the lifecycle phases for the projects A1, B1 and C1.

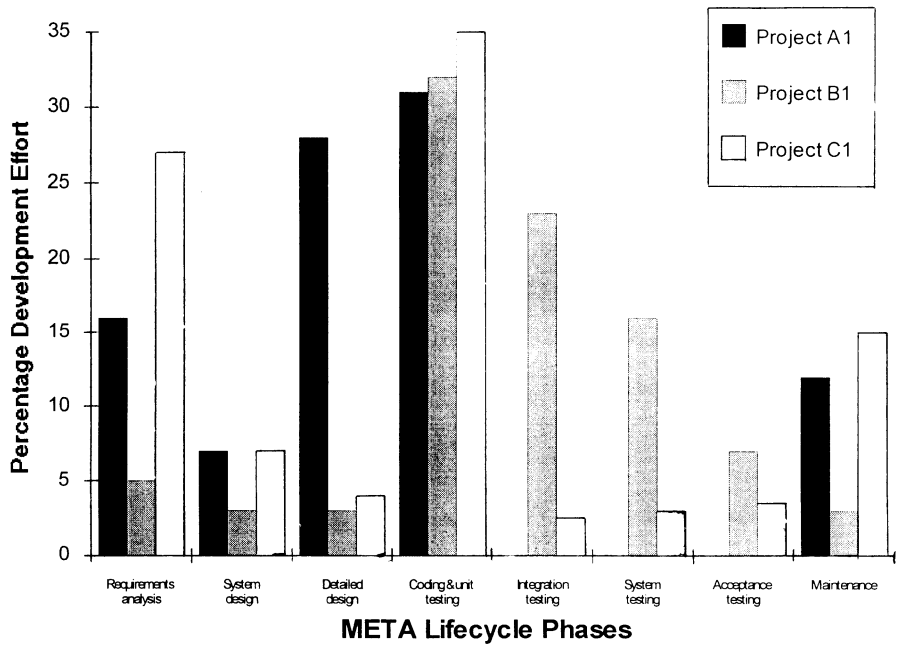


Figure 1: Development effort distribution for projects A1, B1 and C1.

In Table 3 summary values are shown.

Phases	Phase Names	Project A1	Project B1	Project C1
		%	%	%
1	Preliminary investigation	6	8	3
2-3	Requirements to System Design	23	8	34
4	Detailed Design	28	3	4
5	Coding & Unit Testing	31	32	35
6-8	Integration to Acceptance Testing	0 ¹	46	9
9	Maintenance	12	3	15
TOTAL %		100	100	100

Table 3: Development effort distribution for projects A1, B1, & C1

Note: ¹ Integration, system and acceptance testing performed by customer

This data is contextual; that is, there is not necessarily a performance criteria associated with the percentage of effort used in the Detailed Design phase. It gives a context for the other data comparisons however.

Incident origin phase distribution.

In Figure 2, the distribution of incidents originating in different lifecycle phases is shown for projects A1, B1 and C1. An incident is defined to be 'any deviation from expected or specified service of the product or an event of non-corrective product change'.



410 Software Quality Management

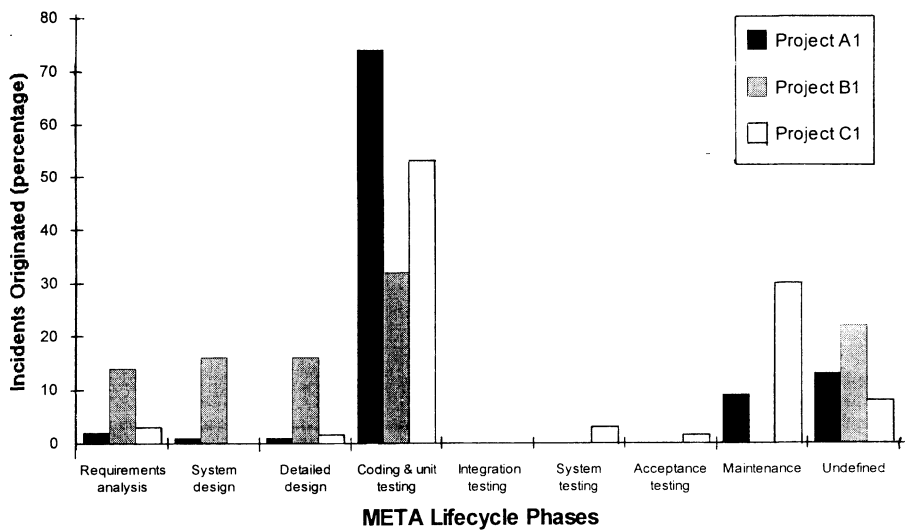


Figure 2: Incident origin distribution for projects A1, B1 and C1

In Table 4, summary values are shown.

Phases		Project A1	Project B1	Project C1
Phase Names		%	%	%
2-3	Requirements to System Design	3	30	3
4	Detailed Design	1	16	1.5
5	Coding & Unit Testing	74	32	53
6-8	Integration to Acceptance Testing	0 ¹	0 ¹	4.5
9	Maintenance	9	0 ¹	30
	Undefined ²	13	22	8
TOTAL %		100	100	100

Table 4: Incident origin distribution for projects A1, B1 and C1

Notes: ¹ Data is not currently available for these categories

² Incidents not identified with specific phases



Supporting data indicates that incidents originating early cost more to put right. From this data, C1 and A1 appear to be less costly than B1, the advantage being with C1. It must be emphasised that percentage distributions are being compared, not absolute numbers of incidents.

Incident origin-detection phase lag distribution.

In Figure 3, the detection lag between an incident originating and being detected in different lifecycle phases is shown for projects A1, B1 and C1.

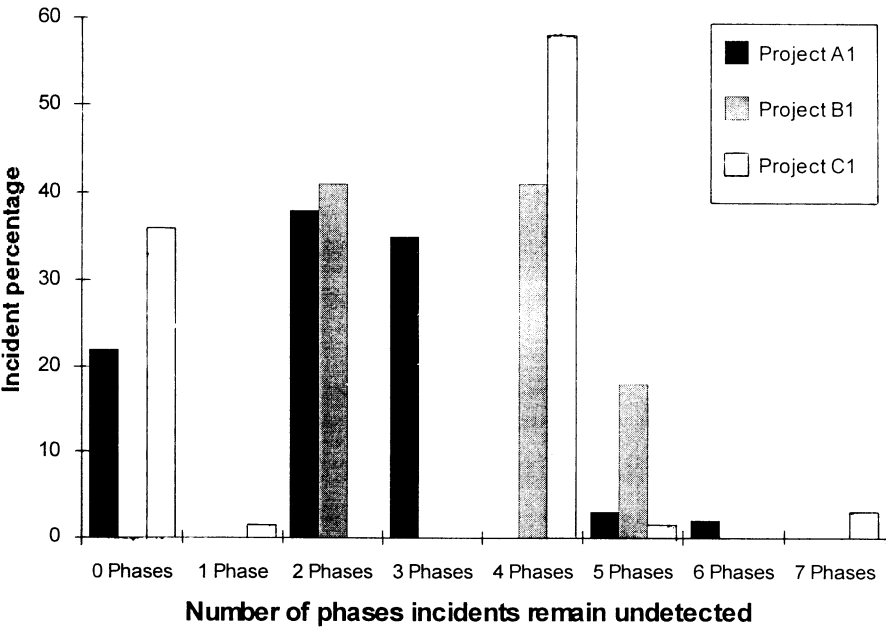


Figure 3: Incident origin-detection lag for projects A1, B1 and C1

In Table 5, the incident origin to detection lag results are summarised.



412 Software Quality Management

Origin-detection Phase lag	Project A1 %	Project B1 %	Project C1 %
0-1	22	0	37.5
2-3	73	41	0
4-5	3	59	59.5
6-7	2	0	3
TOTAL %	100	100	100
Mean	2.1	3.4	2.6
Median	2	4	4

Table 5: Incident origin-detection-lag distribution for projects A1, B1 & C1

Supporting data indicates that incidents corrected late after being detected cost more to put right. From this data, it would appear that A1 is the least costly, C1 the next and B1 the most costly.

Effort estimation accuracy phase distribution.

In Figure 4, the accuracy of the estimates for effort over different lifecycle phases is shown for projects A1, B1 and C1. This is given as a percentage (+ or -) change between the actual value and the final estimated value of the effort.

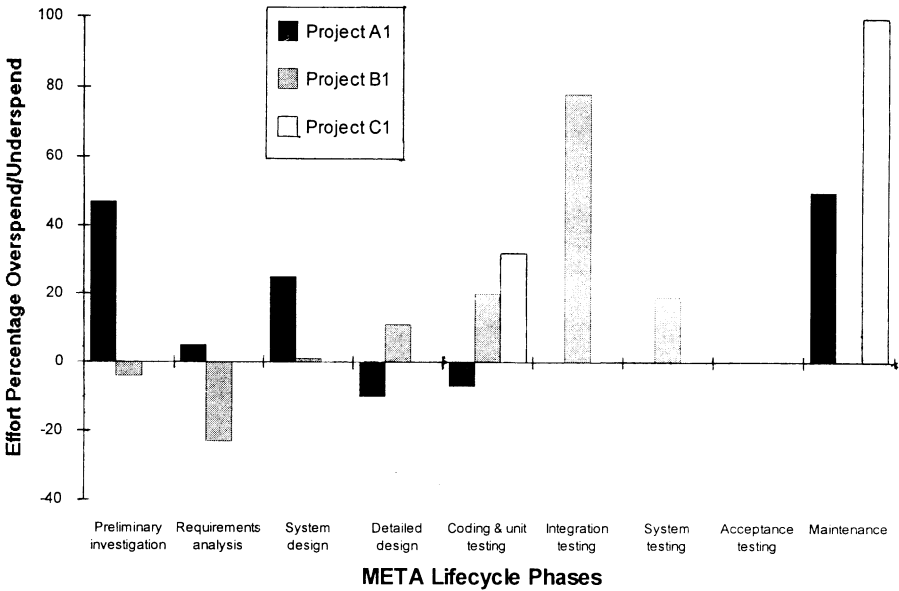


Figure 4: Effort estimation accuracy for projects A1, B1 and C1

In Table 6, the planning effort accuracy is summarised.

Phases	Phase Names	Project A1	Project B1	Project C1
		% change	% change	% change
1-3	Preliminary investigation to System Design	+5 to +47	-23 to +1	0
4	Detailed Design	-10	0	0
5	Coding & Unit Testing	-7	+4	+32
6-8	Integration to Acceptance Testing	0 ¹	+150 to +566	0
9	Maintenance	+50	0 ¹	+100

Table 6: Effort planning accuracy for projects A1, B1 and C1

Notes: ¹ Data is not currently available for this category



414 Software Quality Management

From the data it appears that Project A1 has the best planning record with C1 next and B1 the highest inaccuracy.

Productivity

Table 7 shows the productivity measured by LOC/effort hour. It must be remembered that the language varied and so no strong conclusions can be drawn.

	Project A1	Project B1	Project C1
Productivity (LOC/effort hour)	3.1	13.7	2.4
Reuse	no reuse	40% code reuse	some design reuse
Language(s)	Informix, C	C, SQL, Crystal	Telon generated COBOL

Table 7: Productivity values for projects A1, B1 and C1

If the results from comparing projects A1, B1 and C1 can be associated with their respective development methods MBY, TDP and INF, then Tables 3-7 may be summarised by a single set of performance indicators, as shown in Table 8.

	Project A1:	Project B1:	Project C1:
Performance indicator	Method MBY	Method TDP	Method INF
Incident cost	2nd	3rd	1st
Incident lag	1st	3rd	2nd
Effort estimation accuracy	1st	3rd	2nd
Productivity	2nd	1st	3rd
% Effort of Detailed Design	High	Low	Low
Overall rating	1st	3rd	2nd

Table 8: Development methods performance indicator ratings

On the evidence of the data, it seems that method MBY may lead to a better quality and productivity rating than INF or TDP. Nothing definite can be concluded at this stage, for lack of data, due to the small sample. The projects were carried out in different companies at different sites, with different teams, using different languages and methods.

CONCLUSIONS AND FUTURE WORK

The results to date are encouraging. Clear differences in project ratings are apparent. It remains an open question whether the performance indicator differences are significantly impacted by the development method chosen or swamped by other factors. More data on more projects will lead to more confidence in the results and conclusions.

The work done to date shows that it is possible to collect the same data on different projects in different organisations and aggregate this information for comparison purposes. This work harmonises with the suggested QUANTUM programme (QUANTUM [10]) which aims to create a national UK measurement-based framework for the assurance of software quality.

REFERENCES

- 1 Cole, R.J. & Woods, D., Measurement-based Experiment for Techniques Assessment (META), *UK dti/SERC Information Engineering Advance Technology Programme. Systems Engineering Methods and Tools Project IED P3155. Level II Workplan*, 1992.
2. Woods, D. & Cole, R.J., *Implementing a software data model in a multi-company experiment*, Pro. 3rd Software Quality Workshop, Napier University, Edinburgh., 1993.



416 Software Quality Management

3. Woods, D. & Cole, R.J., *Observation Model First Report*, Report R1.2.1, Measurement-based Experiment for Techniques Assessment (META), Project IED P3155, 1992.
4. Basili, V.R., & Rombach, D., The TAME project - Towards improvement-oriented software environments, *IEEE Transactions on Software Engineering*, 14 (6), pp 758-773. 1988.
5. Kuntzmann-Combelles (Ed) *et al*, ESPRIT project AMI (Assess, Analyse, Metricate, Improve), *The AMI handbook*, pp170, 1992
6. Cole, R.J. & Woods, D., Measurement through the software lifecycle: a comparative case study. *Proceedings of the Centre for Software Reliability Annual Workshop on Software Quality Assurance and Measurement.*, Amsterdam, 1993. Chapman and Hall, 1994.
7. DCSS, *Data Collection and Storage System (DCSS) user manual*, version 2, National Computing Centre Ltd, 1992.
8. ISO/IEC, *International Standard ISO/IEC 9126, Information technology - software product evaluation - Quality characteristics and guidelines for their use*, pp 13, 1992
9. QUALMS. *SML-QUALMS user manual*, version 4.0, Software Measurement Laboratory, Glasgow Caledonian University, 1992.
10. QUANTUM, *The QUANTUM Study Report, A measurement-based framework for the assurance of software quality*, prepared by Praxis on behalf of the Department of Trade and Industry, pp .30, 1992.