The Performance of Business Application, Real-Time and Component Software Projects

An analysis of COSMIC-measured projects in the ISBSG database.

March 2012

Abstract

This report provides an analysis of the data in the ISBSG database with software sizes measured using the COSMIC Functional Size Measurement method, including all data collected in the COSMIC/ISBSG Benchmarking Initiative up to November 15th 2011.

Performance data on 324 projects from the domain of business application software (new developments, re-developments and enhancements), 40 projects from the domain of real-time applications and 22 projects to develop software components are analyzed to produce benchmarks suitable for performance comparisons and for project estimating.

Published by:
The Common Software Measurement International Consortium ('COSMIC'), and
The International Software Benchmarking Standards Group ('ISBSG')

Compiled and edited by Harold van Heeringen and Charles Symons, COSMIC

Reviewed by:
Chris Lokan, ISBSG
Peter Hill, ISBSG
Alain Abran, COSMIC
Luca Santillo, COSMIC

Contents

1. Introduction

This report presents an analysis of the data on projects in the ISBSG database with software sizes measured using the COSMIC Functional Size Measurement (FSM) method, including all data collected in the COSMIC/ISBSG Benchmarking Initiative up to November 15th 2011. The aim is to produce data that will be useful for benchmark comparisons and for input to project estimating methods, such as described in [1]

The projects were completed in the period 1999 – 2011.

This report is an updated version of the first benchmark report, dated June 2009, with the addition of more project data.

Readers are assumed to have a broad understanding of the COSMIC method for measuring a functional size of software. For those needing an overview of the method, go to www.cosmicon.com, or obtain the 'Method Overview' document [2].

After this Introduction, the report has two main Chapters containing the principal benchmark data analyses.

- Chapter 2 concerns the analysis of 324 projects from the domain of business application software.
- Chapter 3 concerns the analysis of the data on
    - 40 projects that have been classified as being concerned with real-time applications,
    - 22 projects that delivered re-usable software components, and
    - some miscellaneous types of software.

The reasons to distinguish the two main categories of business application and real-time software are that it is anticipated that the performance data for these projects will vary with the software domain, and that the audiences for these two main categories are different.

The main performance parameter analysed is that reported by the ISBSG to establish benchmarks for all projects in its database, namely the 'project delivery rate', or 'PDR', defined by:

PDR = Effort (Work-hours) / Size (COSMIC Function Points)

N.B. This parameter is the inverse of the more conventional 'productivity' parameter, so the lower the PDR, the higher the productivity, and vice versa. Graphs in the report that show individual project performance as scatter diagrams have a horizontal x-axis labelled 'Size (CFP)', with the vertical axis labelled 'Effort (Work-hours)' or 'Elapsed months'. On all these graphs, individual projects represented by dots that appear higher up the vertical axis (or above a trend-line) have a poorer performance i.e. lower productivity (=higher PDR) or lower speed respectively, than projects shown by dots lower down or below a trend line.

A few graphs show conventional project 'productivity' (measured in CFP/work-month) and project 'speed' (CFP/elapsed month) versus a parameter such as size of the delivered software. For these graphs, the higher the dot or bar, the higher the productivity or speed.

Another important point to note when using the benchmark figures is that the effort data used for the analyses is not the effort reported to the ISBSG for each project, but the 'Normalized Level 1 effort'. This is because the range of activities accounted for by any reported effort figure is often less than the real total project effort and the 'reported versus real' difference varies from one project to another. The ISBSG attempts to compensate for this variability by calculating the 'Norm L1 effort' which aims to cover all project activities from 'specify' to

'implementation', and 'planning'. How this is achieved and how it affects the effort data used in the analysis is discussed in Appendix A.

Within each project category, the results of other analyses are shown where statistically possible and of interest, for example whether the median[1] PDR varies by language level (3GL, 4GL), specific programming language and with technical platform.

Ideally, the overall performance of any single project should really be determined by examining three parameters:

- Productivity (or PDR)
- Speed (size/elapsed time) and
- Quality (measured via 'defect density', i.e. defects/size)

(Actual versus estimated project effort and time should also be considered, though these data concern other aspects of performance).

It is important to consider all three parameters because they may be traded and the trade-offs can be very significant. For example a project may be delivered faster than 'norm' by adding more staff to the team, resulting in lower productivity (higher PDR). The delivered software from a 'high-speed' project may also be of lower quality (= higher defect density) due to less time spent on defect removal and testing. When examining the PDR for any one project, it is therefore important to compare not only its PDR against the median for its category (i.e. the benchmark) but also to consider the project speed and quality in order to see if the project had experienced a significant trade-off of effort with time and/or quality.

Unfortunately, only a small proportion of projects have submitted any data about defects discovered during the first month of live running. So in this report it has not been possible to work out if quality has been traded for productivity or speed. However, most projects recorded their duration as well as effort, so it has been possible to establish benchmarks for both productivity (PDR) and speed. This should enable users of these data to establish whether or not an individual project has traded effort for time.

The quality of the data submitted appears to be generally good, though it should be mentioned that ISBSG has no way of checking the accuracy of the size and effort data that is submitted. In general, only projects with 'A' and 'B' quality data, as judged by ISBSG (on a four-value 'A' to 'D' scale), were analysed. However, a few probable anomalies were discovered and the data for such projects was not used. This elimination of outliers has to be done with care. On the one hand outlier projects may be particularly interesting to study. On the other hand, outlier project data can easily distort benchmark results, which is clearly undesirable when the benchmarks must be used for estimating – and the outliers may be due to errors in size measurement or effort recording.

After the main chapters 2 and 3, other chapters deal with subjects that may be useful for constructing project estimation models based on COSMIC sizing. Chapter 4 deals with the distribution of effort over project activities and how this distribution can influence project performance. Chapter 5 deals with an analysis of the contributions of the four data movement types to the reported COSMIC sizes.

[1] For most benchmarks, 'median' figures (and percentiles) are reported by the ISBSG rather than 'average' figures, since an average can be seriously affected by outlier projects. The impact of outliers is clearly undesirable for the purpose of establishing benchmarks. The median of a distribution of data values is the point in the distribution where 50% of the projects have higher values and 50% have lower values.

Chapter 6 deals with a comparison of the COSMIC benchmark results against some other data obtained using two other functional size measurement methods (IFPUG and MkII FPA). Finally, chapter 7 provides a summary of the main findings of this report.

Any feedback on the report from data-submitting organizations will be welcome, to ISBSG directly at: admin@isbsg.org (so as to maintain anonymity of the submitting organization).

The detailed raw data for the ISBSG projects analysed in this report are available from the ISBSG. A licence to use this data can be purchased from www.isbsg.org/products.

2. Business Application Software Projects

2.1 General: Overview of the data analysis

Data on 324 business application projects were reported of which 168 were from the banking industry. Other large industry categories represented were:

- Government, Education and Public Administration (47)
- Insurance (23)
- Manufacturing (19)
- Engineering (14)
- Medical & Healthcare (7)

The 324 projects comprised 162 new development projects, 152 enhancement projects and 10 re-development projects. These three sub-sets were analysed separately.

The new development and enhancement projects were further analysed by language level and type, and to investigate how performance varies with the size of the software delivered.

2.2 New Development Projects

2.2.1 Project demographics

The 162 new development projects with usable data ranged in size from 10 to 1958 CFP.

Effort ranged from 60 to over 46,000 work-hours (i.e. up to approximately 30 work-years), with a median of 3,289 work-hours (i.e. approximately 2.5 work-years). Duration ranged from 0.5 to 65 months, with a median of 8 months.

.

All projects involved bespoke developments, i.e. no projects involved any package customisation.

2.2.2 Project productivity (PDR); variation with language level, language, and platform

Figure 1 shows the effort (in work-hours) versus the size (in CFP) for 146 of these projects. This graph distinguishes projects that used a 3GL or a 4GL language.

Three projects were excluded from the 162 total because either the language is missing or it is impossible to determine if they used a 3GL or 4GL language. Thirteen more 3GL projects were excluded because they had an implausible PDR. Eleven of these projects were from one organization; they used Java on a PC platform, sizes ranging from 80 – 223 CFP, with very low PDR and all reporting a project duration of 3 months.

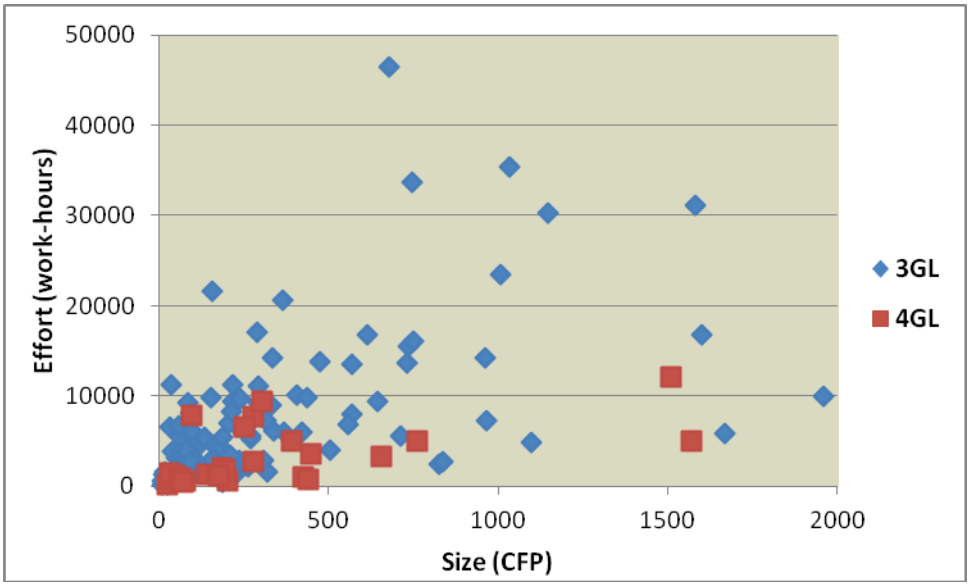

Figure 1: Business application new development projects: Effort vs Size

The benchmark PDR figures (WH/CFP) from this set (where P = 'percentile') are:

Language Level	N	Min	P10	P25	**Median**	P75	P90	Max
3GL	120	2.7	8.2	14.3	**24.5**	40.8	64.8	330.6
4GL	26	1.6	3.0	6.6	**9.2**	23.9	31.2	82.0

Table 1: Business application new development projects using 3GL and 4GL languages: distributions of PDR (WH/CFP)

Projects developed using a 4GL programming language are roughly 2.5 times more productive than those using a 3GL language. (However, read on for refinements to this ratio.)

Breaking down the data for the 3GL projects by language type and ignoring six projects that used miscellaneous languages, we obtain the following PDR benchmark data for the most commonly used languages.

Language Type	N	Min	P10	P25	**Median**	P75	P90	Max
All 'C'[2]	21	4	5	8	**14**	27	43	52
COBOL	46	8	12	18	**28**	55	110	330
Java/J2EE	42	3	11	158	**23**	38	61	139
Visual Basic	5	3	-	4	**23**	38	-	57

Table 2: Business application new development projects using 3GL languages: distributions of PDR (WH/CFP)

Table 2 shows that the median PDR figures for these languages appear to differ significantly, although they are all considered as 3GL languages. However, these median PDR figures by language might be influenced if the mix of projects-per-language varies with some other factors. First, we will examine the effect of hardware platform and later (in section 2.2.3) the effect of the delivered software size on these medians.

[2] All projects using C, C#, C#.Net and C++ languages.

Table 3 shows the PDR data for projects that used a 3GL programming language analysed by platform type (i.e. ignoring the distinction between 3GL languages).

	N	Min	P10	P25	**Median**	P75	P90	Max
Main-frame	60	8	13	18	**29**	51	101	330
Mid-range	7	10	-	14	36	38	-	65
Multi-platform	31	3	8	12	**21**	33	52	66
PC	20	3	4	8	**23**	35	55	139

Table 3: Business application new development projects: median PDR (WH/CFP) by platform

Ignoring the high PDR figure for the projects developed on a mid-range platform (obtained from just 7 projects using a variety of languages), the data in Table 3 indicate that projects developed on a main-frame are noticeably less productive (higher PDR) than those developed on multi-platform or PC hardware.

Having seen apparently important differences between the PDR figures when analysing the projects first by 3GL language and then by hardware platform, it then becomes interesting to analyse the same data by both parameters for the combinations where there are sufficient projects. Table 4 shows the results. (The 26 projects that used a 4GL programming language used eight principal language types, most commonly ASP, .Net, and Oracle, on all three main platforms. The numbers in each sub-group are too small to draw any conclusions about any variation of PDR with language type or platform.)

In Table 4, each entry shows three PDR figures – the 25 percentile / median / 75 percentile – as well as the number of projects in the group.

	Main-frame	**Multi-platform**	**PC**
All C	-	8 /**16**/ 27 (13 projects)	5 /**11/** 12 (5 projects)
COBOL	18 /**28**/ 55 (46 projects)	-	-
Java	21 /**33/** 48 (14 projects)	15 **/19/** 29 (15 projects)	19 **/25/** 37 (10 projects[3])

Table 4: Business application new development projects: median PDR (WH/CFP) by language and platform

The data in Table 4 now suggest that projects developed on a multi-platform are more productive (lower PDR) than those on a main-frame. However, the results for projects developed on a PC platform show a very wide range of productivity/PDR.

2.2.3 Project productivity: variation with delivered software size

Project PDR also varies with the size of the software delivered. Figure 2 shows how the median project **productivity** varies with delivered software size for various size bands, for all new 3GL and 4 GL development projects, excluding those projects already mentioned. (Productivity, expressed in CFP/work-month, has been calculated from PDR assuming an average of 120 work-hours per work-month. Productivity is preferred over PDR for this analysis as it is more natural to associate increasing productivity with 'improving performance'.)

[3] The 11 Java/PC projects from one organization that were excluded from the analysis have a median PDR of 4.1 WH/CFP. Including these projects would reduce the median PDR for the 21 Java/PC projects to 5 WH/CFP.

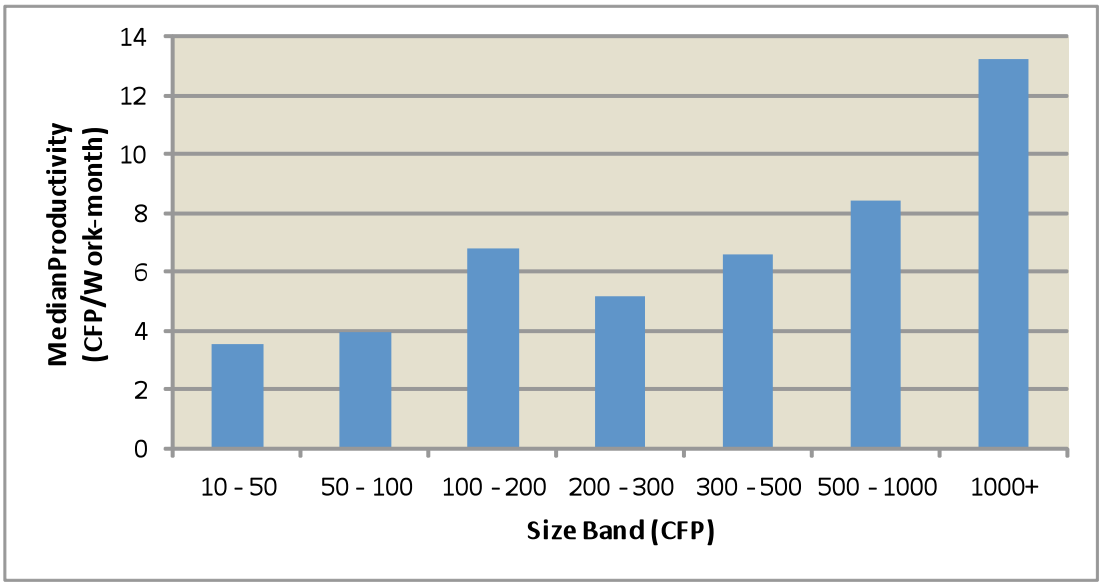

Figure 2: Business application new development projects: Median Productivity per Size Band

The result in Figure 2 is economically important in showing an economy of scale for new development projects of size up to over 1000 CFP. (For the numbers of projects in each band see Table 5.)

There is an economy of scale for new development projects as the size of software delivered increases, up to over 1000 CFP.

Further refining the data of Figure 2, but limiting to projects that used a 3GL programming language, Figure 3 appears to show that the spread of productivity widens, as size increases. Whilst this is true in absolute terms, the spread as measured by the P75/P25 ratio and the P90/P10 ratio for each size band is greatest for the 0-50 CFP band and does not change much as size increases. It is important to understand this graph for project estimating accuracy.

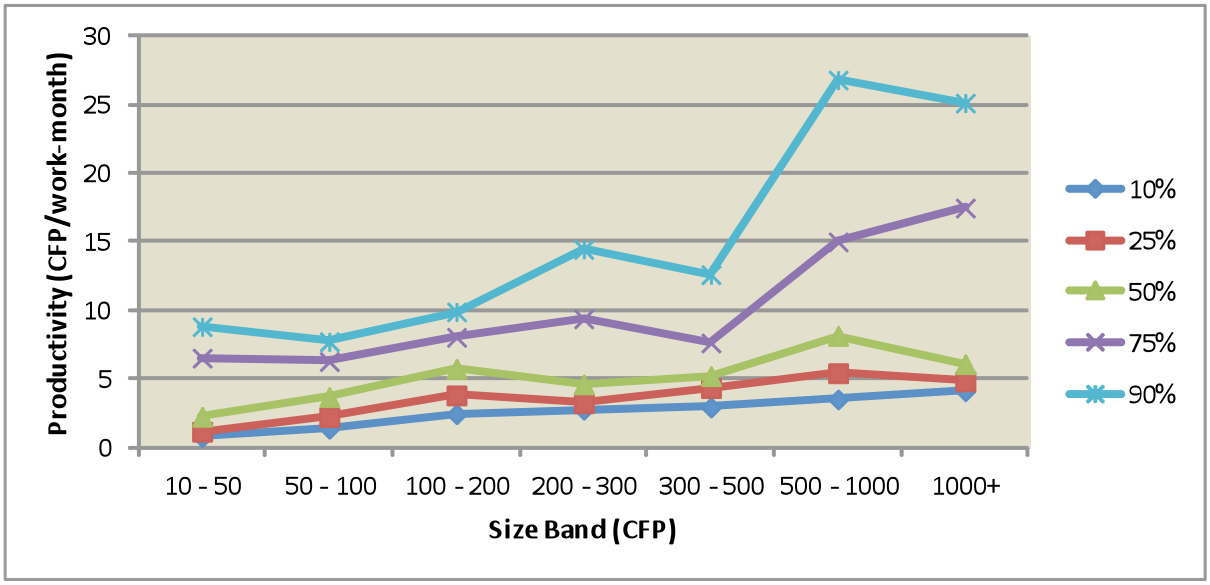

Figure 3: Business application new development 3GL projects: Percentiles of Productivity per Size Band

(In order to help judge the significance of these two results, the number of projects included in each size band is shown in Table 5 below.)

	Number of projects by Size Band (CFP)						
Size Band ->	10 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000+
All projects (Fig. 2)	21	34	21	28	22	18	10
3GL projects (Fig. 3)	16	30	16	22	12	16	7

Table 5: Numbers of projects in each size band of Figures 2 and 3

2.2.4 Conclusions on the drivers of PDR for new development business application projects

Given the findings so far on the drivers of PDR and the trends of the productivity versus size data in Figures 2 and 3, we can now explore if the mix of projects in each size band varies by hardware platform and language. Table 6 shows, for each size band:

- the most-used hardware platform, from Main-frame ('MF'), Multi-platform, and PC
- the most-used language, from 'All C', COBOL, Java, and 4GL languages.

Size Band ->	10 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000+
No. of projects[4]	19	28	19	23	15	16	9
Most-used platform	MF 58%	MF 75%	MF 53%	Multi 52%	Multi 40%	Multi 50%	PC 44%
Most-used language	COBOL 47%	COBOL 54%	COBOL 37%	Java 40%	None[5]	Java 44%	None*

Table 6: Business application new development projects: the most-used hardware platform and most-used language for each size band and the associated percentage of projects using that platform or language

In addition to the data of Table 6, the following are relevant observations on the platform and language mixes:

- Almost all projects using a C language (C++, C# or C#.Net) used only Multi-platform and PC hardware, and almost all occur only in the size bands over 200 CFP
- Projects using Java and Multi-platform hardware tend to dominate the size bands above 200 CFP
- Projects using a 4GL are spread roughly uniformly across all size bands; almost all these projects used Multi-platform and PC platforms
- Projects using a PC are also spread roughly uniformly across all size bands

Pulling these findings together with those of the data in Tables 2, 3, 4, and 6, we can conclude the following. (A proper statistical analysis is not possible due to the small number of projects in relation to three drivers of PDR, namely platform, language and size.)

a) The observed low productivity at small sizes of software delivered, (i.e. 0 – 50 and 50 - 100 CFP in Figure 2) is due in part to the projects in these size bands mostly using COBOL on a main-frame. Both these factors lead to low productivity

[4] The total of projects by size band in this table is slightly less than the number of projects used to produce Figure 2 since some projects included in Figure 2 did not report their hardware platform.

[5] 'None' indicates that none of the four language (groups) accounted for more than 33% of the projects in that size band.

b) Even if we correct for the effects noted in a), project productivity increases with the size of the software delivered as shown in Figure 2 by at least a factor of 2 over the range from the 0 – 50 CFP band to the 1000+ CFP band

Using these observations, we can tentatively suggest the following refinements to the results of Tables 2, 3 and 4 by interpolation, to eliminate the effect of the mix of sizes of software delivered by the projects in each group. ***These PDR figures are normalised for a size in the range of 100 – 200 CFP. The 'plus/minus' figures indicate the probable range from the 25 to the 75 percentiles.***

	Main-frame	Multi-platform	PC
All C	-	**17 ± 7**	**20 ± 10**
COBOL	**25 ± 9**	-	-
Java	**25 ± 9**	**20 ± 7**	**20 ± 10**
4GL	**-**	**9 ± 4**	**9 ± 4**

Table 7: Business application new development projects: Benchmark PDR (WH/CFP) by language and hardware platform, normalized for sizes in the range 100 – 200 CFP

2.2.5 How to use these data

We conclude from this analysis of the three factors considered[6] that *size* of software delivered has the greatest influence on PDR (or project productivity) followed, in descending order of influence, by *3GL versus 4GL language*, then *hardware platform*. Within these factors, the particular 3GL language has relatively little influence on PDR.

Hence grouping the project data by just language group (i.e. 3GL and 4GL in Table 1), or by a single 3GL language (in Table 2), or by hardware platform (in Table 3), or by size band (in Figure 2) produces 'average' figures that depend to some degree on the two other factors that are ignored in each grouping. Similarly, grouping the data by two of these parameters (e.g. by language and platform as in Table 4), gives averages that depend on the third parameter (size) that has been ignored in these grouping.

Lacking sufficient data to cut the data three ways in a statistically meaningful way, if a user of these data must take account of all three parameters, we suggest using the PDR figures in Table 7 as the starting point and then applying a correction for the effect of software size on PDR using the factors in Table 8.

Size Band ->	10 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000+
PDR Correction factor	1.2	1.1	1.0	0.9	0.8	0.65	0.5

Table 8: Factors to apply to the PDR data in Table 9 to correct for software size delivered

[6] Other factors, of course, influence PDR, notably the project 'team size'. However, the aim here is to develop benchmark data that can be used, for example in project performance comparisons and estimating. We therefore choose to focus on the principal factors that are given to a project team, i.e. size (from the requirements), programming language and hardware platform (from policies). The team size arises as a consequence of trading two 'givens' (cost and duration constraints) and a supply-side factor, namely staff availability. In project estimating, this trade-off choice is usually taken into account after considering the first three factors. For a discussion of how effort, duration and hence team size can be traded in estimating and of how actual trade-off can be taken into account in project performance analysis, see [9]. For the projects analysed in this report, average team size varies from 0.5 (one person, working half-time) up large team sizes (e.g. > 20)

Example use of correction factors. Suppose a project to develop some software of size 400 CFP, using Java on Multi-platform hardware. From Table 7, for projects producing software in the 100 - 200 size range using the Java/Multi-platform combination, the average PDR is 20 WH/CFP. From Table 8, the factor to correct the PDR for software sizes in the 300 – 500 range is 0.8. The PDR to use for this project taking into account all three factors (language, hardware and size) would be 20 x 0.8 = 16 WH/CFP. The effort is therefore most likely 16 x 400 = 6,400 work-hours, with a 50% probability that the effort will lie within the range of 4,200 to 8,600 work-hours.

2.2.6 Project speed

Roughly 30% of the new developments of business applications did not report their project duration. Figure 4 shows the duration, measured in elapsed months, versus size for 101 new development projects, distinguishing those that used a 3GL programming language from those that used a 4GL language.

Several projects were removed as outliers for this analysis in addition to those already mentioned in section 2.2.2 above as they would have had a distorting influence on the benchmark results which should reflect 'normal' project behaviour.

- A 3GL project of 65 months duration and a 4GL project of 35 months duration
- Four projects that had extraordinarily low or high average team sizes (two projects reported a duration of 44 months but with an average team size of 0.5, and two projects had an average team size of 15 but lasted only 0.5 and 0.9 months, respectively.)

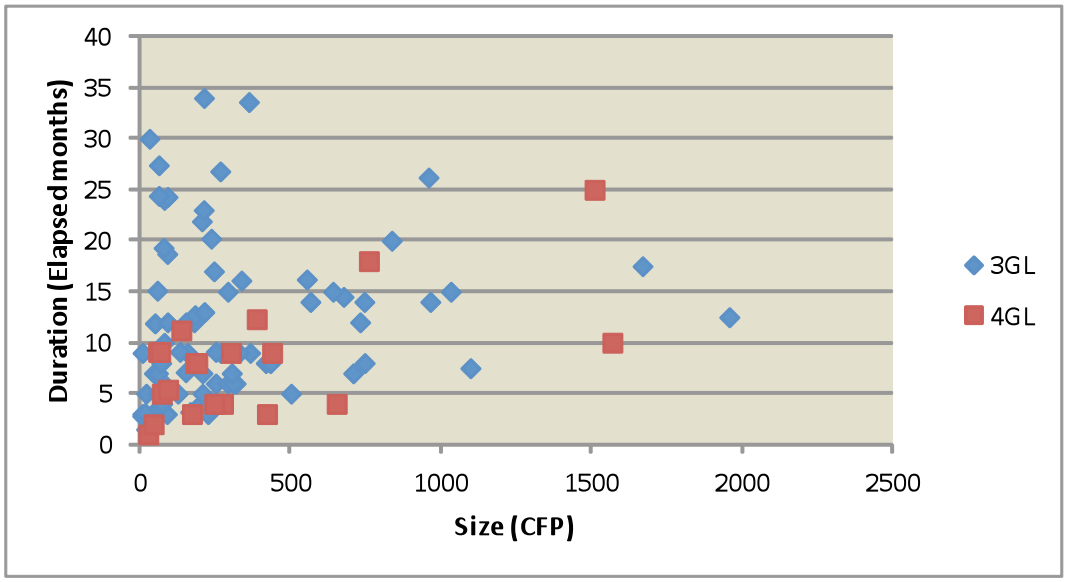

Figure 4: Business application new development projects: Duration vs Size

Figure 5 shows the median project **speed** (Size / elapsed months) for the 3GL and 4GL projects in the same size bands as those used for the median productivity in Figure 2. This chart shows that project speed increases sharply with size of the software delivered, i.e. there is an important economy of speed as well as of scale.

Project speed increases sharply with size of the software delivered, i.e. there is an important economy of speed as well as of scale.

The chart also indicates that projects using a 4GL language can be developed faster, for a given size, than those using a 3GL language – though the statistics are low for the 4GL projects.

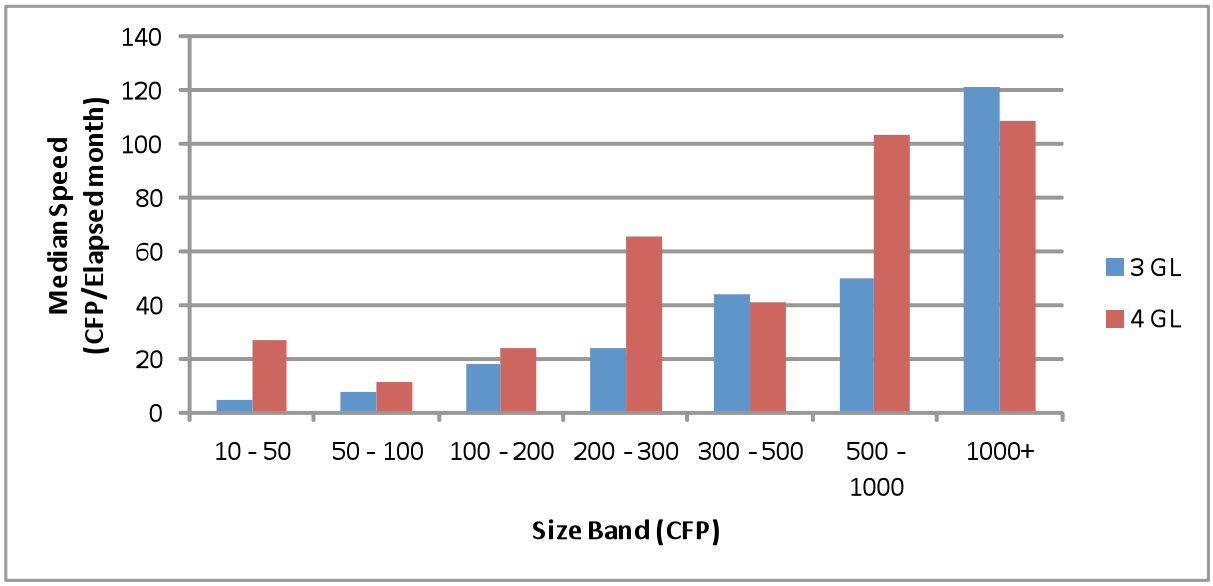

Figure 5: Business application new development projects: Median Project Speed vs Size Band

Projects that use a 4GL language can be developed faster, for a given size, as those that use a 3GL language, as well as roughly 2.5 times as productively.

(In order to help judge the significance of this result, the number of projects included in each size band is shown below.)

Size Band (CFP) ->	Number of projects by Size Band							Total Proj-ects
	10 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000 +	
3GL projects	9	18	9	16	9	12	4	77
4GL projects	2	4	2	2	4	2	2	20

Table 9: Numbers of projects by language group in the size bands of Figure 11

Figure 5 shows a quite smooth increase in speed for 3GL projects as the size of software delivered increases. The irregularity of the trend for 4GL projects is explicable due to the low numbers of projects and the variety of 4GL languages used.

Lacking sufficient data for a proper statistical analysis, and with the added complexity of the much faster-changing relationship between duration and size (compared with the effort/size relationship), we cannot separate the effect of the three performance drivers (platform, language and size) as was possible for the PDR relationship with size, as in section 2.2.3. We therefore present the graphs and relationship for various sub-sets of language versus size and hardware platform versus size, which should be useful for estimating.

First, fitting a power curve to the speed (CFP/elapsed month) versus the size (CFP) for all projects in each group of Figure 5 gives the following formulae:

3GL projects	Speed = 0.31 x (Size)$^{0.67}$	or, Months = 3.2 x (Size)$^{0.33}$
4GL projects	Speed = 1.39 x (Size)$^{0.60}$	or, Months = 0.72 x (Size)$^{0.40}$

Secondly, digging deeper into the data of Figures 4 and 5 we find that, for a given size, project duration varies noticeably with the choice of 3GL language. Figure 6 shows the duration, measured in elapsed months, versus size for 66 projects, distinguishing the three 3GL languages that were mainly used.

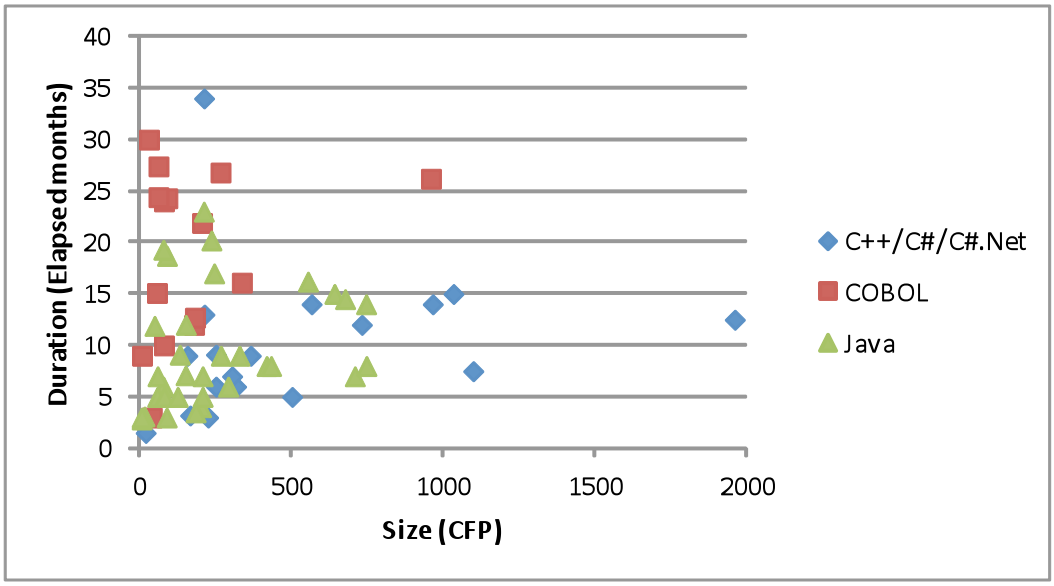

Figure 6: Business application new development projects: Duration versus Size for three programming language groups

Figure 6 shows that for a given size, COBOL projects tend to take longer than Java and the group of C++, C# or C#.Net projects. The power curves fitted to these three groups are as follows:

All C projects:	Months = 0.68 x (Size)$^{0.42}$
COBOL projects:	Months = 6.74 x (Size)$^{0.19}$
Java projects:	Months = 1.89 x (Size)$^{0.28}$

Note: Any power curve such as:

$$Months = A \times (Size)^{B}$$

can be converted to a power curve for project speed using the formula:

$$Speed = [(Size)^{(1-B)}]/ A,$$

where 'Speed' = Size / Months.

COBOL projects clearly take much longer, for a given size, than do projects using the Java or any of the other C languages (C++, C# or C#.Net).

Similarly, we find that, for a given size, project duration varies noticeably with the choice of hardware platform. Figure 7 shows the duration, measured in elapsed months, versus size for 68 projects that were developed with a 3GL language, distinguishing the three hardware platforms that were mainly used.

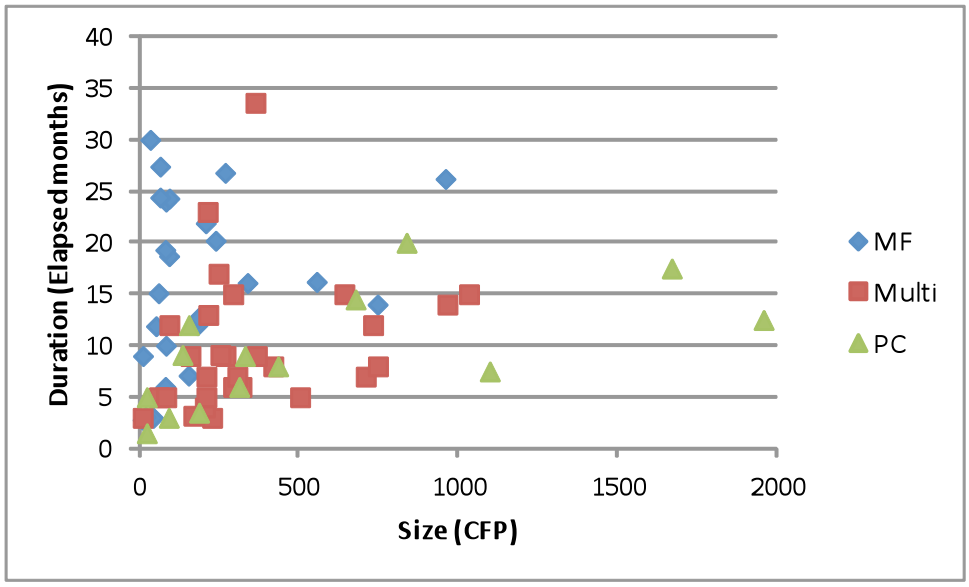

Figure 7: Business application new development projects: Duration versus Size for three hardware platforms

The power curves fitted to these three groups are as follows:

Main-frame (MF) projects: Months = 3.30 x (Size)$^{0.30}$
Multi-platform projects: Months = 1.65 x (Size)$^{0.28}$
PC projects: Months = 1.82 x (Size)$^{0.29}$

In conclusion, as was found for productivity, projects developed with COBOL on a main-frame are not only less productive but slower than the other language platform combinations, and this is true for any software size delivered.

Due to the large scatter of elapsed times for these projects, especially for smaller software sizes, any power curves fitted to the data for the different languages must be used with great caution.

2.2.7 Development of batch versus on-line software

A parameter that ISBSG data collection questionnaires have not captured is whether software to be developed should execute in batch or on-line mode, or a mixture of modes. Some data received privately by the authors from a company in the financial services industry (referred to as 'Finco' for this report) concerned 26 projects, of which 23 are new developments of software programmed using COBOL.

The data included the effort on Design, Build and Test activities, and on whether the software should operate in batch, on-line or mixed mode. Because the effort data does not cover the activities of Plan, Specify and Implement, the PDR figures quoted in this section should not be compared in absolute terms with those quoted above for COBOL new development projects. It is the relative PDR of projects that produced batch versus on-line, versus mixed mode software that is of interest.

Further, due to the small sample sizes the data were checked, and in part corrected, for the fact that the three groups had different average size of software developed. (The results of section 2.2.3 show that productivity of new development projects varies with software size produced; these results were used to provide the correction.)

The PDR figures for the 23 new development projects are as follows:

Software execution mode	No. of projects	Median PDR (WH/CFP)	Median size of projects (CFP)	PDR corrected for size differences	PDR relative to On-line PDR
Batch	15	30	72	30	**1.7**
On-line	4	18	78	18	**1.0**
Mixed	4	31	203	~40	**~2.2**

Table 10 PDR data for batch, on-line and mixed mode projects

The table shows that projects that must develop batch software require 1.7 times as much effort per CFP (or are 1.7 times less productive) than projects that must develop on-line software. Projects that must develop software operating in mixed mode are even less productive.

The explanation for these results might be that:

- the technology for developing on-line software delivers functionality, e.g. GUI interfaces, particularly easily compared with that used to develop batch software
- teams developing batch software systems must often negotiate interfaces with other systems, the effort of which lowers productivity.

Although the statistics behind this result are very limited, the finding that projects delivering batch software are less productive than projects delivering on-line software is consistent with that obtained when using the MkII FPA Method to measure software sizes, with much larger numbers of projects (see section 6.2).

2.3 Enhancement Projects

2.3.1 Project demographics

Data was reported on 152 projects to enhance business application software. The projects accepted for detailed analysis ranged in size from a project delivering 3 CFP, taking 24 work-hours effort, up to projects delivering 750 CFP and taking over 21,000 work-hours (i.e. approximately 15 work-years).

Two exceptionally large enhancement projects that delivered 1,250 and over 2,000 CFP have been excluded from the analysis as they are more like development projects than enhancements. (Their project productivity figures fit neatly into the range shown in Figure 3 for new development projects delivering over 1,000 CFP, using a 3GLlanguage.) Data in Chapter 6 indicates that roughly a quarter of enhancement projects involved only additions of functionality; three-quarters of these projects involved changes and deletions of functionality, as well as additions.

Also excluded from the general analysis are data for 14 enhancement projects that used the C++ programming language, and for 15 projects that used the ABAP language. These all had very low PDR (high productivity) figures, quite out of line with all the other data[7]. It is not clear from the data if these projects were really concerned with enhancing business applications, or possibly infrastructure software components.

[7] The 14 C++ enhancement projects had a median software size of108 CFP and a median PDR of 3.3 WH/CFP. The 15 ABAP projects had a median software size of 4 CFP and a median PDR of 9.3 WH/CFP. ABAP is a language used to enhance SAP systems.

2.3.2 Project productivity (PDR)

Figure 8 shows the Effort (in work-hours) versus the Size (in CFP) for the 121 enhancement projects for which the data are believed to be coherent.

Unlike the business application new development projects, the enhancement projects showed hardly any variation of PDR (work-hours/CFP) with the programming language used.

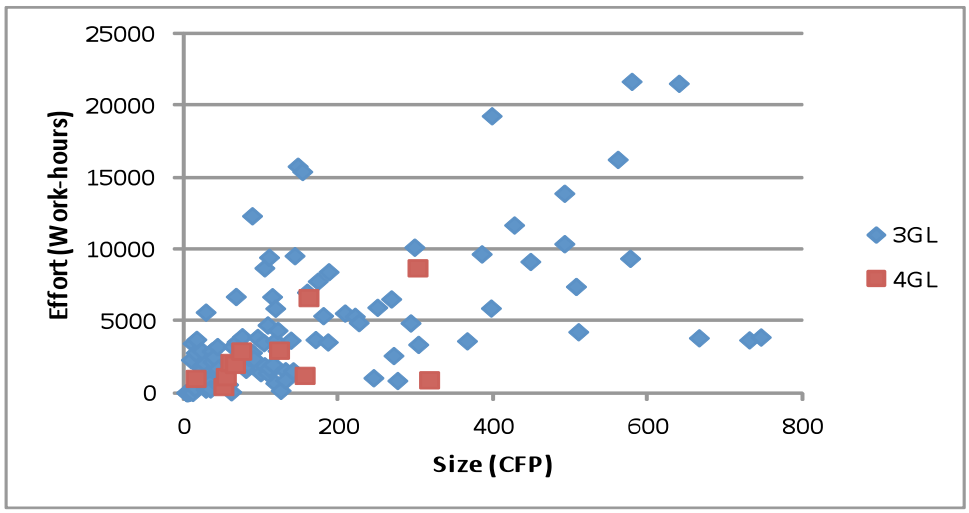

Figure 8: Business application enhancement projects: Effort vs Size

The benchmark PDR figures (WH/CFP) from this set (where P = 'percentile') are:

N	Min	P10	P25	**Median**	P75	P90	Max
121	1.5	7	13	**27**	45	81	315

Table 11: Benchmark PDR figures for business application enhancement projects

When the data of 116 projects are analysed by programming language for the principal languages used, the results are as in the following table.

	N	Min	P10	P25	**Median**	P75	P90	Max
COBOL	72	3	7	14	**27**	48	84	187
Java	24	5	7	14	**26**	55	188	326
VB & VC++	9	1.5	1.5	13	**21**	30	38	52
4GL	11	3	8	15	**29**	38	41	68

Table 12: Benchmark PDR figures for business application enhancement projects by programming language used

Analyzing the same data by hardware (the platform was known for 118 projects)

	N	Min	P10	P25	**Median**	P75	P90	Max
Main-frame	84	3	6	12	**27**	45	80	316
Multi-platform	9	17	-	22	**29**	38	-	49
PC	25	2	6	13	**30**	41	149	248

Table 13: Benchmark PDR figures for business application enhancement projects by hardware platform used

The PDR of the projects is quite consistent across the various languages, also considering the differences in hardware platform. Only the projects that used a 'Visual' language on a

multi- or PC platform have slightly lower PDR (= higher productivity) than the projects that used other 3GL and 4GL languages.

Enhancement project **productivity** also increases with the size of the software delivered.

Figure 9 shows how the median project **productivity** for all 121 enhancement projects varies with delivered software size, using the same size bands as for the new development projects in Figures 2 and 3. (Productivity, expressed in CFP/work-month, has been calculated from PDR assuming an average of 120 work-hours per work-month. Productivity is preferred over PDR for this analysis as it is more natural to associate increasing productivity with 'improving performance'.)

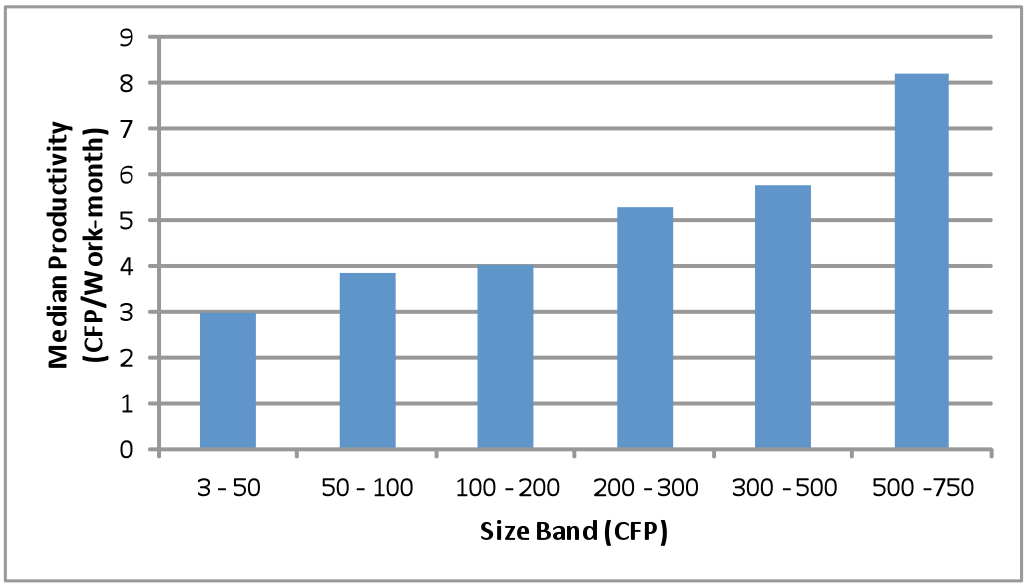

Figure 9: Business application enhancement projects: Median Productivity per Size Band

In order to help judge the significance of these two results, the number of projects included in each size band is shown below.

Size Band (CFP) ->	Number of projects by Size Band					
	3 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 750
All projects (Fig. 9)	36	24	31	10	11	9

Table 14: Numbers of Business Application Enhancement projects by size band

The data of Figure 9 show an almost identical increase in productivity over the same range of size bands for these enhancements projects, as for the new development projects shown in Figure 2.

Enhancement projects show an almost identical pattern of increasing productivity with size of the software enhancements up to 750 CFP, as for new development projects.

Figure 10 shows the percentiles of productivity for all enhancement projects (there is little variation of productivity with programming language or hardware platform used). Note that compared with Figure 9 for new development projects, the first 0 – 50 CFP size band is split into two bands (20 projects in the 3 – 25 CFP size band; 16 projects in the 25 – 50 CFP size band).

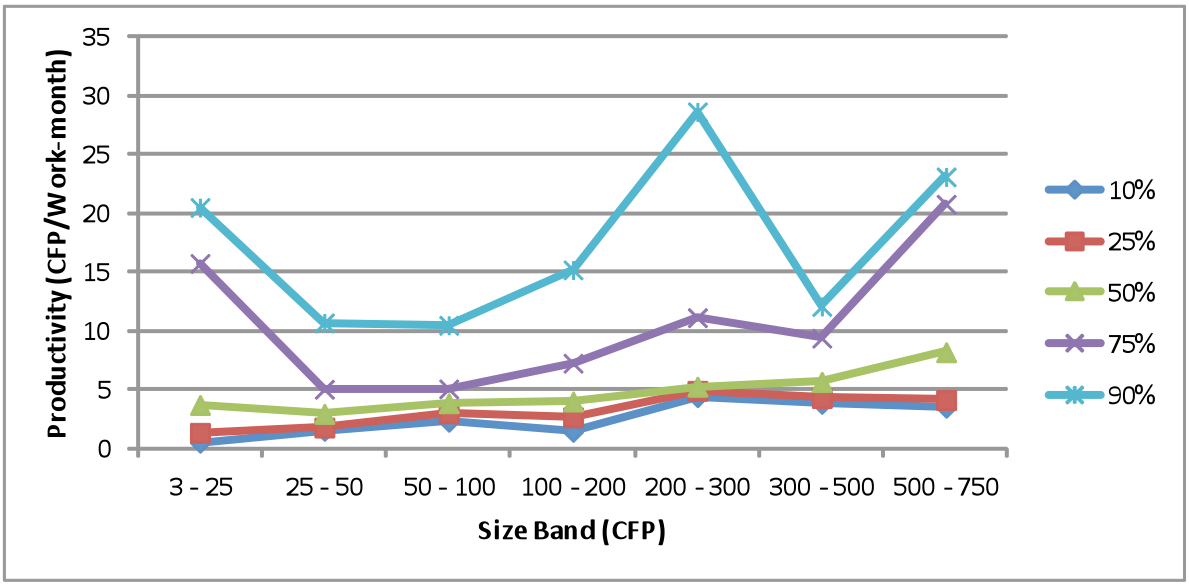

Figure 10: Business application 3GL enhancement projects: Percentiles of Productivity per Size Band

The spread of enhancement project productivity shown in Figure 10 (as measured by the ratio of the 75/25 and 90/10 percentile PDR figures) is significantly greater than for 3GL new development projects, shown in Figure 3. In percentage terms, by far the widest spread of productivity is for projects delivering the smallest size enhancements (3 – 25 CFP).

2.3.3 Project speed

Project duration was reported by 77 of the 121 projects to enhance business applications (64%). Figure 11 shows the duration measured in elapsed months versus size for 67 enhancement projects that used a 3GL programming language and 9 that used a 4GL language (one project reporting a 7-year duration was excluded).

There is no real difference between the speed of 3GL and 4GL enhancement projects (though there are few enhancement projects that used a 4GL language).

Unlike new development projects, there was no real difference between the speed of 3GL and 4GL enhancement projects for business applications.

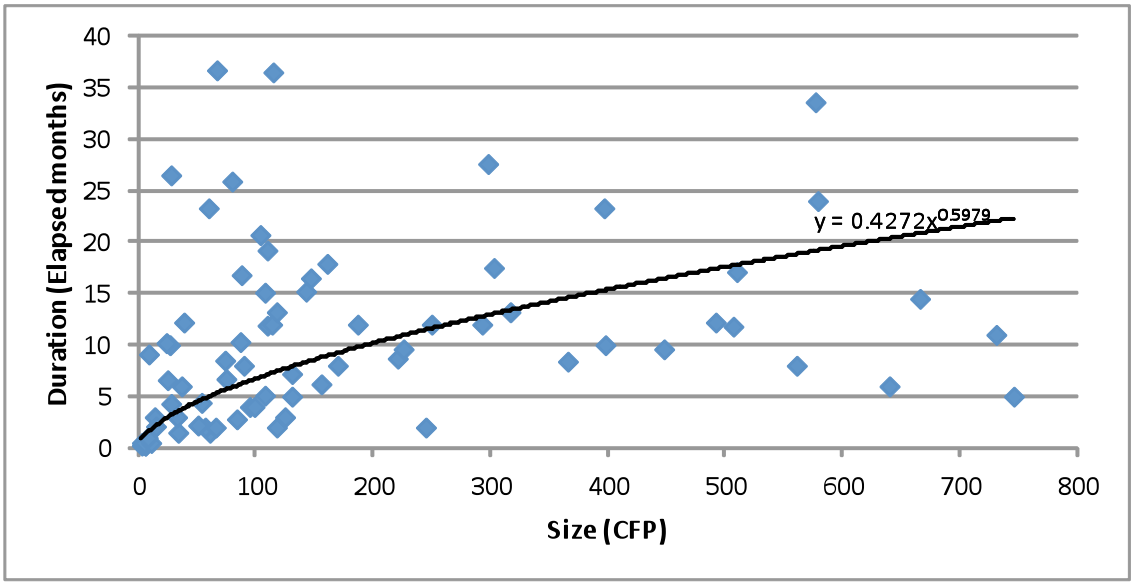

Figure 11: Business application enhancement projects: Duration versus Size

Figure 12 shows the median project **speed** (Size / elapsed months) for the 76 enhancement projects in the same size bands as those used for the median productivity in Figure 8. This chart shows that project speed increases sharply with size of the software delivered, i.e. there is an important economy of scale for project speed as well of productivity.

Enhancement project speed increases sharply with size of the software delivered, i.e. there is an important economy of speed as well of scale.

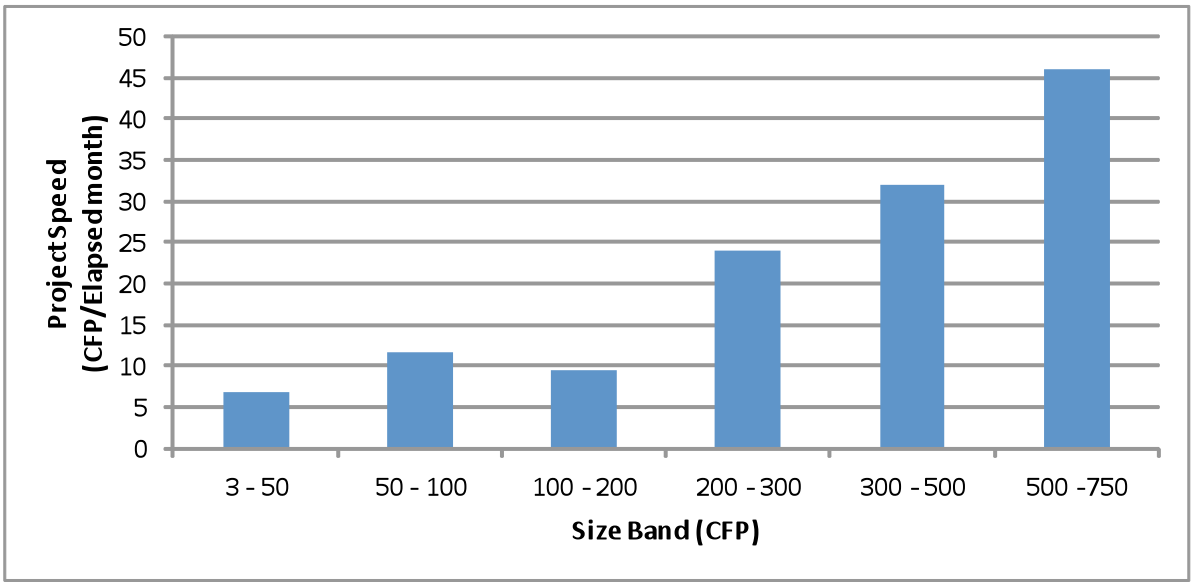

Figure 12: Business application enhancement projects: Median Speed per Size Band

This pattern of increasing speed versus size band for enhancement projects in Figure 12 is very similar to that for new development 3GL projects as in Figure 5.
Converting the power curve for duration versus size shown in Figure 11 to the speed (CFP/elapsed month) versus the size (CFP) for these enhancement projects gives the following formula, which should be useful for estimating.

Enhancement projects: Speed = 2.34 x $(Size)^{0.40}$. Months = 0.43 x $(Size)^{0.60}$

Note that a comparison of the speed of enhancement projects against the speed of new development projects, both using 3GL languages, indicates that enhancement projects are generally faster than new developments for sizes up to about 100 CFP. But above that size there is no real difference in speed, just as there is no real difference in productivity. These data indicate that the performance of such large enhancements is effectively the same as for new development projects.

2.4 Re-development projects

Data were submitted on 10 business application re-development projects that delivered software ranging in size from 55 to over 2000 CFP. One outlier project was excluded [8].

[8] The outlier project, which used VB on a PC, had a PDR eight times higher (i.e. lower productivity) than that of the median of the other nine projects, and was four times slower than would be expected from the other nine projects.

The other nine projects required from 675 work-hours up to over 10 work-years of effort and from 5 to 18 elapsed months. These projects were concerned with a variety of application types and their performance was quite variable. They were programmed in Java (4), C# (2) VB (1) and 4GL (2) languages, to execute on a PC (7) or multi-platform hardware (2).

For these nine projects the PDR figures (WH/ CFP) are as follows.

25 percentile = 4.5; **Median = 7.8**; 75 percentile = 13.9

The best-fit power curve for project duration versus size for these nine projects is

Duration (Months) = 0.97 x $(\text{Size})^{0.36}$

3. Real-time Application, Software Component and 'Miscellaneous' Software Projects

3.1 General: Overview of the data analysis

This chapter deals with the analysis of:

- 40 projects that delivered real-time application software, comprising 24 new development projects, 15 enhancement projects and 1 re-development project
- 22 projects that developed software components
- 7 projects that did not fit into any of the main categories.

The reader is warned that as there are relatively few projects, the statistical uncertainty on some of the benchmark findings is high. Furthermore, it was not always possible to be certain of the classification of a project as delivering 'real-time' software. However, in general the project data showed reasonable consistency and the results quoted here are ones that the authors judge as reasonable, and that should be of interest.

3.2 Real-time Application Software: New Development Projects

3.2.1 Project demographics

The 24 projects that were assumed to be included in this category recorded their 'application type' as:

- Telecom & network management (7)
- Mobile phone related (5)
- Software for machine control (process control) (3)
- Complex process control (2)
- Air traffic control (2)
- Weather observation systems (meteorological event detection) (2)
- Fault tolerance (1)
- Central command/control of sensors (1)
- Simulator (1)

Other demographic data are as follows (the numbers do not always add up to 24 due to unrecorded data).

Platform: Hand-held (1), Mid-range (6), multi-platform (3) and PC (8)
Language type: 3GL (21); 4GL (3).

The projects ranged in size from 8 to 1,384 CFP. The effort needed ranged from 81 to over 19,000 work-hours (i.e. up to 13 work-years); project duration varied from 2 to 34 elapsed months.

In addition to these projects, data on a large number of real-time new development and enhancement projects was submitted showing very much lower PDR (i.e. higher productivity) than all the other project data reported here. The PDR was so out of line with all other project data and there was so little other data on these projects that it was difficult interpret the results. These data have not, therefore, been analysed at present.

3.2.2 Project productivity (PDR)

A first examination of the data revealed four outlier projects which were removed from the analysis. Three projects reported very low number of hours spent compared to the functional size. Especially for a new development, this is not very likely and therefore the data quality is suspect. One other project reported effort and duration data that indicated an average team size of 30 over 5 months, which may be an extreme case of trading effort for speed.

Figure 13 below shows the Effort (in work-hours) versus the Size (in CFP) for the remaining 20 projects. They appear to be a relatively homogenous set. There was no detectable dependence of PDR on platform type or programming language level (3GL versus 4GL)

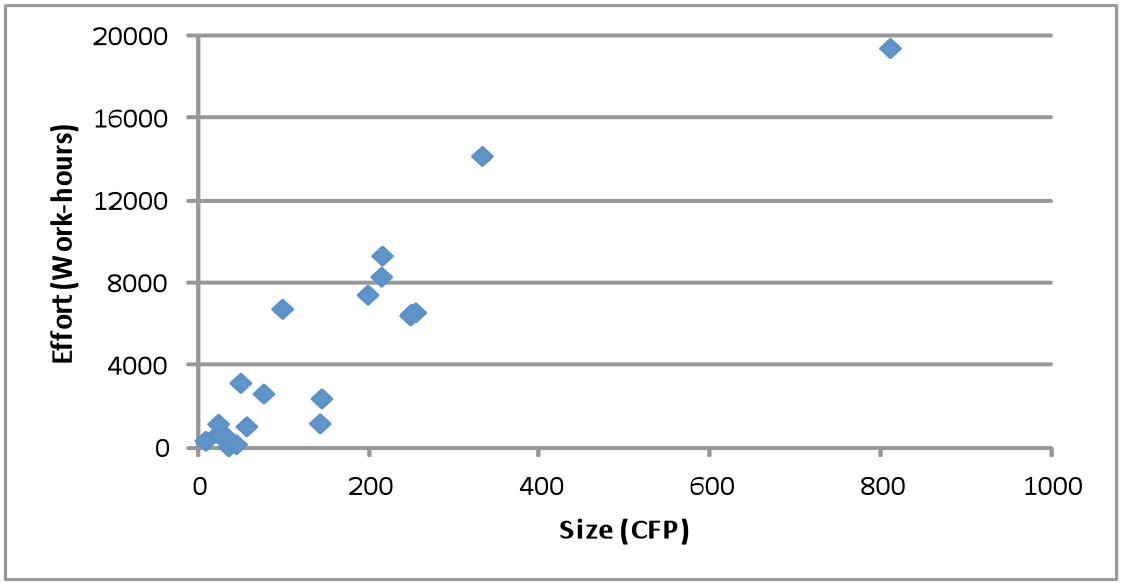

Figure 13: Real-time new development projects: Effort vs. Size

The benchmark PDR figures from this set (where P = 'percentile') are:

N	Min	P10	P25	**Median**	P75	P90	Max
20	2.3	8	18	**28**	43	53	69

Table 15: Real-time application new development projects: Distribution of PDR (WH/CFP)

3.2.3 Project duration

Fifteen (15) of the 20 projects reported the project duration in elapsed months.

Figure 14 shows the duration, measured in elapsed months, versus size for these projects.

The power curve fitted to these data has the form

$$\text{Duration (months)} = 1.53 \times (\text{Size})^{0.42}.$$

Note that this power curve is highly sensitive to the large software size project and the four other outliers (two very long and two very short duration). Ignoring these five projects, the curve for the remaining 10 projects becomes

$$\text{Duration (months)} = 1.35 \times (\text{Size})^{0.45}$$

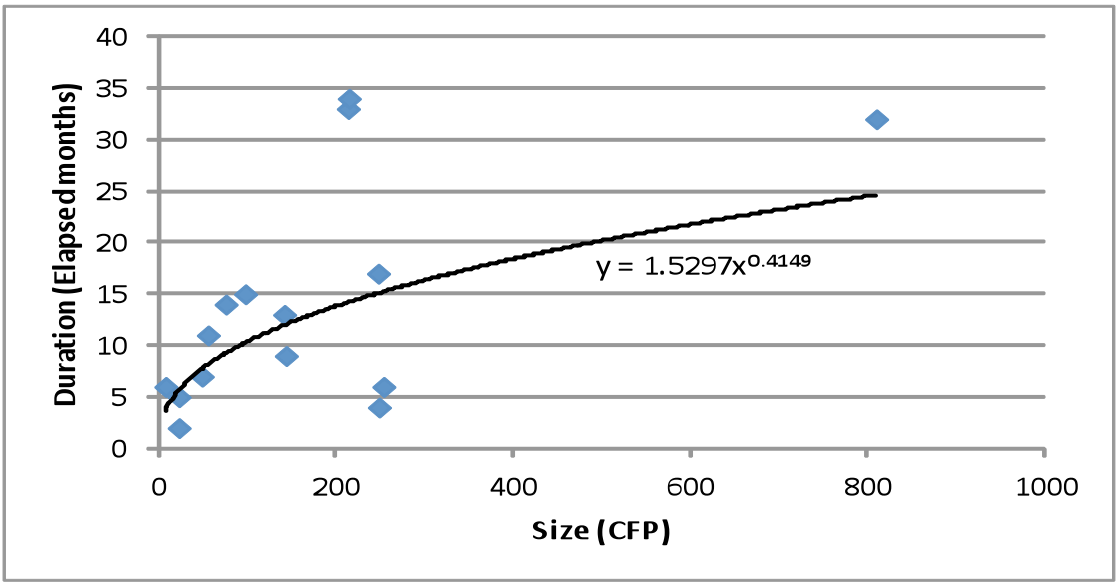

Figure 14: Real-time new development projects: Duration vs. Size

3.3 Real-time Application Software: Enhancement Projects

3.3.1 Project demographics

The data on 15 enhancement projects reported their 'application type' as:

- Telecom & network management (9)
- Device embedded (2)
- Software for machine control (2)
- Sensor control & presentation (1)
- Test equipment (1)

The projects ranged in size from 23 to 1,150 CFP. The effort needed ranged from 1088 to over 25,000 work-hours (i.e. up to 17 work-years); project duration varied from 4 to 17 elapsed months.

3.3.2 Project productivity (PDR)

Figure 15 shows the effort in work-hours versus size for 14 of these 15 enhancement projects. The project that delivered 1,150 CFP had a PDR of 3.3 WH/CFP, far lower (i.e. higher productivity) than any of the other projects, so was excluded from the analysis.

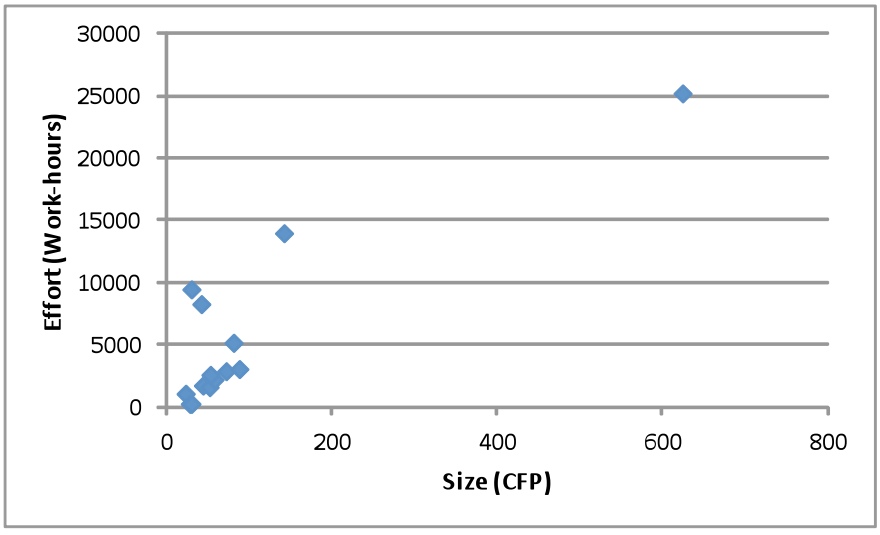

Figure 15: Real-time enhancement projects: Effort vs Size

For these 14 projects, the PDR figures are as in the table below.

N	Min	P10	P25	**Median**	P75	P90	Max
14	9	16	36	**40**	60	168	316

Table 16: Real-time application enhancement projects: distribution of PDR (WH/CFP)

The three projects that had a very much lower productivity were as follows:

- Two projects that enhanced 'software for machine control' delivered 42 and 30 CFP with a very high PDR of 316 and 198 WH/CFP respectively. They were programmed in C, on a mid-range platform.
- Another project delivered 142 CFP with a high PDR of 98 WH/CFP. This was also programmed in C, on a stand-alone PC.

Other than noting that these three projects with much lower productivity than the other 3GL languages all used the C language, the statistics are not such that it seems valid to separate these projects into sub-groups. The PDR figures also seem to be independent of the hardware platform.

3.3.3 Project speed

Figure 16 shows the duration measured in elapsed months versus size for 12 real-time enhancement projects that reported their duration, (again omitting the project delivering 1,150 CFP which reported a duration of 4 months).

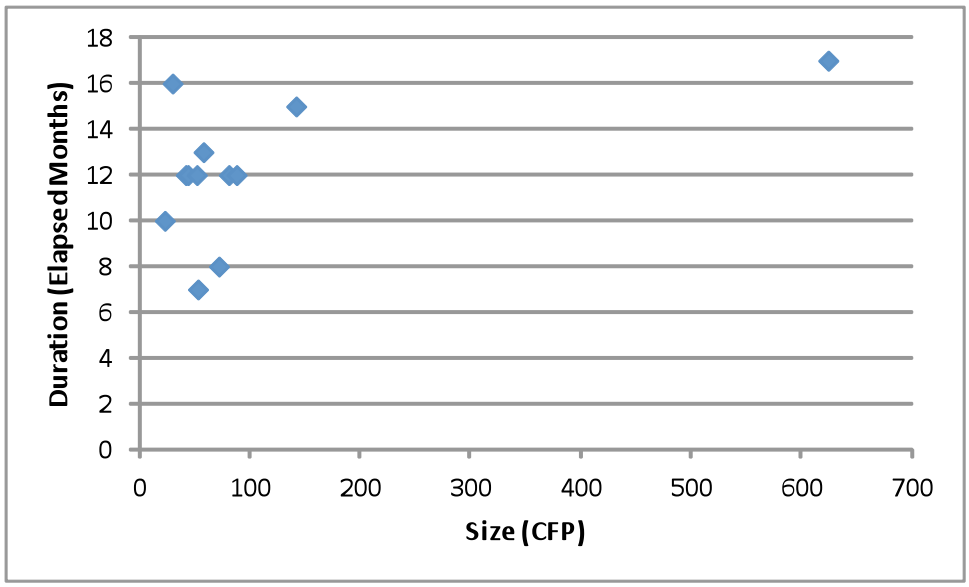

Figure 16: Real time enhancement projects: Duration vs Size

These projects have such large variations in duration for a given size that it is not valid to fit any trend curve.

3.4 Software Components: New Development Projects

3.4.1 Project demographics

Two sets of data fit into this category. They are:

- 15 projects developed in an Engineering R&D organization that delivered software described as 'Glue software' from a 'MIS Linguistic software system', The software was developed for a client-server architecture using Oracle technology on a PC platform
- 7 projects developed in an insurance company that delivered re-usable components for an operating system or utilities, using Java, on a PC platform

These were small projects. They ranged in size from 8 to 470 CFP, needing from 9 to 611 work-hours (i.e. up to about 5 work-months); project duration varied from 1 day to 2.7 elapsed months.

3.4.2 Project productivity (PDR)

Figure 17 shows the Effort (in work-hours) versus the Size (in CFP) for these 22 projects. It appears to be reasonable to treat these data as belonging to one set.

The benchmark PDR figures (WH/CFP) from this set (where P = 'percentile') are:

N	Min	P10	P25	**Median**	P75	P90	Max
22	0.7	0.7	1.0	**1.6**	2.9	6.8	8.8

Table 17 Software Component New development Projects: Distribution of PDR (WH/CFP)

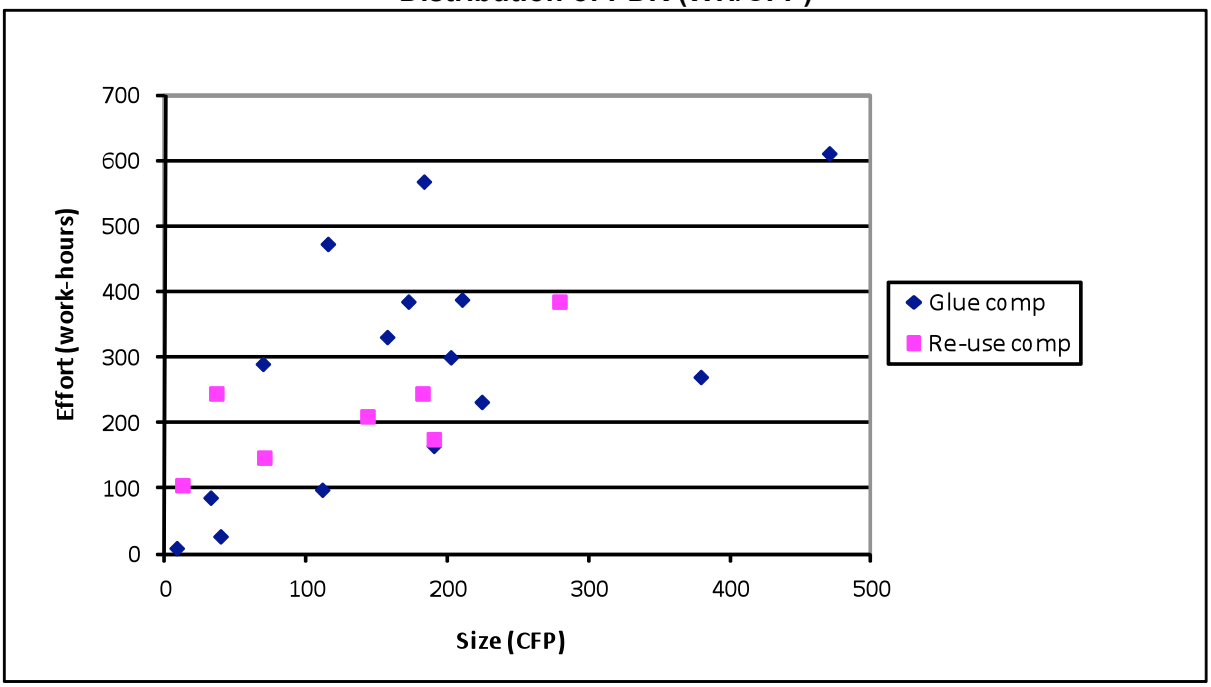

Figure 17: Software component new development projects: Effort vs Size

3.4.3 Project speed

Only the 'glue software' component projects reported credible project durations. Figure 18 shows the elapsed months versus size for these 15 projects.

These ‘glue software’ projects are all ‘small’ projects, requiring on average 2.4 work-months of effort and 1.3 elapsed months duration. All but four of the projects appear to have had a team size of two persons. Not surprisingly therefore, there is quite a scatter in productivity and speed of the projects.

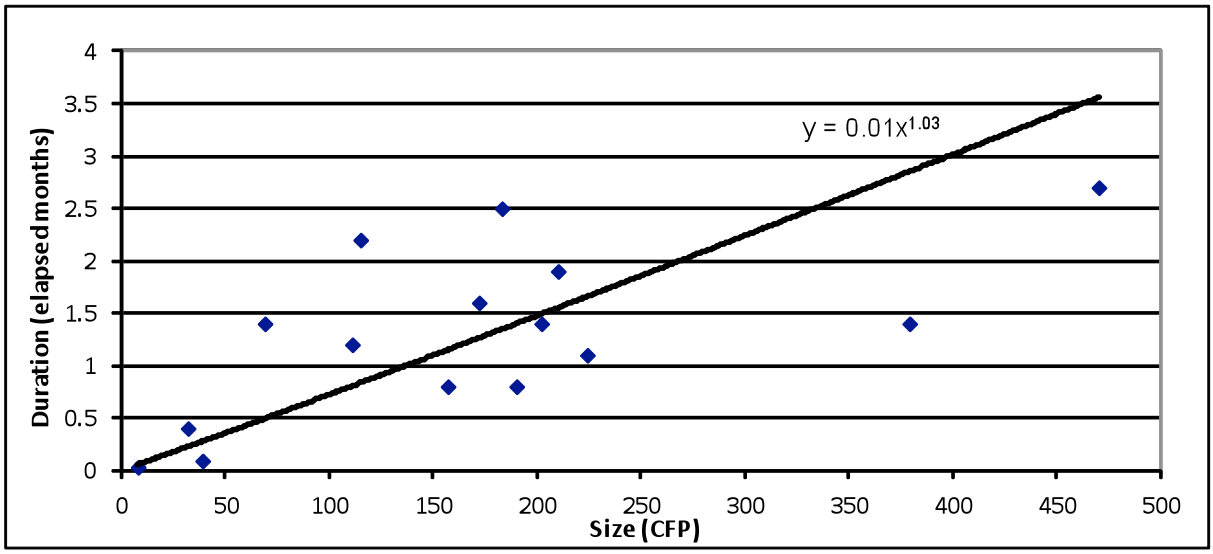

Figure 18: 'Glue' component new development projects: Duration vs Size

The median speed of these ‘glue’ software component projects in units of CFP/elapsed month is about 10 times greater than the median speed of the real-time application software projects for the same size of software delivered. Similarly, the median PDR of the ‘glue’ software component projects at 1.6 Work-hours per CFP is almost 20 times lower (i.e. higher productivity) than the median PDR of the real-time application software projects.

These results illustrate why it is imperative not to mix performance results of projects developing pieces of software at different levels of decomposition.

It is imperative not to mix performance results of projects developing pieces of software at different levels of decomposition.

3.5 Miscellaneous Projects

The data supplied by 7 projects did not allow them to be included as business application, real-time or software component projects. (Data reported in 2009 for two projects and one more recent ‘miscellaneous’ project has been discarded as implausible.)

Four of these projects, all new developments, have the following characteristics (in descending order of PDR):

Application description	Hardware	Language	Size (CFP)	Effort (WH)	Elapsed Months	PDR (WH/CFP)	Speed (CFP/mth)	Av team size
Register of events - OS utility	Multi-tier	3GL/Java	84	1600	5.0	19.0	17	2.7
Graphical modelling	PC Cl-Svr	4GL.Net	1511	12077	25.0	8.0	60	4.0
Graphics publishing tool/system	PC Cl-Svr	3GL/Java	79	408	3.0	5.2	26	1.1
Fault tolerance	PC SA	4GL/Vis Basic	35	82	?	2.3	?	?

(Note: ‘Cl-Svr’ = client-server; ‘SA’ = stand-alone.)

The first three of these four projects have similar performance to the real-time application projects discussed in section 3.2.

The 'fault tolerance' project may fit in the 'software components' category, although the reported elapsed time for this project is not credible.

The other three miscellaneous projects delivered mathematically-intensive software. Their characteristics are shown below.

Application description	Hardware	Language	Size (CFP)	Effort (WH)	Elapsed Months	PDR (WH/CFP)	Speed (CFP/mth)	Av team size
Algorithm + DB	PC SA	3GL Ada	484	40559	36.0	83.8	13.4	9.4
Geographic or spatial info system;	DE Cl-Svr	3GL C++	106	7111	13.6	67.1	7.8	4.4
Math modelling; Soft. Devt. Tool	PC-Multi	3GL C++	456	5408	9.0	11.9	50.7	5.0

The first two, projects, both new developments, have a very high PDR (low productivity) and a low speed compared with other projects of the same size. The third project is a re-development project so the PDR and speed may be plausible.

4. Project Effort Distribution

It is important and instructive for estimating to understand the normal distribution of effort over the various activities of a project. As discussed in Appendix A, the ISBSG aims to collect data on six activities, namely Planning, Specify, Design, Build, Test and Implement.

Almost all projects analysed in this report that recorded their development process used a waterfall process.

As noted in Appendix A, few projects submitted effort data for all six activities; most reported the effort for only 4 or 5 activities. This limits the possibility of producing effort distributions in the detail that is really needed, e.g. for some project estimating tasks.

4.1 Business application new development projects

Forty-five projects supplied data on all of the effort on Specify, Design, Build and Test activities. This enabled the calculation of the percentages of the total effort on each of these four activities. Effort spent on implementation activities was ignored as this varied enormously by project (See Appendix A).

Figure 19 shows these percentages for the 45 projects.

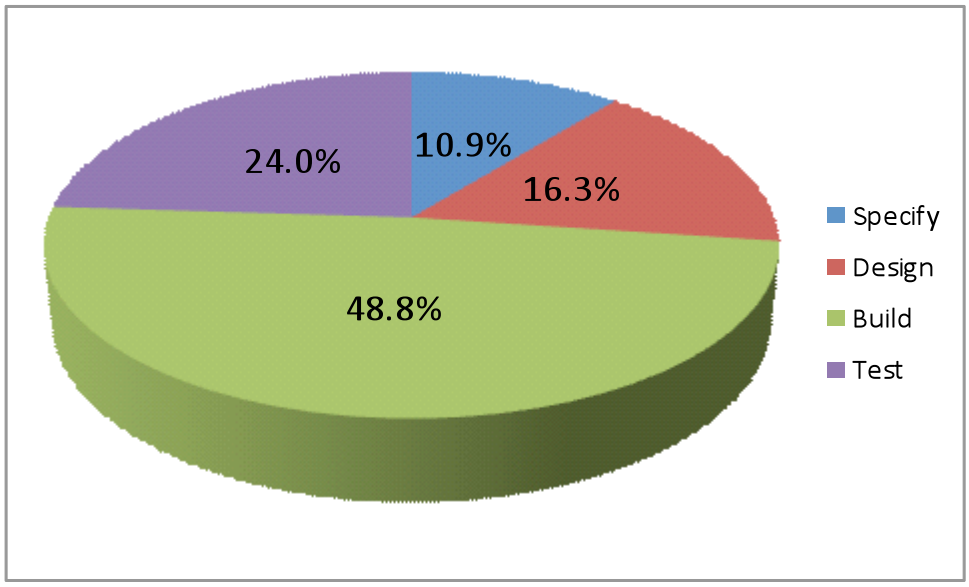

Figure 19: Business application new development projects: Percentage distribution of effort

Seventeen (17) of these 45 projects also supplied the effort on Planning activities. The percentage of effort on planning for these 17 projects was very varied, with two projects reporting over 33% of effort on planning, which may indicate an error in effort recording. Excluding these two projects, the average effort on planning for the 15 projects as a percentage of the five activities (planning ... test) was 4.7%.

The table shows the percentages of the four activities depending on whether COBOL, Java or a 4GL programming language was used. Note that the percentages vary considerably for the different projects and that the numbers from which these averages are calculated are low.

Programming Language	Number of Projects	Percentage of total effort by activity			
		Specify	Design	Build	Test
COBOL	16	9%	19%	39%	33%
Java	13	9%	16%	54%	20%
4GL	6	17%	12%	45%	26%

Table 18: Business Application new development projects: Distribution of project effort by activity for three programming languages

Also note that all of the COBOL projects and 6 of the 13 Java projects delivered software to run on a main-frame. The other 7 Java projects delivered software to run mostly on a PC platform, but the software of the 4GL projects runs on a variety of platforms including multiple platforms.

4.2 Business application enhancement projects

Thirty-five projects supplied data on all of the effort spent on Specify, Design, Build and Test activities. This enabled the calculation of the percentages of the total effort on each of these four activities. The data were analysed in the same way as for business application new development projects, and the same warning applies about the uncertainty on the percentages due to the low statistics.

Figure 20 shows these percentages for all 35 projects and the table shows the percentages depending on whether COBOL or Java programming language was used.

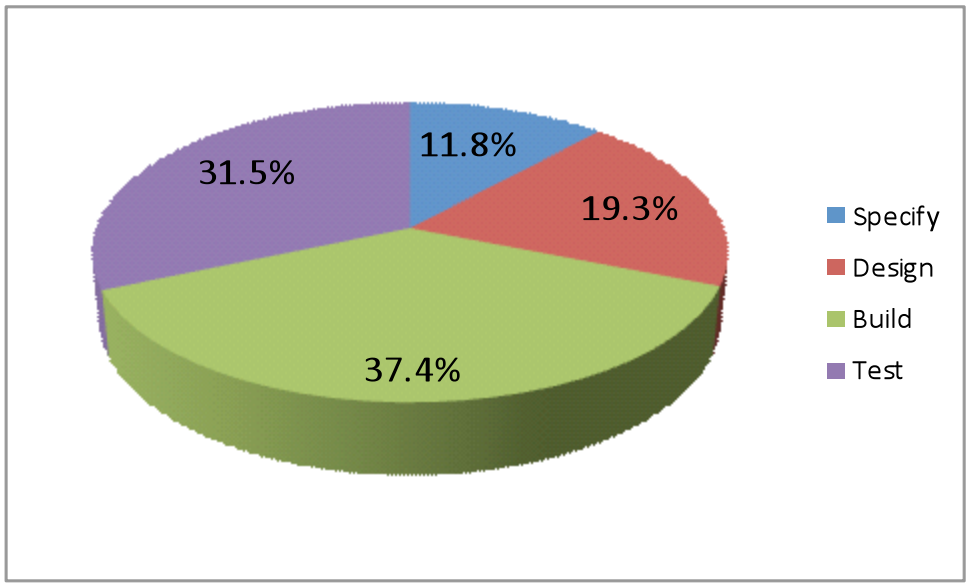

Figure 20: Business application enhancement projects: Percentage effort distribution

Programming Language	Number of Projects	Percentage of total effort by activity			
		Specify	Design	Build	Test
COBOL	22	10%	22%	33%	35%
Java	10	19%	17%	49%	25%

Table 19: Business Application enhancement projects: Distribution of effort by activity for two programming languages

The distributions in Tables 18 and 19 for each language data are consistent with the expectation (e.g. from other data published by ISBSG [3]) that the percentage of effort on

build activities is lower, and testing effort is higher, for enhancement projects compared with these percentages for new developments – though these data do not show a major effect, perhaps due to the low statistics.

4.3 Real-time application new development projects

Seven projects supplied some project effort distribution data. The only effort distribution percentages that can be derived from these data are shown in Figure 21 below.

The effort on specification and design as a percentage of total effort ranges from 5% to 37% for these seven projects. As the data varied greatly, the averages in the chart are not very reliable.

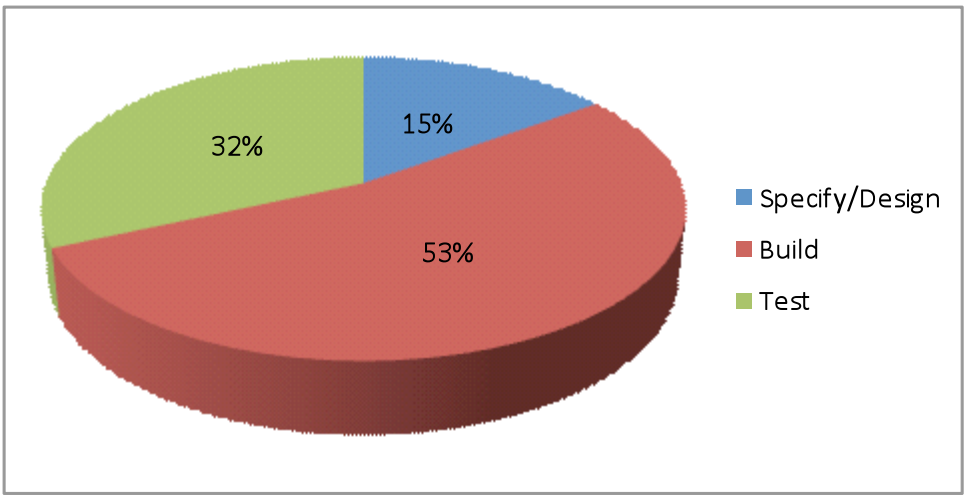

Figure 21: Real-time new development projects: Percentage distribution of effort

4.4 Real-time application new development projects

Four projects supplied project effort distribution data similar to that for the new development real-time projects. The effort distribution percentages derived from these data, which were very consistent across the four projects, are shown in Figure 22.

Although the numbers of projects are very low, the proportion of effort spent on testing for enhancements is much higher than that spent on testing for new development real-time projects, in line with expectations.

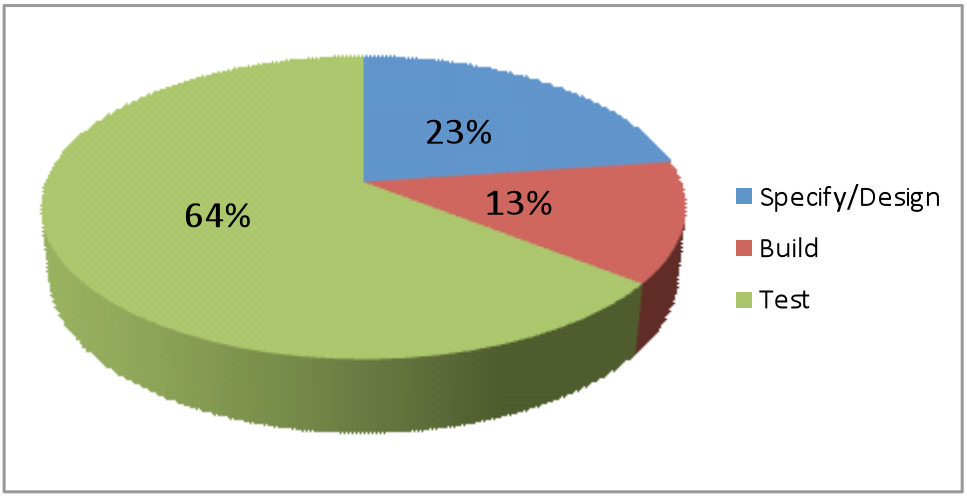

Figure 22: Real-time enhancement projects: Percentage distribution of effort

4.5 The effect of the effort distribution on project productivity

(This section has not been updated from the 2009 report.)

There is no 'correct' distribution of effort over the six activities, but evidence suggests that if the distribution of a project's effort differs significantly from the normal distribution, this is likely to be an indication that the project will experience lower productivity than if it had managed to keep close to the normal distribution.

Examples of such evidence are given in Figures 23 and 24 which show, for business application 3GL new development and enhancement projects respectively, project productivity plotted against the percentage of total project effort devoted to Specify + Design activities.

Only data from projects that reported the effort spent on both Specify and Design activities is included in these Figures. Implementation effort is excluded from the total project effort and from the productivity calculation because of its great variability. The calculation of productivity again assumes 120 work-hours per work-month.

Note certain outlier projects are not shown:
Figure 23 excludes data for two very high productivity projects with Percentage total effort on Specify & Design at 32.5% and 43.5%.

Figure 24 excludes data for three very high productivity projects, with Percentage total effort on Specify & Design at 16%, 24% and 28%.

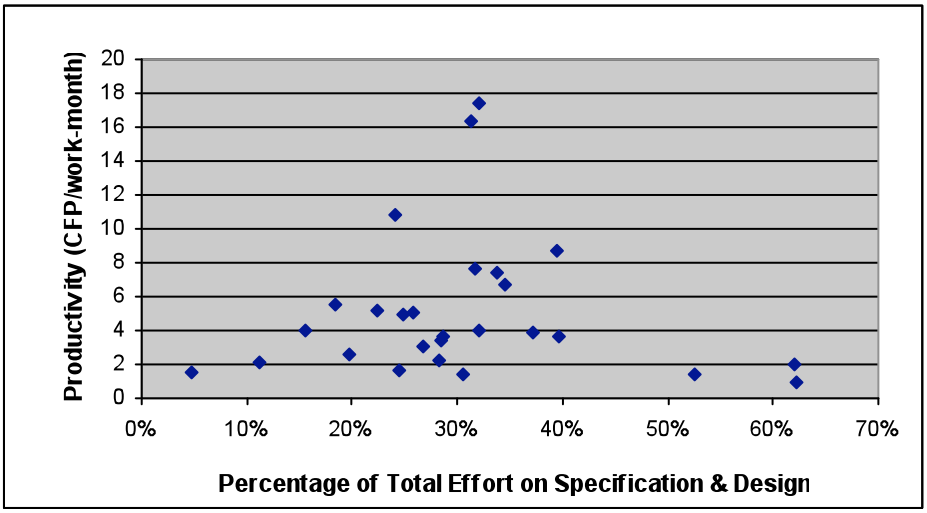

Figure 23 Business application new development 3GL projects

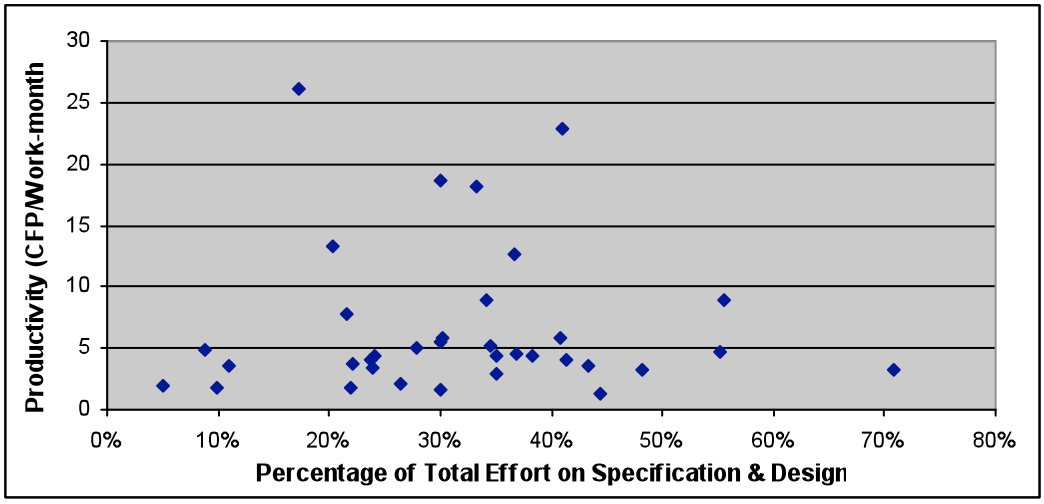

Figure 24 Business application enhancement 3GL projects

These graphs indicate that high productivity is obtained only when the percentage of total project effort devoted to Specify and Design activities lies in the range of roughly 20% - 40%. (The authors have obtained similar results with other, independent data.)

This data suggests that high productivity is obtained only when the percentage of total project effort devoted to Specify and Design activities lies in the range of roughly 20% - 40%

The inferences from these findings seem to be that if the percentage of total effort on Specify and Design:

- is too low (<20%), this is insufficient effort on these two activities and productivity will be low due to re-work later in the project
- is too high (>40%), this is an indication that the project has difficulties with agreeing the specification and/or design, and has got 'bogged down', causing lower productivity

(the statistics are too low, of course, to *prove* this interpretation).

5. Some COSMIC Functional Size Characteristics

The ISBSG data collection questionnaires for COSMIC-measured projects ask for data on the composition of the total software size for each project, namely the count of functional processes and the counts of Entry, Exit, Read and Write data movements for the delivered software.

Table 20 presents data from the 2009 report[9] on the average number of data movements per functional process. These 2009 data show this ratio for two sets of business application projects, namely 34 projects from the ISBSG database and the 23 Finco projects (see section 2.2.7), for (two) real-time projects and for 8 of the software component projects discussed in section 3.4. In addition the data is shown for two extremely large (>11,000 CFP) avionics sub-systems, from an organization referred to as Avco.

Software source & type	No. of projects	Av. no. of data movements per functional process
ISBSG – Business applications	34	7.6
Finco – Business applications	23	8.1
ISBSG – Real-time applications	2	6.7
Avco – Real-time avionics	2	11.2
ISBSG – Software components	8	2.7

Table 20: New Development projects:
Distribution of data movement types per functional process

Table 21 shows the proportions of the different types of data movements for new development and re-development projects of various types.

Software source & type	No. of projects	Average percentage distribution of data movement types			
		E's	X's	R's	W's
ISBSG – Business applications	54	25%	29%	30%	17%
Finco – Banking apps, mostly batch	23	21%	39%	28%	13%
ISBSG – Real-time applications[10]	12	24%	25%	29%	22%
ISBSG – Real-time embedded applications[11]	11	54%	27%	11%	11%
Avco – Real-time avionics	2	30%	16%	35%	10%
ISBSG – Software components	21	38%	40%	11%	11%

Table 21: New Development and Re-development projects:
Distribution of data movement types per functional process

The average patterns, with limited statistical validity in some cases, are:

- Business Application software has 55% of its functionality devoted to taking in existing data (Entries plus Reads) and 45% producing new data (Exits plus Writes)
- The 12 real-time application software have, *on average,* a similar pattern of distribution of data movement types as business applications, but the data on the 11 embedded software projects and for the two large avionics sub-systems are very different from the average
- Software reusable components have

[9] It has not been possible to update this data since the 2009 report
[10] From a variety of sources and applications (embedded, process control, telecoms, airport, etc)
[11] From a single source

a. the lowest number of data movements per functional process and
b. equal amounts of their functionality devoted to taking in existing data and to producing new data. However, 75% of their functionality is devoted to Entries and Exits with only 25% for Reads and Writes – unlike 'whole' application software systems.

Within these average patterns, there are noticeable differences in the pattern between different types of projects

- Some business application software may have its main task to take data in and make it persistent, thus having a relatively higher proportion of Entries and Writes, plus Reads for data validation. Some software may have as its main task to produce output and thus have a relatively high proportion of Reads and Exits. Software that must interface with many other systems will have a relatively high proportion of Entries and Exits.
- Although for the Finco set of business applications the average number of data movements per functional process is only 8.1 (and the median is 7), the distribution of functional process sizes is highly skewed, with a maximum size of 70 CFP. The Finco projects have a relatively high proportion of Exits, suggesting a high proportion of (batch) reporting software.
- The 11 real-time embedded software applications from the single source, of size ranging from 17 to 337 CFP, which all have the pattern of a high ratio of Entries to Exits and make little use of stored data. This pattern is similar to that for the software components.
- The two large Avco projects have a quite different pattern of data movement types and a very skewed distribution of functional process sizes (ranging up to over 100 CFP for a single functional process)
- For software components, the high proportions of Entries and Exits are to be expected given their need to interface with other components. However, there are wide fluctuations about the mean, dependent on the specialised task of the component. For example, the proportion of Write data movements in a software component varies from 0% to 47% across the 25 components analysed.

Overall, these data provide an important lesson for more-refined project estimating. The patterns vary with different types of software and different types of functional processes (e.g. for data entry versus data reporting) and may be associated with different productivity figures.

Collecting data on the characteristics of COSMIC functional sizes at the organizational portfolio level should be helpful for use in refining estimating methods to improve on using benchmarks based only on total functional sizes.

The distributions of data movement types for enhancement projects are of less interest because of the variety of ways in which a system can be enhanced. The following data are for 42 business application enhancement projects.

- Averaging over the 42 projects, the proportions of added, changed and deleted data movements were, respectively:- 56%, 41% and 3%.
- Only 11 of the 42 projects involved entirely new functionality, i.e. most enhancement projects involve adding and changing data movements.

6. Comparing COSMIC benchmark data versus those of other FSM methods

Users of ISBSG benchmark data who may be starting to use the COSMIC method may wish to know how the COSMIC benchmarks compare with those with which they are familiar. In this chapter we explore the general problem of benchmark data comparisons and provide a few examples.

There are a few important general points to consider when comparing benchmark data from projects measured using the COSMIC method against benchmarks from projects using other FSM methods.

- All FSM methods measure on a relative size scale, so comparisons of absolute benchmark figures for the same set of conditions are meaningless. Only comparisons of relative benchmarks are meaningful.
    - For example, comparing the ratio of the benchmark median PDR of business application new development 3GL to 4GL projects according to the COSMIC method (24.5 / 9.2 = 2.7 in section 2.2.2 of this report) versus the ratio according to a non-COSMIC method would be meaningful. But comparing the COSMIC-measured figure of 24.5 for 3GL projects against an IFPUG-measured figure of 11.4 [1] is meaningless.
- Non-COSMIC FSM methods for which benchmark data are available were all designed to measure business application projects, whereas the COSMIC method was designed to work for a wide range of software types. Comparisons of benchmarks will only be valid for software from the business application domain.
- The data used for the comparison of COSMIC versus IFPUG/NESMA-measured projects is gathered for slightly different time-periods. Comparisons of benchmarks may therefore be sensitive to evolution in project performance over the different time periods, e.g. due to using different programming languages, the need to produce web-based software, etc.
- The latest ISBSG benchmark data are based on more than 4,000 business application projects measured using the IFPUG/NESMA methods whereas the COSMIC benchmarks of this report are based on a very much smaller dataset. Users of this report should therefore take careful note of the numbers of projects used for the COSMIC benchmark data, since in some cases they are too small to be statistically very reliable.

6.1 Comparing COSMIC versus IFPUG benchmarks

There are other, specific points to consider when comparing COSMIC benchmarks against those measured with the IFPUG/NESMA FSM methods [4].

- Comparisons should be made only against IFPUG-measured 'Unadjusted Function Points', (UFP) which is what recent ISBSG benchmarks are based on.
- On average, it has been shown [6] that there is a reasonable linear correlation of sizes when the same piece of software is measured by the COSMIC and IFPUG methods. However, measurements of sizes of individual pieces of software according to the two methods can be scattered significantly about the average relationship. These variations will be mainly due to what follows.
- The 'Base Functional Components' (BFC's) of the IFPUG method, i.e. the three 'elementary process types' and the two 'file types', are all measured on a scale that has artificial lower and upper size limits. For example, an elementary process can have a size only from 3 to 7 UFP and a file type only from 5 to 15 UFP. The COSMIC

size scale for its BFC, the 'functional process', has no upper size limit, whilst 'file types' are not separately measured. Data in this report shows that the average size of a COSMIC-measured business application functional process is about 8 CFP, which is larger than the maximum possible size of an IFPUG-measured elementary process. Furthermore, COSMIC functional processes for business applications can range in size from 2 to 70 CFP. COSMIC size measurements are therefore more sensitive to these variations in the actual amount of functionality of the components of a piece of software than are the IFPUG size measurements. At the level of aggregating the sizes of all BFC's to a total functional size for a given piece of software, this difference in measurement approach of the two methods will sometimes be very important, sometimes less so.

- The size of an enhancement is measured quite differently in the two methods. The IFPUG method measures the size of the BFC's of the software that are changed (i.e. added, modified or deleted). The COSMIC method measures the size of the changes to the BFC's that are changed. The PDR for enhancement projects versus that for new development projects according to the two methods cannot therefore be properly compared.

Bearing in mind all these qualifications, a first comparison of interest is whether the IFPUG method shows the same trend for the relationship of project productivity versus size band that was obtained for COSMIC-measured projects in section 2.2.3, Figure 2.

For this comparison, data on 327 IFPUG-measured new development projects from the ISBSG Release 11 database were selected subject to the following criteria:

- Projects with data quality A or B, with size measured in Unadjusted FP by the IFPUG or NESMA methods (the latter being very similar to the IFPUG method) and using 'Normalised L1 effort data'
- The 'project date' was after the year 2000, from when COSMIC-measured data became available

Figure 25 shows the median productivity (in units of size per work-month, measured as above in section 2.2.3) for new development projects whose delivered software was measured by the two FSM methods, in the same size bands as in Figure 2. N.B. It is the trends of productivity with size that should be examined in Figure 25, not the absolute values. The COSMIC productivity values have been doubled from those in Figure 2 to make it easier to compare trends on this chart. (A comparison of the absolute figures is meaningless.)

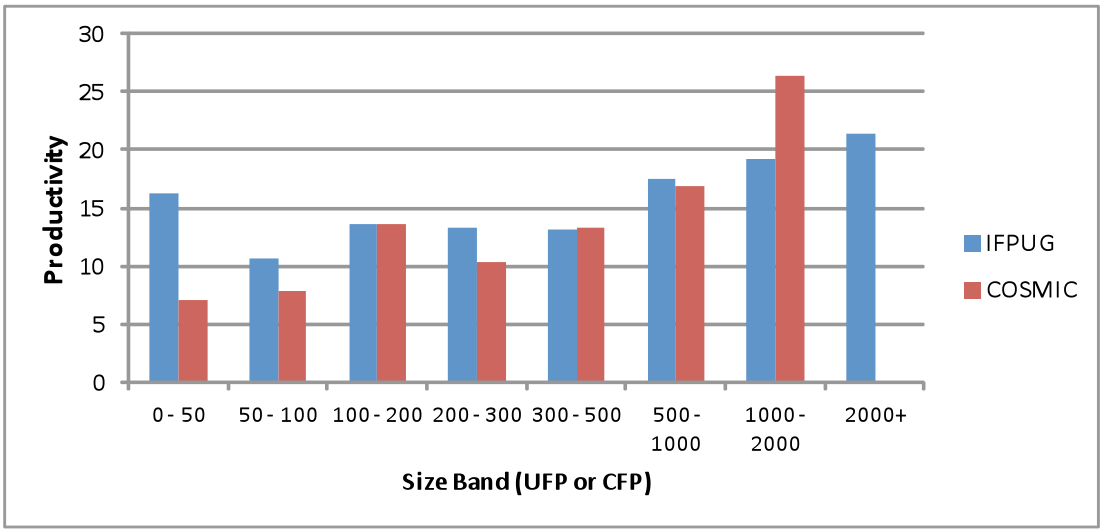

Figure 25: Comparison of the trend of median productivity per size band for business application new development projects, according to the COSMIC and IFPUG methods

This chart is quite difficult to interpret because even if the chart were based on the same projects being measured by both FSM methods, many individual projects would have appeared in different size bands according to the two methods (but this analysis used two entirely independent sets of data). Nevertheless, some observations are possible.

- Both methods show an increasing level of productivity over the size range from 50 to 1000 FP which covers 90% of all projects in the ISBSG database for both methods, which is an important economy-of-scale effect.
- The IFPUG result indicating higher productivity relative to the COSMIC result for very small software sizes is probably due to two factors: a) the IFPUG method's lower cut-off limits for the size of a BFC component and (b) the contribution of logical file BFC's (ILF and EIF) to IFPUG sizes.

In order to help judge the significance of the data in Figure 25, the number of projects in each size band for each FSM method, is given in Table 22.

Method	Number of projects in each size band							
	0 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000 – 2000	2000+
COSMIC	19	28	19	23	15	16	9	0
IFPUG	9	36	47	58	67	51	41	18

Table 22: Numbers of projects by sizing method in each size band for the productivity data in Figure 25

Figure 26 shows the median productivity of business application enhancement projects per size band for IFPUG-measured projects (from ISBSG Release 11 data) compared against the corresponding COSMIC data, for the same size bands as in Figure 25. (Again the COSMIC productivity values have been doubled, simply to make it easier to compare the trends.) The trend of increasing productivity with size is practically identical – this in spite of the two methods measuring different types of sizes for an enhancement project.

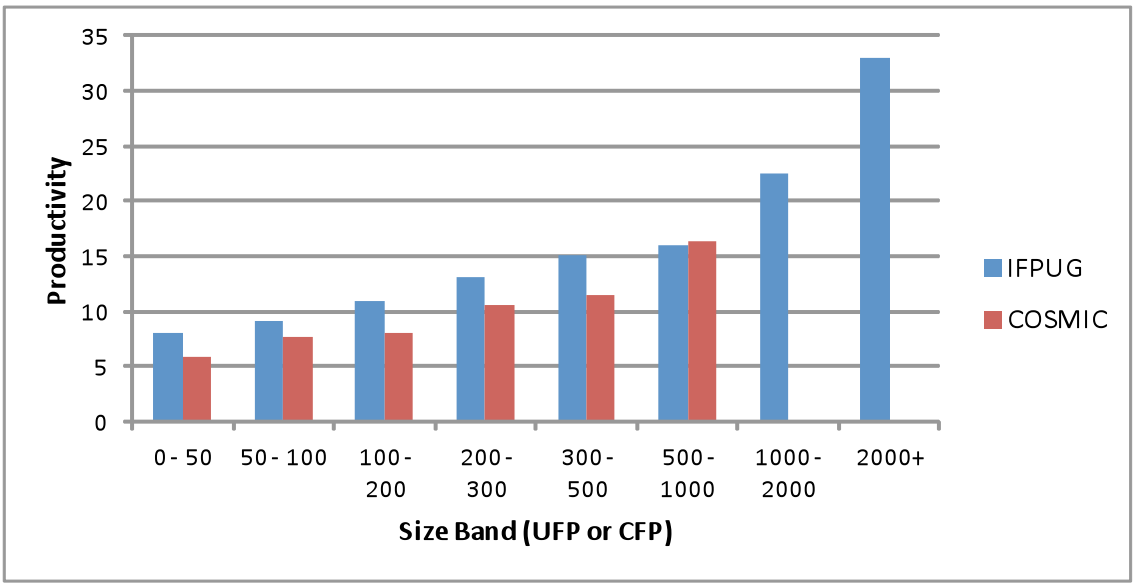

Figure 26: Comparison of the trend of median productivity per size band for business application enhancement projects according to the COSMIC and IFPUG methods

In order to help judge the significance of the data in Figure 26, the number of projects in each size band for each FSM method, is given in Table 23.

Method	Number of projects in each size band							
	0 – 50	50 – 100	100 – 200	200 – 300	300 – 500	500 – 1000	1000 – 2000	2000+
COSMIC	36	24	31	10	11	9	0	0
IFPUG	326	404	384	210	195	105	65	15

Table 23: Numbers of projects by sizing method in each size band for the productivity data in Figure 26

Understanding the relationship between software sizes as measured on the two methods is therefore both difficult and important. Given that both methods show an increase of productivity with size of software delivered, it is important to take into account differences in size mix when comparing any two benchmark figures on any one FSM method, or across methods. Standard ISBSG benchmark analyses sub-divide project data on only one parameter at a time (apart from splitting data across technical platform) and do not consider the effect of size mix on PDR.

Table 24 gives two comparisons of the ratio of median benchmark PDR figures for new development projects, measured on the two FSM methods, where the IFPUG data is taken from [3] for 'All projects', i.e. for projects irrespective of hardware platform and the COSMIC data are similarly uncorrected for the effects of the mix of project characteristics. N.B., a ratio > 1.0 indicates lower productivity.

Ratio of PDR Benchmarks	COSMIC		IFPUG (using UFP)	
	New Devts.	Enhance-ments	New Devts.	Enhance-ments
3GL projects / 4GL projects	2.7	1.0	1.4	1.4
COBOL projects / Java projects	1.2	1.0	2.1	2.2

Table 24: Comparison of IFPUG and COSMIC PDR benchmark ratios

It is not at all clear why these ratios apparently differ according to the two methods. (The numbers of COSMIC projects used in these two comparisons are sufficient that the explanation should not be due to limited statistics.) To explain the differences would require a much more detailed analysis of the data than time has permitted. The explanation might lie in the mixes of projects used for the data on the two methods being different due to:

- COSMIC data are for business application projects; IFPUG data is for 'all projects' which include in the order of 10% of projects concerned with real-time software
- the technology platforms used
- the mix of software sizes delivered
- the age of the projects (COSMIC data are mostly from the last 5 to 10 years, IFPUG data are for projects that used IFPUG version 4.0 or later method, going back 15 years

However, an overall conclusion from the findings of Figure 25 and Table 24 is that the COSMIC-measured project data seems to give more differentiated benchmarks for different environments than do the IFPUG-measured projects, even after allowing for the less differentiated analyses carried out by ISBSG on the IFPUG data.

6.2 Comparing COSMIC versus MkII FPA benchmarks

The MkII Function Point Analysis (FPA) method is very similar in design to the COSMIC method. Both methods define functional processes as their BFC, though the MkII FPA method uses other parameters than data movements to size the input, process and output components of each functional process. Like the COSMIC method, the MkII FPA method

has no upper limit to the size of a functional process. The MkII FPA approach to sizing enhancements is the same as for the COSMIC method, i.e. both methods measure the size of the changes to software required in an enhancement project. Given this similarity of FSM method design, one would expect greater similarity of benchmark findings between the COSMIC and MkII FPA methods than between the COSMIC and IFPUG methods.

The only MkII FPA benchmark data available to the authors dates from 1996 [7], based on measurements of over 600 business application projects, so the data is now only of historic interest. However, it is worth noting the following from a brief comparison with the COSMIC benchmark data in this report.

- The variation of productivity with the size of software delivered by new development projects is roughly the same when measured by both methods
- The ratio of the PDR for 3GL new development projects to that for 4GL projects on the MkII FPA method is about 3 over the whole size range (COSMIC = 2.7)
- The ratio of the PDR for 3GL enhancement projects to that for 3GL new development projects of the same size on the MkII FPA method is about 1.4 (COSMIC = 1.1)
- The ratio of the PDR for batch projects to on-line projects is about 1.7 for both methods

These comparisons are subject to several uncertainties but, on a range of parameters, the MkII FPA and COSMIC methods give similar benchmark ratios, which is to be expected given the similar design of the two measurement methods.

7. Summary Findings and Conclusions

The ISBSG repository of data for software new development and enhancement projects now contains close to 440 projects that have been sized using the COSMIC method and the number submitted to the ISBSG continues to grow steadily. The number of projects is now adequate to publish meaningful benchmark data on several parameters, though the reader is cautioned in several places in this report that some of the benchmarks are based on very small numbers of projects.

Some of the key findings from this study are as follows.

a) Figure 27 shows the median PDR figures for five main project categories.

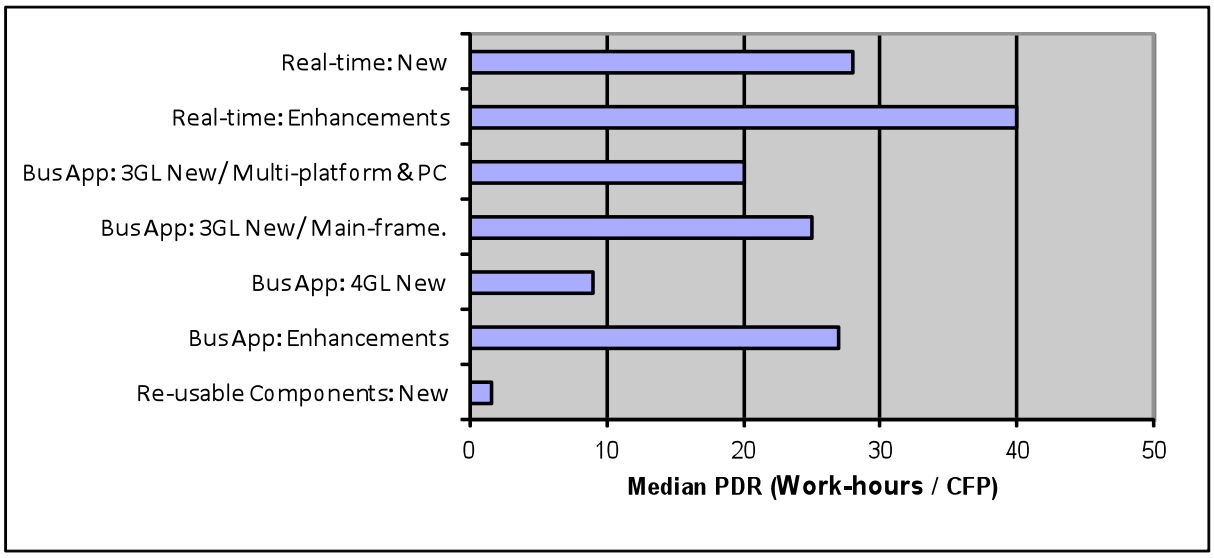

Figure 27: Key PDR benchmarks

These results are unsurprising:

- Real-time projects require more effort per CFP than do business application projects
- Enhancement projects need more effort per CFP than new development projects
- Projects developed using a 4GL require much less effort than those using a 3GL

b) The PDR for developing software reusable components is typically at least an order of magnitude lower (i.e. higher productivity) than for developing whole application systems. This means it is imperative not to mix measurements of the output of projects that deliver whole software systems and those that deliver small components. (The size of a 'whole' software system cannot be obtained by adding up the size of its components – see the rules of the COSMIC method).

c) Median productivity for business application new development projects increases with size of the delivered software, more than doubling for projects delivering in the 500 – 1000 CFP range compared with projects delivering less than 50 CFP.

d) Median productivity for business application enhancement projects increases with the size of the delivered enhancement, more than doubling for projects delivering enhancements in the 500 – 750 CFP range compared with projects delivering less than 50 CFP.

e) The speed of projects (measured as CFP per elapsed month) increases sharply with increasing size of software delivered for all categories of projects. Power curves fitted to the data, of the form *Duration = C x (Effort)^n*, have an exponent 'n' for business application new development projects in the range 0.2 – 0.4 and for enhancement projects of ~0.6. The finding for new development projects is consistent with results reported by other authors, but note the higher value of 'n' for enhancement projects, probably due to the effort required for regression testing.

f) The increasing project productivity and speed with increasing size of software delivered show a clear economy-of-scale though, of course, this benefit must be offset by the increasing risk associated with larger projects. This increasing risk is well illustrated by the data of Figures 3 and 10 which both show an increasing spread of productivity in absolute terms with increasing size of software delivered. However, the biggest percentage spread of productivity is for small projects of less than 50 CFP (or 25 CFP in the case of enhancement projects). This phenomenon may indicate some limitations of using functional size methods as the starting point for estimating very small software projects

g) Comparing Figures 3 and 10 also shows that the scatter of productivity about the median is much broader for enhancement projects than for new development projects over the whole size range. This is at least partly explained by the varying nature of enhancements that must be carried out.

h) For new developments, the median productivity of projects using a 4GL language is more than twice as great as for projects using a 3GL language (when delivering the same size of software) and projects using a 4GL are delivered faster. These gains on both productivity and speed from using a 4GL versus a 3GL for new developments are very important. However, projects to enhance software that was developed using a 4GL are no more productive than projects to enhance 3GL software.

i) New development projects that must deliver software to run in batch mode require about 1.7 times as much effort per CFP as do projects that must deliver software to run in on-line mode. The COSMIC-measured statistics are weak on this result, but it is noteworthy that this same result was observed many years ago on projects where the software size was measured using MkII FPA.

j) The effort distribution over project activities varies with the type of project. Comparing the distributions of effort of Figures 19 to 22 for business and real-time projects (and taking into account the more detailed analyses), we see that enhancement projects spend proportionately more effort on testing and less on build activities than new development projects. The higher proportion of effort on testing, usually including regression testing, for a given amount of changed or added functionality, may largely explain why enhancement projects are less productive than new development projects.

k) The effort distribution data suggest that spending relatively too little (< 20%) or too high (> 40%) a proportion of total project effort on specify and design activities will result in lower productivity. It is therefore important to take into account these norm distributions of effort when planning a project.

l) The report includes average ratios of numbers of data movements per functional process, and distributions of the average percentages of Entry, Exit, Read and Write data movements for different types of software. These data can be useful for checking the credibility of measurements of functional size using the COSMIC method for these various software types and can be used to help refine project estimation methods.

m) The report discusses the general problems of comparing benchmark data obtained from projects where the software size is measured using the COSMIC method versus other functional size measurement methods. Some specific problems are identified on comparing COSMIC against IFPUG benchmarks as published by the ISBSG. Some examples of such comparisons show similar trends (e.g. increasing productivity with size of software delivered) but others suggest that the COSMIC method gives more differentiated benchmark figures for differing sets of project characteristics. However, some of this apparent difference may be due to the way that the ISBSG analyses its project data, which mostly cut the data on only one parameter at a time. (Example ISBSG produces benchmark PDR figures for new development and for enhancement projects and then, separately, for projects using a 3GL or 4GL programming language. In this report we have attempted to derive PDR figures for all four combinations of these two parameters where there are sufficient data).

Overall, the COSMIC benchmark data reported here, after excluding the outliers, seems to be nicely self-consistent in spite of the limited statistics in some areas. The data should be valuable for organizations wishing to make external performance comparisons and for new project estimating.

All COSMIC method users would benefit from having more project data to enable more detailed and precise benchmark analyses. Users of the COSMIC method are therefore strongly urged to submit their project data to the ISBSG database so that the benchmark data can be improved and extended.

References

[1] 'Practical Software Project Estimation: a toolkit for estimating software development effort and duration', the International Software Benchmarking Standards Group, McGaw Hill, 2011, ISBN 978-0-07-171791-5, obtainable via www.isbsg.org

[2] 'The COSMIC Functional Size Measurement Method v3.0: Method Overview, September 2007, obtainable for free download from www.cosmicon.com

[3] 'The Software Metrics Compendium', Release 10, the ISBSG

[4] 'The Function Point Counting Practices Manual', Release 4.2, www.ifpug.org

[5] 'MkII Function Point Analysis Counting Practices Manual', Version 1.3.1, www.uksma.co.uk

[6] 'The COSMIC Functional Size Measurement Method v3.0: Advanced and Related Topics', December 2007, obtainable for free download from www.cosmicon.com

[7] 'What is World-Class IT Performance?', Hassan, D.J., UKSMA Conference, April 1996

[8] 'An Analysis of Software Projects using COSMIC Full Function Points', ISBSG, 2004

[9] 'A Process to explore the Software Project Effort / Duration Trade-off Relationship', Symons, C., IEEE Software (awaiting publication: obtainable via IEEE Xplore, early access)

Appendix A An Explanation of the effort data used in this report

The effort values used in this report for all PDR benchmark figures are what the ISBSG refers to as 'Normalized Level 1 effort'. 'Level 1' means that the effort is only that of software developers and does not include the effort of other staff who may work on the project but that are not formally assigned to it. Examples of such 'other' staff could include specialists such as database administrators or data security staff who advise and support the project, or customer staff who are interviewed about the software requirements.

Organizations that submit data to the ISBSG repository are asked to supply the effort for six activities of the project team, namely planning, specify, design, build, test and implement. In practice the data that is received ranges from specific totals for all six activities through to a single figure for total project effort. Many data submissions received by ISBSG record the effort for only 4 or 5 of these six activities.

ISBSG assumes that if no effort data is reported for a particular activity, then the project never included those activities – but the activities must have been undertaken somewhere, 'outside' the reported project. Hence in order to ensure comparability of data across all projects an estimate of the 'missing' effort must be added to the reported total to obtain the 'real' total project effort.

ISBSG therefore computes the 'Norm L1 effort' to correct for the missing data as follows.

- Where only a total effort for all activities is reported, this is taken as the 'Norm L1 effort'
- Where the effort for one or more of the activities is not reported, ISBSG accounts for the missing activity(s) by applying a correction to the reported data.

The following table shows how the submitted data for four simple examples would be processed by the ISBSG Repository Manager to obtain the 'Norm L1 effort'. (A 'blank' indicates that no data has been submitted.)

Submitted Effort Data (Work-hours)							**Norm L1 Effort (WH)**
Plan	Specify	Design	Build	Test	Implement	Total	
50	100	150	400	200	100	1000	1000
						1000	1000
		200	500	200	100	1000	1200
		250	550	200		1000	1260

There are three important consequences from this approach when the data in this report are used for benchmarking or for estimating software projects.

First, the Norm L1 effort figures used in the calculations of productivity (or PDR) data in this report are on average about 10% higher than the submitted total effort due to the corrections that ISBSG has made.

Second, the Norm L1 figure assumed by the ISBSG for the example project in the second row of this table (1000 work-hours) may or may not be comparable with the equivalent figures calculated for the other three example projects in this table. Lacking any data about which activities are included in the submitted effort data, the ISBSG is unable to apply any correction for possible missing activities.

Third. it is arguable that the effort on 'implement' activities should not be included in the benchmark calculations. The reason is that for estimating software development effort where functional size is the main driver, 'implementation' effort should be confined to tasks that are related to software functional size, such as producing documentation and training related to one implementation. However, the ISBSG data collection form and its definition of 'Implementation' have not made this explicit until recently and do not currently exclude implementation effort following the first site implementation.

This may be the cause of the fact that project data used for the analyses in this report includes implementation effort that is very widely varying. The table below shows some statistics on implementation effort for those business application projects that reported effort under any two or more of the six activity headings.

Parameter	New Development Projects	Enhancement Projects
No. of projects reported on in this table[12]	93	114
Median percentage of total effort devoted to implementation for all projects, i.e. including those that did not report any such effort	7.4%	4.4%
Percentage of projects that did not report any implementation effort	14%	35%
Median percentage of total effort devoted to implementation for projects that reported non-zero implementation effort	8.4%	8.0%
Percentage of projects that reported implementation effort greater than 10% of total effort	35%	24%
Percentage of projects that reported implementation effort greater than 20% of total effort	14%	12%
Maximum percentage of total effort devoted to implementation reported by any project	70%	86%

These data indicate strongly that many projects that reported effort on implementation included effort on activities beyond the first site implementation and/or activities not related to software, e.g. hardware set-up.

For all these reasons, readers using the data in this report for the purposes of benchmarking or estimating purely software-related project activities should note that it is probably the case that the benchmark figures are conservative (or pessimistic) compared with the 'true' figures, by up to around 5 – 10%.

To maintain compatibility with other ISBSG publications this report contains PDR and productivity data that is based on Normalized Level 1 effort. However, a few analyses are included using project effort that excludes effort on implementation, because of its distorting effect. The relevant analyses are indicated clearly in the main text.

Note that from mid-2011 the ISBSG Concise Data Collection Questionnaire has been modified so that for projects where only the total effort is reported, data submitters are asked to indicate which activities are accounted for by the total effort figure. With this information, the ISBSG will be able to correct the total effort figure, if it does not account for all activities, to the 'Norm L1' figure. *Example: for project data shown in the second row of the first table of this Appendix, if the data submitter indicated that the 1000 work-hours accounted for only*

[12] The data in this table have not been updated since the 2009 report.

design, build and test activities (as for the project in the fourth row of this table), then the Norm L1 effort of 1000 work-hours would also be corrected to 1260 work-hours

Also, the ISBSG Glossary now clearly defines ‘implementation’ to be limited to certain software-related activities for the first site. These two changes should enable improved data quality and analysis in the future.