Software Measurement Services Ltd

Benefits to the UK software industry of successfully exploiting metrics and the importance of the COSMIC-FFP method to producing credible metrics

Written by

Charles Symons

Joint Project Leader, the Common Software Measurement International Consortium

December 2006

Ref: 02CSCredibleMetrics

The COSMIC FFP Functional Size Measure was awarded a finalists medal in the 2006 BCS IT Professional Awards. The evidence in this paper was compiled to support the COSMIC entry, and then used as the basis for a paper presented at the 2006 UK Software Metrics Association Conference. The following paper includes revisions made in response to valuable feedback from the UKSMA audience.

Starting points

Managing guesswork

There is an age-old idea that 'you cannot manage unless you can measure'. This is obviously not true in the software industry. Most software projects are 'managed' and efforts are made to improve performance without any proper measurements at all[1]. The phrase should really be something like 'managing software activities without proper metrics to support decision-making is not much better than guesswork'.

Any professionally-managed software metrics programme should be able to monitor and record for each project the three main performance parameters of

- Productivity (= size / effort)
- Speed of delivery (= size / elapsed time)
- Defect density (no. of defects delivered into production / size) – a measure of Quality

All three performance parameters depend on having a measure of software size (i.e. of work-output) that is independent of the technology used for the software and depends only on the required functionality – a 'functional' size. Many other software metrics can be valuable, but these three are the most important and should always be gathered.

There are three main uses for such performance parameters.

Individual project planning. Organizations need to track all three performance parameters because they are tradable. For example, speed of delivery can be increased with more resources, but this usually means lower productivity and carries the risk of lower quality. You can only begin to understand such trade-offs and use this knowledge in project planning when you have your own measurements in your own environment.

Estimating. If software size can be measured from requirements early in a project's life, this can be the prime input to a standard project estimating method or commercial estimating package. Even better, if the performance parameters

[1] A survey of organizations by Meta Group found that 89% collected no performance measurements on their IT projects apart from financial information – like flying a plane by monitoring the rate of fuel burn. ('The Business of IT Portfolio Management: Balancing Risk, Innovation and ROI'. Technical Report, Meta Group, Stamford, CT, USA, January 2002.)

are gathered on completed projects, an organization can set up its own estimating method or can use them to calibrate a standard method or package.

Performance improvement and benchmarking. These three basic performance parameters can be used to guide a performance improvement programme and for 'benchmarking', that is comparing performance e.g. across suppliers and across projects using different technologies. A good benchmark analysis should help managers understand the potential for improvement and where to take action.

Measurement is not essential for initial performance improvement. You can always start improving by simply eliminating obvious bad practice. But without measurements, in the long-term it will be impossible to know if improvement activities are actually yielding positive results, and if not, why.

Indeed, the quantitative management achieved by high-maturity software development organizations (e.g. those achieving CMMI® maturity levels four and five, or 'Six Sigma' equivalent performance) requires intimate knowledge of process performance and variation. This depends on the implementation and institutionalization of efficient measurement methods that provide effective, timely feedback as the basis for decision-making.

As an example, recently there has been much discussion about the benefits of adopting 'RAD' (Rapid Application Development), 'agile' or even 'XP' (Extreme Programming) project management processes. Such processes provide the obvious benefit of delivering functionality earlier to a client than traditional 'waterfall' project management processes and they should help reduce risk. But without measurements, there will be no understanding of the productivity (hence of the cost) or quality of the earlier delivery and of whether the trade-off of cost versus speed over the life of the project is really optimal for the circumstances. Moreover, measuring the whole lifetime cost of the software products, i.e. including on-going maintenance and support costs, will reveal if any further trade-off had been made between the initial speed of development and the lifetime costs. Without such data, the customer will not understand the full implications of the initial development approach.

The fact is that few organizations have any form of long-running software metrics programme. It should be that any metrics programme is better than none, but that's also not true. Many programmes start, fail to deliver benefits, and are cancelled.

Howard Rubin, Senior Advisor to the Gartner, a benchmarking company, carried out research over many years that showed that software metrics programmes typically lasted three years and were then killed off. He recently confirmed at the 2006 UKSMA conference that in his experience the life of metrics programmes seems be getting shorter. They are often cancelled when a new IT Director arrives and can see no benefits from the existing measurement programme. This would not happen if the metrics were credible, useful and used. Later in this paper we will examine why metrics programmes typically lack credibility and what must be done to achieve credibility.

First we will examine what could be the economic benefits from the adoption and long-term successful use of software metrics.

Quantifying the benefits of a credible software metrics programme

A credible long-term software metrics programme should deliver clear benefits. As a minimum it should support performance-improvement activities and help improve estimating.

Apportioning the benefits of succeeding with both these activities between the contributions of a software metrics programme and all the other activities needed to improve performance and estimating requires certain assumptions. The following is a very simple analysis based on some crude input data and conservative (in the author's judgment) assumptions. Our aim is merely to point to the orders of magnitude of potential gains.

Let us first consider the contribution that successful software metrics programmes could make to the benefits of performance improvement activities for the UK software industry.

According to the report 'Survey-based measures of software investment in the UK' published by the Office of National Statistics in February 2006, the UK spends about £20 Billion per annum on software. We first assume that only half of this investment, i.e. £10B pa, could be impacted by the sort of software metrics we have been discussing. (The other 50% would include algorithm-rich software, software that processes audio or video data, investment in infrastructure software e.g. for the desktop in 'small office/home office' installations, and much minor software maintenance. For such situations, either metrics will not help much due to the highly creative work-content or lack of repeatable processes, or people will not bother to measure for small tasks.)

Assume that the UK software industry invested in performance improvement and could double productivity over, say, ten years, a not unreasonable target considering the gap in price-performance between e.g. the UK and India and other evidence (see **A Cross Section of Supporting Evidence** - below).

Almost all branches of UK Government and much of British industry now set performance improvement targets for their major activities. There is no reason to exclude the software industry from quantifying its targets and measuring its progress towards such targets. Indeed, as software is now so ubiquitous, and so much of modern society is dependent upon it, there is an argument that says that improving the software process should be a high priority, as an enabler to performance improvement in other, dependent domains.

Most of the improvement would have to come from adopting better processes and technology, at considerable cost – adding in the order of 7% pa to software costs, according to typical estimates. We can assume the contribution of software metrics to making such a programme succeed, as follows.

Assume the metrics programme's share of the benefits of the overall performance improvement is proportional to its share of the costs of the performance improvement activities, say 10% of these costs. The benefits attributable to the metrics programme would therefore be worth in the order of 10% of the average improvement over 10 years, where the latter is £4.3B pa, net of the improvement costs. On average the benefits of metrics are therefore worth £430M pa, or 4.3% of the current annual software investment. (For simplicity, we ignore complex issues such as whether the benefits are used to reduce costs for the same output, or to increase output for the same costs, the effects of the competition improving its performance over the period thus negating some of the gains, the effects of the cost of money and cash flow, etc. The aim is merely to indicate orders of magnitude.)

The second impact of software metrics should be to help improve estimating. This impact is independent of the above. The analysis below draws partly on the widely-quoted Standish CHAOS report, whose findings we assume apply equally to the UK.

The report (see www.standishgroup.com/press/article.php?id=2) says that in the USA in 2003:

- 34% of projects were 'successful'
- 51% were 'challenged', i.e. they failed to deliver within 10% of budget or time and/or failed to deliver all the promised requirements. On average these challenged projects over-ran on cost by 43% of their original budget
- 15% were 'failures', i.e. they were cancelled before delivering anything. These projects wasted 15% of the $250B spent on software in the US, so it's the bigger projects that have the higher failure rates, as might be expected.

(It's interesting to note that the Financial Times of 1st December 2006 quotes Joe Harley, CIO of the UK Department for Work and Pensions as saying that only 30% of UK Government IT projects deliver on time and budget – very close to the Standish figure for 'successful' projects.)

To quantify the effect of improved estimating, we draw on the ideas of Abdel Hamid and Madnick[2]: They argued that the least cost for a software development project will arise from the most accurate estimate.

If a project is initially over-estimated, a variant of Parkinson's Law tells us that 'work expands to use up the available budget'. So a customer will spend more money than is really necessary and if there is an external supplier of the software, they will make excessive profit.

Alternatively, if a project is initially under-estimated, Abdel-Hamid and Madnick would argue that the costs would have been lower if they had been correctly estimated in the first place. More accurate estimating must give rise to better

[2] The elusive silver lining: how we fail to learn from software development failures', Abdel-Hamid, Madnick, Sloan Management Review, Fall 1990

initial allocation of resources, whilst adding resources to a late-running or over-budget project is more expensive than if allocated correctly at the outset.

The Standish data indicates that 66% (15% + 51%) of projects seriously under-estimated the effort required. It must also be the case that some proportion of the 34% of 'successful' projects was initially over-estimated and could have been delivered at less cost.

If proper project effort estimates were made (often the approach is little better than guesswork), or the accuracy of estimates that were made could be improved, then the following benefits would accrue.

- For projects that are over-estimated, the saving from having an estimate that is, for example 10% more accurate, is actually 10% of the project cost. So the benefits from not over-estimating are potentially very significant – and this conclusion is supported by the evidence given below. But as we wish to estimate our benefits conservatively and do not know what proportion of projects are initially over-estimated, we will ignore altogether the possible benefits of better estimating for this particular group
- For projects which are initially over-estimated and which do not go ahead because the business case is not so good, maybe some potentially good investment opportunities are missed. The size of this group is also difficult to quantify, so again we will ignore this possible source of benefits.
- Now consider those projects that are initially under-estimated but do continue to completion and deliver something. For this 'challenged' group (roughly half of all projects) which over-ran on costs on average by 43%, more accurate estimates should easily save 5% of total costs. The benefits of improved estimating for this group would then be £110M pa (5% of 51% of 43% of £10B, assuming the 51% proportion of projects is also the proportion of their costs).
- Projects that are cancelled, never delivering anything, must have done so because the cost turns out to be much higher than anticipated or some other failure reason. These are the 15% of all projects (15% of costs) according to the Standish report. Obviously a high proportion of these failures were not due just to poor estimating, but if the project team had done a good job on estimating, that would imply good requirements and this would have been a good starting point, with reduced risk of failure. It is reasonable to assume that better estimating in this group alone would also account for at least 5% of the wasted software investment due to lower failure rates. The benefits of improved estimating for this group would then be £75M pa (5% of 15% of £10B)

Combining the benefits due directly to metrics from their contribution to performance improvement (£430M) and to estimating (£185M) suggests the total benefits from establishing successful software metrics programmes would be in the region of £600M pa, or 6% of total software investment for the UK software industry.

This analysis tells us that the various benefits (which are independent and can therefore be added up) attributable to better estimating would be in the region of £185M pa, or nearly 2% of the annual UK investment in software that could benefit from use of metrics.

Note that the overall benefits from the targeted level of performance improvement would be worth several £Billion pa and that it is inconceivable that this improvement could be obtained without a serious investment in metrics. The

So why doesn't most of the software industry invest in software metrics and seek to obtain these benefits from performance improvement and estimating?

indirect benefits of using metrics successfully, (obviously combined with other equally vital investments such as in improved processes, technology, training, etc) are therefore far greater.

Of course, many of the very large software suppliers do have significant software metrics programmes, so some must be already gaining these benefits. But the Standish Group's statistics show the net state of the industry's performance, which must include those organizations that do succeed in using metrics to manage their activities successfully. And we know that even the big players suffer horror stores from time to time. It is therefore safe to assume that these estimates are of net potential benefits that are 'out there', still to be harvested.

Critical Success Factors (CSF's) for a credible software metrics programme and the importance of the COSMIC-FFP method of sizing software

CSF's are the few things you must get right for success in any activity, or it will fail. For a software metrics programme these are as follows:

a. The metrics gathered must be aligned with the software-producing organization's goals
b. The metrics must be credible to the project teams and to management
c. The metrics activity must be seen as complementary to the project processes, not an activity on the side that hinders project progress, and the effort for the metrics must be acceptable within the project budget
d. The organization must be reasonably stable and have certain disciplines such as reasonably repeatable processes and a limited range of technologies (so that performance can be measured on comparable projects)
e. The metrics programme must be used by management to help the organization improve performance over a long period – not to punish or reward individual project performance.

It is imperative that all five CSF's receive attention. Failure on any one CSF will mean failure of the whole software metrics programme.

Based on the author's own consulting experience and confirmed by the observations of Rubin and other colleagues, it is almost certainly true to say that all major software producers have started a software metrics programme at some time, but few have lasted more than a few years. The recurrent reason for cancelling metrics programmes is the failure of CSF's b) (credibility) and c) (complementarity), leading to failure of CSF e) (their use by management).

Time and again, the author has observed that an IT Director receives reports from the software metrics group that look little better than a collection of random numbers, that are difficult to interpret and that give very limited guidance on any action that should be taken. In contrast the computer operations function will produce regular reports on machine utilization which can be used to tune the machine's performance and to forecast accurately when upgrades are needed.

This comparison is a somewhat unfair because the computer operations reports are always produced from data gathered automatically by the machines, whereas the software performance metrics are inevitably gathered by rather laborious manual processes. But the latter aspect does not explain the usual poor quality of the metrics. If a new IT Director does not have a strong IT background and has been parachuted in for a spell of career development (which seems to happen all the time), then little surprise if he or she soon sees an opportunity to save money and scrap the metrics activity.

So why do software metrics lack credibility? Consider the three key performance parameters of productivity, speed of delivery and defect density (a measure of quality). All depend on having a measure of 'size' of the software delivered and, in the case of productivity, – usually the most important measure – a measure of project effort.

Now although everyone understands what is meant by 'project effort', it turns out to be very difficult to measure in a consistent way across multiple projects. There are many good reasons for this (difficulties of definition, etc). And there are bad reasons, principally that project teams tend not to take effort recording very seriously. Especially if they do not know what the data will be used for.

In the author's opinion, this lack of an accepted measure of work-output for software has been the biggest reason why software metrics have lacked credibility and been so little used over the last 10 to 20 years. This weakness means that CSF's b) and c) have not been achievable and consequently CSF e) rarely even gets a chance to be tested.

Getting reliable and consistent measures of project effort, whilst difficult, can be done. But measuring a size of the developed software – a measure of work-output – has always depended on methods that are intrinsically weak, at best. They are particularly difficult to explain to a new IT Director who has no background in the evolution of software engineering principles over the last 30 years.

The two principle approaches to measuring a software size have been to count the number of lines of source code produced by a project (which depends on the technology used) or to attempt to measure the size of the functionality of the software (which should be independent of the technology and thus far more useful). Unfortunately, the most common 'functional size measurement method', known as the IFPUG method, is now increasingly difficult to apply to modern software projects and lacks credibility. The best that can be said for the method today is that it was a great piece of lateral thinking and that it was a pragmatic, credible sizing method when it was first developed in the late 1970's.

In late 1998, COSMIC, the Common Software Measurement International Consortium, a world-wide group of software metrics experts was formed to develop a new method of measuring a functional size of software based on fundamental software engineering principles. The method, known as 'COSMIC-FFP' is now being adopted by major software producers around the world and has been accepted as an International Standard.

Now, at last the software community has available a credible work-output measurement method which should enable setting up a credible software metrics programme and in due course achieving the potential benefits outlined above. So there is hope that the current lamentable state of software metrics will improve in the coming years.

(For those who would like a deeper understanding of the size measurement methods, there are more detailed accounts on www.cosmicon.com. The Appendix to this paper also has a table comparing the three size measurement methods (counting lines of code, the IFPUG method and the COSMIC-FFP method).

A cross-section of supporting evidence

The purpose of this section is to present a variety of evidence from multiple sources which supports the theses of this paper that software metrics are poorly used and that there is enormous potential for benefits to the software industry if they could be used properly.

Evidence that the potential for software development productivity is massive

Organizations that collect software project performance data and provide commercial benchmarking services all seem to show massive variations in project productivity. We give just two examples.

The International Software Benchmarking Standards Group, a not-for-profit organization publishes and analyses data submitted voluntarily by organizations from around the world. Their 2002 analysis reports on the productivity of 97 projects developed using 'main-frame' (=large-scale) computers and '3GL' programming languages (i.e. technology that has been around for 30+ years). The results show a factor x 4 between the boundaries of the lower and upper quartiles of these projects. The boundaries of the upper and lower deciles of performance differ by a factor x 10.

QSM, a USA-based commercial benchmarking services, in its 2006 IT Metrics study reports a factor x 15 difference in 'best-in-class' versus 'worst-in-class' effort for the same work-output, and a corresponding difference of x 5 in elapsed time.

With these sorts of findings, setting a target to improve productivity by a factor x 2 over ten years is really rather modest.

Evidence that project estimates are often made on poor foundations

At the 2006 UKSMA conference a discussion arose about the provision of contingency amounts when estimating. The comments from software metrics experts from two very major software producers were interesting.

One had discovered that an analysis of the total effort in all current project estimates in his organization revealed that 50% of the effort was classified as 'contingency'.

The other representative was asked 'how do your project leaders estimate when asked to do so by a client early in the life of a project, when the requirements are typically not yet well understood?' The answer was 'make the best estimate you can and add 150% contingency'.

The root of this problem is a process issue. Clients, quite reasonably, ask for estimates early in a project life. But quite unreasonably these estimates set expectations and are perceived as 'hard' far too early in the project. Good processes have been developed that help match estimates to requirements as they evolve, whilst maintaining the control of price/performance that the client needs. But such processes require credible software metrics and are very rarely used.

Evidence that project estimates are rubbish and/or are ignored by management

Recently a Department of HM Government received two bids from different suppliers in response to a Request to Tender. One bid estimated a price of £2M; the second bid estimated £0.25M. We can only conclude that either one of the bidders is incompetent (or both?) or they are talking about different requirements. And without proper quantitative methods the client has no way to compare the bids. At stake here is either an over-spend or a saving of up to £1.75M GBP of public money and maybe even the costs of yet another IT project failure due to unrealistic initial estimates.

(This case leads on to the important observation that a software statement of requirements can only be measured for size and hence used as a basis for effort estimation if it is free from ambiguity. It follows that the ability to measure a functional size of a statement of requirements is an extremely valuable check on the quality of those requirements.)

A City of London financial institution committed to develop a large complex system without using any systematic approach to sizing or estimating the development effort, duration or cost. They engaged a number of external suppliers to build distinct sub-systems and agreed contracts, again without reference to any process performance or benchmark productivity data. The development was planned to take 24 months. After many painful mishaps, and the expenditure of £40M, the project was cancelled. Unofficial estimates of the size of the requirements, made by project staff but ignored by management, suggested a functional size of over 100,000 IFPUG function points – indicating a 'mission-impossible' project.

Evidence that software project productivity can be improved

Companies seem reluctant to admit to using software metrics, let alone acknowledge performance improvements, but here are a few examples

Amortizing the benefits over only one year, the unit achieved a 2:1 return on investment (ROI)

ABB, the Swiss/Swedish engineering company reported in August 2006 as follows. "ABB started calculating return on investment (ROI) corporate wide in 2003, and the typical ROI for major (software) process changes is 3:1 to 5:1 (benefit to cost of process improvement activities). … The only benefit included in the ROI calculation is the savings that resulted from the 30 percent reduction in COPQ [Cost Of Poor Quality]. Amortizing the benefits over only one year, the unit achieved a 2:1 return on investment (ROI)."

Northrup Grumman Electronic Systems reported in August 2006. "The average gain in productivity that we have experienced during the past 5 years as we have moved from CMMI® maturity Level 3 to 4 is approximately 20 percent annually.

During our static years, our nominal gain was 10 percent annually. We can thereby conclude that we have accelerated our gain by 10 percent a year based upon a strategy that was heavily Software Process Improvement based. Such acceleration results in a cost avoidance averaging $25 million annually over a five-year investment time span."

Evidence that use of software metrics can make a difference

Rubin presented evidence from Gartner on IT expenditure at the UKSMA 2006 conference. Currently, as a percentage of operating expenses, companies spend in the range of 2% (manufacturing) to 10% (banking) on IT. Of this IT expenditure, on average about 60% is spent on 'run the business' and about 40% on 'grow and transform the business'. Rubin's data showed, however, that businesses that use measurement, especially software metrics, more typically spend less than 50% of their IT budget on 'run the business', thus leaving more than 50% on 'grow and transform'. The benefits of metrics therefore extend way beyond just the performance of the software producers but extend into the business performance.

IBM (Europe, Middle East and Africa) made the following statement at the European SEPG conference in 2005. "Measurements provide a balanced view of delivery performance. Measurements connect goals, behaviour and results, thereby providing a path for improvement & maturity over time. Measurements enhance customer value by providing visibility into accounts' delivery performance. Measurements allow practitioners and management to be sure that their actions are effective."

Evidence that some people are waking up to the importance of measurement

The UK House of Commons Defence Committee HC 572 Session 2003/4, 28 July 2004 into MOD acquisition (including the acquisition of software-intensive systems) concluded that "loose approximations suggest 15% of total procurement spend should be for de-risking", including more emphasis on the early stages of projects. Indeed, two of the seven principles of SMART Acquisition are highly relevant, namely:

Principle 1. Adopt a whole-life approach, typified by applying through-life costing techniques

Principle 5. Establish effective trade-offs between system performance, through-life costs and time.

Some parts of UK Government actually do have a good record of collecting software metrics, though it is less clear that they have been successful in using them to drive performance improvement from their suppliers. However, perhaps things are changing. The Financial Times report of 1st December referred to above (stating that only 30% of UK Government IT projects are delivered to time and budget) went on to say that Government Departments have agreed with a dozen of their principal suppliers to set a target to raise this level to 90% over the next four years.

1% saving per annum COSMIC could make the difference between success & failure

Conclusions

Overall, the claim we have made of a few percent benefit to total software investment from the exploitation of successful software metrics programmes is probably quite conservative. Every 1% saving for the UK software industry is worth £100M per annum, so the rough calculations presented here suggest the savings for the UK software industry could be several £100m pa.

And the exploitation of a successful software metrics programme should help unlock benefits in the order of £ Billions from productivity improvement.

And adopting the COSMIC-FFP method for software size measurement could make the difference between success and failure of a software metrics programmes.

So why hesitate?

Acknowledgements

The author is extremely grateful to Grant Rule of Software Measurement Services Ltd, and to Professor Alain Abran of the École de Technologie Supérieure - Université du Québec, Montréal, joint leader of the COSMIC project, for their helpful contributions to this paper. I am also grateful for feedback and comments supplied by participants at the UK Software Metrics Association 2006 conference where an early version of this paper was presented.

Appendix A comparison of software size measurement methods

The table below shows how the two most commonly used software size metrics compare with using the COSMIC-FFP functional size measure.

Critical Success Factor	Counts of Source Lines of Code	IFPUG Function Points	COSMIC-FFP
b) Credibility of the size metric	- Used only for real-time and embedded software - Rules for line-counting vary enormously. - The size measure is precise, but depends on the programming language. - Conversion ratios between languages are questionable - Limited relationship to size of functional requirements	- Mostly used only for business application software. - Based on a pragmatic model of IBM software in the late 1970's - Lacks credibility for large complex projects due to the limited size scale (the measure is a non-linear, ordinal scale)	- Designed to measure both business application and real-time software, in multi-tier, multi-layered architectures - Based on fundamental software engineering concepts and measurement principles - Provides a linear, ratio scale of measurement
c) Size metric is complementary to project processes	- Can be measured accurately when a project is finished. - Can only be 'guestimated' from requirements, hence limited value for project estimating, especially early in the project - Size depends on skill and expertise of the programmers; 'better' programmers produce 'tighter' code, hence fewer SLOC, so appear to have lower productivity	- The method's basic concepts date from the late 1970's and now have limited relevance to modern practice in requirements engineering and software development. Hence measurement is separate from project processes	- The method's basic concepts are aligned with modern software engineering methods such as UML, but independent of any one method - So measurement can be embedded in typical software development practices, minimizing the cost of data collection - Measuring using CFFP also provides excellent Quality Control of requirements
Acceptable effort for size measurement	- Measurement of completed software can be automatic	- Measurement is manual but with acceptable effort	- Measurement is manual but with acceptable effort - Measurement may be automatable if model-driven software engineering principles are adhered to

Our conclusion from this table is that using the COSMIC-FFP method can make the difference between a credible software metrics programme and one that will

fail and be abandoned. This, in turn, feeds the fifth CSF, which means that the metrics programme should be sustainable over a long period.

……..

Author Biography

Charles Symons is joint project-leader of the COmmon Software Measurement Consortium, a team of software measurement experts from Europe, North America and Asia/Pacific. The Consortium developed the COSMIC-FFP method to measure the functional size of real-time, multi-layered software such as used in telecoms, process control, and operating systems, as well as business application software, all on the same measurement scale. Such wide applicability is unique and a break-though for the world of software project performance measurement and estimating.

The COSMIC-FFP method has progressed from the germ of an idea to approval as an International Standard in the extremely short time of four years. The method has been extensively tested and is now becoming increasingly used, especially in the real-time world. It is compatible with modern specification methods such as UML, and with OO techniques.

After a working life covering all the major disciplines of the Information Systems function, Charles Symons is now semi-retired. His extensive experience in both public and private sectors informs his continuing work in promoting the COSMIC method as a measurement technique designed for the 21st century.

……..

Software Measurement Services is a specialist, independent UK consultancy working with decision-makers in blue-chip companies and government departments to improve the results delivered by the development of software and computer systems. Our consultants are at the forefront of developing and supporting best practice in managing software process performance.

For more on the COSMIC-FFP method, see www.cosmicon.com and/or contact your local member of the COSMIC International Advisory Committee, whose e-mail address is given on the 'cosmic' site.

……..

Software Measurement Services Ltd
124 High Street
Edenbridge
Kent
TN8 5AY

T: +44 (0) 1732 863 760 F: +44 (0) 0732 864 996

http://www.measuresw.com
sales@measuresw.com