COME BACK FUNCTION POINT ANALYSIS (MODERNISED) – ALL IS FORGIVEN!

Charles Symons

ABSTRACT

Function Point Analysis was invented by Allan Albrecht of IBM as a means of sizing business application software independently of the technology used for its development. Albrecht's method was heavily promoted in its early years and has become the most widely used such method. However, the underlying model which Albrecht used for his sizing method, which was valid in the mid 70's when it was first conceived, is increasingly difficult to apply to modern software development. This and other factors have led to a decline in the method's use.

In this paper, we examine the reasons for the decline, and the main advances in thinking on the more general topic of Functional Size Measurement (FSM) which have been made in the last 15 years. Specifically, the COSMIC FFP method is seen as a significant step forward in being the first method designed by an international group of software metrics experts to work for both business application and real-time software.

Furthermore, it has been realised that a reliable FSM Method would be a very great asset with many uses, such as helping improve requirements specification, estimating, project 'scope creep', supplier performance measurement and contract control, etc. The experience of the new methods and the realisation of their potential value indicate that a return to popularity of (modernised) Function Point Analysis, in the guise of more general FSM Methods such as COSMIC FFP is highly likely.

1. BACKGROUND

The original idea of measuring a size of software from its requirements or functional specifications was introduced by Allan Albrecht of IBM over 20 years ago (Ref. 1). At the time it was a genuine breakthrough in thinking by providing the first method for sizing software which was independent of the technology to be used for its development. The method could therefore be used for comparing performance across projects using different technologies (using measures such as 'productivity', defined as size / effort) and as a first step in methods for estimating effort early in a project's life-cycle. This was a big step forward compared with the use of counts of Source Lines of Code (SLOC), which had been the only size measure up to that point.

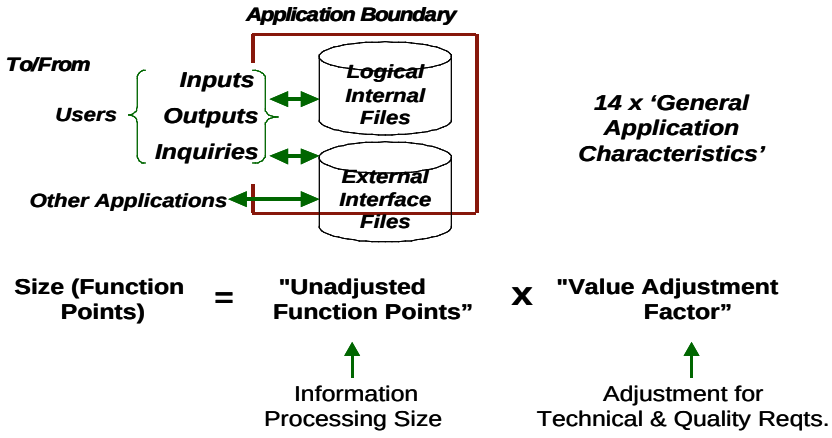

Fig. 1 The Albrecht (IFPUG) 'Function Point' model

Albrecht's model of functional specifications requires the identification of five types of components, namely input, output and inquiry elementary processes, and logical internal and external interface files. Having identified these five types of components in a specification, they are then weighted for complexity and are allocated 'unadjusted function points'. The total of 'UFP's for all components is then multiplied by a Value Adjustment Factor which takes into account the supposed contribution to size of some 14 technical and quality requirements (see Fig. 1).

Albrecht's Function Point method coincided with a push by IBM to promote 'programmer productivity' in its primary markets of business application, or 'management information systems' (MIS) software. The sizing model fitted well with how MIS software functionality was thought about in the mid 80's and the resulting performance measures were plausible. The method was heavily marketed by IBM's sales staff and software engineers, and a whole new micro-specialism of 'function point analysis' was spawned. Estimating tools were developed based on FPA and sold successfully, and within a few years an International Function Point User Group ('IFPUG') was established to take over responsibility for standards and promotion of the method.

The model of the five concepts and the Value Adjustment Factor is a pragmatic model. Albrecht was quite rightly trying to find a sizing method for MIS software which would correlate with development effort and it is a remarkable tribute to his ideas that the model has been successful for so long.

It seems safe to assert that in the late 80's and early 90's FPA was tried and used to some extent by the Information System departments of most major companies and Government departments in North America, much of Western Europe and other parts of the world. It is equally safe to say that the method is now much less widely used. It has in fact followed the well-known pattern of the Gartner 'hype-curve' (see Fig. 2) and only now is FPA climbing out of the 'trough of disillusionment' in new guises and with new realism.

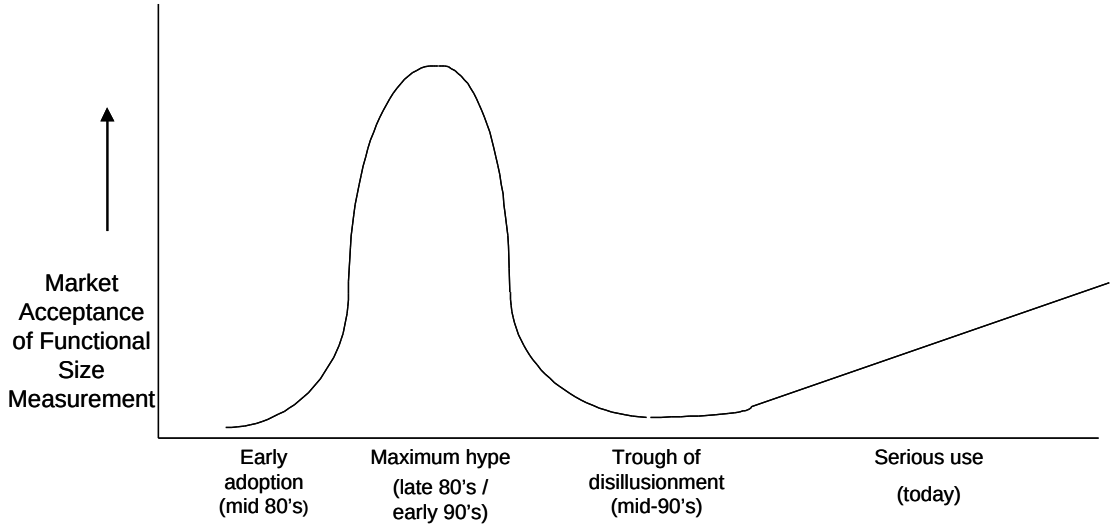

Fig. 2 The Gartner 'Hype Curve' for Functional Size Measurement

The duration of the trough of disillusionment has been long and harmful in the sense that the topics of performance measurement and early life-cycle estimating are quite frankly backwaters in the whole spectrum of software engineering methods. Recent graduates in computer science have heard of FPA, but very few ever find the opportunity to practice it in their real jobs. Progress in improving Functional Size Measurement or FSM (to give the generic name for the topic) following Albrecht's original lateral thinking has been extraordinarily slow. Understanding of performance measurement in software engineering is limited and every week we read reports of the consequences of poor early life-cycle estimating.

This paper examines the factors which led to the period of disillusionment, why it lasted so long, and the changes which are now taking place that are leading new 'modernised' developments of Albrecht's original FPA method into mainstream software engineering.

2. THE TROUGH OF DISILLUSIONMENT WITH FPA

Undoubtedly the reasons for the prolonged trough of disillusionment are much wider than just the limitations of Albrecht's original method. When the method was first promoted it was hyped as the software manager's latest 'silver bullet' which led to setting up comprehensive software metrics programmes in many organisations. These programmes turned out, however, to be more difficult to establish and to produce useful results than was anticipated. With the great benefit of hindsight, one can see the problems which arose.

- Often the gathering of software metrics data was delegated too low down in the organisation. Measuring the performance of software development projects, the results of which would reflect on the performance of hard-nosed project managers, is unavoidably a political matter. Junior metrics staff inevitably have a tough time getting the attention needed to collect reliable data. And if it's 'garbage in', then it surely follows there will be 'garbage out' and the programme will soon be discredited
- If the programmes did not fail due to poor input data, many failed due to poor analysis and presentation of the collected data. FSM methods are based on assumptions about reasonable approximations of size, and the results often tend to show a wide divergence of performance about the mean. Moreover, FSM is inherently a rather academic subject which is not easy to explain simply to a senior manager who may not have a background in software engineering. Such an IS Manager would receive comprehensive performance data from the Computer Centre showing how the processors were loaded and the data could be used to optimise performance and plan for upgrades. So why couldn't the software development group produce convincing, meaningful data about their performance? The effect of poor analysis and presentation of a complex subject leads to suspicion about the validity of the measurements and averages. If senior IS management did not understand the results and/or could not see how to use them, the metrics programme would be an obvious candidate for the next round of budget cuts
- Albrecht's method has been very helpful for sizing software which is to be built to bespoke requirements, which was the dominant activity of Information Systems departments in the 70's and the 80's. But over the most recent decade many major companies switched over to using packaged software rather than creating their own bespoke software. Add in the adoption of rapid application or 'time-boxed' development, where estimating is partly forgotten about, and FP-sizing suddenly seems to be much less relevant.
- The continuous waves of mergers and acquisitions and outsourcing deals automatically led to cost-cutting and to loss of the FPA and software metrics expertise in many organisations. Ten years ago, for example UK Central Government departments all had significant software metrics expertise. Now that expertise has largely been outsourced.
- One might have imagined that entering into a contract to outsource software development and maintenance would automatically lead to an emphasis on measuring the performance of the supplier. But such contracts are negotiated by accountants and lawyers under great time pressure and often in secrecy from the organisation being outsourced. Whatever other skills they bring to the party, it is very rare to find an accountant or a lawyer who has any real understanding about measuring performance in software development. They tend to look for global cost reductions, open-book accounting, ways of agreeing costs in advance of any new project and such-like control methods.

All these factors have tended to diminish interest in FPA, but in addition the concepts of the IFPUG FP method itself now appear simply out-of-date when compared against the concepts of modern software development methods.

The IFPUG organisation has, of course, made valiant efforts to produce guidance on how to interpret Albrecht's original five component-types in terms of modern development methods. But inevitably if the developer is working in terms of concepts such as Objects or GUI's or Use Cases, or API's or whatever, the translation from these concepts back to the twenty-year-old concepts of 'Elementary

Inputs', or 'Internal Logical Files', etc will seem like a distraction to his main purpose. And the Value Adjustment Factor is now totally irrelevant to modern software development.

One of the difficulties of having to continuously add new rules to cope with interpreting an old sizing method in terms of new development methods and technologies is that the method definition and procedures become increasingly complex. It becomes hard to maintain consistency of interpretation for all circumstances (see examples below). Once the size measurement method is no longer in the main stream of the project manager's or the developer's mind-set and methodology, and it becomes difficult to use, it will very easily be forgotten about. It is not at all surprising that Albrecht's successful pragmatic approach of 20 years ago is now running into difficulties.

3. THE EMERGENCE OF 'SERIOUS USE' OF FUNCTIONAL SIZE MEASUREMENT (FSM)

After the hype has died down, we now see clear signs of growing and serious use of FSM, albeit updated in new forms. For this to happen, several conditions must be satisfied.

- There must be clear potential value in the market-place for reliable, practical FSM Methods
- The FSM Methods must be compatible with modern ways of developing and maintaining software and must be demonstrably usable for sizing and estimating. There must be a clear understanding and confidence in what these methods can and cannot do
- If organisations which have built up significant repositories of performance data and estimating methods based on a particular sizing method are to be persuaded to adopt a new sizing method, there must either be conversion rules which will help migration from the old to the new sizing method, or the need to change must be so compelling that the organisation is willing to make a significant investment in the new sizing method
- There must be a truly international quorum of expertise in the subject to provide training, consultancy, user support, tools, standards, etc., that is, all the paraphernalia which a user of these methods will need before he is prepared to invest in them. The software industry is global; local methods are of little interest to major corporations
- The problems cited above of setting up, sustaining and gaining benefits from software metrics programmes must be addressed
- Finally, there must be sustained marketing to make the market aware of the potential value and of the availability of proven solutions

We will examine each of these factors in turn.

4. THE MARKET POTENTIAL FOR RELIABLE FSM METHODS

Supposing we have a reliable, practical method of producing a size of software early in its life based on functional requirements, what might be the potential uses? The 'mind-map' (Fig. 3) shows the potential of using functional size measurement as a component of methods -

- for the requirements analyst to produce requirements that are measurable (as well as unambiguous, structured, traceable, testable, etc)
- for the project leader to assist with estimating, control of requirements 'scope creep', etc
- for investors to help decide on the cost/benefit analysis for new projects and for valuing software assets at their replacement value
- for those concerned with performance improvement to measure and benchmark performance
- for procurement and contract managers to help select a supplier and to measure and control its performance

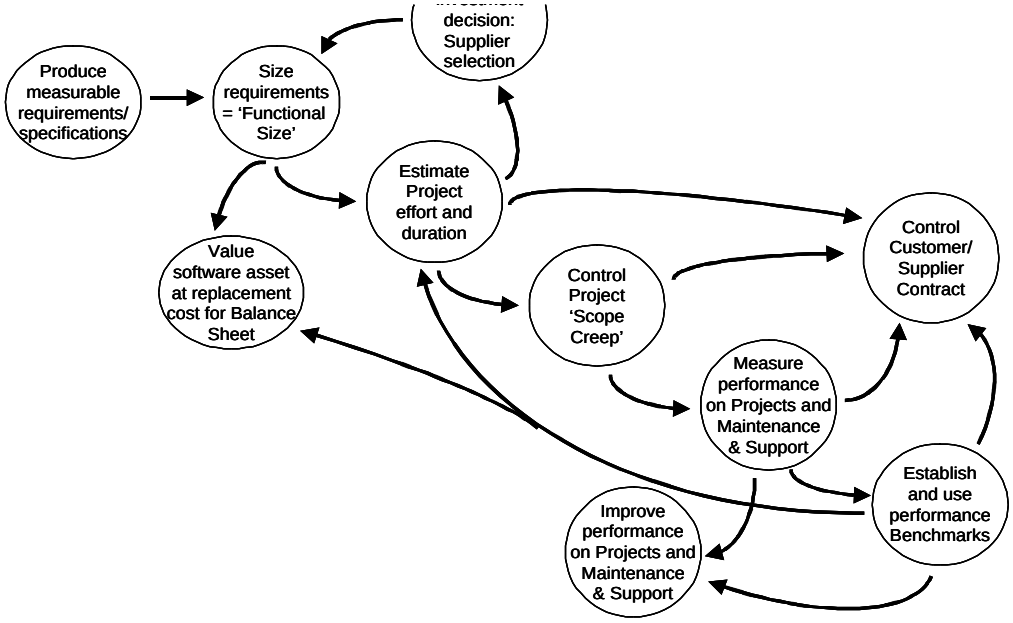

Fig. 3 'Mind-map of the uses of Functional Size Measurement

There is clear market potential demand for all of these potential uses. There is no need to remind readers of this paper of the need to improve early life-cycle estimating, of the needs of project leaders to control requirements, and so on. Even those involved in outsourcing software development and maintenance, having gone through the transition processes of the early contract years, are now increasingly concerned to improve methods of controlling value for money from their suppliers. Some Government Departments, for example, (Ref. 2) are specifying the use of FSM Methods as part of their software procurement procedures.

All these remarks apply to the world of business application or MIS software, for which Function Point Analysis was originally designed. The world of 'real-time' software (by which we mean software for telecoms, process control, operating systems, embedded systems, avionics, etc) has never experienced generally available FSM Methods, so the potential value in that segment of the software industry is even greater.

The economic value of the potential for reliable FSM Methods is therefore undeniably enormous.

5. 'MODERNISING' FUNCTIONAL SIZE MEASUREMENT

Fig. 4 below shows the development of some of the principal ideas for improving Functional Size Measurement since Albrecht's original FPA method.

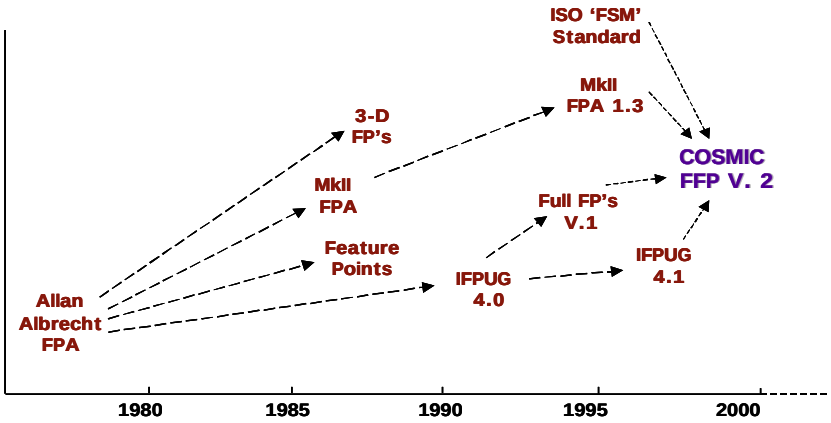

Fig. 4 Evolution of Functional Size Measurement

Albrecht's original FPA method has evolved over the last 20 years into a method now known as 'IFPUG 4.1', though the original basic concepts and weighting methods have not changed since 1984. Over this same period other methods have been put forward, each attempting to overcome weaknesses perceived in Albrecht's original model, or to extend its field of application. Noteworthy amongst these are the following.

- Jones (Ref. 3) extended Albrecht's model in his 'Feature Points' method by adding a sixth component to account for the contribution to size of mathematical algorithms. This method has not really caught on due to the inherent difficulty of agreeing standard ways of defining and assigning a weight to algorithms of increasing size and complexity. It seems fair to say that this problem of sizing mathematical algorithms in a simple way is a problem which has not yet been solved
- MkII FPA (Ref. 4) was proposed by the present author to improve the functional size scale (by comparison with the Albrecht method) so that it takes better account of the internal processing complexity of MIS software. The method also updated some of Albrecht's basic concepts in line with 'Structured Analysis' methods, which were popular in the late 80's and early 90's. The MkII method regards MIS software requirements as composed of 'logical transactions' each with an input, process and output component. The size of the 'process' component is measured by counting the number of entity-types referenced in the processing of the logical transaction. One of the main differences between the Albrecht and MkII methods therefore is that the former method counts 'Logical Files' once per piece of software being measured, whereas the latter method counts 'entity-types' every time they are referenced in each logical transaction. The MkII method originally proposed an extended Value Adjustment Factor, but this was dropped a few years ago when it was realised it was no longer meaningful
- 3-D Function Points were proposed by Whitmire of Boeing (Ref. 5) as a means of extending Albrecht's method into real-time software. Whitmire added 'Control' components to Albrecht's 'Functional' and 'Data' components. It is understood that the 3-D Function Point method is still used successfully in Boeing, but details of the method have not been published outside Boeing.
- Full Function Points, or 'FFP V1' was developed by a team based at the University of Quebec in Montreal, Canada (Ref. 6). This method relies on the IFPUG method to size MIS software, but adds six new component-types to enable sizing of real-time software. The method has been extensively and successfully trialled for sizing real-time software by several industrial partners. Its development has now ceased in favour of the COSMIC FFP method.
- In 1996 the International Standards Organisation started a Working Group (ISO/IEC JTC1 SC7 WG12) on Functional Size Measurement, with the participation of software metrics experts from about ten nations. The Working Group has concentrated on establishing the common principles of functional size measurement, rather than attempting to develop a new FSM Method. The fist publication setting out these basic principles was ISO/IEC 14143-1, published in 1998 (Ref. 7).
- At the end of 1998, a group of software metrics experts, mostly participating in the ISO Working Group established COSMIC, the Common Software Measurement International Consortium. This group set out to develop a new FSM Method which would work equally well for MIS and for real-time software. The principles of the new 'COSMIC FFP V2' method were published in October 1999 (Ref. 8), field trials were held in 2000, and the results of these trials have recently been published (Ref. 9). The COSMIC FFP model for functional sizing is extremely simple, see Fig. 5. In outline, Functional User Requirements are decomposed into 'Functional Processes' which in turn can be decomposed into 'Functional Sub-Processes'. A Functional Sub-Process is a type of Data Movement, namely an Entry, an Exit, a Read or a Write, each assumed to have associated data manipulation. The method explicitly does not claim to measure the size of functionality which includes complex data manipulation (i.e. algorithms), and does not attempt to take into account the effect on size of Technical or Quality requirements.

What has been learned and what progress has been made over this last 20 years of endeavour? In the author's opinion, the following are the most important lessons learned. (In the following, the detail of much of the supporting argumentation has had to be omitted through lack of space.)

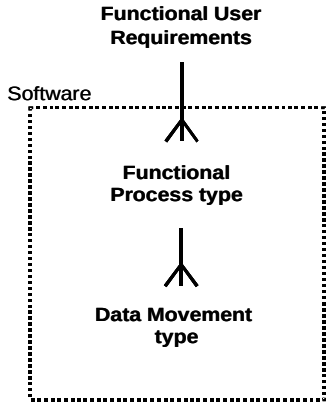

Fig. 5 The COSMIC FFP functional sizing model

5.1 The need for Unambiguity

First, no functional sizing method will ever be successful unless it is based upon completely unambiguously defined concepts. Unless this condition is satisfied, different analysts will produce different sizes from the same requirements document or functional specification. Most importantly, the MkII Logical Transaction and the COSMIC FFP Functional Process share the same conceptual basis (a sequence of sub-processes, triggered by a unique event-type outside the software, which, when complete leaves the software in a coherent state with respect to the event). The IFPUG definition of an Elementary Process is now close to this same concept, which originates with the work of the founders of 'Structured' methods in the 60's and 70's.

As an illustration of the ambiguity associated with some development methods, Jensen of IBM Global Services reported that he had discovered a very large number of variations of interpretation of the UML 'Use Case' concept within IBM. After much analysis, Jensen and the present author concluded that the MkII Logical Transaction (or COSMIC FFP Functional Process) is a specific case of a Use Case. Exploiting this idea as a refinement of using Use Cases leads to establishing statements of requirements with a far higher chance of unique interpretation, and which are measurable (Ref. 10). The MkII Logical Transaction is now a basic component of the IBM LEAD estimating method for Object-Oriented developments (Ref. 11).

One could continue with other examples. The IFPUG 'Internal Logical File' concept is frequently one of the most difficult concepts to interpret in practice. The COSMIC FFP Data Movement, in contrast, is tightly defined; difficulties of ambiguous interpretation were not experienced in the field trials.

5.2 The need for Domain Independence

Second, it is vital to have a functional sizing method which works equally well in the MIS and real-time software domains. Although every experienced software engineer has an instinctive understanding of the difference between these two main broad categories of software, it is actually very difficult (as the ISO Working Group has discovered) to make a clear distinction. And in real-world software engineering, the two domains overlap.

MIS systems frequently receive real-time data feeds or have to interface with real-time software such as for telecoms. And telecoms and process control software may well need to access significant databases which have all the characteristics of MIS software. Similarly, real software projects in both domains frequently involve developing or changing software items in different layers or peer items of multi-tiered infrastructure software architectures. Practitioners concerned with sizing and estimating need common methods which will work across both these two main domains, in any layer or peer software item.

The COSMIC FFP V2 method is the first publicly-available such method which meets these criteria.

5.3 The need for Different Sizes for Different Purposes

Third, we need different sizes for different purposes, depending on the scope and purpose of the sizing activity.

When Albrecht invented the Function Point method his aim was to define a size measure of business application software seen from the viewpoint of the human user's requirements. He also wanted a method which could produce a size with reasonable analysis effort, so inevitably the sizing model is a simplified view of the full detail of how a human user interacts with the software.

For many years this human view of the business application coincided with the application developer's view. So 'first generation' IFPUG and MkII Function Point sizes have been accepted as valid measures of a development team's 'work-output' and therefore likely to correlate with the required effort to develop the software for a given technology

Even when PC / main-frame client-server architecture was first introduced, the rules for FPA could be adapted to cope with the new views. Metrics specialists recognised the need to make a distinction between the human user (or 'end-user') view of the size, and the developer's view. The end-user's view does not change simply because the functionality is distributed over two processors (the end-user is unaware of the distribution) so FPA methods could still be applied to measure size from this view. But they could also be applied to measure the size of the client and the server components separately (see Fig. 6 below). And metrics specialists could explain that the sum of the sizes of the two components measured separately would exceed the size as seen from the end-user viewpoint, due to the inter-processor communication.

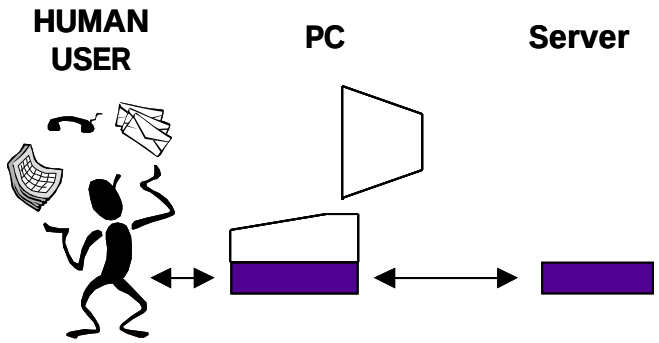

Fig. 6 Two-tier (PC/server) Architecture

For performance measurement purposes, metrics specialists could still use the end-user view of size, and it is interesting to note that performance on projects measured this way often appears lower than if the same functionality were developed on a single platform. But if we are concerned with accuracy, particularly of estimating, it is desirable to use the sizes of the two components measured separately, especially when the two components are developed using different technologies. For this level of architectural complexity, first-generation methods such as IFPUG and MkII FPA are still perfectly adequate.

But these methods have really reached their limit of applicability when it comes to measuring and estimating for a multi-tier, multi-layer software architecture.

The main reason why a major bank joined the COSMIC FFP field trials is shown in Fig. 7 below. This shows the architecture of changes that had to be made to several of their systems to allow everyone in the bank to access a common customer database. For this purpose, the bank had to build enquiries at the PC's in the branches and offices, upgrade the back-office servers to handle the enquiries, upgrade a front-end processor which handles all traffic over the Wide Area Network, and build new, two-layer software on the mainframe to manage the common customer database.

When this system was sized with the IFPUG method (and the same would be true of the MkII method), the bank could measure only the functionality as seen by the human end-user at the PC, or perhaps at most the functionality of the PC component and the database server. The large amounts of functionality which the project team had to develop to support the end-user requirements are simply invisible and cannot be measured by first-generation FPA methods, without stretching the methods beyond what they were designed for. The IFPUG measurement therefore correctly sized the functionality as seen by the human end-user, but grossly under-sized the functionality delivered by the software team. The measurement appeared to show that the team had a very low productivity.

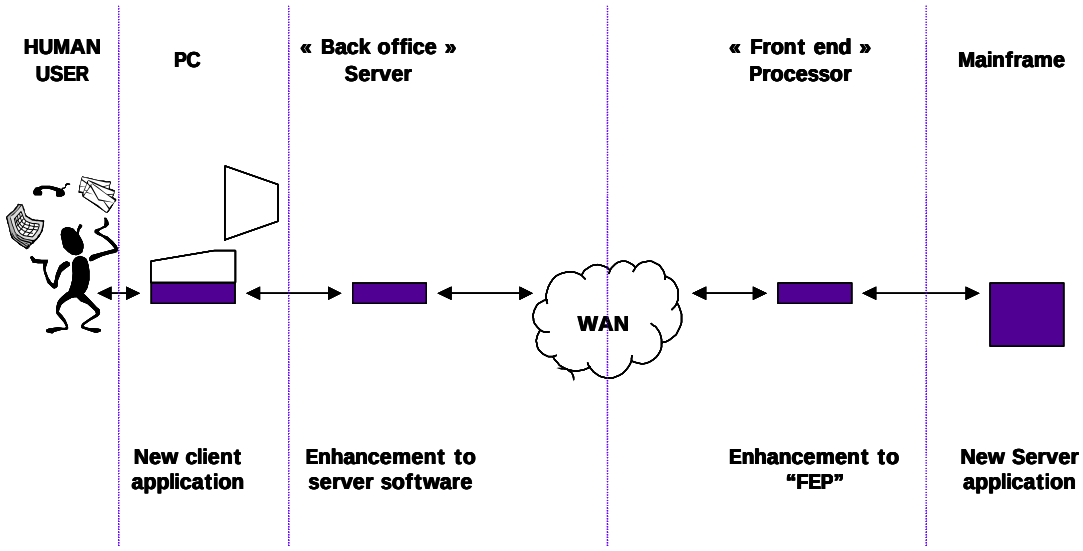

Fig. 7 A four-tier client-server architecture

With the 'second-generation' COSMIC FFP method, all five components of the system could be separately sized, which of course gave a much larger and more realistic view of the functionality delivered from the developers' viewpoint.

Generally the power of the COSMIC FFP method is that it can measure the functional size of any component of software as seen by the *direct user(s) of that component.* And a component can be at any level of decomposition. In COSMIC FFP terms, a 'user' is 'any person, engineered device, software component or thing which interacts with the software being measured'. So in Fig. 7, the user of the back office server is the PC in the branch, and so on.

For future purposes of producing standard size measurements, we may need to distinguish

- The 'external user' view of functional size – a size measure mostly of interest to the human user, only in the MIS domain
- The size of the 'principal components', that is the size of the principal separately developed and maintained sub-systems, maybe also running on separate processors
- The size of any 'building brick' components which have to be built.

The reason to distinguish these 'size types' carefully is that they are not additive. Just as the size of a set of bricks cannot be added to obtain the size of a wall, so the size of a set of objects cannot be added together to obtain the size of a principal component assembled from those objects.

5.4 The Way to deal with Technical and Quality Requirements

The fourth lesson is that Albrecht's approach to account for Technical and Quality requirements via a Value Adjustment Factor which multiplies the size obtained from pure Functional User Requirements is <u>not</u> the best way to deal with such requirements.

Albrecht's approach is rather like saying we will estimate the size of a house in cubic meters, then adjust this size using a scale running from 'Not Present' to 'Significant Impact', for a series of factors such as whether the building site has easy access or not, whether materials are scarce, whether the kitchen needs fitting out, etc. These factors clearly need to be taken into account when estimating the *effort* and *cost* to build the house, but they do not alter its *size*.

Similarly, Functional Size should be based purely on Functional User Requirements. Technical and Quality Requirements should be taken into account in methods to estimate development effort, cost and elapsed time, since their impact clearly depends on the technology to be used (see Ref. 11 for a good example of how Technical and Quality requirements are handled).

Summarising the main lessons learned from developments in FSM Methods over the last 15 years.

- The concepts on which modern FSM Methods are defined, such as the COSMIC FFP method must be, and are believed to be, unambiguously defined, so as to ensure consistent sizing
- FSM Methods must be 'domain independent'; this condition is satisfied at least for MIS and real-time software by the COSMIC FFP method, but no FSM Method is currently capable of sizing complex mathematical algorithms
- We need different functional sizes for different purposes. The human user view of size is quite different from the size seen by the developer of specific components of a system. The same FSM Method may be used at the various levels of abstraction, but the resultant sizes may not be additive
- Technical and Quality Requirements should not be combined with Functional User Requirements to produce a composite size. The result is very difficult to interpret, and the scale factors inevitably depend on assumptions about the technology to be used

6. CONVERSION FROM OLD TO NEW FUNCTIONAL SIZE SCALES

Where two functional size scales attempt to measure the same size, one may reasonably expect a simple conversion formula between the two scales. Such is the case for conversion between the IFPUG Unadjusted FP scale and the MkII FP scale, now that the latter method has discarded a Value Adjustment Factor. Measurements of the 'external user' view of size of the same software carried out using both methods have produced a simple quadratic conversion formula (Ref 12).

Similar experiments using the FFP V1 method showed a close relationship of size with the IFPUG scale, when they were both measuring the functionality as seen by the external user.

But clearly, if one method can only measure the external user view of size whilst another method can measure the size of components of the same software, then no conversion is possible from the former measurements to the latter. (The reverse may be possible, but this needs more research.)

Organisations with large accumulations of measurements of software using the IFPUG or MkII methods who wish to be able to measure the size of components of their infrastructure using a method such as COSMIC FFP V2 therefore have to accept that there can be no conversion from the old measurements to the new. However, it may still be perfectly reasonable to continue to use the 'old' measurement methods for certain purposes, and to add in the 'new' measurements for the new purposes.

7. INTERNATIONAL SUPPORT FOR THE COSMIC FFP METHOD

A specific goal of the COSMIC initiative has been to 'develop, test, bring to market and gain acceptance' of the new FSM Method. This goal is being pursued in several ways.

First the COSMIC Core Team now comprises software metrics experts from Australia, Canada, Japan, Vietnam and six European nations. Field trials of the method were conducted on three continents. The Measurement Manual has been translated from English to French and Spanish, whilst German, Italian and Japanese versions are in an advanced stage. The COSMIC FFP method has been accepted onto the work programme of ISO and an 'ISO-compatible' version of the Measurement Manual will be tabled in May 2001.

Training course and case study material has been developed and training is now available in various parts of the world. The performance measurements from the field trials (mostly from real-time software projects) are being made available to the International Software Benchmarking Standards Group. The Measurement Manual, much case study material and research papers are freely available in the public domain from the University of Quebec at Montreal web-site (www.lrgl.uqam.ca).

In summary, therefore, although the COSMIC project still has a long way to go, a great deal of effort is being deployed towards this goal and much has been achieved over the last two years.

8. IMPROVEMENTS TO AND MARKETING OF SOFTWARE METRICS PROGRAMMES

This goal will also take some time to achieve, but there are significant forces driving towards improving understanding of the processes needed to produce reliable software performance metrics and to use those data for estimating, performance improvement and all the other potential uses identified above. Amongst the most important drivers are the following.

- Efforts in the direction of Software Process Improvement, particularly via using the Capability Maturity Model (or CMM). The new CMM-I model requires attention to measurement at Maturity Level 2, rather than ML 4, as in the original CMM model. (Ref. 13)
- The new ISO 9001 (2000) places greater emphasis on measurement than the previous version
- As stated above, there are increasing signs of awareness amongst clients of outsourced software development and maintenance services of the need for accurate measures and benchmarks of their supplier's performance. Similarly, as more software is contracted out and margins are squeezed, suppliers become more aware of the need for early life-cycle estimating
- National and International Software Metrics Associations and commercial conference organisers continue their good work of holding conferences which promote software process improvement and software metrics. National SMA's in particular are doing good work in developing standards for software metrics where none existed before, and in improving functional size measurement (for examples, see Refs. 14 and 15)
- Providers of estimating tools, software metrics repositories, and project management tools (e.g. for task planning and effort recording) are delivering more and better integrated solutions
- The COSMIC Core Team is seeking and taking every opportunity to promote its new method and to open up its use

All these developments auger well for more attention and more professionalism being given to software metrics in general and to Functional Size Measurement in particular.

9. CONCLUSIONS

We have traced the development of Functional Size Measurement from a very clever and pragmatic beginning through its inevitable hype, followed by disillusionment, and now its development and maturing in new forms. This maturing, particularly with the COSMIC FFP method, has the potential to make significant contributions to software requirements engineering, project management and estimating, software process improvement, software procurement and contract control, and to improving the understanding of the economics of software as an asset.

In this process of maturing, considerable new insights have been gained into software size measurement. The need now is for software managers and engineers to build on this new knowledge and experience, to tackle the many challenges that still lay ahead, to experiment with and promote these methods, and to reap the benefits for the software industry.

Come back Functional Size Measurement, all is indeed forgiven!

Software Measurement Services Ltd
143 High Street
Edenbridge, Kent TN8 5AX
England

Tel: +44 1737 763 674
Fax: +44 1737 761 166
E-mail: charles_symons@compuserve.com
Web: www.software-measurement.com

REFERENCES

1. A.J. Albrecht, "Measuring Application Development Productivity', presented at IBM Applications Development Symposium, Monterey, CA, 1979

2. A.L. Rollo, T Wright, 'Southern SCOPE – a Method for Acquiring Custom-Built Software, or Let the Customer Manage Software Acquisition', ESCOM Conference, London, April 2001

3. C. Jones, 'A Short History of Function Points and Feature Points', Software Productivity Research Inc., USA, 1987

4. C.R. Symons, 'Function Point Analysis: Difficulties and Improvements', IEEE Transactions on Software Engineering, 14(1), pp 2-11, 1998

5. S.A. Whitmire, '3D Function Points: Scientific and Real-time Extensions to Function Points', Pacific Northwest Software Quality Conference, 1992

6. A. Abran, J.M. Desharnais, M. Maya, D. St-Pierre, P. Bourque, 'Design of a Functional Size Measurement for Real-Time Software', Research report no. 13, Software Engineering Management Research Laboratory, Universite du Quebec a Montreal, Canada, November 1998

7. ISO/IEC 14143-1:1998. 'Software Measurement – Functional Size Measurement – Part 1 Definition of Concepts'

8. A. Abran, J.M. Desharnais, S. Oligny, D. St-Pierre, C.R. Symons, 'COSMIC FFP Measurement Manual, Version 2.0, Ed. S Oligny, , Software Engineering Management Research Laboratory, Universite du Quebec a Montreal, Canada, October 1999. Downloadable at www.lrgl.uqam.ca/ffp.html

9. A. Abran, C.R. Symons, S. Oligny, 'An Overview of the COSMIC FFP Field Trial Results', ESCOM Conference, London, 2001

10. P.G. Rule, 'Using Measures to Understand Requirements', ESCOM Conference, London, 2001

11. G. Jensen, 'Estimating Life-Cycle Effort and Duration', Australian Conference of Software Measurement, November 2000

12. C.R. Symons, 'Conversion Between IFPUG and MkII Function Points', obtainable from www.software-measurement.com

13. 'Capability Maturity Model – Integrated. Continuous representation ver 0.2b, IPD additions, Software Engineering Institute, Carnegie Mellon University, USA

14. UK Software Metrics Association ('UKSMA'), 'Quality Standards: Defect Measurement Manual, Release 0.i', obtainable from www.uksma.co.uk

15. Netherlands Software Metrics Association ('NESMA'), 'Definitions and Counting Guidelines for the Application of Function Point Analysis, Version 2.0, 1997