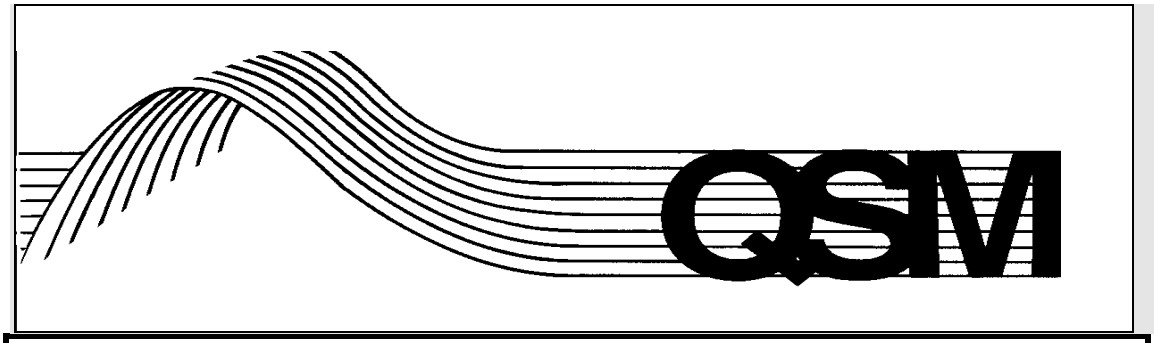

MEASURES FOR EXCELLENCE

Estimating
Software Size
and
Uncertainty

J.W.E Greene
QUANTITATIVE SOFTWARE MANAGEMENT LTD

7 rue Fenoux
Paris 75015
Tel: 33-140-431210
Fax: 33-148-286249
qsm.europe@ pobox.com

93 Blythe Road,
London W14 OHP
Tel: 44-171-603-9009
Fax: 44-171-602-6008
www.qsm.com

1 Introduction

Analysis of software project development data [Ref. 1, 2] shows that a clear correlation exists between the size of the software product and the development variables of time, effort and software errors. The relationship is a power function. This power function relates the software product size to the key development variables.

In organisations developing or purchasing software development it is necessary to estimate the expected size of the software product to be produced. Improved confidence results in the development estimates for producing the software by quantifying this major driver.

However there is always uncertainty in the final size of the software product until it is actually written. This needs to be recognised and quantified in terms of a size range that encompasses the lowest and highest estimate of the software to be developed. The size range technique described here is also a practical way of expressing uncertainty and hence determining the development estimate risk at any given point before the software is written.

Purchasing high technology systems increasingly involves software development. Many high technology companies now recognise that their business traditionally regarded as hardware dependent is now software dependent. [Ref. 3]. Any purchasing decision involving software acquisition needs to clearly identify all software that has to be developed by a Vendor. Increasingly Vendors expect to re-use existing software. This re-used or packaged software is modified and/or extended to meet the functional requirements of the purchasing organisation.

The need is to quantify the amount of software to be modified and extended. This quantification brings benefits that include :

- how well existing re-used software (or a package) meets the requirements
- the availability of software in particular sub-system areas
- the amount of modification required
- the amount of new software to be developed

For identical reasons the software size needs to be determined by development groups in order to plan their developments. For example Vendors such as systems houses when bidding for development work must estimate the software size and its uncertainty when preparing their development proposals. In these cases the same considerations apply regarding re-use or package evaluation.

Our experience indicates that the majority of software development projects today involve modifying and extending existing software. There are relatively few totally new software developments.

2 Sizing Objectives

The objective is to size all the software to be developed, including its uncertainty, in order to evaluate the development proposals made by vendors or to prepare realistic estimates in a development group.

Software is produced to meet a given system specification. Hence it is practical to identify the functional sub-systems that are required in order to meet the specification. The requirement analysis phase identifies the functional content. In a purchasing organisation the specification is frequently prepared internally and forms a part of the Invitation to Tender (ITT) issued to competing Vendors.

To meet the objectives of a purchasing organisation Vendors proposing systems that involve software development are required to :

- clearly identify all proposed software sub-systems that are to be delivered
- detail the sub-systems to give confidence in their stated size
- show how the proposed sub-systems match the requirement
- for each sub-system state if the software already exists and is to be provided unchanged together with the corresponding size
- where existing sub-system software is to be modified, state the size of the existing software and estimate the size range of the modification
- similarly estimate the size range of all proposed new software sub-systems
- quantify the total expected size of the software and its uncertainty
- declare this the baseline size for the proposed software development

This information is requested as part of a structured questionnaire [Ref. 4] that forms part of the Invitation to Tender (ITT). The questionnaire also requires the proposed plan for development including major milestone dates as well as the development environment that defines the proposed team, their skills and experience. The information returned by the Vendor enables an analysis to be made of the assumptions in their plan. This includes comparing the productivity assumed in the plan against industry reference measures [Ref. 5] as well as against measures made using data from their past developments [Ref. 4].

3 Software Size Definitions, Size Range and Data

Definitions of how to quantify software size are issued as part of the ITT questionnaire to give guidance to Vendors. The definitions emphasise the high level quantification of the software and stress the interest is to quantify in macroscopic terms the expected size and uncertainty.

Sizing is required to be expressed in effective lines of code (ELOC) which simply means what the development team expect develop and deliver. Definitions clarify what is to be included and excluded with respect to estimating ELOC size.

For example the ELOC estimates do not include :

- macro expansions
- comments
- included code copied from libraries
- re-used code

A key point is to explain to the Vendors that they should use the size range to express the uncertainty based on the completeness or otherwise of the specification set out in the ITT. Where the specification is imprecise then the Vendor can express this uncertainty as a larger size range.

Each proposed sub-system to be delivered as part of the final product must be identified. The language used together with the size of existing software and size range of modified or new software must be stated.

When responding to the ITT Vendors have complete freedom to amend the number of sub-systems proposed. However these changes to the number of sub-systems must be clearly identified. The functional requirement sub-systems provided in the ITT must be clearly shown with regard to how the changes proposed match the requirement. In this way the Vendor is free to reduce or increase the number of proposed sub-systems.

Constraints are, however, imposed on the size of each sub-system requiring modification or development. The Vendor is required to break down into more detail and identify the sub-systems so that each does not exceed an estimated mean size of 3,000 effective lines of code.

Some companies are able to quantify the size range in terms of Function Points, FP. This is often the case in business system developments. Simple ratios are known that allow each size expressed in FP to be expressed in ELOC. However we find that in many cases insufficient internal design is known to allow a realistic size range to be made in FP.

The size data returned for each sub-system consists of :

- Sub-system identity
- Re-used (Packaged) Size (r)
- Minimum expected ELOC size (a)
- Most likely expected ELOC size (b)
- Maximum expected ELOC size(c)

This size data is then summed to give the mean size of the software and the standard deviation. The result expresses the software product size and uncertainty.

4 Sizing Re-used - Package Software

When evaluating development proposals it is vital to identify how the proposed re-used or package software meets the requirement. There are two reasons for requesting this information, one technical the other commercial.

Technically there has to be a clear statement of what software is available "off the shelf" and how well or otherwise it meets the requirement. By requiring all proposed sub-systems to be declared and sized it is practical to confirm precisely what existing software is being proposed and where it meets the requirement. The availability and suitability of the re-used software has to be confirmed as part of the proposal evaluation.

A commercial consideration can involve intellectual property rights with respect to the amended and new software. Frequently the purchasing organisation having paid for development expects that ownership of that part of the software now rests with them. A commercial agreement can be reached regarding ownership by quantifying the re-used software supplied by the Contractor and that developed and paid for by the purchaser. At its most simple a ratio is established that enables the purchaser to profit from any future sales by the Contractor that involves re-use of the purchasers software.

5 Development Baseline

To meet the technical and commercial objectives outlined above an agreed software size baseline is established. This is calculated statistically to reflect the mean size and the corresponding uncertainty by summing over all the sub-systems to give the total size and uncertainty.

For each sub-system (s) the skewed mean value (sm) is calculated using the formula :

(a+4b+c)/6 (sm) together with the standard deviation (c-a)/2 (sd)

The overall total mean size is then calculated together with the corresponding standard deviation by summing over all the sub-systems using the formula :

Total Mean Size = Σ sm Standard deviation = $\sqrt{(\Sigma(sd)^2)}$

This is used to compute the 99% size range of +/- three standard deviations around the mean size.

The baseline size of extensions to the existing software covering modified and new code are quantified by requiring each sub-system to be sized . This baseline size and uncertainty forms part of the contractual conditions when development is awarded to the winning Vendor. Subsequently the baseline forms an essential part of monitoring development progress and contractor performance.

Most importantly the quantified size baseline is used to quantify and negotiate any additional requirement changes while development is in progress. Requirement changes occur frequently after awarding development contracts. Both the development contractor and the purchaser need the quantified baseline on which to negotiate any requirement changes.

6 Case Study

To show the practical results from employing the techniques described we use a case study. The case study results shown below are from an ITT issued to 6 Vendors (A - F). The major sub-systems expected to be supplied were identified before issuing the structured questionnaire that requested the size information in terms of the three-point size range estimate. In total 21 sub-systems were listed. These sub-systems were identified based on the requirement specification that was produced using SSADM.

The Vendors were required to state where they expected to re-use existing software and the size of this existing software. If a sub-system was to require modification to the existing software and/or new software then the size range was required to be estimated.

The responses from each Vendor are shown in Table 1. (Note that in Table 1 notional size ranges of 1,2, 3 are entered for the sub-systems where the Vendor claims the function is subsumed into another sub-system)

7 Summarised Results and Observations

The overall size range estimates are summarised as follows :

Vendor Identity	Number of Sub-systems	Total Size Re-used Code	Mod./New Size Mean	Std. Dev.	Total Re-Use+Mean
A	**21**	**0**	**18049**	**1000**	**18049**
B	**26**	**8150**	**9040**	**200**	**17190**
C	**27**	**3900**	**3751**	**100**	**7651**
D	**21**	**4500**	**14542**	**850**	**19042**
E	**19**	**49800**	**5282**	**400**	**55082**
F	**13**	**7985**	**2995**	**220**	**10990**

From the summarised information it becomes practical to identify how consistent are the size estimates between the Vendors.

Vendors A, B and D are estimating the total size that includes re-used software as between 17000 and 19000. Vendor A expects to develop the

entire software with no re-use. Vendors B and D anticipate software re-use with Vendor B claiming more "off the shelf".

In contrast Vendors C and F are estimating size at between 7600 and 11000. Vendor E is proposing a large amount of software re-use, 49800, with a small increase of modified and new code amounting to 5280 ELOC.

One factor examined to account for differences in the size estimates from each Vendor is programming language. Higher level languages such as Fourth Generation Languages, Ada or C++ require fewer ELOC to implement a given amount of function. Knowing the language also assists in determining the expected performance characteristics where there may be concerns related to response times and process capability particularly in real-time systems.

Part of the information requested for each sub-system is the programming language that is to implement the function. In all the size estimates shown above the same object oriented language was proposed by each Vendor.

Investigation raised questions on the lower size range estimates put forward by Vendors C and F. Vendor E was able to show the existence of the large amount of "off the shelf code" and how it covered the functional requirements.

The other three-size range estimates from Vendors A, B and D shows the high degree of consistency that is achieved. Each of these Vendors is quite independent with regard to the software available from re-use and their size range estimates.

8 Using The Size Data

The size data is used in a variety of ways to assist in the evaluation of each development proposal.

It enables a clear distinction to be made between available "off the shelf" software and the amount to be developed as modifications and/or new software.

The "goodness of fit" to the requirement specification of this off the shelf software is made visible by checking with each Vendor the claims of proposed available software. This requires the vendor to demonstrate to the evaluation team that an existing operational system really does support the sub-system function as claimed in the proposal.

An evaluation is also made of the high-level development plan data put forward by the vendor. This data consists of the proposed development time, the development effort and the mean size and uncertainty of the modified and new software. Using these three values the assumed process productivity in the plan is measured and compared to that achieved in similar

past projects from the vendor. All these process productivity measures are in turn assessed against know industry reference measures [Ref. 5].

Letting the development contract to the winning vendor now includes a declared development size baseline. Keep in mind that size uncertainty is inevitable until the modified and new software is produced. This uncertainty is quantified and used to determine the risk envelope around the development plan.

Each month high-level progress data is returned by the contractor that enables progress to be determined within the uncertainty envelope. Variance analysis detects if the progress is going outside the agreed limits. In such cases re-planning is performed using the actual measure of process productivity achieved by the contractor at that time. [Ref. 6,7].

One of the most important uses of the development size baseline is to deal with requirement changes that arise while development is in progress. Such changes are common. However the ability of the contractor to absorb such changes is limited without changes being negotiated and agreed leading to a revised development plan.

With the agreed baseline in place all change requests are sized and used to determine their potential impact on the development. By examining the consequences of accepting the change requests the impact on the agreed development plan can be evaluated. This enables an objective negotiation to be held between the client and the contractor that reflects the current position in the development project.

If the client sees such change requests as essential then a revised development plan can be agreed. This includes revised development time scales and costs as well as the revised size baseline. Alternatively the impact of the change request is such that a decision is made to postpone the change request for a future release. Usually this new release follows immediately as a further development project once the current is complete and accepted.

9 Conclusions

To ensure estimates of software development time and effort are realistic the size of the software product must be estimated. In practical terms' estimates of size have to recognise uncertainty. These uncertainties in turn are a reflection of how well the specifications are defined.

By adopting the engineering approach described in the paper purchasing and development managers are able to quantify the expected size of the software and its uncertainty. Significant benefits flow from this quantification.

In a purchasing organisation the evaluation of the availability of "off the shelf" software can be closely matched to the requirement's specification. This

enables the actual availability and suitability to be checked as part of the tender evaluation process. Each vendor's proposal is assessed in part by requiring the formal sizing of the proposed software.

In this way a clear baseline is established that represents the software to be modified and developed on behalf of the purchaser. If necessary the baseline is used to judge intellectual property ownership. In addition the baseline, including the uncertainty, allows the impact of proposed requirements changes to be assessed while development is in progress. Given that such changes are accepted it is then practical to negotiate a revised development plan and a new baseline.

The situation in a development organisation parallels that in purchasing. Here the need is again to quantify the size of software when making development estimates. Commercially it is vital that the development estimators quantify this key driver particularly when fixed price contracts are involved. With a clear statement of the expected size and uncertainty the development estimates can justify the risk protection set out in the proposed time and effort.

Similarly the development is protected against requirements' changes that seek to increase the size of the software without re negotiating the time and effort. This equips both the developer and purchaser with the means to re negotiate requirement changes in an objective and realistic way.

The dominance of software in high technology systems makes it mandatory to quantify size wherever software development is involved. The case study shows the practicality of estimating size. Quantification of the size uncertainty is equally important. Together these values permit realistic risk protected estimates to be produced of development time and effort.

References

Ref. 1 : Putnam L.H. Software Cost Estimating and Life Cycle Control IEEE EHO 165-1
Ref. 2 : Walston C.E. and Felix C.P. IBM Systems Journal Volume 16 No. 1
Ref. 3 : Nouvelle Observateur ALCATEL Business Systems Group January 1992
Ref. 4 : Greene J.W.E.Software Acquisition : Tender Evaluation Method ERA Avionics Conference 1992
Ref. 5 Putnam L.H. Measures For Excellence Prentice Hall ISBN 0-13-567694-0
Ref. 6 : Greene J.W.E Comment Controle un Projet de Developpement Logiciel qui Derape EC2 Genie Logiciel Toulouse No 21 December 1990
Ref. 7 : Greene J.W.E. The Software Control Office EC2 Genie Logiciel Toulouse No 22 December 1991

Table 1 Vendor Size Range Estimates

		A					B					C		
Sub-Sys	Re-use	Low	Most Likely	High	Sub-Sys	Re-use	Low	Most Likely	High	Sub-Sys	Re-use	Low	Most Likely	High
1		1200	1500	2000	1		180	200	240	1		150	200	300
2		1	2	3	2		1	2	3	2	200	1	2	3
3		400	500	2000	3		1	2	3	3		150	250	300
4		1	2	3	4		1	2	3	4		125	200	250
5		1	2	3	5		1	2	3	5		125	200	250
6		1	2	3	6		1	2	3	6		125	200	250
7		1	2	3	7		1	2	3	7		250	300	350
8		1	2	3	8		1	2	3	8		1	2	3
9		800	1000	2000	9		1	2	3	9		1	2	3
10		1000	1500	2500	10		1	2	3	10		450	600	700
11		1	2	3	11		200	250	350	11		100	150	200
12		1	2	3	12	60	240	300	340	12		1	2	3
13		500	1000	1300	13	300	1	2	3	13		1	2	3
14		300	500	700	14		400	500	600	14	500	1	2	3
15		500	1000	2000	15	1650	750	1000	1530	15	1500	1	2	3
16		1200	1500	3000	16	940	1	2	3	16		350	500	600
17		4000	5000	8000	17		1	2	3	17	500	1	2	3
18		400	500	1000	18		1	2	3	18		1	2	3
19		1300	1500	3000	19	1600	1750	1800	1850	19		250	350	450
20		400	500	1000	20		75	100	150	20		250	300	400
21		800	1000	1600	21		75	100	150	21		350	500	600
					22	1000	1700	1800	2400	22	300	1	2	3
					23	900	1150	1200	1250	23	400	1	2	3
					24	1000	1050	1100	1200	24	500	1	2	3
					25	400	1	2	50	25		1	2	3
					26	300	450	500	550	26		1	2	3
										27		1	2	3
99 % Range Re-use	0	15090	18049	21009	99 % Range Re-use	8150	8484	9040	9600	99 % Range Re-use	3900	3450	3751	4052

		D					E					F		
Sub-Sys	Re-use	Low	Most Likely	High	Sub-Sys	Re-use	Low	Most Likely	High	Sub-Sys	Re-use	Low	Most Likely	High
1		500	800	2000	1		1	2	3	1	668	1	2	3
2		300	500	1500	2		1	2	3	2		1	2	3
3		200	400	1000	3	4000	1	2	3	3	110	1	2	3
4		200	300	800	4	200	1	2	3	4		1	2	3
5		200	300	800	5	200	1	2	3	5	400	1	2	3
6		300	400	800	6	200	1	2	3	6	80	1	2	3
7		200	300	800	7	200	1	2	3	7	6127	300	350	500
8	250	200	300	800	8	200	1	2	3	8		100	150	200
9		200	500	800	9	1000	1	2	3	9		200	250	350
10		350	500	800	10	1000	1	2	3	10	600	300	400	550
11		250	500	800	11	1000	1	2	3	11		700	1100	1800
12	500	250	500	1200	12	800	1	2	3	12		150	200	350
13	250	600	800	2000	13	1000	1	2	3	13		200	400	800
14	250	500	700	1500	14	40000	1	2	3					
15		400	700	1500	15		1500	2000	3000					
16	500	400	650	1800	16		1500	2000	3000					
17	1000	1000	1400	3000	17		1	2	3					
18	500	700	1000	2500	18		1	2	3					
19	1000	700	1000	2500	19		500	1000	2000					
20		400	500	1500										
21	250	200	400	1000										
99 % Range Re-use	4500	11980	14542	17103	99 % Range Re-use	49800	3983	5282	6581	99 % Range Re-use	7985	2334	2995	3655

[Notional Sub-System Size Ranges 1,2,3]