Full citation: MacDonell, S.G. (1997) Establishing relationships between specification size and software process effort in CASE environments, Information and Software Technology 39(1), pp.35-45.
doi: 10.1016/0950-5849(96)01125-1

Establishing Relationships Between Specification Size and Software Process Effort in CASE Environments

Stephen G. MacDonell

Software Metrics Research Lab
Department of Information Science
University of Otago, PO Box 56
Dunedin, New Zealand
stevemac@commerce.otago.ac.nz

Abstract

Advances in software process technology have rendered some existing methods of size assessment and effort estimation inapplicable. The use of automation in the software process, however, provides an opportunity for the development of more appropriate software size-based effort estimation models. A specification-based size assessment method has therefore been developed and tested in relation to process effort on a preliminary set of systems. The results of the analysis confirm the assertion that, within the automated environment class, specification size indicators (that may be automatically and objectively derived) are strongly related to process effort requirements.

1. INTRODUCTION

Recent and ongoing studies in the areas of software metrics, software process assessment and software process automation have generated similar ideas in relation to process effort estimation[1,2]. Much of this work suggests that both the data collection procedures required for effort prediction and the estimation of process effort itself should be:

- performed as early as is feasible to obtain estimates *within specified bounds of accuracy*;
- automated as much as possible;
- as objective as possible.

These suggestions are commonsense, but it has taken metrics research a relatively long time to acknowledge the importance of such issues. This is due in part to the rather chaotic approach to development that was common until management of the software process was acknowledged as important in its own right, and in part to the lack of appropriate tools that would enable objective automatic data extraction. Prior to this, data collection and subsequent estimation were often performed on an *ad hoc* basis involving substantial manual procedures, making them highly susceptible to subjective opinions in assessment.

Requirements such as the three listed above, as well as advances in process automation via computer-aided software engineering (CASE) tools and application generators, have lent support to the development of effort estimation models based on relevant aspects of requirements specifications. Using extensive statistical analyses, Mukhopadhyay and Kekre[1], for example, were able to show that, as well as holding the inherent advantage of early derivation, their feature-based effort estimation models proved to be more effective than several other well-known techniques. Although having a similar aim to the Mukhopadhyay and Kekre work, the study reported here has been undertaken in the transaction processing/management information systems (TP/MIS) domain, and is an initial evaluation of a previously proposed assessment and estimation scheme[3].

As a characteristic of all software development artifacts, product size has for some time been generally acknowledged as having a significant impact on other important product attributes, including quality and maintainability[4,5]. Product size has also been recognised as an influential factor concerning the effective *management* of the software process. This is a result of the expectation that, in general, a larger piece of software will require greater development effort, will contain more errors and will be more difficult to maintain and enhance[6,7,8]. Ultimately then, overall development and maintenance costs are affected by product size[9,10]. Although effort may also be influenced by other factors, such as the productivity of development personnel, it is suggested here that system size is the *main* determinant. Once a specification is complete, facilities for schema and code frame generation available in automated environments reduces the extent of developer input[11,12], thus reducing (but not yet eliminating) the impact that individual developers have on this transition process. Attention should therefore be focused on the functional size, hence our concentration on assessing requirements specifications.

Furthermore, DeMarco[13] suggests that implementation effort increases with proportional increases in specification size, and that specification size approaches invariance with respect to the decisions taken by individual developers. For a particular set of requirements, this would therefore suggest that similar size measures would be derived no matter who developed the specification, particularly in an automated development environment[14,15]. In essence then, it is asserted that system size, as represented in a requirements specification, is the overriding cost driver in terms of development effort in an automated environment.

The establishment of useful objective relationships between size and effort/cost at an early stage of the software process is clearly a desirable outcome of the assessment process. The observations of Mukhopadhyay and Kekre[1] concerning the limited usefulness of most effort estimation models due to their late derivation are also supported here, providing the motivation for specification-based assessment and prediction. The effective application of a large number of existing assessment methods is also impeded by other problems - some are oriented more to the assessment of the development methods used and the individual style and ability of programmers than to the actual scope of the software, and several are less than comprehensive in their assessment[16,17].

In general, quantitative software assessment involves the extraction of counts of various product and process attributes, based on the assumption that these counts are useful in their own right, and that they may also be useful in determining or estimating other development attributes. Thus the aim of many proposed size assessment techniques has been to provide product-based predictions of attributes such as development effort or post-implementation error frequency. Given progress in process automation, the functional metric approach, in which measures are extracted from some early functional representation(s) of a system (as opposed to an implementation-oriented representation), appears to hold promise for size assessment and effort model development. The increasing use of application generators and CASE tools presents an opportunity for the development of assessment techniques that can overcome at least some of the problems associated with other metric classes. Due to the degree of automation that these tools provide, the transformation from a system's functional requirement to its implementation is more straightforward[18,19], thus reducing (to some unknown degree) the impact of specific development personnel and implementation methods. Furthermore, the multi-dimensional nature of many CASE specifications enables the assessment of system product size from a number of perspectives, leading to a more comprehensive consideration. Requirement representations are also among the first tangible products of the software development process, so models developed from them are likely to be among the earliest available. Measures derived from functional representations therefore have the potential to be useful in comparing the scope of complete systems, and in assessing the impact that variation in size has on outcomes of the software process.

Two of the most widely cited functional approaches to product assessment are the originally defined version of Function Point Analysis (FPA)[20] and System Bang[13]. In terms of the requirements listed at the start of this paper, these techniques would appear to be unsatisfactory[18,21,22] [21]. Tate and Verner[21] , for example, observe that FPA is difficult to determine automatically, and can involve a significant degree of subjectivity. Furthermore, they remark that some of the information that is required for the calculation of function points is not available from CASE tools. System Bang also demands that a number of subjective assessments be made in its formulation, reducing the general applicability of the results obtained.

An assessment/estimation approach that attempts to overcome some of the problems associated with previously proposed methods, and one that is consequently designed to satisfy the requirements of an effective approach, has been proposed. This approach is described in the next section. (For complete details of its derivation and justification, the interested reader is referred to the original paper[3].) Empirical evaluation of the approach using thirteen systems from ten commercial sites is described. The paper is then concluded with a summary of the findings and recommendations for continuing research.

2. SPECIFICATION SIZE ASSESSMENT

It has been suggested[13,21] that the structured and semi-formal nature of some widely used specification models lends itself to objective size assessment, and may provide a useful basis from which to estimate other product and/or process attributes. The extensive use of such requirements modelling methods in computer aided software engineering (CASE) environments has enabled automatic extraction of measures derived from these representations. Use of these tools has also led to a shift in emphasis from assessment of implementation products (e.g. lines of code, program volume and so on) to the assessment of functional products. Hence, our basis for size assessment is in the outputs of the requirements analysis process.

After applying the Goal/Question/Metric (GQM) and Classification Scheme paradigms[23,24] to the aims of the study, five specification perspectives were selected as being quantifiable in terms of the contribution that each might make to the overall size of a complete specification: data, processing, transaction, function and user interface. Given that software requirements are often specified using entity relationship diagrams (ERDs) and data flow diagrams (DFDs), Tate and Verner[21] suggest that they are appropriate partial representations for early-phase assessment. Both approaches have a data focus, but each provides a differing view of the data: the former emphasises the structural perspective of the data, whereas the latter is more concerned with giving an overall representation of the use and transformation of data. Consideration of just one of these two approaches, in terms of gaining a full and complete picture of system size, is unlikely to be adequate. In fact, the development of a specification is often an iterative process that takes the analyst from one technique to the other in order to derive a 'complete' specification.

Although the data structure and data flow modelling specification methods are the most widespread in the commercial systems domain, they are not totally exclusive of other methods, and are in fact frequently augmented by other representations as these too provide complementary views of system requirements. The three secondary representations considered here are the transaction and user interface representations and the functional decomposition hierarchy (FDH).

The transaction representation is widely employed in the development of database-centred systems common in the business domain. Each elementary function in a system may be considered in terms of the operations that it performs on entities. Given adequate decomposition, low level functions or processes may be specified in this manner, providing assistance for the subsequent development of processing logic. Thus the transaction representation may be considered as a model that combines both data and process requirements.

The user interface representation is a perspective that is particularly applicable to development projects in which prototyping methods and graphical user interface (GUI) development tools are used, as the development of an acceptable interface can be a significant component of the overall effort expended in this type of environment. This representation essentially provides models of the screen and report formats that are to be subsequently implemented in the system. Interactive business systems tend to make extensive use of screen dialogues and are oriented towards the production of management reports. Thus a consideration of system scope as represented in this form should provide further information to enhance process attribute assessment.

A functional decomposition hierarchy is often produced as a levelled description of the functions to be provided by a system, providing a concise representation of the projected module calling structure that will eventually be generated or constructed. The number and interaction of the modules are likely to have an impact on system scope so this representation should also be considered in any functional assessment scheme.

Although there are certainly a number of development methodologies that do not use the above tools and techniques it would still seem reasonable to suggest that automated structured specification and development methods of this type will continue to be used within the business community for the foreseeable future. This is due to a number of reasons, including vendor commitment to automation, user investment in tools and methods and, to a lesser extent, tradition. It is therefore envisaged that the analysis strategy, which assumes the existence of such an environment, will not become obsolete in the near future.

It was considered that quantification of aspects of each representation would help to ensure that the assessment of system scope was as comprehensive as possible[3]. During site interviews, however, it was found that just three of the software processes examined made extensive use of data flow diagrams (DFDs), so the processing perspective (covered by DFD-type models) was disregarded for this analysis. Although this finding may be taken to suggest that DFDs are now uncommon, it is more likely that it is simply a reflection of the particular tools that were encountered in this study. A recent assessment of several CASE products[25] showed that DFD assistants were still among the most widely available features of automated tools. Future studies with systems developed using this and other tools should therefore provide useful results relating to the influence of process model factors.

Figure 1 is an adapted diagrammatic representation of the CASE product model proposed by Tate and Verner[21] as a basis for assessing specification size. The tasks and activities necessary for the production of such artifacts are considered to be typical of software processes used in the automated development of business-oriented transaction processing and reporting systems.

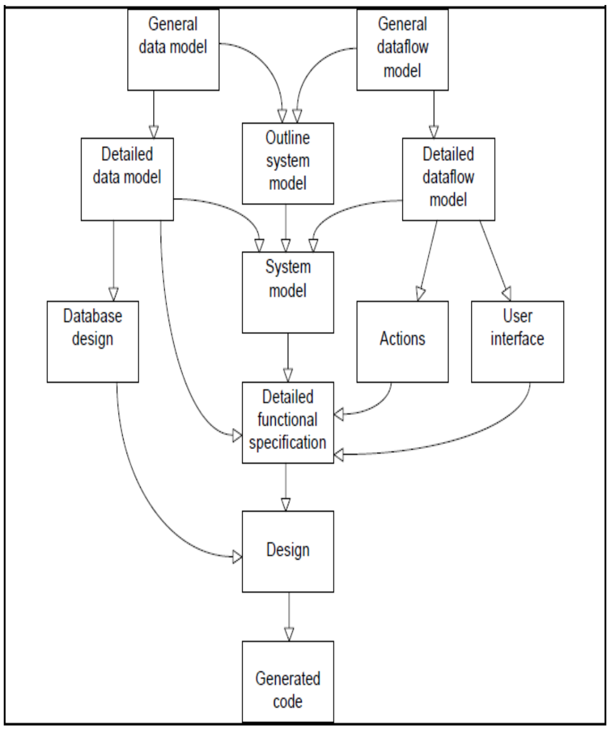


Figure 1. Tate and Verner's CASE product model

As this study was concerned only with the assessment of those products developed within the requirements analysis and specification activities, and since the data flow-oriented products were to be disregarded, a revised product model for this study was developed. This model is shown in Figure 2.

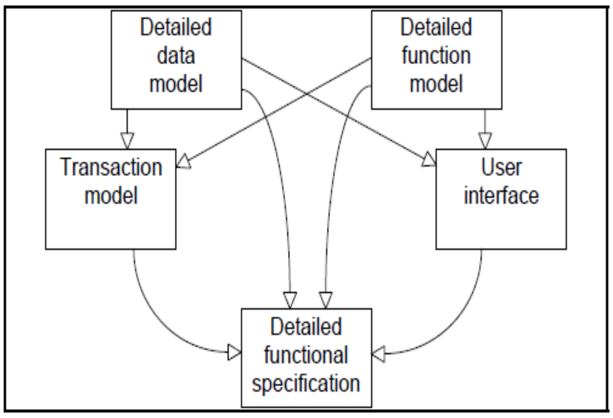


Figure 2. The automated development product model adopted in this study

Those components shown in Figure 2 that are the same as those in Tate and Verner's original proposal (that is, the detailed data model, the user interface and the detailed functional specification) have the same general descriptions. The additional components suggested in this work are similarly described, as follows:

Detailed data model - a fully defined data model incorporating attribute specifications. This model enables measures to be derived relating to the size of the data representation, including characteristics of entities, attributes and relationships.

Detailed function model - similar to Tate and Verner's detailed dataflow diagram, this is a fully decomposed hierarchy to the level of functional primitives, enabling measures relating to processing scope and transaction interaction to be derived.

Transaction model - detailed definitions of attribute manipulations at the functional primitive level, bringing together aspects of system size from both the data and processing perspectives.

User interface - this representation includes menus, screens, reports and messages in detail, enabling size assessment of relevant characteristics. Tate and Verner suggest that this system aspect may contribute significantly to implementation effort in business systems, and may be treated as another system dimension partially derived but quite distinct from those of data and process.

Detailed functional specification - this is an overall view of the system as specified in the previous products. It is not dealt with here in detail, but illustrates the bringing together of the several dimensions of a system into one specification. As such, Tate and Verner suggest that it should be a useful predictor of both total design effort and implementation effort.

There may be some question as to whether the products just described can all be considered to be outputs of the requirements analysis and specification phase. It is certainly acknowledged that in some circumstances the transition from analysis to design can become somewhat blurred, as for example, in a prototyping process. The requirements specifications provided for this study, however, included all four components listed above. This may be indicative of the components and techniques that are generally employed in requirements analysis and specification in automated environments. Specific measures relating to each of the model components are now described in detail.

2.1. Transaction measures

Low level transactions in database systems perform one of the following operations: create a record, read a record (including look-up validation), update a record or delete a record. There is some evidence to suggest that the operations make differing contributions to overall system scope and complexity[3]. Given that this representation combines both data and functional requirements[26] it may provide a sound basis for comprehensive size indications. The measures from this representation collected in this study are defined in Table 1.

Table 1. Transaction measures

TCR	Total number of create transactions performed by the system
TRE	Total number of read transactions performed by the system
TUP	Total number of update transactions performed by the system
TDE	Total number of delete transactions performed by the system

2.2. Functional model measures

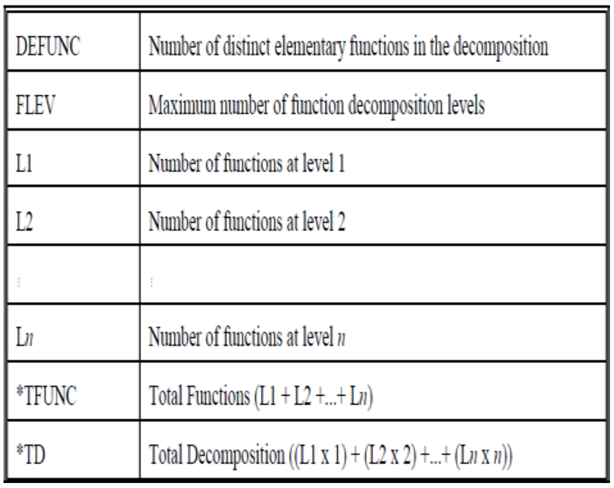
Functional decompositions of some form are central to most development approaches[10,27]. In cases where the functional model is broken down to an elementary level, this representation can provide a quantitative insight into the scope and complexity of the specified system. Functional model size indicators should form a useful basis for developing transaction design and implementation estimates within automated development environments[21]. More specifically, it is suggested here that the number and interaction of functional modules will have an impact on overall system size (and consequently process effort) - this provides the motivation for inclusion of the measures shown in Table 2. (Measures marked with an asterisk are composite measures that is, they may be calculated from the values of other base measures.)

2.3. User interface measures

Particularly for software development in an interactive processing environment, the number of screens, reports and data elements produced for the user is expected to have a significant impact on development effort, as the creation of acceptable screen and report formats is often a major part of interactive (business) system production

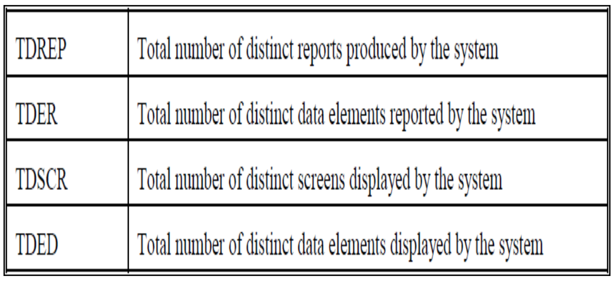
[21,28,29]. As such they should provide a good basis for the development of effort estimates for user interface-related development tasks [21]. Interface measures are defined in Table 3.

Table 2. Functional model measures



DEFUNC	Number of distinct elementary functions in the decomposition
FLEV	Maximum number of function decomposition levels
L1	Number of functions at level 1
L2	Number of functions at level 2
⋮	⋮
L*n*	Number of functions at level *n*
*TFUNC	Total Functions (L1 + L2 +...+ L*n*)
*TD	Total Decomposition ((L1 x 1) + (L2 x 2) +...+ (L*n* x *n*))

Table 3. User interface measures

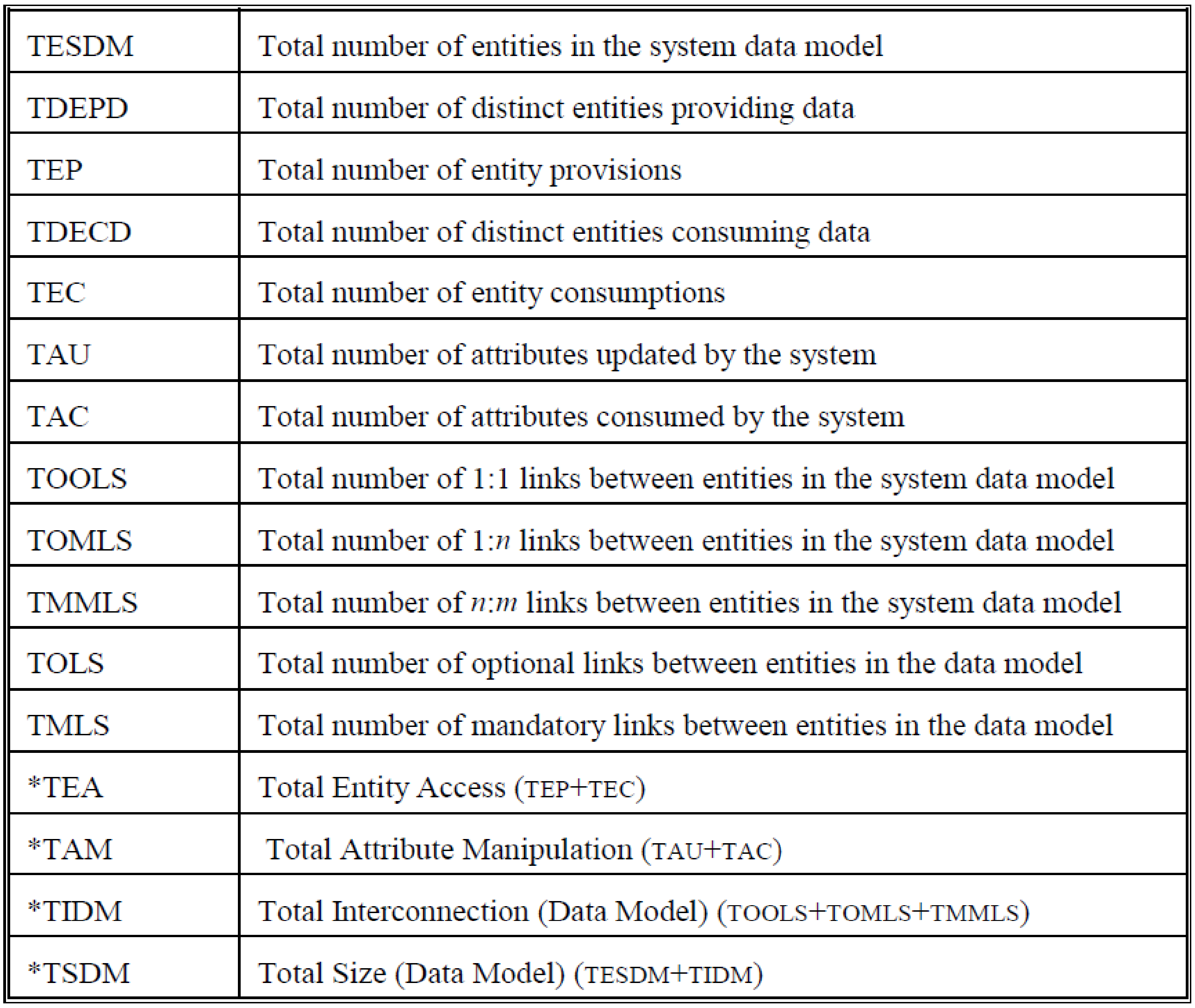


TDREP	Total number of distinct reports produced by the system
TDER	Total number of distinct data elements reported by the system
TDSCR	Total number of distinct screens displayed by the system
TDED	Total number of distinct data elements displayed by the system

2.4. Data model measures

Measures concerned with the size, interconnection and manipulation of data model representations are listed in Table 4 of the assessment scheme, reflecting the assumption that larger, highly interconnected data models, and higher numbers of accesses to entities and attributes, imply a larger and more complex processing system [26,30]. Indicators of data model scope should form a useful basis from which to estimate subsequent database-related task effort [21].

Table 4. Data model measures

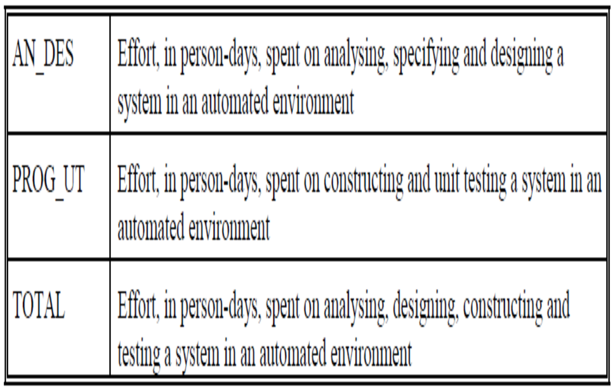


TESDM	Total number of entities in the system data model
TDEPD	Total number of distinct entities providing data
TEP	Total number of entity provisions
TDECD	Total number of distinct entities consuming data
TEC	Total number of entity consumptions
TAU	Total number of attributes updated by the system
TAC	Total number of attributes consumed by the system
TOOLS	Total number of 1:1 links between entities in the system data model
TOMLS	Total number of 1:*n* links between entities in the system data model
TMMLS	Total number of *n*:*m* links between entities in the system data model
TOLS	Total number of optional links between entities in the data model
TMLS	Total number of mandatory links between entities in the data model
*TEA	Total Entity Access (TEP+TEC)
*TAM	Total Attribute Manipulation (TAU+TAC)
*TIDM	Total Interconnection (Data Model) (TOOLS+TOMLS+TMMLS)
*TSDM	Total Size (Data Model) (TESDM+TIDM)

3. EMPIRICAL EVALUATION

One of the assumptions underlying the use of the proposed approach was that higher values of the various specification measures would indicate systems that were more time-consuming to develop. To empirically evaluate the assessment scheme, this assertion had to be tested quantitative indicators of process effort were therefore required. For the purposes of this study, the relevant indicators were defined as shown in Table 5.

Table 5. Process effort measures



AN_DES	Effort, in person-days, spent on analysing, specifying and designing a system in an automated environment
PROG_UT	Effort, in person-days, spent on constructing and unit testing a system in an automated environment
TOTAL	Effort, in person-days, spent on analysing, designing, constructing and testing a system in an automated environment

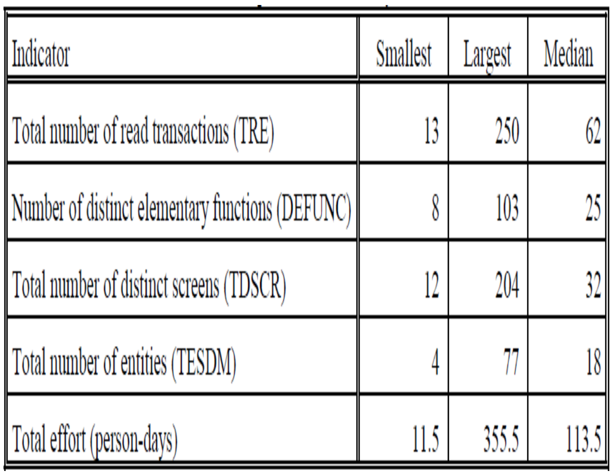
The effort indicators are derived from various well-supported assumptions concerning the intuitive relationship between relative system size levels and development effort[6]. The effort measures therefore reflect the amount of work carried out by personnel using CASE tools and application generators over various phases of development.

3.1. Systems analysed

After an extensive mailing campaign, ten business and government organisations agreed to provide systems for the project. Most agreed to allow one system only to be analysed, giving an overall sample of thirteen systems. The small sample size precluded any realistic opportunity to undertake both relationship development and subsequent validation but it was still hoped that the results obtained would prove to illustrate the feasibility and potential of objective, automated and early product size assessment as a basis for process effort estimation. Moreover, the use of small samples is not uncommon in first-cut analyses of assessment and estimation approaches[31-34] (although larger samples would always be preferred when they are available).

The ten organisations that agreed to participate in the study varied in size and function, from multinational petroleum manufacturers and distributors to government departments, through to small private commercial development sites. Tools used in development included Oracle CASE, AutoMate Plus, IEW/ADW, the IEF, Excelerator and ProKit Workbench/ProIV. The thirteen systems in the sample performed a number of overall TP/MIS functions, including customer and supplier recording, costing and charging, accounting, site and personnel administration, scheduling and rostering. Collection of the specification size data items was performed manually from various original requirements phase documents, that is, ERDs, FDHs, transaction models and screen and report formats. The project data relating to development effort was gathered from a combination of on-line and paper-based records that had been kept as part of the organisations' routine project management procedures. In order to illustrate the scale of the systems examined, Table 6 provides some general descriptive material. (Readers are encouraged to contact the author if they would like to receive a full set of the collected data.)

Table 6. Descriptive indicators of system scale



Indicator	Smallest	Largest	Median
Total number of read transactions (TRE)	13	250	62
Number of distinct elementary functions (DEFUNC)	8	103	25
Total number of distinct screens (TDSCR)	12	204	32
Total number of entities (TESDM)	4	77	18
Total effort (person-days)	11.5	355.5	113.5

Although factors such as specific tools and project personnel varied over the sample, other potential contributors to process effort (apart from system scope) were reasonably consistent. This included a common baseline software process, centred on structured analysis and design using automated tools, and a common application domain. Tate and Verner[17] suggest that, in an automated environment, size measures taken from specifications are less tool-dependent than those taken from lower level software products (e.g. programs and the like), so the influence of particular tools within the CASE class (particularly given the use of a common analysis and design methodology) should have been reduced.

3.2. Analysis results

Correlation procedures identified a number of highly significant associations between variables from all of the specification perspectives and the effort indicators. Many of these relationships were significant at the $\alpha = 0.001$ level; that is, there was less than 0.1% probability that the relationships had been encountered by chance. Since the Spearman correlation coefficient is said to be conservative, except in cases where ties are common[35], it was decided that further analysis would be carried out only on variables that showed highly significant values for *both* the Spearman and Pearson statistics. The specification-based size variables chosen based on this criterion were TRE (total number of read transactions), TD (total decomposition) and TDSCR (total number of distinct screens), plus eight of the data model measures

(TESDM, TDEPD, TAU, TAC, TMLS, *TAM, *TIDM, *TSDM).

Another variable selection method was then employed to ensure that interrelated variables did not go forward for use in further (goodness of fit) tests. Kitchenham and Pickard[35] suggest that closely related predictor indicators should be treated with caution when used together, especially when the overall objective is the development of estimation models. It is often the case that one of a group of interrelated variables is sufficiently powerful to act for the group. In these circumstances, criteria other than the original correlation coefficients should be used to select appropriate independent variables from related groups. In cases where the data are normally distributed and the sample size is sufficiently large, some form of factor analysis may be useful in determining an appropriate representative variable. Full normality in software engineering data distributions is uncommon, however[35]. Hampel et al.[36] suggest (in a general discussion of the topic) that there is practically always no guarantee of normality and that slight departures from the model have a significant effect on the results obtained. Moreover, the data set in this study consisted of just thirteen observations. It was therefore decided that variables should be selected from groups according to their ease of extraction and the time at which they became available - that is, variables that are easily determined and are available as early as possible were to be preferred over more complicated, later-phase variables.

Correlation tables illustrated the significantly high degree of intercorrelation within the group of variables from the data perspective (see Table 7). The eight data model measures listed above were all very highly correlated, except for the TAU (total attributes updated) and TAC (total attributes consumed) variables. Since these two variables were easily extracted, were elementary rather than composite, were available reasonably early in the development process (as part of the functional specification), and appeared to be relatively independent but still highly correlated with the effort indicators, they were both selected for separate use in the procedures to follow. For the current sample this led to a final set of prospective predictive specification variables: TRE (total number of read transactions), TD (total decomposition), TDSCR (total number of distinct screens), TAU (total number of attributes updated by the system) and TAC (total number of attributes consumed by the system). A summary of the correlation test results for these and the process effort variables is provided in Table 8. All correlation coefficients were significant at the α = 0.001 level.

Table 7. Intercorrelation matrix for data perspective measures

	TESDM	TDEPD	TAU	TAC	TMLS	TAM	TIDM	TSDM
TESDM	1.0000							
TDEPD	0.9921#	1.0000						
TAU	0.9223#	0.9346#	1.0000					
TAC	0.6695~	0.6713~	0.4896	1.0000				
TMLS	0.9893#	0.9843#	0.9084#	0.7258~	1.0000			
*TAM	0.8792#	0.8860#	0.7857#	0.9241#	0.9131#	1.0000		
*TIDM	0.9749#	0.9706#	0.9321#	0.7071~	0.9750#	0.9103#	1.0000	
*TSDM	0.9916#	0.9858#	0.9337#	0.6955~	0.9871#	0.9027#	0.9955#	1.0000

(~ significant at α = 0.01; # significant at α = 0.001)

Table 8. Size-effort variable pairs correlation summary

Size measure	Effort measure	Pearson correlation	Spearman correlation
TRE	TOTAL	0.8058	0.6923
TD	TOTAL	0.8876	0.7912
TDSCR	AN_DES	0.9053	0.7180
	TOTAL	0.7800	0.8748
TAC	PROG_UT	0.8471	0.8736
	TOTAL	0.9160	0.9341
TAU	AN_DES	0.9372	0.8077

Based on the results obtained from the correlation tests, a set of possible regression relationships was formulated. The popular least-(mean-)squares regression method (LS) was then used in conjunction with the less common least-median-squares technique (LMS) in an attempt to ensure that robust estimates, that is, estimates that are not overly influenced by outliers, were developed. The LS method has become the cornerstone of classical statistics, due to both ease of computation and tradition[37]. In cases where outliers seldom occur, the LS method is often more than adequate. However, outliers are a common feature of software engineering data sets[35]. The LMS method, as discussed by Rousseeuw and Leroy[37], was therefore also used. The PROGRESS system[37] (Program for RObust reGRESSion) computes both the least-squares and least-median-squares equations, and then automatically computes reweighted least-squares (RLS) equations based on the LMS analysis results. The RLS procedure removes or reweights the outliers identified in the LMS regression and computes a new LS equation based on the remaining data points.

All of the independent variable coefficients computed in the goodness of fit tests were shown to be significant by the PROGRESS system. This indicated that, in cases where the residuals adhered to certain restrictions, the independent (size) variable in each equation did indeed account for the response (effort) variable in a significant way. These restrictions require that estimation model residuals must be independent of one another while being evenly dispersed about the mean (at zero on the vertical axis) and that they should reflect a constant variance. Linear regression models that produce residuals that fail to conform to these requirements are generally inadequate, in that they may be improved only through the inclusion of weighted and/or transformed terms.

Table 9. Effort-size regression test results

Effort variable	Size variable	Least squares (LS)		Least median squares (LMS)		Reweighted least squares (RLS)		
		R^2	Resid. OK?	R^2	Resid. OK?	R^2	Resid. OK?	Points removed
AN_DES	TDSCR	0.90	U	0.85	N	0.78	U	12, 13
	TAU	0.93	U	0.94	Y	0.97	Y	13
PROG_UT	TAC	0.86	U	0.97	U	0.96	U	7, 9, 13
TOTAL	TRE	0.85	U	0.93	N	0.93	U	13
	TD	0.86	N	0.87	N	0.91	N	9
	TDSCR	0.79	N	0.96	Y	0.96	Y	12
	TAC	0.92	N	0.96	U	0.97	Y	11

The overall results of the regression tests, including the R^2 values, are shown in Table 9. Based on the information presented, final goodness of fit equations were chosen for each of the effort variables investigated. Of the two regressions of analysis and design effort (AN_DES), the model based on the TAU variable was the most accurate. The three R^2 values obtained from the regressions using this variable were higher than those achieved with the equations based on the TDSCR variable. Furthermore it was unclear as to whether the residual plots of the TDSCR models were satisfactory (where the ‘Resid. OK?’ column contains the letter 'U'), whereas those obtained from the TAU models were adequate ('Resid. OK?' is 'Y'). Goodness of fit for program and unit test effort (PROG_UT) was only performed in this study with the TAC variable. The results of these tests were mixed, in that the R^2 values obtained were very high but the residual plots were not satisfactory. Moreover, three of the thirteen data points were removed (as significantly overestimated outliers) in the final goodness of fit using the reweighted least squares (RLS) technique, decreasing the value of an already small sample. The choice of model for total development effort (TOTAL) was between the models based on TDSCR and TAC. Both returned very high coefficients of determination, indicating good explanatory ability, and both models produced adequate or good residual plots.

The accuracy of the models was then assessed using the *MRE* and *pred* measures common in software metrics analysis[31,34,38,39]. A high value for the R^2 indicator is evidence of a strong and consistent linear relationship among two data sets, but does not tell us how well individual data pairs relate. The magnitude of relative error (*MRE*), on the other hand, is a normalised measure of the discrepancy between actual values (V_A) and fitted values (V_F):

$$MRE = Abs((V_A - V_F)/V_A)$$

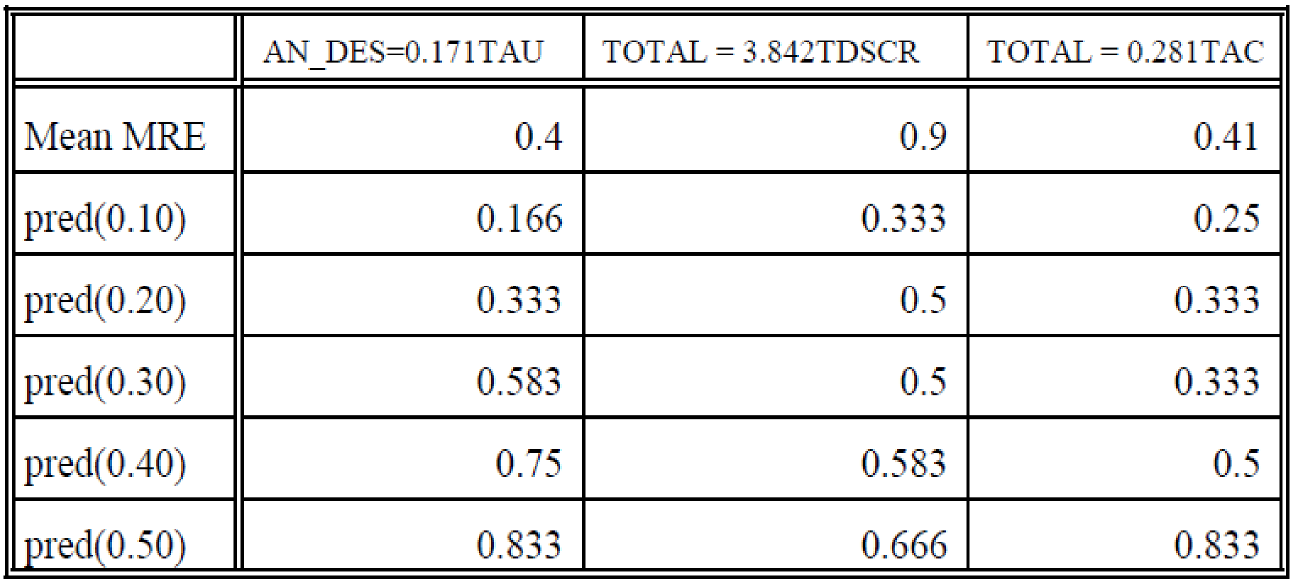
The *pred* measure provides an indication of overall fit for a set of data points, based on the *MRE* values for each data pair:

$$pred(l) = i/n$$

where *l* is the selected threshold value for *MRE*, *i* is the number of data pairs with *MRE* less than or equal *l*, and *n* is the overall number of data pairs in the set.

As an illustration, if pred(0.40) = 0.666, then we can say that 67% of the fitted values fall within 40% of the actual values. The mapping of actual and fitted values for the AN_DES and TOTAL models in terms of the tests just described are shown in Table 10. Plots of each model’s performance are also shown, in Figures 3 to 5. (Since the PROG_UT model appeared to be inadequate from the regression tests, no further analysis was performed on its accuracy.)

Table 10. Goodness of fit tests for the three models



	AN_DES=0.171TAU	TOTAL = 3.842TDSCR	TOTAL = 0.281TAC
Mean MRE	0.4	0.9	0.41
pred(0.10)	0.166	0.333	0.25
pred(0.20)	0.333	0.5	0.333
pred(0.30)	0.583	0.5	0.333
pred(0.40)	0.75	0.583	0.5
pred(0.50)	0.833	0.666	0.833

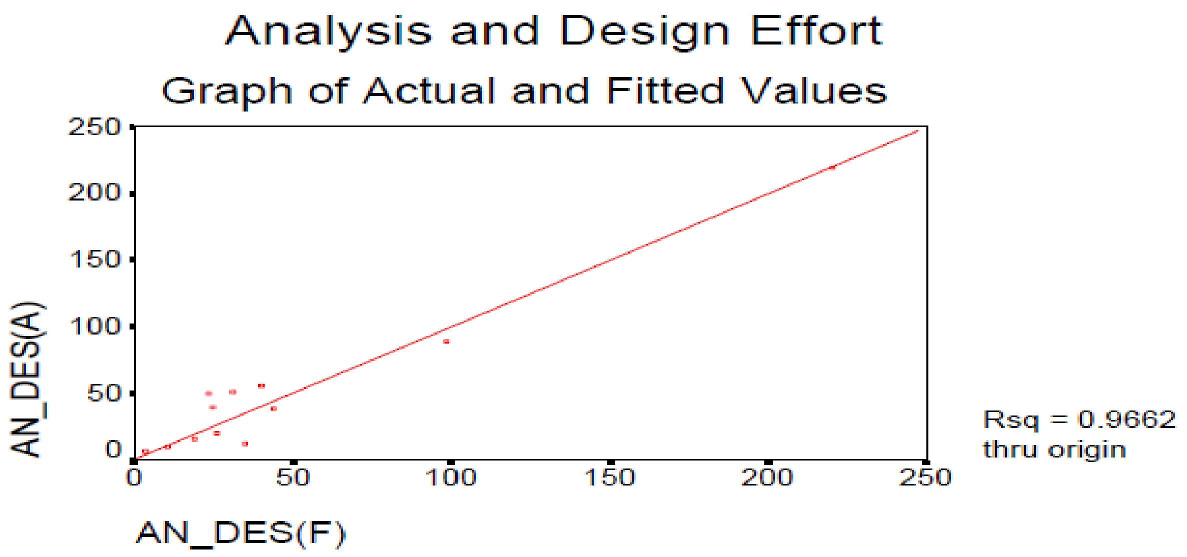


Figure 3. Goodness of fit for AN_DES effort using the number of attributes updated

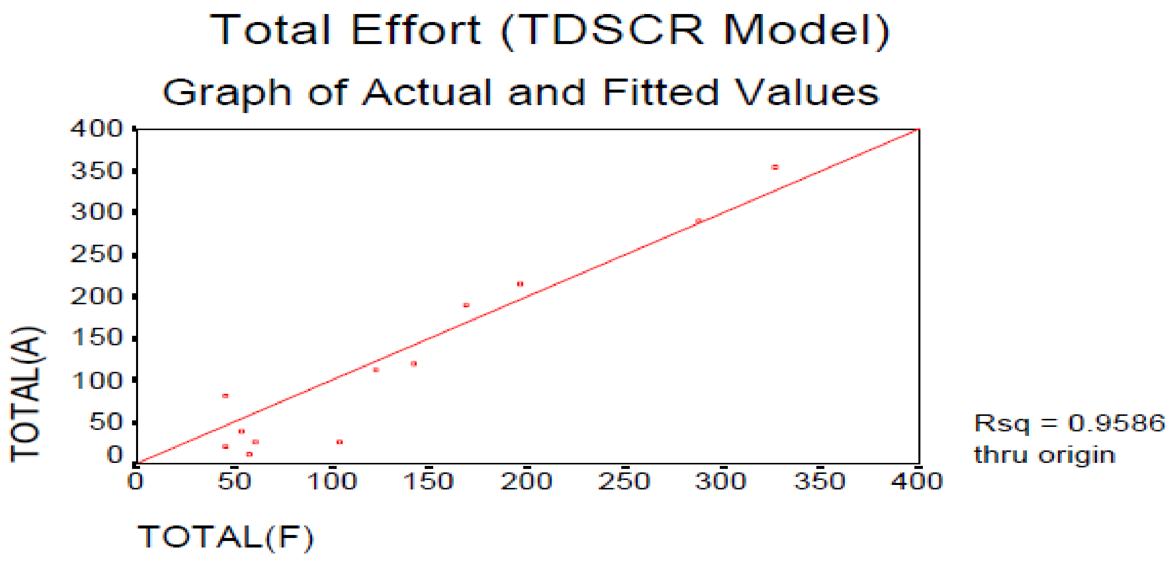


Figure 4. Goodness of fit for TOTAL effort using the number of screens

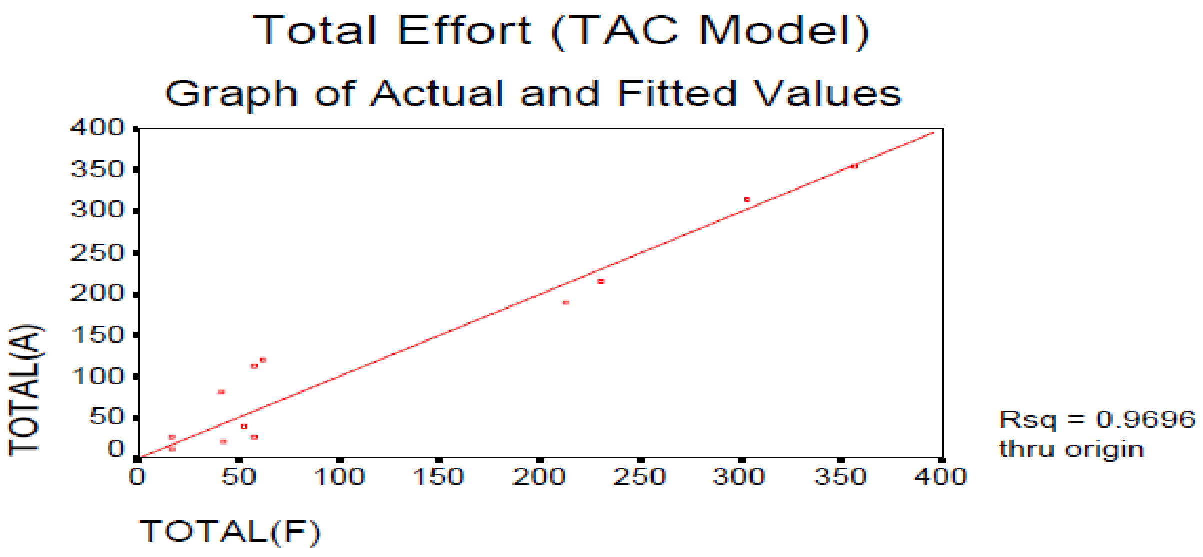


Figure 5. Goodness of fit for TOTAL effort using the number of attributes consumed

In summary it is evident that the three models are not entirely satisfactory in terms of their accuracy, as illustrated by the values attained for the *Mean MRE* and *pred(l*) measures. Conte *et al.*[38] have suggested that, for a model to be considered acceptable, *Mean MRE* should be less than or equal to 0.25 and pred(0.25) should be greater than or equal to 0.75. Tate and Verner[40], on the other hand, suggest that a more realistic level of performance for the *pred(l)* measure is pred(0.30) ≥ 0.70. None of the three models developed here satisfies either condition. The strong underlying linear relationships illustrated by the very high R^2 values, however, would suggest that improvements in indicators like *MRE* and *pred(l)* would be possible given calibration of the models under specific conditions e.g. for projects of a given size or effort range.

The univariate regression tests were of limited success for this small sample. The explanatory power of each of the three final equations was greater than 95% and the residual plots all conformed to the requirements of valid goodness of fit models. These factors suggest that the models are reasonably consistent[1]. However, the *accuracy* of the models, as represented by the *MRE* and *pred* indicators, was not as high as might have been expected. Although disappointing, this does not represent a complete failure, as model accuracy should improve as larger data sets become available and effective calibration is enabled. Under these circumstances it may also be more effective to split the sets of observations according to distributions of system size. Even in the present study, TDSCR-based fitting of TOTAL effort was substantially more effective for the larger systems in the sample. Given that larger systems are likely to represent greater investment by organisations, a lesser degree of fit for small systems may be acceptable to development managers as a trade-off to consistent estimation for large systems.

It could be suggested that multidimensional relationships should have been investigated in order to improve model accuracy. Given that the univariate relationships were so strong, however, it was felt that including further variables would simply complicate the models whilst adding little real value to the actual relationships. Moreover, with such a small set of observations, the use of such multidimensional relationships would have been statistically inappropriate. This is likely to change with larger sets of data. For the present sample of small- to medium-sized CASE-based TP/MIS systems, however, the single variable models were adequately consistent.

4. SUMMARY AND RECOMMENDATIONS

This study aimed to test the strength of possible relationships between measures of specification size and process effort within CASE environments. The proposed assessment scheme was developed as a direct response to the inadequacies and inappropriateness of previous methods, addressing issues such as subjectivity and excessive environment dependence. The scheme was then applied to data sets collected from thirteen projects developed at ten different sites. Evidence of significant, consistent relationships was provided using robust statistical analysis methods, confirming the assertion that specification size measures are related to process effort, at least within extensively automated development environments.

Refinement of the results obtained from small and medium sized systems will be forthcoming as larger samples become available for analysis and as collection becomes increasingly automated within development tools. It is hoped that the current approach will itself be incorporated into a CASE/project management tool. This will introduce two advantages over the current study: first, it will enable more objective, non-intrusive, less error-prone collection of the specification-based data to be carried out; second, it will mean that analysis and prediction may be performed and refined in the background of development as an integral and ongoing part of a project. Tate[41] and Tate and Verner[21] also suggest that on-workbench data, relating to process effort, may soon be collected automatically within CASE environments. Collection of project management data will therefore also be more precise and cost-effective. All of these factors will encourage continuing refinement of estimates, providing relevant feedback to managers whenever required.

Until software development in the commercial environment becomes a totally automated procedure, system size will continue to have an important influence on the progress and outcomes of the software process. Continually rising development costs, coupled with more and more demands for increasingly complicated systems, will encourage extensive research into both quantifiable assessment/estimation methods and development automation. It is hoped that this study, which has empirically considered the interaction of these factors, will provide some form of impetus for continued research in this area.

ACKNOWLEDGEMENTS

Financial support: the Cambridge Commonwealth Trust, the New Zealand Vice-Chancellors Committee, British Telecom plc, Clare College, Cambridge, the Cambridge University Engineering Department and the University of Otago. The author would also like to thank the anonymous referees and Mr Andrew Gray for their constructive comments on earlier versions of this paper.

REFERENCES

1. Mukhopadhyay, T and Kekre, S 'Software Effort Models for Early Estimation of Process Control Applications' IEEE Transactions on Software

Engineering Vol 18 (October 1992) pp 915-924

2. Tate, G, Verner, J and Jeffery, R 'CASE: A Testbed for Modeling, Measurement and Management' Communications of the ACM Vol 35 No 4 (April 1992) pp 65-72
3. MacDonell, S 'Deriving relevant functional measures for automated development projects' Information and Software Technology Vol 35 No 9 (Sept 1993) pp 499-512
4. Curtis, B 'The Measurement of Software Quality and Complexity,' in A J Perlis, F G Sayward, and M Shaw (eds) *Software Metrics* MIT Press, Massachusetts (1981) pp 203-224
5. Weissman, L 'Psychological Complexity Of Computer Programs: An Experimental Methodology' *ACM SIGPLan Notices* Vol 9 (June 1974) pp 25-36
6. Brooks, F P Jr 'No Silver Bullet - Essence and Accidents of Software Engineering' IEEE Computer Vol 20 (April 1987) pp 10-19
7. Henry, S and Lewis, J 'Integrating Metrics into a Large-Scale Software Development Environment' Journal of Systems and Software Vol 13 (1990) pp 89-95
8. Munson, J C and Khoshgoftaar, T M 'Applications of a Relative Complexity Metric for Software Project Management' *Journal of Systems and Software* Vol 12 (1990) pp 283-291
9. Boehm, B W and Papaccio, P N 'Understanding and Controlling Software Costs' IEEE Transactions on Software Engineering Vol 14 (October 1988) pp 1462-1477
10. Paulson, D and Wand, Y 'An Automated Approach to Information Systems Decomposition' *IEEE Transactions on Software Engineering* Vol 18 (March 1992) pp 174-189
11. Kerr, J M ‘The Information Engineering Paradigm’ *Journal of Systems Managemen*t (April 1991) pp 28-35
12. Crozier, M, Glass, D, Hughes, J G, Johnston, W and McChesney, I ‘Critical Analysis of Tools for Computer-Aided Software Engineering’ Information and Software Technology Vol 31 (November 1989) pp 486-496
13. DeMarco, T Controlling Software Projects Yourdon, New York (1982)
14. Tate, G and Verner, J ‘Software Metrics for CASE Development’ in *Proceedings COMPSAC’91* Tokyo (1991) pp 565-570
15. Robinson, K ‘Putting the SE into CASE’ in Spurr, K and Layzell, P (eds) *CASE: Current practice, future prospects* John Wiley (1992) pp 1-20
16. Case, A F Jr., *Information Systems Development: Principles of Computer-Aided Software Engineering* Prentice-Hall, Englewood Cliffs NJ (1986)
17. Samson, W B, Nevill, D G and Dugard, P I 'Predictive Software Metrics Based on a Formal Specification' *Information and Software Technology* Vol 29 (June 1987) pp 242-248
18. Symons, C R *Software Sizing and Estimating: Mk II FPA (Function Point Analysis)* John Wiley & Sons, Chichester (1991)
19. Verner, J, Tate, G, Jackson, B and Hayward, R G 'Technology Dependence in Function Point Analysis: A Case Study and Critical Review' *Proceedings 11th International Conference on Software Engineering*, Pittsburgh PA(1989) pp 375-382
20. Albrecht, A J 'Measuring Application Development Productivity' *Proceedings IBM GUIDE/SHARE Applications Development Symposium* California (1979)
21. Tate, G and Verner, J 'Approaches to Measuring Size of Application Products with CASE Tools' Information and Software Technology Vol 33 (November 1991) pp 622-628
22. MacDonell, S G 'Comparative Review of Functional Complexity Assessment Methods for Effort Estimation' *Software Engineering Journa*l (May 1994) pp 107-116
23. Basili, V R and Rombach, H D 'The TAME Project: Towards Improvement-Oriented Software Environments' *IEEE Transactions on Software Engineering* Vol 14 (June 1988) pp 758-773
24. Bush, M E and Fenton, N E 'Software Measurement: A Conceptual Framework' Journal of Systems and Software Vol 12 (1990) pp 223-231
25. Vessey, I, Jarvenpaa, S L and Tractinsky, N ‘Evaluation of Vendor Products: CASE Tools as Methodology Companions’ *Communications of the ACM* Vol 35 (April 1992) pp 90-105
26. Gray, R H M, Carey, B N, McGlynn, N A and Pengelly, A D 'Design Metrics for Database Systems' BT Technology Journal Vol 9 (October 1991) pp 69-79
27. Parnas, D L ‘On Criteria to be Used in Decomposing Systems into Modules’ Communications of the ACM Vol 14 (April 1972) pp 221-227
28. Boehm, B W, Gray, T E and Seewaldt, T 'Prototyping Versus Specifying: A Multiproject Experiment' *IEEE Transactions on Software Engineering* Vol 10 (May 1984) pp 290-302
29. Lin, C-Y 'Systems Development With Application Generators: An End User Perspective' Journal of Systems Management (April 1990) pp 32-36
30. Eglington, D 'Cost-Effective Computer System Implementation in Medium Sized Companies' in Gillies, A (ed) *Case Studies in Software Engineering* Salford University Business Services Ltd (March 1991) pp 56-59
31. Kemerer, C F 'An Empirical Validation of Software Cost Estimation Models' *Communications of the*

ACM Vol 30 (May 1987) pp 416-429

32. Mukhopadhyay, T, Vicinanza, S S and Prietula, M J 'Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation' *MIS Quarterly* (June 1992) pp 155-171
33. Wittig, G E and Finnie G R 'Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort' *Australian Journal of Information Systems* (May 1994) pp 87-94
34. Srinivasan, K and Fisher, D 'Machine Learning Approaches to Estimating Software Development Effort' *IEEE Transactions on Software Engineering* Vol 21 (February 1995) pp 126-137
35. Kitchenham, B A and Pickard, L M 'Towards a Constructive Quality Model Part II: Statistical techniques for modelling software in the ESPRIT REQUEST project' *Software Engineering Journal* (July 1987) pp 114-126
36. Hampel, F R, Ronchetti, E M, Rousseeuw, P J and Stahel, W A *Robust Statistics* John Wiley & Sons, New York (1986)
37. Rousseeuw, P J and Leroy, A M *Robust Regression and Outlier Detection* John Wiley & Sons, New York (1987)
38. Conte, S D, Dunsmore, H E and Shen, V Y *Software Engineering Metrics and Models* Benjamin/Cummings, Menlo Park CA (1986)
39. Verner, J and Tate, G 'A Software Size Model' IEEE Transactions on Software Engineering Vol 18 (April 1992) pp 265-278
40. Tate, G and Verner, J 'Software Costing in Practice' in Veryard, R *Information and Software Economics* Butterworth Scientific UK (1990)
41. Tate, G 'Management, CASE and the Software Process' *Proceedings 12th New Zealand Computer Conference* Dunedin (1991) pp 247-256