# Comparative review of functional complexity assessment methods for effort estimation

Stephen G. MacDonell

*Computer and Information Science*
*University of Otago, Dunedin, New Zealand*
*stevemac@commerce.otago.ac.nz*

## Abstract

*Budgetary constraints are placing increasing pressure on project managers to effectively estimate development effort requirements at the earliest opportunity. With the rising impact of automation on commercial software development the attention of researchers developing effort estimation models has recently been focused on functional representations of systems, in response to the assertion that development effort is a function of specification content [1]. A number of such models exist - several, however, have received almost no research or industry attention. Project managers wishing to implement a functional assessment and estimation programme are therefore unlikely to be aware of the various methods or how they compare. This paper therefore attempts to provide this information, as well as forming a basis for the development and improvement of new methods.*

## 1. INTRODUCTION

Software development project planning frequently involves the use of estimates in the determination of projected effort requirements. Numerous research studies over the last two decades have therefore attempted to develop and validate estimation models, so that systems development effort can be predicted with some quantitative degree of accuracy and consistency [2, 3]. The intuitive relationship that exists between ‘software complexity’ and development effort has provided the basis for many of these estimation models. This relationship states that a more complex piece of software will generally require greater effort in development than a less complex counterpart. Thus a wide variety of factors thought to contribute to complexity have been proposed as possible determinants of development effort. Most early estimation models, for example, provided post-development estimates of effort (to be used for future projects) based on the number of delivered lines of code [4, 5]. Other effort estimation models have been based on countable attributes of software designs [6]. Some approaches have also considered the impact of external factors, such as system type and developer experience, on projected effort estimates. Thus the influence of many diverse factors has been investigated in the pursuit of an adequate estimation model.

A common element of these studies has been the assessment of product attributes with a view to the subsequent prediction of effort requirements, based on the assumption that the product attributes have an impact on development effort. Underlying this approach is another often stated assumption that the product characteristics examined are considered to be adequate indicators of product ‘complexity’. Although complexity is seldom defined in measurable terms, most of these studies have accepted the intuitive association between aspects of system size and interconnectivity and overall product complexity [7, 8]. It is not the object of this paper to debate the validity or otherwise of this approach - rather, we are concerned here with the adequacy of estimation models *given* this approach. Thus, complexity is considered here to be a function of product size and interconnectivity. It therefore follows that as size and interconnectivity increase, so the complexity of a system increases, and consequently development effort requirements are greater.

The demand for early estimates of effort, that is, before a project is fully under way, provides the motivation behind the development and use of *function-based* assessment and estimation methods which consider the impact of specification product attributes on development effort. In terms of the software process, a system specification product is a logical representation of system functionality, with no consideration of ‘physical’ constraints, for example, the development language to be used or the required hardware platform. Factors such as these are generally incorporated in the design process, with subsequent translation into code during system implementation. Given this classification, a specification product in the commercial systems domain often includes logical or conceptual models of data, process and user interface requirements [1, 9]. Assessing the size and interconnectivity of these models enables the *functional complexity* of a system to be considered, rather than the complexity of a particular implementation. As the degree of automation in the development process has increased, through the use of computer aided software engineering

(CASE) tools, it has been suggested that specification-based indicators derived from these models should provide a useful basis for relatively consistent effort predictions [9]. This paper therefore examines nine such functional assessment approaches for effort estimation according to a set of six characteristics.

The next section of this paper describes the six criteria against which the methods are evaluated. This is followed by a comparative review of currently known functional complexity assessment methods for business systems development effort estimation. An overall comparison of the approaches is presented. Opportunities for improvement are also discussed as a basis for further research.

## 2. CRITERIA FOR COMPARISON

Six characteristics were selected for the evaluation of the methods based on criticisms directed at previously proposed complexity assessment and effort estimation models.

- Automation - Product-based data collection necessary for complexity assessment and effort prediction should now be largely automated, given the development tools available and in use within commercial software development departments. Not only does this help to ensure the integrity of data, it also reduces the intrusive nature of the data collection task [10, 11].
- Comprehensive assessment - In criticising the effectiveness of previous models, Case [12] and Wrigley and Dexter [13] suggest that product factors other than those considered by the models should also have been included, if only so that they could be discarded at a later stage after evidence had illustrated that they were of little consequence. In terms of specification products, the impact of the size and interconnectivity of data, process and user interface models should be assessed, as each may make some contribution to development effort requirements [1, 9].
- Objectivity - Kulkarni et al. [14] and Lederer and Prasad [15] cite the issue of subjectivity as a significant drawback associated with models employing product-based effort predictions. Criticism generally centres around the fact that the impact of the subjective component can overwhelm the usefulness of the approach. Some degree of consistency may be possible when experienced assessors and estimators remain in a development group, but problems can arise when new personnel are required to perform similar tasks.
- Specification basis - As stated in the previous section, one of the main motivating factors behind the development of new effort estimation approaches is the opportunity for the earliest possible predictions to be generated whilst still maintaining some degree of accuracy. If this requirement is to be fulfilled, the product complexity assessment should be performed using conceptual specification system models (rather than those developed during and after the design phase) [16, 17].
- Testing - Munson and Khoshgoftaar [18] state that there have been more than ninety assessment methods proposed within the realm of software measurement. It is almost certain, however, that some of these methods remain untested in the relevant environment. This is a most necessary characteristic if an assessment procedure is to be used with confidence in the software development industry [19].
- Validity - In relation to the previous point, any complexity assessment and effort estimation approach should be validated on data sets derived from systems other than those used in the original model testing [15].

Thus the six characteristics above have been selected as desirable attributes of complexity assessment and effort estimation models derivable from system specifications. For the purpose of repeatability, they are more succinctly and objectively defined as follows:

**Char 1**: Automatic - can the product complexity assessment task be *totally* performed in an automated manner, requiring *no* input from personnel?

**Char 2**: Comprehensive - are aspects of the size and interconnectivity of the data, process and user interface representations considered by the model?

**Char 3**: Objective - will the model as defined *always* produce the same result for a given system at a given point in time (assuming no counting errors) irrespective of the person requiring or performing the assessment and/or estimation?

**Char 4**: Specification basis - can the complexity assessment task be *totally* undertaken using implementation-independent system representations?

**Char 5**: Tested - has the complete model been tested using appropriate real-world data?

**Char 6**: Validated - has the complete model been evaluated using systems other than those employed in testing the model?

These descriptions should now enable objective binary decisions to be made concerning the provision of each characteristic by the various models.

## 3. SPECIFICATION-BASED COMPLEXITY ASSESSMENT AND EFFORT ESTIMATION MODELS

DeMarco [1] suggests that development effort is a function of a system's information content. He further asserts that the information content of a final coded system is a well-behaved function of the information content of that system's specification. Unfortunately the lack of uniformity among specification structures, he continues, prevents direct information theory evaluation of traditional requirements documents - however, he does suggest that the use of standard specification models would provide a consistent framework for structural comparison. In essence, this provides the basis for the development and use of functional assessment methods. A number of existing techniques are now discussed and evaluated according to the six criteria described in the previous section. Although several of these existing assessment methods have size or productivity estimation as their overall goal they have all attempted to consider system complexity and development effort in some way.

### 3.1. Bang metrics

Bang [1] is offered as an implementation-independent, quickly derived approach for effort prediction that can lead to the development of size, cost and productivity estimates. The Bang system of measures is based on a three-view perspective of system specifications, ignoring all details of the method to be used in system implementation. The three views consist of a functional model, a retained data model and a state transition model. This complete representation enables the use of quantitative analysis to provide a measure of the function to be delivered by the system as perceived by the user. DeMarco [1] does state that most systems can be adequately specified using just two of the three views - particularly for business software this would normally consist of the data and functional models.

There are three main basic attributes that can be used as the principal indicators of Bang. They are the count of functional primitives or elementary processes *FP*, the count of inter-object relationships *RE* and the count of data elements flowing out of the system *DEO*. The ratio *RE/FP* is said to be a reasonable measure of data strength. If the ratio is less than 0.7, this implies a function-strong system - that is, a system that can be thought of almost completely in terms of operations, for example, robotic systems; if *RE/FP* is greater than 1.5, this implies a data-strong system, or one that should be thought of in terms of the data it acts upon. The middle range identifies hybrid systems. The *DEO/FP* ratio is indicative of the system's focus on either data movement or data computation. Commercial systems tend to have high levels of *DEO/FP*, scientific systems, low.

For function-strong systems it is suggested that the size or information content of a process can be approximated as a function of the number of tokens *TC*, or data elements, involved in the process. Variations in process complexity can then be accounted for through the assignment of weighting correction factors *W*, based on sixteen functional classes, to each primitive's raw value *BANGf*. These weighted figures are then summed over all elementary processes to provide a final value of function Bang *FBANG* for the system:

$$FBANG = \sum BANGf_i * W_i$$

where

$$BANGf_i = (TC_i * log_2(TC_i))/2$$

The count of objects *OB*, or entities, in the database is the base metric for data-strong systems, corrected for the amount of connectedness among the objects *COB*. Data Bang *DBANG* is the overall result obtained by this procedure:

$$DBANG = \sum COB_i$$

Hybrid systems require separate computation of both function and data Bang so that the two figures can be used in the prediction of different activities. DeMarco [1] states that combining the two totals would be difficult, as it would be almost certain that one should be weighted more heavily than the other but that the magnitudes of these weightings would depend specifically on the system in question.

#### *3.1.1. Evaluation*

Consideration of complexity is achieved in Bang through the use of weightings that are dependent on the flows of data elements or on the amount of entity connectedness. Although DeMarco [1] provides a beginning set of correction factors, these weightings must then be determined through trial and error and with extensive in-house calibration. The amount of work required by a department to determine the appropriate weightings has inhibited the wider use of Bang [20]. Furthermore, results for database-oriented systems, most common in the business domain, are sparse, despite the fact that the technique is now more than ten years old [21].

Bang can be applied at the conceptual modelling phase and does consider the number of data elements processed. However, it fails to distinguish between input and output data elements, even though the effort required to develop their respective processing components is different [22]. Data Bang also considers the number of entity relationships, but no assessment of the relationship types is performed. Furthermore, assignment of the sixteen complexity classes must be performed manually by personnel, reducing the possibility for automatic calculation.

#### ***Outcome: Bang metrics***

- Automatic - No
- Comprehensive - Yes

- Objective - No
- Specification basis - Yes
- Tested - Yes
- Validated - No

## 3.2. Bang metric analysis (BMA)

This is an adaptation of the original Bang method that considers both processing and data requirements in transaction-based systems [23]. Each functional primitive or elementary process is assigned a level of complexity according to the number of create, read, update and delete operations that it performs, with each of these operations carrying a weighting factor. This forms the basis for the calculation of a process' function Bang. The formulation of data Bang is the same as in DeMarco's theory [1], that is, complexity is dependent on the number of entity relationships. Total Bang is the sum of both function and data Bang for each elementary process.

### *3.2.1. Evaluation*

In terms of data-oriented transaction systems this is a much more useful approach, in that database operations are considered instead of DeMarco's sixteen weighted functional classes [1]. The weightings used for the operations were intuitively proposed, but have proved to be useful in testing. Regression techniques have been used to determine the appropriate coefficients for function and data Bang in the prediction of overall development effort. This method, however, still suffers from the same drawbacks as DeMarco's original proposal [1], that is, a failure to distinguish between input and output data elements and non-assessment of relationship types.

***Outcome: BMA***

- Automatic - Yes
- Comprehensive - Yes
- Objective - Yes
- Specification basis - Yes
- Tested - Yes
- Validated - No

## 3.3. CASE size metrics

Tate and Verner [9, 21] and Tate [24] assert that the automatic measurement of size as a function of data dictionary entries should be possible in a CASE environment. Furthermore, they state that the widespread use of graphics within CASE tools and the relative absence of lines of code means that more appropriate size measures should be chosen. They therefore suggest that measures of specification *size* applicable to transaction-oriented database systems may include those based on the data model, the data flow model and the user interface. Examples of specific product measures suggested include counts of entities and attributes, data flows, processes and data stores. It is suggested that measures such as these will be useful in the development of effort estimates. Measurement of *complexity*, on the other hand, is described by Tate and Verner [9] as a relatively well-defined area of conventional development that should follow similar principles within CASE, except that it may be based on data structure and data flow models. At the risk of oversimplification, they suggest that complexity is a measure of component interconnectivity within a software product, an aspect that should be automatically computable within a CASE environment and that should present no particular problems.

### *3.3.1. Evaluation*

As discussed earlier in this paper, complexity is considered to be a combination of aspects of size and interconnectivity. Thus, Tate and Verner's discussion of specification size [9] remains particularly appropriate here as size is certainly thought to have an impact on overall complexity. Therefore the automatically derivable measures suggested above are relevant. Their study was a preliminary examination of metric possibilities and consequently no evidence supporting or refuting their suggestions was provided. Subsequent empirical investigations into the relationship between the measures and development effort, however, have provided some support for the approach [25].

***Outcome: CASE size metrics***

- Automatic - Yes
- Comprehensive - Yes
- Objective - Yes
- Specification basis - Yes
- Tested - Yes
- Validated - No

## 3.4. Entity metrics

Gray et al. [26] describe a set of techniques for the assessment of the complexity of various tasks relating to the development of data-oriented systems. They firstly propose an ER metric for determining the effort required to implement a database design. There are said to be four factors that influence the complexity of a database design: the number of entities in the design, the number of relationships for each entity, the number of attributes for each entity and the distribution of relationships and attributes. The overall complexity of a complete ER diagram is shown as the sum of the complexities of the entities that comprise it. Individual entity complexity is calculated using the values of the number of relationships, functionally dependent attributes and non-functionally dependent attributes for each entity. Weightings for these factors are also used in the formula - it is suggested that these weightings can be used to reflect the impact of characteristics from the local development environment. The calculation also considers the 'functional complexity' of each entity, but this is assumed to have the constant value of one for every entity.

The third measure is an enhancement of Shepperd's structural IF4 metric [27] which was itself derived from Henry and Kafura's Information Flow metric [8]. The original IF4 measure makes no consideration for the use of a database - therefore an extension is suggested. Each entity in a database is regarded as a type of module that can receive information, through create and update transactions, and can also provide information, through read and delete operations. A delete operation is said to be an information extraction because the entity will contain less information after the transaction is completed. Thus the enhanced IF4 metric (IF4+) is said to enable the assessment of both processing and data in a single metric approach.

Finally a measure of database operation complexity is proposed. This treats each operation (create, read, update and delete) as a virtual entity, being composed of the parts of the entities accessed by the operation. The ER metrics as proposed can then be used, with the number of entities replacing the number of relationships in the original formula, to assess the overall complexity of each operation.

### 3.4.1. Evaluation

Overall this would seem to be a positive approach for the analysis of business systems, particularly given that its focus is on the impact of both data *and* processing.

The decision to assign a delete operation as a provision of data is interesting. Although it is certainly true that an entity will contain fewer elements after the operation, it can equally be said that the operation itself is one that writes a blank record, therefore suggesting that it should be classified as a 'receive' by the entity. Placing this issue aside, the new IF4+ metric could be useful as a more comprehensive structural complexity measure. It is not strictly a functional measure, however, because the processing assessment is based on design-phase module structure charts.

The final measurement approach, considering database operation complexity, is also a valid and worthwhile proposal. Again, it would seem to be more comprehensive than many other techniques in that it attempts to consider processing and data in one metric. Moreover, the basic measures could be determined automatically if the representations were stored electronically. However, there is no indication as to whether one type of operation will be inherently more complex than another, without consideration of the data that it manipulates. Furthermore, the number and type of relationships between the entities are not considered, and there is no explicit guidance provided as to how entity look-ups or relationship exclusivity should be treated in the assessment.

***Outcome: Entity metrics***

- Automatic - Yes
- Comprehensive - Yes
- Objective - Yes
- Specification basis - Yes
- Tested - No
- Validated - No

## 3.5. Function point analysis (FPA)

Function point analysis [28] is the most widely investigated of the function-based approaches. Quantification of complexity under this technique is performed as a sub-task of the complete model, the overall original purpose being the determination and prediction of development productivity. Each system is considered in terms of the number of inputs, outputs, inquiries, files and external system interfaces that it contains. The system total for each of these attributes is multiplied by a weighting factor appropriate to its complexity in the system (simple, average or complex), based on the number of data elements and/or file types referenced. The combined total of all of these products is then adjusted for application and environment complexity - this can cause an increase or decrease of up to 35% in the raw function point total. Calculation of the adjustment factor is carried out by considering the need for certain features in the system, for example, distributed processing, on-line data entry, end user efficiency and ease of installation. Each of the fourteen factors is assigned a degree of influence of between zero (no influence) and five (strong influence), and these are summed to give a total degree of influence, denoted $N$. One of the fourteen factors is allocated for the consideration of complex processing. A technical adjustment factor is then calculated as $(0.65 + 0.01(N))$. This adjustment factor is subsequently multiplied by the raw function point total to determine the final function point value delivered by the system. According to Grupe and Clevenger [7] the underlying assumption of FPA is that higher numbers of function points reflect more complex systems; these systems will consequently take longer to develop than simpler counterparts.

### 3.5.1. Evaluation

Complexity is therefore considered in two ways during function point analysis. It is questionable, however, whether this consideration is completely adequate. Albrecht acknowledges that the complexity weights applied to the raw function point counts were "...determined by debate and trial." [29 p.639]. The absence of empirical foundation for these weights has since received criticism from several quarters [30, 31]. Moreover, with respect to the raw counts, the categorisation of the system components as simple, average or complex, although clearly straightforward, seems to be rather simplistic in terms of a comprehensive assessment of complexity - Symons [32] provides the example that a component consisting of over 100 data elements is assigned at most twice the points of a component that contains just one data element. It is also suggested that the weightings are unlikely to be valid in all development situations.

There are similar problems with the technical complexity adjustment process. It would seem unlikely that the consideration of the same fourteen factors would be sufficient to cope with all types of applications. Also, adjustments to the raw counts can only be affected by a factor within the zero to five range which, although simple, is unlikely to be appropriate in all cases. Consideration of processing complexity in only one of the fourteen factors is not only inadequate, it may also not be practically applicable at the software specification stage. It is recommended that the value of the adjustment factor for complex processing should be based on a number of factors, including the need for sensitive control/security processing and extensive logical or mathematical processing [29, 33, 34]. It would seem unlikely, however, that information of this kind would be available at the conceptual modelling stage. This reinforces another drawback of the method, in that it is not based on modern structured analysis and data modelling techniques [21].

Overall, then, the technique tends to underestimate systems that are procedurally complex and that have large numbers of data elements per component [32, 35]. Shepperd [31] and Ratcliff and Rollo [36] also remark that the identification of the basic components from the specification can be difficult and rather subjective - different analysers may therefore use different logic to determine the number and complexity of the functions provided by the system [37, 38]. It has been suggested that this subjective element can dominate the final results, reducing the utility of a seemingly quantitative process [13, 32, 39]. A recent investigation by Kemerer [40], however, has provided some evidence to refute this assertion. Moreover, the method itself is widely used and supported.

#### *Outcome: FPA*

- Automatic - No
- Comprehensive - Yes
- Objective - No
- Specification basis - No
- Tested - Yes
- Validated - Yes

## 3.6. Information engineering metrics

Data representing complexity variables thought to influence development phase effort was collected from a number of information engineering development projects [41]. In producing an information strategy plan for an organisation it was found that the number of entity types had a large impact on project effort, based on twenty-eight projects from seventeen domains. Other important complexity variables were the number of lowest-level functions, the number of proposed data stores and several other factors relating to the structure and personnel of the organisation concerned. For business area analyses, the number of elementary processes to be implemented in a system was found to be highly influential, based on data derived from twenty projects over ten application domains. Other factors included the number of users interviewed, the number of relationships, the number of attributes and the number of action diagrams.

### *3.6.1. Evaluation*

This approach is a practical, empirical evaluation of intuitive relationships with minimal background theory. The results obtained may be useful in the information engineering (IE) environment, but because the formulae derived are totally oriented towards steps of the IE methodology, their general application may be less effective. Furthermore, the effort data was used after the fact for metric analysis. That is, it was not collected specifically for assessment purposes. Therefore much of the data was based on personal notes, personal memory, accounting data and best guesses. Finally, several variables relate to the development and organisational environment, reducing the functional basis of the method. This may have been due to the fact that only some of the projects made use of CASE or similar tools.

#### *Outcome: IE metrics*

- Automatic - No
- Comprehensive - Yes
- Objective - No
- Specification basis - No
- Tested - Yes
- Validated - Yes

## 3.7. Mark II FPA

Symons [22, 32] has developed a specification-based sizing and effort estimation technique based on a revised version of the function point analysis method. He identified several failings with Albrecht's original technique, as outlined earlier in this section, pertaining particularly to the classification and weighting strategies used in the original theory. Symons [32] further suggested that these problems were compounded by technology-driven changes, so that, for example, the original concept of a logical file was no longer appropriate in the database environment that now dominates business systems. Symons [32] therefore adopted the entity type as the basic data equivalent for transaction-centred systems.

The Mark II method involves the identification of all the inputs, outputs and processes associated with each externally triggered logical transaction performed by a system. To assess the size contribution of the input and output components, Symons' method [32] counts the number of data elements that are used in and produced by the transaction. This is founded on the assumption that the effort for formatting and validating an input or an output is proportional to the number of data elements in each. Symons [32] suggests that this provides greater objectivity in the counting procedure when compared to Albrecht's somewhat subjective approach.

Identification and evaluation of the process component is more difficult, in terms of developing an appropriate size parameter for this aspect of a transaction. The method suggested by Symons [32] relies on previous work on internal structure measurement based on code branching and looping [42]. It is suggested that the data structure employed by a system may provide a basis for the assessment of processing complexity. At the specification stage, this is represented by the access path of a transaction through the system entity model. Symons [32] states that since each step in the path correlates to a branch or a loop, the processing complexity will be directly related to the number of entities referenced by the transaction. Although this argument was originally considered to be rather tentative, providing only a crude measure of processing complexity, it has remained intact and has been reinforced in Symons' more recent work [22].

The formula for the raw size factor in unadjusted function points is therefore calculated by multiplying locally calibrated weighting factors with the basic counts of input and output data elements and the number of entity references in the system, and then summing together the three weighted totals for all of the system's transactions. An industry standard set of weightings is available as a starting point. The technical complexity adjustment procedure is very similar to that of the original theory except that the fourteen Albrecht factors [28] are augmented by five or more new characteristics.

### *3.7.1. Evaluation*

Using counts of data elements for the input and output components is a positive and more contemporary approach, as is the adoption of entity-based assessment. Under this method, however, there is no consideration of the entity link types traversed, despite the fact that, as Symons [32] acknowledges, they produce different processing requirements. The technique also counts a maximum of one reference to each entity per transaction, in spite of the fact that a transaction may refer to a given entity more than once in order to manipulate different data elements. Mark II also fails to consider the types of operation that are performed in each transaction (that is, create, read, update or delete), even though others [23, 26] suggest that the operations are of differing complexities. As justification, Symons [22] suggests that operation types should not be counted as they might depend on the logical database design, the file structure or the database tools used, that is, physical considerations. This, he suggests, is contrary to gaining a measure of the logical representation.

The use of McCabe's work as a basis for process complexity in terms of logical structure is certainly valid to an extent; however, evidence has also shown that McCabe's measure is not comprehensive enough to reflect overall complexity and that other contributors are assessed inadequately using this approach [43]. Therefore this basis should be further investigated. In calculating the input and output components, no distinction is made between data elements that are read from/written to the database and those that are provided by/for the user, even though the processing and validation requirements for each of these situations may be quite different.

In order to perform estimation for future project requirements, historical effort data from past development projects must be allocated by staff after the fact to the input/output/process components and to each of the nineteen adjustment factors. Also acknowledged as crude in 1988, Symons [22] has subsequently stated that the method has provided reasonable results in validation studies based on the analysis of more than sixty systems. It is somewhat subjective, however, and may be jeopardised by leading questions from the assessor. Moreover, collection of the data required for the nineteen adjustment factors would be difficult to automate [44]. Finally, Albrecht [45] states that the use of local weights in the initial functional assessment makes the method invalid as a purely functional approach. This seems reasonable, in that he asserts that the functional measure should be derived first and then adjusted or weighted accordingly.

***Outcome: Mark II FPA***

- Automatic - No
- Comprehensive - Yes
- Objective - No
- Specification basis - Yes
- Tested - Yes
- Validated - Yes

## 3.8. Metrics Guided Methodology (MGM)

The Metrics Guided Methodology (MGM) was proposed by Ramamoorthy et al. [46] as a reflection of the need for metrics from all development phases. Discussion of the specification stage is based on the use of requirements specification languages (RSLs). It is suggested that a spectrum of measures is needed to assess the different aspects of a specification, as it is normally not possible to specify requirements fully from just one perspective. Normally, then, both processing and data requirements are developed. A set of metrics that considers the control-flow and entity models of an RSL specification is therefore described. Measures include the number of paths, nesting levels, ANDs and ORs, statements, data types and files.

### *3.8.1. Evaluation*

Although this approach does consider the function of a system, the measurements used are more lexical or topological, due to the language-based form of RSLs. This also means that the technique is not applicable to conceptual data or structured analysis models. Moreover, some of the measures (such as those that are concerned with determining the style and meaning of the RSL specifications) can only be determined in a subjective manner.

***Outcome: MGM***

- Automatic - No
- Comprehensive - Yes
- Objective - No
- Specification basis - No
- Tested - No
- Validated - No

## 3.9. Usability measures

Wilson [47] has described a method for determining the usability of systems, in order to enable the comparison of designs that conform to the same requirements. The approach is based on cognitive issues not generally covered in quantitative assessment. The procedure considers the number of user-visible concepts, terms and inter-relationships in a system, prior to implementation. This practice is said to actually measure the complexity of application problems, system designs and system-supported solutions, based on the semantic analysis of a design model similar to the ER representation. Under this model there are five mutually exclusive concept types:

1. entity - something that (usually) persists in time as (some of) its attributes and relationships change;
2. event - an occurrence of a change in the attributes and/or relationships of one or more things;
3. relationship - a directed association or connection between something and (usually) something else;
4. attribute - an aspect of something that can be qualitatively or quantitatively assessed;
5. value - an assessment of an attribute of something.

Different system design approaches, that is, using different methodologies, can be assessed for complexity using various factors, such as the number of entity types, the number of event types, the number of value types, the number of new terms and the average number of attributes per subject. Generally, the design method with the lowest total number of concepts and terms is the least complex and therefore the most usable. Wilson suggests that the average values of the features mentioned should conform as a general rule to Miller’s 7±2 constraint [48], which is believed to be related to understandability.

The complexity of solutions proposed for a system requirement can be measured using the following factors: number of entity types, number of entity attributes or relationships, number of event types, number of event attributes or relationships and the number of value types - these figures give the total concepts - and the average number of attributes/relationships per subject, the average number of events per subject, number of non 1 to 1 problem-solution choices (the number of times the user is faced with alternative ways to map problem concepts to solution concepts) and the number of non 1 to 1problem-solution relationships (where a problem requires none or more than one solutions) - these values give the total number of problem-solution relationships. The solution with the fewest concepts is generally the one that supports the entities and operations with the best match to the problem and is therefore the easiest to implement. Again, Miller’s constraint [48] is recommended for evaluation of the average figures.

### *3.9.1. Evaluation*

Although a novel approach, this method has seen no further investigation. The focus on understandability reduces the usefulness of this technique as a general, objective procedure. The only consideration of processing in this scheme is the counting of entity event types and only the number of relationships is considered, not the type.

***Outcome: Usability measures***

- Automatic - No
- Comprehensive - No
- Objective - No
- Specification basis - Yes
- Tested - No
- Validated - No

## 4. COMPARISON OF METHODS

The following two tables summarise the relative merits of the nine development effort estimation procedures considered above, in terms of the six characteristics. (Due to restrictions on room the six criteria have been abbreviated in the heading of Table 2.)

| Method | Comments |
|---|---|
| Bang | Intuitive and early, but partly subjective and not validated. |
| BMA | No subjectivity and easy to automate, but minimal testing. |
| CASE Size | Basis in conceptual models, objective and tested. |
| Entity | Early and objective, but as yet untested. |
| FPA | Question over objectivity, but widely used, tested and supported. |
| IE | Several subjective elements but relatively comprehensive. |
| Mark II FPA | Not completely objective or automatable, but well tested. |
| MGM | Partially automatable, but only after conceptual phase. |
| Usability | Somewhat subjective and completely untested. |

**Table 1**: General comments on functional assessment and estimation methods

| Method | Char 1 | Char 2 | Char 3 | Char 4 | Char 5 | Char 6 | Rating |
|---|---|---|---|---|---|---|---|
| | Auto. | Comp. | Obj. | Spec. | Tested | Valid. | (out of 6) |
| Bang | N | Y | N | Y | Y | N | 3 |
| BMA | Y | Y | Y | Y | Y | N | 5 |
| CASE Size | Y | Y | Y | Y | Y | N | 5 |
| Entity | Y | Y | Y | Y | N | N | 4 |
| FPA | N | Y | N | N | Y | Y | 3 |
| IE | N | Y | N | N | Y | Y | 3 |
| Mark II FPA | N | Y | N | Y | Y | Y | 4 |
| MGM | N | Y | N | N | N | N | 1 |
| Usability | N | N | N | Y | N | N | 1 |

**Table 2**: Comparison of functional assessment and estimation methods

The rating assigned to each method in Table 2 is based on the method's satisfaction of the six criteria. If a method completely satisfies the requirements of a characteristic it receives a 'Y' in the table. Each 'Y' is worth one 'mark'. An 'N' in the table denotes that the method does not satisfy the necessary requirement and therefore receives no 'marks' for that characteristic. Clearly this is an arbitrary assignment of value to the six criteria and no weightings have been applied, in spite of the fact that some aspects may be more important than others to project managers. Moreover, characteristics other than those included in the table may also be of greater interest to managers - inclusion of such criteria in the table may lead to changes in the ratings achieved. This reflects the nature of this discussion as an exploratory comparison of the various methods, however. Indeed, it may not be comprehensive, but it should at least provide some comparative information of value to those managers considering their functional assessment and estimation options.

## 5. OPPORTUNITIES AND RECOMMENDATIONS FOR IMPROVEMENT

All nine approaches discussed above have some useful features and a few in particular would appear to be promising avenues for both practice and further research. Several issues of concern, however, have also been identified. In particular, some of the approaches have been criticised for their lack of objectivity, in that much of the assessment can be directly dependent on decisions made by individual evaluators. This is in spite of the fact that automatic measurement extraction would now seem to be a prerequisite for any successful approach [10]. Some of the methods are not completely applicable at the conceptual modelling phase and some are also not comprehensive in their assessment. Most of the methods still suffer from a lack of significant validation and are therefore likely to remain underutilised in industry.

Clearly, then, there are a number of areas in which improvements to the assessment function could be made.

Of particular importance (as illustrated in Table 3) are the issues of automatic collection, subjectivity and validation. All of these issues need to be addressed if any method, new or existing, is to be accepted by the development industry. Any degree of subjectivity places too much emphasis on the working methods of particular individual assessors - if counting methods can be interpreted differently by individuals then the measures obtained from the same system by different people are likely to vary. Consequently any recommendations based on those measures will also vary. Any new method must therefore be totally objective to ensure consistent results and conclusions.

| | Char1 Auto. | Char 2 Comp. | Char 3 Obj. | Char 4 Spec. | Char 5 Tested | Char 6 Valid. |
|---|---|---|---|---|---|---|
| Number of 'Y's | 3 | 8 | 3 | 6 | 6 | 3 |
| Number of 'N's | 6 | 1 | 6 | 3 | 3 | 6 |

**Table 3**: Satisfaction of the six criteria

As well as reducing the influence of subjectivity on the assessment procedure, automated data collection also lessens the work effort imposed on developers and assessors. Furthermore, automatic collection reduces the risk of errors being introduced into the extracted data. Finally, any new analysis procedure needs to be tested *and* validated with real-world systems to illustrate that it is indeed effective in the relevant development domain. Of the six criteria considered here, these were the most poorly fulfilled by the nine techniques.

To summarise, any new functional assessment method should enable:

- early application - the requirements specification is one of the earliest available products of the development process - analysis of this representation would enable rapid measurement and estimate determination
- objective quantification - any assessment scheme should be based totally on the functional specification of system requirements; consequently, all of the measures would be directly quantifiable in an unambiguous, assessor-independent manner
- automatic collection - collection and analysis of the measures should be incorporated into automated development tools so that collection and interpretation errors can be reduced or avoided
- comprehensive assessment - since a specification can be considered from a number of perspectives, for example, data, process and/or user interface, measures applicable to the size and interconnectivity of each perspective should be included in any new assessment scheme
- independent results - given that automation now plays a significant part in the development of business systems (with the use of CASE tools), it has been asserted that the development environment will have far less impact on the data obtained from different sites [9]; therefore results from different environments may be more easily compared
- rapid uptake - as a result of the last point it is also suggested that a lesser degree of calibration will be needed, enabling more rapid uptake of the analysis recommendations by organisations that do not have pools of recent project data
- testing and validation - new assessment schemes should be tested and validated with actual systems developed within the commercial software industry.

## ACKNOWLEDGMENTS

The work described in this paper was carried out while the author was a graduate student at Cambridge University. Financial support for the work came from 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 is also grateful to the referees for their constructive and perceptive suggestions.

## REFERENCES

[1] DEMARCO, T.: 'Controlling software projects' (Yourdon, 1982)

[2] JEFFERY, D.R., and LAWRENCE, M.J.: 'An interorganisational comparison of programming productivity'. Proc. 4th International Conference on Software Engineering, Munich, West Germany, 1979

[3] SAMSON, W.B., NEVILL, D.G., and DUGARD, P.I.: 'Predictive software metrics based on a formal specification', Information and Software Technology, June 1987, 29, (5), pp.242-248

[4] PUTNAM, L.H.: 'A general empirical solution to the macro software sizing problem', IEEE Transactions on Software Engineering, April 1978, 4, pp.345-361

[5] BOEHM, B.W.: 'Software engineering economics' (Prentice-Hall, 1981)

[6] RUBIN, H.A.: 'Macro-estimation of software development parameters: The ESTIMACS system', in 'SOFTFAIR-Software development: Tools, techniques, and alternatives' (IEEE, 1983)

[7] GRUPE, F.H., and CLEVENGER, D.F.: ‘Using function point analysis as a software development tool’, Journal of Systems Management, December 1991, pp.23-26

[8] HENRY, S., and KAFURA, D.: ‘Software structure metrics based on information flow’, IEEE Transactions on Software Engineering, September 1981, 7, (5), pp.510-518

[9] TATE, G., and VERNER, J.: ‘Software metrics for CASE development’. Proc. COMPSAC ‘91, Tokyo, Japan, 1991

[10] NORMAN, R.J., and CHEN, M.: ‘Working together to integrate CASE (Guest editors’ introduction)’, IEEE Software, March 1992, pp.13-16

[11] HENRY, S., and LEWIS, J.: ‘Integrating metrics into a large-scale software development environment’, Journal of Systems and Software, 1990, 13, pp.89-95

[12] CASE, A.F. Jr: ‘Information systems development: Principles of computer-aided software engineering’ (Prentice-Hall, 1986)

[13] WRIGLEY, C.D., and DEXTER, A.S.: ‘A model for measuring information system size’, MIS Quarterly, June 1991, pp.245-257

[14] KULKARNI, A., GREENSPAN, J.B., KRIEGMAN, D.A., LOGAN, J.J., and ROTH, T.D.: ‘A generic technique for developing a software sizing and effort estimation model’. Proc. COMPSAC ‘88, 1988

[15] LEDERER, A.L., and PRASAD, J.: ‘Nine management guidelines for better cost estimating’, Communications of the ACM, 1992, 35, (2), pp.51-59

[16] GRADY, R.B.: ‘Work-product analysis: The philosopher’s stone of software?’, IEEE Software, March 1990, pp.26-34

[17] MUKHOPADHYAY, T., and KEKRE, S.: ‘Software effort models for early estimation of process control applications’, IEEE Transactions on Software Engineering, October 1992, 18, (10), pp.915-924

[18] MUNSON, J.C., and KHOSHGOFTAAR, T.M.: ‘Applications of a relative complexity metric for software project management’, Journal of Systems and Software, 1990, 12, pp.283-291

[19] COTE, V., BOURQUE, P., OLIGNY, S., and RIVARD, N.: ‘Software metrics: An overview of recent results’, Journal of Systems and Software, 1988, 8, pp.121-131

[20] VERNER, J., and TATE, G.: ‘A model for software sizing’, Journal of Systems and Software, 1987, 7, pp.173-177

[21] TATE, G., and VERNER, J.: ‘Approaches to measuring size of application products with CASE tools’, Information and Software Technology, November 1991, 33, (9), pp.622-628

[22] SYMONS, C.R.: ‘Software sizing and estimating: Mk II FPA (Function point analysis)’ (John Wiley & Sons, 1991)

[23] BRITISH GAS: ‘Bang metric analysis’. Document Num. 000763, Process Support, British Gas plc, Dorking, UK, June 1991

[24] TATE, G.: ‘Management, CASE and the software process’. Proc. 12th New Zealand Computer Conference, Dunedin, New Zealand, 1991

[25] MACDONELL, S.G.: ‘Quantitative functional complexity analysis of commercial software systems’. Ph.D. Dissertation, Department of Engineering, University of Cambridge, Cambridge, UK, 1992

[26] GRAY, R.H.M., CAREY, B.N., MCGLYNN, N.A., and PENGELLY, A.D.: ‘Design metrics for database systems’, BT Technology Journal, October 1991, 9, (4), pp.69-79

[27] SHEPPERD, M.: ‘Design metrics: An empirical analysis’, Software Engineering Journal, January 1990, pp.3-10

[28] ALBRECHT, A.J.: ‘Measuring application development productivity’. Proc. IBM GUIDE/SHARE Applications Development Symposium, California, USA, 1979

[29] ALBRECHT, A.J., and GAFFNEY, J.E. Jr: ‘Software function, source lines of code, and development effort prediction: A software science validation’, IEEE Transactions on Software Engineering, November 1983, 9, (6), pp.639-648

[30] ROLAND, J.: ‘Software metrics’, Computer Language (USA), June 1986, pp.27-33

[31] SHEPPERD, M.: ‘An evaluation of software product metrics’, Information and Software Technology, April 1988, 30, (3), pp.177-188

[32] SYMONS, C.R.: ‘Function point analysis: Difficulties and improvements’, IEEE Transactions on Software Engineering, January 1988, 14, (1), pp.2-10

[33] RUDOLPH, E.E.: ‘Measuring information systems’. Seminar Guide and Additional Notes, Auckland, New Zealand, 1987

[34] GORDON GROUP: ‘Before You Leap - A software cost model’. Product User Manual, Gordon Group, San Jose CA, USA, 1987

[35] VERNER, J., and TATE, G.: ‘Estimating size and effort in fourth-generation development’, IEEE Software, July 1988, pp.15-22

[36] RATCLIFF, B., and ROLLO, A.L.: ‘Adapting function point analysis to Jackson system development’, Software Engineering Journal, January 1990, pp.79-84

[37] RUDOLPH, E.E.: ‘Productivity in computer application development’. Working Group Report, University of Auckland, Auckland, New Zealand, 1983

[38] CONTE, S.D., DUNSMORE, H.E., and SHEN, V.Y.: ‘Software engineering metrics and models’ (Benjamin/Cummings Publishing, 1986)

[39] LOW, G.C., and JEFFERY, D.R.: ‘Function points in the estimation and evaluation of the software process’, IEEE Transactions on Software Engineering, January 1990, 16, (1), pp.64-71

[40] KEMERER, C.F.: ‘Reliability of function points measurement: A field experiment’, Communications of the ACM, February 1993, 36, (2), pp.85-97

[41] IE: ‘IE-metrics knowledge base’. James Martin & Co., Reston VA, USA, November 1989

[42] MCCABE, T.J.: ‘A complexity measure’, IEEE Transactions on Software Engineering, December 1976, 2, (4), pp.308-320

[43] SHEPPERD, M.: ‘A critique of cyclomatic complexity as a software metric’, Software Engineering Journal, March 1988, pp.30-36

[44] KING, S.F.: ‘The quality gap: A case study in information system development quality and productivity using CASE tools’, in SPURR, K., and LAYZELL, P. (eds.): ‘CASE: Current practice, future prospects’ (John Wiley & Sons, 1992, pp.35-54)

[45] ALBRECHT, A.J.: ‘Open letter to the secretary of the international function point user group’. IFPUG Memorandum, June 1988

[46] RAMAMOORTHY, C.V., TSAI W.-T., YAMAURA, T., and BHIDE, A.: ‘Metrics guided methodology’. Proc. COMPSAC ‘85, Chicago IL, USA, 1985

[47] WILSON, M.L.: ‘The measurement of usability’, in CHEN, P.P. (ed.): ‘Entity-relationship approach to systems analysis and design’ (North-Holland, 1980, pp.75-101)

[48] MILLER, G.A.: ‘The magical number seven, plus or minus two. Some limits on our capacity for processing information’, Psychological Review, 1956, 63, pp.81-97