A Vector-Based Approach to Software Size Measurement and Effort Estimation
Hastings and Sajeev
IEEE TOSE Apr 01

Comments for the IS3 task

Were the differences in the metrics what you had expected?

Questions on Chidamber 1998?

Abstract
We propose a Vector Size Measure (VSM) that incorporates both functionality and problem complexity in a balanced and orthogonal manner. VSM is used as the input to a Vector Prediction Model (VPM) which can be used to estimate development effort early in the software life cycle.

Summary
The proposed vector representation provides a balanced view where software size is measured in terms of magnitude and gradient. The magnitude provides a measure that takes into account functionality and problem complexity in a balanced and orthogonal manner. The gradient is a ratio of problem complexity and functionality that measures the relative dimensions of systems.
Size - algebraic specification

Algebraic specification is based on Abstract Data Types (ADT) and provides a formal mathematical description of software without “over specification” [25]. It is important, particularly for software size measurement, that software is not over specified nor under specified—we require a concise, consistent, and complete representation at the right level of abstraction.

ADT

An ADT is composed of a set of functions, \( F \), and a set of rules, \( R \), such that:

\[
\text{ADT} := F \cdot R.
\]

(2)

An ADT has two major parts: 1) Functions—a set of function signatures that define the syntactic properties (interface) of an ADT and 2) Rules—a set of axioms and axioms that specify the semantic properties (meaning) of an ADT. Assertions may be specified as preconditions or postconditions. Preconditions ensure that partial functions

ASL example - functions

Account :=
(Syntactic Section)
open (Money): Account;
balance (Account): Money;
deposit (Money, Account): Account;
withdraw (Money, Account): Account;

Functionality

The functionality of an ADT is defined as the distinct number of syntactic properties measured by the sum of OPPs in the syntactic section of an ADT. The following rules apply:

1. A syntactic property is composed of one or more operators (the function identifier plus parameters that are functions) and zero or more operands (parameters).
2. The addition of one atomic unit, that defines a syntactic property, increases the functionality of an ADT by exactly one OPP.

Functionality – part 2

3. The removal of one atomic unit, that defines a syntactic property, reduces the functionality of an ADT by exactly one OPP.
4. An ADT with no syntactic properties has zero functionality.
5. An ADT with exactly one void function (that does not take any parameter nor return any value) has unit functionality.

Thus, the functionality of an abstract data type, \( \lambda \), measured in OPPs is:

\[
\lambda = \sum_{\text{OPP}} [0\text{OPP}].
\]

(4)
Complexity

The problem complexity of an ADT is defined as the distinct number of non-redundant semantic properties measured as the sum of OPs in the semantic section of the ADT. Semantic properties are specified by rules, i.e., assertions and axioms. The following rules apply:

Length

The length of an ADT is defined as the sum of atomic units specified in the syntactic and semantic sections of an ADT. The following rules apply:

The vector – magnitude and direction

Using plain vector algebra [2], magnitude, \( m \), is defined as:

\[
m = \sqrt{f^2 + c^2} \quad \text{[OPs]},
\]

are more interested in the ratio between problem complexity and functionality, i.e., the gradient, \( g \), to indicate the relative dimensions of a software system, where:

\[
g = \frac{c}{f} \quad (f > 0).
\]

Cost Model

From (14) and our previous discussion on the contributing factors, the estimated effort, \( E \), is calculated as:

\[
E' = am^2 q^2,
\]

where \( m \) is the measured magnitude of a software specification and \( g \) is its measured gradient; \( a, b, \) and \( z \) are coefficients. In order to determine the relationship between...

Scales

Magnitude is calculated using a vector transformation from functionality and problem complexity and is, therefore, a discrete ratio scale measure that is mapped to real numbers. Gradient is an absolute scale measure as it is in the form \( g(x) = x \) with no units, i.e., it cannot be logically transformed into another form.

Regression formula

By taking logs on both sides of (15), we get:

\[
\ln E' = \ln a + b \ln m + z \ln g.
\]

From this, we can get the multivariate regression formula:

\[
E' = \beta_0 + \beta_1 M + \beta_2 G.
\]

where \( E' \) is \( \ln E' \), \( M \) is \( \ln m \) and \( G \) is \( \ln g \). The coefficients \( \beta_0 \), \( \beta_1 \), and \( \beta_2 \) are determined empirically. The results of regression analysis are given in Section 7.
Representation Condition

attributes: functionality and problem complexity. Functionality and problem complexity are intuitive to the users and developers of software systems. We have

Model Coefficients

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>FP Test Fit</th>
<th>MIK Test Fit</th>
<th>Confident</th>
<th>COCOMO 2.0</th>
<th>VPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>1.96</td>
<td>2.86</td>
<td>A</td>
<td>0.18</td>
<td>0.03</td>
</tr>
<tr>
<td>$b_2$</td>
<td>5.11</td>
<td>5.34</td>
<td>A</td>
<td>1.1</td>
<td>1.07</td>
</tr>
</tbody>
</table>

Actuals vs estimates

<table>
<thead>
<tr>
<th>TABLE 1 Actual vs Estimated Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

Results

<table>
<thead>
<tr>
<th>TABLE 2 Results Compared to Test Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Criteria</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Correlation $R^2$</td>
</tr>
<tr>
<td>Error Magnitude</td>
</tr>
<tr>
<td>Standard Error</td>
</tr>
</tbody>
</table>

For Thursday

Read Gursaram’s paper and Zhang’s comments