Estimating with Enhanced Object Points vs. Function Points

Erik Stensrud
Andersen Consulting
Drammensveien 165, 0212 Oslo, Norway
&
University of Oslo, Dept. of Informatics
P.O.Box 1080, Blindern, N-0316 Oslo,
Norway
+47-92 28 1903
erik@ifi.uio.no

Introduction

In the research community Function Points is considered as a de facto standard. Even according to Gartner Group “function points will provide the primary means for measuring application size, reaching a penetration of approximately 50% of development organisations by the year 2000” (Hotle, 1996). We do not know what sources this conjecture is based upon. However, we observe that in many major companies, like Andersen Consulting, Function Points are not used to size software for estimating purposes. In stead, various kinds of what we refer to as Enhanced Object Points (EOP) are used by project managers and other practitioners to size the product and estimate the effort. This paper describes EOP and compares the two metrics and points out some reasons why many practitioners may prefer Enhanced Object Points to Function Points.

Function Points

The description of Function Points assumes the reader has some familiarity with function points. The description and discussion of Function Points is based on the IFPUG 4.0 (1994) definition.

Function Point counting involves classifying software items into transactions and data entities. Transaction counts are typically obtained from functional decomposition diagrams or data flow diagrams whereas data entity counts are obtained from entity-relationship diagrams. Transactions are further classified into external inputs (EI), external outputs (EO) and external queries (EQ). Data entities are classified into external (EIF) and internal entities (ILF). All together there are, therefore, five classes of items. Each item is furthermore classified into low, medium or high complexity. For data entity items the complexity classification is based on counts of the number of attribute types (DET) and the number of data entity subtypes (RET) within the type. For transaction types the classification is based on counts of the number of data entity types (FTR) accessed by the transaction and the number of attribute types accessed (DET).

All together there are therefore 15 different items. Each item is then weighted (see figure 7) and the weighted counts are then added together yielding the unadjusted function point count (UFP).

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>EI</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>EO</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>EQ</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>ILF</td>
<td>7</td>
<td>10</td>
<td>15</td>
<td>32</td>
</tr>
</tbody>
</table>
Figure 7: Calculation of unadjusted function points

The unadjusted count is finally adjusted by 14 complexity factors to yield the adjusted function point count (AFP). The 14 factors are each rated on a scale from zero to five and the ratings are added together to yield the adjusted complexity factor (TDI). TDI may therefore take on values in the range 0 - 70 (5x14=70). The adjusted function points are computed by this formula:

\[ \text{AFP} = \text{UFP} \times (\text{TDI} \times 0.01) + 0.65 \]

This means that AFP is in the range 0.65\text{UFP} to 1.35\text{UFP}.

Enhanced Object Points

Our definition of Object Points is an enhancement of the Object Points as presented in Banker et al. (1992) who defined Object Points based on counts of windows, reports, 3GL modules and rules. We use a somewhat different definition of Object Points which we term Enhanced Object Points (EOP). EOP is a suite of multi-dimensional size metrics sets rather than one specific metric such as the one-dimensional function point metric. The reason for having a suite of metrics is that different categories of projects have their idiosyncrasies in terms of size metrics. A mainframe custom system development project typically produces software items such as screens, dialogues, batch programs, interfaces, reports, data entities and application services. A client-server project produces slightly different items, e.g. data services (i.e. the software functions on the server, typically implemented in SQL) in addition to application services and windows rather than screens. Finally, a COTS integration project has its own idiosyncratic elements.

Below, we describe some elements used to size and estimate SAP\footnote{SAP is a subtype of COTS within the category of Enterprise Resource Planning (ERP) software packages.} projects. This is not an exhaustive list of items but rather intended to give a flavour of the metrics used in SAP projects. Some of the items are related to the size of the software deliverables whereas others are related to the size of organisational deliverables. We observe that «windows» is not an item produced in SAP projects (thus different from Object Points in Banker et al.), and we observe that interfaces frequently are subclassified. Also, a clear distinction from custom systems development is that COTS integration projects require the human organisation to adapt to the software whereas custom systems frequently were developed to fit an existing organisation. Therefore, a COTS project is just as much a standardisation effort as a software development effort, standardising work processes and information types. This is reflected in the metrics example below.

Enhanced Object Points for COTS Effort Estimation

The factors below is not a complete list of inputs used to estimate SAP projects but are rather meant to illustrate the flavour compared with Function Points.
# of Users

This drives the amount of training and training modules to be developed.

# of Business Units

This is one of main drivers behind the specification and configuration complexity and thus, effort, of the SAP modules related to operational accounting, profitability reporting and management information.

# of Sites/Countries per Business Unit

A site is generally a physical location of a production unit. A project involving many sites and countries generally requires more specification and configuration effort because of conflicts of interest and probably language barriers and multiple language and currency issues. Also, multiple sites require additional roll-out effort.

# of Plants

This is one of main drivers behind the specification and configuration complexity and thus, effort, of the SAP modules related to logistics.

# of SAP Modules

The number of modules is another indication of how complex and how difficult decision making will be since the different modules are logically grouped into business functions, and more business functions means more departments of the organisations involved and ultimately more conflicts of interest. The number of modules is also an indicator of the effort in business process reengineering and training of users.

# of Interfaces

Interfaces to legacy systems may be subclassed into e.g. EDI interfaces, fax interfaces, bar coding interfaces, etc. because the effort per unit for these subtypes is known to be different. The number of interfaces usually can be estimated early by counting the number of legacy systems that are to coexist and communicate with SAP. In a phased roll-out (as opposed to a «big bang» roll-out where the legacy systems are entirely replaced by SAP), there may be many coexisting legacy systems.

# of Conversions

This factor probably is inversely correlated with # of interfaces since the data from replaced legacy systems have to be saved in SAP before deleting these systems. Conversions may be further subclassified into manual and automated. A manual conversion is entering data from the keyboard whereas an automated usually involves writing a conversion software program plus quality assuring the data during the actual conversion.

# of SAP Enhancements

Enhancements are custom functionality or modules. Thus, the activities involve programming as in a traditional software project. The amount of customisation is not only
used to estimate programming effort, but also may be an indicator of resistance to change work habits.

# of Reports

This is the number of custom reports since SAP comes with a set of reports.

# of Third Party tools integration

This drives programming effort to integrate the tools.

Comparison of Enhanced Object Points and Function Points

The EOP taxonomy has some advantages over Function Points with respect to estimating:

- **EOPs are easy to count and to learn to count.** Learning to count Function Points requires reading several hundred pages with definitions and examples and probably passing exams to be certified. Despite this, interrater reliability is not that impressive (Kemerer’s 1993; Hihn et al.’s 1997). Learning to count interface modules and report modules seems easy in comparison. It requires virtually no training. Counting Function Points is still a manual, time consuming activity. By comparison, counting interface modules, report modules can be easily automated provided that all the objects reside in a repository.

- **EOP items are likely more correlated with effort than Function Point items, thus, more suited to estimating.** For example, we observe that batch programs require 3-5 times as much effort to build and test as application service programs. Likewise, interfaces require 3-10 times as much effort as application service programs. In Function Point taxonomy all these three object types could well have been, say, medium external inputs and therefore, contributed equally much to estimated effort. One reason interfaces require more effort than application services to build and test is that interfaces require writing test drivers to simulate the interfacing system (since the real interfacing system is in operation and cannot be used by the project team in development and test). Function Points do not capture this reality well. In addition, an interface to a legacy system often involves writing code in two languages e.g. Cobol on the legacy side and proprietary COTS languages on the other side. Linking and testing compiled objects in different languages introduces extra problems and effort (such as passing of different data types).

- **EOP is an extensible and modifiable size metric.** This is an advantage because COTS projects use other size metrics than e.g. custom Cobol projects. Also, IT enabled business transformation projects deliver additional products such as redesigned organisations and business processes. Estimating the effort for these kind of products require other size metrics than those used for sizing software.

- **EOP is more suited to prototyping oriented projects and in these cases allows estimating earlier in the life cycle.** Provided that prototyping is performed prior to creating the conceptual model, the counts of windows, reports, interfaces, application service programs will be known earlier than the counts of inputs, outputs, queries and files. The counts of the latter require a conceptual model such as a data model, data flow diagrams, functional decomposition diagrams or equivalent information.

- **EOP lets you estimate separately the effort of developing the technical infrastructure.** It is common practice in Andersen Consulting to divide the software development into application development and technical infrastructure development.
development effort is a function of the amount of functionality to be produced, the technical infrastructure development effort is not. EOP is extensible and lets you count other software items that likely are more correlated with the development effort of the technical infrastructure such as counts of hardware and software components, programming languages, network protocols, database products, etc. Therefore, EOP will likely estimate total software development effort with higher accuracy than Function Points because, contrary to our experience, Function Points assumes there is a correlation between the technical infrastructure development effort and the application development effort since Adjusted Function Points basically adds a percentage for technical complexity in stead of estimating this effort separately from the amount of functionality.

Concluding remarks

EOP seems to have the potential to yield more accurate estimates than Function Points. The down side of EOP versus Function Points is that you no longer have a universal size metric. However, we have to distinguish between the needs of estimating and benchmarking. For estimating it is not really required to have a universal metric for, say, COTS and for custom mainframe systems. However, for comparing a COTS solution with a custom solution, a universal metric is needed. However intellectually pleasing, we do not believe in a “one size fits all”, i.e. a metric that is equally useful for estimating and for benchmarking.

One advantage of Function Points such as the IFPUG 4.0 standard is that it has rigorous definitions of the individual items that are counted. A good use of the IFPUG 4.0 standard would therefore be to adapt these definitions to the EOP to sharpen the definitions of some of the concepts such as application service program, interface, batch module, conversion program, window, etc. to further increase interrater reliability.

References


