CONSISTENTLY MEASURING THE SIZE OF A MULTIMEDIA APPLICATION

Jan-Willem van Aalst
Origin IT Services
Dept. WEBS – Business Communication
Jan-willem.vanaalst@nl.origin-it.com

Charles van der Mast
Delft University of Technology
Dept. ITS – Information systems
c.a.p.g.vandermast@is.twi.tudelft.nl

KEYWORDS
multimedia, software engineering, software process improvement, fundamental research

ABSTRACT
In the research world of software engineering, much work has been done to find unambiguous objective formulas for determining the size of a software application (Putnam and Myers 1996). Software process improvement tools have gratefully employed the results of that research to develop tools that can be used to improve the maturity of the software project process. However, these tools have up to now always failed to take into account the more complex highly interactive systems such as multimedia products (Vaughan, 1994). We present here a relatively easy way to objectively determine the size of a multimedia applications. We call the unit of this size the Siun (for Size Unit).

PROBLEM DEFINITION
When first attempting to determine the size of a software application, researchers started to count non-breaking lines of code. This could not include comment statements or empty lines. Many complex rules have been devised to make this counting an objective and unambiguous task. However, it was not long after this method became commonplace, when people realised that the size of a software application could not be determined by just counting lines of code. Hence, more sophisticated methods were devised. The most widespread of these by far is Function Point Analysis (FPA) (Software Productivity Consortium, 1995). Though this method seems to work in a satisfactory way for many software projects, the field of multimedia needed something in addition, because the effort that is put into the production of a multimedia product is far more complex than software engineering issues alone (England and Finney, 1996, and Van Aalst and Van der Mast, 1998b). More often than not, the heaviest components of effort are in visual and graphic design, video and audio production, or functional requirements analysis (Van der Mast, 1995). Therefore, we cannot suffice by just counting lines of code or function points. In many modern multimedia production tools, lines of code are completely absent. Function points are no good indicator either, because (for example) the megabytes of graphics that are used in multimedia product x, may well have taken about 50% of all the effort put into the project (Van Aalst and Van der Mast, 1998a). The field of multimedia production needs a way of determining the size of a product that takes into account these effort factors.

The question that props up immediately, then, is why do we need to know the size of a multimedia application after all? The answer is: to be able to supply the software improvement researchers or organisations with a handhold for their solutions. Since these companies claim to be able to realise serious process maturity improvement (Software Productivity Consortium, 1995) as long as we only have some basic variables in place (especially size!), we can now supply this size factor. Hence, multimedia project teams are able to supply all necessary variables for their own process maturity improvement.

METHOD AND EXPERIMENT
To construct our formula, we use – on an abstract level – the basic principle of Function Point Analysis, namely to take important relevant aspects (though not functions) of the product and to assign a weight to each of these aspects. This weight is determined according to the average amount of effort that is required to produce that aspect of the product. The average amount of effort, in turn, is determined from a sufficiently large number of projects that result in these multimedia products. Thus, the research method is as follows:
1. Conceive a list of relevant effort-consuming aspects of a product, in such a way that they do not overlap in project tasks;
2. Determine a first estimation for the weight of each of these factors.
3. Analyse five to six multimedia products on these aspects;
4. Fine tune the weight factors according to one-to-one comparison of these product analyses.

In analysing six multimedia products which were all realised in either 1997, 1998 or 1999, we conceived the list of aspects of table 1.

**Table 1. Determining the size of a multimedia application**

<table>
<thead>
<tr>
<th>MULTIMEDIA PRODUCT ASPECT</th>
<th>WEIGHT</th>
<th>UNIT</th>
<th>COMPLEXITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of html code (# characters div. by 40)</td>
<td>0.10</td>
<td>Count(Char)</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Lines of dynamic html code (# characters div. by 40)</td>
<td>1.20</td>
<td>Count(Char)</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Lines of script code</td>
<td>1.80</td>
<td>Count(Char)</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Uncompiled lines of 3rd gen. language code</td>
<td>3.20</td>
<td>Count(Char)</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Uncompiled lines of 2nd gen. language code</td>
<td>3.50</td>
<td>Count(Char)</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Graphics sources</td>
<td>0.30</td>
<td>Kilobytes</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Animated graphics sources</td>
<td>0.40</td>
<td>Kilobytes</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Realtime audio</td>
<td>0.40</td>
<td>Kilobytes</td>
<td>[1,2]</td>
</tr>
<tr>
<td>(Streaming) Video</td>
<td>0.45</td>
<td>Kilobytes</td>
<td>[1,2]</td>
</tr>
<tr>
<td>Database structure: # tables + # fields + #rel.</td>
<td>3.00</td>
<td>Count()</td>
<td>[1,2]</td>
</tr>
</tbody>
</table>

**SYNTHESIS**

The weight of each multimedia aspect not only depends on the amount of effort that is required to produce it, but also on the unit that is chosen for that aspect. For example, graphics sources may well take up several thousands of kilobytes, and if not properly weighted, their influence would unrightfully overwhelm the effort for the lines of code (if any). The unit column denotes what the result is of what is counted.

The complexity column is something special. The fact that, for example, the graphics factor is not large, does not mean that it did not take a large amount of effort to create these graphics. Therefore, we can multiply the value by a complexity factor which ranges from 1 (not complex) to 2 (very complex), with digits inbetween possible (for example, 1.65). Please note that we can also include web-based multimedia products since html and other scripting languages may also be included.

**EXPERIENCES WITH SIUNS**

We have used an initial set of weight factors to compare two multimedia products produced by Origin for international customers. Both of these were the results of similar-scoped projects, with a Computer Based Training as a result. Weights that seemed to have a disproportionally large effect on the total size were then adjusted. The same was done for a third and a fourth project. For a fifth project, we found that the weight factors were by then well-tuned. All projects were in the range of $30,000 to $50,000. This one resulted in 2835 Siuns, which felt right when compared to less complex projects that resulted in products of 1146 and 1520 Siuns, respectively. However…

One notable subjectiveness, one that cannot easily be eliminated, is the complexity factor. One employee that would regard the graphics as highly complex, could be classified as rather simple by another.

Obviously, a larger set of projects would be needed to produce more statistically valid weight factors.

Please note that the Siun is not a measure for the size or complexity of the project; it particularly serves as variable about the size of multimedia products, especially for software process improvement tools.

**CONCLUSIONS**

While the software engineering industry is currently equipped with proven methods to predict project success through variables such as milestones, project cost, required effort, number of defects, and product size, the multimedia industry is now able to supply the most difficult of these variables, namely size, as well. The fact that the multimedia size unit, called Siun, is an unknown to software engineering, is not a problem, because these prediction methods need an arbitrary but
consistent approach to counting. Therefore, the research on multimedia software process improvement now has a useful handhold for methods for predicting project success.

FURTHER RESEARCH

The results of this multimedia research are used in a larger research project on improving control over multidisciplinary projects (Van Aalst and Van der Mast, 1998a). More information about this project may be found on http://is.twi.tudelft.nl/~jwva/.

ACKNOWLEDGMENTS

A thank-you for valuable feedback: Prof. Jan Dietz, Toon Witkam, Johan Vader, Prof. Gerard Kempen, Johan Versendaal, Richard Thoben.

LITERATURE


