Functional Size Estimation Technologies for Software Maintenance

Christof Ebert and Hassan Soubra

Estimating functional size is the key input for building software models. Unlike direct effort estimates, software size estimation gives a measure of the software product itself and can be used to build objective estimation models for predicting project effort and duration, estimating defects for quality and service cost predictions, and obtaining software development productivity ratios.

Although maintenance dominates software projects, the necessary technologies for estimating maintenance effort have been rather poorly used to date. Here, we examine estimation technologies based on the COSMIC (Common Software Measurement International Consortium, ISO/IEC 19761:2011) method as it’s applied to maintenance projects.

Functional Change
Software projects in industry typically fall into two camps—new software development and software evolution—with the latter happening more frequently than the former. Functional change tends to dominate software evolution; it’s driven by a change request to a piece of software’s functional requirements that adds, modifies, or deletes tasks or functions. In practice, change requests can be complex because they might integrate both functional and nonfunctional changes. Moreover, they’re described at various levels of abstraction and with several connotations that can themselves be a mix of different elements.

Figure 1 shows the functional change request process between customer and supplier. Currently, the documentation of such change is poorly performed if at all. To establish a measurement process and reduce the possibility of human errors, functional change should be well documented and modeled.

Estimating a Change’s Functional Size
An estimate is a quantitative assessment of the likely outcome of a future endeavor. It’s usually applied to forecast project costs, size, resources, effort, or duration. Given that estimates are, by definition, not very precise, they should always include some indication of accuracy (such as plus or minus x percent).1–3

Companies use estimation to more effectively manage their budgets and schedules. Typically, they use estimation in

- project management to estimate size and effort before a project starts to better allocate resources within the project;
• developing software products or services, such as when a supplier has to estimate the effort needed for a change request or project during a bid, before having all the technical specifications available;
• outsourcing software projects because estimation provides part of the answer to the “make, reuse, or buy” question that many companies face when optimizing their software and IT production; and
• productivity improvement because what gets measured can be improved. With fierce cost pressure and worldwide competition, productivity benchmarks based on estimation techniques pinpoint how to improve and stay competitive.

The dynamics of the software market, with its increasing use of external components and adaptation of code instead of writing it from scratch, has led to extended or new kinds of technologies for estimation. Gradually, estimation has moved away from being about size toward something that focuses more on functional estimates. Standards are evolving because estimates play a crucial role in business, and enormous amounts of money are at stake.

Estimates are often confused with goals or plans. Projects are scheduled according to needs, but they aren’t always in line with feasibility, and commitments are given to clients about something being “very urgent and important” before checking how this “urgency” relates to existing commitments and capacity planning. In fact, most failures in software projects come from not understanding and considering important gaps among goals, estimates, and plans.1–3

To deliver more reliably and to better manage their projects, companies are interested in measuring and estimating the maintenance effort for their software change deliveries. Most companies use analogies or best guesses, but soon realize that the accuracy is insufficient and that there’s no repeatability. So, they look for more reliable estimation technologies. The manufacturer, with this requirements measurement, can use benchmarks and negotiate with suppliers when developing its software products. For instance, during a tender phase, in which a software supplier provides an offer for a functional change or a maintenance activity, the client is interested in understanding the reasoning behind the change effort and any benefits. Customers often also want to benchmark their suppliers to find out whether their productivity is sufficient compared to other suppliers on the market.

Functional Size Measurement (FSM) is used in industry as a baseline on which to build estimates. It takes as input change requests and their analysis vis-à-vis the system in which the change must be incorporated. Based on this initial analysis of change impacts, an effort model helps derive the effort required to implement the underlying changes. Environmental factors such as software type, team distribution, or project repeatability help when adjusting the effort estimate. Obviously, the same change can cost a lot more effort if the supplier has insufficient processes or project management skills. Figure 2 shows an FSM procedure applied to a functional change to estimate the change’s implementation effort. It takes the project requirements and change requests as inputs and yields functional change size for the overall effort estimation.1

The COSMIC Method
COSMIC brings together software measurement experts from around the world; they built the method in the late 1990s to improve on traditional function point methods. The International Standards Organization classified COSMIC as a second-generation FSM method for both the business application and real-time domains. It’s used worldwide in various industries, and unlike other
methods, it can be used in embedded systems.\textsuperscript{4–6} Although the current measurement guidelines include several refinements, the COSMIC method’s original principles have remained unchanged since they were first published in 1999.

The COSMIC method measures software’s functional user requirements (FUR). The result obtained is a numerical “value of a quantity” (as defined by the ISO) representing the software’s functional size. Functional sizes measured by the method are designed to be independent of any implementation decisions embedded in the operational artifacts of the software to be measured. This means that FUR can be extracted not only from actual software already developed but also from the software model before it’s implemented.

The measurement process consists of three phases:

- The measurement strategy phase, where a software context model is applied to the software to be measured.
- The mapping phase, where a generic software model is applied to the software to be measured.
- The measurement phase, where actual size measurement results are obtained.

Software functionality is considered within the functional flows of data groups. These dataflows can be characterized by four distinct types of data movements: two types of movements (E: ENTRY and X: EXIT) allow the exchange of data with users across a boundary, and the other two types (R: READ and W: WRITE) allow the exchange of data with persistent storage. The measurement result corresponds to the functional size of the measured software requirements.
(FUR) and is expressed in COSMIC function points (CFP).

Functional evolution or change is interpreted as any combination of additions of new data movements or of modifications or deletions of existing data movements. The CFP count for sizing any of these changes is the same—that is, one CFP is attributed to any data group movement that has been added, deleted, or modified. However, it’s important to distinguish the various types of evolution when making performance measurements and estimates.

**FSM Technologies and Tools**

The entirely manual application of FSM procedures is tedious and time-consuming for organizations with a large number of projects to measure within a very short time frame, both for project estimation purposes and for productivity studies. In addition, applying FSM manually to a very large set of requirements requires specialized expertise when these requirements are complex. Recently, technologies and tools have emerged to facilitate automating requirements. They still need formal and defined descriptions depending on modeling language, but Table 1 provides an overview of recent tools that support FSM application.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISBSG Comparative</td>
<td>Used for estimation purposes; produces estimates of effort, delivery rate, duration, and speed of delivery of software projects using function points based on either COSMIC or IFPUG. The estimates are generated using delivery rates and speed of delivery averages of a selection of projects in the ISBSG repository.</td>
<td><a href="http://www.isbsg.org">www.isbsg.org</a></td>
</tr>
<tr>
<td>Estimating Tool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Point Workbench</td>
<td>Estimation embedded in a project management tool. Supports software project sizing and evaluation. Designed to be scalable, and hence can be used for individual projects as well as in large-scale, distributed IT environments.</td>
<td><a href="http://www.charismatek.com">www.charismatek.com</a></td>
</tr>
<tr>
<td>KnowledgePLAN</td>
<td>Software estimation tool to create and refine detailed project plans for project management systems. The tool lets users size projects and estimate work, resources, schedule, and defects. A project’s strengths and weaknesses can be evaluated as well, to determine their impact on quality and productivity.</td>
<td><a href="http://www.spr.com/index.php">www.spr.com/index.php</a></td>
</tr>
<tr>
<td>MeterIT-Cosmic</td>
<td>Support tool that gives support to metrics practitioners for CFP, using software requirement specifications as inputs. The tool supports both MIS and real-time software, giving the possibility of estimating size when software isn’t completely defined.</td>
<td><a href="http://www.telmaco.com/cosmic1.html">www.telmaco.com/cosmic1.html</a></td>
</tr>
<tr>
<td>MeterIT-Project version</td>
<td>Software benchmarking and estimation tool for predicting projects’ time/cost and managing strategies; in addition to the MeterIT-Converter tool, for converting the IFPUG, NESMA, and MKII functional sizes of software to CFP.</td>
<td><a href="http://www.telmaco.com/meteritproject1.html">www.telmaco.com/meteritproject1.html</a></td>
</tr>
<tr>
<td>Scope</td>
<td>Estimation embedded in a project management tool. Supports IFPUG function point analysis (FPA) method. Usable at the initial stages of project development, allowing the evaluation of a decision’s impact to include or exclude functionalities. Lets users keep track of each application’s functional history by providing a documented audit trail of changes to software throughout its life.</td>
<td><a href="http://www.totalmetrics.com">www.totalmetrics.com</a></td>
</tr>
<tr>
<td>Siesta</td>
<td>Supports sizing and estimation by using the NESMA and COSMIC methods of software functional sizing.</td>
<td><a href="http://metrieken.sogeti.nl">http://metrieken.sogeti.nl</a></td>
</tr>
<tr>
<td>SLIM-Estimate</td>
<td>Estimation environment for cost, time, and effort. Helps satisfy a given set of system requirements and determine strategies for designing and implementing software. In addition to cost estimation, it provides configurability to accommodate different design processes (agile development, package implementation, model-based development).</td>
<td><a href="http://www.qsm.com">www.qsm.com</a></td>
</tr>
</tbody>
</table>
the environmental needs to be considered when establishing estimation. There’s no automatic standard for counting because each modeling approach and environment needs specific “translations.” Insufficient data quality and environmental constraints require experienced counting to avoid errors. Thus, the maturity and competences in using the development process, project management, configuration management, and quality management all impact estimation results and must therefore be stabilized to achieve sufficient estimation accuracy.

Companies can use FSM to estimate development effort, manage project scope changes, measure productivity, benchmark, and normalize quality and maintenance ratios. We’ve introduced estimation for software changes here, which is increasingly demanded both for better project estimation and also for supplier management.

We close with some observations about estimation reality:

- Consider business impacts. Clearly distinguish goals, estimates, and plans. Estimation helps mitigate business risks, so target estimation accuracy in line with your business needs.

- Establish repeatability. Immature processes will always create ad hoc behaviors with unpredictable results. Establish a robust process to report and store data.

- A fool with a tool remains a fool. Don’t use models you don’t know. Never use a tool to camouflage insufficient process (“the tool says so”). Carefully introduce an estimation tool and evaluate different tools, considering needs and cost. Consider total cost of ownership. Provide adequate training and coaching on estimation principles.

- Garbage in, garbage out. Tools don’t provide value if the information is missing. Clarify the underlying data collection and estimation approach. Verify and validate data before storing it in historical databases. Use standard measurements.

- Use estimation to grow. Continuously improve. Don’t stay with the same parameters for longer than a year, and benchmark with other companies to focus your improvements. Challenge results and improve your efficiency each year.

We achieved with client companies on this basis an estimation accuracy of 10 to 20 percent between initial effort estimation during the
analysis phase and the final effort to implement the change, which in most cases is sufficient. Such improvement in effort estimation typically takes around a year, during which it’s necessary to build a history database, analyze measurements, and set up the estimation process. Naturally, the requirements as outlined above, such as sufficient process maturity and project management competences, are prerequisites to improve the accuracy of estimating maintenance effort.

References

CHRISTOF EBERT is managing director at Vector Consulting Services, is a senior member of IEEE, and serves on the IEEE Software editorial board. Contact him at christof.ebert@vector.com.

HASSAN SOUBRA is an assistant professor and member of the embedded-systems research team at ESTACA—Engineering School, France. Contact him at hassan.soubra@estaca.fr.