Proposal to add a section on ‘Non-Functional Requirements’ to the COSMIC Method as defined in the ‘Measurement Manual’

1. Current statement of limitations on applicability of the COSMIC Method to measuring ‘Functional User Requirements’

The Measurement Manual version 3.0.1 has a section 1.2 entitled ‘Functional User Requirements’ in which it is made clear that the COSMIC method aims to measure only these ‘FUR’. The COSMIC method is a pure ‘Functional Size Measurement Method’, as defined in the ISO/IEC 14143/1 standard on FSM.

2. Problems with the current statement of limitations

The current description of the design goal of the COSMIC method leaves open the question of “what about Non-Functional Requirements (or NFR)? Can they be measured and if so, how should we measure them?”

In extreme cases, e.g. of mission-critical systems such as air traffic control or financial trading systems, NFR can account for a high proportion of the total requirements documentation. In such cases, estimating the project effort will need to consider the influence of NFR just as much as that of the FUR. The COSMIC method therefore needs to say something about how to deal with NFR.

When the Measurement Manual is next updated, section 1.2 on ‘Functional User Requirements’ will be re-structured and a new section 1.2.4 on ‘Non-Functional Requirements’ will be added. The re-structured text of section 1.2 will convey the following.

Proposed text for a new section 1.2.4 ‘Non-Functional Requirements’

The ISO/IEC 14143/1:2007 standard on the Principles of Functional Size Measurement lists various types of ‘user requirements’ that are not Functional User Requirements (FUR). By implication these are Non-Functional Requirements (NFR). They include but are not limited to:

- quality constraints (for example usability, reliability, efficiency and portability);
- organizational constraints (for example locations for operation, target hardware and compliance to standards);
- environmental constraints (for example interoperability, security, privacy and safety);
- implementation constraints (for example development language, delivery schedule).

NFR can be very significant for a software project. In extreme cases, a statement of requirements for a software-intensive system can require as much documentation for the NFR as for the FUR. As a result, NFR can require as much project effort to implement as the FUR. But the distinction between NFR and FUR is not as simple as it first appears. The COSMIC method can be used to measure some requirements that may first appear as non-functional.

First we need to define NFR.

**DEFINITION – Non-Functional Requirement**

Any requirement for or constraint on a hardware/software system or on a project to develop or maintain such a system, except a functional user requirement for software or a requirement that evolves into a functional user requirement for software.
Several studies\(^1\) have shown that some requirements that initially appear as system NFR evolve as a project progresses into a mixture of requirements that can be implemented in software functions, and other requirements or constraints that are truly 'non-functional'. This is true for many quality and environmental constraints from the list above. Once identified, such software functions can be sized using the COSMIC method, if necessary, just as any other software functions. Not recognizing the functional size that can be 'hidden' in NFR at the beginning of a project is one reason why software sizes can appear to grow as a project progresses.

**BUSINESS EXAMPLE**: The requirements for a new software system include the statement ‘the user shall have the option to secure files by encryption’. The project to develop the system is at the stage of estimating effort and cost. Two options are considered:

- Develop some proprietary encryption software. For project estimating purposes it may be necessary to measure the size of the requirements for the encryption software,

- Purchase an existing COTS package. For project estimating purposes it may be necessary to measure only the size of software needed to integrate the COTS package. The cost of the package and the effort to integrate and test the file encryption package will also need to be considered in the project cost estimate.

**REAL-TIME EXAMPLE**: Fault tolerance on aerospace systems is achieved mostly through a combination of redundancy and backup of the physical systems. A function such as Engine Monitoring is copied on three or four separate embedded computers. This function has a strict timing constraint stated as a NFR: “each separate computer must respond within a specific time. If any one of the computers repeatedly responds later than the required time, or its results disagree with the others, it must be out-voted" (by a mechanism specified as a functional requirement). A requirement that is initially stated as non-functional therefore evolves into FUR that can be measured. The timing mechanism can also be partly implemented in software and this functionality can also be measured (see for example the Guideline for sizing real-time software’, section 3.2).

For more examples, see Appendix ‘XX’ of this Measurement Manual.

These examples and those in Appendix XX also demonstrate that when there is a requirement to measure a size of some software early in the life of a project, it is important to consider whether some NFR could evolve into software FUR and whether the size of these software FUR should also be measured.

**Proposed new Appendix 'XX'**.

**Appendix XX: Examples of Non-functional Requirements that evolve partly or wholly into Functional User Requirements**

The following table lists a few examples of requirements statements that may appear initially at the system level as non-functional but that evolve, wholly or partly as a project progresses, into a mixture of FUR for software and statements of requirements that are truly 'non-functional'.

- Col. 1: examples of statements of system NFR
- Col. 2: examples of software FUR that might result, as a project progresses, from the system NFR in Col. 1. The FUR may be for software to be developed or to be acquired e.g. ‘COTS’ (Commercial Off-the-Shelf) software.

---

\(^1\) (Example of several papers by the same authors) Al-Sarayreh, K.T. and A. Abran, Specification and Measurement of System Configuration Non Functional Requirements, 20th International Workshop on Software Measurement (IWSM 2010), Stuttgart, Germany, 2010
- Col. 3: examples of the requirements and constraints on the system or project that might remain after separating out the software FUR in column 2. These are therefore true ‘non-functional’ requirements.

<table>
<thead>
<tr>
<th>System requirements that may initially appear as Non-Functional</th>
<th>Examples of FUR for software, to be developed or acquired, that may evolve from the initial system NFR</th>
<th>Examples of true NFR that remain after some initial system requirements have evolved into software FUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system response time during peak-hour shall not exceed an average of X seconds.</td>
<td>Software to:</td>
<td>• Specific (fast) hardware</td>
</tr>
<tr>
<td></td>
<td>• Feed external data needed by the system in real-time</td>
<td>• Some software to be written in a low-level language</td>
</tr>
<tr>
<td></td>
<td>• Monitor and report on the average response time</td>
<td>• The specific response time target statement</td>
</tr>
<tr>
<td>The system availability shall exceed Y% averaged over each calendar year</td>
<td>Software to enable fast switching of processing to a back-up processor without interruption to service</td>
<td>• Back-up hardware processor operating in ‘hot standby’ mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The specific availability target statement</td>
</tr>
<tr>
<td>Application parameters shall be easily maintainable by user staff</td>
<td>Software to enable users to maintain parameter tables</td>
<td>(None)</td>
</tr>
<tr>
<td>The system shall be usable by members of the general public with no training and a 2% successful completion rate</td>
<td>Software to:</td>
<td>• Requirements for Braille keyboards</td>
</tr>
<tr>
<td></td>
<td>• provide comprehensive Help facilities</td>
<td>• Extensive testing by members of the general public</td>
</tr>
<tr>
<td></td>
<td>• provide well-structured Menus for ease of use</td>
<td>• The specific target completion rate statement</td>
</tr>
<tr>
<td>The user shall have the option to secure files by encryption</td>
<td>Software to encrypt and decrypt files on demand of the user</td>
<td>Use of a hardware ‘dongle’ or encryption key device</td>
</tr>
<tr>
<td>The system shall be portable across X, Y and Z hardware/software environments</td>
<td>A layer of software to isolate the main functionality from the specific interface requirements of the X, Y and Z environments</td>
<td>Use of a highly portable language such as Java</td>
</tr>
</tbody>
</table>

**Measurement Practices Committee**

Date: December 2012