Measurement Model of Software Requirements Derived from System Maintainability Requirements

Alain Abran¹, Khalid T. Al-Sarayreh², Juan J. Cuadrado-Gallego³
¹²Software Engineering Department, University of Quebec (ETS)
1100 Notre-Dame west, Montréal, Québec H3C 1K3, Canada
³Departamento de Ciencias de la Computación, Universidad de Alcalá
28805 Alcalá de Henares, Madrid, Spain
¹alain.abran@etsmtl.ca, ²khalid.al-sarayreh.1@ens.etsmtl.ca, ³jjcg@uah.es

Abstract—Maintainability is typically described initially as a non functional requirement at the system level. Systems engineers must subsequently apportion these system requirements very carefully as either software or hardware requirements to conform to the maintainability requirements of the system. A number of concepts are provided in the ECSS, ISO 9126, and IEEE standards to describe the various types of candidate maintainability requirements at the system, software, and hardware levels. This paper organizes these concepts into a generic standards-based reference model of the requirements at the software level for system maintainability. The structure of this reference model is based on the generic model of software requirements proposed in the COSMIC – ISO 19761 model, thereby allowing the measurement of the functional size of such maintainability requirements implemented through software.


1. Introduction

Non-functional requirements (NFR) play a critical role in system development, including as selection criteria for choosing among alternative designs and ultimate implementations. NFR may also have a considerable impact on project effort, and should be taken into account for estimation purposes and when comparing project productivity.

Typically, these NFR are described at the system level and not at the software level, and there is no consensus yet on how to describe and measure these system NFR. In practice, NFR can be viewed, defined, interpreted, and evaluated differently by different people, particularly when they are stated vaguely and only briefly [1-3]. Therefore, it is challenging to take them into account in software estimation and software benchmarking: NFR have received less attention in the software engineering literature and are definitely less well understood than other cost factors [3]. Without measurement, it is challenging to take them as quantitative inputs into an estimation process and productivity benchmarking.

In practice, the requirements are initially typically addressed at the system level [4-10] either as high-level system functional user requirements (system FUR) or as high-level system non-functional requirements (system NFR). The latter must usually be detailed, allocated and implemented in either hardware or software, or both, as software FUR, for instance.

For example, a system FUR will describe the required functions in a system, while a system NFR will describe how the required functions must behave in a system. In the software requirements engineering step, system NFR can then be detailed and specified as software FUR to allow a software engineer to develop, test, and configure the final deliverables to system users.

The term “functional” refers to the set of functions the system (including the software) has to offer, while the term “non-functional” refers to the manner in which such functions are performed. FUR is typically phrased with subject or predicate constructions (i.e. noun/verb), such as: “The system must print 5 reports”. NFR, by contrast, are typically phrased with adverbs or modifying clauses, such as: “The system will print 5 reports quickly” or “The system will print 5 reports with a high degree of reliability”.

In the ECSS (European Cooperation on Space Standardization) standards for the aerospace industry [11-14], the ISO 9126 [15] and IEEE 830 [16] standards, a number of concepts are provided to describe various types of candidate maintainability requirements at the system, software, and hardware levels. However, these standards vary in their views, terminology, and coverage of maintainability.

Currently, there exists no generic model for the identification and specification of software FUR for implementing system maintainability requirements (system NFR) based on the various views documented in international standards and in the literature [1-14]. Consequently, it is challenging to measure these maintainability-related software FUR, and take them into account quantitatively for estimation purposes.

This paper focuses on a single type of NFR, that is, system maintainability requirements, and reports on the work carried out to define an integrated view of software FUR for system maintainability NFR, on the basis of international standards including the use of the generic COSMIC – ISO 19761 [17] model of software FUR.

The paper is organized as follows. Section 2 presents the view of software FUR in ISO 19761. Section 3 identifies the standards describing
maintainability requirements. Section 4 presents a standards-based definition of a generic model of requirements for software to implement system maintainability NFR. Finally, a discussion is presented in section 5.

2. A Generic view of software FUR in ISO

ISO 14143-1 [18] specifies that a functional size measurement (FSM) method must measure the software functional user requirements (FUR). In addition, the COSMIC – ISO 19761 [17] model proposes a generic model of software FUR that clarifies the boundary between hardware and software. Fig. 1 illustrates the generic flow of data from hardware to software from a functional perspective. From this generic model of software FUR, depicted in Fig. 1, the following observations can be made:

- Software is bounded by hardware. In the so-called “front end” direction (i.e. center of Fig. 1), software used by a human is bounded by I/O hardware such as a mouse, a keyboard, a printer, or a display, or by engineered devices such as sensors or relays. In the so-called “back end” direction (i.e. the right-hand side of Fig. 1), software is bounded by persistent storage hardware like a hard disk, or RAM or ROM memory.

- Software functionality is embedded within the functional flows of data groups. Such data flows can be characterized by four distinct types of data movements. In the “front end” direction, two types of movements (ENTRIES and EXITS) allow the exchange of data with users across a boundary. In the “back end” direction, two types of movements (READS and WRITES) allow the exchange of data with the persistent storage hardware.

- Different abstractions are typically used for different measurement purposes. In real-time software, the users are typically the engineered devices that interact directly with the software, that is, the users are considered the I/O hardware. For business application software, the abstraction commonly assumes that the user is one or more humans who interact directly with the business application software across the boundary; the I/O hardware is ignored.

As an FSM method, COSMIC is aimed at measuring the size of software based on identifiable FUR. Once identified, those requirements are allocated to hardware and software from the unifying perspective of a system integrating these two “components”. Since COSMIC is aimed at sizing software, only those requirements allocated to the software are considered in its measurement procedure.

3. Identification of standards describing maintainability requirements

This section presents a survey of the maintainability-related views, concepts, and terms in the ECSS, ISO 9126, and IEEE-830 standards. This section identifies which standards currently address aspects of the software FUR derived from system maintainability FUR and NFR – see Fig. 2. The expected outcome is the identification of the various elements that should be included in the design of a standards-based framework for modelling software FUR for system maintainability. The elements of maintainability are dispersed in various system views throughout various ECSS standards and are expressed as either:

- System maintainability functional user requirements (system maintainability FUR), or
- System maintainability non-functional requirements (system maintainability NFR)

![Fig. 2: Mapping system NFR into software FUR for maintainability](image)

3.1 ECSS: views and concepts for maintainability

Maintainability in the ECSS standards is considered as part of the integrated support requirements in system engineering, including related activities and procedures. Table 1 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related maintainability requirements. For instance, the ECSS specifies that, for system maintainability, analysis of failure modes, effects, and criticality (FMECA) must be carried out. ECSS does not specify, however, if such requirements must be implemented in software, hardware, or a combination of the two.

While conducting the survey of all the maintainability concepts and terms described in the ECSS-E-40 and ECSS-Q series and in ECSS-ESA as the integrated standard for ECSS-E and ECSS-Q, it was observed that:

- These various maintainability elements are described differently, and at different levels of detail;
• The maintainability elements are dispersed throughout the various documents: there is, therefore, no integrated view of all types of candidate maintainability requirements;
• There is no obvious link between the maintainability requirements in ECSS-ESA [19] as the integrated standard and all the other ECSS standards that describe maintainability requirements.

Table 1: Maintainability view and vocabulary in ECSS

<table>
<thead>
<tr>
<th>Key view</th>
<th>Concepts and Vocabulary</th>
</tr>
</thead>
</table>
| Part of the integrated support requirements in system engineering, including activities and procedures | • Maintainability activities and procedure.  
• Maintainability operations  
• Environment control and life support systems design (ECLSS)  
• Failure modes, effects, and criticality analysis (FMECA)  
• Failure modes and effects analysis (FMEA)  
• Mean-time-to-repair and System down-time  
• Fault detection and isolation capability  
• System malfunction. |

It is also to be noted that the ECSS does not propose a way to measure such software maintainability requirements, and, without measurement, it is challenging to take such an NFR either as a quantitative input to an estimation process or in productivity benchmarking.

3.2 IEEE: views and concepts for maintainability

IEEE-830 [16] lists maintainability as one of the NFR type, but does not define it, nor does it provide guidance on how to describe and specify the maintainability requirements; of course, it does not provide guidance on how to measure any of these NFR either.

IEEE-14764 [20] and IEEE-982.1 [21] only define the maintainability requirement as the capability of the software product to be modified, without mentioning how to describe and specify the maintainability requirements.

3.3 ISO9126: views and concepts for maintainability

The key view on maintainability in the ISO 9126 series is from the perspective of the quality of the software product: maintainability is presented as a ‘quality characteristic’, which is decomposed into quality sub characteristics and then into proposed derived measures to quantify those quality sub characteristics. The inventory of related concepts and vocabulary on software maintainability, such as analyzability, changeability, is presented in Table 2.

Table 2: Maintainability view & vocabulary in ISO 9126

<table>
<thead>
<tr>
<th>Key views</th>
<th>Concepts and Vocabulary</th>
</tr>
</thead>
</table>
| Maintainability quality characteristic; The capability of the software product to be modified. Modifications may include corrections, improvements, or adaptation of the software to changes in environment | • Analysability  
• Audit Trial Capability  
• Failure Analysis Capability  
• Status Monitoring Capability  
• Diagnostic Function Support  
• Changeability  
• Change Efficiency  
• Software Change Control Capability  
• Modifiability  
• Stability  
• Modification Impact  
• Change Success Ratio  
• Testability  
• Availability of Built-in Test Function  
• Retest Efficiency  
• Test Restart Ability |

A large number of measures are proposed in ISO 9126, but none addresses software FUR, only the maintainability NFR of the software itself. However, nothing precludes the use of these concepts at the system level or looking at what functions must be performed at the software level (i.e. FUR allocation to software) to implement these system level NFR.

4. A standards-based definition of a generic model of software-FUR for system maintainability requirements

This section identifies and assembles the terminologies and concepts of maintainability dispersed throughout the ECSS, IEEE, and ISO standards. These terminologies are mapped next into a proposed model of maintainability software FUR using the generic FUR model proposed in COSMIC. This COSMIC-based generic model then becomes a framework for describing the software FUR from system maintainability requirements based on the ECSS standards.

4.1 Mapping maintainability views and vocabulary from standards

Table 3 presents the system maintainability requirements that are present either as system requirements in the ECSS standard or as maintainability-related concepts in ISO 9126: each of these could be interpreted, and specified, at times as software FUR.

4.2 Types of maintainability requirements

Next, four types of system-related maintainability requirements can be derived:
• System Analyzability  
• System Changeability  
• System Stability  
• System Testability
Table 3: Maintainability in ECSS & ISO 9126

<table>
<thead>
<tr>
<th>System Maintainability Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Data Operation</td>
</tr>
<tr>
<td>Failure Data Monitoring</td>
</tr>
<tr>
<td>Failure Data Control</td>
</tr>
<tr>
<td>System Failure Tasks</td>
</tr>
<tr>
<td>Failure Isolation</td>
</tr>
<tr>
<td>Failure Detection</td>
</tr>
<tr>
<td>Correct Data Faults</td>
</tr>
<tr>
<td>Correct System Defects</td>
</tr>
<tr>
<td>Fault Prevention of Data Control</td>
</tr>
<tr>
<td>Fault Prevention of System Functions</td>
</tr>
<tr>
<td>Fault Allocation Time</td>
</tr>
</tbody>
</table>

Table 4 presents various typical procedures (middle column) for system maintainability requirements and corresponding software functions (right-hand side column) that may be specified to implement such procedures for the four types of system maintainability requirements.

Table 4: System maintainability requirements and related software functions

<table>
<thead>
<tr>
<th>System Maintainability Requirements</th>
<th>Software Procedure for System Maintainability</th>
<th>Software functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Analysability</td>
<td>Maintainability Procedure</td>
<td>System Diagnostic Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure Data Operation function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure Data Monitoring function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure Data Control function</td>
</tr>
<tr>
<td>System Analysability &amp; Changeability</td>
<td>Registered Failures</td>
<td>Failure Isolation</td>
</tr>
<tr>
<td></td>
<td>System Malfunction</td>
<td>Failure Detection</td>
</tr>
<tr>
<td>System Changeability</td>
<td></td>
<td>Correct Data Faults function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correct System Defects function</td>
</tr>
<tr>
<td>System Testability</td>
<td></td>
<td>Fault Prevention of Data Control</td>
</tr>
<tr>
<td></td>
<td>System Time</td>
<td>Fault Prevention of System Functions</td>
</tr>
<tr>
<td></td>
<td>Fault Allocation Time</td>
<td>System Time function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fault Allocation Time function</td>
</tr>
</tbody>
</table>

4.3 Software maintainability functions to be specified

The maintainability functions to be specified (and corresponding entities to be measured) are divided into external and internal maintainability function that may be allocated to software – see Table 5: External maintainability refers to the expected failures that could occur in the system, while internal maintainability refers to the expected correction of the failures occurring in the system.

Table 5: Maintainability functions allocated to software

<table>
<thead>
<tr>
<th>Type</th>
<th>Maintainability functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Maintainability</td>
<td>• Failure Data Operation function</td>
</tr>
<tr>
<td></td>
<td>• Failure Data Monitoring function</td>
</tr>
<tr>
<td></td>
<td>• Failure Data Control function</td>
</tr>
<tr>
<td></td>
<td>• System Failure Tasks function</td>
</tr>
<tr>
<td></td>
<td>• Failure Isolation function</td>
</tr>
<tr>
<td></td>
<td>• Failure Detection function</td>
</tr>
<tr>
<td>Internal Maintainability</td>
<td>• Correct Data Faults function</td>
</tr>
<tr>
<td></td>
<td>• Correct System Defects function</td>
</tr>
<tr>
<td></td>
<td>• Fault Prevention of Data Control function</td>
</tr>
<tr>
<td></td>
<td>• Fault Prevention of System Functions</td>
</tr>
<tr>
<td></td>
<td>• System Time function</td>
</tr>
<tr>
<td></td>
<td>• Fault Allocation Time function</td>
</tr>
</tbody>
</table>

4.4 Identification of the maintainability function types

This section identifies the function types and functional relationships in the software FUR for system maintainability.

Maintainability Function Type 1: Failure Procedure - Fig. 3
- The system diagnostic function sends a data group to the failure data operation, monitoring, control, and system failure tasks.
- Failure data operation, monitoring, control, and system failure tasks functions send a data group to each other.

Maintainability Function Type 2: Registered failures – Fig. 4
- Failure detection function sends/receives a data group to/from failure isolation function and vice versa.

Maintainability Function Type 3: System malfunction – Fig. 5
- Correct data faults function sends/receives a data group to/from correct system defects function, and vice versa.

Fig. 3: Maintainability failure procedure
Fig. 4: Registered failures
Fig. 5: System malfunction
Maintainability Function Type 4: System stability

- Fault prevention of data control function sends/receives a data group to/from fault prevention of system functions, and vice versa.

![Fault prevention of data control](image1)

Fault prevention of system functions

Fig. 6: System stability

Maintainability Function Type 5: System testability – Fig. 7

- The system time function sends/receives a data group to/from fault allocation time, and vice versa.
- The system time and fault allocation time function sends/receives a data group to/from system stability function, and vice versa.

![System time](image2)

Fault allocation time

System stability

Fig. 7: System testability

4.5 Identification of the functional relationships

Fig. 8 presents an overview of the relationships between the function types in the maintainability software FUR, using COSMIC for graphical representation. Specifically:

- The sub model of the Maintainability Function Type 1 can be used to specify and measure the functional size of the external maintainability for the maintainability failure procedure from the received/sent data movements from/to registered failures and system stability function – see Fig. 8.
- The sub model of the Maintainability Function Type 2 can be used to specify and measure the functional size of the external maintainability for the registered failure from the received/sent data movements from/to registered failures and system malfunctions and stability functions – see Fig. 8.
- The sub model of the Maintainability Function Type 3 can be used to specify and measure the functional size of the internal maintainability for the system malfunction Maintainability Function type from the received/sent data movements from/to registered failures and system stability functions – see Fig. 8.
- The sub model of the Maintainability Function Type 4 can be used to specify and measure the functional size of the internal maintainability for the system stability Maintainability Function type from the received/sent data movements from/to malfunction system, system testability and system failure procedure – see Fig. 8.
- The sub model of the Maintainability Function Type 5 can be used to specify and measure the functional size of the internal maintainability for the system testability entity type from the received/sent data movement from/to stability functions – see Fig. 8.

This model is referred to here as a generic model of software FUR for system maintainability.

![COSMIC reference model](image3)

Fig. 8: COSMIC reference model of maintainability requirements allocated to software
5 Discussion
This paper has introduced a procedure for specifying and measuring software requirements for the internal and external maintainability needed to address the system’s maintainability requirements.

The main contribution of this paper is our proposed Generic Model of software FUR for system maintainability. This generic model can be considered as a kind of reference model for the identification of system maintainability requirements and their allocation to software functions implementing such requirements. System requirements allocated to hardware have not been addressed in this paper. Since the structure of the generic model is based on the generic model of software adopted by the COSMIC measurement standard, the necessary information for measuring their functional size is readily available.

Specifically, the generic model of maintainability presented in this paper is based on:

- the ECSS standards for the description of the NFR for system maintainability; and
- The COSMIC measurement model of functional requirements.

The model is independent of the software type and the languages in which the software FUR will be implemented. The proposed generic maintainability model (i.e. reference model) provides:

- A specification model for each type, or all types, of maintainability requirements: for example, the requirements to be allocated to software for the maintainability failure procedures for system analyzability, the registered failures and software/system malfunction for system changeability, and for system/software stability and testability.

- A specification measurement model for each type, or all types, of maintainability requirements.

Future work includes documentation of the traceability of the elements of this generic model to the detailed elements of the ECSS standards, as well verification of this generic model to ensure full coverage of maintainability requirements. Discussions with groups of experts will be necessary to ensure its usefulness across various communities and to develop a consensus on further refinements of such a generic model which could be proposed eventually as a candidate for standardization.

REFERENCES