A generic model for the specification of software reliability requirements and measurement of their functional size

Kenza Meridji
Department of Software Engineering,
Petra University,
Amman 11196, Jordan
E-mail: kmeridji@uop.edu.jo

Khalid T. Al-Sarayreh* and Ahmad Al-Khasawneh
Departments of Software Engineering and
Computer Information System,
Hashemite University,
Zarqa 13115, Jordan
E-mail: khalidt@hu.edu.jo
E-mail: Akhasawneh@hu.edu.jo
*Corresponding author

Abstract: Currently, ISO-9126 standards for the software product quality include reliability as one of the quality characteristics for the embedded and real-time software products. According to ISO, software reliability requirements can be measured internally and externally. Furthermore, ECSS standards focus on the reliability prediction of components. It allows identifying data sources and respective methods for an application. This paper presents a method for measuring the functional size of the reliability requirements of software product. The proposed method suggests a generic measurement model to locate the functional size of the reliability requirements. This functional size is based on the concepts of the ISO-19761 standard, which allows to measure on the basis of the requirements, whether the software has already been delivered or to be built.

Keywords: reliability requirements; design of measurement method; ISO-19761; ISO-9126; ECSS standards; standard etalon.


Biographical notes: Kenza Meridji is an Assistant Professor at Petra University Amman, Jordan. She is pursuing research on the fundamental principles of software engineering and non-functional requirements. She holds a PhD degree from the École de Technologie Supérieure – Université du Québec, Montreal, Canada and a Master degree in Software Engineering from Concordia University, Canada.
Khalid T. Al-Sarayreh is an Assistant Professor of Software Engineering at Hashemite University in Jordan. He holds a PhD in Software Engineering from École de Technologie Supérieure (ÉTS) – University of Québec in Canada. He also holds a Doctoral in Computer Information Systems, MSc in Computer Engineering (Embedded Systems) and BS in Computer Science from Jordanian Universities. From 2002 to 2005, he was at the King Abdullah II Design and Development Bureau (KADDB). His research interests include: software real-time, software quality, non-functional requirements for embedded systems and HCI. 

Ahmad Al-Khasawneh is an Associate Professor of Computer Information Systems at Hashemite University. He holds a PhD of Information Technology from Australian Universities. His research spans the topics of e-business, e-commerce, e-governments, IS and software development process for emerging technologies and electronic commerce applications. He is involved in many presentations and workshops in academic area in Jordan. He has 16 years of experience in ICT field and in ICT applications. Currently, he is the Dean of the Prince Hussein Bin Abdullah II for Information Technology.

1 Introduction

Software measures are tools used to quantify several aspects of the software products, processes and projects. Software measures are used for different purposes which include assessing software quality (Kan, 2002), estimating complexity (Podgorelec and Heričko, 2007), estimating cost and effort (Idri et al., 2002; Bourque, 2003) as well as controlling and improving processes (Dawson and O’Neill, 2003). In spite of the existence of large number of software measures, the majority of them are unsuccessful (Kokol and Brest, 1997) due to a number of weaknesses: for instance, software measures are, usually, informally defined (Alikacem and Sahraoui, 2006), incomplete and/or inaccurate (Dawson and Nolan, 2003) and, hence, such measures do not produce the information required.

The ECSS European international standards ECSS-E-40-Part-1B (2003), ECSS-E-40-Part-2B (2005) and ECSS-Q-80B (2003) present software reliability as a non-functional requirement for embedded software. The ECSS-E-40-Part-1B (2003) is a cooperative effort of the European Space Agency, the national space agencies and European industry associations for the purpose of developing and maintaining common standards. The international standard ECSS-E-40-Part-1B reference addresses the management, engineering and product assurance in the space projects and applications. This part of the standard is a ‘level 2’ standard: it is derived from ISO/12207.

Recently, ECSS quality standard ECSS-Q-30-80-03A (2006) focuses on the reliability prediction of components and was devised to identify data sources and respective methods for application. In addition the ECSS-Q-30-80-03A (2006) identifies into their contents that suppliers shall define and apply measures to assure the reliability of critical software. These measures can include: the use of software design or methods that perform successfully in similar applications; failure mode analysis of the software, with the insertion of appropriate features for failure isolation and handling (see ECSS-Q-30-80-03A); defensive programming techniques, such as input verification and
consistency checks; prohibiting the use of language commands and features that are unpredictable; use of formal design language for formal proof; code branch coverage at unit testing level; full inspection of source code, witnessed or independent testing, gathering and analysis of failure statistics and removing deactivated code or showing through a combination of analysis and testing the means by which such code could be inadvertently executed are prevented, isolated or eliminated.

More recently, IEEE standard IEEE-1220 (2005) defines a set of reliability models and these models should be established to support the analysis of system effectiveness for each operational scenario. While, ISO standard ISO/IEC-24765 (2008) defines software reliability as a function of inputs and the use of the system as well as a function of the existence of faults in the software. Furthermore, ISO/IEC-9126 (2004) defines the reliability as capability of the software product to maintain a specified level of performance when used under specified conditions.

The paper will report the design measurement method to identify the functional size of the software reliability based on international standards and using COSMIC standard as an autonomous method to identify the functional size of the software reliability independently of the software languages types, which avoids the weaknesses observed in the reliability measures currently available.

The measurement scope for the paper is to identify separately all functionality allocated to software reliability as a piece of the application in the requirements for embedded and real-time software, whether it has yet to be built or it has already been delivered. Furthermore, the main contribution of this paper is the proposed ‘Generic measurement model’ of the reliability requirements. The proposed generic model is considered as a kind of the ‘reference model’ in the sense of an ‘etalon’ standard that is being used for the measurement of reliability.

This paper is organised as follows. Section 2 presents the steps for designing a measurement method. Section 3 presents an overview of the ISO-19761 international standard for measuring the functional size of software. Section 4 presents the design of a measurement method for the ‘reliability’ as non-functional requirements. Section 5 presents a discussion and analysis of the proposed measurement results. Section 6 presents functional size measurement (FSM) paradigms for the reliability. Section 7 presents the evaluation of the proposed measures for the reliability and a conclusion is presented in Section 8.

2 Design of measurement method

Four steps are recommended to carry out the design of a measurement method (Abran, 2010):

Step 1  **Purpose of the measurement objectives**: before designing a software measure.

Step 2  **Characterisation of the concept to be measured**: in order to enable the measurable perception to be defined into a measurable construct and from it the measurement method to be built, the perception to be measured must be clearly defined.
Step 3  
*Design of the metadata:* the set of descriptions selected to signify the software or a software piece, together with the set of their relationships, comprise the meta-model of the software to which the proposed measurement method will be applied. This meta-model should be illustrated broadly: it should not be specific to particular software.

Step 4  
*Numerical assignment rules:* consists in significant measurable perceptions and pragmatic relational set, to complete the design of a measurement method, a numerical relational set and a homomorphism between these two relational sets must be defined.

3 A generic view of ISO-19761

In the collection of ISO standards, it is specified in the ISO-14143-1 (ISO/IEC-14143-1, 2008) that a FSM method must measure the software functional user requirements (FUR). In addition, ISO-19761 – COSMIC (ISO/IEC-19761, 2011) proposes a generic model of software-FUR that clarifies the boundary between hardware and software. Figure 1 illustrates the generic flow of data from a functional perspective from hardware to software. From this generic model of software functional requirements in Figure 1 the followings can be observed:

- Software is bounded by hardware. In the so-called ‘front-end’ direction (i.e., left-hand side in Figure 1), software used by a human user 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., right-hand side of Figure 1), software is bounded by persistent storage hardware like a hard disk and RAM and ROM memory.

- The software functionality is embedded within the functional flows of data groups. Such data flows can be characterised 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 the 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 the ‘I/O hardware’. For business application software, the abstraction commonly assumes that the users are 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.
4 Design of measurement method of reliability requirements

Reliability requirements views and concepts stated by the European ECSS, ISO-9126 and IEEE-1220 standards. In addition to design measures steps stated by Abran (2010) in Section 2 of this paper and (ISO/IEC-19761, 2011) standard in Section 3 of this paper, the design measurement method for the reliability as non-functional requirement merge between the suggesting reliability design in the basis of ISO standards through the design measures steps and ISO-19761 standard to measure the functional size of the defined reliability requirements.

4.1 Step 1: determination of the measurement objectives

The measurement objective includes the following:

- the measurements objective: is to measure the functional size of the reliability requirements as defined in ISO-9126, ECSS and using the ISO-19761 standard as a measurement method
- the measurement point of view: software perspective
- the intended uses of the measurement results: throughout the software life cycle where the functional size of the reliability for a software product, whether it has yet to be built or it has already been delivered.

4.2 Step 2: characterisation of the concept to be measured

The characterisation of the concepts to be measured can include the following:

- The definition of the concept to be measured: is the functional size of the reliability; where reliability measurements can be internal or external. Although the ECSS standard deals with reliability specific to software-embedded system developed as
part of a space project, the proposed measurement method is to be applicable for
non-embedded software reliability. The reliability requirements are defined as the
probability that software will not cause the failure of a system for a specified time
under specified conditions. The reliability is a function of the inputs and use of the
system as well as a function of the existence of faults in the software. The inputs to
the system determine whether existing faults, if any, are encountered. ISO-9126
define the reliability as the capability of the software product to maintain a specified
level of performance when used under specified conditions.

- In ISO-9126 software quality product, there are two types of reliability requirements
  measures (ISO/IEC-9126, 2004):
    a  External reliability measures: should be able to measure attributes related to the
        behaviours of the system of which the software is part during execution testing
        to indicate the extent of reliability of the software in that system during
        operation. Systems and software are not distinguished from each other in most
        cases and including the following basic measurements:
        1  maturity measures such attributes for the software freedom of failures
            caused by faults existing in the software itself
        2  fault tolerance measure related to the software capability of maintaining a
            specified performance level in cases of operation faults or infringement of
            its specified interface
        3  recoverability measure such attributes for the software with system being
            able to re-establish its adequate level of performance and recover the data
            directly affected in the case of failure.
    b  Internal reliability measures: are used for predicting if the software product in
        question will satisfy prescribed reliability needs, during the development of the
        software product (ISO/IEC-9126, 2004):
        1  internal maturity measures indicate a set of attributes for assessing the
            maturity of the software
        2  internal fault tolerance measures indicate a set of attributes for assessing the
            software products capability in maintaining a desired performance level in
            case of operational faults or infringement of its specified interface
        3  internal recoverability measures indicate a set of attributes for assessing the
            software product’s capability to re-establish an adequate level of
            performance and recover the data directly affected in case of failure.

- The entities to be measured:
  1  External reliability measurements includes the following:
     a  *maturity*: it will include the following: (estimated fault density; failure
         resolution and test maturity)
     b  *fault tolerance*: it will include the following: (breakdown avoidance and
         incorrect operation avoidance)
     c  *recoverability*: it will include the following: (availability; breakdown time
         and recovery time).
Internal reliability measurements includes the following:

a. **maturity**: it will include the following: (fault removal and test coverage)
b. **fault tolerance**: it will include the failure avoidance
c. **recoverability**: it will include the following: (restart ability and restorability).

### 4.3 Step 3: design or selection of the meta model

In this step we identification the model of the entity types and the relationship among entities for software reliability requirements of the software product.

- The generic model of FSMs of software maturity requirements based on ISO-9126 aligned with ISO-19761 can be defined as follows – see Figure 2.

1. **entity type 1 (external measurement for maturity)**
   a. entity name: estimated fault density
   b. the input of entity type 1 is predicted faults in software product
   c. the output of entity type 1 is actual detected faults
   d. entity type 1 measures the functional size of estimated fault density
   e. entity relationship: many-many detected faults on the system

2. **entity type 2 (internal measurement for maturity)**
   a. entity name: fault removal
   b. the input of entity type 2 is correcting faults
   c. the output of entity type 2 is actual detected faults
   d. entity type 2 measures the functional size of fault removal
   e. entity relationship: many-many faults correction on the system

3. **entity type 3 (external measurement for maturity)**
   a. entity name: failure resolution
   b. the input of entity type 3 is correcting failures
   c. the output of entity type 3 is actual detected failures
   d. entity type 3 measured the functional size of failure resolution
   e. entity relationship: many-many failure correction on the system

4. **entity type 4 (internal measurement for maturity)**
   a. entity name: test coverage
   b. the input of entity type 4 is the number of test cases to be performed to cover requirements
   c. the output of entity type 4 is actual number of test cases performed to cover requirements
   d. entity type 4 Measures the functional size of test coverage
   e. entity relationship: many-many test coverage numbers on the system
5 entity type 5 (external measurement for maturity)
   a entity name: test maturity
   b the input of entity type 5 is the number of test coverage performed to cover requirements
   c the output of entity type 5 is success number of test coverage performed to cover requirements
   d entity type 5 measures the functional size of test maturity
   e entity relationship: many-many coverage tests on the system.

Figure 2  Generic of FSM model of software product maturity

In the following design, the evaluation model of the maturity is composed of five entity types described as follows – see Figure 3:

- **entity type 1**: can be used to measure the functional size of external software reliability throughout predicting future faults using a reliability growth estimation model for one functional process – see Figure 3
- **entity type 2**: can be used to measure the functional size of internal software reliability throughout faults detected during a defined trial period for one functional process – see Figure 3
- **entity type 3**: can be used to measure the functional size of external software reliability throughout failures detecting during a defined trial period for one functional process – see Figure 3
- **entity type 4**: can be used to measure the functional size of internal software reliability throughout test cases performed during testing for one functional process – see Figure 3
• entity type 3: can be used to measure the functional size of external software reliability throughout test cases performed during testing that are required to obtain adequate test coverage for one functional process – see Figure 3.

Figure 3  Quality evaluation of maturity for software product

- The generic model of the FSMs of software fault tolerance requirements is built based on ISO-9126 aligned with ISO-19761. It can be defined from the following – see Figure 4:

1 entity type 6 (external measurement for fault tolerance)
   a entity name: break down avoidance
   b the input of entity type 6 is the number of software failures
   c the output of entity type 6 is number of breakdown occurrence
   d entity type 6 measures the functional size of the break down avoidance
   e entity relationship: many-many failures and breakdown on the system

2 entity type 7 (internal measurement for fault tolerance)
   a entity name: failure avoidance
   b the input of entity type 7 is the number of executed test cases of fault pattern
   c the output of entity type 7 is the number of avoided critical and serious failures
   d entity type 7 measures the functional size of the failure avoidance
   e entity relationship: many-many failures and fault pattern on the system
entity type 8 (external measurement for fault tolerance)

a. entity name: incorrect operation avoidance
b. the input of entity type 8 is the number of executed test cases of incorrect operation patterns
c. the output of entity type 8 is number of avoided critical and serious failures
d. entity type 8 measures the functional size of the incorrect operation avoidance
e. entity relationship: many-many failures and fault pattern on the system.

**Figure 4** Generic FSM model of software fault tolerance

In the following design the evaluation model of the fault tolerance is composed of three entity types described as follows – see Figure 5:

- **entity type 6**: can be used to measure the functional size of external software reliability throughout breakdown occurrence with respect to failures for one functional process – see Figure 5
- **entity type 7**: can be used to measure the functional size of internal software reliability throughout control fault pattern to avoid critical and serious failures for one functional process – see Figure 5
- **entity type 8**: can be used to measure the functional size of external software reliability throughout test cases of incorrect operations which were avoided to cause critical and serious failures for one functional process – see Figure 5.
The generic model of the FSMs of software recoverability requirements is built based on ISO-9126 concepts and views aligned with ISO-19761. It can be defined from the following – see Figure 6:

1. Entity type 9 (external measurement for recoverability)
   a. Entity name: availability
   b. The input of entity type 9 is the software operation time
   c. The output of entity type 9 is time to repair
   d. Entity type 9 measures the functional size of the availability
   e. Entity relationship: many-many time operations on the system

2. Entity type 10 (external measurement for recoverability)
   a. Entity name: breakdown time
   b. The input of entity type 10 is the software down time
   c. The output of entity type 10 is observed breakdown
   d. Entity type 10 measures the functional size of the breakdown time
   e. Entity relationship: many-many down time in operations on the system

3. Entity type 11 (external measurement for recoverability)
   a. Entity name: recovery time
   b. The input of entity type 11 is time to recovery downed software system
   c. The output of entity type 11 is observed software recovery time
   d. Entity type 11 measures the functional size of the breakdown time
In the following design, the evaluation model of the recoverability is composed of five entity types described as follows – see Figure 7:

- **entity type 9**: can be used to measure the functional size of external software reliability throughout the repair time period when the system was unavailable during the trial for one functional process – see Figure 7
• **entity type 10:** can be used to measure the functional size of external software reliability throughout the down time when the system was unavailable during the trial for one functional process – see Figure 7

• **entity type 11:** can be used to measure the functional size of external software reliability throughout the recovery time when the system brought down during the trial for one functional process – see Figure 7

• **entity type 12:** can be used to measure the functional size of internal software reliability throughout the restarts time when the system was down during the trial for one functional process – see Figure 7

• **entity type 13:** can be used to measure the functional size of internal software reliability throughout the restoration time of the system during the trial for one functional process – see Figure 7.

Figure 7  Quality evaluation of recoverability for software product

4.4 Step 4: numerical assignment rules

The numerical assignment rules can be described through a descriptive text (a practitioner’s description) or through mathematical expressions (a formal theoretical viewpoint).

In this step, the basis for these numerical assignment rules are the proposed meta-models in Figures 2, 4 and 6.
4.4.1 ISO-19761 measurement procedure

The FSM procedures have been developed for applying the ISO-19761 measurement method to particular methods of software production. A subset of these measurement procedures is focused on the measurement of the functional size of applications from their conceptual models, allowing the generation of indicators in early stages of the development cycle of a software product.

4.4.2 Identification of data groups

The data sources and destinations of software reliability requirements for the software product quality can be defined in Table 1:

<table>
<thead>
<tr>
<th>Categories</th>
<th>Data sources</th>
<th>Data destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity</td>
<td>• The predicted faults in software product.</td>
<td>• Estimated fault density</td>
</tr>
<tr>
<td></td>
<td>• Actual detected faults.</td>
<td>• Fault removal</td>
</tr>
<tr>
<td></td>
<td>• The correcting faults.</td>
<td>• Failure resolution</td>
</tr>
<tr>
<td></td>
<td>• The output of the entity type 2 is actual detected faults.</td>
<td>• Test coverage</td>
</tr>
<tr>
<td></td>
<td>• The correcting failures.</td>
<td>• Test maturity</td>
</tr>
<tr>
<td></td>
<td>• Actual detected failures.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test cases to be performed to cover requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Actual number of test cases performed to cover requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test coverage performed to cover requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Success number of test coverage performed to cover requirements.</td>
<td></td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>• Number of software failures.</td>
<td>• Break down avoidance</td>
</tr>
<tr>
<td></td>
<td>• Number of breakdown occurrence.</td>
<td>• Failure avoidance</td>
</tr>
<tr>
<td></td>
<td>• The number of executed test cases of fault pattern.</td>
<td>• Incorrect operation avoidance.</td>
</tr>
<tr>
<td></td>
<td>• Number of avoided critical and serious failures.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The number of executed test cases of incorrect operation pattern.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Number of avoided critical and serious failures.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1

Data sources and destinations for reliability requirements (continued)

<table>
<thead>
<tr>
<th>Categories</th>
<th>Data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recoverability</td>
<td>• The software operation time.</td>
</tr>
<tr>
<td></td>
<td>• Time to repair.</td>
</tr>
<tr>
<td></td>
<td>• The software down time.</td>
</tr>
<tr>
<td></td>
<td>• Observed breakdown.</td>
</tr>
<tr>
<td></td>
<td>• Time to recovery downed software system.</td>
</tr>
<tr>
<td></td>
<td>• Observed software recovery time.</td>
</tr>
<tr>
<td></td>
<td>• Restart during system.</td>
</tr>
<tr>
<td></td>
<td>• Restart which met to required time during testing.</td>
</tr>
<tr>
<td></td>
<td>• Testing restorability and Successful restoration</td>
</tr>
<tr>
<td></td>
<td>• Availability</td>
</tr>
<tr>
<td></td>
<td>• Breakdown time</td>
</tr>
<tr>
<td></td>
<td>• Recovery time</td>
</tr>
<tr>
<td></td>
<td>• Restartability</td>
</tr>
<tr>
<td></td>
<td>• Restorability</td>
</tr>
</tbody>
</table>

#### 4.4.3 Identification of the functional processes and data movements

- The FSM of maturity requirements from data destinations to data sources with respect to the proposed generic meta-model in Figure 2 for one data movement and one data group as follows – see Table 2.
  
  a. the functional size of the maturity (externally) for one process
     
     = Σ data movement (data group) (estimated faults density + failure resolution + test maturity)
     
     = 6 + 7 + 5 CFP through predicting faults and failures and detecting actual faults and failures
     
     = 18 CFP

  b. the functional size of the maturity (internally) for one process
     
     = Σ data movement (data group) for fault removal and test coverage
     
     = 7 + 7 CFP through predicting faults and failures and detecting actual faults and failures
     
     = 14 CFP

  c. the total functional size for the maturity [for all the above cases for one process]
     
     = the functional size of the maturity (externally and internally)
     
     = the functional size of the maturity (externally) + the functional size of the maturity (internally)
     
     = 18 + 14 = > 32 CFP

  d. the total functional size of the maturity [for the all functional processes]
     
     = N × Σ functional size of the maturity (internally) + N × Σ the functional size of the maturity (externally)

N: number of functional processes for the maturity.
Table 2  Maturity FSM with respect to generic meta model

<table>
<thead>
<tr>
<th>Reliability destinations</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type 1: estimated fault density</td>
<td>• Entry the predicted number of faults in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the predicted number of faults in software product throughout predicts future faults using a reliability growth estimation model.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the predicted number of faults in software product throughout predicts future faults using a reliability growth estimation model.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the actually detected faults throughout faults detected during a defined trial period.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the actually detected faults throughout faults detected during a defined trial period.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with actually detected faults from predicted faults and defined trial period.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Entity type 2: fault removal</td>
<td>• Entry the correcting faults in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read correcting faults in software product throughout predict future faults using a reliability growth estimation model and faults detected during a defined trial period.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write correcting faults in software product throughout predict future faults using a reliability growth estimation model and faults detected during a defined trial period.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read and compare the actually detected faults throughout faults detected during a defined trial period.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the actually detected faults throughout faults detected during a defined trial period.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with actually detected faults from predicted faults and defined trial period.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Entity type 3: failure resolution</td>
<td>• Exit from estimated faults and faults removal.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Entry the correcting failures in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read correcting failures in software product throughout failures detected during a defined trial period.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write correcting failures in software product throughout failures detected during a defined trial period.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read and compare the actually detected failures throughout failures detected during a defined trial period.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the actually detected failures throughout failures detected during a defined trial period.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with actually detected failures from predicted faults and defined trial period.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit from failure resolution</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Total functional size = 32 CFP.
A generic model for the specification of software reliability requirements

Table 2 Maturity FSM with respect to generic meta model (continued)

<table>
<thead>
<tr>
<th>Reliability destinations</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type 4: test coverage</td>
<td>• Entry the number of test cases to be performed to cover requirements in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the number of test cases to be performed to cover requirements in software product throughout test cases that be required to obtain adequate test coverage.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the number of test cases to be performed to cover requirements in software product throughout test cases that be required to obtain adequate test coverage.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read and compare the actually performed test cases during testing throughout test cases performed during testing that are required to obtain adequate test coverage.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the actually performed test cases during testing throughout test cases performed during testing that are required to obtain adequate test coverage.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with actually performed number of test cases.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit from test coverage</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Entity type 5: test maturity</td>
<td>• Entry the number of test coverage to be performed to cover requirements in software product to achieve test maturity.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the number of test coverage to be performed to cover requirements in software product to achieve test maturity.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the number of test coverage to be performed to cover requirements in software product to achieve test maturity.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with the number of test coverage that achieves test maturity.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit from test maturity</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Total functional size = 32 CFP.

- The FSM of fault tolerance requirements from data destinations to data sources with respect to the proposed generic meta-model in Figure 4 for one data movement and one data group – see Table 3.
  a the functional size of the fault tolerance (externally) for one process
    = Σ data movement (data group) (breakdown avoidance + incorrect operation avoidance)
    = 6 + 6 CFP during breakdown occurrence with respect to failures
  b the functional size of the fault tolerance (internally) for one process
    = Σ data movement (data group) for failure avoidance
    = 6 CFP during the reliability process under control fault pattern to avoid critical and serious failures
c. the total functional size for the fault tolerance [for all the above cases for one process]

\[= \text{the functional size of the fault tolerance (internally and externally)}\]
\[= \text{the functional size of the fault tolerance (externally)} + \text{the functional size of the fault tolerance (internally)}\]
\[= 12 + 6 \geq 18 \text{ CFP during the reliability process manage breakdown occurrences with respect to failures and control fault pattern to avoid critical and serious failures}\]

d. the total functional size of the fault tolerance [for all functional processes]

\[= N \times \sum \text{the functional size of the fault tolerance (internally)} + N \times \sum \text{the functional size of the fault tolerance (externally)}\]

N: number of functional processes for the fault tolerance.

<table>
<thead>
<tr>
<th>Fault tolerance destination</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type 6: break down avoidance</td>
<td>• Entry the number of failures in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the failures in software product throughout breakdown occurrence with respect to failures.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the failures in software product throughout breakdown occurrence with respect to failures.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the breakdown occurrence in software product with respect to failures.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the breakdown occurrence in software product with respect to failures.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with breakdown avoidance.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Entity type 7: failure avoidance</td>
<td>• Entry the executed number of test cases of fault pattern in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the executed test cases of fault pattern in software product throughout control fault pattern to avoid critical and serious failures.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the executed test cases of fault pattern in software product throughout control fault pattern to avoid critical and serious failures.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the avoided critical and serious failures in software product.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the avoided critical and serious failures in software product.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with failure avoidance.</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Total functional size = 18 CFP.
### Table 3  
Fault tolerance FSM with respect to generic meta model (continued)

<table>
<thead>
<tr>
<th>Fault tolerance destination</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type 8: incorrect operation avoidance</td>
<td>Entry the executed number of test cases of incorrect operation in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Read the executed test cases of incorrect operation in software product throughout test cases of incorrect operations which were avoided to cause critical and serious failures.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Write the executed test cases of incorrect operation in software product throughout test cases of incorrect operations which were avoided to cause critical and serious failures.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Read the avoided critical and serious failures in software product.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Write the avoided critical and serious failures in software product.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Exit with incorrect operation avoidance.</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Total functional size = 18 CFP.

- The FSM of recoverability requirements from data destinations to data sources with respect to the proposed generic meta-model in Figure 6 for one data movement and one data group – see Table 4.
  
  a the functional size of the recoverability (externally) for one process
  
  \[ \Sigma \text{data movement (data group) (availability + breakdown time + recoverability time)} \]
  
  \[ = 6 + 6 + 6 \text{ CFP during reliability process measured repair, down and recovery time when the system is unavailable.} \]
  
  \[ = 18 \text{ CFP}. \]

  b the functional size of the recoverability (internally) for one process
  
  \[ \Sigma \text{data movement (data group) for (restartability and restorability)} \]
  
  \[ = 6 + 6 \text{ CFP during the reliability process measured restatability and restorability time of the downed system} \]
  
  \[ = 12 \text{ CFP}. \]

  c the total functional size for the stability [for all the above cases for one process]
  
  \[ = \text{the functional size of the recoverability (internally and externally)} \]
  
  \[ = \text{the functional size of the recoverability (externally)} + \text{the functional size of the recoverability (internally)} \]
  
  \[ = 18 + 12 \geq 30 \text{ CFP} \]

  d the total functional size of the recoverability [all functional processes]
  
  \[ = N \times \Sigma \text{functional size of the recoverability (internally)} + N \times \Sigma \text{the functional size of the recoverability (externally)} \]

N: number of functional processes for the recoverability.
<table>
<thead>
<tr>
<th>Recoverability destination</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type 9: availability</td>
<td>• Entry the operation time in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read the operation time in software product throughout the repair time period when the system was unavailable during the trial.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write the operation time in software product throughout the repair time period when the system was unavailable during the trial.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Read time to repair in software product.</td>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Write time to repair in software product.</td>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• Exit with software availability.</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

| Entity type 10: breakdown time | • Entry the software down time in software product. | E   | 1   |
|                                | • Read the software down time in software product throughout the down time when the system was unavailable during the trial. | R   | 1   |
|                                | • Write the software down time in software product throughout the down time when the system was unavailable during the trial. | W   | 1   |
|                                | • Read observed breakdowns in software product. | R   | 1   |
|                                | • Write observed breakdowns in software product. | W   | 1   |
|                                | • Exit with software breakdown time.            | X   | 1   |

| Entity type 11: recovery time | • Entry the time to recovery downed software system in software product. | E   | 1   |
|                              | • Read the time to recovery downed software system in software product throughout the recovery time when the system brought down during the trial. | R   | 1   |
|                              | • Write the time to recovery downed software system in software product throughout the recovery time when the system brought down during the trial. | W   | 1   |
|                              | • Read the observed software recovery time in software product. | R   | 1   |
|                              | • Write the observed software recovery time in software product. | W   | 1   |
|                              | • Exit with software recovery time.             | X   | 1   |

| Entity type 12: restartability | • Entry the number of restarts during testing in software product. | E   | 1   |
|                                | • Read the restarts during testing in software product throughout the restarts time when the system was down during the trial. | R   | 1   |

Note: Total functional size = 30 CFP.
Table 4  Recoverability FSM with respect to generic meta model (continued)

<table>
<thead>
<tr>
<th>Recoverability destination</th>
<th>Data movement description (data sources)</th>
<th>DMT</th>
<th>CFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Write the restarts during testing in software product throughout the restarts time when the system was down during the trial.</td>
<td>W</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Read the number of restarts which met to required time during testing in software product.</td>
<td>R</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Write the number of restarts which met to required time during testing in software product.</td>
<td>W</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Exit with software restartability.</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Entity type 13: restorability</td>
<td>Entry the number of testing restoration in software product.</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>• Read the testing restoration in software product throughout the restoration time of the system during the trial.</td>
<td>R</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Write the testing restoration in software product throughout the restoration time of the system during the trial.</td>
<td>W</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Read the number of successfully restoration in software product during the trial.</td>
<td>R</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Write the number of successfully restoration in software product during the trial.</td>
<td>W</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>• Exit with software restorability.</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Note: Total functional size = 30 CFP.

5 Discussion and analysis

With regards to the proposed generic models, the reliability functional size (internally and externally) is equal to 80 CFP for one functional process throughout predicting faults and failures, detecting the actual faults and failures, breakdown occurrence for faults and failures and recovery time when the system is unavailable or down during the trial.

The reliability software can be measured as follows:

• the functional size of the reliability (externally) for one process
  \[ = \sum \text{data movement for (maturity + fault tolerance + recoverability)} \]

• the functional size of the reliability (internally) for one process
  \[ = \sum \text{data movement for (maturity + fault tolerance + recoverability).} \]

• the total functional size for the reliability [for all the above cases for one process]
  \[ = \text{the functional size of the reliability (internally and externally)} \]

\[ = \text{the functional size of the reliability (externally) + the functional size of the reliability (internally)} \]
the total functional size of the reliability [for the all functional processes]

\[ N \times \sum \text{functional size of the reliability (internally)} + N \times \sum \text{the functional size of the reliability (externally)}. \]

N: number of functional processes for the reliability.

6 Functional measurement size examples for the generic reliability models

Example 1: Find the functional measurement size for the reliability, assuming the following specifications are given:

- the functional measurement size required for three functional processes
- the required functional size needed for recoverability process.

Solution: With regards to the ‘generic models’ of the reliability (Figure 6) and (Table 4)

1. the total functional size for the recoverability [for one functional process]
   \[ = \text{the functional size of the recoverability (internally and externally)} \]
   \[ = \text{the functional size of the recoverability (externally)} + \text{the functional size of the recoverability (internally)} \]
   \[ = 18 + 12 \]
   \[ = 30 \text{ CFP} \]

2. the total functional size of the recoverability [for three functional processes]
   \[ = N \times \sum \text{functional size of the testability (internally)} + N \times \sum \text{the functional size of the testability (externally)} \]
   N = 3 according to example.
   \[ = 3 \times 18 + 3 \times 12 \]
   \[ = 54 + 36 \text{ CFP} \]
   \[ = 90 \text{ CFP}. \]

Example 2: Find the functional measurement size for the reliability, assuming the following specifications are given:

- the functional measurement size required for five functional processes
- the required functional size needed for recoverability and maturity process
- internal measurement is needed.

Solution: With regards to the ‘generic models’ of the reliability (Figure 2) and (Table 4)

1. the functional size of the reliability (internally) for one process.
   \[ = \sum \text{data movement (data group) for (maturity + recoverability)} \]
   \[ = 14 + 12 \]
The total functional size of the reliability [for the all functional processes]

\[ = N \times \sum \text{functional size of the reliability (internally)} + N \times \sum \text{the functional size of the reliability (externally)} \]

\[ N: \text{number of functional processes for the reliability} = 5. \]

\[ = 5 \times 14 + 5 \times 12 + 0 + 0 \]

\[ = 70 + 60 + 0 \]

\[ = 130 \text{ CFP.} \]

7 Evaluation of the proposed models

With respect to the international vocabulary of basic and general terms in metrology a standard etalon is: “A material for measurement, measuring instrument, reference material or measuring system intended to define, realizes, conserve or reproduce a unit or one or more values of a quantity to serve as a reference” (Abran, 2010).

With regards of method of design etalon and aligned with ISO-14143, the proposed generic models in this paper are considered as a reference model for measuring the functional size of the reliability requirements for the following reasons:

- the definitions and the interpretation of the reliability requirements are taken from the definitions of reliability requirements in the European international standard series (ECSS), and ISO-9126; this could be considered as a primary material measures to the proposed generic model of reliability
- a design measurement method used in this paper includes four steps, regarding to (Abran, 2010) these steps ensure that measurements are performed in a consistent manner, a baseline is established as a primary reference
- using ISO-19761 standard as international method to identify the FSM of the reliability requirements as well as to provide measurement units
- the calibration between step 1, step 2 and ISO-19761 standard procedure identify the proposed generic measurement model of reliability requirements; this is equivalent to a measurement instrument or the reference material with respect of software etalon.

The proposed ‘generic measurements models’ of the reliability requirements or reference models with respect to etalon standards offer:

- the reliability can be measured internally and externally based on the number of functional processes
- the proposed generic models provide a measurement for each type or all types of the reliability requirements, for example the measurement for reliability of maturity, fault tolerance and recoverability
- the interrelations between the internal and external measurements are defined, for example each process between the internal and external reliability measurements
the FSMs for the software reliability requirements are defined for all functional processes (internally and externally)

- through the proposed generic models of the reliability (reference models) could trace the FSM of the reliability for internal or external measurements
- the proposed generic models provide a control and stability of the measurement results
- the generic models provide a measurement unit
- the proposed measurements models of the software reliability in this paper are identified separately
- all the functionality allocated to software reliability as a piece of application in the requirements for embedded and real time software, whether it has yet to be built or it has already been delivered
- the clarification paradigms in this paper are not enough to explain the hundreds of possibilities of the functional size of the reliability interactions among the system
- the automated computations and equations in this paper for reliability not built yet.

8 Conclusions

This paper introduced a new design measurement method that measures the internal and external attributes for the reliability and as a non-functional requirement, as well as, the proposition of a generic functional size models for the reliability requirement using ISO-19761 standard independently of the software type or languages.

Moreover, the design of the measurement method specifies the strategy of the measurement rules to perform the mapping between the concepts of ISO-19761 and the concepts of the suggesting design of the generic reliability meta models and rules to identify the data movements and to perform the measurement.

It is important to remark that the design measurement procedure for reliability requirements for the embedded software has been developed to apply the ISO-19761 measurement method to the reliability requirements in order to obtain the functional size of the reliability as a separate piece of software in the early stages of the software development process.

Moreover, the clarification examples explained how to use the proposed generic models or the reference models at the sense of standard etalon.

In this paper, the proposed generic models are described among the method of design Etalon by comparing the generic models for measuring the functional size for reliability requirements with the Etalon contents and methodology. Moreover, the evaluation of the proposed generic models is stated after its adaptation with standard etalon.

The future work will concentrate on the improvements of the generic models based on the advantages and the limitations stated in this paper sections. As well as the future application of these generic models in the industry.
A generic model for the specification of software reliability requirements

References


