Original Research Paper

Towards a Reference Model of Software Resources Quality

[1]Khaled Almakadmeh, [2]Kenza Meridji and [1]Khalid T. Al-Sarayreh

[1]Department of Software Engineering, The Hashemite University, Zarqa, Jordan
[2]Department of Software Engineering, University of Petra, Amman, Jordan

Article history
Received: 02-12-2017
Revised: 25-01-2018
Accepted: 02-02-2018

Corresponding Author:
Khaled Almakadmeh
Department of Software Engineering, The Hashemite University, Zarqa, Jordan
Email: khaled.almakadmeh@hu.edu.jo

Abstract: International standards for software product quality classify software resources as a non-functional requirement for software product. Resources requirements based-standards describe required resources requirements related to software and hardware requirements, in which hardware resources requirements specify requirements relevant to hardware environment in which the software will operate. Whereas, software resources requirements specify sizing and timing requirements required by software product. This paper propose a reference model to identify and measure resources requirements of software product quality based on ISO international standards. The proposed reference model is experimented to present its applicability using the software specifications of an ATM machine to identify and measure the functional size of resources requirements independently from development technology. This measure take place at an early phase of the software development life cycle and used by software project managers as one of the primary inputs for the effort estimation process of software products.

Keywords: Software Resources, Software Product Quality, Measurement, ISO19761, ISO25010

Introduction

Software products face a traditional challenge to obtain required resources to accomplish needed functionality for its users. Software resources requirements are considered as primary requirements for several computing environments (Alur and Weiss, 2008). For example, simple business transaction might struggle to utilize memory resources to be fully executed. The selection of such resource utilities has lead to the proposal of algorithms based on predefined metrics related to customer satisfaction or even performance metrics. For example, the quantity of advertisement pamphlet without affecting their quality requirements.

Software development organizations require software engineers to identify, measure and implement all requirements of software products. In particular, software engineers are responsible to identify all resources requirements such as I/O devices, memory resources utilization and expansion transmission of software resources. Developing such software products within time and budget is the primary challenge for many software development organizations.

A few research is found in the literature on the identification and measurement of software requirements based on international standards (Abran *et al*., 2013; Al-Sarayreh *et al*., 2013a; Meridji *et al*., 2013; Al-Sarayreh *et al*., 2013b; 2012; 2014; Al-Sarayreh and Abran, 2010). Most of research studies target the identification and measurement of software requirements at late phase of the software development life cycle (Khatter and Kalia, 2013).

Further, several software products failed because software engineers have poorly identified and measured software requirements (Al-Sarayreh *et al*., 2013a). This comes from the fact that software engineers have no reference model that is built based on an international standard to justify the need for such software requirements.

Software resources requirements illustrate the need of software components for resources from its environment in order to execute its tasks. In addition, it express the restrictions of software resources associated with workstation resources, such as microprocessor chip load and upper limit memory volume.

Software resources requirements are primary requirements in software development cycle, in which they guarantee software suitability and availability of resources for every task executed in such software products (Khatter and Kalia, 2013).

The ISO25010 international standards series (i.e., systems and software engineering - systems and Software Quality Requirements and Evaluation (SQuaRE) standards) express software resources requirements as the ability of software product to utilize correct quantities and varieties of software resources to execute certain tasks for its users under pre-defined conditions (ISO25010, 2011).

In spite of the existence of several measures for software resources requirements, most of these measures are still subjective and ineffective. For example, such these measures are defined informally, either used in an imperfect context or using poorly defined procedures. Therefore, such measures cannot be justified by software engineers for their project managers in order to use them in the effort estimation process of software products.

The motivation of this research paper is to help software development organizations and in particular software project managers and technical leaders to build more accurate effort estimation models, by improving one of primary inputs (i.e., measurement of software resources requirements) for the effort estimation process. This improvement will improve planning, management and development of software at different phases of the software development life cycle. Further, the measurement results of the proposed reference model can be used for software benchmarking purposes conducted by specialized groups such as International Software Benchmarking Group (ISBSG).

The contribution of this paper is a new reference model to identify and measure software resources requirements based on international ISO standards ISO19761 and ISO14143-1. This reference model measures functional size of software resources requirements allocated to software independently from development technology used to develop the software product.

This paper is organized as follows: Section 2 presents the literature review and section 3 presents the international standard for software functional measurement - ISO 19761: COSMIC. Section 4 presents the design of the reference model for software resources requirements. Then, section 5 presents verification of the applicability for the proposed reference model using the software specifications of an ATM machine. Finally, section 6 presents conclusions and future work directions.

Literature Review

Khatter and Kalia (2013) studied several software development approaches to analyze the impact of software non-functional requirements on requirements evolution and on software quality during the software development life cycle. This study focused on views and representation of non-functional requirements and on the description of non-functional requirements in different software development approaches. It reported that there is a strong need for an accurate modeling and quantification of software non-functional requirements in order to produce high quality software.

Several research studies are presented in the literature targeted the measurement and/or approximation of software resources. For instance, Perez *et al.* (2015) proposed an algorithm that applies linear regression and maximum probability measures to calculate an approximation of the required software resources. The algorithm is experimented using real application datasets. For each CPU request, the proposed algorithm inputs response time and resource queue measures and yields the value of CPU consumption.

Arnold *et al.* (2014) proposed a generic architecture to calculate resources requirements such as computing, network usage and storage for a software application in a cloud computing environment. The software application is transformed and called a "workload" using a declarative workload definition language. It is then deployed in the cloud infrastructure using a workload orchestration and optimization layer. A case study is conducted on a real application to illustrate coordination and functionalities optimization in IBM connections.

Eklov *et al.* (2014) proposed a software profiling method to report the impact of scalability bottleneck on the scalability of multi-threaded program. These reports are aimed to help a software engineer to analyze resources requirements and to improve the architecture of such multi-threaded program.

Fotrousi *et al.* (2014) proposed an approach to identify the relationships between levels of software quality and their impact on quality attributes and software stakeholders. This approach is aimed to help software engineers to identify the quality of "good" software from the perspective of software stakeholders. The stakeholders are given a prototype of software product to determine its quality and record their subjective feedback.

Arfeen *et al.* (2011) proposed a framework for network resource allocation in cloud computing environment. Several strategies for network resource allocation are evaluated for possible application. Their work focused on optimizing such network allocation strategies and on network awareness.

Kocsis and Ekler (2012) defined "advertising type" for advertising websites. They collected websites usage statistics and users' behavior using a web analytics program to calculate total system startup. This program collected various information about users from advertising websites including location, number of visits and time spent in each website visit on a daily basis.

Shilun *et al.* (2011) proposed a resource-ability design to identify the resources requirements of an equipment. The proposed design allows for defining an equipment attributes in order to guarantee the capacity of resources conservation and environment responsiveness. This study reported that consumption, function, structure and environmental impacts are qualitative requirements for equipment resources.

Wang *et al.* (2008) proposed an approach for requirements analysis and proposed a workflow model for resource-constrained business processes. This study explained that business processes are typically constrained by insufficient software resources and such shortage of resources might cause a conflict and therefore delay the achievement of higher-level business goals. The proposed approach is aimed to meet such business goals by managing the required resources as well as making available facilities for monitoring and controlling capabilities.

Jia *et al.* (2012) analyzed a technique for maintenance task allocation. They proposed a system for computer-aided decision analysis to improve the efficiency of maintenance-task analysis and maintenance-support analysis of software resources requirements.

Liu *et al.* (2011) reported that most software resources on the internet do not provide justifiable quality indicators. Therefore, they proposed an approach that collect comments on software resources automatically. This approach is aimed to provide software engineers with justifiable quality indicators; such indicators typically help software engineers in the selection process of software resources and reusability.

Li *et al.* (2014) studied the impact of two consolidating n-tier web applications in a cloud environment; they measured the CPU utilization and performance of two consolidated n-tier web applications, in which these two systems deploy two different soft resource allocation strategies. The study reported that a web application with more software resources has overused the CPU with a percentage of eight percent than the other consolidated application. They observed that software resource allocation in cloud environment is a primary factor on the performance of an n-tier software application.

Doulamis *et al.* (2014) proposed an algorithm to improve software resource selection in a distributed computing environment. The proposed algorithm assigned software resources to tasks in a way that improves the utilization of software resources and minimize time requirements of tasks. It employed the concepts of graph partitioning in order to minimize time overlapping of tasks for a specific software resource and maximize time overlapping of tasks for different software resources.

Seth *et al.* (2012) conducted an empirical study to assess the role software requirements, stakeholders and resources in the development of a high quality software product. In this study, an interview is conducted with eleven participants involved in different roles in software development projects such as software programmers, testers, requirements managers and quality control personnel. The study reported that quality attributes of a software product depend on the type of software, software users and its application domain. Further, it reported that software product quality is highly dependent on allocated resources.

Chen *et al.* (2015) proposed an architecture to manage resources utilization of distributed datacenters and optical networks. They adopted two strategies for resources allocation along with two strategies for virtual network composition. The proposed architecture is experimented using three metrics: CPU utilization, latency and virtual network failure.

Yuan (2015) proposed a prediction algorithm for virtual resource scheduling in a cloud-computing environment. The proposed algorithm is built based on Support Vector Machine (SVM) to predict dynamic software resources requirements. Experimentation of the proposed algorithm is conducted two phases, training and prediction phases using real virtual resource data. The experimental results indicated that building the prediction algorithm based on the concepts of support vector machine has improved the prediction accuracy and has opened the opportunity to use such concepts to achieve real-time performance and high accuracy software resources requirements.

Li *et al.* (2016) proposed a framework to improve the communications between machines in software-defined cellular networks. The proposed framework is aimed improve the random access process of machine-to-machine communications. They developed a dynamic feedback and control loop to update the number of available resource blocks in a virtual machine-to-machine communications network.

Triwijoyo *et al.* (2017) proposed an approach to measure software reliability based fault analysis and categorization. This approach divided failure data into three groups and five modules. However, this is rather late in the software development life cycle.

Iskandar *et al.* (2016) used the use case point (Karner, 1993) method to measure software size of a knowledge management software in Bina Nusantara University. However, the measurement result is not based on internationally standardized method and it is only applicable to object-oriented software requirements and does not cover other types of requirements specifications.

Wang *et al*. (2017) proposed a regression-based model to identify software security requirements of three open source software products. However, the study reported that this model is not able to identify all security requirements specifications without an additional support of other software tools such as GitHub tool. This means that this model requires that software requirements specifications should be written in a certain style/format.

The International Standard for Software Functional Size Measurement: ISO 19761

The ISO19761: COSMIC (ISO, 2011) international standard proposes a general model of software functional requirements that explains the borderline among hardware and software. This standardized method measures functional size of a software product independently of the technology used to develop such a product based on the identified functional user requirements. The COSMIC measurement method propose generic model of software functional user requirements in order to clarify the boundary between hardware and software.

Figure 1 presents the COSMIC generic model that demonstrate the generic flow of data from a functional perspective. In this model, software is typically bounded by hardware and it is used either by a human user or by an engineered device. The human user interacts with software using a variety of input/output devices. Furthermore, software is bounded by storage hardware such as RAM memory.

The functionality of software is enclosed within the data groups of functional flows. In order to specify these functional flows, four data movement types are identified by COSMIC as follows:

- Two data movement types (i.e., Entry and eXit) are identified specify the functional flows between the human users and engineered devices from one side and software from the other side
- Two data movement types (i.e., Read and Write) are identified to specify the functional flows between storage and software

Diverse perceptions normally used for different measurement purposes. For example, in embedded and real time software, users are typically "engineered devices" which interact straightforward with software. For business and management application software, COSMIC considers that the users are one or more humans who interact directly with the business or management applications software across the border. In other words, the "I/O hardware" is ignored.

The ISO 19761 method measures the size of software based on identifiable of functional user requirements. Then, they are allocated to hardware and software from the unifying perspective of a system integrating these two "components". Since ISO 19761 is aimed at sizing software, only those requirements allocated to software are considered in its strategic measurement procedure.

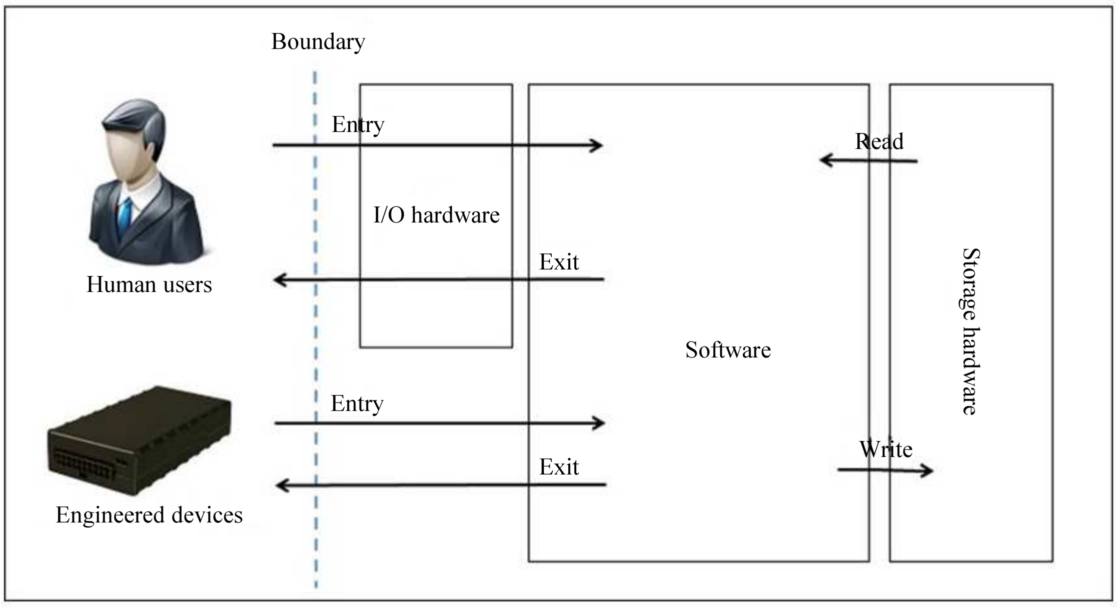

Fig. 1: A generic model of ISO 19761: COSMIC measurement method

Design of Reference Model of Software Resources Requirements

This section present the design of reference model to identify and measure software resources requirements based on international standards. Four steps are recommended by Abran (2010) to design a reference measurement model as follows:

- Determination of measurement objectives for software product
- Characterization of software resources terms
- Identification of resources entity types and relationship among entities
- Numerical assignment rules for software resources requirements

Determination of Measurement Objectives for Software Product

This part presents the main objective of the proposed reference model as a portion of a software product quality, along with the point of view of the reference measurement model and the anticipated uses of the measurement results:

- Measurement objective: To measure the functional size of software resources requirements using ISO19761: COSMIC as an intentional standard for software functional measurement
- Measurement point of view: Software resources perspective allocated to software resources requirements
- Intended use of measurement results: The uses of measurement results spans the whole software development life cycle. These functional size measures represent one of the primary inputs for the effort estimation process of software products. Further, these measures can be used for software benchmarking purposes

Characterization of Software Resources Terms

This part presents the terms and vocabulary of software resources requirements as defined by ISO standards ISO19761 (ISO, 2011) and ISO 14143-1 (IOS, 2007). Software resources requirements are classified as external and internal software resources. External software resources include I/O resources, memory resources and transmission resources. On the other hand, internal software resources include consumptions of hardware resources in system environment together with the software resources product during testing and/or operations. Therefore, software resources entities that are to be measured using the proposed reference model are as follows:

- External resources entities
  a. I/O resource measurements: include two entities to measure:
     - I/O devices utilization
     - User waiting time of I/O devices utilization
  b. Memory resource measurements: include one entity to measure:
     - Memory Utilization
  c. Transmission resource measurements: include three entities to measure:
     - Maximum transmission utilization
     - Transmission capacity utilization
     - Media device utilization
- Internal resources entities: There are two entities for internal measures of the I/O devices; meanwhile the ISO standards do not list any internal measures for memory and transmission resources requirements
  a. I/O related errors
  b. I/O loading

Identification of Resources Entity Types and Relationships among Entities

This part presents the identification of software resources entity types and the relationships among such entity types. Eight entity types are identified to help software engineers to identify software resources requirements based of ISO international standards. The technical specifications of such entity types are presented using the following entity type template:

Entity Type # (what software resources it identify):

- Entity name: Each entity type should have a descriptive meaningful name.
- Input of entity type: Designed data resource group Process used between input and output: Manipulation carried out on input resource data-group
- Output of entity type: Actual measured resource data group
- Entity type measurement role: Usage in functional size measurement using ISO standards
- Entity relationship: Type of relationship with other entity types

Further, this part present three metamodels to capture external and internal software resources requirements. A metamodel is an effective candidate to present visually different entity types, existing relationships, rules and constraints of a requirement-modeling problem. A metamodel provides software engineers with a roadmap in order to improve the representation of stakeholders needs, in a "language" that is understandable to other software development teams in software development project. Based on the identification of the eight entity types, three metamodels are proposed to present visually the entity types and their relationships.

Metamodel of I/O Device Resources

There are four entity types to capture the requirements of I/O devices resources; they are I/O devices utilization, I/O related errors, I/O loading and user waiting time of I/O devices. Figure 2 presents a metamodel that represents the four identified entity types and their corresponding relationships. This metamodel represent the relationship between entity types in terms of input, process and output.

Entity Type 1 (for external measurement for I/O device resources):

- Entity name: I/O devices utilization
- Input of entity type 1 is specified time that designed to occupy I/O devices
- Process used between input and output: execute concurrently a large number of tasks, record I/O device utilization and compare with design objectives
- Output of entity type 1 is actual time of I/O devices occupied
- Entity type 1 measures the functional size of the I/O devices utilization
- Entity relationship: many-many of I/O devices time on software

Entity Type 2 (for internal measurement for I/O device resources):

- Entity name: I/O related errors
- Input of entity type 2 is user-operating time during user observation
- Process used between input and output: calibrate test conditions
- Output of entity type 2 is number of warning messages or system failures
- Entity type 2 measures the functional size of the I/O related errors
- Entity relationship: many-many of I/O devices related errors on software

Entity Type 3 (for internal measurement for I/O device resources):

- Entity name: I/O loading
- Input of entity type 3 is designed I/O loading limits
- Process used between input and output: emulate conditions weather the system reaches a situation of maximum I/O load
- Output of entity type 3 is occupied of I/O loading limits
- Entity type 3 measures the functional size of the I/O loading limits
- Entity relationship: many-many of I/O devices loading period on software

Entity Type 4 (for external measurement for I/O device resources):

- Entity name: User waiting time of I/O devices
- Input of entity type 4 is designed waiting time of I/O devices
- Process used between input and output: Run the application and record number of errors due to I/O failures and warnings
- Output of entity type 4 is actual waiting time of I/O devices
- Entity type 4 measures the functional size of user waiting time of I/O devices
- Entity relationship: Many-many of I/O devices waiting

Entity Type 5 (for external measurement for memory resources):

- Entity name: Memory utilization
- Input of entity type 5 is designed required memory
- Process used between input and output: Calibrate test conditions, emulate conditions weather system reaches a situation of maximum I/O load, run the application and record number of errors due to I/O failures and warnings
- Output of entity type 5 is the actual memory needed
- Entity type 5 measures the functional size of memory utilization
- Entity relationship: Many-many of memory utilization on software

Metamodel of Transmission Resources

There are three entity types to capture the requirements of transmission resources; they are maximum transmission utilization, transmission capacity utilization and media devices utilization. Figure 4 presents a metamodel that represents the three identified entity types and their corresponding relationships. This metamodel represent the relationships between entity types in terms of input, process and output.

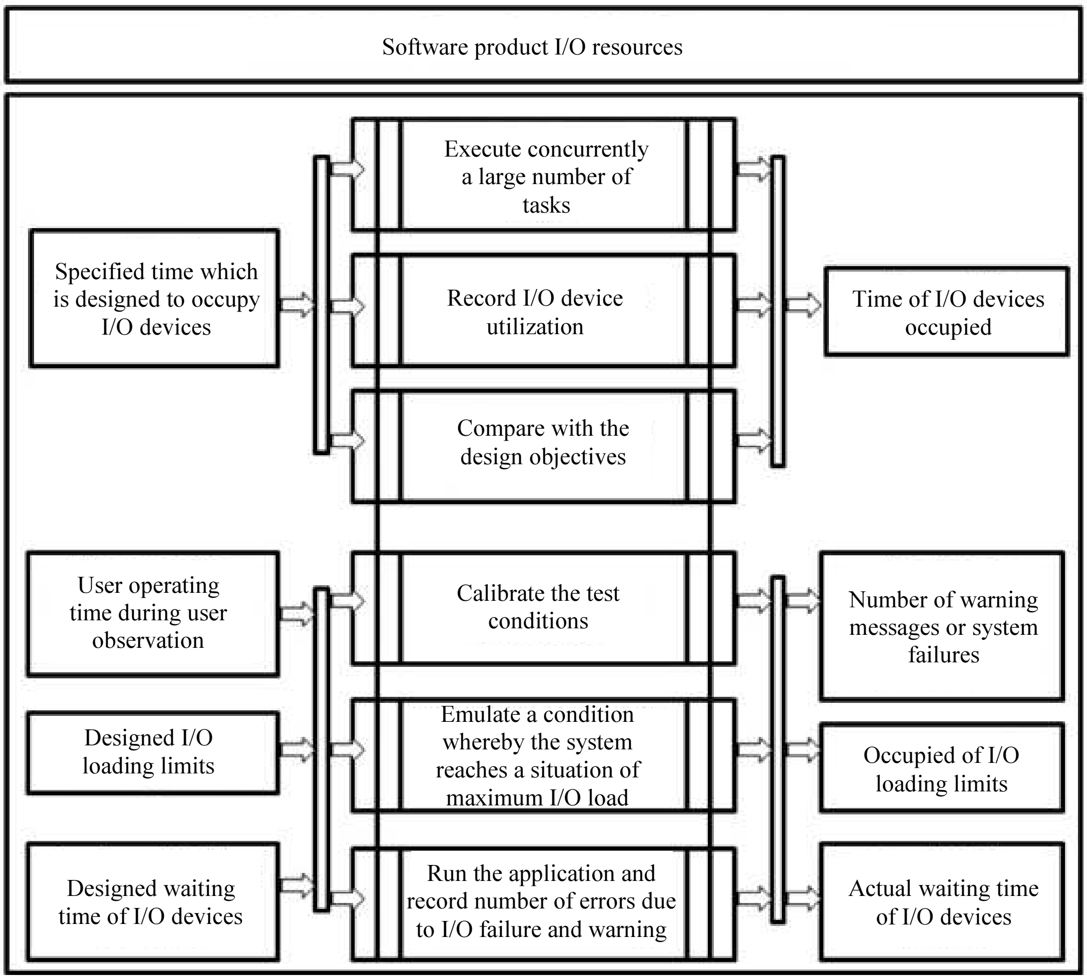
Software product I/O resources
Specified time which is designed to occupy I/O devices
Execute concurrently a large number of tasks
Record I/O device utilization
Compare with the design objectives
Time of I/O devices occupied
User operating time during user observation
Designed I/O loading limits
Designed waiting time of I/O devices
Calibrate the test conditions
Emulate a condition whereby the system reaches a situation of maximum I/O load
Run the application and record number of errors due to I/O failure and warning
Number of warning messages or system failures
Occupied of I/O loading limits
Actual waiting time of I/O devices

Fig. 2: A metamodel of software I/O devices resources

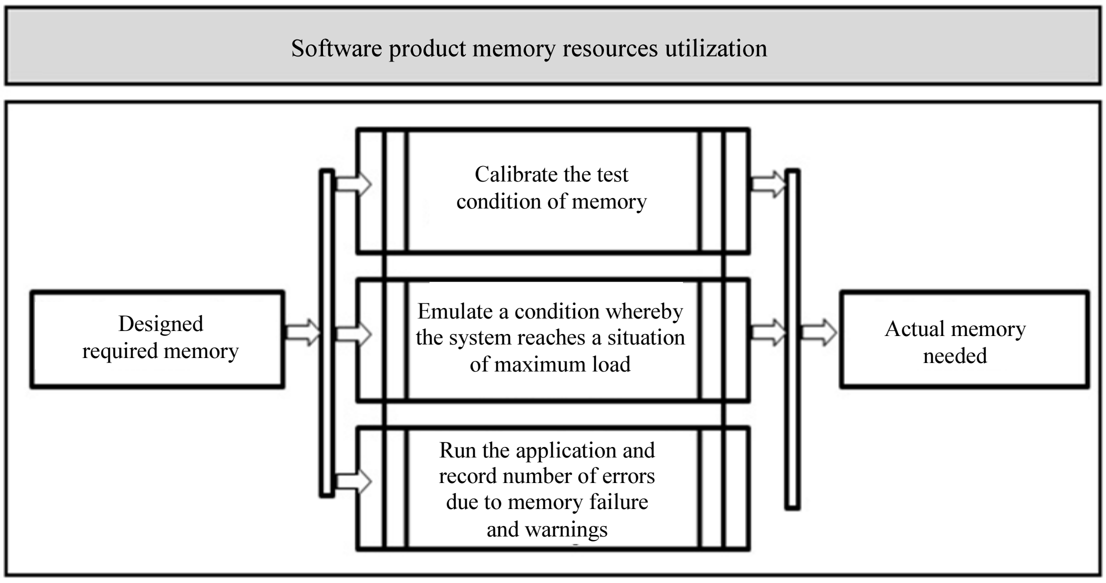
Software product memory resources utilization
Designed required memory
Calibrate the test condition of memory
Emulate a condition whereby the system reaches a situation of maximum load
Run the application and record number of errors due to memory failure and warnings
Actual memory needed

Fig. 3: A metamodel of software memory resources

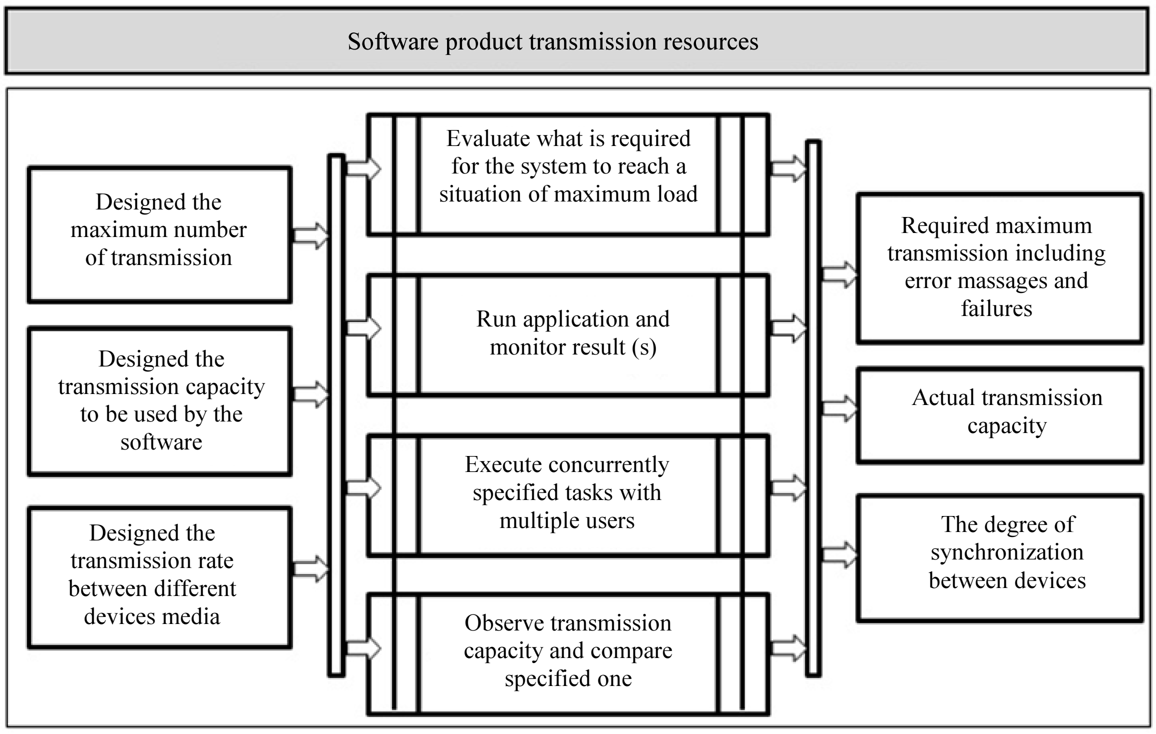
Software product transmission resources
Designed the maximum number of transmission
Designed the transmission capacity to be used by the software
Designed the transmission rate between different devices media
Evaluate what is required for the system to reach a situation of maximum load
Run application and monitor result (s)
Execute concurrently specified tasks with multiple users
Observe transmission capacity and compare specified one
Required maximum transmission including error massages and failures
Actual transmission capacity
The degree of synchronization between devices

Fig. 4: A metamodel of software transmission resources

Entity Type 6 (for external measurement for transmission resources):

- Entity name: maximum transmission utilization
- Input of entity type 6 is designed maximum number of transmission
- Process used between input and output: Evaluate what is required for the system to reach a situation of maximum load
- Output of entity type 6 is the required maximum transmission including error messages and failures
- Entity type 6 measures the functional size of maximum transmission utilization
- Entity relationship: many-many of transmission load on software

Entity Type 7 (for external measurement for transmission resources):

- Entity name: Transmission capacity utilization
- Input of entity type 7 is designed transmission capacity
- Process used between input and output: Run application and monitor the results
- Output of the entity type 7 is the actual transmission capacity
- Entity type 7 measures the functional size of transmission capacity utilization
- Entity relationship: Many-many of transmission capacity on software

Entity Type 8 (for external measurement for transmission resources):

- Entity name: Media devices utilization
- Input of entity type 8 is: Designed transmission rate between different media devices
- Process used between input and output: Execute concurrency specified tasks with multiple users, observe transmission capacity and compare with the specified one
- Output of entity type 8 is the degree of synchronization between devices
- Entity type 8 measures the functional size of media devices utilization
- Entity relationship: Many-many of media devices utilization on software

Numerical Assignment Rules for Software Resources Requirements

The foundations of the numerical assignment rules for software resources requirements are presented in the previous metamodels of I/O device resources, memory resources and transmission resources (Fig. 2 to 4). Numerical assignments rules can be described using descriptive text (i.e., practitioner's description) or using mathematical expressions (i.e., formal theoretical viewpoint).

According to the international standard for software functional size measurement – ISO19761, a functional process is defined as an elementary component of a set of functional user requirements. It includes a unique cohesive and independently executable set of data movement types (ISO, 2011). Four data movement types are identified by ISO19761: An 'Entry' data movement type moves a data resource group into software from a functional user and an 'eXit' data movement type moves a data resources group out. Further, the 'Write' and 'Read' data movement types move a data group to and from persistent storage, respectively (ISO, 2011). One (1) CFP (i.e., COSMIC Function Point) represent a functional size measurement of each counted data movement type (ISO, 2011).

Data resources groups form sources and/or to data destinations for software resources requirements. Table 1 and 2 presents data sources/destinations of software resources requirements. In both tables, software resources are categorized into the three classes, I/O device resources, memory resources and transmission resources - see column #1, data sources/destinations are next presented in column #2 and finally the objects of interest (i.e., resource type) are presented in column #3.

Quality Evaluation of Software Resources

This section presents an extension of the proposed reference model software resources requirements. It presents building numerical assignments rules based on mathematical expressions using descriptive text rules in ISO25010 (2011). The numerical assignment rules are appended to the I/O resources devices, memory resources and transmission resources metamodels. The resulting metamodels presented in this section represent instantiation metamodels of the proposed reference model. They can be used to identify and measure software resources requirements based on the concepts in ISO25010 (2011), which can be considered as quality evaluation of software resources requirements in addition to the measurement benefit.

Table 1: Resources data sources

Categories	Data Sources	Objects of Interest
I/O devices resources	• Specified time, which is designed to occupy I/O devices	• Time
	• Actual time of I/O devices occupied	• Time
	• User operating time during user observation	• Number
	• Number of warning messages or system failures	• Number
	• Designed I/O loading limits	• Loading limit
	• Occupied of I/O loading limits	• Loading limit
	• Designed waiting time of I/O devices	• Time
	• Actual waiting time of I/O devices	• Time
Memory resources	• Designed required memory	• Size
	• Actual memory needed	• Size
Transmission resources	• Designed maximum number of transmission	• Transmission no.
	• Required maximum transmission including error messages and failures	• Transmission no.
	• Designed transmission capacity	• Transmission capacity
	• Actual transmission capacity	• Transmission capacity
	• Design of transmission rate between different devices media	• Transmission rate
	• Degree of synchronization between devices	• Transmission rate

Table 2: Resources data destinations

Categories	Data Destinations
I/O devices resources	• I/O devices utilization
	• I/O related errors
	• I/O loading
	• User waiting time of I/O devices
Memory resources	• Memory utilization
Transmission resources	• Maximum transmission utilization
	• Transmission capacity utilization
	• Media devices utilization

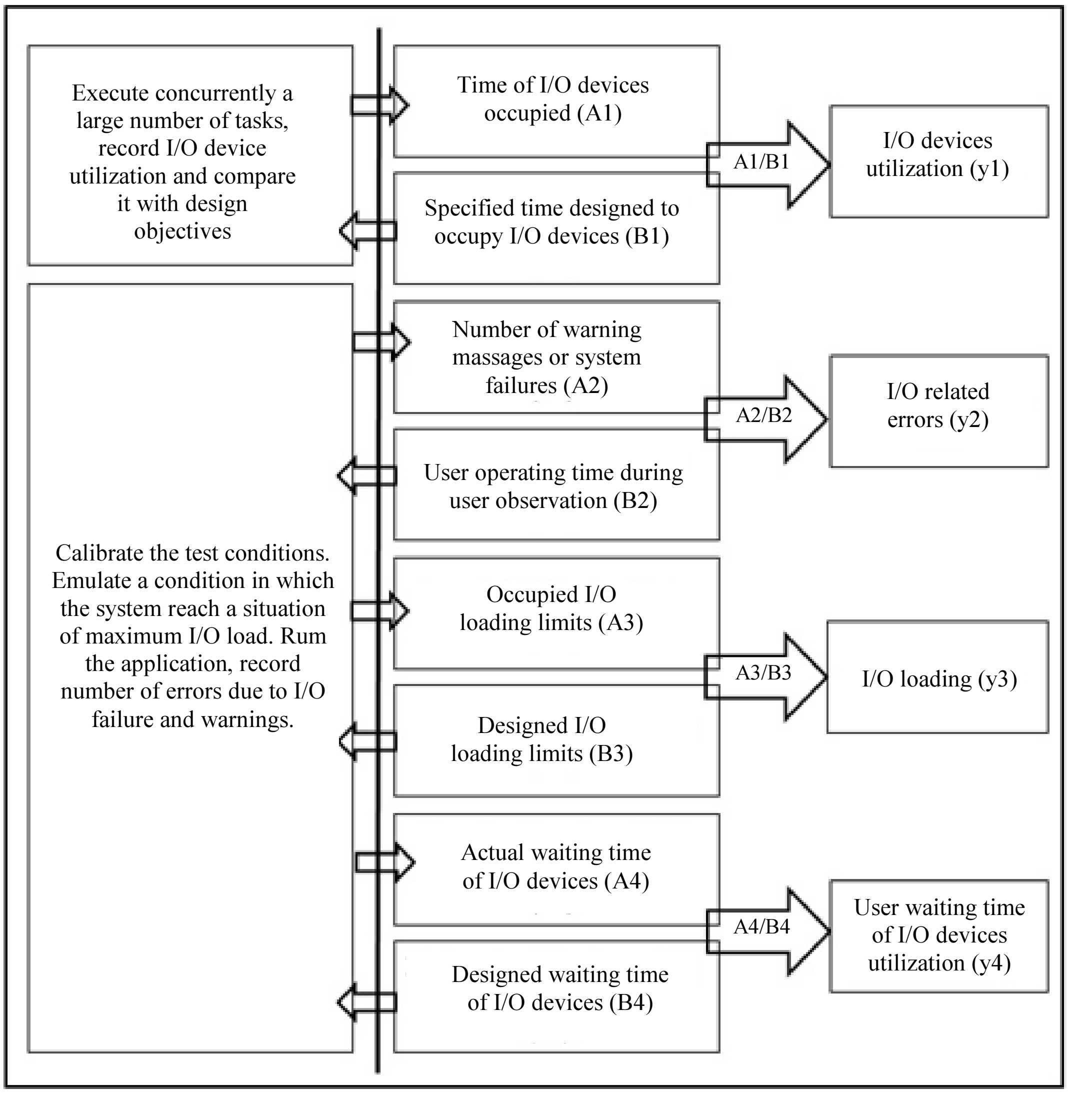
Execute concurrently a large number of tasks, record I/O device utilization and compare it with design objectives
Time of I/O devices occupied (A1)
Specified time designed to occupy I/O devices (B1)
A1/B1
I/O devices utilization (y1)
Number of warning massages or system failures (A2)
User operating time during user observation (B2)
A2/B2
I/O related errors (y2)
Calibrate the test conditions. Emulate a condition in which the system reach a situation of maximum I/O load. Rum the application, record number of errors due to I/O failure and warnings.
Occupied I/O loading limits (A3)
Designed I/O loading limits (B3)
A3/B3
I/O loading (y3)
Actual waiting time of I/O devices (A4)
Designed waiting time of I/O devices (B4)
A4/B4
User waiting time of I/O devices utilization (y4)

Fig. 5: Metamodel of I/O devices resources with numerical assignment rules

Figure 5 presents an instantiation metamodel to measure I/O device resources externally and internally for one functional process. The measurement of I/O device externally is based on entity type (1) and entity type (4). Entity type 1 is used to measure the external software resources throughout executing concurrently a large number of tasks and record I/O device utilization (Equation 1). Further, entity type 4 is used to measure the external software resources throughout run the application of record of errors due to I/O failures and warning (Equation 2). It is worth mentioning that the measurement result of Equation 1 and 2 is in Time and Equation 3 is used calculate all the arithmetic summation of all data movement types in one functional process:

$$I\,/\,O\,devices\,utilization(y1) = \frac{A1}{B1} \tag{1}$$

$$User\,watining\,time\,I\,/\,O\,devices\,utilization(y4) = \frac{A4}{B4} \tag{2}$$

$$\sum Data\,movement(data\,resource\,group) = \sum(y1 + y4) \tag{3}$$

Where:

$A1$	=	Time of I/O devices occupied
$B1$	=	Specified time designed to occupy I/O devices
$A4$	=	Actual waiting time of I/O devices
$B4$	=	Designed waiting time of I/O devices

On the other hand, the measurement of I/O devices internally of one functional process is based on entity type (2) and entity type (3). Entity type 2 is used to measure internal software resources throughout calibrating the test conditions and emulate a condition whereby the system reaches a situation of maximum I/O loading to define the

I/O errors (Equation 4). Entity type 3 is used to measure internal software resources throughout calibrating the test condition to define maximum I/O loading (Equation 5). It is worth mentioning that the measurement result of Equation 4 and 5 is in Number. Equation 6 is used calculate all the arithmetic summation of all data movements in one functional process:

$$I/O\,related\,errors(y2) = \frac{A2}{B2} \tag{4}$$

$$I/O\,loading\,limits(y3) = \frac{A3}{B3} \tag{5}$$

$$\sum Data\,movement(data\,resource\,group) = \sum(y2 + y3) \tag{6}$$

Where:

$A2$	=	Number of warning msgs or system failures
$B2$	=	User operating time during user observation
$A3$	=	Occupied I/O loading limits
$B3$	=	Designed I/O loading limits

Figure 6 presents an instantiation metamodel to measure memory resources - externally - for one functional process and it is based on entity type (5). Entity type 5 is used to measure external software resources throughout executing concurrently a large number of tasks and run the application and record number of errors due to memory failures and warnings for one functional process (Equation 7). It is worth mentioning that there are no internal memory measures as defined in ISO19761 (2011). Further, the measurement result of Equation 7 is in Size and Equation 8 is used calculate all the arithmetic summation of all data movements in one functional process:

$$Momery\,utilization(y5) = \frac{A5}{B5} \tag{7}$$

$$\sum Data\,movement(data\,resource\,group) = \sum(y5) \tag{8}$$

Where:

$A5$	=	Actual memory needed
$B5$	=	Designed required memory

Figure 7 presents an instantiation metamodel to measure transmission resources - externally - for one functional process and it is based on entity type 6, entity type 7 and entity type 8. Entity type 6 is used to measure the external software resources throughout evaluate what is required for the system to reach a situation of maximum load. Further, entity type 7 is used to measure the external software resources throughout observing transmission capacity and compare it to the specified one. Finally, entity type 8 is used to measure the external software resources throughout executing concurrently specified tasks with multiple users. The measurement results of Equation 9 to 11 is in transmission number, capacity and rate, respectively:

$$Maximum\,tramission(y6) = \frac{A6}{B6} \tag{9}$$

$$Transmission\,capacity(y7) = \frac{A7}{B7} \tag{10}$$

$$Media\,devices\,utilization(y8) = \frac{A8}{B8} \tag{11}$$

Where:

A6	=	Maximum required transmission including error messages and failures
B6	=	Designed maximum number of transmissions
A7	=	Actual transmission capacity
B7	=	Designed transmission capacity used by software
A8	=	Degree of synchronization between devices
B8	=	Designed transmission rate between different media devices

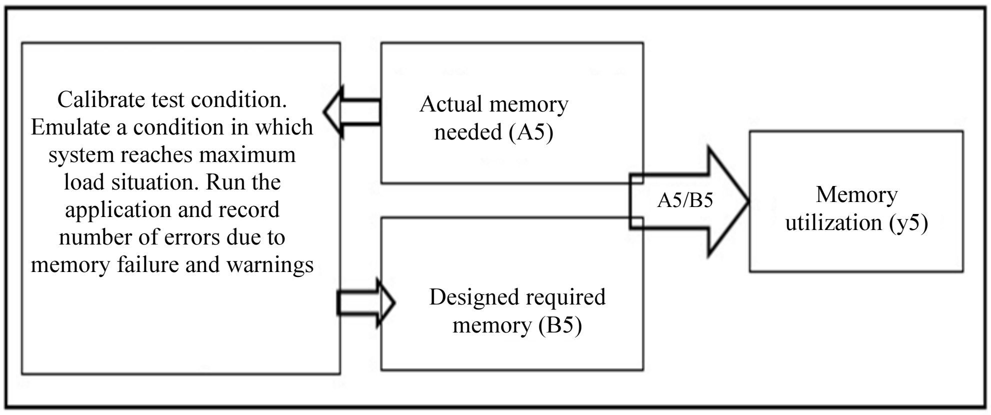
Calibrate test condition. Emulate a condition in which system reaches maximum load situation. Run the application and record number of errors due to memory failure and warnings
Actual memory needed (A5)
Designed required memory (B5)
A5/B5
Memory utilization (y5)

Fig. 6: Metamodel of memory resources with numerical assignment rules

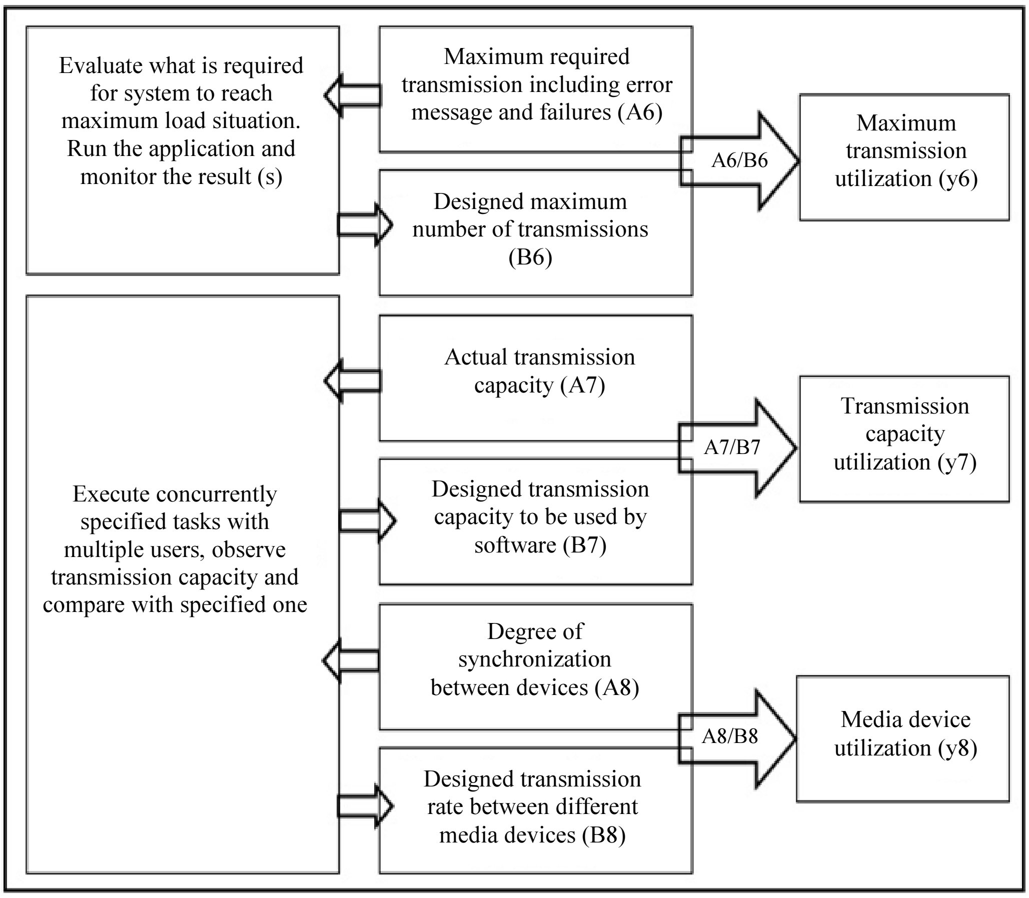
Evaluate what is required for system to reach maximum load situation. Run the application and monitor the result (s)
Maximum required transmission including error message and failures (A6)
Designed maximum number of transmissions (B6)
A6/B6
Maximum transmission utilization (y6)
Execute concurrently specified tasks with multiple users, observe transmission capacity and compare with specified one
Actual transmission capacity (A7)
Designed transmission capacity to be used by software (B7)
A7/B7
Transmission capacity utilization (y7)
Degree of synchronization between devices (A8)
Designed transmission rate between different media devices (B8)
A8/B8
Media device utilization (y8)

Fig. 7: Metamodel of transmission resources with numerical assignment rules

Verification of the Proposed Reference Model with an Automated Teller Machine

Scope and Objective

This section presents a verification of the proposed reference model of software resources requirements using the software specifications of an Automated Teller Machine (ATM) system. The automated teller machine system is a real time system that is developed for banks' clients to conduct several financial services without the interference of bank personnel. A sample of requirements specifications is selected for the withdrawal process of the automated teller machine. These requirements specifications represent an explanation of the withdrawal process that a typical user normally conduct to withdraw money from the automated teller machine. It is worth mentioning that the authors have not selected certain ideal (complete) requirements specifications of the withdrawal process with possibly quality attributes identified for two reasons:

- During an early phase of the software development life cycle, it is expected to obtain software requirements specifications in which they are vague, incomplete, or inaccurate. Therefore, the selected specifications can emulate a similar case.
- Selecting ideal requirements specifications will prevent quality evaluation of software resources requirements using the proposed reference model.

Software Specifications of an Automated Teller Machine

An automated teller machine wait for the user to start interaction process by inserting the bank client-card into the card reader to read all necessary information from magnetic strip and/or the microprocessor chip. This information includes a unique card number and other encrypted personal and account information. Upon insertion of user card into the card reader, the user shall wait until the "insert PIN" screen appears; it is considered as the starting step for client authentication. When the "insert PIN" screen appears, the user shall enter her/his personal identification number. Then, the system need to verify whether the entered information (i.e., PIN) is correct. If the entered personal identification number is wrong, the system will automatically eject the client-card. However, if it is correct, the system will ask the user to enter the withdrawal amount. After that, the system need to check the account balance in order to complete the withdrawal transaction and withdraw cash for the user. On the other hand, if the account balance is not enough or user entered a wrong PIN, the transaction information is saved in the account table (on the server database). Figure 8 presents a flow of activities diagram for withdrawal process of an automated teller machine system.

Experimentation of the Proposed Reference Model

The withdrawal process in the automated teller machine is analyzed based on flow of functionality from the user view. The following steps represent the flow of functionality steps: Waiting to insert card, waiting to enter PIN, waiting to check PIN, waiting to enter amount, verify balance and get cash.

Two approaches are adopted for the purpose of this experiment:

- First approach identifies software resources requirements using the proposed reference model and then measures the functional size of the software resources requirements using the concepts exist in ISO19761 (IOS, 2007)
- Second approach identifies software resources requirements using the proposed reference model and then measures such identified requirements based on the concepts exist in the international standard ISO25010 - systems and Software Quality Requirements and Evaluation (SQuaRE) standard (ISO25010, 2011)

Table 3 presents the identification and the measurement of software resources requirements of the automated teller machine using the first approach. Using this approach, there are seven functional processes are identified and they are presented in column #1. It can be noticed that there are two functional processes identified with the name "verify balance". One functional process is associated with memory resources and another process is associated with transmission resources.

For each identified functional process, the corresponding resource type and description of measured resource is presented in column #2 and column #3. The description of the measured resource represents a data resource group, which this is moved by a one data movement type. Each data movement type that moves one data resource group is measured as one CFP (COSMIC Function Point). For example, the functional process "waiting to insert card" is measured using ISO19761 as follows: it represents an I/O resource type and includes two data resource groups. The first data resource group is moved when the user enters her/his client-card using an I/O device (i.e. card reader); this movement is defined as an Entry (E) data movement type and equals to (1) CFP. The second data resource group is moved from the system using the I/O device (i.e. ATM screen); this movement is defined as an eXit data movement type and equals to (1) CFP. Therefore, the total functional size measurement for the "waiting to insert card" functional process equals to (2) COSMIC Function Points (CFPs).

Table 3: Identification and measurement of resources requirements of ATM system using first approach

Functional process	Resource type	Description of measured resource	Data movement type	CFP
Waiting to insert card	I/O	Specified time which is designed to occupy input resources to ATM	Entry (E)	2
		Time of I/O devices occupied	eXit (X)	
Waiting to enter PIN	I/O	User operating time to enter PIN during user observation	Entry (E)	2
		Number of warning messages or system failure	eXit (X)	
Waiting to check PIN	I/O	Designed loading limits to check PIN code	Entry (E) Read (R)	4
		Occupied of I/O loading limits	eXit (X) Write (W)	
Waiting to enter amount	I/O	Designed loading limits to check PIN code	Entry (E)	2
		Occupied of I/O loading limits	eXit (X)	
Verify balance	Memory	Designed required memory	Read (R)	2
		Actual memory needed	Write (W)	
Verify balance	Transmission	Designed transmission rate between different devices media	Read (R)	2
		Degree of synchronization between devices	Write (W)	
Get cash	I/O	Designed waiting time of I/O devices	Entry (E)	2
		Actual waiting time of I/O devices	eXit (X)	
Total functional size of software resources				16 CFP

Table 4: Identification and measurement of resources requirements of ATM system using second approach

Functional Process	Resource Type	Description of measured resource	Unit of measure	Equation used
Waiting to insert card	I/O	Specified time which is designed to occupy input resources to ATM	Time	(1)
		Time of I/O devices occupied	Time	(1)
Waiting to enter PIN	I/O	User operating time to enter PIN during user observation	Time	(2)
		Number of warning messages or system failure	Number	(3)
Waiting to check PIN	I/O	Designed loading limits to check PIN code	Loading limit	(4)
		Occupied of I/O loading limits	Loading limit	(4)
Waiting to enter amount	I/O	Designed loading limits to check PIN code	Loading limit	(4)
		Occupied of I/O loading limits	Loading limit	(4)
Verify balance	Memory	Designed required memory	Size	(5)
		Actual memory needed	Size	(5)
Verify balance	Transmission	Designed the transmission rate between different devices media	Rate	(8)
		The degree of synchronization between devices	Rate	(8)
Get cash	I/O	Designed waiting time of I/O devices	Time	(2)
		Actual waiting time of I/O devices	Time	(2)

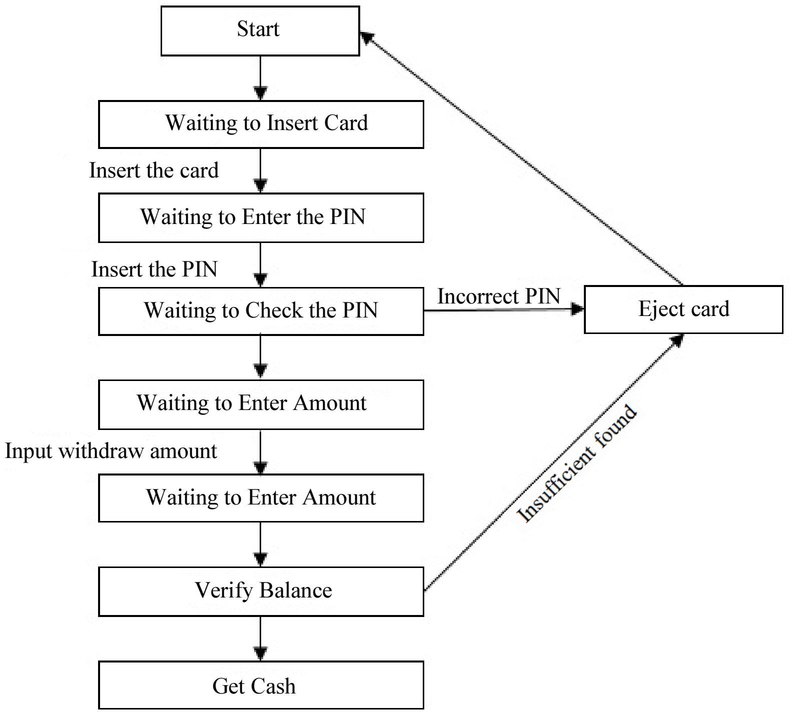

Fig. 8: Flow of activities diagram for withdrawal process of an automated teller machine system

The measurement concepts of ISO19761 applies to the other six functional process in the similar fashion. For example, the fifth functional process "verify balance" is measured using ISO19761 as follows: It represents a memory resource type and includes two data resource groups. The first data resource group is moved from the storage hardware into the ATM software designated functionality; this movement is defined as Read (R) data movement type and equals to (1) CFP. The first data resource group is moved from the ATM software designated functionality and written into storage hardware; this movement is defined as Write (W) data movement type and equals to (1) CFP. Therefore, the total functional size measurement for the "verify balance" functional process equals to (2) COSMIC Function Points (CFPs).

The total functional size measurement of software resources requirements of the withdrawal process for the

ATM system, equals to the arithmetic summation of the functional size of all identified functional process and this is equal to (16) COSMIC Function Points (CFPs).

Table 4 presents the identification and the measurement of software resources requirements of the automated teller machine using the second approach. Using this approach, seven functional processes are identified and they are presented in column #1. It can be noticed that there are two functional processes identified with the name "verify balance". One functional process is associated with memory resources and another process is associated with transmission resources.

For each functional process out of the seven identified processes, there is a corresponding "resource type" and "description of measured resource" that is presented in column #2 and #3, respectively. Furthermore, the measurement units and the corresponding mathematical equations used to calculate resources size for each functional process are presented in column #4 and #5, respectively. For example, the 'waiting to insert card' functional process use I/O resource and include two data resource groups, the measurement unit for both data resource groups is in 'time' and this resource is measured using Equation 1.

Summary of Findings

Both approaches adopted in this experiment are able to identify the same number of functional processes. The functional size measurement using the first approach, measures the functional size of software resources requirements independently of the technology used to develop such a product based on the identified functional user requirements. The resulting measures have a unified measurement unit (i.e., CFP - COSMIC function point) and they are aggregated arithmetically to be used in the effort estimation process of a software product.

On the other hand, using the second approach to measure software resources requirements do not yield unified size units of software resources requirements. For example, the measurement unit for I/O device resources is in 'time' unit and the measurement unit of memory resources is in 'size' unit. Therefore, software engineers who need a unified measurement - for effort estimation purposes - cannot aggregate these measures arithmetically.

Further notice, not all the proposed equations using the second approach are used to measure software resources requirements of the ATM system. This can be referred to the fact these specifications are not developed (i.e., identified and modeled) using a model that is built based on an international standard, such as ISO25010. However, this can be considered as quality evaluation of software resources requirements. Finally, the measurement of software resources using the second approach has not lead to numerical values result; it is be because the requirements specifications of the ATM system have provided no details about designed values of software resources requirements.

Threats to Validity

An internal validity threat is associated with any changes in the design of the experiment such as lack of description for the concepts to be evaluated in the experiment. To mitigate the risk of this threat to validity, the principal researcher who designed the reference model has not conducted the experiment himself. However, another researcher (i.e., author) has conducted a pilot test verify the validity the experimental steps and finally a third researcher (i.e., author) has conducted the experimentation of the proposed reference model.

An external validity threat is expressed as the extent that the experimental results can be generalized beyond the experimental settings. The proposed reference model of software resources requirements is experimented only using the requirements specifications of the withdrawal process for an automated teller machine. To mitigate the risk of this threat to validity, further experiments should be conducted in the future using the requirements specifications of different software products of different types (i.e., real time software, business application software, or even a hybrid of both types).

Conclusion

This paper proposed a new reference model to identify and measure software resources requirements based on ISO international standards ISO19761 and ISO14143-1. This proposed reference model measures functional size of software resources requirements allocated to software using the concepts of ISO19761 and independently from development technology used to develop the software product. Further, the proposed reference model measures software resources requirements using the concepts in ISO25010, which can be considered as quality evaluation of software resources requirements.

The experimental results showed that the proposed reference model is capable of identifying and measuring the functional size of software resources requirements. The industrial impact of this paper is improving one of the inputs for the effort estimation process. Therefore, it will improve planning, management and development of software at different phases of the software development life cycle. Further, the measurement results of the proposed reference model can be used for benchmarking software purposes conducted by research groups such as ISBSG software benchmarking group.

Future work will be directed to conduct more experiments using requirements specifications of different software products of different types, in order to generalize the results reported in this study. In addition, future work will be directed to automate the measurement of software resources requirements through building (or add to an existing) automated measurement tool.

Acknowledgment

We would like to thank the Hashemite University and the University of Petra for supporting us in using their facilities to conduct this research paper.

Authors Contributions

Khaled Almakadmeh: Design of the Reference Model of Software Resources Quality.

Kenza Meridji: Conduct of the case study.

Khalid T. Al-Sarayreh: Conduct of the literature review.

Ethics

Authors should address any ethical issues that may arise after the publication of this manuscript.

References

Abran, A., 2010. Software Metrics and Software Metrology. 1st Edn., IEEE Computer Society Press. ISBN: ISBN-10: 0470597208, pp: 348.

Abran, A., K.T. Al-Sarayreh and J.J. Cuadrado-Gallego, 2013. A standards-based reference framework for system portability requirements. Comput. Standards Interfaces, 35: 380-395. DOI: 10.1016/j.csi.2012.11.003

Al-Sarayreh, K.T. and A. Abran, 2010. A generic model for the specification of software interface requirements and measurement of their functional size. Proceedings of the 8th International Conference on Software Engineering Research, Management and Applications, May 24-26, IEEE Xplore Press, Montreal, Canada, pp: 217-222. DOI: 10.1109/SERA.2010.35

Al-Sarayreh, K.T., A. Abran and J.J. Cuadrado-Gallego, 2013a. A standards-based model of system maintainability requirements. J. Software: Evolut. Process, 25: 459-505. DOI: 10.1002/smr.1553

Al-Sarayreh, K.T., I. Al-Oqily and K. Meridji, 2013b. A standard-based reference framework for system operations requirements. Int. J. Comput. Applic. Technol., 47: 351-363. DOI: 10.1504/IJCAT.2013.055328

Al-Sarayreh, K.T., I. Al-Oqily and K. Meridji, 2012. A standard based reference framework for system adaptation and installation requirements. Proceedings of the 6th International Conference on Next Generation Mobile Applications, Services and Technologies, Sept. 12-14, IEEE Xplore Press, Paris, France, pp: 7-12. DOI: 10.1109/NGMAST.2012.19

Al-Sarayreh, K.T., K. Meridji, E. Fayyoumi and S. Idwan, 2014. A novel approach to build a generic model of photovoltaic solar system using sound biometric techniques. Int. J. Inform. Technol. Web Eng., 9: 31-44. DOI: 10.4018/ijitwe.2014010103

Alur, R. and G. Weiss, 2008. Regular specifications of resource requirements for embedded control software. Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, Apr. 22-24, IEEE Xplore Press, St. Louis, MO, USA, pp: 159-168. DOI: 10.1109/RTAS.2008.13

Arfeen, M.A., K. Pawlikowski and A. Willig, 2011. A framework for resource allocation strategies in cloud computing environment. Proceedings of the 35th Annual Computer Software and Applications Conference Workshops, Jul. 18-22, IEEE Xplore Press, Munich, Germany, pp: 261-266. DOI: 10.1109/COMPSACW.2011.52

Arnold, W.C., D.J. Arroyo, W. Segmuller, M. Spreitzer and M. Steinder *et al*., 2014. Workload orchestration and optimization for software defined environments. IBM J. Res. Dev., 58: 1-11. DOI: 10.1147/JRD.2014.2304864

Chen, H., J. Zhang, Y. Zhao, J. Deng and W. Wang *et al*., 2015. Experimental demonstration of datacenter resources integrated provisioning over multi-domain software defined optical networks. J. Lightwave Technol., 33: 1515-1521. DOI: 10.1109/JLT.2015.2395079

Doulamis, N.D., P. Kokkinos and E. Varvarigos, 2014. Resource selection for tasks with time requirements using spectral clustering. IEEE Trans. Comput., 63: 461-474. DOI: 10.1109/TC.2012.222

Eklov, D., N. Nikoleris and E. Hagersten, 2014. A software based profiling method for obtaining speedup stacks on commodity multi-cores. Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, Mar. 23-25, IEEE Xplore Press, California, USA, pp: 148-157. DOI: 10.1109/ISPASS.2014.6844479

Fotrousi, F., S.A. Fricker and M. Fiedler, 2014. Quality requirements elicitation based on inquiry of quality-impact relationships. Proceedings of the 22nd International Conference on Requirements Engineering, Aug. 25-29, IEEE Xplore Press, Karlskrona, Sweden, pp: 303-312. DOI: 10.1109/RE.2014.6912272

IOS, 2007. Information Technology-software measurement-functional size measurement Part 1: Definition of concepts (ISO/IEC-14143-1). International Organization for Standardization, Geneva, Switzerland.

Iskandar, K., F. Gaol, B. Soewito, H Leslie and H. Warnars *et al.*, 2016. Software size measurement of knowledge management portal with use case point. Proceedings of the International Conference on Computer, Control, Informatics and its Applications, Oct. 3-5, IEEE Xplore Press, Tangerang, Indonesia, pp: 42-47. DOI: 10.1109/IC3INA.2016.7863021

ISO, 2011. COSMIC: International standard for software functional size measurement: COSMIC. International organization for standardization, Geneva, Switzerland.

ISO25010, 2011. Systems and software engineering-systems and Software Quality Requirements and Evaluation (SQuaRE)-System and software quality models. International Organization for Standardization, Geneva, Switzerland.

Jia, Y., L. Sun, W. Yabin and G. Yubo, 2012. Research on maintenance task allocation and support resource requirement analysis for ordnance equipment. Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance and Safety Engineering, Jun. 15-18, IEEE Xplore Press, Chengdu, China, pp: 459-465. DOI: 10.1109/ICQR2MSE.2012.6246274

Karner, G., 1993. Metrics for objectory, Diploma Thesis, University of Linkoping, Sweden. No. LiTHIDAEx-9344:21.

Khatter, K. and A. Kalia, 2013. Impact of non-functional requirements on requirements evolution. Proceedings of the 6th International Conference on Emerging Trends in Engineering and Technology, Dec. 16-18, IEEE Xplore Press, Nagpur, India, pp: 61-68. DOI: 10.1109/ICETET.2013.15

Kocsis, G. and P. Ekler, 2012. Resource requirement estimation of advertising websites. Proceedings of the International Symposium on Computational Intelligence and Informatics, Nov. 20-22, IEEE Xplore Press, Budapest, Hungary, pp: 207-211. DOI: 10.1109/CINTI.2012.6496761

Li, J., Q. Wang, C.A. Lai, J. Park and D. Yokoyama *et al.*, 2014. The impact of software resource allocation on consolidated n-tier applications. Proceedings of the 7th International Conference on Cloud Computing, Jun. 27-Jul. 2, IEEE Xplore Press, Anchorage, AK, USA, pp: 320-327. DOI: 10.1109/CLOUD.2014.51

Li, M., F. Richard Yu, P. Si, E. Sun and Y. Zhang, 2016. Random access and resource allocation in software-defined cellular networks with M2M communications. Proceedings of the IEEE Global Communications Conference, Dec. 4-8, IEEE Xplore Press, Washington, DC, USA, pp: 1-6. DOI: 10.1109/GLOCOM.2016.7842194

Liu, C., Y. Zou, S. Cai, B. Xie and H. Mei, 2011. Finding the merits and drawbacks of software resources from comments. Proceedings of the International Conference on Automated Software Engineering, Nov. 6-10, IEEE Xplore Press, Lawrence, KS, USA, pp: 432-435. DOI: 10.1109/ASE.2011.6100091

Meridji, K., K.T. Al-Sarayreh and A. Al-Khasawneh, 2013. A generic model for the specification of software reliability requirements and measurement of their functional size. Int. J. Inform. Quality, 3: 139-163. DOI: 10.1504/IJIQ.2013.054279

Perez, J.F., G. Casale and S. Pacheco-Sanchez, 2015. Estimating computational requirements in multi-threaded applications. IEEE Trans. Software Eng., 41: 264-278. DOI: 10.1109/TSE.2014.2363472

Seth, F.P., E. Mustonen-Ollila, O. Taipale and K. Smolander, 2012. Software quality construction: Empirical study on the role of requirements, stakeholders and resources. Proceedings of the 19th Asia-Pacific Software Engineering Conference, Dec. 4-7, IEEE Xplore Press, Hong Kong, China, pp: 17-26. DOI: 10.1109/APSEC.2012.119

Shilun, L., N. Mingfang, H. Kaikai and Y. Ma, 2011. Study on the main requirements of equipment resource-ability design. Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance and Safety Engineering, Jun. 17-19, IEEE Xplore Press, Xi'an, China, pp: 709-713. DOI: 10.1109/ICQR2MSE.2011.5976709

Triwijoyo, B., F. Gaol, B. Soewito and H. Warnars, 2017. Software reliability measurement base on failure intensity. Proceedings of the 3rd International Conference on Science in Information Technology, Oct. 25-26, IEEE Xplore Press, Bandung, Indonesia, pp: 176-181. DOI: 10.1109/ICSITech.2017.8257106

Wang, J., W. Tepfenhart, D. Rosca and A. Tsai, 2008. Workflow resource requirement modeling and analysis. Proceedings of the International Conference on Networking, Sensing and Control, Apr. 6-8, IEEE Xplore Press, Sanya, China, pp: 246-251. DOI: 10.1109/ICNSC.2008.4525219

Wang, W., N. Hussein, A. Gupta and Y. Wang, 2017. A regression model based approach for identifying security requirements in open source software development. Proceedings of the IEEE 25th International Requirements Engineering Conference, Sept. 4-8, IEEE Xplore Press, Lisbon, Portugal, pp: 443-446. DOI: 10.1109/REW.2017.56

Yuan, S., 2015. Virtual resource scheduling prediction based on a support vector machine in cloud computing. Proceedings of the 8th International Symposium on Computational Intelligence and Design, Dec. 12-13, IEEE Xplore Press, Hangzhou, China, pp: 110-113. DOI: 10.1109/ISCID.2015.303