Model of Resources Requirements for Software Product Quality Using ISO Standards

Kenza Meridji, Khalid T. Al-Sarayreh and Tatiana Balikhina

Abstract— **Resources requirements according to ISO standards describe all requirements related software product quality including resources list of the hardware environment in which the software is specified to operate and resource utilization requirements list of the sizing and timing requirements applicable to the software item under specification and computer software requirements description of the computer software to be used with the software under specification or incorporated into the software item; for instance operating system and software items to be reused. This paper presents a proposed model of resources requirements on the basis of ISO standards for measuring the software resources product quality; whether the software it has already been delivered or has yet to be built**.

Keywords—Resources Requirements, Software Product Quality, ISO Standards

I. INTRODUCTION

Resource collection and task scheme are fundamental function in software system environments, for instance, the cloud computing tasks try to win system resources. The choices made by the parallel algorithms ought to be judged based not only on measurements related to customer satisfaction, such as the proportion of tasks hand out without affecting their quality requirements, but also stands on resources-related performance measurements, such as the number of resources used to hand out the tasks and their exploitation competence.

Developers of software products are in charge for identifying the requirements of any product, developing software that put into practices the requirements, and for allocating suitable resources such as processors and communication networks. Improvement such quality software systems has challenge for software developers. In practice, identifying the non-functional resources requirements are often captured at high level while the focusing only on the functionality of the system [1-7]. Several products have failed because of neglect of such non-functional requirements

Resources Requirements describes what the component needs from its environment to perform its function and define the limits of software budgets associated with computer resources such as: CPU load and maximum memory size to be considered by the supplier as well as [8] and [9] Indicates to computer hardware resource requirements on the utilization (e.g. processor capacity and memory capacity) available for the software item (e.g. sizing and timing). Moreover indicates to Computer software resource requirements on the software items to be used by or incorporated into the system (or constituent software product) (e.g. a specific real time operating system).

Resources requirements are considered as an important part in the software life cycle to assure from the suitability and for the availability of resources to implement it by all functions involved in its application [1-7]. Consequently ISO standards [8] describe the resources requirements as the capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions.

This paper will report the design measurement method to identify of the software resources based on international standards as an autonomous method to identify the size of the software resources independently of the software languages types, which avoids the weaknesses observed in the resources measures currently available.

The paper scope is to identify separately the all functionality allocated to software resources 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 a standard based model of software resources requirements. The proposed generic model is considered as kind of a ‘reference model’ in the sense of an ‘etalon’ standard that is being used for the measurement of resources**.**

This paper is organized as follows. Section 2 presents the related works. Section 3 presents design the measurements method for resources requirements as defined in ISO. Section 4 presents design a Meta model of resources requirements. Section 5 presents Design numerical rules of software resources requirements. Discussion and a conclusion are presented in section 6.

F. A. Kenza Meridji is now with the University of Petra, Collage of Information Technology, Department of Software Engineering, 11196 Amman, Jordan. (e-mail: kmeridji@uop.edu.jo).

S. B. Khalid T. Al-Sarayreh is now with the Hashemite University, Prince Hussein Bin Abdullah II for Information Technology, Department of Software Engineering, 13115 Zarqa, Jordan. (e-mail: khalidt@hu.edu.jo).

S. C. Tatiana Balikhina is now with the University of Petra, Collage of Information Technology, Department of Computer Science, 11196 Amman, Jordan. (e-mail: tbalikhina@uop.edu.jo).

II. Related Work

Much of the work done up to date on resource non functional requirements was considering resources in general without dealing with them in detail. For instance a model in [10] suggest the assignment of tasks to resources to be able to reduce the problems related the tasks' time requirements at the same time increasing the resources' utilization efficiency for a given number of resources. The proposed method takes concepts derived from graph partitioning, and collects tasks together to be able to reduce the overlapping time of the task that is assigned to a given resource and to be able to increase the time overlapping with tasks assigned to dissimilar resources [10].

Furthermore, [11] outlined five steps concerning resources quality requirements defined by equipment resources consumption, function and structure environmental impacts. The five steps are described and encourage the following first, to decrease the dependence of equipment on non-sustainable resources: this will allow decreasing the non-sustainable resource usage and consumption; in addition it will allow utilizing vigorously the sustainable resource as alternative supply. Second, is to decrease wastes and to improve the use of resources efficiently. Third, renew and modernize equipments, to be able to use entirely the equipment potential and to effectively decrease the retired equipment. Fourth, reuses and retrieve the equipment resources, to be able to improve the recycling of resources. Fifth, decrease environment damage.

While in [12] the authors proposed a model to tackle the Stake Cloud community platform. This model has ability to work as a cloud resources marketplace. By permitting the users to input their resource needs and give them the matching cloud services.

Moreover, [13] defined eleven steps by conducting an empirical study on the role that requirements and resources play in the building a software product quality. This will allow observing and defining how software quality is constructed in software development organizations. Therefore eleven software programmers, testers, quality control personnel, requirement managers and research and development personnel were interviewed and common practices of quality construction were analyzed. The result showed that quality construction practices differ significantly among different organizations. Differences were mentioned about and the degree of involvement of the customer in the software development, methods used for requirements elicitation, and objectives of software testing.

For instance, [14] studied the impact of non-functional requirements on requirements evolution; this paper listed and analyzed different approaches, available in the literature related to non-functional requirements during software development. This paper focused on three issues: Different views on non-functional requirements, Representation of non-functional requirements and how to deal with non-functional requirements.

Whilst [15] proposed a definition and a discussion of the most used agile software development methods and they investigate the software SMEs challenges and for comparison purposes formulate it into criteria. In addition these methods were compared against the defined criteria and as a result their similarities and differences were outlined.

Finally, [16] introduced a datacenter resources integrated provisioning (DRIP) architecture using synchronized virtualization of distributed datacenters and operate multi-domain software defined optical networks. The DRIP architecture objective is to achieve the integration and allocation of IT resources and optical network resources. In order to examine the feasibility and efficiency of the anticipated architecture, two IT resources allocation strategies and two virtual networks composition strategies are evaluated [17] and [18].

The motivation of this research paper is to contribute to better define, describe and measure some of the NFR inputs required for the adequate *a priori* cost estimation of software projects. The measurement scope in this paper is to identify separately all functionalities allocated to software resources requirements for software product quality.

The focus of this paper is on a single type of NFR that is, resources requirements. This paper reports on the work carried out to define an integrated view of software functional user requirements for resources requirements for the software product on the basis of ISO international standards.

III. Design Measurement Method of Resources Requirements as Defined in ISO

Based on the resources requirements definitions stated by ISO standards the design measures steps for resources requirements as follows:

A. *Determination of measurement objectives for Software Product*

This section illustrates the measurement objectives of resources requirements as a piece of a software product quality, followed by the measurement point of view and the intended uses of the measurement results.

1) **The objective**: is to measure the size of the resources requirements as defined in ISO.
2) **Measurement point of view**: Software perspective.
3) **Intended uses of the measurement results**: throughout the software life cycle: the size of the resources for a software product, whether it has yet to be built or it has already been delivered.

B. *Characterization Resources Concepts to Measured*

This section illustrates the resources requirements concepts and the identified resources to be measured

1) **Definition of the concept to be measured:** the resources measurements can be internal or external. The proposed measurement method is to be applicable for non-embedded software resources.

   ***External resources Measures*:** should be able to measure such attributes as the utilised resources behaviour of computer system including software during testing or operating and can be measured based on the following resource utilization: (I/O resource measurements, Memory resource measurements and Transmission resource measurements).

***Internal resources Measures*:** indicate a set of attributes for predicting the utilization of hardware resources by the computer system including the software product during testing or operating.

2) **The Resource entities to be measured**

- External Resources Entities
  1) I/O resource measurements
     - I/O Devices Utilization
     - User Waiting Time of I/O Devices Utilization
  2) Memory resource measurements
     - Memory Utilization
  3) Transmission resource measurements.
     - Maximum Transmission Utilization
     - Transmission Capacity Utilization
     - Media Device Utilization
- Internal Resources Entities
  1) I/O Related Errors
  2) I/O loading

IV. Design A Meta Model of Resources Requirements

This section presents the meta model of the software resources requirements on the basis of the previous section.

A. *I/O devices resources*

In the following design of the Meta models- see Figure 1:

- Entity type 1 can be used to measure the e external software resources throughout executing concurrently a large number of tasks and record I/O device utilization for one functional process.
- Entity type 2 can be used to measure the 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 for one functional process.
- Entity type 3 can be used to measure the internal software resources throughout calibrating the test condition to define maximum I/O loading for one functional process.
- Entity type 4 can be used to measure the external software resources throughout run the application of record of errors due to I/O failures and warning for one functional process.

B. *Memory resources*

In the following design of the Meta models- see Figure 2:

- Entity type 5 can be 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.

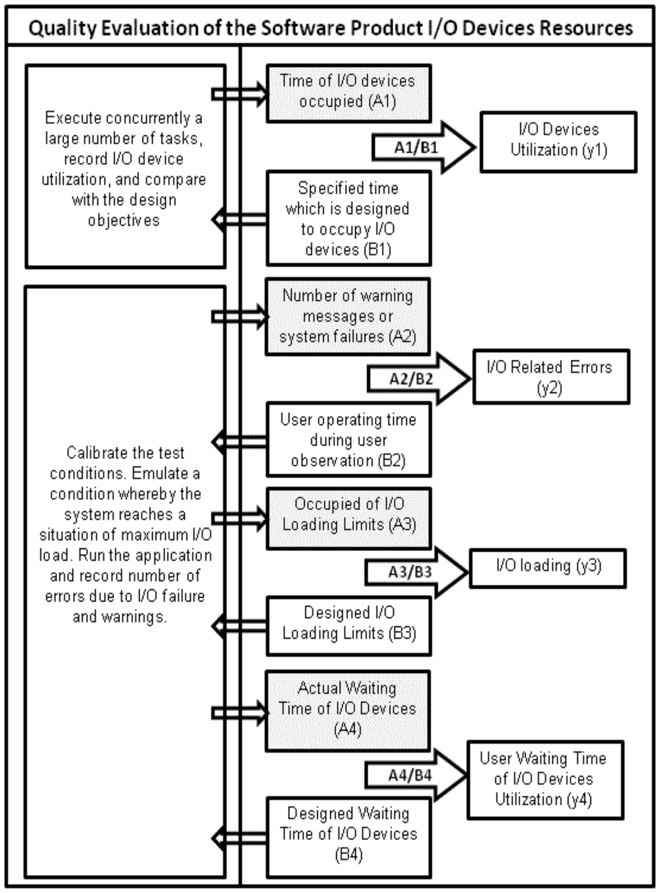
Quality Evaluation of the Software Product I/O Devices Resources
Execute concurrently a large number of tasks, record I/O device utilization, and compare with the design objectives
Time of I/O devices occupied (A1)
A1/B1
I/O Devices Utilization (y1)
Specified time which is designed to occupy I/O devices (B1)
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 warnings.
Number of warning messages or system failures (A2)
A2/B2
I/O Related Errors (y2)
User operating time during user observation (B2)
Occupied of I/O Loading Limits (A3)
A3/B3
I/O loading (y3)
Designed I/O Loading Limits (B3)
Actual Waiting Time of I/O Devices (A4)
A4/B4
User Waiting Time of I/O Devices Utilization (y4)
Designed Waiting Time of I/O Devices (B4)

Fig. 1: I/O Devices Resources Meta Model

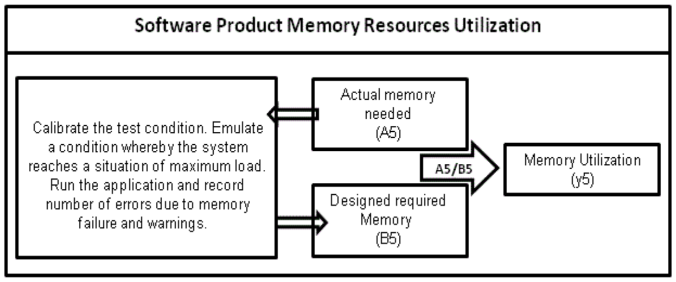
Software Product Memory Resources Utilization
Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.
Actual memory needed (A5)
A5/B5
Memory Utilization (y5)
Designed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

Fig. 2: I/O Memory Resources Meta Model

C. *Transmission resources*

In the following design of the Meta models- see Figure 3:

- Entity type 6 can be used to measure the external software resources throughout evaluate what is required for the system to reach a situation of maximum load for one functional process.
- Entity type 7 can be used to measure the external software resources throughout observe transmission capacity and compare specified one for one functional process.
- Entity type 8 can be used to measure the external software resources throughout execute concurrently specified tasks with multiple user for one functional process.

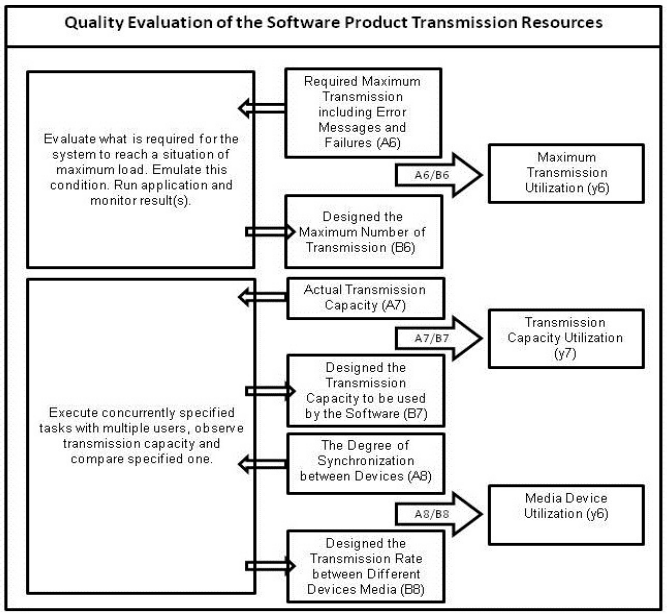
Quality Evaluation of the Software Product Transmission Resources
Evaluate what is required for the system to reach a situation of maximum load. Emulate this condition. Run application and monitor result(s).
Required Maximum Transmission including Error Messages and Failures (A6)
A6/B6
Maximum Transmission Utilization (y6)
Designed the Maximum Number of Transmission (B6)
Actual Transmission Capacity (A7)
A7/B7
Transmission Capacity Utilization (y7)
Designed the Transmission Capacity to be used by the Software (B7)
Execute concurrently specified tasks with multiple users, observe transmission capacity and compare specified one.
The Degree of Synchronization between Devices (A8)
A8/B8
Media Device Utilization (y6)
Designed the Transmission Rate between Different Devices Media (B8)

Fig. 3: Transmission Resources Meta Model

V. NUMERICAL ASSIGNMENT ROLES FOR RESOURCES REQUIREMENTS

In this section the basis for these numerical assignment rules are the proposed meta-models as defined in Figures 1, 2 and 3. Numerical assignments rules can be described through a descriptive text or what so called (a practitioner's description) or through using mathematical expressions or as defined as (a formal theoretical viewpoint). In this paper the numerical assignments rules are built based on mathematical expressions while the descriptive text rules are defined in ISO.

A. Identification of data Resources Groups

This section illustrates data resources groups as defined in ISO (i.e. Input and output) form sources and/or to data destinations for Resources Requirements for details see-table 1 and table 2.

Table 1 : Resources Requirements Data Sources

Categories	Data Sources	Objects of Interest
I/O Devices Resources	• Specified time which is designed to occupy I/O devices. • Actual time of I/O devices occupied. • User operating time during user observation. • The number of warning messages or system failures. • Designed I/O loading limits. • The occupied of I/O loading limits • Designed waiting time of I/O devices. • The actual waiting time of I/O devices.	• Time • Time • Number • Number • Loading limit • Loading limit • Time • Time

Table 1 : Resources Requirements Data Sources (Contd)

Categories	Data Sources	Objects of Interest
Memory Resources	• Designed required memory. • The actual memory needed	• Size • Size
Transmission Resources	• Designed maximum number of transmission. • The required maximum transmission including error messages and failures. • The designed the transmission capacity. • The actual transmission capacity. • The design of the transmission rate between different devices media. • The degree of synchronization between devices.	• Transmission no. • Transmission no. • Transmission Capacity • Transmission Capacity • Transmission Rate • 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

B. Numerical Roles for Resources

Regards to three meta models, resources requirements (internally and externally) for **I/O devices resources**: measures the executing concurrently the tasks, record maximum I/O loading and errors due I/O failures., **Memory resources**: measures the memory maximum load and the errors due to memory failures and warnings and **Transmission resources**: measures maximum transmission, transmission capacity and synchronization between media devices.

1) Mathematical Rules of Resources I/O device:

This section presents mathematical assignments rules for resources I/O devices follows:

- The measurement size of the I/O device (externally) for one process

  ∑ Data Movement (Data Recourses Group)

  ∑ (I/O Devices Utilization + User Waiting Time of I/O Devices).

- The measurement size of the I/O devices (internally) for one process.

∑ Data Movement (Data Resource Group)

∑ (I/O related errors + I/O Loading).

- The measurement size of the I/O devices (internally and externally)

(∑ (I/O Devices Utilization + User Waiting Time of I/O Devices) + ∑ (I/O related errors + I/O Loading))

- The **T**otal Measurement **S**ize of the I/O devices [for the all functional processes]

N x (∑ (I/O Devices Utilization + User Waiting Time of I/O Devices) +∑ (I/O related errors + I/O Loading))

N: number of functional processes for the I/O devices.

2) *Mathematical Rules for Memory Resources:*

- The measurement size of the memory resources (externally) for one process

∑ Data Movement (Data Resource Group)

∑ (memory Utilization).

Note: There are no internal measures as defined in ISO.

- The **T**otal **S**ize of the memory resources [for the all functional processes]

N x ∑ (memory resources)

N: number of functional processes for the memory resources.

3) *Mathematical Rules of Transmission Resources:*

- The measurement size of the transmission resources (externally) for one process

∑ (Data Movement (Data Resource Group)

∑ (Maximum Transmission + Transmission Capacity + Media Devices Utilization)

- The **T**otal **F**unctional **S**ize of the transmission resources [for the all functional processes]

N x ∑ (Maximum Transmission + Transmission Capacity + Media Devices Utilization)

N: number of functional processes for the transmission resources.

4) *Total measurement Size of Resources Requirements*

The **T**otal **M**easurement **S**ize of the resources [for the all functional processes]

N x (∑ (I/O Devices Utilization + User Waiting Time of I/O Devices) +∑ (I/O related errors + I/O Loading))

+

N x ∑ (memory resources)

+

N x ∑ (Maximum Transmission + Transmission Capacity + Media Devices Utilization)

N: number of functional processes for the resources.

VI. Conclusion

This paper introduced a new design method to measure the resources non-functional requirements internally and externally a . As well as proposed three meta models for resources requirement as defined in ISO 9126 standards independently of the software type or languages used .

Moreover, the design of the measurement method is defined to specify the strategy of the measurement rules. This will allow performing the mapping between the concepts of ISO 19761 and the concepts of the suggesting design of the generic resources Meta models and rules and then identification of the data movements and the performance of the measurement process.

It is important to mention that the design of the measurement procedure for resources requirements for the software product quality have been developed to apply the ISO 19761 and to apply a measurement method to the resources requirements in order to obtain the functional size of the resources as a separate piece of a software in early stages of the software development process.

The future work will concentrate on experimental test for the proposed resources model, comparison this model with the previous ones and list all the strength and weaknesses after experiment process as well as mapping our proposed model with the definitions of standard Etalon.

References

[1] Abran, A., K. T. Al-Sarayreh, and J. J. Cuadrado-Gallego, " A Standards-based Reference Framework for System Portability Requirements", Computer Standards and Interface, Elsevier, 2013. http://dx.doi.org/10.1016/j.csi.2012.11.003

[2] Al-Sarayreh, K. T., A. Abran and and J. J. Cuadrado-Gallego," A Standards-based model of system maintainability requirements", Journal of Software: Evolution and Process, John Wiley & Sons, Ltd, 2013. http://dx.doi.org/10.1002/smr.1553

[3] Meridji, Kenza, Khalid T. Al-Sarayreh, and Ahmad Al-Khasawneh. "A generic model for the specification of software reliability requirements and measurement of their functional size." *International Journal of Information Quality* 3, no. 2 (2013): 139-163.

[4] Al-Sarayreh, Khalid T., Ibrahim Al-Oqily, and Kenza Meridji. "A standard-based reference framework for system operations

requirements." *International Journal of Computer Applications in Technology* 47, no. 4 (2013): 351-363.

[5] Al-Sarayreh, Khalid T., Ibrahim Al-Oqily, and Kenza Meridji. "A standard based reference framework for system adaptation and installation requirements." In *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2012 6th International Conference on*, pp. 7-12. IEEE, 2012.

[6] Al-Sarayreh, Khalid T., Kenza Meridji, Ebaa Fayyoumi, and Sahar Idwan. "A Novel Approach to Build a Generic Model of Photovoltaic Solar System Using Sound Biometric Techniques." *International Journal of Information Technology and Web Engineering (IJITWE)* 9, no. 1 (2014): 31-44.

[7] K. T. Al-Sarayreh and A. Abran, "A Generic Model for the Specification of Software Interface Requirements and Measurement of Their Functional Size," 8th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2010, Montreal, Canada, pp. 217-222, 2010.

[8] ISO/IEC-9126, "Software Engineering - Product Quality - Parts 1-4: Quality Model ", International Organization for Standardization, Geneva (Switzerland), 2008.

[9] ISO/IEC-19761, "Software Engineering - COSMIC v 3.1 - A Functional Size Measurement Method", International Organization for Standardization, Geneva (Switzerland), 2011

[10] Doulamis, N.D.; Kokkinos, P.; Varvarigos, E., "Resource Selection for Tasks with Time Requirements Using Spectral Clustering," *Computers, IEEE Transactions on* , vol.63, no.2, pp.461,474, Feb. 2014

[11] Shilun Liu; Mingfang Ni; Kaikai Hu; Ma Yu, "Study on the main requirements of equipment resource-ability design," *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on* , vol., no., pp.709,713, 17-19 June 2011

[12] Todoran, I., "StakeCloud: Stakeholder requirements communication and resource identification in the cloud," *Requirements Engineering Conference (RE), 2012 20th IEEE International* , vol., no., pp.353,356, 24-28 Sept. 2012

[13] Seth, F.P.; Mustonen-Ollila, E.; Taipale, O.; Smolander, K., "Software Quality Construction: Empirical Study on the Role of Requirements, Stakeholders and Resources," *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific* , vol.2, no., pp.17,26, 4-7 Dec. 2012

[14] Khatter, K.; Kalia, A., "Impact of Non-functional Requirements on Requirements Evolution," *Emerging Trends in Engineering and Technology (ICETET), 2013 6th International Conference on* , vol., no., pp.61,68, 16-18 Dec. 2013

[15] Hamed, A.M.M.; Abushama, H., "Popular agile approaches in software development: Review and analysis," *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on* , vol., no., pp.160,166, 26-28 Aug. 2013

[16] Chen, H.; Zhang, J.; Zhao, Y.; Deng, J.; Wang, W.; He, R.; Yu, X.; Ji, Y.; Zheng, H.; Lin, Y.; Yang, H., "Experimental Demonstration of Datacenter Resources Integrated Provisioning over Multi-Domain Software Defined Optical Networks," *Lightwave Technology, Journal of* , vol.PP, no.99, pp.1,1. 2015.

[17] Penzenstadler, B.; Raturi, A.; Richardson, D.; Tomlinson, B., "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century," *Software, IEEE* , vol.31, no.3, pp.40,47, May-June 2014

[18] Dal Bianco, V.; Myllarniemi, V.; Komssi, M.; Raatikainen, M., "The Role of Platform Boundary Resources in Software Ecosystems: A Case Study," *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on* , vol., no., pp.11,20, 7-11 April 2014