

TOWARDS A SOFTWARE PRODUCT SUSTAINABILITY MODEL

Coral Calero¹, M^a Ángeles Moraga¹, Manuel F. Bertoa²

¹*Instituto de Tecnologías y Sistemas de la Información*

University of Castilla-La Mancha. Spain

Coral.Calero@uclm.es; MariaAngeles.Moraga@uclm.es

²*Departamento de Lenguajes y Ciencias de la Computación*

University of Malaga

Malaga, Spain

bertoa@lcc.uma.es

1. Introduction

The need to adapt current products and services to an environmentally friendly means of working is already a social and economic demand. Although GreenIT can be considered as a mature discipline, software sustainability in both its process and use has only become a topic of interest in the last few years. In this respect, we believe that it is fundamental to define what we consider software sustainability to be, and how to evaluate it properly.

Sustainable software development refers to a mode of software development in which the aim of resource use is to meet product software needs while ensuring the sustainability of natural systems and the environment, and the Sustainability of a software product can be defined as the capacity to develop a software product in a sustainable manner (Calero et al., 2013).

As remarked upon in Calero et al. (2013), software sustainability is a means to improve a software product, this being part of its quality and related to non-functional requirements (requirements that constrain or set certain quality attributes on functionalities, Glinz, 2007).

2. Modeling software product sustainability

As previously mentioned, our starting point is that sustainability is part of a software product's quality. ISO/IEC 25000 is a series of standards that are specific to System and software product Quality Requirements and Evaluation, namely SQuaRE(ISO/IEC 25000, 2010). Three quality models are defined in the ISO/IEC 25010 division (ISO/IEC 25010 (2010)). The targets of the quality models and their related entities are shown in Fig. 1.

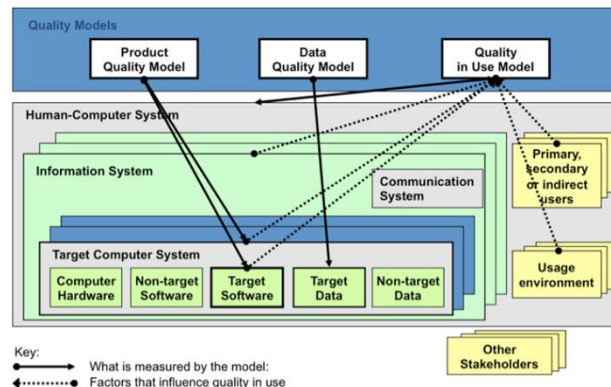


Fig. 1. Targets of quality model

In this figure it is possible to observe that product quality is reflected in the Product quality model which is related to the target software. In the standard, the product

quality model is composed of eight characteristics, each of which is subdivided into several sub-characteristics (see Fig. 2).

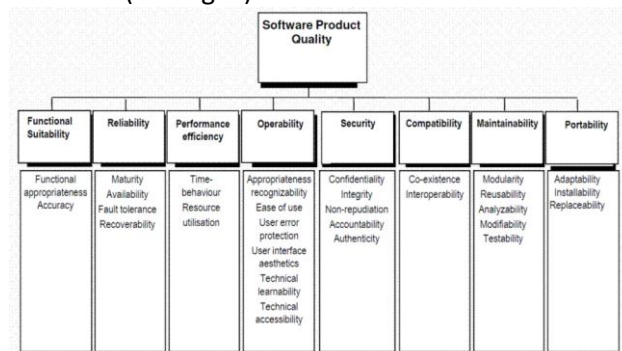


Fig. 2. Software product quality model in ISO/IEC 25010

As is stated in the standard, “the product quality model is useful for specifying requirements, establishing measures, and performing quality evaluations. The quality characteristics defined can be used as a checklist in order to ensure a comprehensive treatment of quality requirements, thus providing a basis that can be used to estimate the consequent effort and activities that will be needed during systems development. The characteristics in the product quality model are intended to be used as a set when specifying or evaluating software product quality”.

This signifies that if we wish to consider sustainability as part of the quality model, it is necessary to define it together with its sub-characteristics.

To do this, it is necessary to take into account that when a software product is being developed, its sustainability can be considered from two points of view.

First, we must ensure that the software product is energy-efficient when it works, and that the resources are used in the most appropriate manner. Previously, we have proposed the following sub-characteristics for sustainability, which would fit with this "short-term" point of view:

- **Energy consumption.** Degree to which the amounts of energy used by a software product when performing its functions meet requirements.
- **Resource optimization.** Degree to which the amounts and types of resources used by a product when performing its functions meet sustainability requirements. As in the standard, the authors consider that resources can include: other software products, the software and hardware configuration of the system, and materials (e.g. print paper, storage media).

Furthermore, we must ensure that the software product will endure over time, with its replacement only being required if adapting it to the new circumstances is very difficult to achieve. This is referred to as **Perdurability**.

The idea of making a piece of software perdurable is that of achieving a software product that is long-lasting, modifiable, reusable, i.e. those aspects that make the software developed last over time, and is able to adapt to change without losing its functionality or other features related to its quality.

In order to define perdurability it is first necessary to identify what must be considered and, this will be done by using the standard (ISO:25010, 2010). This standard contains 3 characteristics that could be related to what we are seeking:

Reliability. Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time

- **Maturity.** Degree to which a system meets needs for reliability under normal operation

- **Availability.** Degree to which a system, product or component is operational and accessible when required for use
- **Fault tolerance.** Degree to which a system, product or component operates as intended despite the presence of hardware or software faults
- **Recoverability.** Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

Maintainability. Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

- **Modularity.** Degree to which a system or computer program is composed of discrete components such that a change to one component has a minimal impact on other components
- **Reusability.** Degree to which an asset can be used in more than one system, or in building other assets
- **Analysability.** Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified
- **Modifiability.** Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
- **Testability.** Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

Portability. Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

- **Adaptability.** Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
- **Installability.** Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment
- **Replaceability.** Degree to which a product can be replaced by another specified software product for the same purpose in the same environment

However, if these sub-characteristics and their definitions are viewed in detail, many of them can be discarded since they are not related to long-term issues (those that we are seeking). As a result, we obtain a final set of characteristics that could be related to perdurability. These characteristics must be taken into account when defining the Perdurability characteristic: **Reusability, Modifiability and Adaptability.**

These characteristics will not be used to define perdurability exactly as they are defined, but will be used as a basis for doing so, and by taking into account the objective of perdurability.

It is therefore possible to define **perdurability** as the degree to which a software product can be modified, adapted and reused in order to perform specified functions under specified conditions over a long period of time.

Having identified and defined the sub-characteristics of sustainability from both the points of view described, we shall now add a new characteristic for the sustainability defined to the software product quality model proposed by the ISO/IEC 25010 (2010), as shown in Figure 3:

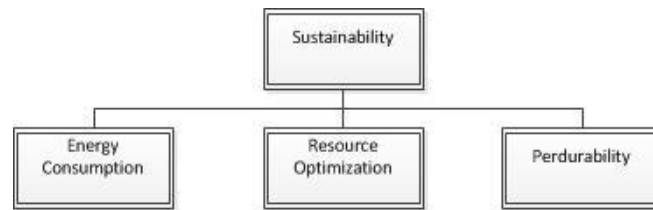


Fig. 3. The sustainability characteristic

3. Conclusions

In this paper we have presented the importance of software sustainability. We have used the assumption that sustainability is part of the quality of a software product as a starting point to present a specific model for software sustainability that can be added to the software product quality model from the ISO/IEC 25010 (2010) standard.

The definition of software sustainability and its characteristics will make it possible to incorporate sustainability into the development of a software product in the form of non-functional requirements and to ensure that the final products are environmentally friendly.

The definition of the sustainability characteristic will also make it possible to define measures and indicators for the sustainability of a software product, which it will be possible to use to evaluate, to detect weaknesses or to improve the sustainability of a software product. These are our future lines of work and research.

4. Acknowledgment

This work is part of the GEODAS-BC (TIN2012-37493-C03-01), the PEGASO (TIN2009-13718-C02-01) and Sistemas Inalámbricos de Gestión de Información Crítica (TIN2011-23795) projects funded by the Spanish Ministerio de Economía y Competitividad and by FEDER (Fondo Europeo de Desarrollo Regional).

5. Referencias

Calero, C., Bertoa, M.F. and Moraga, M.A. (2013). Sustainability and Quality: icing on the cake. Second International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) in RE'13 (July 15th-19th).CEUR-WS.org/Vol. 995. ISSN: 1613-0073. Paper 5.

Glinz, M. On non-functional requirements. in International Conference on Requirements Engineering. 2007.

ISO/IEC 25010, Systems and software engineering - Software product Quality Requirements and Evaluation (SQuARE) - Software product quality and system quality in use models.ISO. 2010.

ISO/IEC 25000, Systems and software engineering - Software product Quality Requirements and Evaluation SQuARE. 2010.