Applying the ISO/IEC 25010 Quality Models to Software Product

John Estdale[1(✉)] and Elli Georgiadou[2]

[1] IT Architecture Specialists Ltd, Newbury, UK
john.estdale@acm.org
[2] Faculty of Science and Technology, Middlesex University, London, UK
e.georgiadou@mdx.ac.uk

Abstract. The software development process focuses on the delivery of a software implementation – its 'product', whether COTS or bespoke. However, potential acquirers are attracted by the promise of a future 'service' from that product: those aspects of the software's behavior visible outside it, particularly those that deliver value in the real world.

ISO/IEC 25010: 2011 provides the leading models for assessing software product. This is an important contribution towards establishing the delivery performance of software processes and proposed improvements. This paper explores the scope and interpretation of the ISO/IEC 25010 quality models, in the light of this broad, lifetime service-oriented view, also identifying other significant aspects of product that concern acquirers of software, and for which quality requirements and quality evaluation are potentially needed. Suggestions for refinement and extension of the standard complete the paper.

Keywords: Software product quality · Software behavior · Digital service
ISO/IEC 25010 · SQuaRE · Quality models · 7Ms

1 Introduction

The value of software to users and organizations arises from its actual behavior in use, rather than from any qualities of the source code or the intended behavior as described in design documents etc. As Hofemann [1] says:

> "It requires a change in the mindset to consider software as a service rather than as a product. It is more than a change in business or delivery model, as in the case of changing to SaaS."

Furthermore, most applications of significant value will go through a series of maintenance releases, thus the simple model of software development producing software product (e.g. Sjøberg [2]) must be extended to cover post-delivery support and maintenance. In considering Software Process Improvement, we need to include post-delivery activities.

ISO/IEC 25010: 2011 *Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models* [3] presents the leading quality models for software products and computer systems known as 'software-intensive computer systems' [4]. This paper examines the two quality models in this standard in

X. Larrucea et al. (Eds.): EuroSPI 2018, CCIS 896, pp. 492–503, 2018.
https://doi.org/10.1007/978-3-319-97925-0_42

Sects. 4 and 5, with some additional aspects in Sect. 6. The standard mentions "secondary users who provide support" but the quality of such advice and support is not assessed in either model, so the In-life experience is considered here in Sect. 7. Section 8 draws the conclusions.

The present authors have previously applied ISO/IEC 25010 to smartphone applications destined for the Apple and Microsoft App Stores [4], and to the differences between products and prototypes [5]. These provide practical trials of the application of ISO/IEC 25010. This paper considers Digital Services from delivered software and comments on some less obvious issues with ISO/IEC 25010 interpretation.

2 ISO/IEC 250xx Series: SQuaRE

The ISO/IEC 25000 to ISO/IEC 25099 series of International Standards is entitled *Systems and software engineering – Systems and software Quality Requirements and Evaluation*, hence the acronym: 'SQuaRE'. The guide to the series, now in its 2nd edition [6] states that "the general goal … was to … [cover] two main processes: software quality requirements specification and system and software quality evaluation; supported by a system and software quality measurement process. The purpose … is to assist those developing and acquiring systems and software products with the specification and evaluation of quality requirements".

The traditional ISO 9001 position was that quality concerned "conformance to specified requirements". This has been broadened to "satisfy stated and implied needs". As the universe of such needs is not well-defined and standardized, evaluation of quality is ultimately purchaser-dependent.

2.1 The ISO/IEC 25010: 2011 Quality Model

SQuaRE has simplified the analysis from its predecessor, ISO 9126 [7], and now divides software quality characteristics into two quality models. Quality in Use is "the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals… in specific contexts of use" and the catchall Product Quality is "characteristics … that relate to static properties of software and dynamic properties of the computer system". Given the previously quoted focus on needs, one might ask why include the second group at all, but ISO/IEC 25000 explains this as providing targets to drive development and verification, and to predict Quality in Use before delivery [6]. For SPI, it's helpful to have these supplier measures and exclude the impact of users and their context.

SQuaRE includes a Technical Specification: ISO/IEC TS 25011:2017 *Information technology—Systems and software Quality Requirements and Evaluation (SQuaRE)—Service quality models* [8]. This focusses on services that "make use of IT systems as tools to provide value", so includes a range of service management issues not directly relevant in this context.

Italics are used throughout this paper to denote the 13 characteristics and 40 subcharacteristics defined in ISO/IEC 25010. Definitions quoted are all taken from ISO/IEC 25010 [3].

Table 1. ISO/IEC 25010: 2011: quality models, characteristics and subcharacteristics (These are taken direct from the standard, following its English spelling, but American English is used for the remainder of this paper)

Quality in use	Product quality	Product quality (cont.)
• Effectiveness • Efficiency • Satisfaction – Usefulness – Trust – Pleasure – Comfort • Freedom from risk – Economic risk mitigation – Health and safety risk mitigation – Environmental risk mitigation • Context coverage – Context completeness – Flexibility	• Functional suitability – Functional completeness – Functional correctness – Functional appropriateness • Performance efficiency – Time behaviour – Resource utilization – Capacity • Compatibility – Co-existence – Interoperability • Usability – Appropriateness recognisability – Learnability – Operability – User error protection – User interface aesthetics – Accessibility	• Reliability – Maturity – Availability – Fault tolerance – Recoverability • Security – Confidentiality – Integrity – Non-repudiation – Accountability – Authenticity • Maintainability – Modularity – Reusability – Analysability – Modifiability – Testability • Portability – Adaptability – Installability – Replaceability

3 Applying the ISO/IEC 25010: 2011 Quality Models

Although ISO 9126 has been much used as a quality model in academic papers, many alternative quality models have been published over the years [9–11]. Oriol et al. [12] compared 47 quality models for web services from 65 papers with ISO/IEC 25010 and found little consistency. Unlike the physical world, with clearly independent dimensions and well-defined measures (length, mass, time, electric current, thermodynamic temperature, luminous intensity, etc.), software quality concepts such as compatibility are somewhat nebulous, and indeed have been redefined and reorganized as part of the ISO/IEC 25010 work [3].

Biscoglio and Marchetti [13] found similar difficulties in applying ISO/IEC 25000, which they described as “a conceptual framework and not a ready-to-use solution”.

3.1 ISO/IEC 25010 Applied to Digital Services

Behavior is what the software does as it is executed, i.e. it is the observable aspects of a computerized process. To identify all the causes of variation in the behavior of a process, one can apply the 7Ms model from quality management [14] shown in Table 2 to ensure completeness.

Table 2. Process variation across digital services

'M'	Normal interpretation in manufacturing context	Digital service equivalent
Machine	The equipment and technology required	The platform (operational environment), with the software application installed on it
Method	How the process is performed by the human element: policies, procedures etc.	Limited in this context to supporting processes in Operations and Support
Material	Raw materials and other ingredients	Requests and their parameters from the service consumer (human or app)
Man (power)	People involved in providing the service	Humans are not involved in servicing most requests, but are needed to keep the service running longer term, e.g. Operations and Support staff
Management	The organization of the manpower	The Digital Service Manager
Milieu	Environmental conditions	Existing data, infrastructure services and other apps sharing the platform
Measurement	Process measurements used to monitor and control the process	These are not further defined, so the ISO/IEC 25010 characteristics will be used

4 Quality in Use

Sections 4 and 5 follow the structure of the standard shown in Table 1, looking at each defined characteristic in turn, and commenting on how it may be interpreted, going down to subcharacteristic level where appropriate.

The definition of Quality in Use quoted in Sect. 2.1 is concerned with measuring the service to be delivered, so follows the service emphasis of this paper. In a traditional procurement, Quality in Use would be measured in the Buyer's environment as part of Acceptance Test.

4.1 Achievement of Needs

ISO/IEC 25010 does not include the evaluation of an application's specific functionality and features, as their value can only be measured in relation to the needs of the acquiring organization. The Quality in Use model therefore seeks to quantify the 'usability' (*effectiveness*, *efficiency* and *satisfaction*) of the application, when specific users attempt to meet their specified goals. ISO/IEC 25010 defines *effectiveness* as the "accuracy and completeness with which users achieve specified goals", and *efficiency* as "resources expended in relation to the accuracy and completeness with which users achieve goals".

As an example of the sort of interpretation of the standard that may be needed to apply it, much of the content of App Stores consists of games, or items for

entertainment. Apple stated the need as: "If your App doesn't provide some form of lasting entertainment value, or is just plain creepy, it may not be accepted." [15] However, Apple does attempt to restrict the silly, witness their statement: "We don't need any more Fart apps" [16], (which raises the question of how they determine whether a new one provides a significant advantage over the many already in store!). This is quite a stretch from SQuaRE's solemn discussion of "stated and implied needs".

4.2 Freedom from Risk

As explained in ISO/IEC 25010, compliance was dropped in the move from ISO/IEC 9126-1:2001, as compliance with laws and regulations should be stated customer requirements, presumably 'Musts', not quantifiable qualities.

In complex industries, such as telecommunications it is known that legal and regulatory requirements, developed at different times, by different authorities, often conflict. More generally, many customers are likely to have no knowledge of the potential constraints on functionality and features of issues such as Privacy, Data Protection, the Distance Selling Regulations or the contractual rules of the five main Credit Card schemes for different industries and scenarios. Product suppliers should take responsibility for making themselves aware of all restrictions on the use of their product in the sales territory, and make explicit any usage restrictions and assumptions.

Whilst we may assume that acquirers state a need for overall legal compliance in their operating territories, it is likely that this will not be developed in detail, leaving the implications to be explored during product evaluation. We propose that Compliance be reinstated, this time as Legal Compliance under the *Freedom from risk* characteristic, to cover the product's claim for Legal, Regulatory and Contractual compliance. An associated quality measure might be "Clarity of Obligations to be met by the Customer to achieve Legal Compliance, when adopting this system", leaving the customer to decide whether the issues reported by the Supplier are acceptable under *functional suitability*.

5 Product Quality

These characteristics are primarily of interest to the supplier, and to those acquirers who wish to get more involved technically.

Functional suitability is an early view of the product's likely effectiveness, discussed in Sect. 4.1. ISO/IEC 25010 divides it into the subcharacteristics *functional completeness*, *appropriateness* and *correctness*. It is left to potential acquirers to do their own assessment and selection, based on their particular needs.

Performance efficiency includes *time behavior, resource utilization* and *capacity*. For software products, measures taken during development may not use the acquirer's exact platform/environment, so extrapolation to the delivered service may be difficult.

5.1 Compatibility

This is divided into *interoperability* between applications – the exchange and use of information, and *co-existence* – the impact on other products sharing the same platform. Note that these are concerned with how functionality is implemented, so are perhaps best thought of as 'features' not functions.

Co-existence is not a purely passive concept. Even after the arrival of properly protected multi-programming operating systems on PCs, it remains important that a product be 'well-behaved' rather than 'misbehaved' [17, 18], so should use supplied services for co-operative sharing of resources with the other (uncontrolled) applications running on the platform [5].

Interoperability is defined as "the degree to which two or more systems, products or components can exchange information and use the information that has been exchanged". Even if the product has no advertised interfaces to other products, in practice, users will assume interoperability with the native operating system capabilities, such as cut/copy/paste text between fields on a display screen, and objects between application windows [5].

5.2 Usability

Before the product is released, enabling Quality in Use to be measured as an emergent property (see Sect. 4.1), *usability* can be measured by relevant product subcharacteristics: *appropriateness recognizability*, *learnability*, *operability*, *user error protection*, *user interface aesthetics*, and *accessibility*. *Appropriateness recognizability* is described as "the degree to which users can recognize whether a product or system is appropriate for their needs", so covers the supplier's marketing literature. The availability of formal training and on-line help assist in several areas, but they are features of the supplier's solution, and it is the resulting *usability* of the product that is important.

5.3 Reliability

This includes *availability* and *recoverability*. In the real world, these are rarely pure application features, but aspects of the overall service, obliging human users and supporting staff to co-operate to handle whatever the application does not do for itself.

5.4 Security

Today's customers demand built-in mechanisms for controlling access, ensuring data *integrity* and protecting *confidentiality*. On a smartphone, much of the *security* surrounding a service is provided by the operational/usage environment, co-existing with the other applications and settings chosen by the platform owner. For example:

> "iOS is designed and built to … accept and install software that has been approved by Apple and run through the App Store. As such Apple has pretty much guaranteed that you won't encounter any malicious software on your iOS device" [19].

5.5 Maintainability

ISO/IEC 25010 defines *maintainability* as "the degree of *effectiveness* and *efficiency* with which a product … can be modified by the intended maintainers". The description of *maintainability* includes modifications carried out by "business or operational staff, or end users", so would include the deployment of supplier fixes.

Analyzability covers assessing the impact of an intended change, diagnosing deficiencies and failures, and identifying parts to be modified. *Analyzability* also covers problems noticed first in the field, requiring reliable configuration identification [5].

Testability is important, both at low-level (by allowing components to be tested independently and automatically) and at high-level, by developing and maintaining regression test suites to demonstrate that the previous service has not been downgraded. For the service, it is also useful to have post-installation and post-recovery tests, to confirm the service is working properly after human intervention.

Siakas and Georgiadou [20] proposed an extension (to ISO 9126), that Extensibility be considered as a primary level characteristic asserting that modular and object-oriented software "has become more extensible as tried and tested classes can be integrated into existing systems without the need to construct from scratch, or re-test the whole system".

5.6 Portability

Traditionally software *portability* has been a concern of developers working with the source code, involving recompilation, re-building etc. Applications are generally bought as implementations that execute only on specified platforms/computer ranges, and a customer wishing to move elsewhere has to buy the appropriate implementation for the new platform. Customers will then need to migrate their existing data, configuration, backups, users, etc. to the new environment.

Portability in ISO/IEC 25010 includes adaptation by end users and for "different … operational or usage environments", so it includes the purchased implementation's ability to run on any instance of the supported platform, including plug-compatible, virtual, outsourced or cloud-based environments.

The specification of the platform required is actually a critical matter to all commercially-minded software product vendors, as they will want to ensure that their software continues to work on any new and improved platforms added to the range in the future [4].

Replaceability focuses on the replacement of an existing product, or one version by another. Developers should also consider broader service issues such as the migration of existing data and minimizing changes to the user interface.

Established commercial practice is that any new release will at least maintain all previous functionality and features, and maintain access to existing customer data, i.e. be 'upward compatible', with no 'regression'. In some cases suppliers guarantee to meet the costs of any incompatible upgrade. Customers need surety that key product characteristics will be continued for as long as they want, so 'promises' should be backed up by contractual obligations on the supplier and any future substitute.

5.7 Supportability

The 'supportability' of a product is similar to maintainability in Sect. 5.5, contributing as one factor in the prediction of future experience, where neither the demand nor the response are known. It can be improved through the inclusion of appropriate functions and features for support activities. Most *maintainability* aids are useful contributors, although their emphasis must be changed to focus on defect reporting: the identification of defects in terms of errors in behavior and when exactly these happen. Other measures should take into account typical support scenarios from the acquirer's viewpoint, e.g. urgent calls for assistance, remote diagnosis, and support without access to source code or proprietary Intellectual Property: information or tools.

6 Other Characteristics of the Product at Delivery

6.1 Honesty and Openness

App Stores are a distinct world, with many unexpected perspectives [4]. There is an explicit requirement for open and honest communication – both for an acquirer contemplating a purchase, and for a user downloading an app. Reasons for rejection by Apple can include [15]:

- "Apps that do not perform as advertised by the developer"
- "Apps that include undocumented or hidden features inconsistent with the description"
- "Apps that are intended to provide trick or fake functionality that are not clearly marked as such".

Misleading documentation is not considered in the SQuaRE quality model.

6.2 Product Maturity

Bennett and Rajlich [21] suggested that products go through a maturity lifecycle, a staged model of initial development, active evolution, servicing and phase out. Acquirers looking for reliability might prefer a product in the third stage, whereas those hoping for growing functionality would prefer the second. In 1974 Lehman [22] emphasized that "first software systems must be continually adapted, or they become progressively less satisfactory. At the same time, software is becoming more and more complex and more expensive than before. As a software system evolves its complexity increases unless work is done to maintain or reduce it". In [23], Lehman also showed the strong effect that past product maintenance has on future quality.

7 In-life Experience, Post-deployment

This is what ultimately matters to the acquirer, but is unknown until a particular product has been bought, deployed, and a reasonable settling-in period allowed. Predictions of Quality in Use may be attempted from the supplier's Quality in Use data,

from reference sites, or from previous, related work by this supplier. The ISO/IEC 25010 quality models are inevitably focused around acquisition, but could be applied to normal operation, and to mid-life or supplier reviews.

Other activities affecting the acquirer's post-deployment experience can be split in various ways, e.g. Cancian [24]. In-life administration, production, support, maintenance and their manageability should be considered before selecting a product, but SQuaRE does not currently address them. They are all Post-deployment processes, subject to the performance variability of the 7Ms (see Sect. 3.1), so further Quality in Use and Product Quality characteristics could be defined. The acquirer could request the product supplier to provide all these supporting services for an agreed price, in which case a comprehensive set of additional service level measures would be needed.

Application-specific activities that support the functionality and features provided to direct and indirect users can be split into those that satisfy the acquirer's needs as normal, and other supporting activities introduced by the developer. The former are needed by the acquirer, so are covered by the 25010 *usability* models, whereas the latter are just part of the cost of ownership of that particular product and hence overheads.

7.1 Customer Support by Supplier

Product selection will be much better informed, if the acquirer has a clear idea of the operational environment to be used, and the associated IT services it hopes to give its users (see Trinkenreich [25]). Support is sometimes confusingly included under maintenance. ISO/IEC 15504-5: 2006 [26] makes a clear distinction, with a Customer Support process (OPE-2) whose purpose is "to establish and maintain an acceptable level of service through assistance and consultation to the customer to support effective use of the product". Customer Support's objective should be to resolve issues and minimize the need for repeat or follow-up calls: Seddon's 'failure demand' [27], that is "demand caused by a failure to do something or do something right for the customer". Tourniaire [28] suggests that less than 5% of calls are actually the result of a software bug.

The QuEST Leadership Forum's TL 9000 aims to meet the "supply chain quality requirements of the worldwide telecommunications industry" [29] and has defined support metrics, which are collected from registered suppliers every month.

As support is a post-acquisition activity, with only minor influence from the product's technical implementation, ISO/IEC 25010 does not cover it. Nevertheless, the supplier's support capability is an important element in software product evaluation, and should at least be mentioned, even if it is then explicitly excluded from the scope of the ISO/IEC 250xx series.

7.2 In-life Maintenance

Most software is bought-in, so acquirers want to understand the likely work, effort and cost required to maintain the product in service for their users, depending on their management goals [30]. This will inevitably be affected by the frequency and complexity of upgrades, resulting from the policies and procedures of the supplier, in terms

of commitment to fixing reported defects, quality and timeliness of new releases containing bug fixes, etc.

8 Conclusions

We have not attempted to look at all the criteria involved in acquiring software and creating and sustaining an ongoing relationship with its supplier, but restricted ourselves to the delivered software, and the product-specific implications, post-deployment.

The ISO/IEC 25000 series is a major step forward in software quality requirements specification and quality evaluation. Whilst the definitions and their notes are useful, readers may miss or dismiss many important interpretations. The acquirer's activities post-deployment are not considered, and yet must be a major part of their decision-making.

Our contribution has been to explore the scope of ISO/IEC 25010 and introduce further significant properties: legal, regulatory and contractual compliance, extensibility, supportability, honest description, product maturity, and in-life activities including Customer Support by Supplier and In-life maintenance.

Process improvement initiatives change the process, but also affect the resulting product. Management's goals may be to improve specified 'dimensions' of the product (the characteristics and subcharacteristics discussed earlier), maintain the existing levels, or disregard them (i.e. leave them unmanaged except where they appear in client requirements). ISO/IEC 25010, together with our proposed extensions, provides a comprehensive quality model which can be used as the basis for specifying whether a changed process should improve, maintain or ignore each dimension.

Future work will investigate case studies in order to assess the various impacts identified in this paper and also validate our suggestions for clarification and extension of the standard.

We thank the anonymous referees for their help in clarifying the text.

9 Relation to SPI Manifesto

This paper addresses Principle 3 of the SPI Manifesto [31]: *Base improvement on experience and measurements*. ISO/IEC 250xx series can be used to baseline the performance of existing processes in terms of the output product, and to specify any corresponding performance change intended from process improvement. ISO/IEC 25010 provides a comprehensive model to ensure all relevant dimensions are addressed.

References

1. Hofemann, S., Raatikainen, M., Myllärniemi, V., Norja, T.: Experiences in applying service design to digital services. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 134–148. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13835-0_10
2. Sjøberg, D.I.K.: The relationship between software process, context and outcome. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (eds.) PROFES 2016. LNCS, vol. 10027, pp. 3–11. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49094-6_1
3. ISO: ISO/IEC 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models
4. Estdale, J.: App stores & ISO/IEC 25000: product certification at last? In: Phalp, K., et al. (eds.) SQM XXIV: Systems Quality: Trends and Practices, pp. 37–48. Southampton Solent University (2016)
5. Estdale, J.: Products and prototypes: what's the difference? In: SQM XXV: Achieving Software Quality in Development and in Use, pp. 65–76. Southampton Solent University (2017)
6. ISO: ISO/IEC 25000:2014, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE
7. ISO: ISO/IEC 9126, various parts and dates. Software engineering – Product quality. Being superseded by SQuaRE [5]
8. ISO/IEC TS 25011:2017 Information technology—Systems and software Quality Requirements and Evaluation (SQuaRE)—Service quality models
9. Georgiadou, E.: Software process and product improvement: a historical perspective. Cybern. Syst. Anal. **39**(1), 125–142 (2003). https://doi.org/10.1023/A:1023833428613
10. Côté, M.A., Suryn, W., Georgiadou, E.: In search for a widely applicable and accepted software quality model for software quality engineering. Softw. Qual. J. **15**(4), 401–416 (2007)
11. Miguel, J.P., Mauricio, D., Rodriguez, G.: A review of software quality models for the evaluation of software products. Int. J. Softw. Eng. Appl. (IJSEA) **5**(6), 1–24 (2014). https://doi.org/10.5121/ijsea.2014.5603
12. Oriol, M., Marco, J., Franch, X.: Quality models for web services: a systematic mapping. Inf. Softw. Technol. **56**(10), 1167–1182 (2014). https://doi.org/10.1016/j.infsof.2014.03.012
13. Biscoglio, I., Marchetti, E.: Definition of software quality evaluation and measurement plans: a reported experience inside the audio-visual preservation context. In: Holzinger, A., Cardoso, J., Cordeiro, J., Libourel, T., Maciaszek, L.A., van Sinderen, M. (eds.) ICSOFT 2014. CCIS, vol. 555, pp. 63–80. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25579-8_4
14. Bergman, B., Klefsjö, B.: Quality from Customer Needs to Customer Satisfaction. McGraw-Hill, Maidenhead (1994)
15. App Store Review Guidelines. https://developer.apple.com/app-store/review/guidelines/. Accessed 17 Apr 2018
16. McCann, T.: The Art of the App Store. John Wiley, Indianapolis (2012)
17. Gibson, S.: 'Well-behaved' and 'misbehaved' software: past and present. InfoWorld **8**(43), 65 (1986)
18. Gibson, S.: Programs that 'behave' lend themselves to compatibility successfully. InfoWorld **8**(43), 69 (1986)

19. Haslam, K.: iOS security risks: after the XcodeGhost exploit is Apple's iOS really safer than Android? Plus: what security apps do you need for iPad & iPhone. http://www.macworld.co.uk/feature/iosapps/is-ipad-iphone-ios-safe-xcodeghost-what-security-software-need-3453938/#antivirus. Accessed 16 Apr 2018
20. Siakas, K.V., Georgiadou, E.: PERFUMES: a scent of product quality characteristics. In: International Software Quality Management Conference, pp. 211–220. BCS, London (2005)
21. Bennett, K., Rajlich, V.: Software maintenance and evolution: a roadmap. In: International Conference on Future of Software Engineering, pp. 73–90. ACM, New York (2000). https://doi.org/10.1145/336512.336534
22. Lehman, M.M.: On understanding laws, evolution, and conservation in the large-program life cycle. J. Syst. Softw. **1**(1), 213–221 (1980). https://doi.org/10.1016/0164-1212(79)90022-0
23. Lehman, M.M., Perry, D.E., Ramil, J.F.: Implication of evolution metrics on software maintenance. In: International Conference on Software Maintenance, pp. 208–217. IEEE (1998). https://doi.org/10.1109/icsm.1998.738510
24. Cancian, M.H., Hauck, J.C.R., von Wangenheim, C.G., Rabelo, R.J.: Discovering software process and product quality criteria in software as a service. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 234–247. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13792-1_19
25. Trinkenreich, B., Santos, G., Barcellos, M.P.: SINIS: a method to select indicators for IT services. In: Abrahamsson, P., Corral, L., Oivo, M., Russo, B. (eds.) PROFES 2015. LNCS, vol. 9459, pp. 68–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26844-6_6
26. ISO: ISO/IEC 15504-5:2006 Process assessment – Part 5: An exemplar Process Assessment Model
27. Seddon, J.: Freedom from Command and Control: A Better Way to Make the Work Work, 2nd edn. Vanguard Education, Buckingham (2005)
28. Tourniaire, F., Jarrell, R.: The Art of Software Support: Design and Operation of Support Centers and Help Desks. Prentice Hall, London (1998)
29. TL 9000 Overview. http://www.tl9000.org/about/tl9000/overview.html. Accessed 16 Apr 2018
30. Woherem, E., Neil, M., Estdale, J.: Software process improvement through the GQM approach: a maintenance case study. In: 3rd International Conference on Software Quality, pp. 147–156. ASQC, Milwaukee (1993)
31. Pries-Heje, J., Johanson J. (eds.): SPI Manifesto. http://www.iscn.com/Images/SPI_Manifesto_A.1.2.2010.pdf. Accessed 28 May 2018