

# The Software Quality Star:

## A conceptual model for the software quality curriculum

**Ronan Fitzpatrick**

School of Computing,  
Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland.  
Tel: +353 (1) 4024835, Fax: +353 (1) 4024985  
Email: ronan.fitzpatrick@comp.dit.ie

### Abstract

Usability is a significant factor in the creation of quality software products which is an important focus of the software engineering syllabus. Usability is also a significant factor in the study of Human-Computer Interaction (HCI). Despite the connection, software quality and the rigours of the software engineering discipline are not used in the teaching of HCI and visa versa. Consequently, students undertaking software engineering courses are not fully exposed to usability issues and HCI students often miss out on the disciplines of software engineering. This paper presents a quality focused conceptual model – the Software Quality Star - which can be used in both syllabi.

## 1 Introduction

Software quality is a well researched and understood discipline in software engineering. Quality is an aspiration of everyone. External quality factors are accepted as those that impact end users. Fitzpatrick and Higgins (1998) argue that the combined set of external quality factors constitute usability which is a significant consideration for HCI professionals. However, while quality is an objective of HCI professionals, the HCI curriculum is not driven by a quality perspective. At the same time software engineering professionals want to produce quality products which will be used by end users, but their syllabus is not driven by a focus on usability.

The aim of this paper is to introduce to IT educators and practitioners a conceptual model for software quality, a model of quality perspectives, which is motivated by the International Standard ISO/IEC 12207 (1995). This standard addresses life

cycle processes and the model considers quality throughout those processes. User considerations are core to those processes, so, the model incorporates elements, which impact HCI. The model was developed specifically for undergraduate programmes, where it is being successfully used.

Section 2 introduces the Software Quality Star and explains the originating philosophy underpinning the model. Section 3 provides an overview of the principal topics addressed by the various components of the model, illustrating how quality is the principal motivator. Section 4 clarifies how the model supports the transfer of software engineering theory to the discipline of Human-Computer Interaction. Section 5 outlines current usage in an academic context and Section 6 offers some concluding comments.

## 2 The Software Quality Star

This section presents the conceptual model of the Software Quality Star and clarifies the philosophy underpinning its origin. The section also comments on the comprehensive paper that supports the understanding of the model.

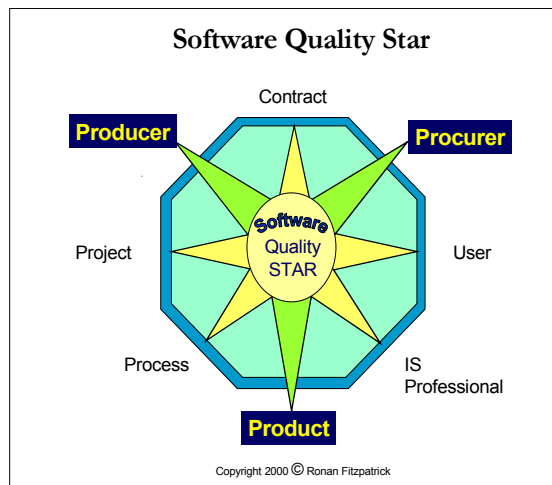
### 2.1 The conceptual model

The software quality star is a conceptual model for presenting the different perspectives of software quality as seen by the different software product stakeholders and is based on the Acquirer and Supplier as defined in ISO/IEC 12207 (1995). The model is illustrated in Figure 1.

There are three significant elements in the star – the **Producer** (Supplier), the **Procurer** (Acquirer) and the **Product**. In the model the Procurer enters into a contract with the Producer to create a software product. This contract will clearly specify the

quality characteristics of the product. The Procurer's perspective of the producer organisation is that they will use best project management practice and engage in first-rate processes to create a quality product. The Procurer's perspective of the product is that it will be acceptable by the user community and that it can be serviced and maintained by their IS professionals.

The model considers the Procurer to be the lead party in any contractual arrangement as it is the Procurer's users and technical support professionals



**Figure 1** – *Software Quality Star*

who will dictate the success or failure of the software product. It is also the Procurer who will dictate the profile and maturity of the selected Producer organisation.

The model also accommodates the Producer's perspective of software quality and focuses on the maturity of the Producer organisation as software developers and the development processes that they used to create quality software products.

## 2.2 Motivating philosophy

The motivating philosophy for the model is software quality and the principal aim is to show that quality permeates all of the stakeholder's perspectives. The philosophy is not limited to the belief that quality is "conformance to specification" or "fitness for purpose." Instead, the model relies on the philosophy of Kaoru Ishikawa, the founding father of Japanese thinking on quality who writes "that

*Broadly interpreted, quality means quality of work, quality of service, quality of information, quality of process, quality of division, quality of people including workers, engineers, managers and executives, quality of system, quality of company, quality of objects"* (Ishikawa, 1985). Section 3 illustrates this quality focus in more detail.

## 2.3 Supporting papers

All of the perspectives outlined in the model are comprehensively explained from a quality perspective in a supporting paper. These explanations rely on acknowledged and seminal work in the area and are fully illustrated. The extended *Software Quality – Strategic Driver Model* (Fitzpatrick, 2001) is also used as supporting reading.

A summary of the components and topics addressed are set out in the next Section.

## 3 Components and topics

There are three main perspectives in the model, the Producer's perspective, the Procurer's perspective and the product perspective. Each is explained now in turn.

### 3.1 Producer's perspective

The Producer's perspective is driven by the desire to be a quality organisation employing first-rate staff who engage in first-rate processes using first-rate tools and techniques to create quality software products. In keeping with the House of Quality model (Hauser and Clausing, 1988), the Producer will be driven by an enlightened philosophy and leadership. This philosophy is underpinned by the belief that to achieve quality there must be political (organisational) stability and that there is a need for continuing education among the actors involved. The organisational ability will support organisational desire to create quality products and the education will support the ability to specify quality requirements. This is especially important for aligning the Acquirer's corporate processes and their software requirements. The need to address software development strategies like developing reusable code and software portability are significant issue for the Producer organisation. These topics considerably impact the cost of creating software products and might not be of interest to the Acquirer. The accepted approach to successfully creating these

first-rate software products is to engage in software project management and an expression often used to provide focus for the objective of project management is “*to complete the right project, on time and within budget*”.

Engaging in a project is a software industries approach to creating a software product and ISO/IEC 12207 (1995) gives project management guidance in section 5.2 of the standard. This means that project management best practice is used in order to assure the successful delivery of the product and this best practice employs planning, organising, controlling and directing throughout the development life cycle. It is through this best practice that the Supplier demonstrates to the Acquirer their standing and competence as a Supplier organisation.

ISO 12207 (1995) places the onus on the Supplier *to develop and document project management plans* (section 5.2.4.5) and continues that *the Supplier shall implement and execute the project management plans* (section 5.2.5.1). So, there are two critical aspects to the Supplier's responsibility (develop and document AND implement and execute) and their level of competence in these is what the Acquirer should evaluate.

In this perspective the process for creating the software product is all-important. As part of the creation of software products, supplier organisations will engage in a set of processes. In the early days of software development these processes were established almost by trial and error and sets of standards evolved to suit developer understanding and practice. More recently, organisations like the International Organisation for Standardisation, the Software Engineering Institute and different developer companies have devised standards and models, which quantify full and comprehensive sets of processes. The philosophy behind this approach is that by addressing these comprehensive processes, supplier organisations will create quality software products. The International Organisation for Standardisation (ISO) and the International Electrotechnical Commission (IEC), have published ISO/IEC 12207 (1995) relating to software life cycle processes, the Software Engineering Institute has developed the Capability Maturity Model (Paulk *et al.*, 1993b) and ISO/IEC are developing the SPICE standard (ISO/IEC TR 15504:1998). The discussion in this chapter will be confined to these three

approaches but the reader should be aware that there are other solutions, especially in the commercial sector.

### 3.2 Procurer's perspective

In the context of software quality, the Procurer is obviously interested in knowing that the Producer is a first-rate organisation, which uses first-rate processes to create software products that incorporate all of the most appropriate quality factors. However, there are also strategic issues, which the Procurer must address. For example, the Procurer will be interested to know that there is alignment between the software product and the organisation's business processes. The Procurer will also be concerned to know that the software product will be usable, that it contains the highest technical excellence and that the organisation can afford the software and secure a return on the investment. There are also legal consideration relating to the quality of software which the Procurer must be satisfied are conformed to by the software. Competitive advantage or competitive survival is also a Procurer's concern. Typically, these are Strategic Quality Drivers which have been addressed by Fitzpatrick (2001)

For the purpose of this section the IS professionals being considered are the technical professionals of the Acquirer organisation who have line responsibility for IS. Their role begins with advise to management regarding the specification, selection and acquisition processes and would typically address technical excellence, user empowerment, corporate alignment and investment efficiency together with supplier organisation profile. They have responsibility through to the retirement of the software at the end of its operational life.

During the specification, selection and acquisition processes they will identify the quality characteristics required of the software product and will ensure that contract documents address these issues.

During the operational stage they are responsible for supporting the users or operators of the software and for servicing and maintaining the software during its operational life. As part of their supporting role these IS Professionals are concerned with all of the quality factors as outlined in section 3.4. For example, the installability of the software might be especially of interest to them, or, the

reliability of the system might seriously impact their workload. From the time that a new system is delivered, technical support relies heavily on user and technical manuals. So, from this IS professional’s perspective an essential requirement of a quality software product is a full and comprehensive set of these.

As part of the servicing role they will be addressing such quality issues as reinstalling sections of software where modules become corrupted and will require systems that support this type of servicing activity.

And, as part of their maintenance role they are required to **adapt** software to suit changes such as government directives, changing rates like pay scales, tax rates and similar items. They will be required to **correct** any bugs or other errors that manifest themselves during the life of the product and they will be required to **perfect** the software especially by fine-tuning algorithms to improve their efficiency and to meet new user requirements. To assist them in these activities they will be especially interested to know that good designs, documentation and best programming practice has been used during the project phases when the software was being created.

In addition to supporting other users and maintaining systems, these professionals are often themselves the users of products like network operating systems and management tools, so, they too will be impacted by quality of use issues. While for some, the internal quality factors may be their

primary interest they will also be concerned that the external quality factors fully support users.

### 3.3 Product perspective

From this perspective a software product is considered to be a quality product if it supports a set of quality factors or product characteristics. Software quality factors were first defined in the 1970’s by researches like McCall *et al.*, (1977) and Boëhm, (1978). Their research was later complemented by standards like IEEE and ISO. More recently, Fitzpatrick and Higgins (1998) conducted a methodical analysis and synthesis of three strands - quality (as explained by McCall *et al.* and by Boëhm), statutory obligations, and human-computer interaction, which influence software quality. This established a comprehensive set of quality factors, and those factors that related to users, they called the **attributes of a usable software product**. More recently this set has been extended to show that, depending on the domain, additional quality factors will be involved. Figure 2 shows Fitzpatrick and Higgins original set of CORE QUALITY FACTORS together with the DOMAIN SPECIFIC QUALITY FACTORS appropriate to the World Wide Web.

Figure 2 is divided to categorise the quality factors into External quality factors (those that significantly impact end users), Internal quality factors (those of a “technical” nature which are of specific interest to IS professionals, and a new category - Strategic quality factors – which are of special interest to strategic managers and IS Procurers. This last category is new in that

	EXTERNAL QUALITY FACTORS	INTERNAL QUALITY FACTORS	STRATEGIC QUALITY FACTORS
<b>CORE QUALITY FACTORS</b>	<ul style="list-style-type: none"> <li>• Suitability</li> <li>• Installability</li> <li>• Functionality</li> <li>• Adaptability</li> <li>• Ease-of-use</li> <li>• Learnability</li> <li>• Interoperability</li> <li>• Reliability</li> <li>• Safety</li> <li>• Security</li> <li>• Correctness</li> <li>• Efficiency</li> </ul>	<ul style="list-style-type: none"> <li>• Maintainability</li> <li>• Testability</li> <li>• Flexibility</li> <li>• Reusability</li> <li>• Portability</li> </ul>	
<b>DOMAIN-SPECIFIC QUALITY FACTORS</b>	<ul style="list-style-type: none"> <li>• Visibility</li> <li>• Intelligibility</li> <li>• Credibility</li> <li>• Engagibility</li> </ul>		<ul style="list-style-type: none"> <li>• Differentiation</li> </ul>

Figure 2 – Quality factors

researchers have not addressed it in the past and is now the focus of some researchers. The grid also shows how it is necessary to consider special domains (in this case the World Wide Web) and illustrates the additional quality factors required for this domain. For example, if the domain is mobile computing or virtual computing then new, as yet unquantified quality factors might be needed for those domains. The set of Core quality factors was identified in relation to traditional data processing applications. So, it is also necessary to interpret the Core quality factors for each new domain. For example, in a traditional data processing environment, maintenance was concerned with **corrective**, **adaptive** and **perfective** maintenance. Maintenance of a Web site is a very different occupation with content in particular often needing to be updated daily.

#### 4 Bridging the SE-HCI gap

There are two significant contributions that the Software Quality Star makes towards bridging the gap between software engineering and Human-Computer Interaction.

First, it is quite common that students of the many disciplines that impact information systems don't fully appreciate how the different study modules fit together. It is not until they experience a project in action that their understanding and appreciation is completed. Using quality as a common strand and an international standard of life cycle processes the Software Quality Star becomes a tool which helps to overcome this difficulty.

Second, the Software Quality Star creates an awareness and understanding of the contribution that can be made by other disciplines. Consequently, it becomes appropriate for informed students of one discipline to question if practice in the other discipline might be appropriate for use in theirs. A typical questioning might refer to quality metrics which are well understood in the software engineering discipline, but they are not researched or studied to the same extent in the HCI domain. This is significant because HCI for E-Commerce is becoming a major consideration for Web site owners who are looking for quality Web sites to secure competitive advantage.

#### 5 Conclusion

The Software Quality Star was originally developed for teaching software quality topics. It has been successfully used at undergraduate level with computer science and business studies students in their software engineering, project management and HCI modules. It can also be successfully used as a framework for teaching other modules which address legal and contracting issues, and similar IT management topics. At postgraduate level during Information Technology and Strategic Management seminars an enhanced version of the model, *Software Quality – Strategic Driver Model (SQ\_SDM)*, is used. This extended version of the Software Quality Star focuses further on the competitive advantage perspectives of the Acquirer and the Supplier. Both models could be easily interpreted for new domains like the WWW and new and evolving technologies like mobile solutions.

#### References

- Boehm, B. (1978) *Characteristics of software quality*, Vol 1 of TRW series on software technology, North-Holland, Amsterdam, Netherlands
- Fitzpatrick, R. and Higgins, C. (1998). Usable software and its attributes: A synthesis of software quality, European Community law and human-computer interaction, In: *People and Computers XIII. Proceedings of HCI'98 Conference*, Springer, London, UK
- Fitzpatrick, R. (2000) Additional Quality Factors for the World Wide Web, *Proceedings of the Second World Congress for Software Quality*, Yokohama, Japan, Union of Japanese Scientists and Engineers (JUSE), Tokyo, Japan
- Fitzpatrick, R. (2001) Strategic Drivers of Software Quality: Beyond external and internal software quality, *Second Asia-Pacific Conference on Quality Software, Proceedings of APAQS 2001*, Hong Kong; IEEE Computer Society Press, California, USA.
- Ishikawa, Kaoru (1985) *What is Total quality control? : The Japanese way*. Prentice Hall, Englewood Cliffs, London, UK, p44/5
- ISO/IEC 12207 (1995) *International Standard. Information technology - Software life cycle processes*, International Organisation for Standardisation, Genève, Switzerland
- McCall, J., Richards, P. and Walters, G (1977) *Factors in software quality*, Vols I-III, Rome Aid Defence Centre, Italy