An Enterprise Framework for Evaluating and Improving Software Quality

Philip Lew
philip.lew@xbosoft.com

Abstract

With the world's economy increasingly driven by software products, there has been a relentless pursuit of software quality with research in model development, agile methodologies, and quality measurement. However, attention to and attempting to improve the quality of the product or its development process alone may not be the most effective means to improve quality. There are many other parts of the enterprise and various other factors that influence quality, especially where user perception of quality (sometimes called "quality in use") may be somewhat different than the product quality in a laboratory environment. Although requirements are often a cited reason for product quality problems, the quality of the sales process can also have a direct influence on requirements (i.e., custom software), and thereby influence quality. Customer service can also have great influence on the customer perception of the product's quality. Considering the many standards for software product quality, quality frameworks, and development models, we propose a framework called the Enterprise Quality in Use as an alternative view —including other parts of the organization's processes and outputs to improve quality not only in development phase, but also through each phase of the product lifecycle.

Key words: Quality model, Quality in use, Enterprise quality model, Software quality improvement.

Biography

After working for over twenty years in various management and technical positions in product development and management, CEO Philip Lew now leads the direction and strategy of XBOSoft (www.xbosoft.com). Phil works with organizations to assess the quality of their software, examine software quality processes, and set forth measurement plans so they can consistently improve software quality using systematic methods. He has authored articles in IEEE and ACM journal publications and trade journals; presented at several conferences on software usability, user experience, and quality evaluation; and is conducting post-doctorate research focused on software quality measurement. Phil has a B.S. and Master of Engineering in Operations Research from Cornell University and Ph.D. in Computer Science Engineering from Beihang University.

1 Introduction

As software becomes more ubiquitous in our everyday lives, shorter development cycles put pressure on software product quality. Abundant research and standards development has occurred in the areas of software quality, software quality models, and software quality processes. The ISO 9000 family of standards was developed to assist organizations implement quality management systems [2]. Similarly, ISO 25010 [3] was developed for use in the field of software engineering — to help organizations identify relevant quality characteristics for establishing requirements, their criteria for satisfaction and the corresponding measures specifically addressing product quality and quality in use (QinU).

Despite research focused on modeling and improving and organizations' capability to build quality products (ISO 9000 and TQM [1]), and standards addressing the evaluation of software products' quality (ISO 25010), a gap exists in the area of measuring and evaluating the end user's view on quality as influenced by the organization. Development models such as Agile, Scrum, Spiral, V-model, and others begin at requirements and end with acceptance testing. Yet there are other parts of an organization and its processes, prior to requirements and after acceptance testing, that can significantly influence a customer's perception of software quality.

To address this shortfall, this paper proposes a novel quality framework for examining and improving software quality — the Enterprise Quality in Use (EQinU) framework. EQinU is a flexible framework that can be used in any organization based on concepts similar to the ISO 25010 where the outputs of one phase of quality influences the quality at the next phase.

2 Related work on development and quality models

This section discusses quality and development models to provide a general background and lay the foundation for EQinU which is based on similar concepts.

2.1 ISO 25010 Quality Model

ISO 25010[3] is the newest standard on system and product quality models. You can think of a model as a way to break down abstract concepts such as quality into something we can get our hands around. The ISO 25010 [3] standard's views of quality can be summarized as follows:

1) Product Quality (PQ) - Specified by a quality model (i.e. a set of eight characteristics–Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability and Portability - and a set of sub-characteristics per each characteristic are prescribed), as shown in Figure 1.

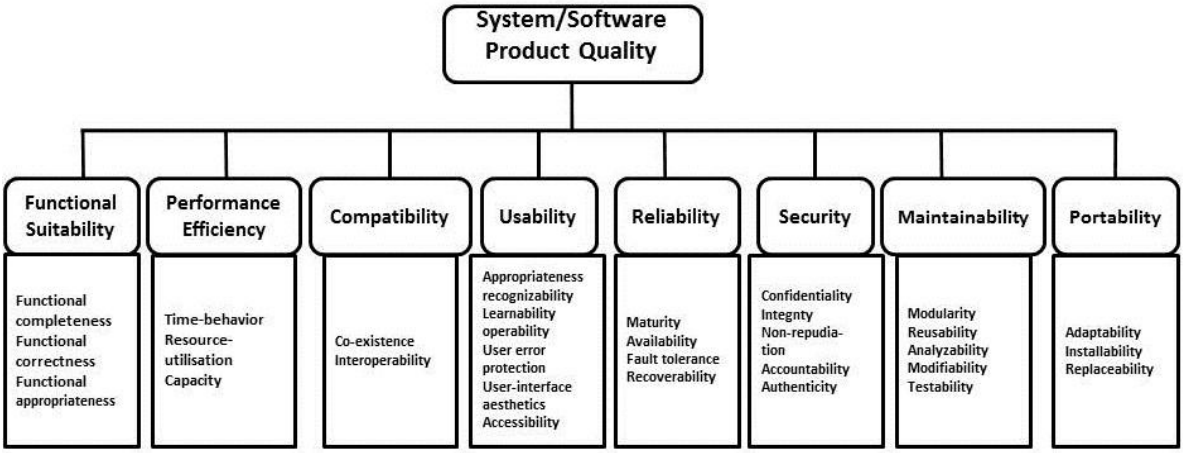

Figure 1. ISO 25010 product quality model [3]

The ISO product quality model defines quality characteristics such as Usability by using sub-characteristics, i.e. Learnability. This is a common method where decomposing a characteristic into sub-characteristics enables us to better understand the meaning of the characteristic.

2) Quality in Use (QinU) - Specified by a quality model (i.e. a set of five characteristics–Effectiveness, Efficiency, Satisfaction, Freedom from risk and Context coverage) as shown in Figure 2. Note that each characteristic can be measured and evaluated by the extent to which specific user needs in an actual, specific context of use are met.

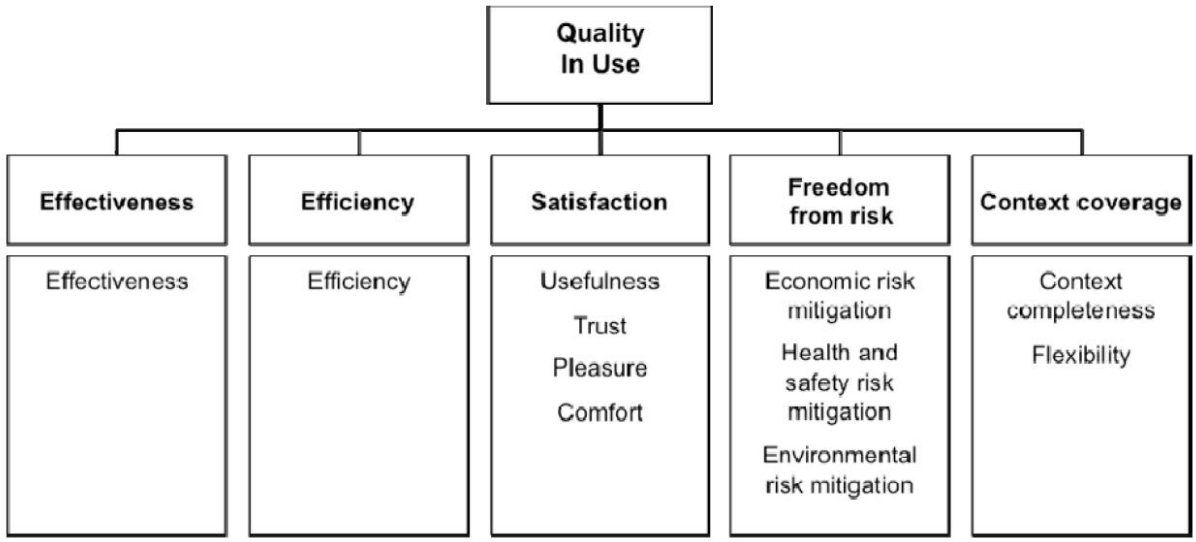

Figure 2. ISO 25010 Quality in Use model [3]

The ISO 25010 QinU model was developed to clearly differentiate product quality from the product's effect in a real situation of use. A product could have 'good' quality at a product level, with very good Performance Efficiency, yet 'in-use', Satisfaction could be very poor. As an example, suppose that a software application was designed to have the menus on the left rather than at the top. From a product quality point of view, it may fully satisfy user interface aesthetics criteria. This design may also satisfy Functional Completeness and Correctness criteria as well. However, from the user point of view, the unexpected menu location may negatively impact the Operability and Learnability of the software. The user perceives the application as having very low Efficiency.

Quality models can be used to specify and evaluate software quality from different perspectives in the acquisition, requirements definition, development and evaluation of software. In practice, depending on the domain and the end users, when modeling quality we typically include a handful of characteristics and sub-characteristics that are most important to the evaluator. For instance, a person in a purchasing department may use the model to specify requirements for each characteristic that vendors must adhere to. For example, under Performance Efficiency (time behavior), all printing response times shall be less than 3 seconds for the first page. For a person evaluating software usability, they may use only one characteristic of the model and add more precision and depth only to the usability characteristic. For a stakeholder involved in software design and allocation of resources, they may put different weights on the different characteristics of quality depending on the domain of their application. Rather than saying 'quality is important', quality models give us a means to better define our meaning.

My previous research utilized the Product Quality/Quality in Use paradigm and developed a flexible framework called 2Q2U (Quality, QinU, actual Usability and User experience) designed for evaluation of Quality in Use [4]. For 2Q2U, we used the ISO 25010 premise that Product Quality (PQ) influences QinU. For instance, if help is contextually based, then this influences the user's ability to learn the software when using it for a particular task. In this research, it was found that some characteristics, if improved at a

product level, definitely influenced performance at the user level. As shown in Figure 3 [5], if help completeness is improved, this will lead to improvement in the users' ability to complete tasks.

Related Quality in Use Attribute	Related External quality Related Attribute
Learnability in use: Sub-task completeness learnability	1.1.2.2 Learnability.Helpfulness.HelpCompleteness
	2.1.1 Information quality.InfoSuitability.Consistency
Effectiveness in use: Task Successfulness	1.2.1.2 Ease of use.Controllability.StabilityofMainControls
	1.1.2.2 Learnability.Feedback Suitability.Task Progress Feedback Appropriateness
Efficiency in use: Sub-task completeness efficiency	1.1.1.3 Learnability.Feedback Suitability.Entry Form Feedback Awareness
	1.1.2.1 Learnability.Helpfulness.Context-sensitve help availability
	1.2.3.1 Ease of use.Data Entry Ease.Defaults
	1.2.3.2 Ease of use.DataEntryEase.MandatoryEntry
	1.2.3.3 EaseofUse.DataEntryEase.ControlAppropriateness
	2.1.2.1 Information quality.InfoSuitability.InfoCoverage.Appropriateness
Effectiveness in use: Sub-task completeness	2.1.1 Information quality.InfoSuitability.Consistency
Effectiveness in use: Sub-task correctness	1.1.2.2 Learnability.Helpfulness.HelpCompleteness
	1.2.2.1 Ease of use.Error Mgmt.Error Prevention
	1.2.3.1 Ease of use.Data Entry Ease.Defaults
	1.2.3.3 EaseofUse.DataEntryEase.ControlAppropriateness
	2.1.2.2 Information quality.InfoSuitability.InfoCoverage.Completeness
	2.1.2.1 Information quality.InfoSuitability.InfoCoverage. Appropriateness

Figure 3. Relationships between Quality in Use (end user quality) and product quality attributes [5]

Using Product Quality and QinU models, Figure 4 shows the 'influences' and 'depends on' relationships from ISO 25010 [3] where one phase influences quality at the next phase including process quality as well.

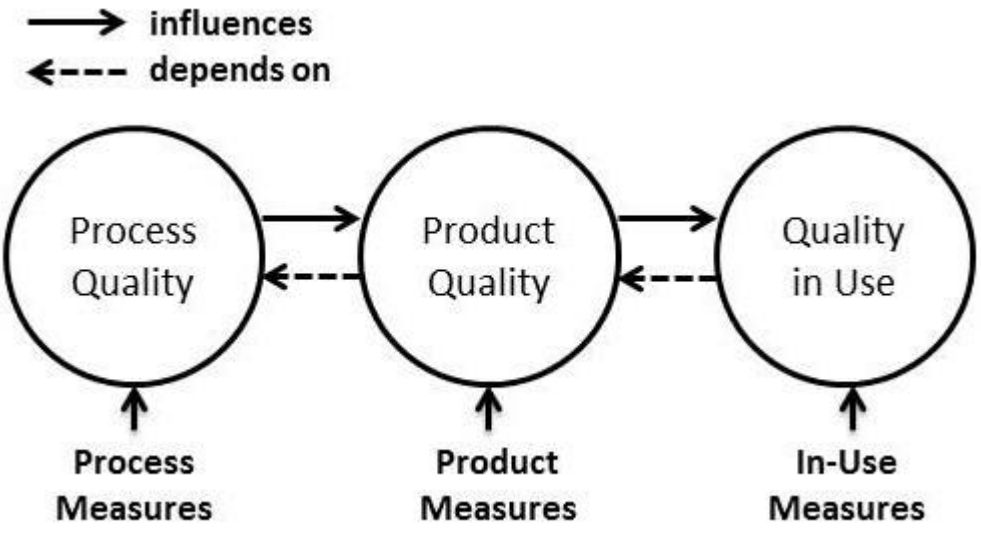

Figure 4. Quality in the lifecycle

The ISO 25010 standard and quality models are very general and most practitioners only use them as a guideline or starting point for what to consider when modeling and measuring quality. ISO 25010 model

usage states: “tailor the quality model giving the rationale for any changes.” What we can take away from this discussion on ISO 25010 are 3 main points:

- One phase of quality can influence the next phase as shown in Figure 4. We all know from CMMI that great processes do not necessarily lead to great product, but they do have influence.
- In practice, in previous research, we have been able to demonstrate a positive correlation made between product attributes and quality-in-use performance, thus showing the usefulness of this phase by phase ‘influence’ modeling concept.
- Modeling quality using a hierarchical tree format makes it easier for us to conceptualize and understand what quality is, which is the first step toward improvement.

2.2 ISO 9000 Quality Standard

- The ISO 9000 [2] family of standards was developed to assist organizations with implementing and operating effective quality management systems. ISO 9000 is founded upon eight quality management principles. The principles most applicable to our quality modeling work include:
- *Process approach*: Efficiently achieving desired results through activities and related resources that are managed as a process.
- *System approach to management*: Identifying, understanding and managing interrelated processes as a system which contributes to the organization's effectiveness and efficiency in achieving its objectives.
- *Continual improvement*: Continual improvement of the organization's overall performance should be a permanent objective of the organization.

The ISO 9000 standard has a broad and general reach and can apply to all organizations striving to increase the quality of their products and services by applying these principles in their operations. However, it does not contain details on quality characteristics or using decomposition as a means to modeling and evaluating quality. What we can take away from ISO 9000 is:

- Quality practices should be applied to the organization as a whole and not just one specific department.

There are other models similar to ISO 9000, including Total Quality Management [1]. These methodologies and models are oriented towards general quality processes and organizational capabilities to develop quality products and services, but still lack quality modeling characteristics and sub-characteristics for more specific understanding such as in ISO 25010.

2.3 Development models

There are many development models, but the objective and framework of the models are generally designed to solely model and measure software products or processes within the sphere of influence of the development organization. The V-model [7], for instance as shown in Figure 5, only shows requirements through acceptance testing.

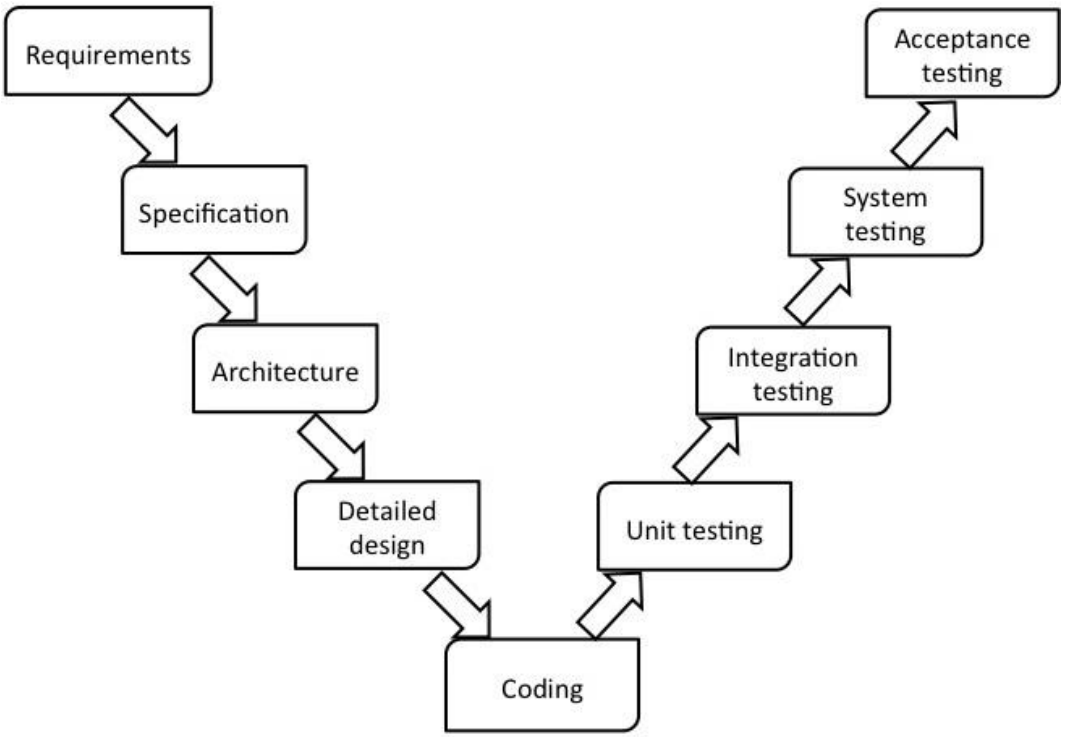

Figure 5. Simplified V-model Development Paradigm

The V-model and others including agile, waterfall, spiral, etc. all primarily focus on development and QA as the primary drivers of quality. A few critical elements are lacking:

- The end user's perception of quality is influenced by many other factors outside of development and quality assurance.
- Other parts of the organization also influence the quality of not only the product itself, but also the user's perception of quality.

3 EQinU Framework

In developing the EQinU framework, we kept in mind an overriding philosophy not to just set up a framework for a good product, but for a quality product from the end user's perspective. We also considered the main points of the previous discussion:

- Quality should be modeled in a hierarchical manner for ease of understanding and therefore improvement.
- Quality should incorporate the processes from other parts of the organization, not just development and quality assurance.
- Quality from one phase in a product's lifecycle development can influence the quality at the next phase.

Given this, let's examine a typical product lifecycle, as shown in Figure 6, where a company conceives of a product, then selling and producing it, and finally maintaining and servicing it. During the course of maintaining and servicing the product, the company runs into new opportunities for different or adapted products and the cycle begins again.

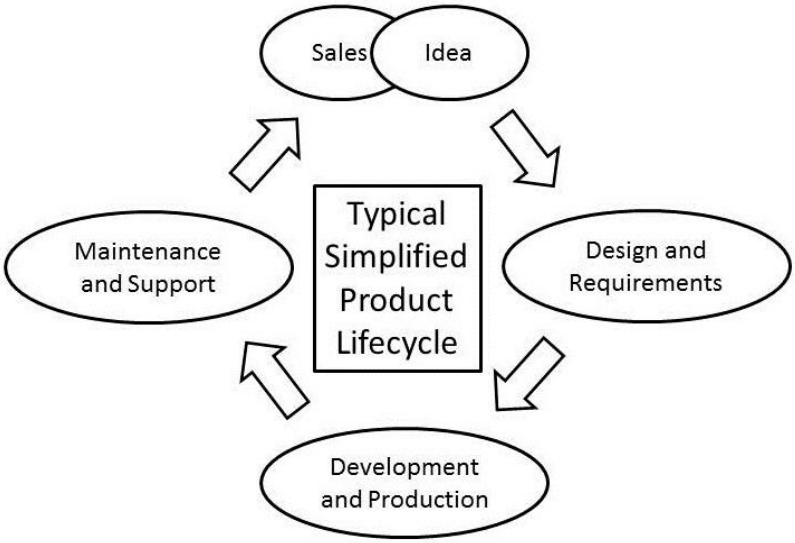

Figure 6. Typical Product Lifecycle

Now, we take the concept from ISO 25010 where one phase of quality has influence on another downstream phase, and each phase's quality is dependent on the output of one or more previous phases. In a general sense, quality of the outputs of phase N influences quality at phase N+1. By transforming Figure 6 from a cycle into a linear production line, this is conceptualized in Figure 7.

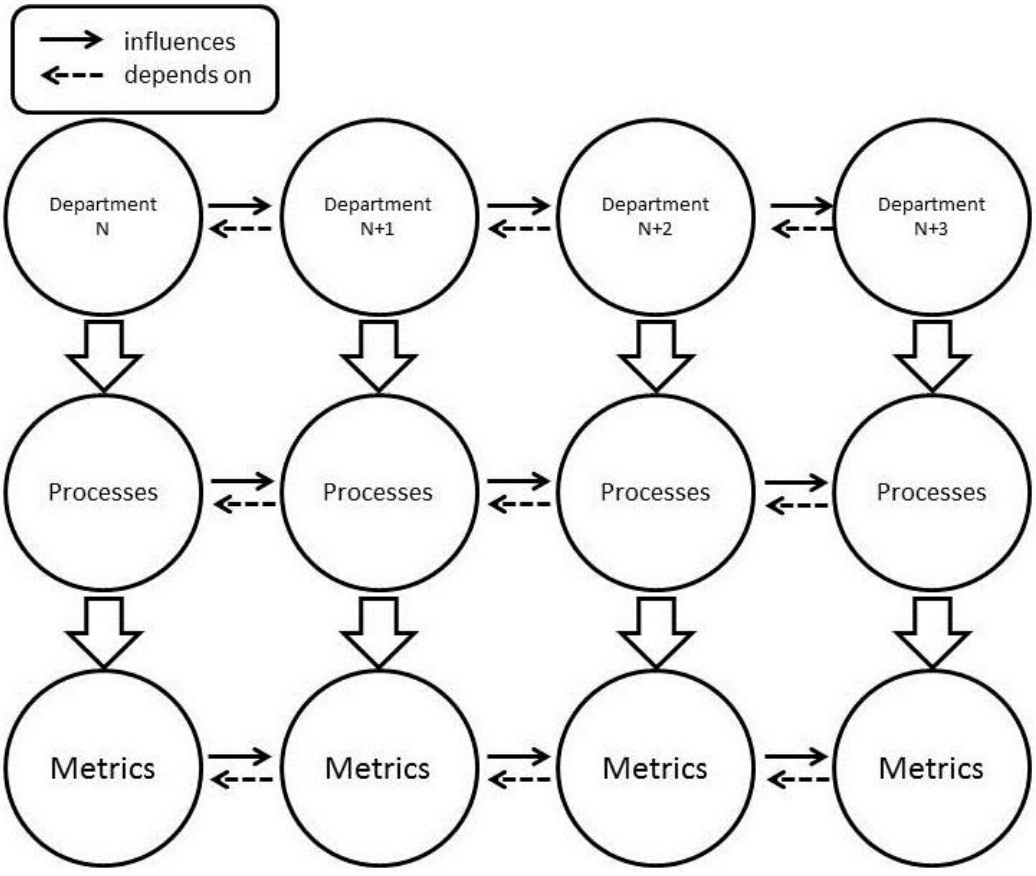

Figure 7. Proposed EQinU modeling framework.

More specifically, we instantiate the framework with a more specific model oriented towards organizations producing software as shown in Figure 8.

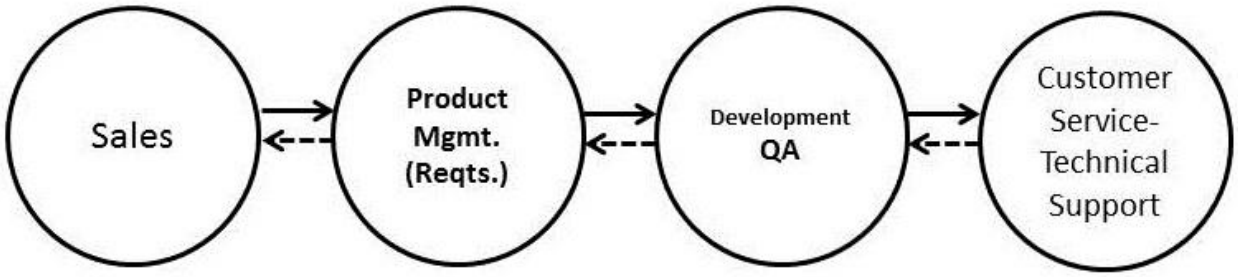

Figure 8. EQinU Software company general instantiation.

This extends the V-model forwards and backwards in an entire software development organization, where sales processes can influence the quality of product requirements, and in turn, the product requirements processes can influence the quality of development. Further in the product cycle, customer service or technical support can give negative impressions to an existing customer by entering a service ticket either incompletely or incorrectly such that it takes longer than it should to resolve the issue, or the customer needs to call again for the same issue. It's easy to see that the output of one process influences the performance and outputs of the next process down the line.

Therefore, the concept of a Total Quality Lifecycle of a product, from inception to usage by the end user, should model not only the product quality, but also phases prior to the product, and after the product has been developed, as shown in Figure 9.

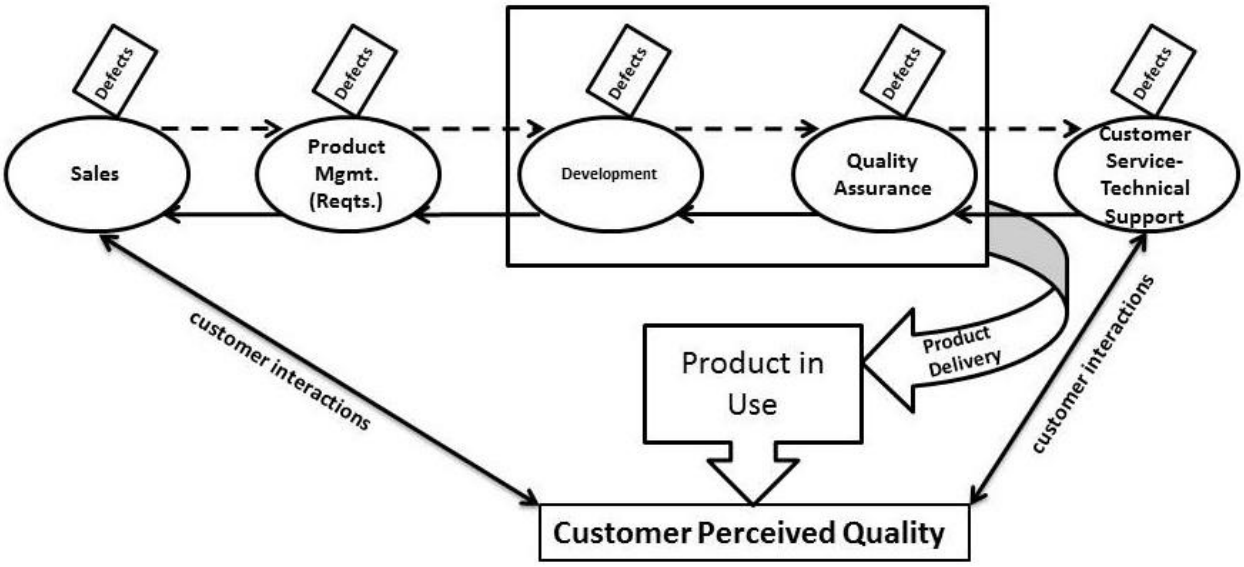

Figure 9 Total Quality Lifecycle

Each phase produces output, and that output could contain defects influencing later phases of the entire process that in the end, affect customer perceived quality. Below are a few examples:

- Sales: Sales presents the product and its features and capabilities to prospective customers. If they present the product such that after the sale, customer expectations are not met (for example

delivery date or product capabilities), then this could contribute to a low perception of product quality. Misrepresentation in its various shapes and forms could be considered a sales defect.

- Product Management (Requirements): Product management is responsible for many things and one of them is gathering from customers features to prioritize and include in the product roadmap. When features are specified and requirements are written and handed off to developers, if the developers do not understand the requirement fully, or must go back to ask the product analyst for clarity, this constitutes a requirements defect. Additionally requirements that are incomplete or not captured also represent a defect the same way as if you delivered a drawing of a table to a furniture maker that only had three legs.
- Development: These are familiar defects that already have plenty of metrics and measurements. A developer writes code according to a requirement or user story, and there is an error in the code causing behavior to be different from the requirement. Derivatives of these defects could be defect regressions or defects that, when fixed, create other defects.
- Quality Assurance: Testers also produce defects in their work by not clearly documenting defects so that developers can understand them or not thoroughly investigating an incident such that the defect cannot be reproduced. Note that this is a defect in testers' work product, not the product itself. Ultimately, errors or defects in the output from QA can affect the quality of the end product.
- Customer Service/Technical Support: Support and service representatives can create defects when they take calls from customers and give incorrect or incomplete information causing a customer either not to be able to solve their problem, or need to call back or both.

4 Discussion and usage of the framework

As experience with CMMI has shown, stringent processes and documentation do not guarantee a quality product. The EQinU framework is not intended to be a standard, or anything similar to CMMI where all the 'I's need dotting and 'T's crossed with an auditor by your side. Rather, it is intended for use at a departmental level, to uncover elements that can influence quality, in particular in departments that traditionally were thought to have minimal influence on quality.

4.1 Field example: sales and product management

You may think that a sales call and marketing literature are unrelated to customer satisfaction, but we found in working with clients that quality problems can start way before development. Let's look at an example as shown in Figure 10.

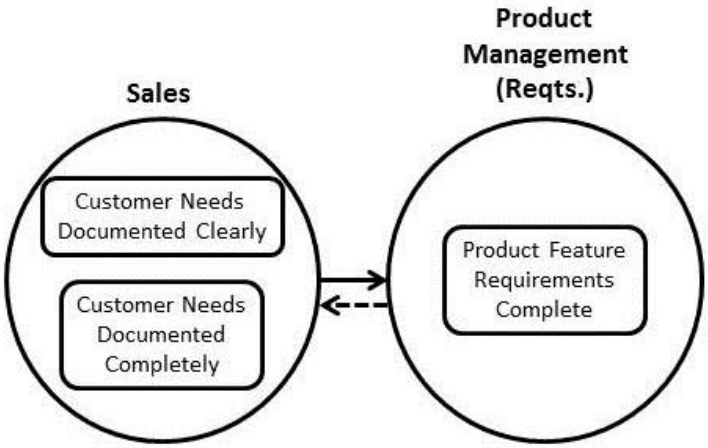

Figure 10. Sales Defects Influence Product Management (Reqts.)

Let's assume that the salesman, in a hurry to make the sale, makes a few mistakes:

1. He forgets that the customer has a special requirement regarding Euro currency, for online conversion each day automatically converting the rate according to a certain website (Customer needs not documented completely).
2. The customer tells him that they need to have the software delivered before the end of the summer. The salesman documents this as, "must deliver end of Q3" (Customer needs not documented clearly).

As a result of these omissions and inaccuracies, the Product Management department has errors in their requirements. They pass these requirements to development. Using an agile development methodology, this might be caught if the customer is deeply involved, but there is also a distinct probability that it may not be discovered until the product is delivered. Delivering the product on September 30, or even October 6, may be beyond what the customer thought of as 'end of summer' and there is no automatic conversion from Euros to USD. The customer is unsatisfied.

4.2 Field example: Customer service

If we look at defects in an organization, we can find them in many places other than those in development and QA. Let's examine an example in customer service where we were able to listen to sample calls from several customer service representatives and investigate the service tickets that resulted:

1. The customer called in because they had a problem posting changes from the client application into the cloud. After posting the changes, the system did not provide feedback that the changes were posted. They called into customer service to report the problem.
2. Because many people were having the same problem, there were heavy call loads that day. The heavy load caused many service representatives to take longer breaks, which contributed to both long wait times and exceptionally long call times. We noticed that sometimes, the representative would say "can you hold please while I investigate this error". But in fact, they were just resting, thereby extending call lengths.
3. When the customer service representative entered the service ticket, they forgot to ask what environment the customer was using (Windows, Apple, etc.) so the ticket was incomplete.

All of the situations above actually represent defects in other parts of the organization's work products that are not what we traditionally think of as software defects, but which can have a significant influence on the product quality and the end user's perception of quality

5 Conclusions and future work

In this paper, we have proposed a framework for modeling Enterprise Software Quality in Use (EQinU). In doing so, we have provided reasoning for extending the ISO 25010 quality modeling premise that phases in a product's development influence later phases and quality of later phases depends on earlier phases. EQinU extends this premise in the organization — into customer service and sales. As such, quality can be influenced, by areas of the organization other than software development and QA.

To illustrate the applicability of the proposed approach, examples from real clients in the field were presented to demonstrate the need to view software quality from a different mindset. It is no longer solely the job of development and quality assurance, but the entire organization, including sales and technical support to produce high quality software.

Ongoing research is focused on further utilizing the EQinU framework to model and understand the relationships among processes in an organization, their influence on product quality, and ultimately the end user's perception of software quality using measurement and metrics. In [6], we used survey methods to directly correlate quality metrics to end user satisfaction and we hope to extend that same

principle in this line of research. In the end, our goal is to discover that improvements or decline in performance in a characteristic of one organizational department influences the performance of another. Example correlations could include:

- Technical support call length and defects resolved.
- Time to fix a defect and number of calls to technical support.
- Customer needs documented accurately and completely by sales and product management's feature requirements.

The above are just examples of how one phase in the product lifecycle can have an influence on the end user's view of quality and that one weak point in an earlier phase can ultimately have impact on customer satisfaction and the end user's perception of quality. With this alternative paradigm in viewing quality, the amount of low hanging fruit may surprise you.

References

1. Deming W.E. et al, Total Quality Management, http://en.wikipedia.org/wiki/Total_quality_management.
2. ISO 9000, International Standard, Quality Managements Systems, 2000.
3. ISO/IEC 25010: Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models, 2011.
4. Lew P., Olsina L., Li Zhang, Quality, Quality in Use, Actual Usability and User Experience as Key Drivers for Web Application Evaluation, Lecture Notes in Computer Science, 2010, Volume 6189, Web Engineering, Pages 218-232
5. Lew, P. and Olsina, L., Instantiating Web Quality Models in a Purposeful Way, Lecture Notes in Computer Science, 2011, Volume 6757, Web Engineering, Pages 214-227.
6. Lew P., Qanber Abbasi M., Rafique I., Wang X., Olsina L.: Using Web Quality Models and Questionnaires for Web Applications Evaluation. IEEE proceedings of QUATIC, Lisbon, Portugal, 2012.
7. V-Model Software Development, http://en.wikipedia.org/wiki/V-Model_(software_development).