A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector

DANIEL RUSSO and PAOLO CIANCARINI, University of Bologna
TOMMASO FALASCONI and MASSIMO TOMASI, Deloitte Consulting

Information Systems Quality (ISQ) is a critical source of competitive advantages for organizations. In a scenario of increasing competition on digital services, ISQ is a competitive differentiation asset. In this regard, managing, maintaining, and evolving IT infrastructures have become a primary concern of organizations. Thus, a technical perspective on ISQ provides useful guidance to meet current challenges. The financial sector is paradigmatic, since it is a traditional business, with highly complex business-critical legacy systems, facing a tremendous change due to market and regulation drivers. We carried out a Mixed-Methods study, performing a Delphi-like study on the financial sector. We developed a specific research framework to pursue this vertical study. Data were collected in four phases starting with a high-level randomly stratified panel of 13 senior managers and then a target panel of 124 carefully selected and well-informed domain experts. We have identified and dealt with several quality factors; they were discussed in a comprehensive model inspired by the ISO 25010, 42010, and 12207 standards, corresponding to software quality, software architecture, and software process, respectively. Our results suggest that the relationship among quality, architecture, and process is a valuable technical perspective to explain the quality of an information system. Thus, we introduce and illustrate a novel meta-model, named SQuAP (Software Quality, Architecture, Process), which is intended to give a comprehensive picture of ISQ by abstracting and connecting detailed individual ISO models.

CCS Concepts: • **Information systems → Enterprise applications**; • **Social and professional topics → Management of computing and information systems**; • **Applied computing → Enterprise computing**; **Business-IT alignment**;

Additional Key Words and Phrases: Information systems quality, software quality, software architecture, software process, management information systems, delphi study, mixed methods

ACM Reference format:
Daniel Russo, Paolo Ciancarini, Tommaso Falasconi, and Massimo Tomasi. 2018. A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector. *ACM Trans. Manage. Inf. Syst.* 9, 3, Article 11 (September 2018), 38 pages.
https://doi.org/10.1145/3230713

This research has been partially funded by the Consorzio Interuniversitario Nazionale per l'Informatica (CINI) and the Italian National Research Council (CNR-ISTC). The authors thank all the panelists for their time and care in answering to our questions. The article was greatly improved by the suggestions of the anonymous reviewers.
Authors' addresses: D. Russo and P. Ciancarini, Computer Science & Engineering Department, University of Bologna, Mura Anteo Zamboni, 7, 40126 Bologna, Italy; emails: {daniel.russo, paolo.ciancarini}@unibo.it; T. Falasconi and M. Tomasi, Deloitte Consulting, Via Tortona, 25, 20144 Milan, Italy; emails: {tfalasconi, matomasi}@deloitte.it.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
2158-656X/2018/09-ART11 $15.00
https://doi.org/10.1145/3230713

1 INTRODUCTION

The quality and flexibility of Information Systems are among the most relevant sources of competitive advantage [104, 126]. While new digital "platform" companies arise, disrupting traditional business models and entire industries, established companies are striving to compete with these new entrants [101]. Incumbent organizations developed their information systems over years with long-lasting IT plans with mainframe-like systems [72]. Considering the level of pervasiveness of software, which is increasing at a path we have never experienced before, the digitalization of services and products are the new challenges of incumbents. While consumers are becoming more keen to use technology for their daily applications, businesses are rethinking the values they offer to customers and the corresponding business models for their competitive differentiation [17]. Thus, ISQ assurance is the grounding asset to meet customers' expectations. Indeed, the ISQ performance enables greater system reliability and flexibility, improving users' satisfaction and technology acceptance [132].

Information Systems research dealt widely with the notion of quality, intended as the sum of the high-level constructs of Information Quality, Systems Quality, and Service Quality [43]. The Software Engineering (SE) and Management Information Systems (MIS) communities provide complementary perspectives, addressing IT usage in organizations. The point is that they emphasize different activities and methods for controlling and improving ISQ. In fact, according to [61], the borderline between the quality of a software system and that of an information system is rather subtle, since software normally refers to programs, whereas an information system is the organizational context in which software is used. Although several frameworks have been developed to evaluate ISQ [87, 102], we did not find recent papers with a focus on the IT-related characteristics. We gathered several concerns in these years about *low information systems quality* of an ever-increasing number of financial applications. However, we did not find a comprehensive picture of the problem in the literature. This article addresses this contingent research gap.

As a matter of fact, the motivation for this study comes from industrial practice. Several domain experts raised concerns about critical issues, e.g., the growing complexity of the information systems, unjustified applicative layers and middleware stratification, difficult reverse engineering of their legacy software, and cost explosion. We started to explore effective techniques able to tackle the problems raised by the customer community. In doing so, we realized that a comprehensive model, to address informants' concerns related to the quality of their information systems, were still missing. Consequently, we started this journey to approach this research gap. This leads to major concerns about "what is inside" any financial information system.

Thus, we are interested to find suitable answers to the following research questions (RQs):

- RQ_1: What are the major IT-related concerns of the financial sector?
- RQ_2: Are these concerns shared among the community of experts of the IT financial sector?

In this article, we do not highlight technical problems and solutions *as such*. Our aim is to provide a valuable model to analyze and understand this compelling issue for the financial community. We identified the most relevant IT quality factors, investigating the concerns of several stakeholders, i.e., banks and outsourcing companies, system integrators, software vendors, and consultants. Our focus is industry specific. We have chosen the financial sector for the following reasons. First, a longitudinal study would have been too generic, whereas we wanted to grasp in-depth details of the phenomenon. Second, the financial sector is a traditional business, with highly complex legacy systems, which is facing a radical transformation due to market and regulation drivers. Finally, due to its industry structure, it is a quite homogeneous sector. In order to pursue the study, we surveyed the opinions of two panels of experts. For the panels' composition, we used the well-profiled

contacts of an established IT consulting firm. The panels included more than 100 senior IT financial experts who are mostly in top managerial positions. At the end of the research process, we gathered results that we consider highly valuable and generalizable. We used an innovative research method, based on the epistemological paradigm of Mixed-Methods research [37]. Finally, we report on the emergence of a meta-model, which links software quality, architecture, and process. Our study may be helpful in understanding in depth and with high internal and external validity (due to the Mixed Methods approach) IT quality and problems related to systems evolution and maintenance. For this reason, the first stage of our research is inductive, since we wanted to elicit the relevant items of concern related to ISQ. We used a research methodology that merged inductive research (through Delphi) with a deductive one (survey-like) to provide a comprehensive analysis of the problem. In these terms, we integrated both forms of data collection within the same research. So, we embedded one form of data within another to analyze different types of research questions [37]. Finally, we mapped the induced factors into the ISO standards 25010, 42010, and 12207, which refer to the three categories of software quality, software architecture, and software process, respectively. We argue that the developed model is of actual use, since it provides an abstract and general reference for ISQ, grounded in empirical evidence.

The structure of this article is as follows. In Section 2, we pursue the literature review, highlighting the research gap. Moreover, we explain our motivation along with IT financial market similarities, to motivate our research journey and its generalization. Then, we present the research design and details of the Delphi-like study that we conducted in Section 3. This is followed by a presentation of the results of our study in Section 4. In Section 5, we expose our findings and present the SQuAP (Software Quality-Architecture-Process) model, as well as implications for research and practice. The theoretical contribution of this study is discussed in Section 6. Finally, in Section 7, we conclude our article, discussing its limitations. Furthermore, we outline future research. Our main considerations are summarized after each section.

2 LITERATURE REVIEW

The definition of Information Systems is blurred and has changed in time [63]. Information Systems has referred to datalogical and infological systems [76], reporting and control systems [18], formal specified technical systems [123], inquiry systems [32], behavioral systems [45], sociotechnical systems [91], and human-activity systems [28].

Generally speaking, "Information Systems research is to study the effective design, delivery, use, and impact of information technologies on organizations and society" [71, p. 3]. Accordingly, our research focused on quality aspects of information technologies related to financial organizations, in the spirit of [61].

ISQ appeared to be from the first moment of our research journey a highly sensitive issue. This concern is also largely shared in the literature. Typically, billions of dollars are spent in IT projects since their success is an important competitive advantage of any organization [56]. Scholars argue that IT quality is the most important success factor of information systems: "Software quality can determine the success or failure of a software product in today's competitive market" [124, p. 84]. The quality of the software developed for information systems is an important source of competitive advantages for contemporary companies [106]. An analysis of the software engineering literature highlights the importance of product quality "in use" for industries, which strategically exploit software functions. The available literature also largely supports the view that a critical success factor for most of the initiatives in a domain like the IT sector is the information available to guide and support the management of software quality efforts [57].

The importance of the three dimensions of software quality, software development process, and software architecture to manage and evolve information systems has been extensively reported in

the literature [107]. Accordingly, the three related communities proposed significant advances to structure, assess, and improve the respective domains.

Therefore, we will now analyze the most relevant literature of the software engineering and information systems research communities, highlighting the research gap in Section 2.5.4.

2.1 Information Systems Quality

In the literature, information systems quality has been linked by DeLone and McLean to the concept of information systems success [42, 43]. It is composed of *system quality*, *information quality*, and *service quality*. System quality measures users' perception of adaptability, availability, reliability, response time, and usability [43]. Information quality relates to the content issue and can be measured with item completeness, ease of understanding, personalization, relevance, and security [43]. Finally, service quality measures the overall support delivered by the service provider with assurance, empathy, and responsiveness [43]. Subsequent literature explored and validated these constructs as key IS success factors [50, 103, 131].

However, these contributions strictly related to the individual perception and use of information systems. Their focus is on the *social* pillar, rather than the *technical* one [101].

In an epistemic view of a complementary contribution of the two Software Engineering and Management Information Systems communities to address IT usage in organizations [61], the technical pillar also needs to be solid and studied. Consequently, in the next subsections, we address such a research gap.

2.2 Software Quality

The most important reference for software quality is the ISO/IEC 25010:2011 standard on the quality of software products and their usage (in-use quality). This standard is based on eight characteristics that qualify a software product and five characteristics that assess its quality in use.

The empirical validation of quality characteristics has been carried out in the literature in several ways. Software quality assessment has been one of the first concerns of software-related literature [86]. A survey-based study of 75 end-users and developers to assess the standard structure was conducted in 2004 with ambiguous results [68]. In 2007, a similar study led to the same conclusions, which were helpful for the ISO/IEC 9126 standard revision [67].

The need to decrease software life cycle costs while at the same time enhancing software quality is a very well-known issue, which primarily leads the research on this topic [21]. Since the quality issue directly impacts companies, one of the first comprehensive software quality assessment models (SQM) was developed by NEC [121]. They used the Goal-Question-Metric (GQM) structure to define the Factor (e.g., correctness), Criteria (e.g., traceability), and Metric.

The Software Engineering literature also had growing expectations for integrated approaches to quality modeling. Scholars proposed using meta-models as grounds to develop a base quality model [129]. Notably, quantitative quality models have been developed taking into consideration several quality metrics, suggested by the standard itself [127, 128]. Nevertheless, several other software quality metrics tools have been developed [27], inspired by the ISO 25010 standard, introducing also new software-quality-related metrics like SQUALE [88].

The Management Information Systems literature gives a limited contribution to understand the software quality dimension. The quality construct relates usually to some definition of Product Quality [10], which we found is not very useful for assessment purposes.

2.3 Software Process

A structure for the software process life cycle has been defined by the industrial standard ISO/IEC 12207:2008 to outline the tasks required for developing and maintaining software [117]. Regardless

of the development methodology chosen (i.e., Agile or Waterfall [107]), this standard includes all the relevant concepts of the life cycle. Therefore, it is a useful structured tool for software houses, to assess if they have undertaken all recommended actions or not. Nevertheless, the software engineering literature also developed other process assessment tools, e.g., Software Process Improvement and Capability Determination (SPICE) [48]. Derived from ISO 12207 concepts, SPICE has become a new standard, i.e., ISO/IEC 15504. It was built on already existing software assessment methods, such as the Capability Maturity Model (CMM) by the Software Engineering Institute [99] and TRILLIUM developed by Bell Laboratories in Canada [7], among the most known ones. Rather than depicting the concepts of a software process, CMM focuses on the maturity of the process itself. In other words, it provides a framework to assess a process through six capability levels, ranging from 0 (ad hoc) to 5 (optimizing). So, it is possible to evaluate an organization's capability to deliver software artifacts through an objective assessment. Ad hoc processes are occasional and chaotic, without any proof that any kind of process quality was guaranteed; on the other side of the maturity spectrum, optimized processes are continuously improved through innovation, benchmarking, simplification, controlling, and change management [49].

However, the ISO 15505 is not a static framework. Several advancements have been proposed with the successor of the CMM, the Capability Maturity Model Integration (CMMI), which aimed to improve the usability of maturity models by integrating many different models into one framework [122]. Although SPICE is considered to be the leading reference for process assessment, several other frameworks have been proposed by scholars [125]. Still, the use of these other frameworks in industrial environments is doubtful [125].

Management Information Systems researchers usually do not focus directly on the software process as a model, but rather as a variable to understand other factors (e.g., Competitive Performance [93] or Project Performance [94]). High-level constructs like process flexibility [98] and process predictability [24] are used to define the software process, derived per analogy from the software engineering and manufacturing literature [94]. Nevertheless, there is a clear idea that the quality of software development processes is crucial for the performance of organizations and their competitive advantages [95]. Other scholars take as a reference the CMM model and try to build on that a higher model [41] to assess project performance. Generally speaking, the MIS literature sees the software development process as input for, e.g., project performance rather than as an outcome of different characteristics, like the software engineering literature does.

2.4 Software Architecture

The dimension of software architecture has a reference standard, ISO/IEC 42010:11. This is basically a *glossary* where the most relevant terms of software architectures are defined and put into context. Traditional software engineering literature is focused on the well-known suitable patterns for software design [116]. Hence, typical research in this domain is about how architectural patterns and guidelines impact software components and configurations [53]. A survey study analyzes in this perspective architectural patterns to identify potential risks and to verify which quality requirements have been addressed in the design [46]. With regard to software architecture evaluation intended as a way to achieve quality attributes (i.e., maintainability and reliability in a system), some approaches have emerged, mainly ATAM proposed by the Software Engineering Institute [14, 15, 36, 70]. More recently an approach based on quality requirements has been introduced, in order to offer guidance on the choice of the most appropriate method for an evaluation [9, 85]. Similarly, the practitioner's communities developed their frameworks; see, for instance, [4]. However, none of these techniques are related to the ISO 42010 standard. Apparently, the debate about the assessment of software architecture is less mature compared to the software quality and process domains.

From a Management Information Systems research perspective, several advances have been made in the literature regarding Enterprise Architecture Management assessment [6, 130]. Such a research stream focuses on the activities carried out in an organization to install, maintain, and develop the Enterprise architecture, to deal with its different architectural layers and foster a holistic and integrated view of the enterprise IT architecture [6]. This approach studies the interaction between the technological components of the information systems and the organization, to achieve common business objectives [69]. While this framework initially defined a quite conceptual perspective [23], recently empirical studies are gaining momentum [130]. High-level constructs (i.e., Product Quality, Infrastructure Quality, Service Delivery Quality, Organizational Anchoring, Intention to Use, Organizational and Project Benefits) are empirically assessed [75] to depict the level of IT-business alignment [79].

2.5 Relationships among Dimensions

With regard to the mutual relationships of the three dimensions of software quality, process, and architecture, we found scarce empirical literature. Most papers refer to the *obviousness* of the interaction of these dimensions, providing little evidence for such statements, like "it would be extremely unusual to find a high quality software system with a poor design" [81, p. 471].

2.5.1 Process and Quality. Software quality and process are among the most studied issues. The very idea itself of process maturity is closely related to quality, one of the most relevant issues studied by the Software Engineering Institute at Carnegie Mellon [64]. Like we have seen before, such research efforts developed the Capability Maturity Model [99] and its evolutions [31]. This kind of interaction is also the most evident. In fact, recent literature lets emerge empirically this kind of relationship [112]. Similar studies noticed how this kind of relationship is magnified with a Continuous Development methodology [108]. In models like CMM, architecture is considered exogenous. It is taken for granted, and thus the model focuses on the development process to enhance quality. Moreover, it also has to be noted that the effort to structure software development in order to be in control of its quality went back to the 1960s by the US Department of Defense (DoD). Accordingly, SEI was started up at Carnegie Mellon by the US military in the 1980s, where the first version of CMM was published in 1988 [64]. The first architecture standard IEEE 1471, on the other hand, was issued in August 1995. The reason might be that software architecture, as a discipline, started to rise years later with respect to process and quality, which were considered more important.

2.5.2 Process and Architecture. The relationship between software process and architecture has been studied in several ways, mainly as a consequence of the process model. According to this view, the architecture supports a structural decomposition of the development cycle into tasks, and the decomposition continues until each defined task is performed by an individual or single management unit [64]. The idea is still that of an exogenous element. Similarly, iterative [19] and noniterative [111] software process models consider the software design as an element of the process. Notably, with the Twin Peaks model [96], software process and architecture are on the same level, since one influenced the other. It is also the theoretical ground for Agile architecting [35]. Consequently, with the Agile movement, the differences between the development phases narrowed [62]. So, software process and architecture were considered as one comprehensive issue [8]. However, recent studies show how this idea was poorly adopted by the Agile developers, since none of the architecting approaches have been widely used in combination with Agile practices [134].

2.5.3 Architecture and Quality. Regarding the relationship between software quality and architecture, some theoretical contributions were made to understand how an architecture is inspired and even driven by quality concerns [70]. A good reference for this relationship and its value in practice is Bass et al. [12]. A recent book highlights the importance of software quality analysis for software architecting [85]. Still, this remains a relationship actively in evolution, especially in the context of the novel proposal of DevOps processes for cloud and microservice architectures [13].

2.5.4 Quality – Process – Architecture. To the best of our knowledge, no comprehensive model to study information systems as the triple interaction of software quality, process, and architecture has been developed in the literature, identifying a clear research gap.

In the last decade, there has been a considerable effort, especially by the Management Information Systems research community, to study the phenomenon of the alignment of business and information system's architecture [5]. What emerged is the importance of such alignment for both businesses' competitiveness and technical efficiency. In fact, when it comes to integrating new solutions, modules, or interfaces, such alignment is of key importance. Several other scholars found similar results, suggesting the importance of standard governance defining key architecture roles, involving key stakeholders through liaison roles and direct communication, institutionalizing monitoring processes, and centralizing key IT decisions [22].

Especially in the financial sector, architectural governance is a key issue for IT efficiency and flexibility [113]. Generally speaking, this finding is also largely shared beyond the financial sector [75]. The need for people from different backgrounds (mainly business and technical ones) to align the organization is the greatest insight of this research stream. This pattern has been first theorized by DeLone and McLean [42] and then revised in 2003 [43] by the same authors. A recent literature review shows how in over 90 studies this pattern has been empirically observed as successful [102].

In this regard, special tools also have been developed to monitor the alignment and coevolution of business process and enterprise software systems to estimate the change propagation caused by a change request in business processes or software systems based on the software architecture and the process design [109, 110].

2.6 The IT Financial Market

Information systems' effectiveness in general, and quality in particular, has always been considered by the MIS literature a key strategic issue for the financial sector [84]. We recognize that key determinants of banking service quality, which includes ISQ, have been proposed by the literature [66]; however, a general framework for ISQ of this crucial sector is missing.

To improve readers' context understanding, we pinpoint the two most relevant homogenization drivers of this sector: market and regulation.

From a market perspective, cost-cutting needs are the same at least for EU and US banks due to low interest rate levels set by the European Central Bank (ECB) and the Federal Reserve (FED) and insolvency issues in the mortgage market. This reduces sensibly banks' margins and profitability, leading to generalized internal cost cutting. The maintenance and evolution of information systems of financial organizations are influenced by such business goals by delivering within a lower budget (and time, due to fast regulatory changes). The same custom applications and software packages are implemented in European banks because software vendors sell standardized industry solutions. The COTS (Commercial Off-the-Shelf products) market is quite similar worldwide since it offers industry solutions to get benefits from the low marginal cost of software [39]. Typically, if one competitor offers one new application, others also are willing to follow in order to avoid losing crucial market shares. Finally, banks are among the most globalized businesses, so the

same information systems are shared among more branch countries. For this reason, they share the same problems.

From a regulatory perspective, similarities become even more clear. Currently, the most important financial regulations are standardized within the European Union, e.g., MiFID, and the European financial authority is coordinating the national authorities, so the regulation differences among countries are decreasing. IT concerns among banks are quite similar, since most regulation is provided by shared European financial regulators. More precisely, all the efforts in the direction of a European financial framework are pushing information systems of financial organizations toward new issues, like the Payment Service Directive (PSD 1) [2]. Since the goal is to integrate a Single Euro Payments Area (SEPA), major efforts are asking European banks and their information systems to interoperate properly, leading also to new financial business models. In this regard, the Payment Service Directive 2 (PSD 2) is offering novel architectural challenges, since it will allow third parties, through banks' APIs, to establish new payment channels with the customer's account, breaking the traditional monopoly of the bank over its own channels [3]. In other words, third parties will have access to a customer's account via APIs to connect a merchant and the bank directly. This will dramatically impact in the near future the financial information system architecture, leading to new payment service models [119]. In this regard, the European financial authority (EBA) recently issued the *Regulatory Technical Standards on Strong Customer Authentication and Secure Communication under PSD2* [1] for enhancing the security level of payment services across the European Union. These technical standards, as do other regulations, compel banks of the Euro Payments Area to comply with uniform protocols and procedures of their information systems to support the *Free Movement of Capital* EU principle, according to articles 63 to 66 of the Treaty on the Functioning of the European Union (TFEU).

Moreover, since banks are de facto global businesses, as their mission is to operate worldwide (e.g., global wire transfer), they have to comply with common standards. The Bank for International Settlements (BIS), whose purpose is the collaboration among central bank authorities, is the most relevant organization that supports global financial and financial standardization. BIS also supported the development of crucial industrial standards for the financial sector. Just to cite a few, we remember the ISO 19092-1 Financial Services - Biometrics - Part 1: Security framework and ISO/IEC 15944-8:2012 Information technology – Business operational view – Part 8: Identification of privacy protection requirements as external constraints on business transactions. Those standards have been developed to support modeling international requirements for identifying and providing privacy protection of personal information throughout any kind of information and communications technology (ICT)-based business transaction.

3 RESEARCH DESIGN

As researchers, we have our epistemological biases, which usually remain hidden or implicit, even if they deeply influence our research [118]. When we approached this research, we decided to use *Pragmatism*. It derives from the work of Peirce, James, Mead, and Dewey, and arises out of actions, situations, and consequences rather than antecedent conditions (as in postpositivism) [29, 11]. It focuses on the research problem rather than the method. This is the philosophical ground for the Mixed-Methods approach.

These four scholars paved the epistemological ground of our approach. Although with different points of view, they complemented the pragmatic approach.

According to Peirce (1839–1914), human concepts are defined by their consequences [100]. So, people's independence of will to decide the actions to undertake is crucial in any experimentation. Ideas and concepts are the main drivers for truth.

On the other hand, James (1842–1909) highlights the pluralistic view, which recognizes the multiplicity of human truth. Due to his psychology background, he argued that human thought may be revealed only in action.

The Logic of Controlled Inquiry was proposed by Dewey (1859–1952) [44]. He argued that reasoning by itself cannot provide change. Only the combination of action and reasoning reorders the setting.

Finally, Mead (1862–1931) suggests that any human action is socially reflective [80]. Any human behavior that elicits responses from another individual constitutes a social act. The social consciousness is the reflection of ourselves, mirrored in the reactions of others.

The pragmatic view does not commit to any system of philosophy and reality. Like Mixed Methods, both quantitative and qualitative assumptions are used. This is related to the view that the world has an absolute unity. Truth is what works at the time; it is not based in a duality between reality independent of the mind or within the mind [37]. Therefore, in Mixed Methods research, investigators use both quantitative and qualitative data because they aim at the best understanding of a research problem [37].

This article reports the results of a Delphi-like study modeled on the Delphi methodology (qualitative) and survey (quantitative). The first research phase has been devoted to address RQ_1, to let emerge the most compelling IT quality concerns. In the second phase, the elicited concerns were validated by the expert community at large to give an answer to RQ_2. In order to investigate the concerns, we used a phenomenological approach [89]. Moreover, the survey is a semistructured one, so besides closed questions based on a hybridized Likert scale, experts are asked to express openly their opinions about any single item. Thus, the survey itself is mixed quantitative and qualitative.

Generally speaking, empirical software engineering is developing new research designs according to its research questions, which may cross the threshold of traditional borders of the discipline due to the pervasiveness of software [34, 33]. The Delphi method is becoming a popular tool in the software engineering discipline [73], even though it is still not well known. On the other hand, Likert-based surveys are a popular quantitative research method in information systems [30]. We first present a brief discussion of the Delphi method, since it is the underpinning method of this research. Then we discuss how the Delphi panel was selected and provide details of the Delphi process, i.e., threats validation and the survey one. Finally, we discuss the outcomes.

3.1 The Delphi-Like Method

The Delphi method has proven to be a popular tool in Software Engineering [73] and, more generally, in Information Systems research [25, 115, 60, 90, 115]. It allows for capitalizing on the experiences of the expert panel in identifying key issues of software developers and identifying the most important factors by continuous feedback. The objective is to create valuable information through a structured process of knowledge collection from a panel of experts with controlled opinion feedback [47]. The process consists of a series of rounds in which each expert communicates his or her opinion through a structured form, i.e., questionnaire, structured interview, collected by the researcher. It is an inductive data-driven approach, ideal for explanatory studies for which little empirical evidence is available [59]. Using a step-wise methodology, the research question is narrowed down, from a multitude of issues, to a bunch of a few consensus-based factors [114].

After gathering experts' concerns on the IT financial sector, we consolidated it into 28 factors (RQ_1) with a Delphi-style methodology. Then, we collected opinions and evaluations about the factors with a "target panel" of 124 profiled experts (RQ_2). The administration of the whole process is represented in Figure 1.

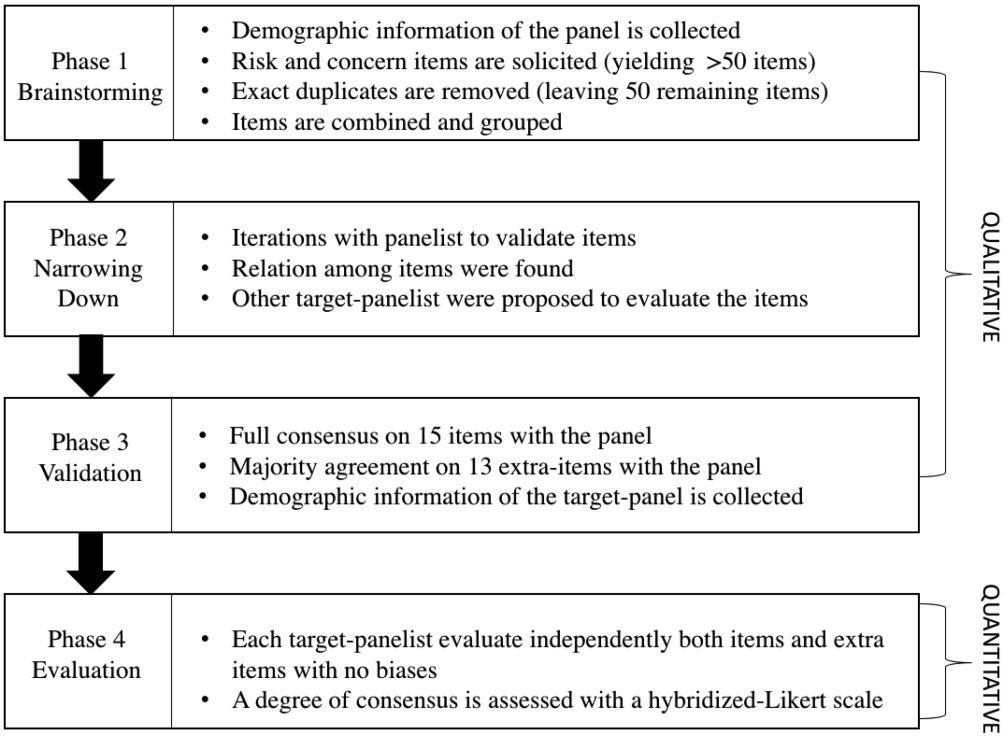

Fig. 1. Delphi-like administration process.

3.2 Selection of Delphi Panelists

We are aware that the selection of panelists is the most critical aspect of Delphi, especially with respect to the validity threats: a lousy panel selection may compromise the whole qualitative research.

Thus, we mitigated this risk exploiting an established private dataset of an affirmed consultant firm; moreover, the use of a double panel using Mixed Methods enforces the validation of both the questions and the answers. However, it still remains *the most important yet most neglected aspect of the Delphi method* [97]. The methodological literature agrees on the fact that the choice of experts is the single most difficult factor in panel selection [65].

Hence, our greatest efforts were devoted to the selection of the first and then the second panel. Since we intended to perform a vertical study (i.e., sector specific), we put a lot of care into the choice of experts invited to participate on the panels.

We started from a privileged position. In fact, we could use an expert pool of an established IT consulting firm, specialized in the financial sector, which works with all main financial groups. So, we were able to address personally highly qualified experts from the IT financial industry. This makes our panel highly reliable and representative.

The first panel was chosen using a stratified random sampling. Strata were defined upfront, to define the sample population. Companies and roles were chosen to have a fair representation of the IT financial sector.

The first panel did not mirror the population representation task regarding experience, since we addressed explicitly senior experts, in agreement with [65]. Also the target panel was chosen with a stratified random sampling, but inside a larger pool.

Table 1. Panel Composition

		#	%
Company	Consultants	7	25%
	Bank	6	21%
	System Integrator	6	21%
	Outsourcing	5	18%
	SW Vendors	4	14%
	Total	28	100%
Experience	11–20 years	4	31%
	21–30 years	8	62%
	More than 30 years	1	8%
	Total	13	100%
Role	CEO/CIO	6	29%
	Chief Data Officer	5	24%
	Appl. Maint. Group Exp.	4	19%
	IT Architect	4	19%
	Maintenance Manager	2	10%
	Sales	0	0%
	Total	21	100%

We asked panelists to give their opinions about the sector composition (companies and roles) and also the name of other experts. Therefore, we adjusted the sample population representation along our research journey, following a suggestion found in [37]. However, also for the target panel, we looked for more senior opinions, adapting the research method to our purposes [38].

In Table 1, we describe the demographic composition of the first panel, and in Table 2 that of the target panel. We profiled our panels in great detail, much more than traditional Delphi studies. The guarantee of anonymity we gave to all experts allowed a very open and truthful discussion.

We took into consideration three dimensions: (1) years of experience, (2) sector in which the experts worked, and (3) relevant roles they served in their careers. Obviously, sector and role may be more than one per expert, since it is normal to change jobs during any career path. In detail, these are the dimensions and subdimensions surveyed:

Experience. For the Panel composition, we used only senior informants. Thus, we had three groups, composed by experts with more than 10 years and less 21 years, between 21 and 30 years, and more than 30 years of experience. We divided the experts into five groups in the target panel: less than 5 years, between 5 and 10 years, between 10 and 20 years, between 20 and 30 years, and more than 30 years of experience. The greatest majority of our experts have more than 20 years of experience in the sector. Thus, we consider the opinions of our panel of high value.

Companies. We also profiled different company types involved in the IT financial sector. Most experts declared some experience as interns within a *bank*. The companies second by relevance are *system integrators*, since a high degree of customization is needed for bank products. *Consultants* and *outsourcing companies* are also very important actors in the IT financial sector, since most work is outsourced to external workers or companies. *Software vendors* deliver software packages products for banks. As stated before, software packages are commonly used by small/medium-sized financial institutes to either address specific needs (e.g., loans, deposit or transaction accounts) or

Table 2. Target-Panel composition

		#	%
Company	Bank	72	35%
	System Integrator	41	20%
	SW Vendors	34	17%
	Consultants	33	16%
	Outsourcing	23	11%
	Total	203	100%
Experience	0–5 years	8	6%
	6–10 years	1	1%
	11–20 years	23	19%
	21–30 years	72	58%
	More than 30 years	20	16%
	Total	124	100%
Role	Appl. Maint. Group Exp.	59	39%
	CEO/CIO	32	21%
	IT Architect	18	12%
	Chief Data Officer	18	12%
	Maintenance Manager	13	9%
	Sales	8	5%
	Other	2	1%
	Total	150	100%

support common processes (e.g., accounts payable and receivables). According to our experience, the composition of both the panel and the target panel represent a trustful representation and distribution of companies in the IT financial sector.

Role. The *Chief Executive Officer (CEO)* is the top manager of software integrator or vendor companies, while the *Chief Information Officer (CIO)* is a board member in charge of the information system of the bank. The *Maintenance Manager* is in charge of the development and maintenance of IT financial systems. The *Application Maintenance Group Expert* is responsible for complex groups of IT financial applications. The expert in charge of the IT financial architecture is the *IT Architect*. *Sales* is a senior manager of the sales department of a software vendor or system integrator. Finally, the *Chief Data Officer* is an expert in industry standards and methodologies (e.g., IEEE, ISO, ITIL, DAMA) and in charge of the bank's data governance. Also, the distribution of roles within both the panel and target panel is representative of the IT financial sector, to the best of our professional knowledge.

The methods we used to compose the two panels varied slightly. For example, we took into consideration for the target panel the role of sales managers, not considered before. We had feedback that their role is also critical for the IT financial sector, so we adjusted it along the research process (following [37]). Senior experts with at least 10 years of experience within the sector were chosen for the first panel. In the target panel, we wanted a wider and more trustworthy representation of the sector, so we also included junior people in the target panel. However, about two-thirds of the target panel had more than 20 years of experience.

Most of our panelists and target panelists experienced more than one position in more than one company. We did not find it meaningful to show just, e.g., their last job or the longest one. Experts

expressed their opinion according to their general knowledge of the domain, which was gained through different professional experiences. This enriches the research, since experts were able to judge the factors from different, highly qualified viewpoints within the same sector. This complies with our epistemological paradigm and research approach.

3.3 Data Collection and Analysis

The Delphi-like study took over a year for the four phases (see Figure 1), which are described next. According to the Mixed-Methods approach, this research includes both qualitative and quantitative approaches. In order to develop the qualitative part, the first three phases focused on identifying the relevant items. Then, we used a survey-based approach to validate the panel's items by a wider panel (target panel).

The qualitative research started in October 2014 and lasted until April 2015. The survey took less time, from June 2015 up to November 2015.

Phase 1: Brainstorming. After collecting the demographic information from our pool and composing the panel, we conducted a brainstorming round to elicit as many concerns as possible. This phase was helpful to broadly understand the problem and seek concerns. It was an unstructured process, where more than 50 items were solicited. Afterward, exact duplicates were removed, leaving 50 items. Finally, items were combined and grouped by the authors.

Phase 2: Narrowing Down. In the second phase, further iterations involved the panel to validate the items. Relations among items and their grouping were discussed, again in an unstructured way. Experts were asked to discuss the grouping made by the authors of this article and to redefine it, if needed. Items were represented in sticky notes on a blackboard. This helped the discussion over the items and their relations. Afterward, authors wrote down the items list, in the terms discussed by the panel. Finally, the panelists discussed the target-panel composition and also proposed other experts to invite.

Phase 3: Validation. The goal of the third phase was to validate the items. The panel considered more than 30 concerns, but it came finally to an agreement about the 28 concerns we will discuss in this article. Since we were able to have all our panelists within one room, there was no need to use statistical techniques to assess their level of agreement. Typical Delphi studies, where informants are distributed, ask panelists to rank concerns according to their importance and then aggregate them through, e.g., Kendall's coefficient of concordance (W) to assess the degree of consensus among the panelists themselves [114]. We had a narrow group of highly qualified experts and managed them as a working group. The introduction of statistical methods for the validation of the final items was not grounded in our phenomenological approach. We wanted to grasp the essence of their experiences as described by the participants [89]. At the end of this phase, a full consensus was reached on 15 items. The panelists did not always agree on the degree of such consensus (i.e., *strongly agree* or just *agree*). However, the baseline (i.e., *agree*) was always met for each item.

There were then 13 more items that did not reach full consensus but were strongly advocated by some panelists. Although both brainstorming and narrowing-down phases involved a lot of personal confrontation, these 13 extra items were either advocated or opposed by at least one panelist. While some panelists supported one item, some others argued that for their respective industrial segment the proposed item was not relevant. Therefore, during the validation phase, there was a consolidated opinion about the first 15 items and mixed feelings about the 13 extra ones.

We were interested also in evaluating the 13 extra items. We included them in our survey since we wanted to evaluate if the panelists' opinions were shared by the financial community. Finally, some demographic information about the target panel was collected.

Phase 4: Evaluation. The last phase concerned the quantitative inquiry approach, to evaluate the concerns by a larger stratified sample group (target panel). We prepared an online survey with the concerns and made personal invitations to experts. Target panelists could start the survey only after they inserted their personal credentials, given by email or phone. To have control over the research and the randomized stratified sampling, no answer was anonymous. We used a hybridized Likert scale for the concerns' evaluation. The aim was to compute semantic differentials (i.e., "Strongly Agree," "Agree," "Disagree," "Strongly Disagree") to define the level of agreement, typical of Likert scales [77]. We hybridized the Likert scale with even bipolar values (negative and positive), symmetric to 0, without an average effect. To stress possible differences, we gave higher values to both extremes, also to avoid an average effect. So, we assigned the following values: "Strongly Agree" = 3, "Agree" = 1, "Disagree" = −1, "Strongly Disagree" = −3. The idea was to highlight different semantic values of the two extremes and suggest equidistance between the center point of the scale and the two extremes. To overcome the bias that can result from the order in which items were presented, we randomized the questions and did not tell the participants about the difference between items and extra items.

4 RESULTS

The Delphi-like study resulted in a set of 15 concerns presented in Table 3 along with the target-panel evaluation. All concerns were shared, at least, by 70% of the target panelists with a different intensity degree (measured with the hybridized-Likert scale). In Table 4, we represent the 13 extra items. Interestingly, all these extra items have a lower share degree than those in Table 3. The reason may be that the mixed feelings of the first panel were also shared among the target panel. This generally confirms that a high consensus degree of a Delphi panel also reflects an eventual validation. However, we found the concerns that emerged from Table 4 of interest and show them separately.

To improve our theorization, we abstracted the explicit informants' concerns into corresponding factors. We are aware of the fact that dealing with 28 factors provides a huge amount of knowledge and is not really parsimonious. However, we only dealt with unique items. This means that for our panelists, the 28 concerns had to be treated differently. Thus, to be consistent with our research design, we had to managed them separately. Finally, according to [51], in order to broaden the generalization, we pursued a mapping of the concerns into high-level constructs proposed by industrial ISO standards.

Each factor is summarized followed by a short discussion where panelists' and target panelists' opinions are directly quoted. For the sake of readability, only in this section, we use the terms "panelist," "target panelist," "informants," and "experts" as synonyms.

Factor 1: Module interface complexity. A financial information system is characterized by a high number of modules; if these are strongly coupled, this increases the number of interfaces and their complexity.

One panelist mentioned the spaghetti-like architecture, stating that "we are experiencing a growing complexity in delivering new projects due to past implementation of point-to-point architectures delivered in the last years." Integration costs and higher complexity compared to a green field make the business case less effective. Even worse, another one said that "sometimes the interfaces to be updated are so complex that an ad hoc middleware is required." The lack of knowledge seems to be another root cause since a third panelist said, "I believe that the lack of knowledge about application functionalities led to code duplication over the years."

Factor 2: Interface architectural complexity. This second factor is a direct consequence of the first one. Module interface complexity led to a typical antipattern [26]. According to a panelist,

Table 3. Concerns and Results of the Delphi-Like Study

#	Concern	"Agree" and "Strong Agree"	Average Score
1	Software modules interfaces are characterized by a high level of complexity and represent an important part of each module in terms of number of objects and LOC.	96%	1.95
2	Interfaces among modules are developed in a stratified way in time. This led to a complex architecture hardly manageable and not future-proof.	96%	2.47
3	Software quality for custom development is decreasing in the last years (especially Cobol/CICS/DB2).	77%	1.15
4	SW maintenance & enhancement evolution costs and time of information systems are increasing due to the (i) stratification of software, (ii) poor documentation, and (iii) low quality of the source code.	82%	1.58
5	Low software quality depends on increasing pressure for enhancement evolutions in shorter time with a lower budget.	79%	1.57
6	Low software quality depends on a poor level of functional and technical analysis and detail.	79%	1.00
7	System analysis is hindered by an inadequate documentation and database.	87%	1.70
8	It is difficult to build and maintain effective documentation because of low budget and time shortening.	84%	1.46
9	New software packages have more functionalities than in the past but their increased complexity leads to difficult evolution management.	83%	1.58
10	Software packages are poorly documented for effective maintenance and evolution.	82%	1.30
11	Software packages documentation main gap is due to a poor description of the data managed by the system (i.e., not only the record layout but also the fiscal and logical data model, the data dictionary).	77%	1.09
12	"Application Maintenance" contracts do not improve the software documentation.	77%	1.15
13	International software applications are of higher quality but are not per se more maintainable.	76%	1.29
14	Domestic software applications are characterized by more functionalities and lower quality without any significant impact on software maintainability.	75%	0.92
15	Software quality cannot be reliably measured through tools and methodologies.	70%	0.93

"stratified software interfaces affect old developed applications; only using Service Oriented Architectures we took advantage of the strong benefits related to an integrated architecture." Legacy layered software is a remarkable problem, as stated by another panelist: "there is a well-known problem related to platform software, which was developed years ago and never replaced. Only a tactical update with a short run perspective" was carried out. Continuous update of legacy layered software seems an attractive and affordable way, but long-term sustainability is questioned. Most concerns were related to the maintainability of such architecture. No refactoring solution was

Table 4. Extra Concerns and Results of the Delphi-Like Study

#	Concerns	"Agree" and "Strong Agree"	Average Score
16	Lower software quality is related to decreasing skills of IT professionals.	48%	0.04
17	The use of external developers increases the difficulty to assess developers' skills.	66%	0.71
18	Software engineering methodologies and tools to assess software's quality are not reliable.	70%	0.93
19	It is hard to agree with software vendors on software engineering methodology to enhance quality and to assess it.	66%	0.57
20	Lower IT professionalism depends on the poor use of software engineering methodologies due to shrinking IT budget and time.	72%	0.94
21	Lower IT professionalism depends on decreasing professional rates.	58%	0.62
22	Web technologies are developed with less software engineering rigor, also misinterpreting the Agile paradigm.	70%	0.85
23	Low software quality depends on unclear user requirements.	72%	0.86
24	Unclear user requirements depend on poor business & IT elicitation processes.	63%	0.75
25	Unclear user requirements depend on different "jargon" of IT & business departments.	58%	0.59
26	Poor data analysis (in terms of data modeling and data structure) influences directly the overall functional analysis.	70%	0.93
27	Fine-granular functional analysis is hindered by poor data modeling and data structuring.	73%	0.98
28	There are no effective software documentation methodologies and tools.	61%	0.36

seriously taken into consideration, due to cost. However, this short-term view did not decrease costs because it is very likely that the system will stop working properly in a medium period and the replacement cost could be very high. Moreover, this leads also to unexpected problems that usually rise with such complex systems. A third panelist said, "we decided to replace the client data module and its interfaces, since we reached a point where we were unable to manage the evolution required by the business." Due to the high degree of coupling of these modules (Factor 1), proper reverse engineering is required.

The next factors will show how the lack of documentation hinders reverse engineering. All these elements make it difficult to improve the bank's information system.

Factor 3: Custom software quality. Apparently, the quality of custom software applications is decreasing. Moreover, several modules were developed with old programming languages like COBOL, which are still widely adopted in the financial industry, while there is a lack of junior experts because such languages are not included in the current formal IT education programs.

A panelist explained that "the number of developers able to develop in COBOL is rapidly decreasing due to retirement. New developers are not skilled enough with COBOL and other mainframe languages because they are focused on the newer languages like Java."

The interaction among old and new coding paradigms is another point raised by the panelists. One said that "software quality is getting worse because we developed using a stack paradigm,

adding new software adapting layers on old software. Unfortunately, this paradigm prevents the use of new technologies." Furthermore, the decrease of quality "is perceived also in all other used programming languages," according to other panelists. This may also depend on the actual training of developers. According to one expert, "several developers we hired did not go through a formal computer science education." This may be due to the high demand on the job market of software developers who, however, get low salaries. Skilled developers usually have many job offers and tend to choose the most profitable one.

Factor 4: Increase of maintenance costs. Some factors have a direct impact on maintenance costs. The overall architectural complexity, the decreasing software quality, and incomplete documentation are the most important drivers of high maintenance costs and time. As declared by a panelist, "during the last six years our software modules have been impacted by a deep reengineering project and the most heavy effort was related to building new documentation."

Another reason for the growing costs seems to be linked to skills: a panelist pointed out poor competencies as a primary cause of frequent module rebuilds that cause an increase in the application complexity. This is related to the use of different technologies: "it often happens that instead of updating the software, new applications are built on it. The coexistence of different technologies is an important cause of high maintenance costs." According to one panelist, "most costs are related to continuous regulatory changes requested i.e. by the ECB." This is apparently another element of stratification and architectural complexity.

Factor 5: Quality versus time and budget. The whole panel agreed unanimously that there is a direct relationship among quality and time and budget. One expert stated that "a relevant cause of poor software quality is related to time constraints, these aspects have impact on quality." More time and budget to develop and evolve software properly would increase quality and decrease maintenance costs. In fact, "an already well-written code could be evolved with low budget, while a stratified software which has been poorly written makes updating difficult, increasing costs and time." This is a chain effect; one said that "poor software quality is related also to software stratification due to low investments and the obsolescence of the information system. Poor implementation of software engineering methodologies due to low budget magnifies this crucial issue day after day." Another element that emerged is the relationship between the organization structure of banks and its impact on the information system. One panelist stated that "it is of high relevance the organization structure of the customer to define the proper information system." Apparently, this element impacts on the quality of the system itself.

Factor 6: Quality versus system analysis. Even though the design phase is perceived as the most important upfront activity, it is poorly implemented. One panelist stated that "it is necessary to invest in this phase, to get benefits within the whole life-cycle." However, "it is poorly carried out, due to shrinking budget and time," said another informant. The problem may depend on the fact that often the role of IT departments is not perceived as highly critical by top management. So, "there is not an adequate collaboration between business functions and the IT departments." Therefore, system analysis activity is often skipped because it is hardly tangible. Stakeholders are not willing to invest in some activities where they do not see an immediate return. This leads to long-term problems, as identified before. Another element is that the formalization of the business requirements and a complete and effective vision of the information system are difficult. In this regard, one panelist affirmed that "customers usually give poor and not coordinated requirements, leading to silos-like solutions instead of a fully integrated development."

Factor 7: System analysis versus documentation. Inadequate documentation impacts on the system analysis and so on software quality. One expert stated that "documentation is inadequate

to the scope, being or too technical or too business-like with poor information abut the system." Moreover, a wrong interpretation of Agile methodologies results in poor documentation; in fact, "along with the stratification problem the misleading interpretation of Agile led to a poor and ineffective documentation." It is important to underline that the lack of data models' documentation and a meta-data definition "leads to an inadequate and dangerous system analysis."

Factor 8: Documentation versus time and budget. Time and budget constraints have a direct impact on software documentation. Due to low budgets for new developments and urgency for new applications, documentation is the first element that is skipped. A panelist said that it is impossible to keep documentation aligned with software both for the frequency of software updates and the lack of methodologies used to develop and manage software. According to another panelist, "constraints on project costs and time causes poor or no documentation. In fact, the documentation is usually delivered only if you have enough budget." Due to budget limitations, often banks prefer to skip documentation if they are offered a discount on the application cost. Also, time plays an important role. Often, there is no time for documentation, or, even worse, it is perceived as a waste of time. A panelist explained that "budget constraints clearly impact on documentation, since we are not able to justify the budget required to keep documentation aligned. The only affordable way is to insert control points on the software development process (SDLC) to define the least amount of information necessary for maintenance." In this regard, documentation tools appear to play an important role. Another panelist declared that "we can limit the problems that we have in software documentation thanks to the adoption of new generation tools (thanks i.e. to metadata) and the Agile approach."

Factor 9: New packages functionalities versus complexity. For the reasons analyzed before, the demand for more functionalities rose in the last years, along with their complexity. Moreover, software vendors do not always apply industry standards. One panelist explained that "new software packages require methodologies and standards that only few big vendors can adopt. In this situation when we want to customize those packages we must rely on those big vendors but with a low degree of control and the danger of lock in." Some experts also said that "recent packages are too complex and have lower quality than in the past." This factor explains the relationship between the market trend, i.e., more functionalities with architectural complexity (factor 2) and dependency on vendors (factors 10 to 12).

Factor 10: Packages versus documentation. The lack of documentation for software packages is perceived as a "commercial strategy of suppliers to lock-in customers." This appears natural, since "the development is often given to software houses." For one expert, "documentation is always lacking" and it is not uncommon to "buy packages without any kind of technical and functional documentation." Another reason may be the fact that "suppliers tend to hide technical documentation as IPR protection strategy, delivering only the functional documentation." However, as another expert said, "this problem needs to be tackled with a good Service Level Agreement," according to an expert.

Factor 11: Packages documentation versus system analysis. The lack of documentation in packages impacts directly on the logical data model and quality controls. As stated by a panelist, "information about data is one of the biggest problems, as well as the role that data plays on business lines." Moreover, "process logic is needed to achieve a correct level of documentation. Just data are not enough." Also, this factor suffers from low budget and scarce time.

Factor 12: *Application & Maintenance* contracts versus documentation. Application & Maintenance (AM) contracts are set to outsource the development and maintenance, to decrease internal costs. Typically, they do not provide an adequate documentation. Therefore, when the

supplier is changed, system evolution becomes rather difficult. Lock-in situations are very common since "suppliers want to defend their know-how to maintain their competitive advantages." Like factor 10, "a good Service Level Agreement is key to overcome problems." However, what happens is that SLAs are not respected properly, to get higher discounts on services.

Factor 13: International applications versus quality and maintainability. According to the panel, there is a difference between domestic and international software products, which is partially a concern. Apparently, international applications are more maintainable but have fewer functionalities. The reason seems to be that "international applications are less flexible than domestic ones because they implement simpler functions and not because they are written or designed better." One panelist stated that for a well-known, specific software application, "a low level of customization usually means lower license and maintenance costs."

Factor 14: Domestic applications versus quality and maintainability. Regarding domestic applications, they appear to have more functionalities but incur higher maintenance costs. For one panelist, the reason seems to be that "Domestic applications are really rich of functionalities due regulatory requirements defined by the financial Authority." One expert expressed a specific concern, stating that "software applications should comply with international standards. Before the acquisition each customer has to test this compliance. In reality, this never happens, and if it were the case most would fail. The only exception are applications delivered to NASA, US Air Force and so on, but at which costs?"

Factor 15: Measurement of software quality. Losing control over the system quality is a concern. First of all, "poor use of software engineering methodologies are the first killer of quality," according to one panelist. According to another expert, "even though methodologies are well-known and some tools are available, they are not implemented within the software development process." The discussion about tools was quite interesting. "Tools do exist but are extremely expensive and not suited for small banks," stated one panelist. For another informant, "a tool suited for us does not exist on the market place, so we built one ourselves." Finally, an expert very frankly explained that "tools and methodologies are well-known, however neither customers nor suppliers use them since none is willing to pay for them."

Factor 16: Lower developers' expertise and professionalism. This was one of the most debated factors. Experts have quite a different opinion on such a topic. The confrontation on this issue gave some extremely interesting insights. The problem is quite broad and complex. The general view is that the software development landscape changed dramatically along the years. So, developers could not be blamed for poor-quality work. Again, *short time to market* expectations and *shrinking IT budget* are generally considered the main reasons for low software quality. One panelist was able to give us a comprehensive view on the topic. According to this person, reasons should be found in:

- decreasing developers' daily rates and extensive use of junior professionals,
- low-quality and superficial functional analysis given to developers,
- compression of the development time following business needs (e.g., regulatory constraints, C-level decisions, strategy), and
- weak methodology and programming skills.

Our experts complained about low IT budget and use of junior professionals for highly complex applications. "Software quality decreases because developers' daily rates decreased in the last years," stated one informant. This opinion is largely shared among our panel. In fact, many other informants shared similar statements, like "the IT supply ecosystem changed dramatically in the last years. The pressure on unit prices led to the proliferation of a purely tactical development. So,

long term-view maintenance & evolution (with a higher unit price) has been penalized, increasing future system's cost." Or "shrinking time to market of software projects contribute to the low level of software quality. Moreover, the decrease of daily developer's rates forces vendors and consultants to hire people which accept very low rate. Typically, they are not the best developers on the market." "In my opinion, the contraction of IT budget costs and lower salaries payed to developers is the main cause of the decline of software quality, driving to an overall increase in the total cost of ownership of the information system," stated another. Since "shrinking IT budget leads to lower developers' daily rates, senior figures are less involved in crucial projects." Not surprisingly, one expert complained that "software vendors and consultants use for our projects only junior professionals with no supervision, so, the result respects this aspects. We really have to control their work step by step." According to another informant, "more than a result of a decline in individual professionalism, I would say that it is a drop in quality of delivery of increasingly smaller teams to cost cutting." Finally, one said that "I did not notice a decrease of individual professionalism. Rather I observe poor deliveries by shrinking teams due to lower IT budget."

Moreover, our experts found that low software quality is due to poor (or missing) software engineering methodologies. "In my experience, technical skills (i.e., knowledge of the programming language) is not lacking. Poor software engineering skills are the problem. Developers write directly the code, without planning and test it. So, testing is carried out by the customer after the release." Another informant said that "I observe a shrinking quality in such environments where stringent time to market requirements are set without an organizing proper development process, including testing, according to state of the art best practice (e.g., DevOps)." Moreover, "poor management of requirements in the demand phase leads to poor functional analysis and software quality," according to another panelist. Also, "the fast deploy (with low testing) of new functionalities impacts negatively on the entire software life cycle." Probably, "the complex information system architecture forces developers to sub-optimal solutions." Finally, "more that lacking professionalism, what is missing are proper integration methodologies for system integration required by our complex information system."

However, some are also complaining for lacking of specific skills. "On older architectures (i.e., mainframes) developers are not enough skilled," or "lacking legacy system management skills as also a poor vision of end-to-end multiplatform development are the cause of low quality."

Complexity, of both regulation and the application's environment, impacts negatively on software quality, according to other panelists. "We have to consider that information system's complexity raised exponentially in the last years. Thus, the level of professionalism has to be considered also on this regard." Concerning regulation, "the most relevant issue is the stringent financial compliance, which changes rapidly. So, the information system needs to be updated with a really tight time to market."

Also IT-business alignment is considered to be a reason for low software quality. "In my opinion, it is a side effect of complaining with business goals, losing the architectural and strategic view on the information system, more than people individual skills."

Factor 17: Contracting and skills. Our informants had different opinions and held a variety of positions concerning contracting and assessment. For some, outsourcing is the main cause of poor software quality. "According to my opinion this is the key problem. Outsourcing policies of the last years are the reason of poor software." Similarly, another person stated that "the depletion of internal personnel skill, in favor of outsourcing will hurt long term sustainability. I always suggest to outsource only non-critical functionalities."

Others are quite in favor of outsourcing projects, since they are easily assessable. "Contracts with good KPIs which ensure objectivity of delivery's evaluation is a viable and crucial solution. Such

kind of artifacts are more manageable and of better quality than internally developed software" Or "with a well defined sourcing model and metrics projects are fairly manageable." For another informant, "there are methodologies which enables a good control on suppliers." And "to ensure proper control on information system management there is the need for a good IT governance; through which each process related to the evolution and maintenance of the software is managed and controlled," stated one panelist. Generally speaking, "for Time & Material contracts, previous skills assessment is very important for project's success. Whereas, in traditional contracts, the focus is shifts on the quality of the deliverable and to its process."

Other experts generally supported outsourcing, with some criticality. "A good initial setup is crucial for a good project management. However, I saw a more sloppy management in the last years, so that most project's aspects are disregarded." Or "recurring to vendors and consultants leads to more complex project management but ensures the right skills and experience to get the job done. However, internal personnel is pivotal to control vendor's artifact quality."

Finally, some others expressed a more neutral opinion on the factor. "There is no such right or wrong solution. Outsourcing development, evolution and maintenance needs to be carefully evaluated case by case."

Factor 18: Lacking tools and methodologies. We gathered different opinions for this factor. As usual, the broadness of the domain shows various positions. Some fully support the factor, stating that "since we didn't find an adequate tool on the market, we developed it by ourselves. We are experiencing enormous benefits from it." Others complain (again) about tight schedules: "I agree in those contexts with 'forced' tight time to market deployments." And others already started the path of full methodology and tool implementation: "Our bank is already implementing methodologies (i.e., DevOps) to improve software quality through a testing phase of new applications."

However, the relation between tools and software engineering methodology is quite broadly stressed. When the process method is poor, the best tool will fail. "Now effective tools are available on the market. However, they require a strong methodological and organizational support. Software quality measurement by itself is useless if not supported by a software engineering culture and knowledge." Or, "Before lacking tools, I would argue that it's the poor software engineering methodology which 'kills' software quality." Moreover, "there are adequate tools and methodologies, the problem is that they are not applied within the software life cycle," said another panelist. Training is also an issue. One informant said that his organization doesn't have skilled employees to use highly complex tools. So, this is a key reason they are not implemented: "There may be adequate tools but using them is quite a complex task. You need to be very skilled to use them."

Another interesting aspect is the tradeoff between software quality assessment and costs: "There are methodologies but fully implementing them is quite expensive." Another stated that "I think there are enough tools and methodologies, the problem is that software quality assessment is rather expensive, so no one is willing to pay for it. Thus, they are simply not used." Especially, small banks complained about licensing costs: "For our small bank, existing tools are too expensive." And "there are enough tools but they are very expensive."

Finally, vendors complain about the market maturity and ability to value both tools and methodologies as viable solutions for software quality: "There are tools and methodologies but it's the market which lacks to recognize them."

Factor 19: Establishment of internal and external development processes. The establishment of internal and external development processes appears not to be an issue per se. "It's possible to codify them contractually," affirmed one panelist. Another affirmed that "I think that software engineering methodologies and tools are mature enough to guarantee a good software quality for both internal and external developments." Furthermore, one highlighted how the

process definition needs some iteration to be efficient: "Processes need to evolve according to needs and experience. However, you need proper custom tools to verify the compliance. In that way, the developers become accountable for their work." Rather, the problem is with monitoring and assessment: "You have just to impose a set of guidelines to outsources, but then you have to be monitor them by internal employees. I think here is the problem, since the IT does not have an internal audit structure." Or, "I think that besides asking for process guidelines, you have also to assess them." Some experts noted that customers may not be skilled enough to negotiate standards with a big software house: "I question myself whenever software quality guidelines need to be negotiated, especially with highly certified software houses."

According to some, on the other hand, it is very difficult to establish the process externally: "For outsourced custom software, this is rather difficult to assess, since any software house uses its internal standards." At least, you have to differentiate: "You have to differentiate internal and external development. In the first case you have to manage it properly accordingly to business goals and costs. In the second case you can not impose process decision but you may pretend a certain quality standard."

Internal skills are quite important for this task: "The most relevant aspect is that internal employees do not properly review outsourced developed software due to lack of time and skill." Organizations that put value on such skills reported quite positive experiences: "It depends on the business culture. If the organization retained internally IT functionalities, it is easily assessable, with very good outputs for system's quality; otherwise it is not." Also, the use of state-of-the-art tools is suggested by one expert in this regard: "Innovative software engineering tools should be more exploited within organizations. There are already automatic testing tools which are very helpful."

Typically, IT cost-benefit tradeoff is a constant issue. In fact, "it depends on how much you want to invest for good software quality (trade off between costs and benefits)," said one informant. Schedule is also here an issue: "often the time to market is detrimental for continuous testing of IT projects."

Finally, one informant shared his experience with suppliers: "The point is not the process definition, which is quite consolidated in best practices, but the assessment and accountability of suppliers. According to my experience there are three patterns:

- suppliers who do not follow the agreed methodology;
- suppliers who already use an adequate methodology/best practice, negotiate it with the customer according to his needs;
- suppliers who already use an adequate methodology/best practice, but do not find an appropriate agreement with the customer.

The best solution is always an agreed process and its monitoring with adequate tools."

Factor 20: Developer's professionalism versus skills. The aim of this factor is to understand the relationship between decreasing a developer's professionalism with the use of software engineering methodologies and best practice due to time and budget constraints. Some experts argued that "decreasing professionalism does not depend on budget, rather on education." Others saw the problem as domain specific: "this may be true in other domains, not in digital–web ones."

Other experts related this factor to process issues. For example, one informant said that "most of the time is devoted to the requirement understanding, thus time dedicated to the development shrinks." Another expert stated that "since software is intangible, it is very hard to manage properly its purchasing from vendors. On the other side, vendors still struggle to

find a good trade off between quality (intended as use of standards and best practices) and cost."

However, education is considered a central issue by most experts: "Junior developers are not experienced with concurrency and integration since they are trained with stand-alone platforms." Similar statements emerged quite frequently: "I would also add that there is a larger use of low skilled developers." Or, "I strongly believe that this is a central issue. However, I also noticed the use of developers with no technical background."

Probably, again, we may explain this phenomenon with shrinking budgets: "Sourcing policies had a great impact on that." And "often, due to strict time-to-market schedule, quality is sacrificed." Since there are not adequate resources to invest in education and to train developers with no formal education in IT, just because they are willing to accept lower wages is a sound interpretation of this factor: "I agree that time and budget are key issues to understand this tendency," affirmed another panelist.

Factor 21: Developer's professionalism versus rates. The aim of this factor is to understand the relationship between decreasing developers' professionalism and the decrease of professional rates. Apparently, experts expressed quite a variegated picture on the topic: "Professionalism in general is not decreasing. However, choosing cheaper suppliers leads to some compromises." On the contrary, another stated that "I agree in general but I disagree in particular, since we have very skillful experts in our organization." Another shared that "today customers are willing to pay more for more quality."

With regard to system integrators, "as long as I can remember, system integrators learn the job by doing it, while we never argued about senior's professional rates." Training appears to be a qualified issue also for others: "My company invests a lot in developer's training, regardless of their daily rate. We believe that this is the reason because we sell a lot of software products and system integration services." The issue appears to be more complex than the problem statement. In fact, an expert said that "you can not relate professional rates to training. Internal policies are too complex to understand the phenomenon. In a market economy, where a company wants to increase its customer, value creation through its employee is key to its survival. Otherwise, it would be substituted by other companies."

Finally, an informant suggested that it is not about individual professionalism, but rather team composition: "I do not think this is just a problem of training, rather of sizing and team management due to low budget. Project management, Business Analysis and Design (System and Module Design) are penalized in favor of purely development tasks."

Factor 22: Web technologies versus methodologies. With this factor, panelists expressed their opinions concerning the use of web technologies with development paradigms. There is the idea that the adoption of such technologies leads to a less rigorous approach to software engineering. In this regard, Agile is considered a scapegoat of such sloppiness: "Agile software development is largely misunderstood. Documentation is automatically extracted by metadata but the design is completely lacking." Or, "Behind the Agile buzzword people hide methodological shortcuts." Similarly, "this problem rises when methodologies are not understood or applied, like Agile ones." Others made more punctual observations, stating that "the use of Agile works very good for prototyping. However, it is a time bomb for the design of patterns or architectures." There is apparently a scarce culture for Agile. "In my experience, I only saw Waterfall methodologies. This because the customer rarely is able to express clearly requirements," said another expert. This could be a reason it is not seen as a methodology, but rather a shortcut to Waterfall. Culture is apparently an important (and lacking) issue.

Others noticed that "I do not think it depends on technologies, rather on low attention about quality." And, "Web technologies lead to high subjectivity in the development. Developers tend to use less software engineering methodologies within the development phase." Moreover, "some new technologies may change something (like Big Data) but not the way in general to develop software." With respect to the use of tools, one said that "this aspect relates also to wizard development tools, where personalizations are managed without a methodology." And to development environments: "I think that the development on open environment leads to more flexibility with respect to host ones. However, this leads to less rigor, so companies have to manage it properly."

Finally, a clear recommendation by one expert: "The customer has to choose the methodology. Otherwise, the supplier will provide software developed with different standards and methodologies."

Factor 23: Quality versus requirements. This factor analyzes the relationship between low software quality and clarity of user requirements as provided by the customer. The idea is that poorly defined requirements lead to misunderstandings with software developers, which will deliver lower-quality software. The functional quality will be low if the requirements' clarity is low.

"I would say that it is quite obvious." Or, "It is very hard to have high software quality if requirements are not clear." For some informants, it is a key issue of professionalism: "It is correct, but IT professionalism means to me to transform business objectives in viable requirements."

Other panelists expressed different viewpoints: "To me it seems much more like an alibi." Or they shifted the responsibility: "The IT should be accountable to understand the right requirements."

With regard to the user requirements, elicitation from our experts brought about interesting viewpoints: "I do not think that user requirements are generally not clear enough, rather it is the process to capture and formalize them which has been reduced to a simple wish list." And, with reference to specific professional key figures, one informant said that "this is particularly true in the financial sector, where key figures like Demand and Service Manager, which are able to translate business language into technical requirements, are lacking. Usually, managers talk directly to developers with frequent incomprehensions." Also the quality of interaction matters: "It depends on the interaction between customer and developer. A good developer (with more experience) will interpret better businesses' needs, however, he won't be cheap." Also, "developers should be more proactive and ask if they do not correctly understand one requirement." More generally, "this is a very important cause, since the clarity of the demand cycle is very important for a robust development phase."

Finally, an important element that emerged is the difference between functional and nonfunctional dimensions of software quality. "It is partially true. If the requirement is badly or partially defined, it does not mean that software is per se of low quality. It will just work as it was (partially) defined." Moreover, "I partially agree. Even though functional requirements are of low quality (due to misunderstanding), it does not mean that also non-functional ones are also of low quality." So, if generally experts affirmed that there is a relationship between requirements clarity and functional requirements quality, this may not influence the nonfunctional dimension.

Factor 24: Requirements versus methodologies. This factor analyzes the relationship between badly defined user requirements provided by the customer and the available software engineering methodologies to elicit them. The idea is to see if poor methodologies to elicit business goals and needs leads to the definition of unclear software requirements. One panelist expressed this concern: "Requirements are usually elicited orally. Organizations are rarely structured to develop properly business needs into software requirements according to software engineering methodologies and best practices."

The relation between the business and IT appears to be a critical issue to explain this point: "Poor requirements depend on the fact that often the customer does not know what he wants!"

Thus, startups with less defined organizational routines [92] seem to be more flexible and have fewer problems on requirements elicitation: "New organizations reduced the distance between business and IT, so, requirements are more easily elicited."

Although elicitation methodologies are considered important, the collaboration with business is perceived as pivotal: "A robust process of Demand Management is alone not the answer to bad requirements elicitation. Rather, the business has to express clearly its goals and expectations." Moreover, "processes and methodologies are important to manage the demand properly and to turn businesses' needs into software requirements. However, these needs should be expressed with enough clarity."

Customers' skills are also considered crucial for high functional requirements quality: "Poor customer culture and skills lead to badly defined user requirements. There is the need to educate and support the business to define them better." And, "According to my opinion the problem lies in a low professionalism at the business side, which ask for more specialized consultancy." The cause may also be poor prototyping: "I agree. I think it depends also on the difficulty to provide a viable prototype to show how the future application will work."

Interestingly, one informant expressed his concerns to consider the topic as just procedural: "In my experience bad requirements are due to several factors related to doing teamwork. It is much better to focus on collaboration and work through goal oriented target rather than use highly structured protocols and methodologies which enhances barriers among people." By the way, contemporary paradigms, especially Agile and DevOps, are built according to this belief.

Generally speaking, we noticed how from our informants emerged Humphrey's law, according to which requirements are not known by the customer until he or she will use them.

Factor 25: Requirements versus technical jargon. Communication among different professional figures is usually not trivial. This factor explores the impact of the (mis-)use of technical jargon of different departments for requirement elicitation.

Although this may be true for many, since "IT experts do not know business processes, they know about technology and I think that there is the need of a "translation" phase," it appears to be very much related to the single business culture". "The use of jargon is related to the business culture." So, organizations that are aware of it tend to solve the problem upfront. "When business and IT teams are co-located requirements are elicited effectively. Traditional silos approaches do not work." Or, "It is an old issue, nowadays business and IT speak the 'same language,' especially in our organization." Moreover, "I would agree if collaboration among IT and business is poor. With regard to my personal experience interaction and collaboration works very well, so we developed a common 'dialect' within the company."

One panelist suggested that excessive specialization may also be an important issue: "I think that especially in the financial industry, people with a cross-cutting view on both business and IT processes are lacking. This may depend on excessive vertical specialization, so that there are no people around with the broad picture. To have such picture you have to put around a table many different figures to define the requirements, which is quite difficult. This is the reason why it is so difficult to elicit requirements."

Other explained that prototyping and preliminary analysis are of crucial importance for a good functional requirement definition: "I think that poor prototyping is a major issue of this topic. They enhance incredibly requirements specifications." And, "If there is a good common preliminary analysis, elicitation is not a big issue."

Finally, new professional figures are merging: "It is true, for this reason new figures like the CDO (Chief Digital Officer) are emerging." So, linking functions are quite important: "A good demand manager is the 'trait d'union' between both departments. This figure has to be strengthened using best practice tool, like business glossaries, metadata and impact analysis."

Factor 26: Data analysis versus functional analysis. This factor explores whenever poor data analysis influences functional analysis and consequently the system integrity: "The 'functional centric' view is quite misleading, since it relegates the importance of data. Data give the 'static view' of existing functionalities, which is very important for the functional analysis. One software product may have all possible functionalities required by the user but lacking of fundamental data its deployment and system integration becomes impossible." Moreover, "data analysis skills are generally lacking and poorly used in functional analysis." Apparently, this issue is present market-wide: "In my opinion, in the market there is a lacking perception about the importance of data analysis as preliminary phase of functional analysis."

However, other experts disagree: "Saying that poor analysis is due to poor data understanding is a quite generic (it is obvious that data processing is the main IT goal) and old issue." Other issues are also relevant to understand the factor: "Also knowing what different data means is important." And, "It is not only an issue of poor technical skills." Furthermore, "personally, I saw poorer knowledge of banks operation processes."

Apparently, there was a shift after 2010 that gives interesting insights: "It is true for applications developed before 2010. Data governance is now more relevant and data analysis is performed before the functional one. So I see a clear discontinuity with the past."

Factor 27: Functional analysis versus data modeling. Our panelists generally stated that difficulties in functional analysis lie in bad data modeling and identification of data sources: "In our bank, we need a data mapping, otherwise any further analysis would be useless." The reason may be that for many, "the knowledge of data is the real starting point."

The deep knowledge of data sources is considered a main driver for software quality, since it enhances functional requirements.

Factor 28: Documentation standards and tools. In this last factor, the issue of lacking software documentation standards and tools emerged. Experts expressed that poor documentation hinders software maintainability and increases evolution costs: "I think that the real problem is the time/effort trade off. If developers are pushed just to write some code, without documenting it properly, it becomes a problem in the medium run. Maintenance and evolution are critical since, due to developers' turnover, it is hard to have control over the software." And, "Following development and architectural standards, enables also those who did not write the software to manage it properly afterwards."

However, other panelists affirmed that both standards and tools already exist; they just need to be implemented: "There are already dedicated documentation tools, also open source ones, like 'Alfresco,' you just have to use them." And, "There are tools but developers have to use them. From alone they do not work." Furthermore, "there are very effective standards, but often they are not used." Or, "There are valuable tools and the market need to recognize them. Software development is very similar to an assembly line of mechanical products. Tools and methodologies increase quality and reduce cost." One company also developed for its own purposes a documentation tool: "Recently, we developed an 'ad hoc' documentation tool which is very effective." However, an expert disagreed on this point: "In my company, we already defined documentation standards. However, we still did not find an adequate tool to increase documentation efficiency."

Also here training appears to be a crucial point: "Often standards are defined but no training is planned for developers." So also, "the real problem is that minimal development standards are not followed."

Finally, the role of the IT department is stressed by one informant. He used the word "dignity" in the sense that such technical decisions may not lie in another departments' domain or even be the results of top-down decisions: "I would argue that it is the IT department which should have the 'dignity' to impose standards, not waiting decisions from the business side."

5 DISCUSSION

Our research questions aimed to identify (RQ_1) and validate (RQ_2) the main quality-related concerns in the IT financial sector. What emerged is a strong relationship among software quality, software architecture, and software process. Cost cutting had a direct impact on architectural aspects (both on ex ante planning and ex post analysis/reverse engineering), documentation, and data modeling. Traditional quality models appear to be insufficient to explain these outcomes. Moreover, none of our informants affirmed using any comprehensive quality model for their systems, reporting to struggle with their IT infrastructure.

Indeed, legacy systems are the most important ones for the day-to-day business operations [72]. The old-fashioned COBOL language is predominant for such systems since it is very efficient to process millions of batch transactions per day, which is the core activity of any financial organization. Such systems are truly business critical. They are reliable and have been running for decades, have been well tested in time, and run virtually with no errors.

Unfortunately, we report the tendency to stratify core systems, in order to implement rapidly new functionalities due to budget constraints. Architectural integrity has been challenged by two opposed drivers: new requirements and scarce resources. Testing and documentation are often skipped. Reverse engineering becomes a highly complex and expensive activity.

Furthermore, the coexistence of codes written in different times and different programming paradigms is perceived as a barrier to software evolution and maintenance. The use of languages like COBOL impacts negatively code understandability, as well as the possibility to be translated in a more modern language. Moreover, none of the organizations we interviewed admitted to having a complete overview and map of the source code running on their information systems. Although a considerable portion of information systems of financial organizations is provided by vendors with COTS products under licenses, code ownership is fragmented and not centralized.

This raises another issue: the total cost of ownership [40]. The levels of application complexity, system stratification, and fragmented ownership increase tremendously the total cost of ownership of information systems of financial organizations. CIOs do often prefer incremental working solutions rather than a comprehensive approach to such issue. Informants suggested that this behavior is caused by a fast turnover of C-levels and a short-term view. They do not have real incentives to tackle this problem; rather, they focus on short-term, quarterly goals. This pattern has also been observed in the literature [52].

Generally speaking, information systems of financial institutions appear to be characterized by a highly complex and stratified architecture, with a sinking quality. Moreover, with respect to the past, experts' opinions let emerge gloomy scenarios, due to an increase of functionalities driven both by market and regulations requirements and inadequate budget provisions.

5.1 Theoretical Coding

Although we did not commit to Grounded Theory [54], we used theoretical coding to map the items to the relevant literature [120]. According to our *pragmatic* Mixed-Methods approach, we were able to build a valuable model from empirical evidence. In Grounded Theory theoretical codes are formalized after open and axial coding, since they are the highest-level constructs [55]. Indeed, theoretical codes are flexible and do not explain just one theoretical construct; rather, they link several constructs. Glasser explained how "they are not mutually exclusive, they overlap considerably... [and] one family can spawn another" [54, p. 73]. The use of theoretical coding provides a sharp analytical edge in the results. It helps to provide clarity, coherence, and theoretical relevance of data.

Since 28 items are not parsimonious for a model development, we clustered the factors into high-level categories, following [51]. So, we did not just provide the description of the elicited items; we mapped them into some relevant ISO standards. Using ISO standards instead of literature has several advantages in our case.

Information systems of financial organizations are supposed to comply with such standards. We dealt with concepts that were very well known to our informants, enabling a homogeneous and coherent model. Another advantage of dealing with standards is that they are de facto second-order theories, built on grounded pre-existing ones and shared among scholars' and practitioners' communities. Three categories emerged from our study: software quality, software process, and software architecture. The 28 factors we identified were mapped in the corresponding standards. With regard to software quality, we mapped our factors within the Product Quality Model of the standard ISO/IEC 25010:2011, as shown in Table 5. To map the factors regarding the software process, the ISO/IEC 12207:2008 standard for the software life cycle processes was used, shown in Table 6. Finally, to see the most relevant elements related to software architecture, we performed our mapping over the ISO/IEC/IEEE 42010:2011 standard in Table 7.

To map our factors to subcharacteristics, we used a Delphi approach derived from software cost estimation [20]. Each author elaborated autonomously the categorization. After that phase, we met personally and took for granted the unanimous cases. The other cases, where one of the authors was in disagreement, were discussed until consensus was reached. Since the factors that came from the panel were rather cross-cutting along different quality characteristics, we assigned some factors on more than one characteristic. We chose to be not restrictive in our categorization since the aim was to see if the standards are sufficiently comprehensive for the identified factors. The final factor mapping, as described in Tables 5, 6, and 7, is the unanimous outcome of the iterations of all authors. As a result, multidimensional and overlapping concepts provided by experts were referenced to relevant theory, developing, de facto, a model that is intended to give a comprehensive picture of ISQ by abstracting and connecting from more detailed individual ISO models contained within it (i.e., a meta-model [83]).

5.2 A Model for Information Systems Quality

A *meta-dimension* of quality emerged from empirical evidence. We specifically propose the prefix *meta-*, intended to define another subject that analyzes the original one but at a more abstract and higher level [82]. Indeed, factors may be explained through a cross-cutting analysis of the three standards, since they are not related just to one category (i.e., quality, architectural, or process). So, having a comprehensive idea about ISQ means managing a meta-view on these three categories. Basically, each category influences the others and has an impact on them.

For instance, business goals have a direct impact on the information system. For example, IT cost cutting by a bank executive board to invest in other departments or to be more profitable the next quarter is a clear business goal. Short-term-view decisions or even not-taken decisions are business goals that influence all three categories of quality, architecture, and process. As a result of our research, non-IT executives are willing to pursue short-term goals, according to our experts, increasing the total cost of ownership. We did not collect non-IT experts' opinions in our study, so we cannot detail this phenomenon. However, the literature tried to explain this common pattern [52]. Like the panelists remarked, this led to low maintainability and evolution capabilities as well as to some architectural anti-patterns. Several quality aspects are influenced by such business goals, like quality, maintenance and evolution, processes, and architecture. In other words, the proposed ISQ model is able to provide a comprehensive view about the key quality characteristics of an information system. Furthermore, it helps to trace the impact of decisions on one category into the others.

Table 5. Factor Mapping According to ISO/IEC 25010 – System and Software Quality Models

Characteristics	Subcharacteristics	Factors
Functional suitability	Functional completeness	6, 7, 12, 23, 24
	Functional correctness	6, 7, 12, 15, 24, 25, 26
	Functional appropriateness	3, 6, 7, 27
Performance efficiency	Time behavior	5
	Resource utilization	5
	Capacity	
Compatibility	Coexistence	5
	Interoperability	5
Usability	Appropriateness recognizability	7
	Learnability	8
	Operability	3, 9
	User error protection	
	User interface aesthetics	
	Accessibility	
Reliability	Maturity	8, 15, 18, 19, 20, 21, 22
	Availability	
	Fault tolerance	7
	Recoverability	
Security	Confidentiality	
	Integrity	2, 7, 15
	Nonrepudiation	15
	Accountability	2, 7, 15
	Authenticity	
Maintainability	Modularity	1, 2, 10, 18
	Reusability	4, 8, 9, 10, 13, 14, 18, 28
	Analyzability	4, 5, 6, 8, 9, 10, 11, 12, 18
	Modifiability	2, 4, 5, 8, 10, 11, 12, 18
	Testability	4, 5, 8, 10, 18
Portability	Adaptability	4, 8, 9, 10, 13, 14, 28
	Installability	10, 13, 14
	Replaceability	4, 6, 8, 9, 10, 11, 12, 13, 14

Therefore, we do not induce (intended as the outcome of a qualitative research) another quality dimension of IT systems. We induce, instead, a new *meta-model* for information systems quality composed by software quality, software architecture, and software process.

Our contribution is of a qualitative nature: we inductively report emerging factors of IT quality into the three categories of the ISQ model that have not been observed before in a real-world context. The proposed SQuAP (Software Quality, Architecture, and Process) meta-model is represented in Figure 2 and it links to the ISO/IEC 25010 standard regarding software quality,

Table 6. Factor Mapping According to ISO/IEC 12207 – Software Life-Cycle Processes

Characteristics	Subcharacteristics	Factors
Primary	Maintenance	1, 24, 5, 9, 10, 12, 13, 14
	Operation	
	Development	2, 5, 18, 22, 23, 24, 25, 26, 27
	Supply	13, 14, 16 17, 19, 20, 21
	Acquisition	3, 13, 14, 17, 19
Supporting	Documentation	4, 7, 8, 10, 11, 12, 28
	Configuration Management	9
	Quality Assurance	1, 2, 3, 6, 12, 13, 14, 15, 18, 19
	Verification	3, 9, 15
	Validation	15
	Joint Review	
	Audit	15
	Problem Resolution	9
Organizational	Management	5, 8, 12, 20, 21, 25
	Infrastructure	17
	Improvement	12, 17
	Training	16, 17, 20

Table 7. Factor Mapping According to ISO/IEC 42010 – Architecture Description

Characteristics	Subcharacteristics	Factors
Objective	Model Kind	
	Architecture Model	1, 2, 18, 26
	Architecture View	1, 6, 15, 18, 23, 26, 27
	System of Interest	1, 11, 12
	Architecture Rational	1, 8
	Correspondence	4, 14, 22
	Correspondence Rules	4, 9, 13, 22, 24, 28
	Architecture Framework	9, 13, 26, 27
Subjective	Concern	3, 5, 12, 14, 15, 16, 17, 18, 19, 20, 21, 23, 28
	Stakeholder	1, 3, 6, 15, 12, 16, 24, 25
	Architecture Viewpoint	10, 23
	Environment	11, 21, 22
	Architecture Description	1, 7, 8, 10, 18, 28

ISO/IEC 42010 about architecture description, and ISO/IEC 12207 regarding software life-cycle processes.

6 THEORETICAL AND PRACTICAL CONTRIBUTION

From a research perspective, this work makes a theoretical contribution by providing a new model of ISQ based on the three categories of software quality, software architecture, and software

Fig. 2. SQuAP (Software Quality-Architecture-Process) meta-model.

process. Also, the explanation of the links between these categories, into ISO standards' subcategories and experts' IT quality factors, has been provided in Tables 5, 6, and 7. Additionally, it relates an organization's behavior and its impact on ISQ, through the relationships among the categories.

The literature highlights that software quality aspects and architecture are tightly coupled [107]. Conscious or unconscious faulty decisions on an architectural level lead to low system-wide quality attributes, in particular poor maintainability and evolution capabilities [78]. We were able to validate this literature insight also in our research. We observed a tendency to stratify applications rather than evolving the existing ones. This is performed for several reasons due to poor documentation, testing, and reverse-engineering capabilities.

As for practice, our model provides a management tool to map business decisions into IT categories, to trace their impact on the system at large. SQuAP thus becomes a tool that can structure strategic decisions. This may lead to a general rethinking of IT-business alignment. Typically, the failure of IT to deliver value, poor understanding of IT by business executives, lack of a clear vision with respect to IT, different views of business executives and technology specialists, lack of a shared vision in relation to IT, poor technical skills of CIOs, and criticism of legacy IT are the reasons for such a structural misalignment [74]. Mutual understanding of CIOs with other C levels is considered in the literature as a main driver to IT-business alignment [16]. Not surprisingly, the relationship between IT-business alignment and competitive advantages is a well-known topic in research, since it leverages business capabilities through automation and business intelligence [105].

SQuAP addresses how to deal with both the managerial and the technical perspectives, and their IT-business alignment, and in particular the total cost of ownership of information systems of financial organizations. It is intended to bridge the understanding of IT to non-IT C levels. Indeed, a wider use of standards (i.e., IEEE/ISO/IEC) is an accountable way to move in this direction. In this regard, a model like SQuAP, based on industrial standards, is useful to visualize and assess the relevant assets of information systems.

7 CONCLUSIONS AND LIMITATIONS

This study used a qualitative Delphi-like methodology to identify IT quality concerns regarding information systems of financial institutions. The concerns were validated through a survey of a large target panel composed of carefully selected domain experts. The picture that came out of our research is a consequence of the short-term vision of C levels. This short-sightedness impacts the strategies of maintenance and evolution of information systems of financial organizations. All concerns made explicit or implicit reference to software quality, process, or architecture issues. Thus, we mapped them within the ISO 25010, 42010, and 12207 standards. A new *meta-model* for ISQ emerged from our inductive research approach regarding the IT financial sector. Such a model explains the tight relationships between software quality, process, and architecture. We have followed an inductive, theory-building approach. The focus is not on the single standard, but rather on the interactions among them, proposing a cross-cutting view. This model provides valuable insights to map and assess ISQ.

Limitations were our concerns during this study, since we used a new inquiry methodology through Mixed Methods. We use both qualitative and quantitative validity paradigms. To analyze the qualitative dimension, we adopt credibility, transferability, dependability, and confirmability, following [58]. For the quantitative dimension we use traditional statistical conclusion, internal, construct, and external validity, as suggested by [133]. Even though validity dimensions of qualitative and quantitative researches are different, we aggregated them but discussed them separately for epistemological reasons. The research was homogeneous and both qualitative and quantitative dimensions gained trustworthiness from one another.

Credibility and Internal. Factors identified are all credible. We identified them through a Delphi-like process, which lasted about 1 year. Panelists were sector experts, with daily exposure to the researched concerns. The whole panel had the chance to discuss, refine, and group the elicited items, through different phases. None of the experts argued that any of the concerns should be excluded. Extra items, which did not reach full consensus among the panel, were isolated and treated properly. Moreover, they are presented separately from the first 15 items. However, target panelists were asked to validate all 28 items through survey randomized items. Random stratified assignment of the research subjects was designed to maximize internal validity. Representativeness of the sample is high for two reasons. Experts were chosen among a highly qualified pool of an

established IT consulting firm of the financial sector. Strata were first assumed by the author's experience and then integrated into the panel. Moreover, the guarantee of anonymity given to both panelists and target panelists allowed an unbiased and frank discussion. This increased the credibility of the study, since experts were able to answer and express openly their knowledge.

Transferability and External. The Mixed-Methods approach aims to explore and build new theory (induction) and also to validate it (deduction). The degree to which the results of qualitative research are transferable to other settings may be interesting. Admittedly, we focused on one sector, to enhance internal validity; however, our findings are still fairly generalizable. The financial industry is one of the most standardized sectors worldwide due to market and regulation similarities. All those points were discussed in Section 2.6. Therefore, it is reasonable to believe that most concerns are largely shared among the entire industry, for the reasons before described. Furthermore, our findings may also be generalizable for other industries with similar characteristics. This study focuses on the most important factors that threaten the IT financial sector. Threats to external validity are not considered very harmful since we consider a standardized industry.

Dependability and Construct. The authors of this research are well aware of the context. Experts were carefully chosen through a stratified randomized sampling, using the highly refined pool of an established IT consulting firm, specialized in the financial sector. We described the three dimensions (experience, company, and role) and subdimensions, as well as the stratification of the samples. At the end of the qualitative research, a target panel of 124 experts carefully selected through stratified randomized sampling were asked to evaluate the panel's outcome. The outcome was a substantial consensus on all concerns with a degree of agreement above 70% for all of the first 15 concerns, with high average scores. However, even though the first 15 concerns were highly shared among target panelists, this does not mean that these are the most relevant for them. Thus, following our research approach, we also added 13 extra items not fully shared by the panel. Interestingly, such extra items have a lower agreement degree among the target panel. However, this approach provided a full *pragmatic* picture of our research journey.

We remark that the concerns were elaborated by a high-level panel of experts in a three-phase round. The aim of the quantitative part was to verify the construct. Hence, we fully comply with our Mixed-Methods approach.

Confirmability and Statistical Conclusion. All items were discussed within the first panel, through different phases, which led to a continuous check. After the whole qualitative research process, concerns were evaluated by a large target panel composed of 124 experts. We computed our results with MS Excel using representative sample sizes to increase statistical power. Measures and treatment implementation are considered reliable.

It was concluded that the threats would not to be regarded as critical. As we know, there is a constant tradeoff between internal and external validity [133], and our sampling strategy took this into account. With Mixed Methods, we aim to get useful insights for theory building (external validity and transferability), while we also try to validate it within the same study (internal validity and credibility).

Our research will continue in several directions. Mixed-Methods research is needed to validate and extend this model to other industries. Several software quality metrics, as well as managerial and organizational evaluation techniques, can be exploited. We are exploring the support of semantic technologies, especially ontologies, to provide a formal knowledge representation layer on top of which different methods and technologies can be validated and developed. Finally, empirical validation through case study research may broaden the use of the model as a central reference for ISQ.

REFERENCES

[1] European Banking Authority. 2017. *Regulatory Technical Standards on Strong Customer Authentication and Secure Communication under PSD2, Author=European Banking Authority*. Final Draft. EBA/RTS/2017/02.

[2] European Parliament and of the Council. 2007. Directive 2007/64/EC. Retrieved from http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02007L0064-20091207.

[3] European Parliament and of the Council. 2015. Directive 2015/2366/EU. Retrieved from http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015L2366.

[4] G. Abowd, G. Bass, P. Clements, R. Kazman, and L. Northrop. 1997. *Recommended Best Industrial Practice for Software Architecture Evaluation*. Technical Report. Software Engineering Institute.

[5] A. T. M. Aerts, J. B. M. Goossenaerts, D. K. Hammer, and J. C. Wortmann. 2004. Architectures in context: On the evolution of business, application software, and ICT platform architectures. *Information & Management* 41, 6 (2004), 781–794.

[6] S. Aier, B. Gleichauf, and R. Winter. 2011. Understanding enterprise architecture management design – An empirical analysis. In *Proc. 10th Int. Conf. Wirtschaftsinformatik*.

[7] A. April and F. Coallier. 1995. Q. Bell Canada, "trillium: A model for the assessment of telecom software system development and maintenance capability." In *Proc. Software Engineering Standards Sym.*

[8] M. A. Babar, A. W. Brown, and I. Mistrík. 2013. *Agile Software Architecture: Aligning Agile Processes and Software Architectures*. Newnes.

[9] M. A. Babar, L. Zhu, and R. Jeffery. 2004. A framework for classifying and comparing software architecture evaluation methods. In *Proc. Australian Software Engineering Conference*. IEEE, 309–318.

[10] R. D. Banker, I. Bardhan, and O. Asdemir. 2006. Understanding the impact of collaboration software on product design and development. *Information Systems Research* 17, 4 (2006), 352–373.

[11] R. Baskerville and M. D. Myers. 2004. Special issue on action research in information systems: Making IS research relevant to practice: Foreword. *MIS Quarterly* 8, 3 (2004), 329–335.

[12] L. Bass, P. Clemens, and R. Kazman. 2012. *Software Architecture in Practice* (3rd ed.). Addison-Wesley.

[13] L. Bass, I. Weber, and L. Zhu. 2015. *DevOps: A Software Architect's Perspective*. Addison-Wesley.

[14] S. Bellomo, I. Gorton, and R. Kazman. 2015. Toward Agile architecture: Insights from 15 years of ATAM data. *IEEE Software* 32, 5 (2015), 38–45.

[15] P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet. 2004. Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software* 69, 1 (2004), 129–147.

[16] A. Benlian and I. Haffke. 2016. Does mutuality matter? Examining the bilateral nature and effects of CEO–CIO mutual understanding. *Journal of Strategic Information Systems* 25, 2 (2016), 104–126.

[17] S. J. Berman. 2012. Digital transformation: Opportunities to create new business models. *Strategy & Leadership* 40, 2 (2012), 16–24.

[18] S. C. Blumenthal. 1969. *Management Information Systems; A Framework for Planning and Development*. Technical Report.

[19] B. Boehm. 1988. A spiral model of software development and enhancement. *IEEE Computer* 21, 5 (1988), 61–72.

[20] B. Boehm, C. Abts, and S. Chulani. 2000. Software development cost estimation approaches–A survey. *Annals of Software Engineering* 10, 1–4 (2000), 177–205.

[21] B. W. Boehm, J. R. Brown, and M. Lipow. 1976. Quantitative evaluation of software quality. In *Proc. 2nd Int. Conf. on Software Engineering (ICSE'76)*. ACM/IEEE, 592–605.

[22] W. F. Boh and D. Yellin. 2006. Using enterprise architecture standards in managing information technology. *Journal of Management Information Systems* 23, 3 (2006), 163–207.

[23] V. Boucharas, M. van Steenbergen, S. Jansen, and S. Brinkkemper. 2010. The contribution of enterprise architecture to the achievement of organizational goals: Establishing the enterprise architecture benefits framework. Department of Information and Computing Sciences, Utrecht University, Utrecht.

[24] A. C. Boynton, B. Victor, and Pine J.1993. New competitive strategies: Challenges to organizations and information technology. *IBM Systems Journal* 32, 1 (1993), 40–64.

[25] J. Brancheau, B. Janz, and J. Wetherbe. 1996. Key issues in information systems management: 1994-95 SIM Delphi results. *MIS Quarterly* 20, 2 (1996), 225–242.

[26] W. Brown, R. Malveau, H. McCormick, and T. Mowbray. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons.

[27] G. Campbell and P. Papapetrou. 2013. *SonarQube in Action*. Manning Publications Co.

[28] P. Checkland. 1981. *Systems Thinking, Systems Practice*. John Wiley.

[29] C. Cherryholmes. 1992. Notes on pragmatism and scientific realism. *Educational Researcher* 21, 6 (1992), 13–17.

[30] T. Chow and D. Cao. 2008. A survey study of critical success factors in Agile software projects. *Journal of Systems and Software* 81, 6 (2008), 961–971.
[31] M. B. Chrissis, M. Konrad, and S. Shrum. 2003. *CMMI Guidelines for Process Integration and Product Improvement.* Addison-Wesley Longman Publishing.
[32] C. Churchman. 1971. *The Design of Inquiring Systems Basic Concepts of Systems and Organization.* Basic Books.
[33] P. Ciancarini, D. Russo, A. Sillitti, and G. Succi. 2016. A guided tour of the legal implications of software cloning. In *Proc. 38th Int. Conf. on Software Engineering (ICSE'16).* ACM/IEEE, 563–572.
[34] P. Ciancarini, D. Russo, A. Sillitti, and G. Succi. 2016. Reverse engineering: A european IPR perspective. In *Proc. 31st Annual ACM Symposium on Applied Computing (SAC'16).* 1498–1503.
[35] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, and M. Mirakhorli. 2013. The twin peaks of requirements and architecture. *IEEE Software* 30, 2 (2013), 24–29.
[36] P. Clements, R. Kazman, and M. Klein. 2002. *Evaluating Software Architectures.* Addison-Wesley.
[37] J. Creswell. 2013. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* Sage.
[38] J. Creswell, V. Clark, and L. Plano. 2007. *Designing and Conducting Mixed Methods Research.* Wiley.
[39] M. A. Cusumano. 2004. *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad.* Simon and Schuster.
[40] J. S. David, D. Schuff, and R. St Louis. 2002. Managing your total IT cost of ownership. *Communications of the ACM* 45, 1 (2002), 101–106.
[41] C. Deephouse, T. Mukhopadhyay, D. R. Goldenson, and M. I. Kellner. 1995. Software processes and project performance. *Journal of Management Information Systems* 12, 3 (1995), 187–205.
[42] W. DeLone and E. McLean. 1992. Information systems success: The quest for the dependent variable. *Information Systems Research* 3, 1 (1992), 60–95.
[43] W. DeLone and E. McLean. 2003. The DeLone and McLean model of information systems success: A ten-year update. *Journal of Management Information Systems* 19, 4 (2003), 9–30.
[44] J. Dewey. 1938. *Logic: The Theory of Inquiry.* Holt.
[45] G. W Dickson. 1968. Management information-decision systems: A new era ahead? *Business Horizons* 11, 6 (1968), 17–26.
[46] L. Dobrica and E. Niemela. 2002. A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering* 28, 7 (2002), 638–653.
[47] E. Doke and N. Swanson. 1995. Decision variables for selecting prototyping in information systems development: A Delphi study of MIS managers. *Information & Management* 29, 4 (1995), 173–182.
[48] A. Dorling. 1993. SPICE: Software process improvement and capability determination. *Software Quality Journal* 2, 4 (1993), 209–224.
[49] K. E. Emam, W. Melo, and J.-N. Drouin. 1997. *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination.* IEEE.
[50] J. Etezadi-Amoli and A. Farhoomand. 1996. A structural model of end user computing satisfaction and user performance. *Information & Management* 30, 2 (1996), 65–73.
[51] D. Feeny, M. Lacity, and L. Willcocks. 2005. Taking the measure of outsourcing providers. *MIT Sloan Management Review* 46, 3 (2005), 41.
[52] D. F. Feeny, B. R. Edwards, and K. M. Simpson. 1992. Understanding the CEO/CIO relationship. *MIS Quarterly* 16, 4 (1992), 435–448.
[53] D. Garlan and D. E Perry. 1995. Introduction to the special issue on software architecture. *IEEE Transactions on Software Engineering* 21, 4 (1995), 269–274.
[54] B. G. Glaser. 1978. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory.* Sociology Press.
[55] B. G. Glaser. 1992. *Basics of Grounded Theory Analysis: Emergence vs Forcing.* Sociology Press.
[56] N. Gorla and S. C. Lin. 2010. Determinants of software quality: A survey of information systems project managers. *Information and Software Technology* 52, 6 (2010), 602–610.
[57] N. Gorla, T. Somers, and B. Wong. 2010. Organizational impact of system quality, information quality, and service quality. *Journal of Strategic Information Systems* 19, 3 (2010), 207–228.
[58] E. Guba. 1981. Criteria for assessing the trustworthiness of naturalistic inquiries. *Educational Communication and Technology Journal* 29, 2 (1981), 75–91.
[59] U. G. Gupta and R. E. Clarke. 1996. Theory and applications of the Delphi technique: A bibliography (1975–1994). *Technological Forecasting and Social Change* 53, 2 (1996), 185–211.
[60] S. Hayne and C. Pollard. 2000. A comparative analysis of critical issues facing Canadian information systems personnel: A national and global perspective. *Information & Management* 38, 2 (2000), 73–86.
[61] L. A. Von Hellens. 1997. Information systems quality versus software quality a discussion from a managerial, an organisational and an engineering viewpoint. *Information and Software Technology* 39, 12 (1997), 801–808.

[62] J. Highsmith and M. Fowler. 2001. The Agile manifesto. *Software Development Magazine* 9, 8 (2001), 29–30.
[63] R. Hirschheim and H. K Klein. 2011. Tracing the history of the information systems field. *The Oxford Handbook of Management Information Systems: Critical Perspectives and New Directions*, 16–61.
[64] W. Humphrey. 1988. Characterizing the software process: A maturity framework. *IEEE Software* 5, 2 (1988), 73–79.
[65] R. Judd. 1972. Use of Delphi methods in higher education. *Technological Forecasting and Social Change* 4, 2 (1972), 173–186.
[66] M. Jun and S. Cai. 2001. The key determinants of internet banking service quality: A content analysis. *International Journal of Bank Marketing* 19, 7 (2001), 276–291.
[67] H. W. Jung. 2007. Validating the external quality subcharacteristics of software products according to ISO/IEC 9126. *Computer Standards & Interfaces* 29, 6 (2007), 653–661.
[68] H. W. Jung, S. G. Kim, and C. S. Chung. 2004. Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software* 21, 5 (2004), 88–92.
[69] S. H. Kaisler, F. Armour, and M. Valivullah. 2005. Enterprise architecting: Critical problems. In *Proc. 38th Annual Hawaii Int. Conf. on System Sciences (HICSS'05)*. IEEE, 224b.
[70] R. Kazman, L. Bass, G. Abowd, and M. Webb. 1994. SAAM: A method for analyzing the properties of software architectures. In *Proc. Int. Conf. on Software Engineering (ICSE'94)*. ACM/IEEE, 81–90.
[71] P. G. W. Keen. 1987. MIS research: Current status, trends and needs. *Information Systems Education: Recommendations and Implementation*, R. A. Buckingham, R. A. Hirschheim, F. F. Land, and C. J. Tully (Eds.). Cambridge University Press, Cambridge, England, 1–13.
[72] Ravi Khadka, Belfrit V. Batlajery, Amir M. Saeidi, Slinger Jansen, and Jurriaan Hage. 2014. How do professionals perceive legacy systems and software modernization?. In *Proc. of the 36th Int. Conf. on Software Engineering*. ACM/IEEE, 36–47.
[73] M. Krafft, K. Stol, and B. Fitzgerald. 2016. How do free/open source developers pick their tools?: A Delphi study of the Debian project. In *Proc. 38th Int. Conf. on Software Engineering (ICSE'16)*. ACM/IEEE, 232–241.
[74] V. Krotov. 2015. Bridging the CIO-CEO gap: It takes two to tango. *Business Horizons* 58, 3 (2015), 275–283.
[75] M. Lange, J. Mendling, and J. Recker. 2016. An empirical analysis of the factors and measures of enterprise architecture management success. *European Journal of Information Systems* 25, 5 (2016), 411–431.
[76] B. Langefors. 1973. *Theoretical Analysis of Information Systems*. Technical Report.
[77] P. Lavrakas. 2008. *Encyclopedia of Survey Research Methods*. Sage.
[78] Z. Li, P. Liang, and P. Avgeriou. 2015. Architectural technical debt identification based on architecture decisions and change scenarios. In *Proc. 12th Work. Int. Conf on Software Architecture (WICSA'15)*. IEEE, 65–74.
[79] J. Luftman and H. S. Zadeh. 2011. Key information technology and management issues 2010–11: An international study. *Journal of Information Technology* 26, 3 (2011), 193–204.
[80] G. H. Mead. 1913. The social self. *Journal of Philosophy, Psychology and Scientific Methods* 10, 14 (1913), 374–380.
[81] N. Medvidovic and R. N. Taylor. 2010. Software architecture: Foundations, theory, and practice. In *Proc. Int. Conf. Software Engineering (ICSE'10)*. ACM/IEEE, 471–472.
[82] meta. 2013. *Oxford Dictionary of English* (3rd ed.). Oxford University Press.
[83] metamodel. 2013. *Oxford Dictionary of English* (3rd ed.). Oxford University Press.
[84] J. Miller and B. A Doyle. 1987. Measuring the effectiveness of computer-based information systems in the financial services sector. *MIS Quarterly* 11, 1 (1987), 107–124.
[85] I. Mistrik, R. Bahsoon, P. Eeles, R. Roshandel, and M. Stal. 2014. *Relating System Quality and Software Architecture*. Morgan Kaufmann.
[86] S. N. Mohanty. 1979. Models and measurements for quality assessment of software. *Computing Surveys* 11, 3 (1979), 251–275.
[87] D. Moody. 2005. Theoretical and practical issues in evaluating the quality of conceptual models: Current state and future directions. *Data & Knowledge Engineering* 55, 3 (2005), 243–276.
[88] K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, and P. Vaillergues. 2009. The Squale model–A practice-based industrial quality model. In *Proc. Int. Conf. on Software Maintenance (ICSM'09)*. IEEE, 531–534.
[89] C. Moustakas. 1994. *Phenomenological Research Methods*. Sage.
[90] P. Mulligan. 2002. Specification of a capability-based IT classification framework. *Information & Management* 39, 8 (2002), 647–658.
[91] E. Mumford. 1974. Computer systems and work design: Problems of philosophy and vision. *Personnel Review* 3, 2 (1974), 40–49.
[92] R. R. Nelson and S. G. Winter. 2009. *An Evolutionary Theory of Economic Change*. Harvard University Press.
[93] S. Nidumolu. 1995. The effect of coordination and uncertainty on software project performance: Residual performance risk as an intervening variable. *Information Systems Research* 6, 3 (1995), 191–219.

[94] S. Nidumolu and G. W. Knotts. 1998. The effects of customizability and reusability on perceived process and competitive performance of software firms. *MIS Quarterly* 22, 2 (1998), 105–137.
[95] S. Nidumolu and M. Subramani. 2003. The matrix of control: Combining process and structure approaches to managing software development. *Journal of Management Information Systems* 20, 3 (2003), 159–196.
[96] B. Nuseibeh. 2001. Weaving together requirements and architectures. *IEEE Computer* 34, 3 (2001), 115–119.
[97] C. Okoli and S. Pawlowski. 2004. The Delphi method as a research tool: An example, design considerations and applications. *Information & Management* 42, 1 (2004), 15–29.
[98] R. Parthasarthy and S. Sethi. 1992. The impact of flexible automation on business strategy and organizational structure. *Academy of Management Review* 17, 1 (1992), 86–111.
[99] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber. 1993. Capability maturity model, version 1.1. *IEEE Software* 10, 4 (1993), 18–27.
[100] C. S. Peirce. 1905. The architectonic construction of pragmatism. *Collected Papers of Charles Sanders Pierce* 5 (1905), 3–6.
[101] J. Peppard and J. Ward. 2016. *The Strategic Management of Information Systems: Building a Digital Strategy*. Wiley.
[102] S. Petter, W. DeLone, and E. McLean. 2008. Measuring information systems success: Models, dimensions, measures, and interrelationships. *European Journal of Information Systems* 17, 3 (2008), 236–263.
[103] L. Pitt, R. Watson, and B. Kavan. 1995. Service quality: A measure of information systems effectiveness. *MIS Quarterly* 19, 2 (1995), 173–187.
[104] M. E. Porter and V. E. Millar. 1985. How information gives you competitive advantage. *Harvard Business Review* 63, 4 (1985), 149–160.
[105] T. C. Powell and A. Dent-Micallef. 1997. Information technology as competitive advantage: The role of human, business, and technology resources. *Strategic Management Journal* 18, 5 (1997), 375–405.
[106] C. K. Prahalad and M. S. Krishnan. 1998. The new meaning of quality in the information age. *Harvard Business Review* 77, 5 (1998), 109–18.
[107] R. Pressman. 2014. *Software Engineering: A Practitioner's Approach*. McGrawHill.
[108] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M. Verner, and Markku Oivo. 2017. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software* 123 (2017), 263–291.
[109] K. Rostami, R. Heinrich, A. Busch, and R. Reussner. 2017. Architecture-based change impact analysis in information systems and business processes. In *Proc. Int. Conf. on Software Architecture (ICSA'17)*. IEEE, 179–188.
[110] K. Rostami, J. Stammel, R. Heinrich, and R. Reussner. 2015. Architecture-based assessment and planning of change requests. In *Proc. 11th Int. Conf. on Quality of Software Architectures (QoSA'15)*. ACM, 21–30.
[111] W. Royce. 1987. Managing the development of large software systems: Concepts and techniques. In *Proc. Int. Conf. Software Engineering (ICSE'87)*. ACM/IEEE, 328–338.
[112] D. Russo, P. Ciancarini, T. Falasconi, and M. Tomasi. 2017. Software quality concerns in the italian bank sector: The emergence of a meta-quality dimension. In *Proc. 39th Int. Conf. on Software Engineering (ICSE'17)*. ACM/IEEE, 63–72.
[113] C. Schmidt and P. Buxmann. 2011. Outcomes and success factors of enterprise IT architecture management: Empirical insight from the international financial services industry. *European Journal of Information Systems* 20, 2 (2011), 168–185.
[114] R. Schmidt. 1997. Managing Delphi surveys using nonparametric statistical techniques. *Decision Sciences* 28, 3 (1997), 763–774.
[115] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule. 2001. Identifying software project risks: An international Delphi study. *Journal of Management Information Systems* 17, 4 (2001), 5–36.
[116] M. Shaw and D. Garlan. 1996. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall Englewood Cliffs.
[117] R. Singh. 1996. International standard ISO/IEC 12207 software life cycle processes. *Software Process Improvement and Practice* 2, 1 (1996), 35–50.
[118] B. Slife and R. Williams. 1995. *What's Behind the Research?: Discovering Hidden Assumptions in the Behavioral Sciences*. Sage.
[119] Deloitte Italy S.p.A. 2016. *Payments Service Directive 2 (PSD2): Il Nostro Approccio*. Technical Report. Deloitte Consulting.
[120] A. Strauss and J. M Corbin. 1997. *Grounded Theory in Practice*. Sage.
[121] T. Sunazuka, M. Azuma, and N. Yamagishi. 1985. Software quality assessment technology. In *Proc. 8th Int. Conf. on Software Engineering (ICSE'85)*. ACM/IEEE, 142–148.
[122] CMMI Product Team. 2002. Capability maturity model® integration (CMMI), version 1.1–Continuous representation.

[123] D. Teichroew. 1972. A survey of languages for stating requirements for computer-based information systems. In *Proc. Fall Joint Computer Conference*. ACM, 1203–1224.
[124] J. Tian. 2004. Quality-evaluation models and measurements. *IEEE Software* 21, 3 (2004), 84–91.
[125] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. Cheng, R. Permadi, and R. Feldt. 2012. Evaluation and measurement of software process improvement–A systematic literature review. *IEEE Transactions on Software Engineering* 38, 2 (2012), 398–424.
[126] N. Venkatraman. 1994. IT-enabled business transformation: From automation to business scope redefinition. *Sloan Management Review* 35, 2 (1994), 73.
[127] Stefan Wagner, Klaus Lochmann, Lars Heinemann, Michael Kläs, Adam Trendowicz, Reinhold Plösch, Andreas Seidl, Andreas Goeb, and Jonathan Streit. 2012. The Quamoco product quality modelling and assessment approach. In *Proc. 34th Int. Conf. on Software Engineering (ICSE'12)*. ACM/IEEE, 1133–1142.
[128] Stefan Wagner, Andreas Goeb, Lars Heinemann, Michael Kläs, Constanza Lampasona, Klaus Lochmann, Alois Mayr, Reinhold Plösch, Andreas Seidl, Jonathan Streit, and others. 2015. Operationalised product quality models and assessment: The Quamoco approach. *Information and Software Technology* 62 (2015), 101–123.
[129] S. Wagner and F. Deissenboeck. 2007. An integrated approach to quality modelling. In *Proc. 5th Int. Workshop on Software Quality*. IEEE Computer Society, 1–6.
[130] R. Winter, C. Legner, and K. Fischbach. 2014. Introduction to the special issue on enterprise architecture management. *Information Systems and e-Business Management* 12, 1 (2014), 1–4.
[131] B. Wixom and H. Watson. 2001. An empirical investigation of the factors affecting data warehousing success. *MIS Quarterly* 25, 1 (2001), 17–41.
[132] B. H. Wixom and P. A. Todd. 2005. A theoretical integration of user satisfaction and technology acceptance. *Information Systems Research* 16, 1 (2005), 85–102.
[133] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.
[134] C. Yang, P. Liang, and P. Avgeriou. 2016. A systematic mapping study on the combination of software architecture and Agile development. *Journal of Systems and Software* 111 (2016), 157–184.

Received May 2017; revised May 2018; accepted May 2018