Requirements Eng (2016) 21:87–106
DOI 10.1007/s00766-014-0211-1

ORIGINAL ARTICLE

An ASPIRE-based method for quality requirements identification from business goals

Rachida Djouab · Alain Abran · Ahmed Seffah

Received: 4 July 2012 / Accepted: 14 September 2014 / Published online: 21 September 2014

Abstract Quality requirements are the main drivers for modeling and evaluating software quality at an early stage, and ASPIRE is an engineering method designed to elicit and document the quality requirements of embedded systems. This paper proposes an extension to ASPIRE to identify quality requirements from the business goals of the organization and ensure their traceability. This extension includes a set of added components created from the main concepts of the SOQUAREM methodology, including the BMM (business motivation model), derivation rules, the quality attribute utility tree, the quality attribute scenario template, the quality attribute documentation template, and ISO 9126. The applicability of the extended method is illustrated with a wireless plant control system as an example.

Keywords Software quality engineering · Quality requirements (QRs) · Non-functional requirements (NFRs) · Quality attributes (QAs) · BMM (business motivation model) · Business goals (BGs) · QR elicitation method

R. Djouab (✉) · A. Abran
Department of Software and IT Engineering, École de technologie supérieure, 1100, Notre-Dame Street West, Montreal, QC H3C 1K3, Canada
e-mail: rdjouab@hotmail.com; Rachida.djouab.1@ens.etsmtl.ca; rdjouab@gmail.com

A. Abran
e-mail: Alain.Abran@ele.etsmtl.ca

A. Seffah
Software Engineering and Information Management, Lappeenranta University of Technology (LUT) , CS 20, FI-53851, Lappeenranta, Finland
e-mail: ahmed.seffah@lut.fr

A. Seffah
School of Industrial Engineering and Management, Lappeenranta, Finland

1 Introduction

Over the last two decades, the software engineering and business communities have paid a great deal of attention to quality requirements (QRs) in an effort to identify a system's potential problems early enough, reduce development time and cost, and overcome the communication barriers between the stakeholders (business manager, software developers, and customers) [1–4]. Most related studies [5–11] highlight the importance of QR specifications and their inherent difficulties, and suggest various reasons why those difficulties arise: Quality is an afterthought, quality engineering practices consume a great deal of time and a large amount of resources, and QRs are often difficult to trace and are not usually available for measurement in the early stages of the life cycle.

Various QR engineering methods have been proposed, such as MOQARE [12], Prometheus [15], Soft Goal Notation [16], ASPIRE [17], and Space-UFO [47]. MOQARE (Misuse Oriented QuAlity Requirements Engineering) supports the systematic identification of quality requirements: It was developed by integrating and adapting concepts from other methods (like misuse cases) and provides a general conceptual model of QRs as well as a checklist-based process for deriving them in a top-down fashion [12]. The "quality attribute (QA) model" method defines a process to identify and specify QAs that cut across requirements and integrate them into the functional requirements at an early stage of the software development process [13]. QAs are specified at the requirement elicitation stage using a template, and use cases and sequence

diagrams are extended [14] to achieve the integration. The Prometheus (Probabilistic Method for early evaluation of NFRs) method is an approach to modeling QRs that combines the characteristics of various quality models to meet flexibility, reuse, and transparency requirements. It integrates the quantitative and qualitative approaches to software product line (SPL) quality modeling. Prometheus combines two types of quality models (fixed and customized) to define QRs [15]. The Soft Goal Notation uses a graph structure to record and structure quality goals, design alternatives, and decisions made based on the modeling rationale adopted [16]. The ASPIRE (Analysis of Software Product in Requirements Engineering) method is an experience-based approach that elicits and documents the QAs of embedded systems [17–20]. The Space-UFO method [47] deals with the fit between the software product characteristics and the user's needs for that product (explicit and implicit). The method described as "quality models in software packages" [48] addresses quality requirements definition and decomposition, and the method consisting of "quality specification strategies for embedded systems" [49] is a multi-party chain approach to modeling the quality of embedded software.

Common to all these methods is the issue of how to derive software QRs from business goals (BGs) and have the ability to trace the QRs back to the BGs from which they were derived. BGs are described from the business manager's viewpoint, and the software QRs are described from the software developer's viewpoint, and yet it seems logical to derive software QRs from the business process requirements.

Although some approaches have proposed a link between software requirements and business goals, the QR engineering methods mentioned above do not address the QR issue specifically. However, SIKOSA [21] proposes an approach for deriving software requirements from BGs and uses MOQARE to derive the operational quality goals. The Goal Question Metric (GQM) + Strategies approach proposed by Basili establishes a link between BGs and strategies at all levels of an organization based on goal-oriented measurement [47]. The *Tropos* methodology proposes to model requirements early and reduce the semantic gap between the software system and its operational environment [30]. The SOQUAREM (SOftware QUAlity Requirements Engineering Method) methodology, based on ISO 25030, is designed to manage QRs at the definition phase of the software development life cycle [22] with the aim of supporting the systematic identification of QRs from business specifications. Although this methodology does provide a general process model for QR engineering and a business-based process for deriving QRs in a top-down fashion, its major contribution is to build a bridge between business and software specifications.

This paper proposes an extension to ASPIRE to support the identification of QRs from BGs and provide the ability to trace them back to the BGs from which they were derived using six main concepts of the SOQUAREM methodology: the BMM, derivation rules, the QA utility tree, the QA scenario template, the QA documentation template, and ISO 9126 [23]. Our proposed extension, referred to as ASPIRE^{B} (B for Business), is illustrated with an example adapted from a wireless control plant system [17]. Section 2 describes related work on NFR/QR and BGs. Section 3 describes the ASPIRE method, including its process model [17], and discusses its advantages with particular focus on the difficulty of tracing the elicited QRs back to their respective BGs. Section 4 describes the design of the extended ASPIRE^{B} method with the components added for identifying QRs. Section 5 illustrates the application of the extended method using the adapted wireless plant example. Finally, Sect. 6 concludes with directions for future work.

2 Related work

This section presents the related work on non-functional requirements (NFR)/QR as well as the corresponding variety of definitions, including the NFR/QR relationships with a focus on: (a) quantifying and measuring NFRs/QRs and (b) managing conflicts among NFRs/QRs. It also includes an overview of existing software QR engineering methods and a description of the QR/BG relationships, highlighting the importance of the ability to trace QRs back to the BGs from which they were derived.

2.1 Review of the literature of the NFR/QR

2.1.1 QR definitions

Before delving into the definition of software NFR/QR, it is important to point out the difference between requirements and needs. According to Azuma [23]: "Needs for a product are expectations of stakeholders (of) the effects of the product when it is actually (in use), which means (actions applied to the software product, such) as development, distribution, release, installation, use, and maintenance." In his view, needs are divided into stated needs and implied needs, all of which should be transformed into requirements. The author clarifies the relationships between needs and requirements as follows: "Requirements are the external specification of specific needs that a product is expected to satisfy." The relationship between needs and requirements is illustrated in Fig. 1. Stakeholders' needs (stated and implied) are collected and identified as stakeholders' (business) needs which are next selected and

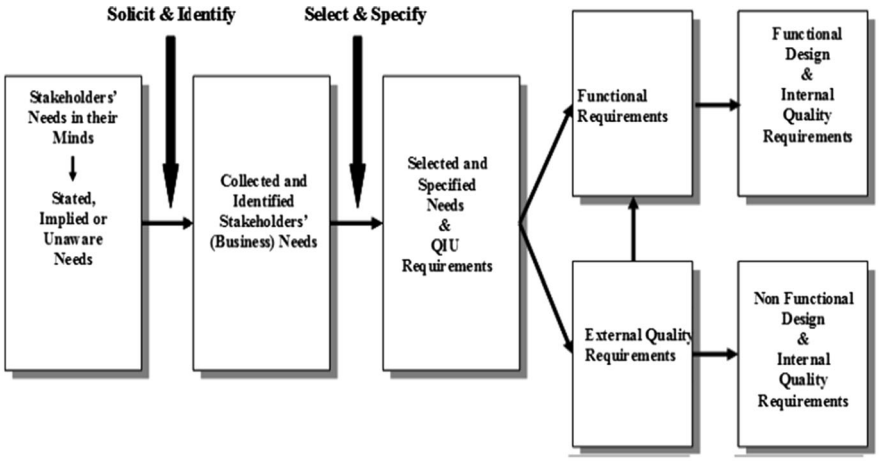

Fig. 1 Relationships between needs and requirements (adapted from [62])

Adapted from: Zubrow Dave. (2004) Software Quality Requirements and Evaluation, the ISO 25000 Series. PSM Technical Working Group. CamegieMellon University. Pittsburgh, PA 15213-3890. http://www.psmsc.com/Downloads/TWGFeb04/04ZubrowISO25000SWQualityMeasurement.pdf. 35 p. Accessed Augut 2014

Fig. 2 QR life cycle model [61]

specified with the quality in use (QIU) requirements to produce functional requirements (FRs) and external quality (EQ) requirements.

The quality requirements of the software product can be categorized into users quality needs (or QIU requirements), EQ requirements, and internal quality (IQ) requirements [25, 62] (Fig. 2).

- Users quality needs constitute the user's view of the quality of the software product when it is used in a specific environment and in a specific context. They are mainly derived from business requirements, functional requirements, and application domain-specific requirements. They are normally used for software validation (i.e., is the software fit for its intended purpose?) [25].
- External quality requirements are mainly derived from quality in use requirements, functional requirements, and application domain-specific requirements. External quality requirements are used for software validation and verification (i.e., is the software built according to specifications?) [25].

Table 1 NFRs approaches [27]

Product-oriented NFRs	Process-oriented NFRs
Usability—ease of use	Modifiability—easily extended with new features
Capacity—volume of users	Maintainability—easily modified (fixes, extensions)
Reliability—mean time to failure—data integrity	Testability—easy to verify
Availability—extent to which system is available to users	Portability—able to work on different platforms
Security—e.g., permissible information flows, or who can do what	
Survivability—e.g., system will need to survive fire, natural catastrophes	
Time/space bounds—workloads, response time, throughput, and available storage space—e.g., "the system must handle 1,000 transactions per second"	
Robustness—time to restart after failure—percentage of events causing failure	

- Internal quality requirements are mainly derived from external quality requirements and development policy and limitations. External quality requirements are normally used for quality verification and control during development [25].

According to Sect. 1.3 (Models and Quality Characteristics) of the Guide to the SWEBOK (Software Engineering Body of Knowledge) [26], the ISO defines "three related models of software product quality (internal quality, external quality, and quality in use) (ISO 9126 Part 1) and a set of related parts (ISO 14598)." Quoting from chapter 11 of the SWEBOK Guide, on the Software Quality

knowledge area (KA), "The software engineer has a responsibility to elicit quality requirements which may not be explicit at the outset and to discuss their importance as well as the level of difficulty in attaining them."

2.1.2 Research on NFRs

The related work on NFRs can be organized into two main complementary categories as illustrated in Table 1 [27]:

- Process-oriented NFRs focus on the process of engineering NFRs during the development of the software system and how NFRs can be used in the design process. The main research challenge here is developing the appropriate methods to facilitate various activities in the engineering of NFRs, such as how to elicit, model, and manage NFRs. It aims to have a way of making appropriate design decisions and qualitative measures are used to study the relationships between goals and the reasoning behind trade-offs.
- Product-oriented NFRs focus on the software system as the product of software development. The main research challenge is evaluating the software product and having a way of measuring the product once it is built. Quantitative measurement is used to ascertain to which degree the product meets its NFRs. Typically, product-oriented NFRs' research investigates various quantitative models to measure and evaluate NFRs.

Borg [28] reports that NFRs are generally hard to get hold of and specify in measurable terms and that most software development methods focus on functional requirements (FRs). His study investigates the knowledge of the real-world treatment of NFRs and presents a case study and a literature survey. Interview series with practitioners have shown that few NFRs are considered in development and that they are stated in vague terms. Moreover, it was observed that processes need to be well suited for dealing with NFRs and suggested that processes can be better suited to handle NFRs by adding the information of actual feature use.

The goal-oriented approaches are the first to tackle NFRs in more depth and in particular, the NFR framework that treats NFRs as softgoals, i.e., goals that need not be absolutely addressed [8]. Several types of contributions are offered by the NFR framework whereby a softgoal satisfies, or denies, another softgoal. Each softgoal or contribution has a label, indicating the degree to which it is satisfied or denied. The NFR framework application is described by an iterative and interleaving process to satisfy softgoals to be addressed or achieved through AND/OR decompositions, operationalizations, and argumentations. The SIG (softgoal interdependency graph) visually represents the process and keeps track of softgoals and their interdependencies, along with the impact of various decisions through labels. The i* family: i* [29], *Tropos* [30], and GRL (goal requirements language) [31] use the concept of softgoal inherited from the NFR framework to deal with non-functionality-related attributes as a first-class modeling concept.

The NFR framework is extended in several ways:

- With the domain-specific quality information to address the problem of quantifying NFRs and resolve design conflicts that arise when trying to satisfy multiple NFRs. These domain characteristics are numeric values that describe some features of the domain [32].
- To build high-quality data warehouse specifications and define catalogs of major data warehouse NFR types (performance and security requirements) and related operational methods, for later reuse during the specification stage [33].
- With a RE-Tools toolkit for modeling NFRs with an extension to support both closed and open world assumptions for goal satisfying label evaluation [34].
- To bridge the gap between NFRs and implementation [35], the extended NFR framework has been implemented (called NFR + Framework) as a modeling language including a softgoal interdependency graph validation tool with a MetaCase MetaEdit + language workbench. The NFR framework was extended with a concept of measurable NFRs that enables one to empirically verify the realization of defined NFRs in a product. The usage of the extended NFR framework was demonstrated with a laboratory case.

Further, Hill and Wang propose a framework for quantifying NFRs that uses quality characteristics of the execution domain, application domain, and component architectures to refine qualitative requirements into quantifiable ones [32]. Conflicts are resolved during the refinement process to produce more useful and realistic non-functional requirements.

Malik et al. [36] suggested an approach to measure NFRs such as cost and performance and to scale NFR like usability. The approach integrates three things: functional requirements (FRs), measurable NFRs (M-NFRs) and scalable NFRs (S-NFRs). Fuzzy logic and Likert scale techniques are effectively used for the handling of discretely measurable as well as scalable NFRs in contrast to vague, ambiguous, imprecise, noisy, or missing NFRs.

On the other hand, one of the important aspects during NFR specification is managing conflicts among interacting NFRs. The research work of Dewi et al. [37] investigates the state of the art on conflicts among NFRs as well as identifies the gaps within the existing methods of managing these conflicts during software development. Authors explain that conflicts arise as a result of inherent

contradiction among various types of NFRs and that two different aspects of conflict could exist: the relationships and the trade-offs. Relationships focus on how NFRs contribute negatively to the other NFRs, while the trade-offs focus on the situation where two or more NFRs could not be achieved at the same time. Their research about conflicts among NFRs mentions the lack of a systematic framework that allows developers to identify and analyze which NFRs of the system are in conflict and which NFRs are not. They argue that such a framework should be able to identify not only the existence of conflicts, but also the type and significance of the conflict and the appropriate potential strategy to resolve the conflict [37]. Finally, they state that managing conflicts among NFRs is based on two points [37]:

(a) *Understanding the nature of complex relationships among NFRs*: Dewi et al. [37] presented various models such as a matrix of positive and negative relationships among NFRs, a model of potential conflicts and cooperation, and a model of relationships among ISO 9126 quality characteristics. In addition, NFRs can interact in a number of ways: They are, indeed, *goals* that are either synergistic or potentially conflicting. A NFR can *refine* its relationship with another NFR (as represented by "And" and "Or"), and it can *contribute* to the relationship (a positive "Make" contribution, or a negative "Break" contribution) [16]. QRs can undergo *decomposition*, when high-quality characteristics are broken down into detailed subcharacteristics, and they can have *influence*, when one characteristic affects the value of other characteristics [15]. The following relationships can also be identified [48]: *collaborative*, *damaging*, and *dependency*.

(b) *Dealing with conflicts among NFRs*: Dewi et al. [37] identified three main activities: conflict identification, conflict analysis, and conflict resolution. The conflict resolution process is complex, owing on the one hand to the difficulty of coordinating the interests and priorities of the many stakeholders involved, and on the other to the need to find a middle ground among the requirements to which they are committed. This process is supported by the requirements prioritization process, which makes the tasks of analysts and project leaders easier to schedule and execute, as well as helping the project manager resolve conflicts, plan for staged deliveries, and make the necessary trade-off decisions. Requirements are labeled high priority, medium priority, or low priority, and various prioritization techniques are available, such as Triage [38] and the Analytical Hierarchy Process (AHP) [40] for prioritizing security requirements; Win–Win, QARCC, S-COST [39, 43] and the requirements traceability technique (RT) [42] for prioritizing quality requirements; and the planning game technique (PG) [41].

Table 2 Definitions of QRs and NFRs, and their similarities and differences

	Main definitions	Similarities	Differences
QRs	Related to stakeholder needs (stated and implied) of a software product [24] Three views of software QRs: quality in use, external quality, and internal quality [25, 26, 62]	Often contradictory [45] Often conflicting Cut across FRs [13] Stated informally [45] Easy to specify, but difficult to validate [19, 28], e.g., Given the statement: "The system shall be easily maintainable" What does "easily maintainable" mean? How does the designer ensure this property? How does a tester validate this requirement?	Terminological Stakeholder needs versus quality in use, external quality, and internal quality defined as quality characteristics of the software product [25, 26]
NFRs	Constraints on services or functions offered by the system [5] Three main types: product requirements, organizational requirements, and external requirements [45] Constraints on the solution [26]		Terminological Constraining effect Defined as product, organizational, or external requirements [45]

2.1.3 Discussion

In summary, NFRs/QRs are essential when considering software quality in that they shall represent the right quality of the intended software. For a number of authors, the QR and NFR terminologies are interchangeable: Table 2 summarizes the main definitions of QR/NFR given by some of them, along with their major similarities and differences. Some authors associate NFRs with QRs, while others dissociate them from QRs [44]. For instance, Sommerville [5] defines NFRs as constraints on the services or functions offered by the system. This includes timing constraints, constraints on the development process, and standards. NFRs often apply to a system as a whole, rather than to individual system features or services. They may be classified into three main types: product requirements, organizational requirements, and external requirements. They are generally hard to model, usually stated informally, often contradictory, difficult to enforce during

Table 3 Classification of software QR engineering methods

QR engineering methods	Description
MOQARE [12]	(Misuse Oriented QuAlity Requirements Engineering): QAs are described by business goals and the misuse concept
QA model [13]	QAs are specified by the aspect concept
Prometheus (a method for modeling and evaluating the quality of software products early in the development process) [15]	Quality goals are defined by the system users and other stakeholders by applying the MGT (Measurement Goal Template)
Soft Goal Notation [16]	Quality goals are recorded and structured using a goal graph
ASPIRE (Analysis of Software Product In Requirement Engineering) [17]	The QAs of embedded systems are represented using Soft Goal Notation
Space-UFO [47]	The quality needs of stakeholders (manager, designers, and customers) are specified from business issues
Quality models for software packages [48]	The ISO 9126 quality model is proposed to deal with QR definition and decomposition
Quality specification strategies for embedded systems [49]	The multi-party chain method is proposed to deal with the quality of embedded software

development, difficult to evaluate for the customer prior to delivery and hard to make them measurable requirements. Software developers would like to state them in a way that they can measure how well they have been met [45]. In addition, the quality standards ISO 25030 and ISO 9126 [23, 25] identify three categories of software QRs (quality in use, external quality, and internal quality requirements) that represent different stakeholder's quality views (customers, developers, end user, business manager, and other stakeholders). The standard ISO 25030 provides a guide to use the quality model for requirement definition, but the difficulty remains in the way QRs are measured, tracked, validated, and managed. These challenges will be addressed in more depth in the future research work (see Sect. 6).

Examples of NFRs include safety, security, usability, reliability, and performance requirements [45], although in ISO 9126 and ISO 25030, they are considered as QAs. At the same time, the Guide to the SWEBOK [26] defines NFRs in its Software Requirements KA and QRs in its Software Quality KA. In addition, there are a few hybrid NFR-QR engineering methods (e.g., the NFR framework deals with quality goals and the ASPIRE method deals with QRs), while other methods use the NFR terminology instead of the QR terminology (Prometheus [15] deals with quality goals). In this paper, we use the QR terminology as defined in ISO 25030 [25].

2.2 QR engineering methods

Over the past two decades, research on software QRs has resulted in the development of several software QR engineering methods [50] (Table 3). MOQARE is applicable to QRs derived from business goals and the misuse concept. The "QA model" method defines QAs as crosscutting concerns and specifies them in a template. The Prometheus method provides a detailed process for building a quality model for a specific domain project. Soft Goal Notation is applicable to all types of QR and focuses on the documenting and negotiating QRs, treating them as goals to be achieved during the design process and using them to drive the design, support architectural design, and deal with change [46]. ASPIRE is designed to elicit complete instantiations of QAs, using ISO 9126 to define high-level QAs and Soft Goal Notation to capture dependencies and conflicts among QAs. Space-UFO identifies quality needs according to the characteristics of the business system. The method that selects quality models for software packages deals with QRs in that domain, and the method that selects quality specification strategies for embedded systems captures QRs in a multi-party chain model.

In most of the software QR engineering methods described above, there is no link between the QRs and the BGs of the organization [50, 51], nor is there any indication of how to describe and model business characteristics in a structured way in order to identify the QAs. In addition, they focus mainly on notation and techniques for modeling the qualities of a software system without taking into account the relationships between the system's QRs and the organization's BGs. Although this relationship may seem to be self-evident, it is apparently not well understood in the business literature. There is clearly a need to align QRs and BGs, and to devise concrete means for deriving QRs from business process models.

2.3 How are software QRs related to BGs?

Achieving BGs is crucial for system success. Two definitions of BGs are the following: "…sometimes referred to as mission statements,… the things an organization hopes to achieve during its time in operation" [52], and "…the foundation on which software systems are justified, analyzed, and built," the purpose of these systems being "*to realize business or mission goals*" [53]. Kazman and Bass [53] proposed a categorization of possible BGs for software-intensive systems to enable stakeholders to obtain guidance in their elicitation, expression, and documentation. To construct this categorization, they created an affinity diagram made up of 190 distinct BGs, elicited using the architecture trade-off analysis method (ATAM) [62].

Because of the critical importance of BGs, the challenging task of linking them with software requirements needs to be addressed. The method developed in [54] bridges the domains of business process modeling and software requirements engineering to resolve the following issues:

(1) The differing syntax and semantics of the languages used in the two domains;
(2) The different ways in which business process quality and software process quality are perceived in the two domains.

In addition, Basili et al. [55] proposed an approach to link the BGs and strategies (GQM + Strategies) at all levels of an organization by means of goal-oriented measurement. The approach is based on the integration of GQM into measurement programs to ensure that goals and strategies are aligned at all levels of the business, from the highest strategic level to the individual development project level.

The research study of Bleistein and Cox [56] proposes an integrated approach to requirements engineering for the organizational IT. Their business strategy alignment is based on a goal-oriented requirements engineering notation to model the organization's business strategy, and the use of a single model (according to Jackson's problem diagram framework) to represent both the business context and the IT context.

Castro and Mylopoulos [30] proposed a software development methodology (called *Tropos*) to model requirements at the earliest stage of the process. It is based on the i_* organizational modeling framework (with its notions of *actor*, *goal*, and (actor) *dependency*) for modeling early and late requirements, and for architectural and detailed design. The methodology is aimed at reducing the semantic gap between the software system and its operational environment.

In Bleistein and Aurum [57], a Strategy-Oriented Alignment in Requirements Engineering (SOARE) approach is proposed to align requirements for e-business systems and the business strategies they are intended to support. The SOARE approach uses goal modeling to represent the business strategy in a requirements engineering context and to link high-level strategic objectives to low-level requirements through goal refinement.

In Morris [58], the process of translating the corporate strategy into projects and programs is addressed. Strategy implementation and translation are discussed through four case studies and a survey that reveals the processes, practices, and people issues involved in efforts to systematically move from strategy to projects and programs.

In Clements and Bass [59], the aim is to reliably elicit BGs and to understand how they influence QA requirements and architectures. As QA requirements are often captured and represented poorly in requirements specifications (which focus on functionality), these authors suggest BGs as drivers of the design, creation, and evolution of software-reliant systems. They examine BGs from the point of view of the software architect and present a wide range of business goal categories from the business literature to produce a BG classification. They use the concept of a business goal scenario (a systematic way to elicit and express BGs) for eliciting architecturally significant BGs and for producing a set of derived QA requirements.

In Yu and Gross [46], a strategic agent-oriented and goal-oriented approach is proposed to systematically relate business goals to architectural design decisions and architectural structures during software development and evolution. Their research is aimed at understanding and modeling the relationships between BGs and system qualities, since most of the system quality requirements (also referred to as non-functional requirements) originate at the business level and are determined during the architectural design process. A notation including goals, strategic agents, and intentional dependency relationships is used to support the architectural modeling and rationale.

The SOQUAREM methodology is proposed in [22] to support the identification of QRs from BGs and functional requirements. It provides a general process model of QR engineering and a business-based process for deriving QRs in top-down fashion. The ISO 25030 quality standard is used to support the QR derivation process.

Section 3 presents the initial ASPIRE method, and Sect. 4 presents the proposed ASPIRE^{B} method and its extended components used to derive the QRs from the BGs.

3 The ASPIRE method for QR engineering

3.1 Description of the method

ASPIRE is referred to as a systematic and experience-based approach that documents and analyzes the QRs[1] of embedded systems, with the aim of arriving at a minimal and sufficient set of measurable QRs [17]. As defined by Doerr et al. [17], ASPIRE is "*a process for common treatment of the various high-level* QA*s, e.g., maintainability, efficiency, portability, usability, security, and reliability.*" QRs are instantiations of QAs, which ASPIRE describes as "*a certain value… that should be achieved in a specific project.*" A QR "*constrains a* QA *by determining*

[1] The term "NFR" is used in the ASPIRE method, while in this paper, we prefer to use the term "QR."

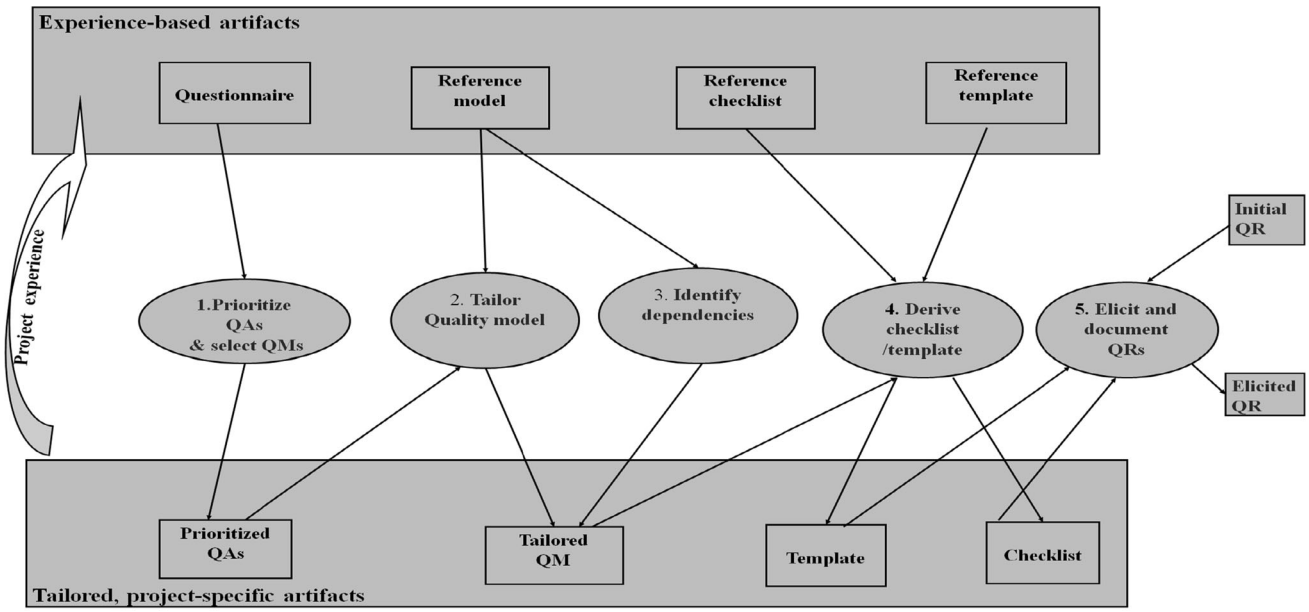

Fig. 3 ASPIRE process [adapted from 17]

Table 4 Activities of the ASPIRE process

ASPIRE process			
Activities	Stakeholders involved	Techniques/ artifacts used	Artifacts produced and results
1. Prioritize QAs and select QMs	Software developer	Use cases Questionnaire ISO 9126 Soft Goal Notation	Prioritized QAs
2. Tailor QM	Domain experts Software developer	Reference QM Workshops Prioritized QAs	Tailored QM
3. Identify dependencies	Domain experts Software developer	Reference QM	Tailored QM
4. Derive checklist and template	Domain experts Software developer	Reference checklist Reference template Tailored QM	Checklist Template
5. Elicit and document QRs	Software developer	Checklist Template Initial QR Workshops	Elicited QR

a value for a metric associated with that QA." For example, associated with the statement, "*the database for our new system shall handle 1,000 queries per second*" are two constraints: (a) the QA, "*database workload*" and (b) the associated ratio, "*number of jobs per unit of time.*"

The ASPIRE process consists of two basic stages and five activities (Fig. 3):

1. *Quality Model* (*QM*) *Tailoring:* The reference quality model is tailored to the needs of the project, and checklists and templates are produced for the next step, which is elicitation.
2. *Elicitation*: Based upon the previously created artifacts, the different types of activities that formulate the QRs are defined. These QRs are analyzed for possible conflicts and documented. The checklist provides the means to identify these conflicts and to resolve them [17]. Table 4 describes activities of the ASPIRE process.

3.2 Applicability of the ASPIRE method

ASPIRE has illustrated its applicability in three case studies for embedded systems [17]. The first case study is the wireless plant control (WPC) system which develops embedded systems for controlling and monitoring a manufacturing plant by means of handheld computers and cell phones. The elicited QAs are efficiency, reliability, and

maintainability. The second case study deals with the multi-functional printer (MFP) system that provides digital office equipment. The elicited QA is efficiency. The third case study addresses the geographical information system (GIS), a web-based geographical information system for farmers to access maps and data related to their area from the Internet, using a standard browser. The elicited QA is security. The benefits of the ASPIRE method as revealed in the three case studies are as follows: (a) Additional and important QRs are identified for the project; (b) measures are found for most of the QRs; (c) the QR elicitation in the workshop is encouraged by the reference QM and checklists; (d) a common understanding is created as a result of standardizing the terminology; and (e) little domain knowledge and few domain experts are needed in workshops, given the reference QM and checklists.

However, for the WPC [17], the tailoring process is time-consuming, owing to the failure to structure the workload for the workshops. In addition, no link is made between the organization's BGs and the elicited QAs. This is an important step that should have been taken, as mentioned in Sect. 2.2.

3.3 Analysis and discussion of the ASPIRE method

The ASPIRE method is organized around stakeholder workshops, where the QMs are selected and tailored, and then used to elicit and document the QRs. The method starts by prioritizing the high-level QAs that are the most important to the project and selecting the QMs associated with these QAs. Next, these selected QMs are tailored to the needs of the project during the workshops. The checklists and templates are derived from the QMs to be used (in the workshops) for the elicitation process, where the QRs are defined. The dependencies among the QAs (general and lower level) in the QMs are included in the checklists and are used to identify the QRs and the conflicts among them. As stated in [17], the objective of the ASPIRE method is to achieve a minimal and sufficient set of measurable QRs. It is also demonstrated in the three cases studies that the elicited QRs are mostly measurable:

(a) In the WPC case study, the elicited QRs are efficiency, reliability, and maintainability, and 49 of 54 QRs are measurable;
(b) In the MFP case study, the elicited QR is efficiency and all 16 QRs are measurable, and
(c) In the GIS case study, the elicited QR is security, and although it was hard to attach measures to the QAs, the set of QRs are testable.

From these observations, more evaluation and validation of the ASPIRE method is required since not all QRs can be measured, or easily quantified [28, 32, 35, 36]. Another aspect to be discussed is the conflicts management in ASPIRE. As defined by the authors, the dependencies among the QAs (general and lower level) in the QMs are included in the checklists and are used to identify the QRs and the conflicts among them. However, there is no indication in the ASPIRE references as to how to address the management of these conflicts. These two challenges remain to be addressed in more depth in the future research work (see Sect. 6).

4 The extended method: ASPIRE^{B} (business ASPIRE)

4.1 The ASPIRE^{B} process

To identify quality requirements from the business goals of the organization and ensure their traceability, the improvements proposed to the ASPIRE method include the addition of BGs, as well as new activities and concepts for identifying and eliciting QRs. The six main components added to the ASPIRE method are as follows—see also the gray elements in Fig. 4.

1. The BMM concept, which focuses on the organization's intentions, motivations, and rationale, as well as on their BGs, which are important drivers for identifying the QAs of the software product. The BMM contains and organizes the elements of business governance: vision and mission, influencers and assessments, goals and objectives, and strategies. The goal, strategy, and influencer elements are used in the extended method, which we refer to as ASPIRE^{B} and which is presented in more detail in Sects. 4.2 and 4.3;
2. A QA scenario template (Table 5), to provide QA meanings in detail, so that the right one can be inferred;
3. A QA utility tree (Fig. 7), to make it possible to trace the QAs back to the BGs from which they are derived;
4. A QA documentation template, to record the QAs and enhance the communication between the stakeholders involved (Table 6);
5. A set of derivation rules at the tailoring stage to derive QAs from the BGs [50];
6. The quality model of ISO 9126 [23] to support ASPIRE^{B} during the derivation process of the QAs from the BGs.

The BMM and the derivation rules are described in more depth in [50, 61].

The ASPIRE^{B} process consists of two stages: the tailoring and elicitation (separated by the purple line) and eight activities (Fig. 5):

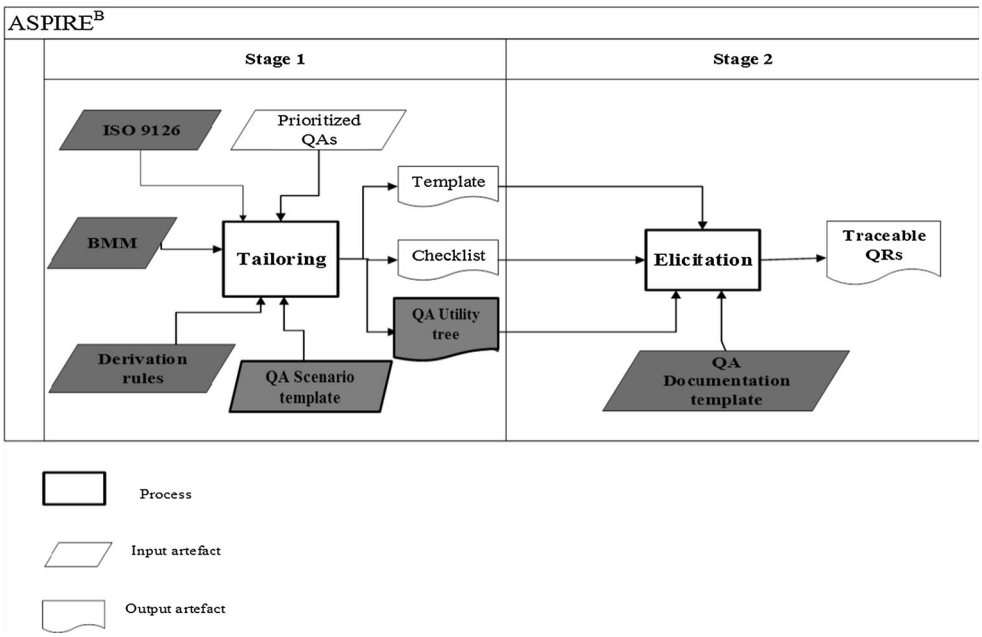

Fig. 4 ASPIRE^B: The main components added (in *gray*) to the ASPIRE method

Table 5 QA scenario template

QA	Definition of the QA				
Sub-QAs	QA11	…			QA1n
Scenario number	N1	…			Nm
Scenario action	Action11	Action21	…	Action2j	Action_nm
Activity undertaken to achieve the QA					
Scenario asset	Asset1	Asset21	…	Asset2j	Asset_nm
Any part of the system (hardware, software, personnel, development process, data) involved in achieving the QA					

Table 6 QA documentation template

QA	Definition of the QA according to the ISO 9126 quality model
Tailoring context	Description of the QA in its project context of use
Source	Information sources, like stakeholders, vision document, use case artifacts
Target stakeholders	Class of stakeholders interested in the QA, e.g., manager, software developers, customers, other stakeholder(s)
Business goals (BGs)	The BGs from which this QA derives
Priority	The importance of the QA for the stakeholders (MAX, HIGH, LOW, or MIN)
QRs elicited	The QRs elicited following the tailoring process

(1) The *tailoring* stage, which is composed of seven activities (Table 7), starts with the setting up of the *BGs*, the *derivation rules*, and the *QA scenario template*. The activities added at this tailoring stage are (shaded activities in Fig. 5—left-hand side) as follows:

1. Identification of the BGs from the BMM concepts;
2. Derivation of the QAs from the BGs according to the derivation rules and ISO 9126;
4. Derivation of the QA scenarios by using the QA scenario template;
6. Derivation of the QA utility tree from the tailored QM and the QA scenario template.

(2) The *elicitation* stage (Fig. 5—right-hand side) is enriched with the QA documentation template to facilitate communication among the stakeholders and its process to elicit traceable QRs is presented in more detail in Fig. 6.

(3) The eighth activity added (Fig. 5; Table 7) is related to the analysis of conflicting QAs from the checklists and their documentation from the QA utility tree and templates.

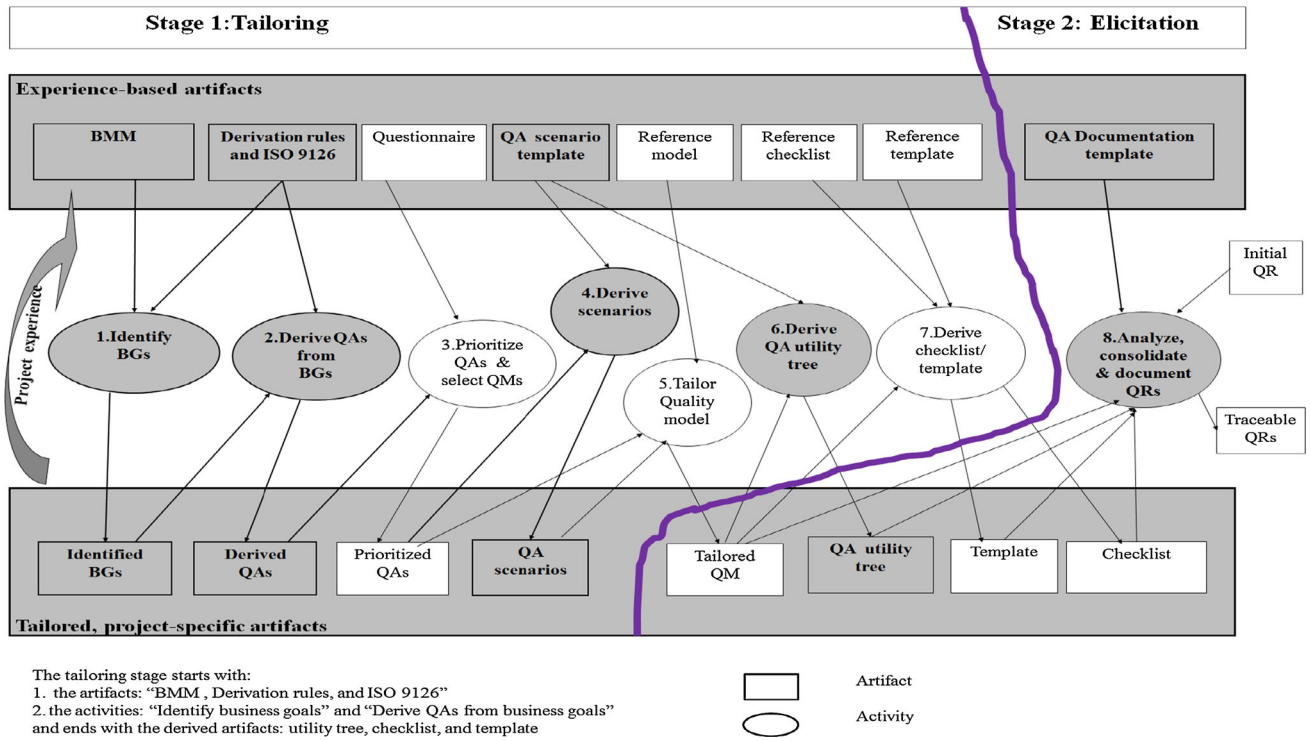

Fig. 5 Activities for the ASPIRE^{B} process (activities and components added are in *gray shade*)

4.2 Description of the added components of the ASPIREB method

4.2.1 BMM (business motivation model)

The BMM is used to define the motivation of the business context, to state the goals and subgoals of the business and the related strategies, and to identify relevant stakeholders, along with their expectations. The main elements of the BMM used to derive the QAs are as follows:

(a) The *goals* of the business, which are refined to specify business goals and to derive QAs;
(b) The *strategies* to be implemented by the business to achieve its goals; and
(c) The *influencers* (external and internal) affected by the goals, required to identify the actors responsible for achieving the QAs and to derive the actions to be undertaken by these actors to achieve the QAs.

4.2.2 Derivation rules

The derivation rules consist of a set of statement, refinement, and linkage rules [50]. The statement and refinement rules are applied to ensure that the BGs are stated according to a business mandate and are detailed according to the business strategies adopted, the technological constraints, and the organizational culture of the business. The linkage rules are applied to ensure that the QAs are derived from the refined BGs according to the stakeholders' quality needs and the ISO 9126 quality model. These are examples of the rules:

(a) Each BG is stated according to the domain characteristics of the business.
(b) Each BG is refined according to the technological constraints, regulations, and desired compliance.
(c) Each QA is derived according to the organization's business quality needs, the refined BG, and the ISO 9126 quality model.

4.2.3 QA scenario template

The QA scenario template is used to build the QA scenarios and to structure the tailoring process during the workshop sessions. It provides a context for detailing the QAs and for helping the stakeholders select the right QA. The template contains the definition of each QA and the scenarios detailing its meaning. Each QA can have a maximum of three scenarios (for the operationalization purpose of the utility tree), and each scenario details the meaning of the QA, in order to infer the right one. A scenario is composed of (a) the action undertaken to achieve the QA and (b) the asset to which the QA is applied. Table 5 lists the QA scenario template items.

Table 7 Description of the activities for the ASPIRE^{B} process (components added are in bold characters)

ASPIRE^{B} process			
Activities	Stakeholders involved	Techniques/ artifacts used	Artifacts produced and results
1. Identify business goals	Software developer Business manager	**BMM** **Derivation rules :statement and refinement rules**	**Identified BGs**
2. **Derive QAs from business goals**	Software developer Business manager	**Identified BGs** **Derivation rules: linkage rules** **ISO 9126**	**Derived QAs**
3. Prioritize QAs and select QMs	Software developers Business manager	Use cases Questionnaire Derived QAs Soft Goal Notation	Prioritized QAs
4. Derive QAs scenarios	Domain experts Software developer	Prioritized QAs **QA scenario template**	**QA scenarios**
5. Tailor QM	Domain experts Software developers	Reference QM Workshops **QA scenarios** Prioritized QAs	Tailored QM
6. **Derive the QA utility**	Software developer	Tailored QM **QA scenario template**	**QA utility tree**
7. Derive checklist, template	Software developers Domain experts	Reference checklist Reference template Tailored QM	Checklist Template
8. Analyze, consolidate, and document QRs	Software developers Business manager	Workshops **QA documentation template** Checklist Template **QA utility tree** Tailored QM Initial QR	Traceable QRs

4.2.4 QA utility tree

The utility tree is provided to make it possible to trace the QAs back to their original BGs. It shows the derived QAs, their refined BGs, and the QA scenarios generated in a goal graph (Fig. 7) structured in three levels. The **business level** represents the stated BGs and the refined BGs. The priority of the related refined business goal is also represented. The **QA at the system level** shows the QAs derived from the detailed business goals. The actor responsible for achieving the QA is also represented at this level. The "actor" concept is defined according to the use case approach: "*An actor is a role that a user or another system has*" [14]. The **scenario at the system level** details the QAs with scenarios from the QA scenario template.

4.2.5 QA documentation template

The template proposed to document the QAs includes the main items related to the elicited QRs and summarizes the QA data used during the tailoring and elicitation stages (Table 6). The items on this template describe the QM tailoring context, the source of the information contributing to the definition of the QA (stakeholders, documents), the target stakeholders involved in the workshops, the BGs contributing to the derivation of the QA, the priority of the QA, and the elicited QRs. This template is used to help stakeholders communicate about QAs.

4.2.6 ISO 9126

The ISO 9126 quality model [23] helps stakeholders focus on the most widely recognized quality characteristics and is used to infer the right QA from the refined BGs.

4.3 Description of the added activities of the ASPIRE^{B} method

ASPIRE^{B} includes the following added activities (in bold characters in Table 7):

1. *Identification of the BGs*: The important BGs are identified from the BMM concept [50]. They are obtained by applying the statement rule: *Each business goal is defined according to the "desired results" item of the BMM*. The BGs are detailed according to additional business information, such as organizational culture, regulations and guidelines, technological constraints, and business strategies to achieve BGs [50]. The refined BGs are obtained by applying the refinement rule: *Each business goal is detailed according to technological constraints, existing regulations and compliance, and high-level functional requirements.* The outputs of this activity are the main BGs and the refined BGs of the system.
2. *Derivation of the QAs from the BGs:* The QAs are derived (along with their actors and actions undertaken to achieve them) from the BGs according to the linkage rule: *Each QA is derived according to business quality needs, the refined business goals, and ISO 9126*. Relevant actors associated with the achievement

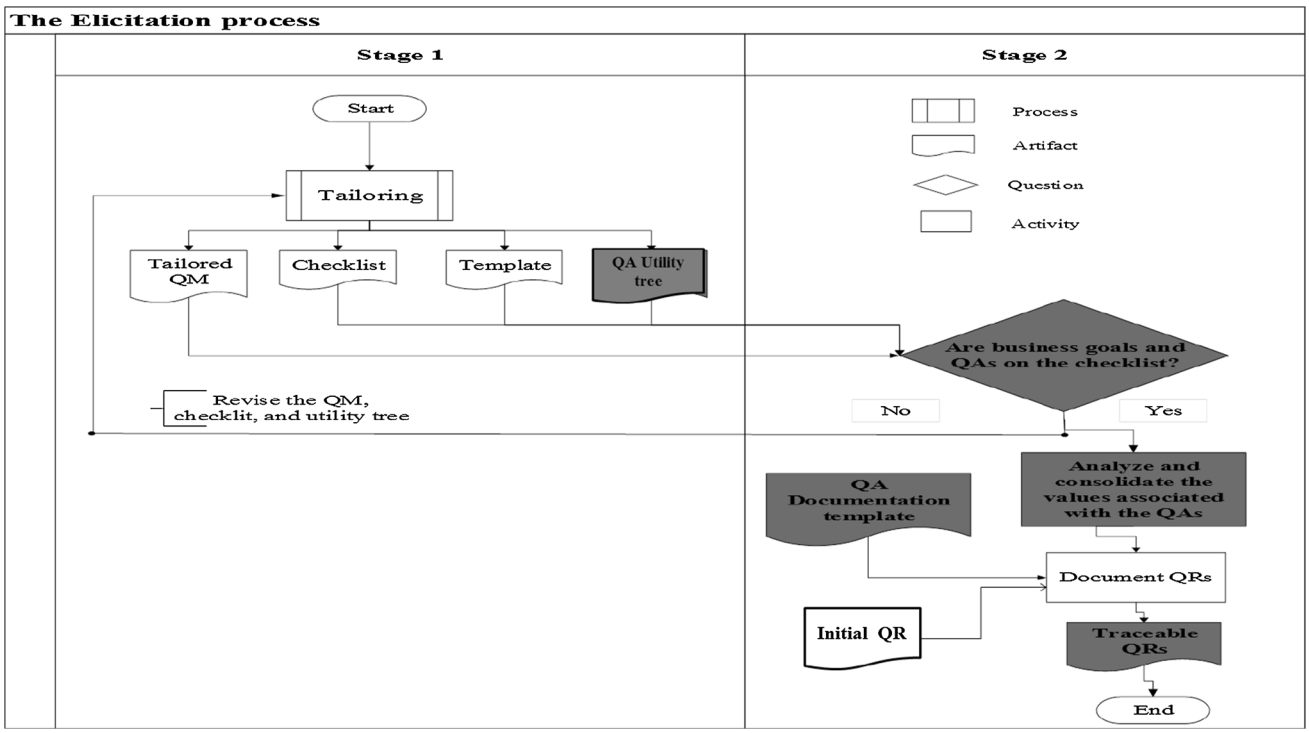

Fig. 6 ASPIRE^{B}: The process of the elicitation stage (components and activities added are in *gray shade*). Description of the added components of the ASPIRE^{B} method

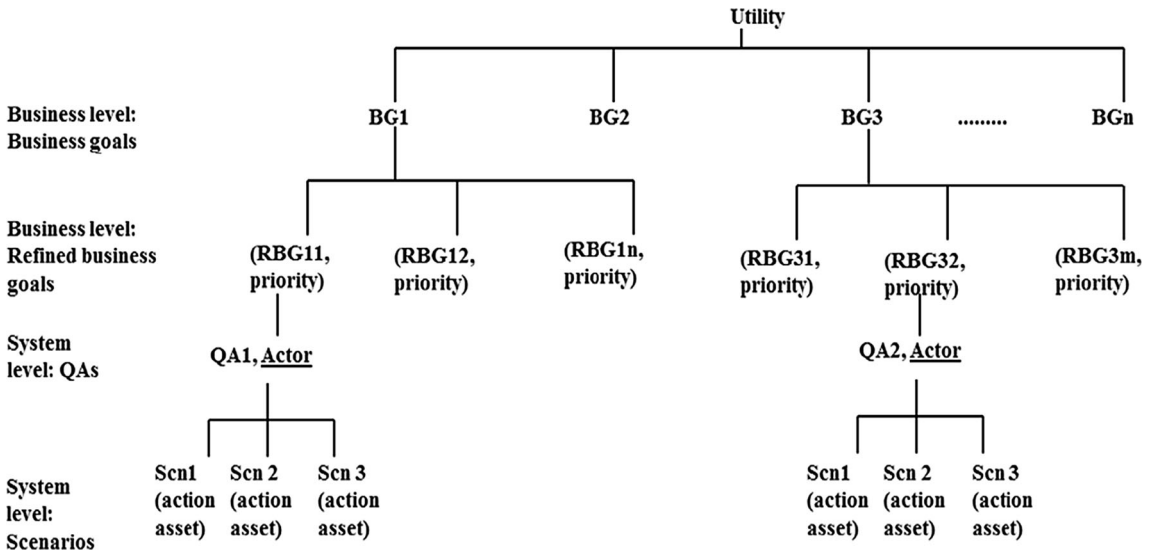

Fig. 7 Utility tree of the BGs with their derived QAs

of QAs are derived from the BMM concept. Actions are identified with questions about the possible actions that could be derived from the refined BGs. The output of this activity is the derived QAs structure based on the following data: BGs, refined BGs, priority, derived QAs, relevant actors, and actions (Table 8).

4. *Derivation of the QA scenarios:* The QA scenarios are built according to the following:

- Structure of the QA scenario template (Table 5);
- Derived QAs and relevant actors and actions required to achieve them (Table 8).

The QA scenarios are built as follows:

- The Action item of the QA scenario template is mapped to the relevant action fields of the derived QAs (Table 8);

Table 8 Derived QAs

Business goals	Refined business goals	Priority	Derived QA	Relevant actors	Actions
BG1: improve the productivity of the manufacturing system	BG11: increase the use of the handheld computers and cell phones	H	Efficiency Boot Time	WPC system device	Specify the start-up time of the software Specify the start-up time of the hardware
		H	Throughput	WPC system device	Read data from the machines in the plant Receive alarms Transmit the plant device data Control the plant device data
			Workload Distribution	WPC system device	Distribute the work

- The Asset item of the QA scenario template is specified from the definition of the refined BG and the relevant action fields of the derived QAs (Table 8).

The priority of a QA is defined according to its importance relative to the actor.

The outputs of this activity are the QA scenarios list.

6. Derivation of the QA utility tree: The QA utility tree is derived from the tailored QM and the QA scenario template.
8. Analysis, consolidation, and documentation of QRs: The QAs are analyzed for conflicts, consolidated from the checklists, and documented in the QA documentation template to elicit traceable QRs.

Table 7 describes the activities of the ASPIRE^{B} process and the components added are in bold characters.

In the next section, the application of the proposed ASPIRE^{B} method is illustrated with an example.

5 ASPIRE^{B} application

This section presents an example to illustrate the use of the added components of the ASPIRE^{B} method, in particular: (a) integration of BGs into the QM and checklist; (b) use of the QA scenario template during the tailoring stage to help infer the right QA; and (c) use of the QA utility tree to trace the elicited QAs to their respective BGs.

5.1 Scope of the example

This example is based on the wireless case study described in [17] and was selected because it provides initial data on the QAs and efficiency requirements to illustrate the ASPIRE^{B} process. The focus in the example is on the efficiency requirements and the productivity BG. The target stakeholders are the business manager and the software developers. The data used in the illustrative example include data from the wireless case study (efficiency QM and efficiency requirements) [17] and from the literature (QM tailored for efficiency, efficiency checklist, and efficiency QRs) [19].

The data generated in the example are represented in the following artifacts:

- BGs of the wireless system; QM for Efficiency QA (Fig. 8);
- Derived QAs (Table 8);
- Efficiency scenarios (Table 9);
- Tailored QM for Productivity BG and Efficiency QAs: Boot Time and Throughput (Fig. 9);
- Utility tree of the productivity business goal BG1 with its derived Efficiency QA (Fig. 10);
- Elicitation of the Boot Time and Throughput Efficiency QAs (Fig. 11);
- Efficiency documentation template (Table 10).

The detailed application of the BMM concept and the derivation rules is described in more depth in [50, 61].

5.2 Application of the example

The added activities undertaken in the example (according to Table 7) are as follows:

(1) Identification of BGs of the WPC using the BMM concept, and (2) Derivation of the QAs;
(4) Derivation of the QA scenarios, and (5) Tailoring the QM to the WPC project;
(8) Documentation of Efficiency QRs.

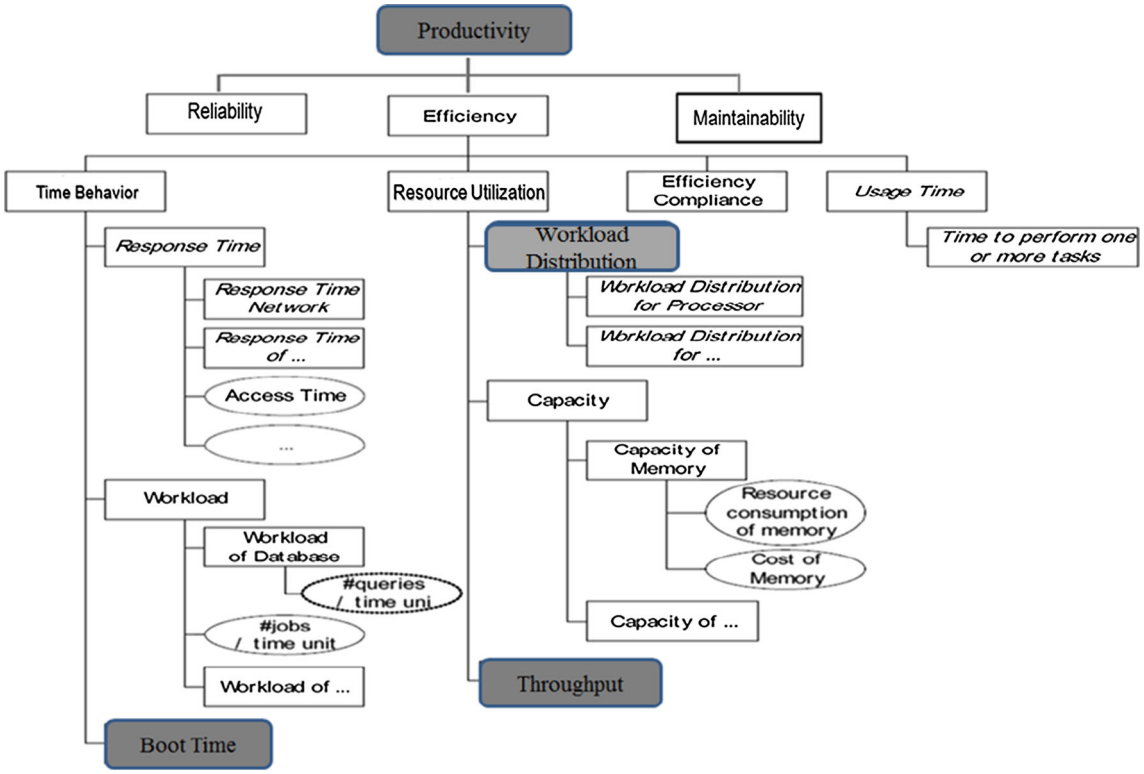

Fig. 8 QM for Efficiency QA [17]

Table 9 Efficiency scenarios

QA		Efficiency									
Sub-QA		Boot Time		Throughput						Workload Distribution	
Scenario number		1		3						1	
Scenario	Action	S11	Start	S21	Transmit data	S22	Control data	S23	Receive alarms	S31	Distribute work
	Asset		WPC system		WPC system		WPC system		WPC devices		WPC devices

5.2.1 *Identification of the BGs of the WPC and derivation of the QAs*

Since the organization wants to improve the monitoring and control of its manufacturing system by wireless means, and also wants to be highly competitive in the marketplace, business goal BG1 is obtained by applying the statement rule: *Each business goal is defined according to the "desired results" item of the BMM*—that is, BG1: "Improve the productivity of the manufacturing plant system." The organization would have to consider different wireless standards and integrate the emerging technological applications to meet the customer's needs. BG1 is refined into business goal BG11: "Use handheld computers and cell phones" by applying the refinement rule: *Each business goal is detailed according to technological constraints and existing regulations and desired compliance.*

BG1: "Improve the productivity of the manufacturing plant system"
BG11: "Use handheld computers and cell phones"

The Efficiency QA is derived from the BGs identified (BG1 and BG11) by applying the linkage rules—see Table 7:

- Each QA is derived according to the business quality needs, the refined business goals, and ISO 9126.
- Define the relevant actors who should achieve the quality attribute from the "external influencer" item of the BMM.
- Define the relevant actions from definition of refined business goals and the internal influencer of the BMM.

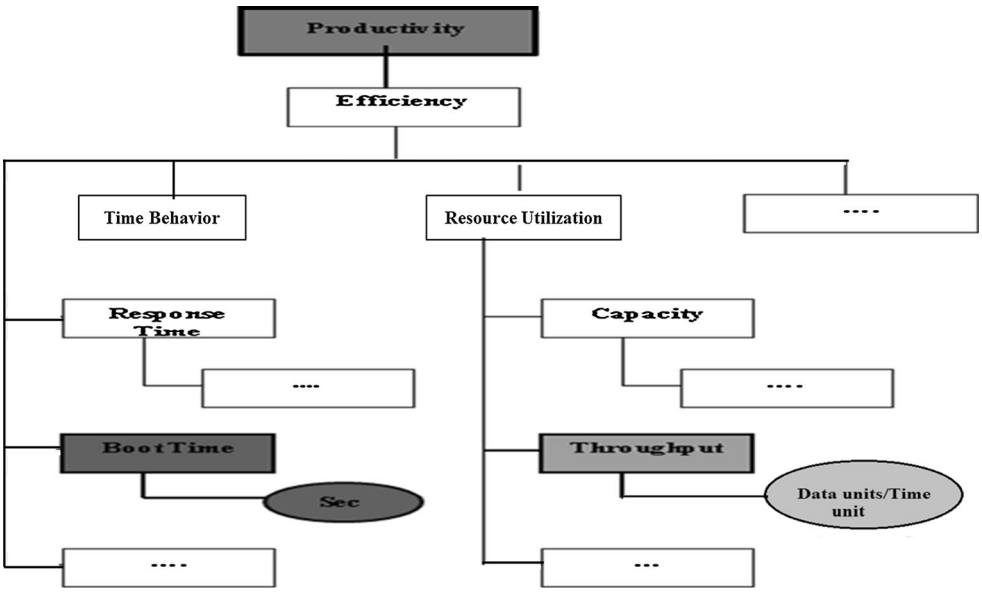

Fig. 9 Tailored QM for productivity BG and Efficiency QAs: Boot Time and throughput [adapted from 19]

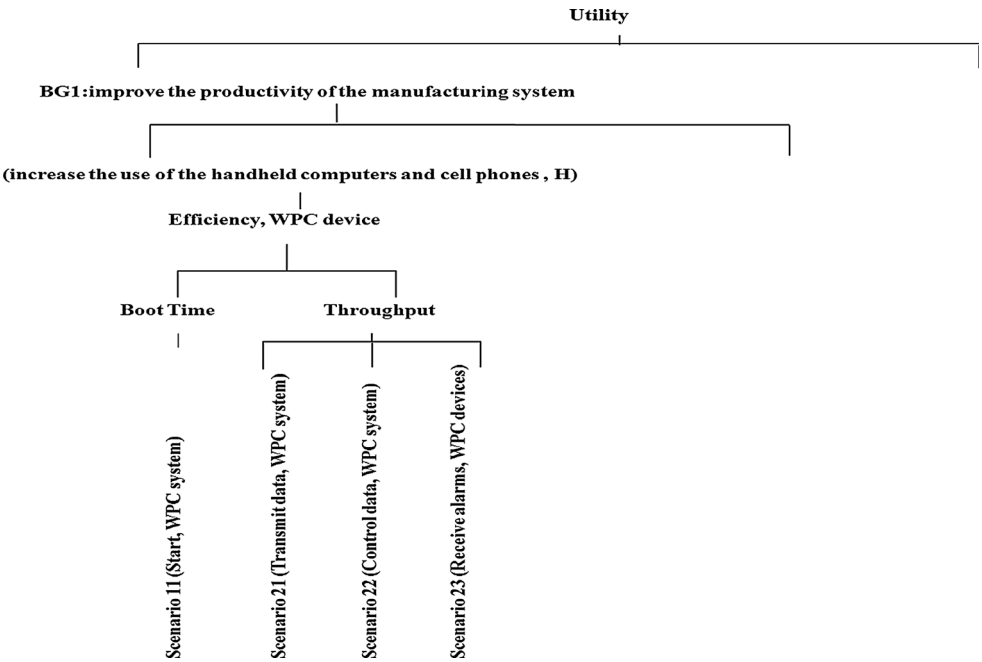

Fig. 10 Utility tree of the productivity business goal BG1 with its derived Efficiency QA

Figure 8 shows the reference QM for the business goal productivity and the Efficiency QA prior to the tailoring stage. Efficiency is broken down into three sub-QAs (in gray shaded rounded rectangles in Fig. 8), according to the efficiency quality model:

- Boot Time (time required to start the component);
- Throughput (amount of data to be transmitted and controlled);
- Workload Distribution (how the work is distributed among the components).

5.2.2 Derivation of the QA scenarios and tailoring the QM to the WPC project

This activity is aimed at structuring the process of tailoring the efficiency QM. The QA scenarios associated with the Boot Time, Throughput, and Workload Distribution are built as follows (Table 9):

- The Action item of the QA scenario template is mapped to the relevant action fields of the derived QAs (Table 8).

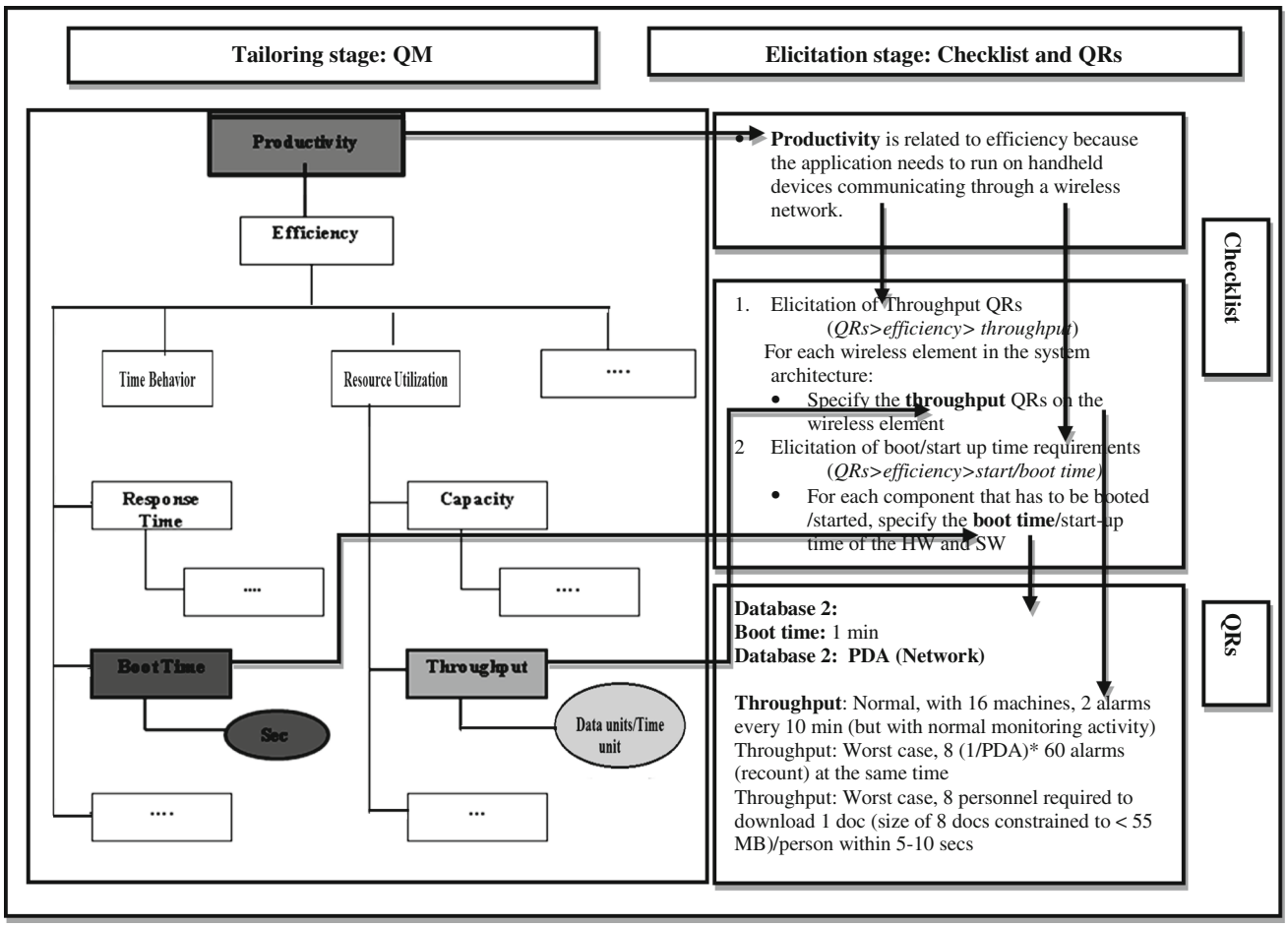

Fig. 11 Elicitation of the Boot Time and throughput Efficiency QAs [adapted from 19]

Table 10 Efficiency documentation template

Efficiency	Defined according to ISO 9126
Tailoring context	A wireless plant control system to monitor and control a manufacturing plant using handheld devices and cell phones
Sources	Stakeholders, BMM, and vision document
Target stakeholders	Business manager, software developers
Business goals	BG1: productivity
Priority	High for the business manager
	Medium for the software developers
QRs elicited	Throughput: Normally, with 16 machines, 2 alarms every 10 min (but with normal monitoring activity)
	Boot Time = 1 min

- The Asset item of the QA scenario template is defined from the refined BGs and the relevant action fields of the derived QAs (Table 8).

The scenarios identified are as follows:

- S11: (start, WPC system) associated with the Boot Time QA
- S21: (transmit data, WPC system) associated with the Throughput QA
- S22: (control data, WPC system) associated with the Throughput QA
- S23: (receive alarms, WPC devices) associated with the Throughput QA
- S31: (distribute work, WPC devices) associated with the Workload Distribution QA

The Boot Time and Throughput attributes are selected by the stakeholders (the business manager and software developers), based on their priority and on the specifications of the WPC system project. Figure 9 shows the tailored QM for the Productivity BG and the Efficiency QAs: Boot Time and Throughput.

Figure 10 depicts the part of the utility tree that shows the productivity business goal BG1, with its derived Efficiency QA. It could be said, for example, that the productivity of the WPC product (reading data from the machines in the plant, receiving alarms, and controlling

aspects of the machines) is of prime importance in terms of the Efficiency QA in this example.

5.2.3 Documentation of the efficiency QRs

Once the efficiency QM has been tailored, the next step is to elicit the Efficiency QRs and document them. Figure 11 shows the elicitation of the Boot Time and Throughput QRs using the checklist. Efficiency could contribute to achieving the Productivity BG by running the application (on handheld devices communicating through a wireless network) with the selected QR values (Boot Time = 1 - min, and Throughput = 2 alarms every 10 min) for each wireless and system component element.

The elicited QR Boot Time and Throughput are documented next according to the QA documentation template (Table 10).

6 Summary and further work

Quality requirements (QRs) drive software quality and are themselves driven by business goals (BGs). This makes QRs crucial to the success of the business. In this paper, we have presented ASPIRE^{B}, an extension to the ASPIRE method, for identifying QRs from BGs and for tracing QRs back to the BGs that drove them. This method addresses an important shortcoming in software engineering, which is how to identify the QRs that are important to the business. Previously, authors such as Bass and Yu [46, 60] had investigated the relationships between BGs and QAs from the architectural perspective, that is, how these goals are achieved during the architectural design process, and the effects of changes to BGs on a system's architecture, while SOQUAREM [22] had proposed to derive QRs from BGs and link them to functional requirements.

Building from these two proposals of Bass and SOQUAREM, our ASPIRE^{B} method extends the original ASPIRE method, creating a structured way to identify BGs and to derive QRs from them.

More specifically, we have reused six existing core components of the SOQUAREM methodology [22] for the QR engineering process. For example, the SOQUAREM components range from top-level business motivation modeling (the BMM) to, for instance, building the QA utility tree and the QA documentation template.

To provide end-to-end traceability and to make the transition from BGs to QRs, we used the business and scenario components, as well as new activities and concepts. In addition, we introduced the concept of business-level information (the BMM), as well as quality model information (ISO 9126), derivation rules, a QA scenario template, and a QA documentation template to derive a QA utility tree that shows traceability among BGs, QAs, and scenarios, and to specify the QA data used during the tailoring and elicitation stages.

For example, at the tailoring stage, the BMM and the QA scenario template have been integrated, in order to ensure that the right QAs are identified and can be traced back to their related BGs (the QA utility tree). The QA scenario template supports the tailoring process to resolve terminological differences arising among the stakeholders with respect to QA priorities.

At the elicitation stage, the QA documentation template is used to specify the elicited QAs with all the data relevant to engineering them and is aimed at enhancing the communication among the organization's stakeholders (business manager and software developers).

The applicability of ASPIRE^{B} has been illustrated with an example based on a case study involving a wireless plant control system, which demonstrates the way in which the added components make it possible to derive QAs and link them to BGs. In this specific example, the proposed ASPIRE^{B} approach makes it possible to:

1. Derive QAs from BGs (using the derivation rules, the BMM, and ISO 9126);
2. Trace the Efficiency QAs back to their respective productivity BGs (using the BMM and the QA utility tree concepts);
3. Select the highest priority QAs for the stakeholders (Boot Time and Throughput were selected by applying the Efficiency scenario template—Table 9); and
4. Establish communication between business managers and developers (using the Efficiency documentation template—Table 10).

Larger-scale experimentation is still needed, such as evaluating ASPIRE^{B} in other application scenarios and validating the ASPIRE^{B} method in industrial contexts. However, the illustrative example presented does provide some insight into the practical implications of this method. For example, we can say that the use of its various concepts will offer industry a flexible model to enable a better understanding of the process of managing software QRs and of how to deal with these concepts appropriately. Application of the business concept (the BMM) will uncover the business motivations with clarity, the QA scenario template will explain the purpose of each QA, and the QA utility tree will provide an easy way to trace the QRs back to their original BGs.

In summary, the ASPIRE^{B} method provides a structured way to derive QAs from BGs and gives managers the ability to align the BGs with the QAs and to trace them back to those BGs. Finally, the BMM, ISO 9126, and the proposed QA utility tree will provide an interesting way to trace BGs, QRs, and QAs and help transition BGs to QRs.

To facilitate implementation of ASPIRE^{B}, organizations will also need a QM management system to tackle the ambiguities in the classification of the QAs and to improve the experience-based artifacts (QMs, checklists, and templates) and IT tools in order to enhance communication between stakeholders, developers, and business managers.

Further work will also be needed to address the challenges of measurability and quantification of QRs. One option is to introduce the new ISO 25030 quality standard [25] and link its measures to the derived QAs. Additional refinements could be developed for ASPIRE^{B} through a study of the interaction of QRs and resolution of any conflicts arising among them. Furthermore, the traceable links between QAs and BGs could be explored through an investigation of the research of Kazman and Bass [53] that provides a categorization of BGs that allows stakeholders to obtain guidance in their elicitation, expression, and documentation of requirements.

References

1. Callele D, Penzenstadler B, Wnuk K (2013) Risk identification at the interface between business case and requirements. REFSQ'13 19th international conference on requirements engineering: foundation for software quality. Springer-Verlag Berlin, Heidelberg, pp. 253–268
2. Zowghi D, Coulin C (2005) Requirements elicitation: a survey of techniques, approaches, and tools. Engineering and managing software requirements. Springer Berlin Heidelberg, pp. 19–46. doi: 10.1007/3-540-28244-0_2
3. Al-Rawas A, Easterbrook SM (1996) Communication problems in requirements engineering: a field study. Cognitive science research papers, University of Sussex at Brighton, Brighton
4. Sajjad U, Qaisar HM (2010) Issues and challenges of requirement elicitation in large web projects. Master Thesis Computer Science Thesis no: MCS-2010:05, School of Computing Blekinge Institute of Technology, Ronneby, Sweden
5. Sommerville I (2006) Software engineering, 8th edn. Addison-Wesley Longman Publishing Co., Inc., Boston. ISBN 0321313798
6. Cysneiros LM, Sampaio do Prado Leite LC (2004) Non-functional requirements: from elicitation to conceptual models. IEEE Trans Softw Eng, 30(5):328–350, IEEE Press Piscataway, NJ, USA. doi:10.1109/TSE.2004.10
7. Bredemeyer D, Malan R (2001) Defining Non-functional requirements. Bredemeyer Consulting, White Paper. 8 p. http://www.bredemeyer.com. Accessed 19 Mar 2013
8. Chung L, Sampaio do Prado Leite LC (2009) On non-functional requirements in software engineering. Conceptual modeling: foundations and applications. Springer-Verlag Berlin Heidelberg, pp. 363–379. doi:10.1007/978-3-642-02463-4_19
9. Hill RL, Wang J, Nahrstedt K (2004) Quantifying non-functional requirements: a process oriented approach. 12th IEEE international conference on requirements engineering (RE), September 6-10, Kyoto, Japan, pp. 352–353
10. Wiegers K (1999) Writing QRs. process impact. http://www.processimpact.com/articles/qualreqs.html. Published in Software Development Magazine. Accessed 05 May 2013
11. Poort ER, de With PHN (2004) Resolving requirement conflicts through non-functional decomposition. 4th working IEEE/IFIP conference on software architecture (WICSA), 12–15 June Oslo, Norway, pp 145–154
12. Herrmann A, Paech B (2008) MOQARE: misuse-oriented quality requirements engineering. Requirements Eng J 13(1):73–86. Springer-Verlag New York, Inc. Secaucus, NJ, USA
13. Brito I, Moreira A, Araujo J (2002) A requirements model for quality attributes. Workshop on "early aspects: aspect-oriented requirements engineering and architecture design", 1st international conference on aspect-oriented software development. 22–26 April University of Twente, Enschede, Holland, pp. 1–6
14. Jacobson I, Christerson M, Jonsson P, Övergaard G (1992) Object-oriented software engineering: a use case driven approach. Addison-Wesley, Wokingham, England
15. Trendowicz A, Punter T (2003) Quality modeling for software product lines. 7th ECOOP workshop on quantitative approaches in object-oriented software engineering, QAOOSE, Darmstadt, Germany
16. Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. Kluwer Academic Publishing
17. Doerr J, Kerkow D, Koenig T et al (2005) Non-functional requirements in industry: three case studies adopting the ASPIRE^{TM} NFR method. 13th IEEE international conference on requirements engineering (RE), August 29–September 2, Paris, France, pp 373–384
18. Kerkow D, Kohler K, Doerr J (2003) Usability and other quality aspects derived from use cases. Performance by design. USE - second international conference on usage-centered design, October 18–22, Portsmouth, New Hampshire, pp 135–154
19. Doerr J (2011) Non-functional requirements. lecture: requirements engineering WS 2010/2011. http://www.agse.informatik.uni-kl.de/teaching/re/ws2010/Vorlesung%20RE_WS1011_NFR.pdf. Accessed 23 May 2013
20. Institute Experimentelles Software Engineering. http://www.iese.fraunhofer.de/index.jsp. Accessed 15 Oct 2013
21. Herrmann A, Weiß D (2009) Alignment of software specifications with quality-and business goals in the SIKOSA method. PRIMIUM: Process Innovation for Enterprise Software, Mannheim, pp 27–42
22. Djouab R, Suryn W (2011) SOQUAREM: software quality requirements engineering method. SQM conference on quality management. 18–20 Apr Loughborough University, Leicestershire, UK
23. ISO, IEC 9126–1 (2001) Software engineering: product quality—Part 1: quality model. International Organization for Standardization, Geneva
24. Azuma M (2004) Applying ISO/IEC 9126-1 quality model to QRs engineering on critical software. Sixth international workshop on requirements for high assurance systems (RHAS 2004), Kyoto, Japan, pp 3–10
25. ISO/IEC 25030 (2007) Software engineering—software product quality requirements and evaluation (SQuaRE)—quality requirements. International Organization for Standardization
26. Abran A, Bourque P, Dupuis R, Moore JW, Tripp LL (eds) (2004) Guide to the software engineering body of knowledge-SWEBOK. IEEE-CS Press, Piscataway
27. Campbell J (2007) Lecture 9, Part 2: Non-functional requirements. CSC340 - Winter 2007, University of Toronto. http://www.cdf.toronto.edu/~csc340h/winter/lectures/w10/L9-part2-6up.pdf. Accessed Aug 2014
28. Borg A (2004) Contributions to management and validation of non-functional requirements. Linköpings universitet, PELAB - Laboratoriet för programmeringsomgivningar, Linköpings

universitet, Tekniska högskolan (författare), BokAvhandling Engelska
29. Amyot D, Mussbacher G (2003) URN: towards a new standard for the visual description of requirements. Telecommunications and beyond: the broader applicability of SDL and MSC. Edel Sherratt (Ed.) SAM, Lecture Notes in Computer Science. 25(99):21–37
30. Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the *Tropos* project. Inf Syst 27(6):365–389
31. Dutoit AH, Paech B (2002) Rationale-based use case specification. Requir Eng 7(1):3–19
32. Hill R, Wan J, Nahrstedt K (2004) Quantifying non-functional requirements: a process oriented approach. 12th IEEE international conference on requirements engineering. 6–11 Sept pp. 352–353
33. Rilston F, Paim S, Castro JFB (2002) Enhancing data warehouse design with the NFR framework. 5th Workshop on requirements engineering (WER2002), Valencia Spain, pp 11–12
34. Supakkul S, Chung L (2012) The RE-tools: a multi-notational requirements modeling toolkit. 20th IEEE International conference on requirements engineering. 24–28 Sept pp 333–334
35. Yrjönen A, Merilinna J (2009) Extending the NFR framework with measurable non- functional requirements. 2nd international workshop on non-functional system properties in domain specific modeling languages. Denver, Colorado, USA
36. Nasir MM, Arif M, Samina K, Tehmina K, Faisal MM (2009) Measurable and scalable NFRs using fuzzy logic and likert scale. International journal of computer science and information security (IJCSIS), *2*(1). http://dblp.uni-trier.de/db/journals/corr/corr0906.html#abs-0906-5393. Accessed Aug 2014
37. Dewi M, Didar Z, Nurie N (2009) Managing conflicts among non-functional requirements. 12th Australian workshop on requirements engineering (AWRE '09), Sydney Australia
38. Davis AM (2003) The art of requirements triage. Comput J 363:42–49. IEEE Computer Society Press Los Alamitos, CA, USA. doi:10.1109/MC.2003.1185216
39. Boehm HI, Barry W, Rodgers TL, Deutsch M (2001) Applying win–win to quality requirements: a case study. 23rd International Conference on Software Engineering Toronto, Ontario, Canada IEEE Computer Society, pp 555–564
40. Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York
41. Beck K (1999) Extreme programming explained. Addison-Wesley Longman Publishing Co., Inc., Boston
42. Egyed A, Grunbacher P (2004) Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help. IEEE Softw 21(6):50–58
43. Boehm HI, Barry W (1999) Cost versus quality requirements: conflict analysis and negotiation aids. Softw Q Prof 1(2):38–50
44. Glinz M (2007) On non-functional requirements. 15th IEEE international requirements engineering conference (RE). 15–19 Oct New Delhi, India, pp 21–26
45. Mylopoulos J (2004) Non-functional requirements, information systems analysis and design. http://www.cs.toronto.edu/~jm/340S/Slides2/NFR.pdf. Accessed Jan 2013
46. Gross D, Yu E (2001) Evolving system architecture to meet changing business goals: an agent and goal-oriented approach. 5th IEEE international symposium on requirements engineering (RE), 27–31 Aug Toronto, Canada, pp 316–317. doi:10.1109/ISRE.2001.948602
47. Punter T, Van Solingen R, Trienekens JM (1997) Software product evaluation: current status and future needs for customers and industry. 4th IT evaluation (EVIT-97), Delft, the Netherlands, pp 1–11
48. Carvallo JP, Franch X (2003) Using QMs in software package selection. IEEE Softw 20(1):34–41
49. Van Solingen R, Kusters RJ, Trienekens JM (1999) Identifying embedded software quality: two approaches. Q Reliab Eng Int 15(6):485–492. doi:10.1002/(SICI)1099-1638(199911/12)15:6<485:AID-QRE295>3.0.CO;2-3
50. Djouab R (2012) Software product quality requirements engineering method: SOQUAREM. Ph.D dissertation, École of Technologie Supérieure—Université du Québec, Montréal, QC, Canada
51. Djouab R, Suryn W (2007) Applicability analysis of two QRs treatment methods: ASPIRE and FDAF. International conference on software and systems engineering and their applications (ICSSEA), 4–6 Dec Conservatoire National des Arts et Métiers, Paris, France
52. Fagnani S (2014) What are business goals? eHow Contributor. http://www.ehow.com/facts_5758424_business-goals_.html#ixzz2zjhOx0Rc. Accessed April 2014
53. Kazman R, Bass L (2005) Categorizing business goals for software architectures. Technical report CMU/SEI- 2005-TR-021, Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/library/abstracts/reports/05tr021.cfm. Accessed May 2014
54. Weiß D, Leukel J, Kirn S (2008) A method for aligning business process modeling and software requirements engineering. PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI), (February 26–28, Garching, Germany). CEUR-WS.org CEUR Workshop Proceedings
55. Basili VR, Heidrich J et al (2013) Linking software development and business strategy through measurement. IEEE Comput 43(4):57–65. doi:10.1109/MC.2010.108
56. Bleistein SJ, Cox K, Verner J (2006) Validating strategic alignment of organizational IT requirements using goal modeling and problem diagrams. J Syst Softw 79(3):362–378
57. Bleistein SJ, Aurum A, Cox K, Ray PK (2004) Strategy-oriented alignment in requirements engineering: linking business strategy to requirements of e-business systems using the SOARE approach. J Res Pract Inf Technol 36(4):259–276
58. Morris PWG, Jamieson A (2005) Moving from corporate strategy to project strategy. Project Manag J. 36(4):5–18. ISSN 87569728
59. Clements P, Bass L (2010) Relating business goals to architecturally significant requirements for software systems. Technical note CMU/SEI-2010-TN-018 Research, Technology, and System Solutions Program
60. Business Rules Group (BRG) (2007) Business motivation model (Version 1.3). http://www.businessrulesGroup.org. Accessed April 2013
61. Zubrow D (2004) Software quality requirements and evaluation, the ISO 25000 series. PSM technical working group. Carnegie Mellon University. Pittsburgh, PA 15213-3890, 35 p. http://www.psmsc.com/Downloads/TWGFeb04/04ZubrowISO25000SWQualityMeasurement.pdf. Accessed Aug 2014
62. Kazman R, Klein M, Clements P (2000) ATAM: method for architecture evaluation. Technical report CMU/SEI-2000-TR-004. Software Engineering Institution, Carnegie Mellon University, Pittsburgh, PA, USA, 83 p