0.1 Software requirements defined

0.1.1 Best definition of "requirement"

Many decades after the invention of computer programming, software practitioners still have raging debates about exactly what a "requirement" is. Rather than prolong those debates, we simply present some definitions that we have found useful.

Our favorite definition comes from Ian Sommerville and Pete Sawyer [3]:

Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.

This definition acknowledges the diverse types of information that collectively are referred to as "the requirements." Requirements encompass both the user's view of the external system behavior and the developer's view of some internal characteristics. They include both the behavior of the system under specific conditions and those properties that make the system suitable-and maybe even enjoyable-for use by its intended operators.

0.1.1.1 The pure dictionary "requirement"

Software requirements include a time dimension. They could be present tense, describing the current system's capabilities. Or they could be for the near-term (high priority), mid-term (medium priority), or hypothetical (low priority) future. They could even be past tense, referring to needs that were once specified and then discarded. Don't waste time debating whether or not something is a requirement, even if you know you might never implement it for some good business reason. It is.

0.1.2 Levels and types of requirements

Table 1: Some types of requirements information.

Term	Definition
Business requirement	A high-level business objective of the organization that builds a product or of a customer who procures it.
Business rule	A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements.
Constraint	A restriction that is imposed on the choices available to the developer for the design and construction of a product.
External interface requirement	A description of a connection between a software system and a user, another software system, or a hardware device.
Feature	One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.
Functional requirement	A description of a behavior that a system will exhibit under specific conditions.
Nonfunctional requirement	A description of a property or characteristic that a system must exhibit or a constraint that it must respect.
Quality attribute	A kind of nonfunctional requirement that describes a service or performance characteristic of a product.
System requirement	A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.
User requirement	A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute.

Software requirements include three distinct levels:

- **business requirements,**
- **user requirements,**
- **and functional requirements.**

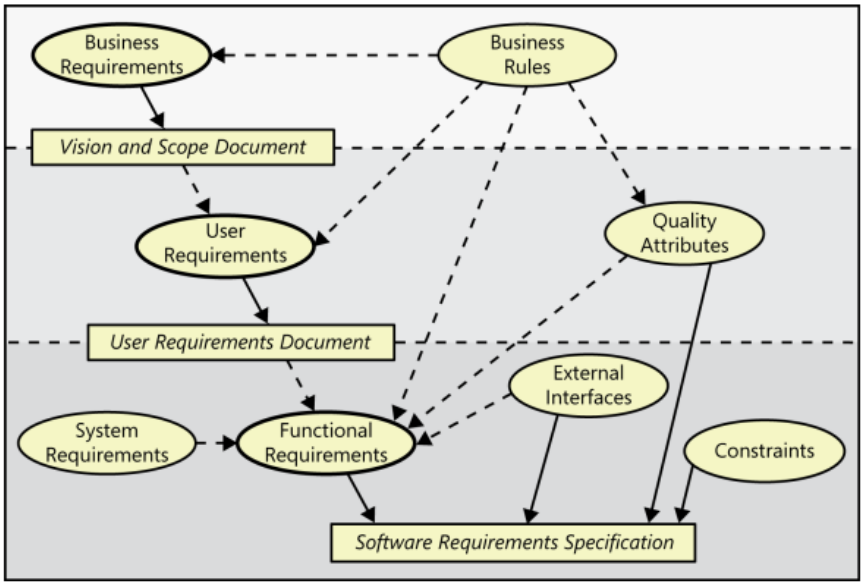
In addition, every system has an assortment of nonfunctional requirements.

The model in Figure 1 illustrates a way to think about these diverse types of requirements. As statistician George E. P. Box famously said, "Essentially, all models are wrong, but some are useful" [4]. That's certainly true of Figure 1. This model is not all-inclusive, but it does provide a helpful scheme for organizing the requirements knowledge you'll encounter.

The ovals in Figure 1 represent types of requirements information, and the rectangles indicate documents in which to store that information. The solid arrows indicate that a certain type of information typically is stored in the indicated

document. (Business rules and system requirements are stored separately from software requirements, such as in a business rules catalog or a system requirements specification, respectively.) The dotted arrows indicate that one type of information is the origin of or influences another type of requirement. Data requirements are not shown explicitly in this diagram. Functions manipulate data, so data requirements can appear throughout the three levels.

Figure 1: Relationships among several types of requirements information. Solid arrows mean "are stored in"; dotted arrows mean "are the origin of" or "influence.".



0.1.2.1 Business requirements

Business requirements describe why the organization is implementing the system—the business benefits the organization hopes to achieve. The focus is on the business objectives of the organization or the customer who requests the system.

Suppose an airline wants to reduce airport counter staff costs by 25 percent. This goal might lead to the idea of building a kiosk that passengers can use to check in for their flights at the airport.

Business requirements typically come from the funding sponsor for a project, the acquiring customer, the manager of the actual users, the marketing department, or a product visionary.

We like to record the business requirements in a vision and scope document. Other strategic guiding documents sometimes used for this purpose include a project charter, business case, and market (or marketing) requirements document.

0.1.2.2 User requirements

User requirements describe goals or tasks the users must be able to perform with the product that will provide value to someone. The domain of user requirements also includes descriptions of product attributes or characteristics that are important to user satisfaction.

Ways to represent user requirements include use cases [6], user stories [7], and event-response tables. Ideally, actual user representatives will provide this information. User requirements describe what the user will be able to do with the system.

An example of a use case is *"Check in for a flight" using an airline's website or a kiosk at the airport.* Written as a user story, the same user requirement might read: *"As a passenger, I want to check in for a flight so I can board my airplane."*

It's important to remember that most projects have multiple user classes, as well as other stakeholders whose needs also must be elicited. Chapter **??**, "Understanding user requirements," addresses this level of the model. Some people use the broader term "stakeholder requirements," to acknowledge the reality that various stakeholders other than direct users will provide requirements. That is certainly true, but we focus the attention at this level on understanding what actual users need to achieve with the help of the product.

0.1.2.3 Functional requirements

Functional requirements specify the behaviors the product will exhibit under specific conditions. **They describe what the developers must implement to enable users to accomplish their tasks (user requirements), thereby satisfying the business requirements.** This alignment among the three levels of requirements is essential for project success.

Functional requirements often are written in the form of the traditional "shall" statements:

"The Passenger shall be able to print boarding passes for all flight segments for which he has checked in" or *"If the Passenger's profile does not indicate a seating preference, the reservation system shall assign a seat."*

The business analyst (BA) documents functional requirements in a software requirements specification (SRS), which describes as fully as necessary the ex-

pected behavior of the software system.

The SRS is used in development, testing, quality assurance, project management, and related project functions. People call this deliverable by many different names, including business requirements document, functional spec, requirements document, and others. An SRS could be a report generated from information stored in a requirements management tool. Because it is an industry-standard term, we will use "SRS" consistently throughout this document (ISO/IEC/IEEE 2011)[8]. See Chapter **??**, "Documenting the requirements," for more information about the SRS.

0.1.2.4 System requirements

System requirements describe the requirements for a product that is composed of multiple components or subsystems (ISO/IEC/IEEE 2011)[8]. A "system" in this sense is not just any information system. A system can be all software or it can include both software and hardware subsystems.

People and processes are part of a system, too, so certain system functions might be allocated to human beings. Some people use the term "system requirements" to mean the detailed requirements for a software system, but that's not how we use the term in this document.

A good example of a "system" is the cashier's workstation in a supermarket. There's a bar code scanner integrated with a scale, as well as a hand-held bar code scanner. The cashier has a keyboard, a display, and a cash drawer. You'll see a card reader and PIN pad for your loyalty card and credit or debit card, and perhaps a change dispenser. You might see up to three printers for your purchase receipt, credit card receipt, and coupons you don't care about. These hardware devices are all interacting under software control.

The requirements for the system or product as a whole, then, lead the business analyst to derive specific functionality that must be allocated to one or another of those component subsystems, as well as demanding an understanding of the interfaces between them.

0.1.2.5 Business rules

Business rules include corporate policies, government regulations, industry standards, and computational algorithms. Business rules are not themselves software requirements because they have an existence beyond the boundaries of any specific software application. However, they often dictate that the system must contain functionality to comply with the pertinent rules. Sometimes, as with corporate security policies, business rules are the origin of specific quality attributes that are then implemented in functionality. Therefore, you can trace the genesis of certain functional requirements back to a particular business rule.

0.1.2.6 Nonfunctional requirements

In addition to functional requirements, the SRS contains an assortment of nonfunctional requirements. Quality attributes are also known as quality factors, quality of service requirements, constraints, and the "-ilities."
They describe the product's characteristics in various dimensions that are important either to users or to developers and maintainers, such as performance, safety, availability, and portability. Other classes of nonfunctional requirements describe external interfaces between the system and the outside world. These include connections to other software systems, hardware components, and users, as well as communication interfaces. Design and implementation constraints impose restrictions on the options available to the developer during construction of the product.

If they're nonfunctional, then what are they?

For many years, the requirements for a software product have been classified broadly as either functional or nonfunctional. The functional requirements are evident: they describe the observable behavior of the system under various conditions. However, many people dislike the term "nonfunctional." That adjective says what the requirements are not, but it doesn't say what they are. We are sympathetic to the problem, but we lack a perfect solution.

Other-than-functional requirements might specify not what the system does, but rather how well it does those things. They could describe important characteristics or properties of the system. These include the system's availability, usability, security, performance, and many other characteristics.

Some people consider nonfunctional requirements to be synonymous with quality attributes, but that is overly restrictive. For example, design and implementation constraints are also nonfunctional requirements, as are external interface requirements.

Still other nonfunctional requirements address the environment in which the system operates, such as platform, portability, compatibility, and constraints. Many products are also affected by compliance, regulatory, and certification requirements. There could be localization requirements for products that must take into account the cultures, languages, laws, currencies, terminology, spelling, and other characteristics of users. Though such requirements are specified in nonfunctional terms, the business analyst typically will derive numerous bits of functionality to ensure that the system possesses all the desired behaviors and properties.

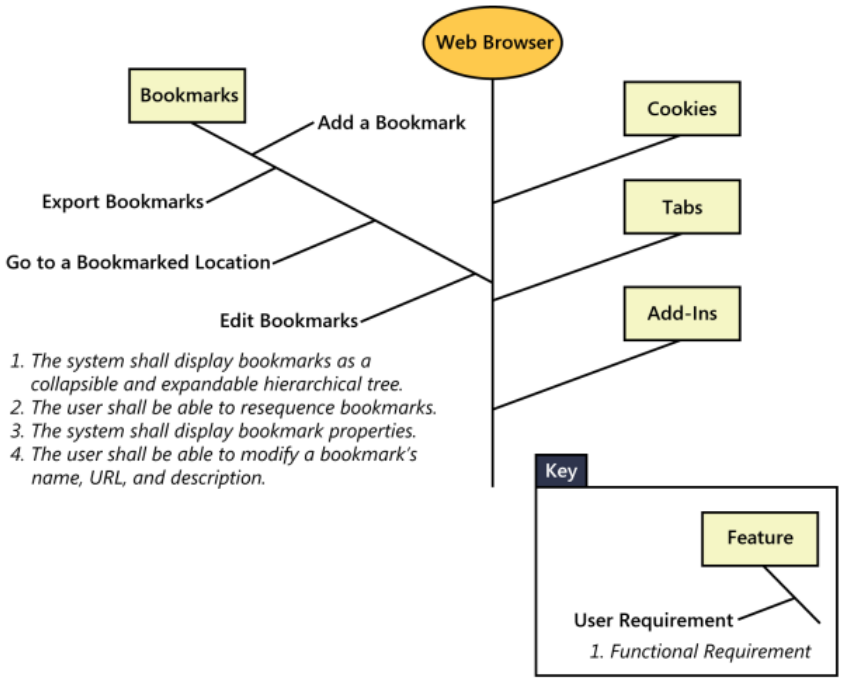
In this document, we are sticking with the term "nonfunctional requirements," despite its limitations, for the lack of a suitably inclusive alternative. Rather than worry about precisely what you call these sorts of information, just make sure that they are part of your requirements elicitation and analysis activities. You can deliver a product that has all the desired functionality but that users hate because it doesn't match their (often unstated) quality expectations.

0.1.2.7 Features

A feature consists of one or more logically related system capabilities that provide value to a user and are described by a set of functional requirements. A customer's list of desired product features is not equivalent to a description of the user's task-related needs. Web browser bookmarks, spelling checkers, the ability to define a custom workout program for a piece of exercise equipment, and automatic virus signature updating in an anti-malware product are examples of features.
A feature can encompass multiple user requirements, each of which implies that

certain functional requirements must be implemented to allow the user to perform the task described by each user requirement. Figure 2 illustrates a feature tree, an analysis model that shows how a feature can be hierarchically decomposed into a set of smaller features, which relate to specific user requirements and lead to specifying sets of functional requirements [16].

Figure 2: Relationships among features, user requirements, and functional requirements.

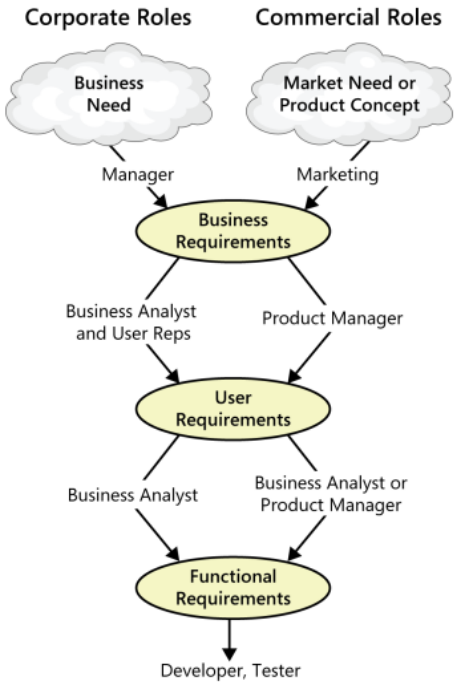


To illustrate some of these various kinds of requirements, consider a project to develop the next version of a text editor program. A business requirement might be "Increase non-US sales by 25 percent within 6 months." Marketing realizes that the competitive products only have English-language spelling checkers, so they decide that the new version will include a multilanguage spelling checker feature. Corresponding user requirements might include tasks such as "Select language for spelling checker," "Find spelling errors," and "Add a word to a dictionary." The spelling checker has many individual functional requirements, which deal with operations such as highlighting misspelled words, autocorrect, displaying suggested replacements, and globally replacing misspelled words with corrected words. Usability requirements specify how the software is to be localized for use with specific languages and character sets.

0.1.3 Working with the three levels

Figure 3 illustrates how various stakeholders might participate in eliciting the three levels of requirements. Different organizations use a variety of names for the roles involved in these activities; think about who performs these activities in your organization. The role names often differ depending on whether the developing organization is an internal corporate entity or a company building software for commercial use.

Figure 3: An example of how different stakeholders participate in requirements development.

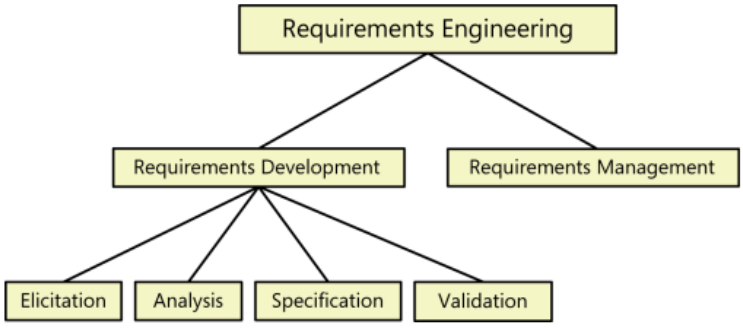


0.2 Requirements development and management

Confusion about requirements terminology extends even to what to call the whole discipline. Some authors call the entire domain requirements engineering (our preference). Others refer to it all as requirements management. Still others refer to these activities as a subset of the broad domain of business analysis.
We find it useful to split requirements engineering into requirements development (addressed in Part II) and requirements management, as shown in Figure 4.

Regardless of what development life cycle your project is following—be it pure waterfall, phased, iterative, incremental, agile, or some hybrid—these are the things you need to do regarding requirements. Depending on the life cycle, you will perform these activities at different times in the project and to varying degrees of depth or detail.

Figure 4: Subdisciplines of software requirements engineering.



0.2.1 Requirements development

As Figure 4 shows, we subdivide requirements development into elicitation, analysis, specification, and validation [9]. These subdisciplines encompass all the activities involved with exploring, evaluating, documenting, and confirming the requirements for a product. Following are the essential actions in each subdiscipline.

0.2.1.1 Elicitation

Elicitation encompasses all of the activities involved with discovering requirements, such as interviews, workshops, document analysis, prototyping, and others. The key actions are:

- Identifying the product's expected user classes and other stakeholders.
- Understanding user tasks and goals and the business objectives with which those tasks align.
- Learning about the environment in which the new product will be used.
- Working with individuals who represent each user class to understand their functionality needs and their quality expectations.

Usage-centric or product-centric?

Requirements elicitation typically takes either a usage-centric or a product-centric approach, although other strategies also are possible. The usage-centric strategy emphasizes understanding and exploring user goals to derive the necessary system functionality. The product-centric approach focuses on defining features that you expect will lead to marketplace or business success. A risk with product-centric strategies is that you might implement features that don't get used much, even if they seemed like a good idea at the time. We recommend understanding business objectives and user goals first, then using that insight to determine the appropriate product features and characteristics.

0.2.1.2 Analysis

Analyzing requirements involves reaching a richer and more precise understanding of each requirement and representing sets of requirements in multiple ways. Following are the principal activities:

- Analyzing the information received from users to distinguish their task goals from functional requirements, quality expectations, business rules, suggested solutions, and other information
- Decomposing high-level requirements into an appropriate level of detail
- Deriving functional requirements from other requirements information
- Understanding the relative importance of quality attributes
- Allocating requirements to software components defined in the system architecture
- Negotiating implementation priorities
- Identifying gaps in requirements or unnecessary requirements as they relate to the defined scope

0.2.1.3 Specification

Requirements specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion. The principal activity is:

- Translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences.

0.2.1.4 Validation

Requirements validation confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives. The central activities are:

- Reviewing the documented requirements to correct any problems before the development group accepts them.
- Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives.

Iteration is a key to requirements development success. Plan for multiple cycles of exploring requirements, progressively refining high-level requirements into more precision and detail, and confirming correctness with users. This takes time and it can be frustrating. Nonetheless, it's an intrinsic aspect of dealing with the fuzzy uncertainty of defining a new software system.

Bibliography

[1] Bass, Len, Paul Clements, and Rick Kazman. 2013. Software Architecture in Practice, 3nd ed. Reading, MA: Addison-Wesley.

[2] Davis, Alan M. 1995. 201 Principles of Software Development. New York: McGraw-Hill.

[3] Sommerville, Ian, and Pete Sawyer. 1997. Requirements Engineering: A Good Practice Guide. Chichester, England: John Wiley & Sons Ltd.

[4] Box, George E. P., and Norman R. Draper. 1987. Empirical Model-Building and Response Surfaces. New York: John Wiley & Sons, Inc.

[5] Brown, Norm. 1996. "Industrial-Strength Management Strategies." IEEE Software 13(4):94-103.

[6] Kulak, Daryl, and Eamonn Guiney. 2004. Use Cases: Requirements in Context, 2nd ed. Boston: Addison-Wesley.

[7] Cohn, Mike. 2004. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley.

[8] ISO/IEC/IEEE. 2011. "ISO/IEC/IEEE 29148:2011(E), Systems and software engineering—Life cycle processes—Requirements engineering." Geneva, Switzerland: International Organization for Standardization.

[9] Abran, Alain, James W. Moore, Pierre Bourque, and Robert Dupuis, eds. 2004. Guide to the Software Engineering Body of Knowledge, 2004 Version. Los Alamitos, CA: IEEE Computer Society Press.

[10] Beatty, Joy, and Anthony Chen. 2012. Visual Models for Software Requirements. Redmond, WA: Microsoft Press.

[11] Wiegers, Karl E. 2006. More About Software Requirements: Thorny Issues and Practical Advice. Redmond, WA: Microsoft Press.

[12] Wiegers, Karl E. 2007. Practical Project Initiation: A Handbook with Tools. Redmond, WA: Microsoft Press.

[13] Moore, Geoffrey A. 2002. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. New York: HarperBusiness.

[14] Robertson, James, and Suzanne Robertson. 1994. Complete Systems Analysis: The Workbook, the Textbook, the Answers. New York: Dorset House Publishing.

[15] Robertson, Suzanne, and James Robertson. 2013. Mastering the Requirements Process: Getting Requirements Right, 3rd ed. Upper Saddle River, NJ: Addison-Wesley.

[16] Beatty, Joy, and Anthony Chen. 2012. Visual Models for Software Requirements. Redmond, WA: Microsoft Press.

[17] Nejmeh, Brian A., and Ian Thomas. 2002. "Business-Driven Product Planning Using Feature Vectors and Increments." IEEE Software 19(6):34-42.

[18] Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. The Unified Modeling Language User Guide. Reading, MA: Addison-Wesley.

[19] Podeswa, Howard. 2009. The Business AnalystŚs Handbook. Boston: Course Technology.

[20] Armour, Frank, and Granville Miller. 2001. Advanced Use Case Modeling: Software Systems. Boston: Addison-Wesley.

[21] Business Rules Group. 2012. http://www.businessrulesgroup.org.

[22] von Halle, Barbara. 2002. Business Rules Applied: Building Better Systems Using the Business Rules Approach. New York: John Wiley & Sons, Inc.

[23] Ross, Ronald G. 1997. The Business Rule Book: Classifying, Defining, and Modeling Rules, Version 4.0, 2nd ed. Houston: Business Rule Solutions, LLC.

[24] Ross, Ronald G., and Gladys S. W. Lam. 2011. Building Business Solutions: Business Analysis with Business Rules. Houston: Business Rule Solutions, LLC.

[25] Ross, Ronald G. 2001. "The Business Rules Classification Scheme." DataToKnowledge Newsletter 29(5).

[26] Morgan, Tony. 2002. Business Rules and Information Systems: Aligning IT with Business Goals. Boston: Addison-Wesley.

[27] von Halle, Barbara, and Larry Goldberg. 2010. The Decision Model: A Business Logic Framework Linking Business and Technology. Boca Raton, FL: Auerbach Publications.

[28] Boyer, Jérôme, and Hafedh Mili. 2011. Agile Business Rule Development: Process, Architecture, and JRules Examples. Heidelberg, Germany: Springer.

[29] Gilb, Tom. 1988. Principles of Software Engineering Management. Harlow, England: Addison-Wesley.

[30] Ambler, Scott. 2005. The Elements of UML 2.0 Style. New York: Cambridge University Press.

[31] Withall, Stephen. 2007. Software Requirement Patterns. Redmond, WA: Microsoft Press.