CHAPTER 14
Beyond functionality

Objectives

- After finish this chapter, student could:
  - detect and specify nonfunctional requirements
  - Understand the role of quality requirements in software development

Contents

Software quality attributes

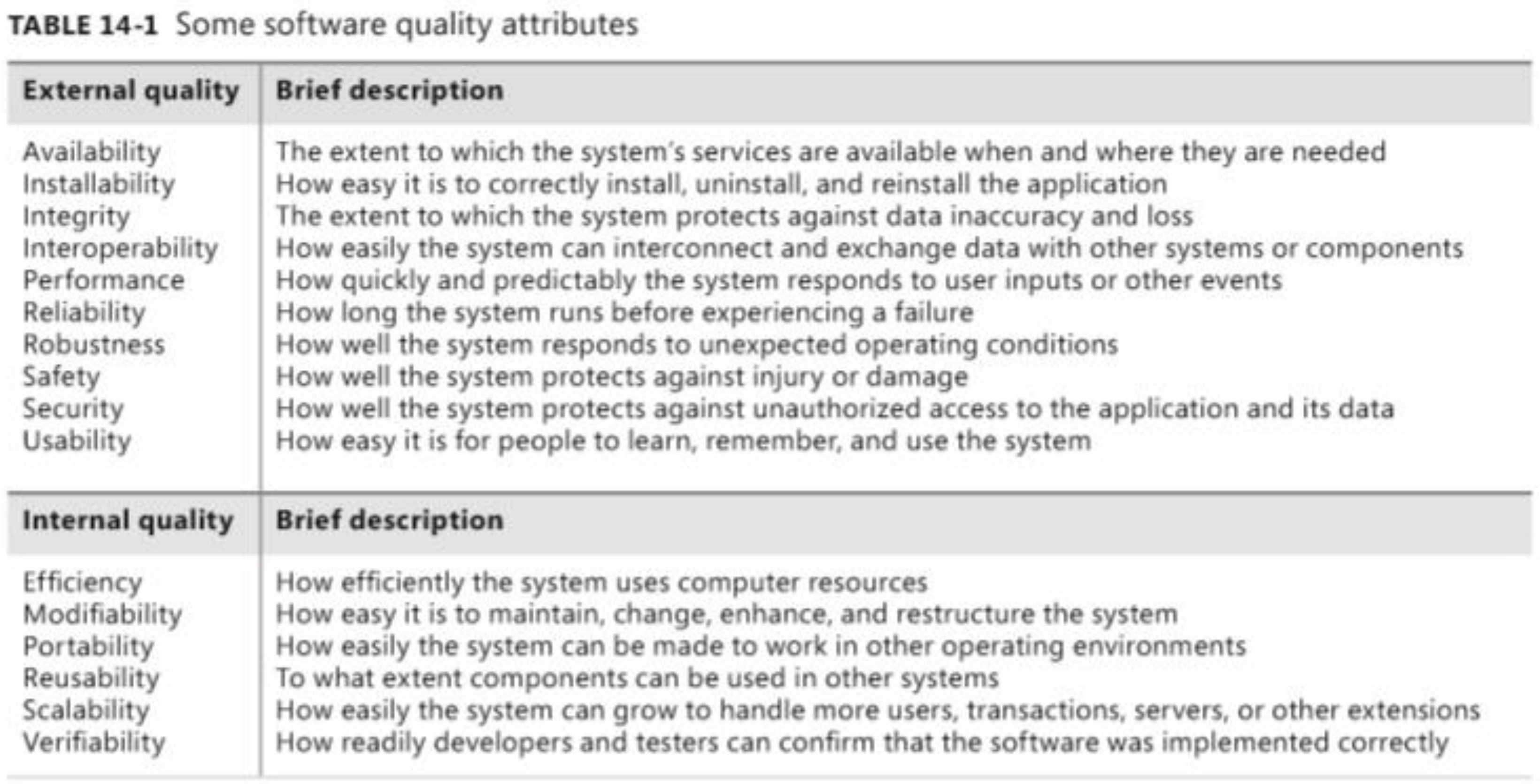


TABLE 14-1 Some software quality attributes

External quality	Brief description
Availability	The extent to which the system's services are available when and where they are needed
Installability	How easy it is to correctly install, uninstall, and reinstall the application
Integrity	The extent to which the system protects against data inaccuracy and loss
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system

Internal quality	Brief description
Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent components can be used in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, or other extensions
Verifiability	How readily developers and testers can confirm that the software was implemented correctly

Exploring quality attributes

- Step 1: Start with a broad taxonomy
- Step 2: Reduce the list
- Step 3: Prioritize the attributes
- Step 4: Elicit specific expectations for each attribute
- Step 5: Specify well-structured quality requirements

Defining quality requirements

- External quality attributes
  – Availability, Installability, Integrity, Interoperability, Performance, Reliability, Robustness, Safety, Security, Usability,
- Internal quality attributes
  – Efficiency, Modifiability, Portability, Reusability, Scalability, Verifiability

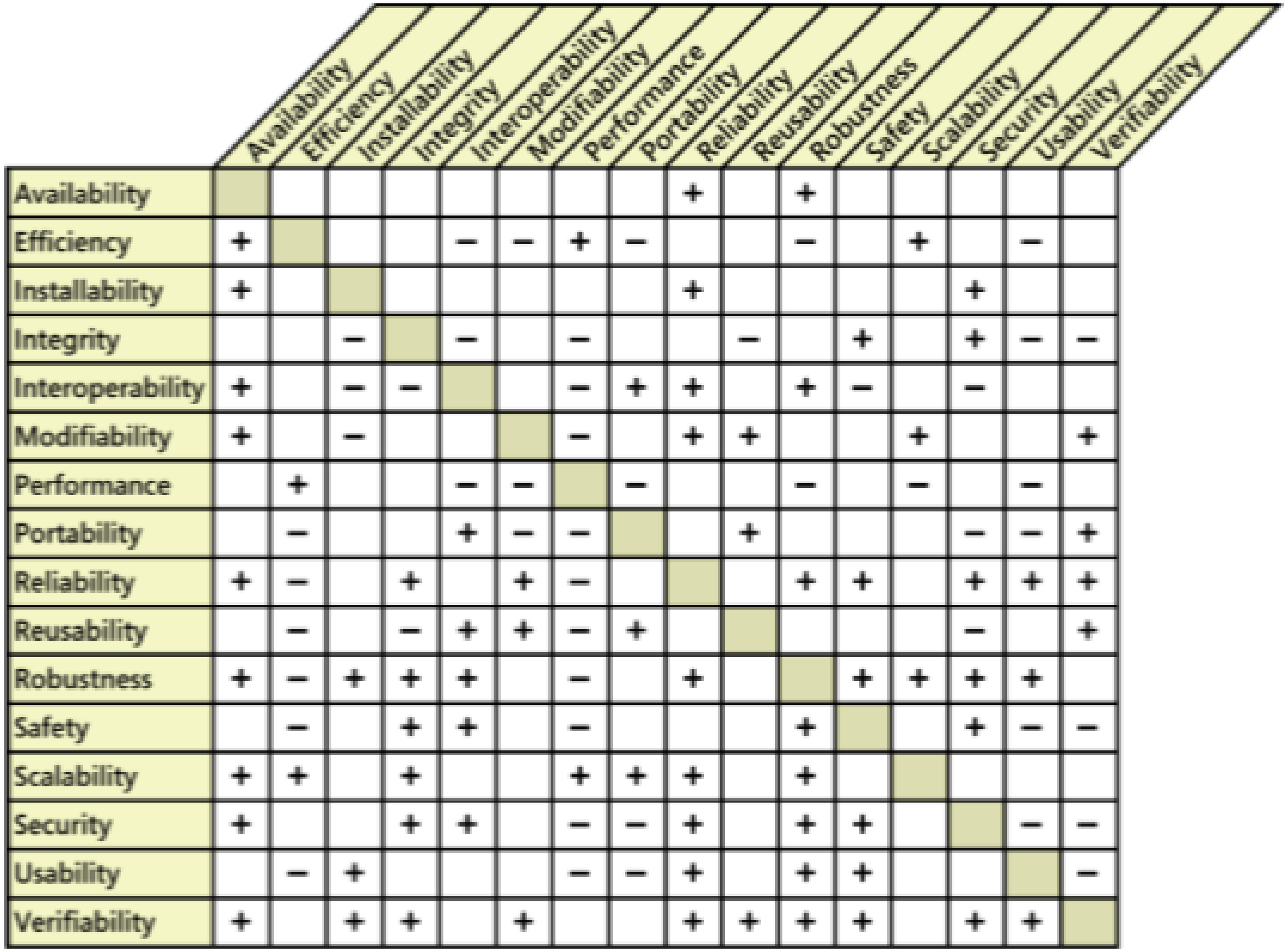
Specifying quality requirements with Planguage

- Definition: Planguage, a language with a rich set of keywords that permits precise statements of quality attributes and other project goals (Simmons 2001).
- Purpose: address the problem of ambiguous and incomplete nonfunctional requirements
- Example: “At least 95 percent of the time, the system shall take no more than 8 seconds to display any of the predefined accounting reports.”

Specifying quality requirements with Planguage

- TAG Performance.Report.ResponseTime
- AMBITION Fast response time to generate accounting reports on the base user platform.
- SCALE Seconds of elapsed time between pressing the Enter key or clicking OK to request a report and the beginning of the display of the report.
- METER Stopwatch testing performed on 30 test reports that represent a defined usage operational profile for a field office accountant.
- GOAL No more than 8 seconds for 95 percent of reports. -<-Field Office Manager
- STRETCH No more than 2 seconds for predefined reports, 5 seconds for all reports.
- WISH No more than 1.5 seconds for all reports.
- Base user platform DEFINED Quad-core processor, 8GB RAM, Windows 8, QueryGen 3.3 running, single user, at least 50 percent of system RAM and 70 percent of system CPU capacity free, network connection speed of at least 30 Mbps.

Quality attribute trade-offs

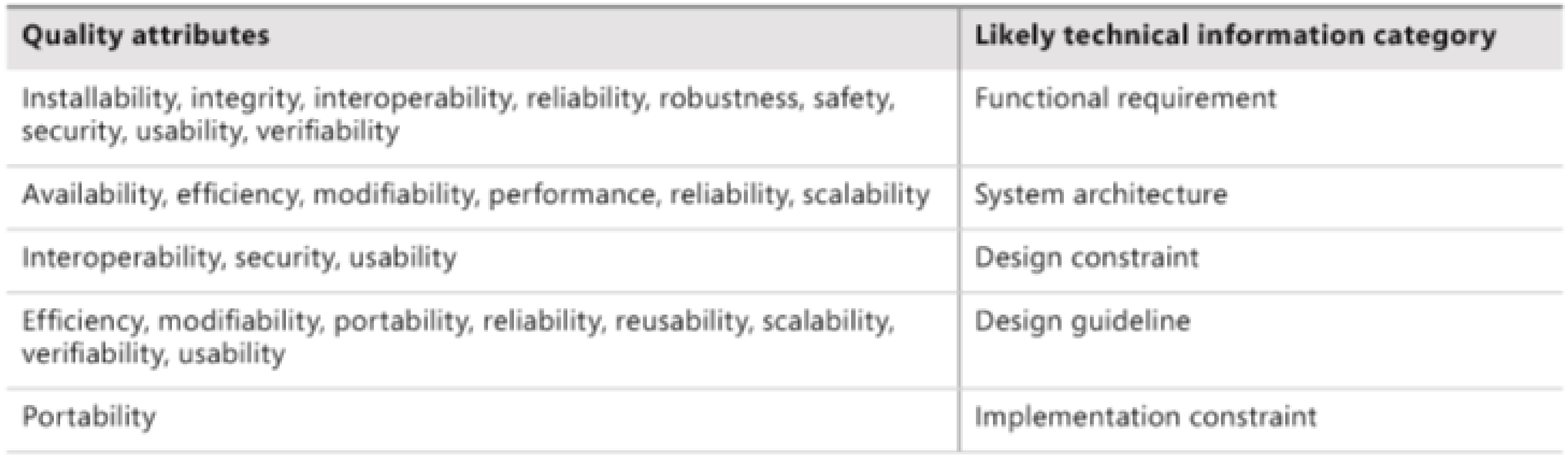


	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability									+		+					
Efficiency	+				−	−	+	−			−		+		−	
Installability	+								+					+		
Integrity			−		−		−			−		+		+	−	−
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−				+			+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

FIGURE 14-2 Positive and negative relationships among selected quality attributes.

Implementing quality attribute requirements

TABLE 14-5 Translating quality attributes into technical specifications



Quality attributes	Likely technical information category
Installability, integrity, interoperability, reliability, robustness, safety, security, usability, verifiability	Functional requirement
Availability, efficiency, modifiability, performance, reliability, scalability	System architecture
Interoperability, security, usability	Design constraint
Efficiency, modifiability, portability, reliability, reusability, scalability, verifiability, usability	Design guideline
Portability	Implementation constraint

Constraints

- Definition
- Purpose
- Sources of constraints
- Example
  - CON-1. The user clicks at the top of the project list to change the sort sequence. [specific user interface control imposed as a design constraint on a functional requirement]
  - CON-2. Only open source software available under the GNU General Public License may be used to implement the product. [implementation constraint]
  - CON-3. The application must use Microsoft .NET framework 4.5. [architecture constraint]
  - CON-4. ATMs contain only $20 bills. [physical constraint]
  - CON-5. Online payments may be made only through PayPal. [design constraint] CON-6. All textual data used by the application shall be stored in the form of XML files. [data constraint]

Handling quality attributes on agile projects

- It can be difficult and expensive to retrofit desired quality characteristics into a product late in development or after delivery.
- That’s why even agile projects that develop requirements and deliver functionality in small increments need to specify significant quality attributes and constraints early in the project.
- This allows developers to make appropriate architectural and design decisions as a foundation for the desired quality characteristics.
- Nonfunctional requirements need to have priority alongside user stories; you can’t defer their implementation until a later iteration.
- Example