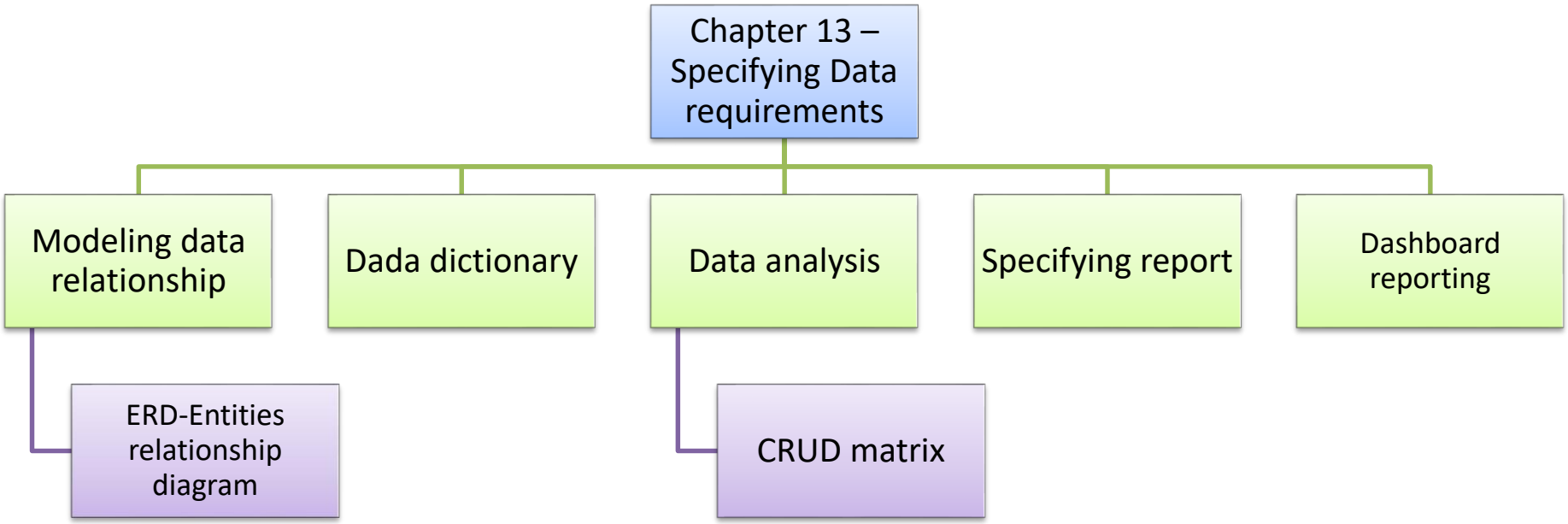
FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

CHAPTER 14

Beyond functionality

Review chapter 13



Objectives

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

Contents

1. Software quality attributes.
2. Exploring quality attributes.
3. Defining quality requirements.
4. Specifying quality requirements with Planguage.
5. Quality attribute trade-offs (sự cân bằng).
6. Implementing quality attribute requirements.
7. Constraints.
8. Handling quality attributes on agile projects.

Beyond functionality

- There's more to software success than just delivering the right functionality. Users also have expectations, often unstated (thường ko nói ra), about *how well* the product will work.
- Such expectations include how easy it is to use, how quickly it executes, how rarely it fails, how it handles unexpected conditions—and perhaps, how loud it is.
- Such characteristics, collectively known as *quality attributes*, *quality factors*, *quality requirements*, *quality of service requirements*, constitute a major portion of the system's nonfunctional requirements.

Software quality attributes

- External quality factors are primarily important to users, whereas internal qualities are more significant to development and maintenance staff.
- Internal quality attributes indirectly contribute to customer satisfaction by making the product easier to enhance, correct, test, and migrate to new platforms.
- Certain attributes are of particular importance on certain types of projects:
  - Embedded systems: performance, efficiency, reliability, robustness, safety, security, usability,…
  - Internet and corporate applications: availability, integrity, interoperability (khả năng tương tác), performance, scalability, security, usability,…
  - Desktop and mobile systems: performance, security, usability,…

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

- Different projects will demand different sets of quality attributes for success.
- the following practical approach for identifying and specifying the most important attributes for your project (5 steps):
  - Step 1: Start with a broad taxonomy: Begin with a rich set of quality attributes to consider, such as those listed in Table 14-1. This broad starting point reduces the likelihood (khả năng có thể xảy ra) of overlooking an important quality dimension.

a less-than sign (<) in the cell indicates that the attribute in the row is more important; a caret symbol (^) points to the attribute at the top of the column as being more important.

Attribute	Score	Availability	Integrity	Performance	Reliability	Robustness	Security	Usability	Verifiability
Availability	2		^	^	^	<	^	^	<
Integrity	6			<	<	<	^	<	<
Performance	4				<	<	^	^	<
Reliability	2					<	^	^	^
Robustness	1						^	^	<
Security	7							<	<
Usability	5								<
Verifiability	1								

FIGURE 14-1 *Sample quality attribute prioritization for an airport check-in kiosk.*

Exploring quality attributes

- Step 2: Reduce the list
  - Hold a meeting for cross-section of stakeholders to assess which of the attributes are likely to be important to the project.
  - EX: An airport check-in kiosk needs to emphasize usability (because most users will encounter it infrequently) and security (because it has to handle payments).
- Step 3: Prioritize the attributes:
  - Prioritizing the pertinent attributes sets the focus for future elicitation discussions.
  - Pairwise ranking comparisons (so sánh xếp hạng theo cặp) can work efficiently with a small list of items like this.
- Step 4: Elicit specific expectations for each attribute:
  - The comments users make during requirements elicitation supply some clues about the quality characteristics they have in mind for the product.
- Step 5: Specify well-structured quality requirements:
  - When writing quality requirements, keep in mind the useful SMART mnemonic—make them *Specific*, *Measurable*, *Attainable*, *Relevant*, and *Time-sensitive*.
  - Quality requirements need to be measurable to establish a precise agreement on expectations among the BA, the customers, and the development team.

Defining quality requirements

- External quality attributes:
  - External quality attributes describe characteristics that are observed when the software is executing. They profoundly influence the user experience and the user's perception of system quality. It includes Availability, Installability, Integrity, Interoperability, Performance, Reliability, Robustness, Safety, Security, Usability,
  - Availability: Availability is a measure of the planned up time during which the system's services are available for use and fully operational.
  - Installability: Software is not useful until it is installed on the appropriate device or platform. Installability describes how easy is it to perform these operations correctly.
    - Initial installation
    - Recovery from an incomplete, incorrect, or user-aborted installation
    - Reinstallation of the same version
    - Installation of a new version
    - Reverting to a previous version
    - Installation of additional components or updates
    - Uninstallation

Defining quality requirements

- External quality attributes:
  - Integrity: Integrity deals with preventing information loss and preserving the correctness of data entered into the system. Integrity requirements have no tolerance (bao dung) for error: the data is either in good shape and protected, or it is not.
  - Interoperability: Interoperability indicates how readily the system can exchange data and services with other software systems and how easily it can integrate with external hardware devices. To assess interoperability, you need to know which other applications the users will employ in conjunction with your product and what data they expect to exchange.
  - Performance: Performance represents the responsiveness of the system to various user inquiries (yêu cầu) and actions, but it encompasses much more than that.

TABLE 14-2 Some aspects of performance

Performance dimension	Example
Response time	Number of seconds to display a webpage
Throughput	Credit card transactions processed per second
Data capacity	Maximum number of records stored in a database
Dynamic capacity	Maximum number of concurrent users of a social media website
Predictability in real-time systems	Hard timing requirements for an airplane's flight-control system
Latency	Time delays in music recording and production software
Behavior in degraded modes or overloaded conditions	A natural disaster leads to a massive number of emergency telephone system calls

Defining quality requirements

- External quality attributes:
  - **Reliability:** The probability of the software executing without failure for a specific period of time is known as reliability. Reliability problems can occur because of improper inputs, errors in the software code itself, components that are not available when needed, and hardware failures.
  - **Robustness:** Robustness is the degree to which a system continues to function properly when confronted with invalid inputs, defects in connected software or hardware components, external attack, or unexpected operating conditions. Robust software recovers gracefully (mềm mại) from problem situations and is forgiving of (bỏ qua) user mistakes.
  - **Safety:** Safety requirements deal with the need to prevent a system from doing any injury (gây thương tích) to people or damage to property (tài sản). Safety requirements might be dictated by government regulations or other business rules, and legal or certification issues could be associated with satisfying such requirements. Safety requirements frequently are written in the form of conditions or actions the system must not allow to occur.

Defining quality requirements

- External quality attributes:
  - Security: Security deals with blocking unauthorized access to system functions or data, ensuring that the software is protected from malware attacks, and so on. Security is a major issue with Internet software. Users of e-commerce systems want their credit card information to be secure. Web surfers don't want personal information or a record of the sites they visit to be used inappropriately. .
  - Usability: Usability is often called as *user-friendliness, ease of use*, and *human engineering*. Usability measures the effort required to prepare input for a system, operate it, and interpret its outputs.

TABLE 14-3 Possible design approaches for ease of learning and ease of use

Ease of learning	Ease of use
Verbose prompts	Keyboard shortcuts
Wizards	Rich, customizable menus and toolbars
Visible menu options	Multiple ways to access the same function
Meaningful, plain-language messages	Autocompletion of entries
Help screens and tooltips	Autocorrection of errors
Similarity to other familiar systems	Macro recording and scripting capabilities
Limited number of options and widgets displayed	Ability to carry over information from a previous transaction
	Automatically fill in form fields
	Command-line interface

Defining quality requirements

- Internal quality attributes
  - Internal quality attributes are not directly observable during execution of the software. They are properties that a developer or maintainer perceives while looking at the design or code to modify it, reuse it, or move it to another platform.
  - Some internal quality attributes: Efficiency, Modifiability, Portability, Reusability, Scalability, Verifiability.
  - Efficiency: Efficiency is closely related to the external quality attribute of performance. Efficiency is a measure of how well the system utilizes processor capacity, disk space, memory, or communication bandwidth. If a system consumes too much of the available resources, users will encounter degraded (sụt giảm) performance.

Defining quality requirements

- Internal quality attributes
  - **Modifiability:** Modifiability addresses how easily the software designs and code can be understood, changed, and extended. Modifiability encompasses several other quality attribute terms that relate to different forms of software maintenance. If developers anticipate making many enhancements, they can choose design approaches that maximize the software's modifiability.
  - **Portability:** The effort needed to migrate software from one operating environment to another is a measure of portability. Portability has become increasingly important as applications must run in multiple environments, such as Windows, Mac, and Linux; iOS and Android; and PCs, tablets, and phones. Data portability requirements are also important.
  - **Reusability:** Reusability indicates the relative effort required to convert a software component for use in other applications. Reusable software must be modular, well documented, independent of a specific application and operating environment,... Numerous project artifacts offer the potential for reuse, including requirements, architectures, designs, code, tests, business rules, data models, user class descriptions, stakeholder profiles, and glossary terms.

Defining quality requirements

- Internal quality attributes
  - Scalability: Scalability requirements address the ability of the application to grow to accommodate (phù hợp với) more users, data, servers, geographic locations, transactions, network traffic, searches, and other services without compromising (ko ảnh hưởng) performance or correctness. Scalability has both hardware and software implications.
  - Verifiability: Verifiability refers to how well software components or the integrated product can be evaluated to demonstrate whether the system functions as expected. Systems with high verifiability can be tested both effectively (hiệu quả công việc) and efficiently (hiệu quả về kinh tế). Designing software for verifiability means making it easy to place the software into the desired pretest state, to provide the necessary test data, and to observe the result of the test.

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** (thang đo) Seconds of elapsed time (trôi qua) between pressing the Enter key or clicking OK to request a report and the beginning of the display of the report.
- **METER** (dụng cụ đo) Stopwatch (đồng hồ bấm giờ) 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** (tính co dãn) 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

- Certain attribute combinations have inescapable trade-offs.
- Users and developers must decide which attributes are more important than others, and they must respect those priorities when they make decisions.
- A plus sign in a cell indicates that increasing the attribute in the corresponding row usually has a positive effect on (tác động tích cực lên) the attribute in the column.
- A minus sign in a cell means that increasing the attribute in that row generally adversely (bất lợi) affects the attribute in the column.
- An empty cell indicates that the attribute in the row has little effect on the attribute in the column.

	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

- Designers and programmers will have to determine the best way to satisfy each quality requirement.
- Although these are nonfunctional requirements, they can lead to derived functional requirements, design guidelines, or other types of technical information that will produce the desired product characteristics.
- This translation from external or internal quality requirements into corresponding technical information is part of the requirements analysis and high-level design processes.

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: A constraint places restrictions on the design or implementation choices available to the developer.
- Purpose:
  - Constraints can be imposed by external stakeholders, by other systems that interact with the one you're building or maintaining, or by other life cycle activities for your system, such as transition and maintenance.
  - Other constraints result from existing agreements, management decisions, and technical decisions (ISO/IEC/IEEE 2011).
- Sources of constraints:
  - Specific technologies, tools, languages, and databases that must be used or avoided.
  - Restrictions because of the product's operating environment or platform, such as the types and versions of web browsers or operating systems that will be used.
  - Required development conventions or standards. (For instance, if the customer's organization will be maintaining the software, the organization might specify design notations and coding standards that a subcontractor must follow.)
  - Backward compatibility (tương thích ngược) with earlier products and potential forward compatibility, such as knowing which version of the software was used to create a specific data file.

Constraints

- Sources of constraints:
  - Limitations or compliance requirements imposed by regulations or other business rules.
  - Hardware limitations such as timing requirements, memory or processor restrictions, size, weight, materials, or cost.
  - Physical restrictions because of the operating environment or because of characteristics or limitations of the users.
  - Existing interface conventions to be followed when enhancing an existing product.
  - Interfaces to other existing systems, such as data formats and communication protocols.
  - Restrictions because of the size of the display, as when running on a tablet or phone.
  - Standard data interchange formats used, such as XML, or RosettaNet for e-business.

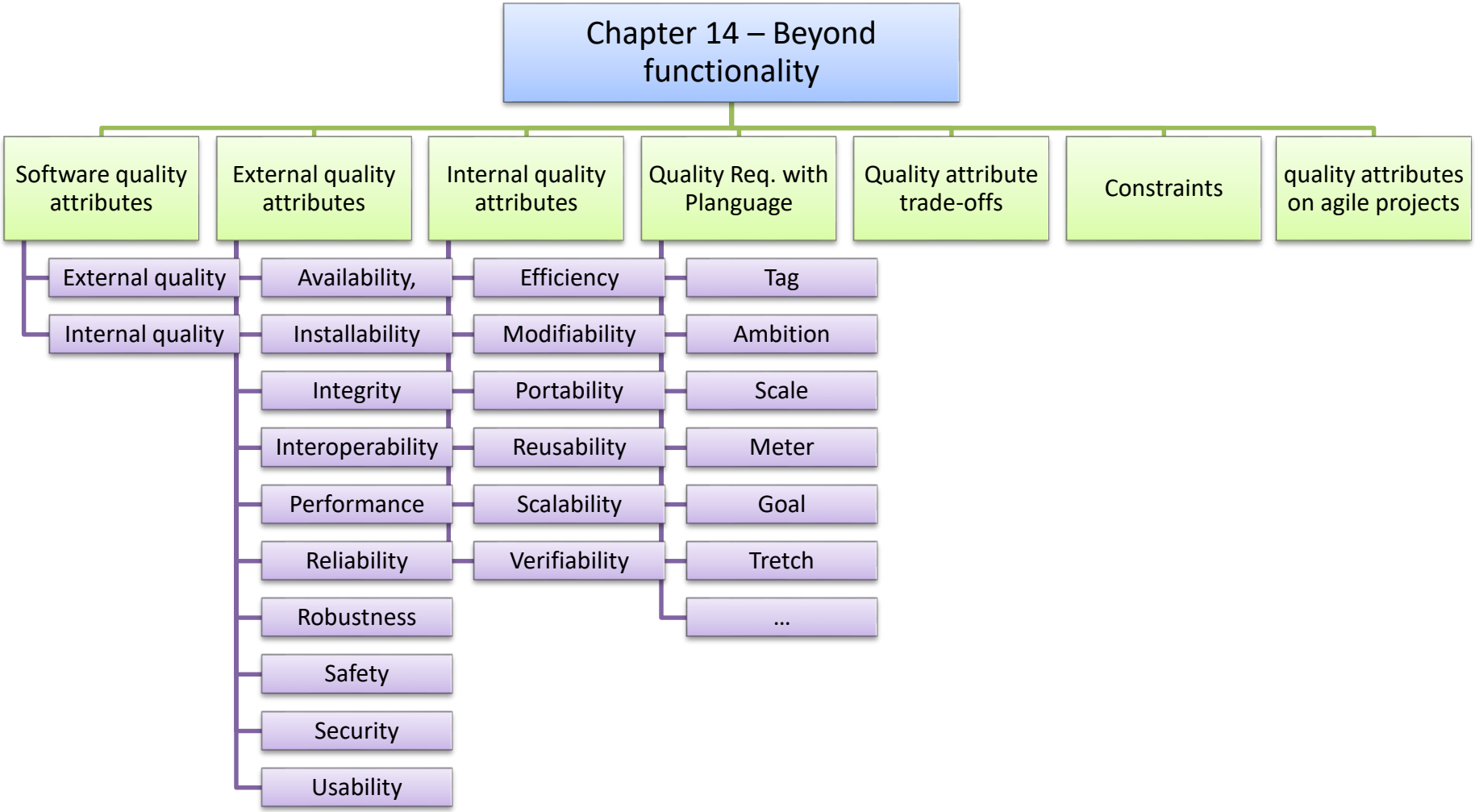
Constraints

- 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 (trang bị thêm) desired quality characteristics into a product late in development or after delivery.
- That's why even (đó là lý do tại sao ngay cả) 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 (dọc theo) user stories; you can't defer their implementation until a later iteration.
- The quality requirements can span multiple stories and multiple iterations.
- Example:
  - *"As a help desk technician, I want the knowledge base (cơ sở kiến thức) to respond to queries within five seconds so the customer doesn't get frustrated and hang up (cúp máy)."*
  - *"As an account owner, I want to prevent unauthorized users from accessing my account so I don't lose any money."*

Review chapter 13



THE END
THANK YOU!