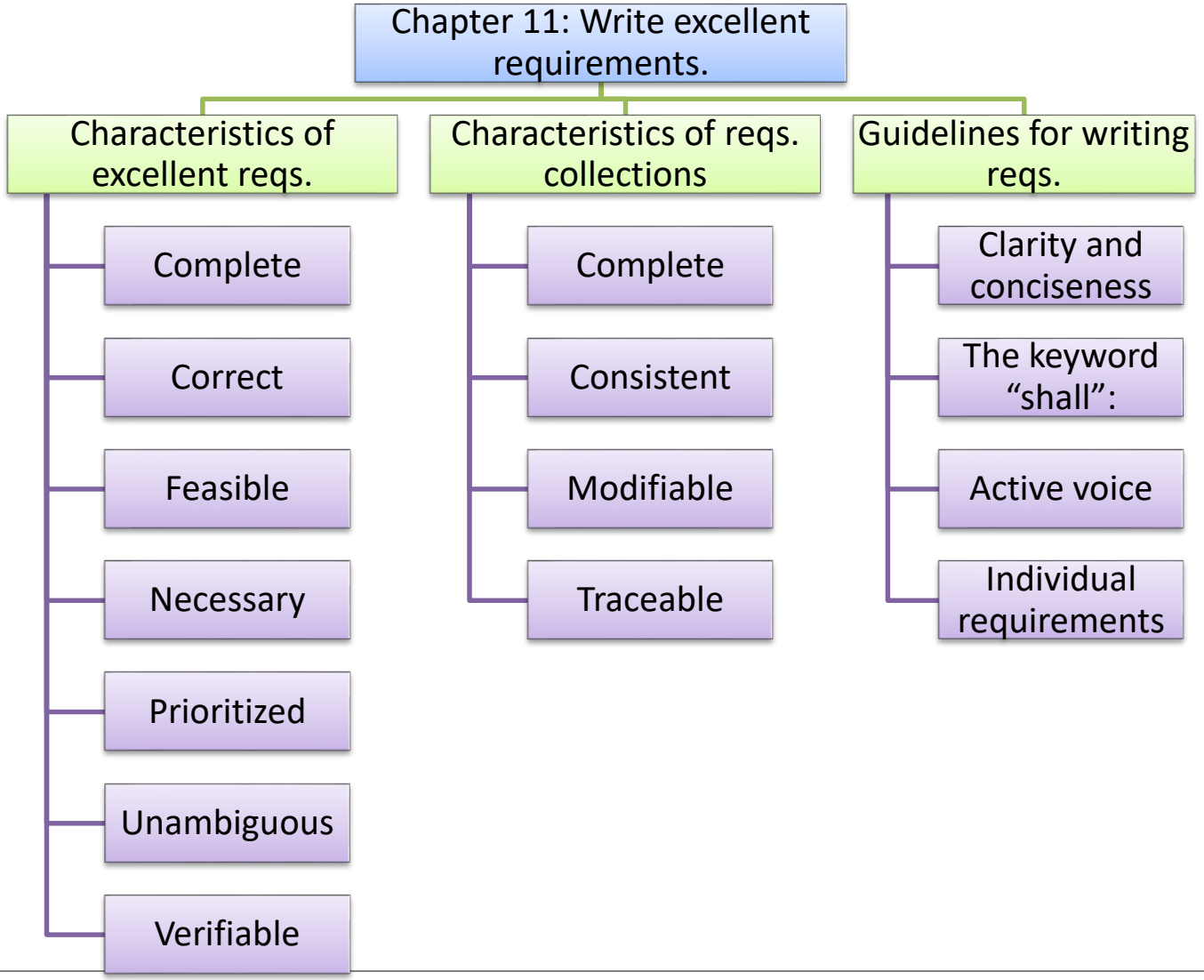
FPT Education
FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

CHAPTER 12

A picture is worth 1024 words

Review chapter 11



Objectives

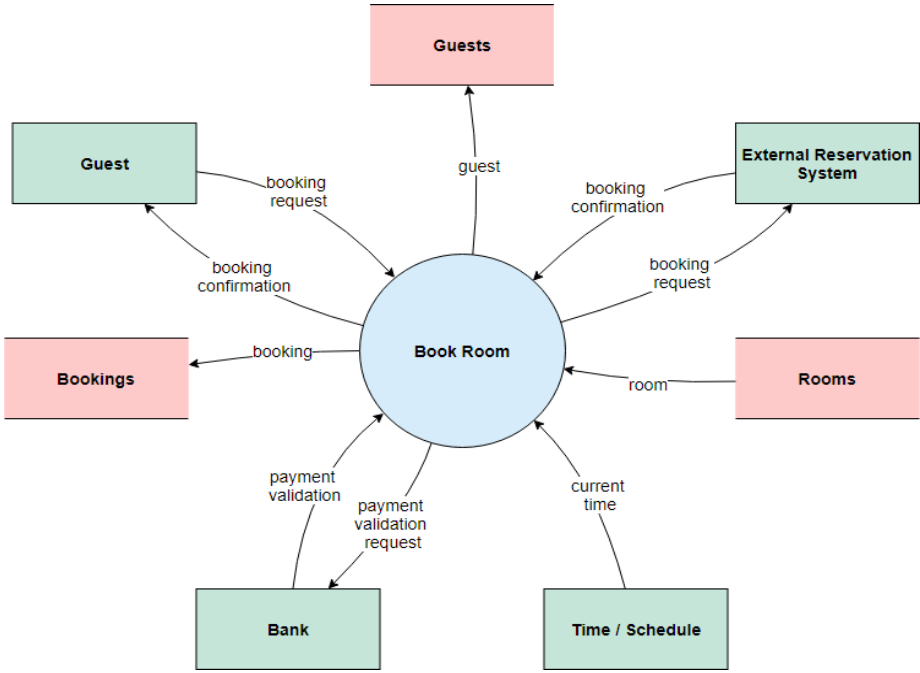
- After finish this chapter, student should understand the purpose of every type of diagrams, when and how to build them.
- Student could choice the plan to model requirements.

Contents

1. A useful of picture in requirements representation.
2. Modeling the requirements.
3. From voice of the customer to analysis models.
4. Selecting the right representations.
5. Data flow diagram.
6. Swimlane diagram.
7. State-transition diagram and state table.
8. Decision tables and decision trees.
9. Modeling on agile projects.

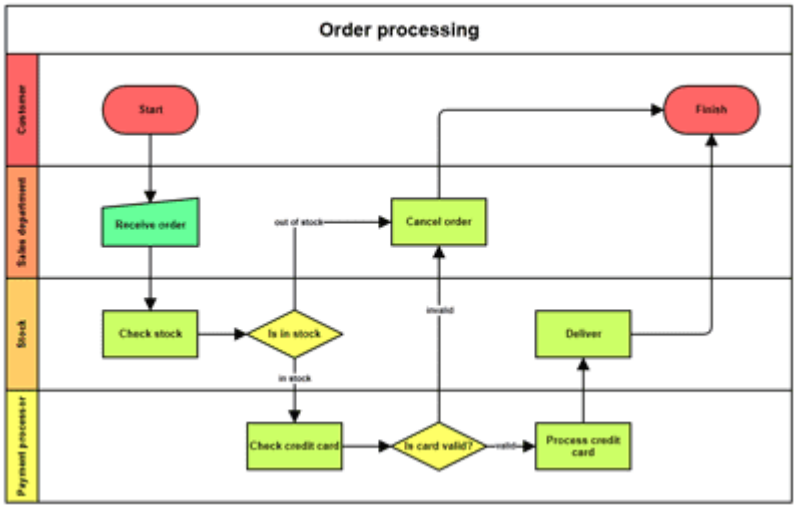
A useful of picture in requirements representation

- Diagrams communicate certain types of information more efficiently than text can.
- Pictures help bridge (thu hẹp) language and vocabulary barriers among team members.
- There are many different diagrams and modeling techniques to choose from to create visual representations of the requirements.



Modeling the requirements

- Visual requirements models can help you identify missing, extraneous (ko liên quan), and inconsistent requirements.
- Given the limitations (với sự hạn chế) of human short-term memory, analyzing a list of one thousand requirements for inconsistencies, duplication, and extraneous requirements is nearly impossible.



Swimlane diagrams sample

Modeling the requirements

- Visual requirements models described in this book include:
  - Data flow diagrams (DFDs).
  - Process flow diagrams such as swimlane diagrams (Sơ đồ bơi).
  - State-transition diagrams (STDs) and state tables
  - Dialog maps.
  - Decision tables and decision trees.
  - Event-response tables.
  - Feature trees (discussed in Chapter 5, “Establishing the business requirements”).
  - Use case diagrams (discussed in Chapter 8, “Understanding user requirements”).
  - Activity diagrams (also discussed in Chapter 8).
  - Entity-relationship diagrams (ERDs) (discussed in Chapter 13, “Specifying data requirements”).

From voice of the customer to analysis models

- By listening carefully to how customers present their requirements, the BA can pick out keywords that translate into specific model elements.

TABLE 12-1 Relating the customer's voice to analysis model components

Type of word	Examples	Analysis model components
Noun	People, organizations, software systems, data elements, or objects that exist	■ External entities, data stores, or data flow (DFD) ■ Actors (use case diagram) ■ Entities or their attributes (ERD) ■ Lanes (swimlane diagram) ■ Objects with states (STD)
Verb	Actions, things a user or system can do, or events that can take place	■ Processes (DFD) ■ Process steps (swimlane diagram) ■ Use cases (use case diagram) ■ Relationships (ERD) ■ Transitions (STD) ■ Activities (activity diagram) ■ Events (event-response table)
Conditional	Conditional logic statements, such as if/then	■ Decisions (decision tree, decision table, or activity diagram) ■ Branching (swimlane diagram or activity diagram)

From voice of the customer to analysis models

- EX: Significant unique nouns are highlighted in **bold,** verbs are in *italics*, and conditional statements are in ***bold italics***; look for these keywords in the analysis models shown later in this chapter.

A **chemist** or a member of the **chemical stockroom staff** can *place* a **request** for one or more **chemicals *if the user is an authorized requester***. The request can be *fulfilled* either by *delivering* a **container** of the chemical that is already in the chemical stockroom's **inventory** or by placing an **order** for a new container of the chemical with an outside **vendor**. ***If the chemical is hazardous***, the chemical can be delivered only ***if the user is trained***. The **person** placing the request must be able to *search* **vendor catalogs** online for specific chemicals while *preparing* his request. The system needs to *track* the **status** of every chemical request from the time it is prepared until the request is either fulfilled or *canceled*. It also needs to track the **history** of every chemical container from the time it is *received* at the **company** until it is fully *consumed* or *disposed* of.

Selecting the right representations

- Rarely does a team need to create a complete set of analysis models for an entire system.
- Focus your modeling on the most complex and riskiest portions of the system and on those portions most subject to ambiguity or uncertainty.
- Safety-critical, security-critical, and mission-critical system elements are good candidates (những ứng viên tốt) for modeling because the impact of defects in those areas is so severe.
- Next slide shows you some representation techniques you can choose.

Selecting the right representations

TABLE 12-2 Choosing the most appropriate representation techniques

Information depicted	Representation techniques
System external interfaces	■ *The context diagram* and *use case diagram* identify objects outside the system that connect to it. The context diagram and *data flow diagrams* illustrate the system inputs and outputs at a high level of abstraction. The *ecosystem map* identifies possible systems that interact, but includes some that do not interface directly as well. *Swimlane diagrams* show what happens in the interactions between systems. ■ External interface details can be recorded in input and output *file formats* or *report layouts*. Products that include both software and hardware components often have *interface specifications* with data attribute definitions, perhaps in the form of an application programming interface or specific input and output signals for a hardware device.
Business process flow	■ A top-level *data flow diagram* represents how a business process handles data at a high level of abstraction. *Swimlane diagrams* show the roles that participate in executing the various steps in a business process flow. ■ Refined levels of *data flow diagrams* or *swimlane diagrams* can represent business process flows in considerable detail. Similarly, *flowcharts* and *activity diagrams* can be used at either high or low levels of abstraction, although most commonly they are used to define the details of a process.
Data definitions and data object relationships	■ The *entity-relationship diagram* shows the logical relationships between data objects (entities). *Class diagrams* show the logical connections between object classes and the data associated with them. ■ The *data dictionary* contains detailed definitions of data structures and individual data items. Complex data objects are progressively broken down into their constituent data elements.

Selecting the right representations

Information depicted	Representation techniques
System and object states	■ *State-transition diagrams* and *state tables* represent a high-abstraction view of the possible states of a system or object and the changes between states that can take place under certain circumstances. These models are helpful when multiple use cases can manipulate (and change the state of) certain objects. ■ Some analysts create an *event-response table* as a scoping tool, identifying external events that help define the product's scope boundary. You can also specify individual functional requirements with an event-response table by detailing how the system should behave in response to each combination of external event and system state. ■ *Functional requirements* provide the details that describe exactly what user and system behaviors lead to status changes.
Complex logic	■ A *decision tree* shows the possible outcomes from a set of related decisions or conditions. A *decision table* identifies the unique functional requirements associated with the various combinations of true and false outcomes for a series of decisions or conditions.
User interfaces	■ The *dialog map* provides a high-level view of a proposed or actual user interface, showing the various display elements and possible navigation pathways between them. ■ *Storyboards* and *low-fidelity prototypes* flesh out the dialog map by showing what each screen will contain without depicting precise details. *Display-action-response models* describe the display and behavior requirements of each screen. ■ *Detailed screen layouts* and *high-fidelity prototypes* show exactly how the display elements will look. *Data field definitions* and *user interface control descriptions* provide additional detail.

Selecting the right representations

Information depicted	Representation techniques
User task descriptions	■ *User stories*, *scenarios*, and *use case specifications* describe user tasks in various levels of detail. ■ *Swimlane diagrams* illustrate the business process or interplay between multiple actors and the system. *Flowcharts* and *activity diagrams* visually depict the flow of the use case dialog and branches into alternative flows and exceptions. ■ *Functional requirements* provide detailed descriptions of how the system and user will interact to achieve valuable outcomes. *Test cases* provide an alternative low-abstraction view, describing exactly what system behavior to expect under specific conditions of inputs, system state, and actions.
Nonfunctional requirements (quality attributes, constraints)	■ Quality attributes and constraints are usually written in the form of *natural language text*, but that often results in a lack of precision and completeness. Chapter 14, "Beyond functionality" describes a definitive technique for precisely specifying nonfunctional requirements called *Planguage* (Gilb 2005).

Data flow diagram – context diagram

- The *data flow diagrams (DFD)* is the basic tool of structured analysis. A DFD identifies the transformational processes of a system, the collections (stores) of data or physical materials that the system manipulates, and the flows of data or material between processes, stores, and the outside world.
- Purpose
  - DFDs provide a big-picture view of how data moves through a system, which other models don't show well.
  - A DFD gives context to the functional requirements regarding (liên quan đến) how the user performs specific tasks.
  - DFDs can be used as a technique to identify missing data requirements.

Dataflow diagram: example

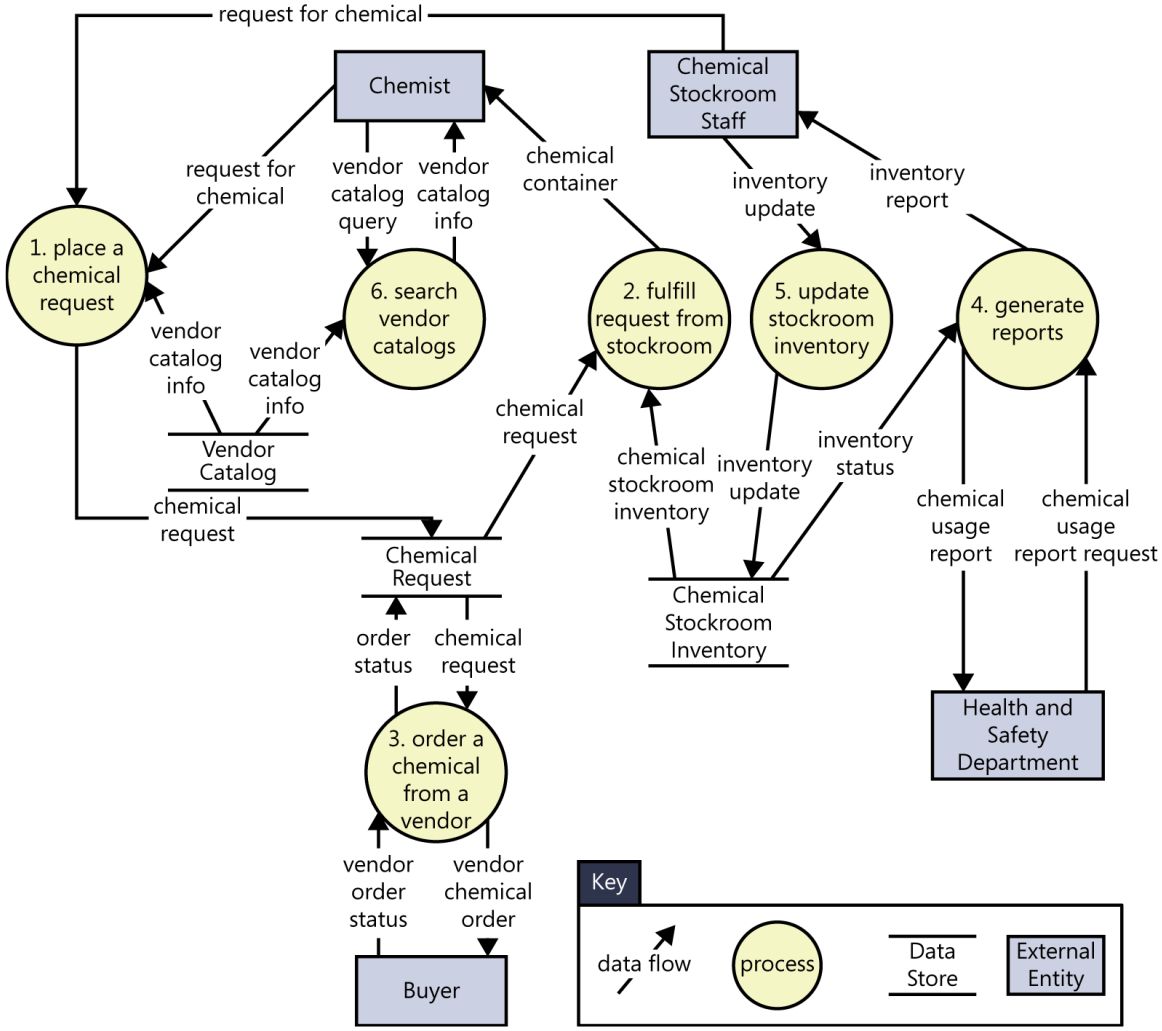


FIGURE 12-1 Partial level 0 data flow diagram for the Chemical Tracking System.

Dataflow diagram: Explanation

- Processes communicate through data stores, not by direct flows from one process to another. Similarly, data cannot flow directly from one store to another or directly between external entities and data stores; it must pass through a process bubble.
- Name each process as a concise action: verb plus object (such as “generate reports”). Use names that are meaningful to the customers.
- Number the processes uniquely and hierarchically. On the level 0 diagram, number each process with an integer. If you create a child DFD for process 3, number the processes in that child diagram 3.1, 3.2, and so on.
- Don’t show more than 8 to 10 processes on a single diagram or it will be difficult to draw, change, and understand.
- Bubbles with flows that are only coming in or only going out are suspect (nghi ngờ). The processing that a DFD bubble represents normally requires both input and output flows.

Swimlane diagram

- Purpose:
  - Provide a way to represent the steps involved in (tham gia vào) a business process or the operations of a proposed software system.
  - They are a variation of flowcharts, subdivided into visual subcomponents called lanes (con đường). The lanes can represent different systems or actors that execute the steps in the process.
  - Swimlane diagrams are most commonly used to show business processes, workflows, or system and user interactions.
  - They are similar to UML activity diagrams. Swimlane diagrams are sometimes called cross-functional diagrams.
  - The swimlane diagram is one of the easiest models for stakeholders to understand because the notation is simple and commonly used.

Swimlane example

- Process steps, shown as rectangles.
- Transitions between process steps, shown as arrows connecting pairs of rectangles.
- Decisions, shown as diamonds with multiple branches leaving each diamond. The decision choices are shown as text labels on each arrow leaving a diamond.
- Swimlanes to subdivide the process, shown as horizontal or vertical lines on the page. The lanes are most commonly roles, departments, or systems. They show who or what is executing the steps in a given lane.

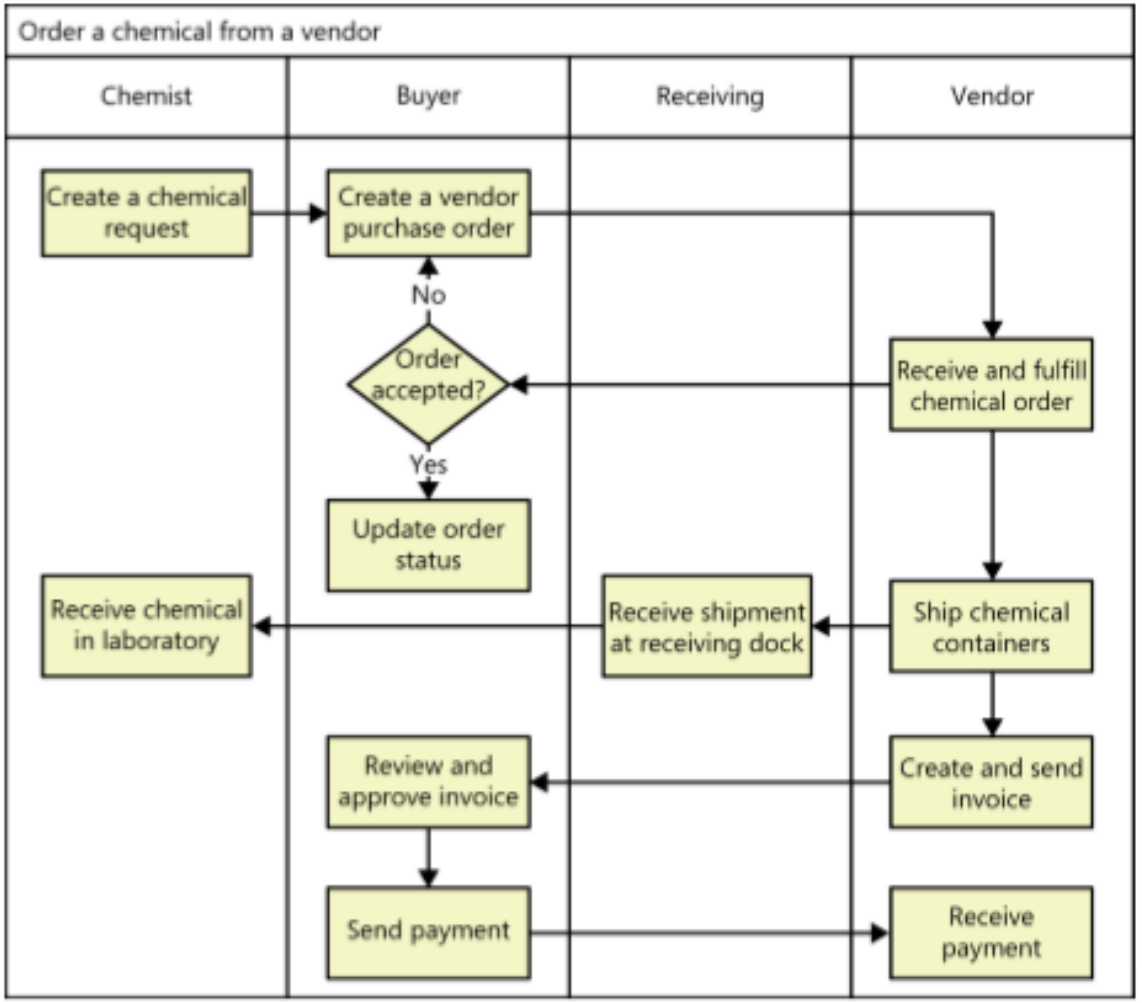


FIGURE 12-2 Partial swimlane diagram for a process in the Chemical Tracking System.

State-transition diagram and state table

- Purpose: Software systems involve (bao hàm) a combination of functional behavior (hoạt động), data manipulation, and state changes. Real-time systems and process control applications can exist in one of a limited number of states at any given time.
- A state change can take place (xảy ra) only when well-defined criteria are satisfied, such as receiving a specific input stimulus (kích thích) under certain conditions.
- State transition diagram contains three type of elements:
  - Possible system states, shown as rectangles.
  - Allowed state changes or transitions, shown as arrows connecting pairs of rectangles.
  - Events or conditions that cause each transition to take place, shown as text labels on each transition arrow

State-transition diagram example

- **In Preparation** The Requester is creating a new request.
- **Postponed** (trì hoãn)The Requester saved a partial request for future completion without either submitting the request to the system or canceling the request operation.
- **Accepted** The Requester submitted a completed chemical request and the system accepted it for processing.
- **Placed** The request must be satisfied by an outside vendor and a buyer has placed an order with the vendor.
- **Fulfilled** The request has been satisfied, either by the delivery of a chemical container from the chemical stockroom to the Requester or by receipt of a chemical from a vendor.
- **Back-ordered** The vendor didn't have the chemical available and notified the buyer that it was back-ordered for future delivery.
- **Canceled** The Requester canceled an accepted request before it was fulfilled, or the buyer canceled a vendor order before it was fulfilled or while it was back-ordered.

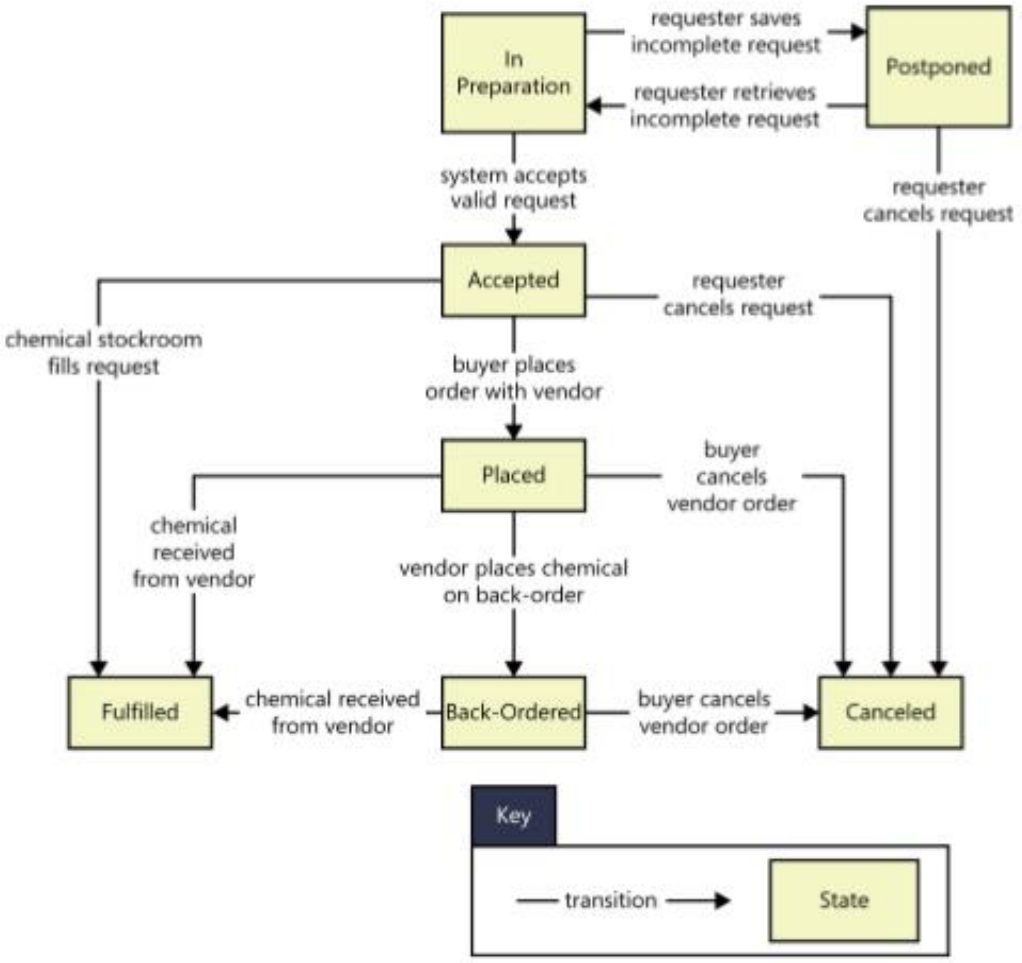


FIGURE 12-3 A partial state-transition diagram for a chemical request in the Chemical Tracking System.

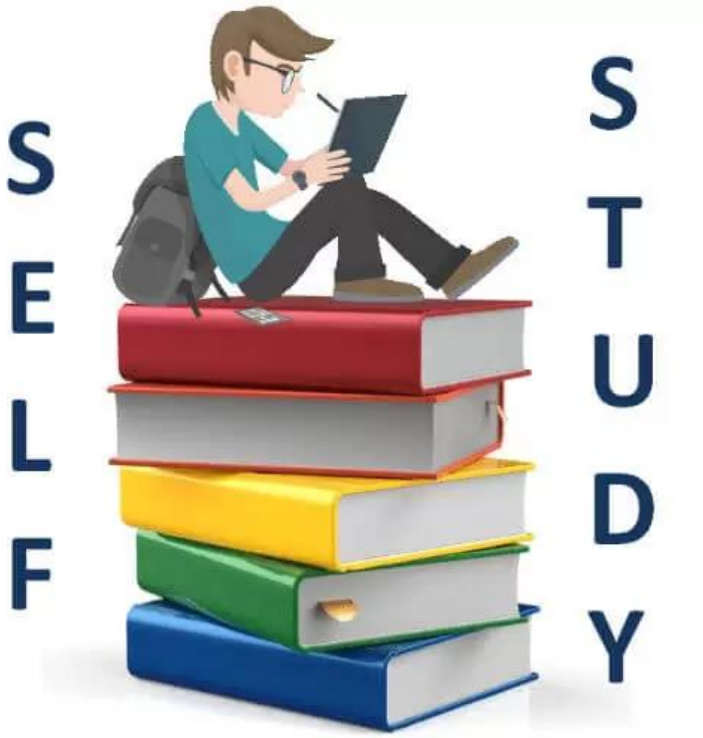
State table example

	In preparation	Postponed	Accepted	Placed	Back-Ordered	Fulfilled	Canceled
In Preparation	no	user saves incomplete request	system accepts valid request	no	no	no	no
Postponed	user retrieves incomplete request	no	no	no	no	no	no
Accepted	no	no	no	buyer places order with vendor	no	chemical stockroom fills request	requester cancels request
Placed	no	no	no	no	vendor places chemical on back-order	chemical received from vendor	buyer cancels vendor order
Back-Ordered	no	no	no	no	no	chemical received from vendor	buyer cancels vendor order
Fulfilled	no	no	no	no	no	no	no
Canceled	no	no	no	no	no	no	no

FIGURE 12-4 State table for a chemical request in the Chemical Tracking System.

Several diagrams for self-studying

- Dialog map
- Decision tables and decision trees
- Event-response tables



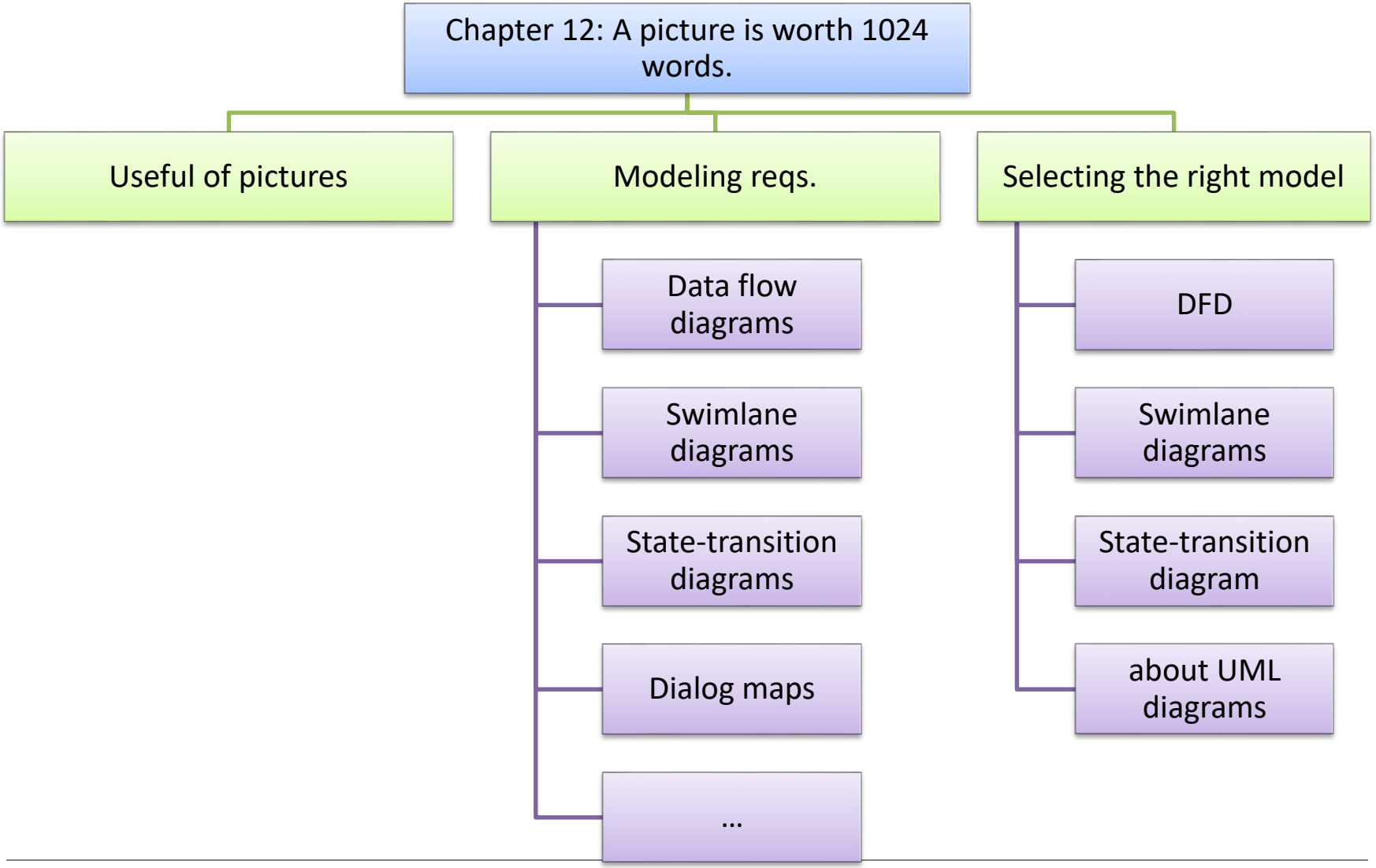
A few words about UML diagrams

- Many projects use object-oriented analysis, design, and development methods. *Objects* typically correspond to real-world items in the business. They represent individual instances derived from a generic template called a *class*.
- Class descriptions encompass both attributes (data) and the operations that can be performed on the attributes.
- A *class diagram* is a graphical way to depict the classes identified during object-oriented analysis and the relationships among them.
- The standard object-oriented modeling language is the Unified Modeling Language. The UML is primarily used for creating design models.
  - Class diagrams, to show the object classes that pertain to (gắn với) the application domain; their attributes, behavior, and properties; and relations among classes.
  - Use case diagrams, to show the relationships between actors external to the system and the use cases with which they interact.

Modeling on agile projects

- The difference in how traditional and agile projects perform modeling is related to when the models are created and the level of detail in them.
- The key point in using analysis models on agile projects—or really, on any project—is to focus on creating only the models you need, only when you need them, and only to the level of detail you need to make sure project stakeholders adequately understand the requirements.

Review chapter 12



THE END
THANK YOU!