The University of Iowa

22C:22 (CS:2820)

Object-Oriented Software Development

Fall 2013

Requirements and Use Cases

by

Cesare Tinelli

System Requirements

Capabilities and conditions the system, and more generally the project, must conform to

System Requirements

- On average, 25% of the requirements change on software projects
- The Unified Process
  - includes a systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system
  - UP embraces change in requirements as a fundamental driver on projects

Main Requirement Types

- **Functional**—features, capabilities, security
- **Usability**—human factors, documentation, help
- **Reliability**—frequency of failure, recoverability, predictability
- **Performance**—response times, throughput, accuracy, availability, resource usage
- **Supportability**—adaptability, maintainability, internationalization, configurability

Other Requirement Types

- **Implementation**—resource limitations, languages and tools, hardware, ...
- **Interface**—constraints imposed by interfacing with external systems
- **Operations**—system management in its operational setting
- **Packaging**—for example, a physical box
- **Legal**—licensing and so forth

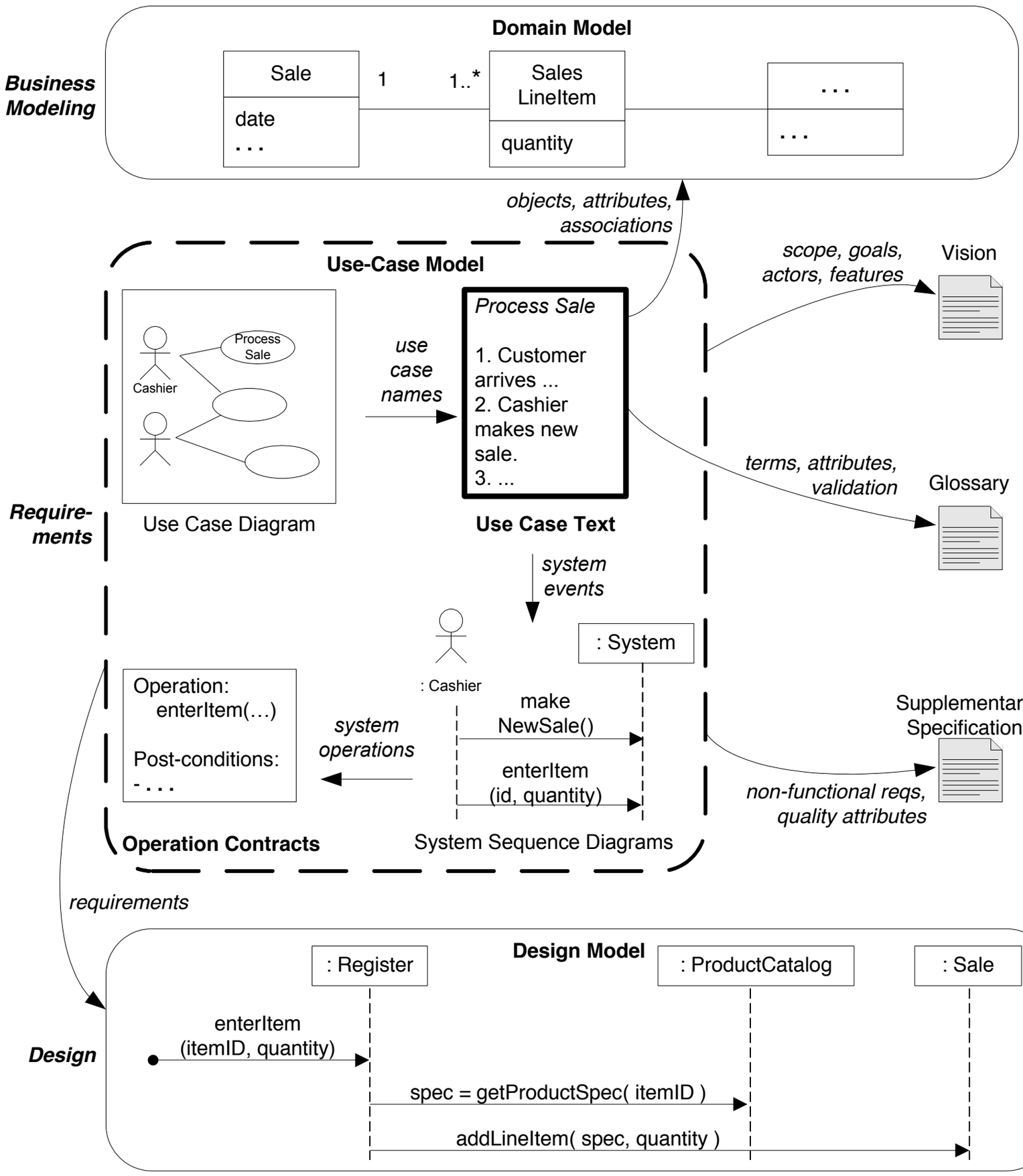
Requirement Artifacts

- **Use Case Model**—a set of typical scenarios of using a system
- **Supplementary Specs**—all non-functional requirements
- **Glossary**—definitions of noteworthy terms and data
- **Vision**—higly level requirements and business case
- **Domain Rules**—requirements and policies that transcend one software project

Use Cases

- Text stories, widely used to discover and record requirements.
- They influence many aspects of a project, including OOA & D
- Are used as input to many subsequent artifacts in the case studies

Sample UP Artifact Relationships



Uses Cases are Stories

- Use cases are *text* stories of some actor using a system to meet goals
- The essence of use cases is
  - discovering and recording functional requirements
  - by writing stories of using a system to fulfill user goals

POS System Example

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Terminology

- **Actor**—Entity with behavior, such as a person (identified by role), computer system, or organization
- **Scenario**—a specific sequence of actions and interactions between actors and the system
- **Use case**—a collection of related success and failure scenarios that describe an actor using a system to support a goal
- **Use case model**—set of all written use cases

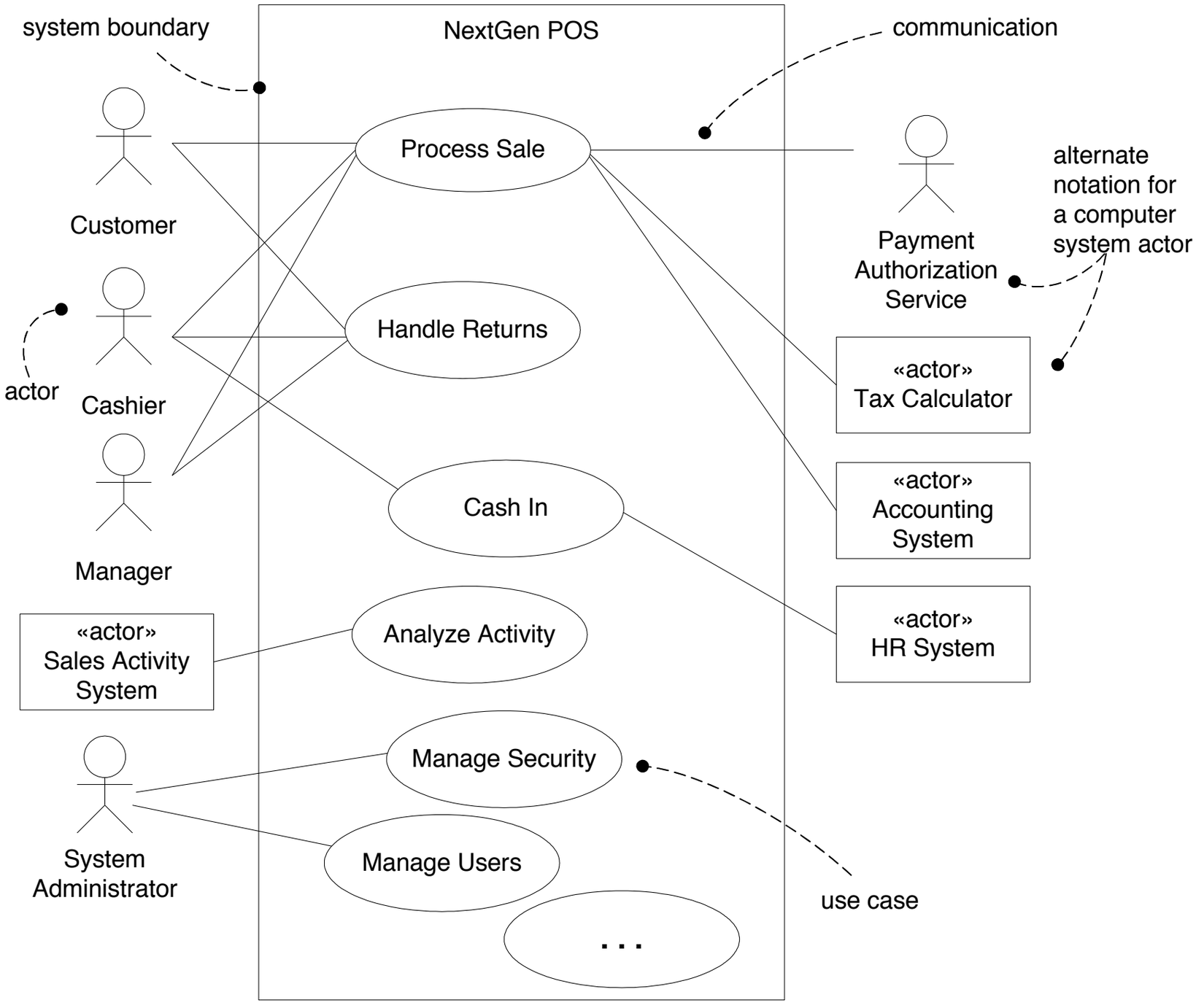
Use Cases and UML

- The Use Case Model may optionally include a UML *use case diagram*
- The diagram shows the names of use cases and actors, and their relationships
- This gives a visual contextual information of a system and its environment
- While there nothing objected-oriented about use cases, they are a key requirements input to OOA/D

Use Cases

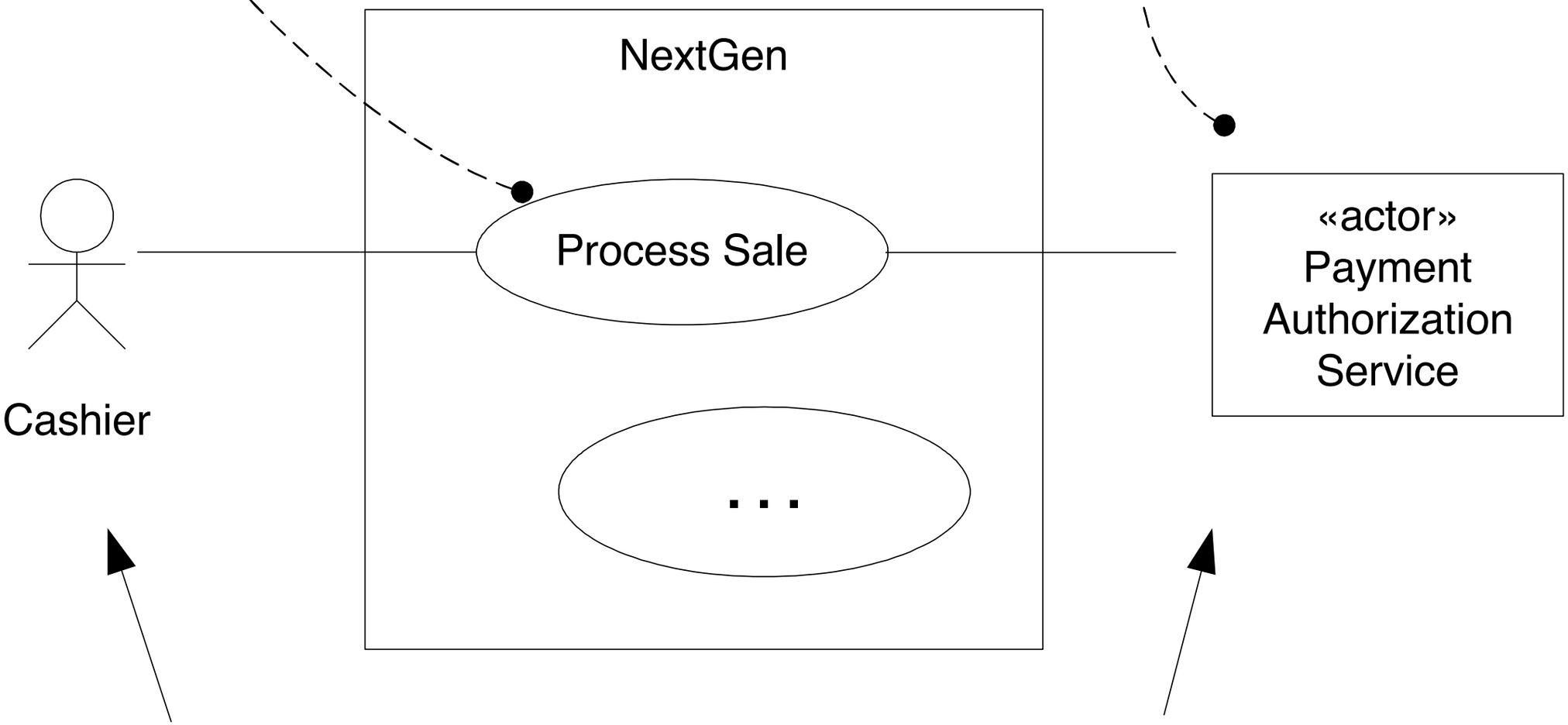
- Use cases are a good way to help keep it simple
- They can be written by or with domain experts or customers
- They can be seen as defining a *contract* of how a system will behave
- They emphasize the user goals and perspective
  - *Who is using the system, what are their typical scenarios of use, and what are their goals?*

A Use Case Diagram



For a use case context diagram, limit the use cases to user-goal level use cases.

Show computer system actors with an alternate notation to human actors.



primary actors on the left

supporting actors on the right

Credits

Notes and figures adapted from

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development by C. Larman. 3rd edition. Prentice Hall/Pearson, 2005.