# Requirements Analysis

# Overview

- What is **requirement?**
- Classification of requirements
- Iterative and evolutionary requirements analysis
- Use Cases

# Requirements

- Definition [LAR]
  - Capabilities and conditions to which the system—and more broadly, the project—must conform
- Focusing on the **WHAT** not the **HOW**

# Requirements Analysis Is Hard

- Major causes of project failures
  - Incomplete requirements
  - Changing requirements
  - Poor user input
- Essential solutions
  - Classification of requirements
  - Iterative and evolutionary requirements analysis
  - Use Cases

# Classification of Requirements

- Functional: features, capabilities, security
  - "The system reads employee records and prints paychecks"
  - All other reqs are non-functional
- Usability: human factors, help documentation
  - "Text on the display must be visible from 1 meter."

# Classification of Requirements

- Reliability: frequency of failure, recoverability, predictability
  - "When doing search, the radar should have 28 hours MTBF(mean time between failures)"
- Performance: response times, throughput, accuracy, availability, resource usage
  - "The server response time is <1 sec for 90% of the accesses"

# Classification of Requirements

- Supportability: adaptability, maintainability, internationalization, configurability
  - "The system should allow frequent and easy changes in the network configuration"
- Implementation: resource limitations, languages, tools, hardware
  - "Must use Linux and Java"

# Iterative and Evolutionary Requirements Analysis

- Motivation
  - 20-50% of the original reqs change because of miscommunication or changing business needs
- Strategies
  - 10-20% of the most architecturally significant, risky, and high-business-value requirements are specified before the initial implementation
  - The short duration of iterations allows quick adaptation and increments of reqs.

# Requirements Elicitation

- Brainstorming
  - Gather stakeholders, collect ideas and prune
- Interviewing
  - Formal or informal interviews with stakeholders
- Ethnography
  - A social scientist observes and analyzes how people actually work
- Strawman/Prototype
  - GUI, flow charts of UIs

# How to do iterative requirement analysis?

- Inception, 2 days
  - Identify names of use cases and features, and key non-functional requirements
  - 10% are analyzed in detail due to high-risk, high-business-value, and architecture significance
- Iteration planning meeting
  - Choose a subset of the 10% for implementation, break them down to detailed iteration tasks

# Possible Timeline

- Elaboration, iteration #1, 4 weeks
  - Design, implement, and test selected features
  - Demo it to collect feedback
  - Pick another 15-20% to analyze in detail (2 days)
- Iteration planning meeting
- Elaboration, iteration #2, 4 weeks
  - Repeat iteration #1

… …

- Elaboration, iteration #4, 4 weeks

# At the end of Elaboration, …

- 80-90% of use cases are analyzed and written in detail
- 10% implementation done
- Other phases do very little work on use cases

# Definitions—Stakeholders

- People who support, benefit from, or are affected by a software project
  - Managers
  - Communicators
  - Software engineers
  - Maintainers
  - System administrators
  - Users
  - Customers

# Definitions

- Use case is a story of using the system to fulfill stakeholder goals
  - It is a **text document**, not a diagram
  - Its name usually contains a verb
- Use-Case Model: the set of all written use cases
- Use-Case Modeling: primarily an act of **writing text**, not drawing diagrams

# Use Cases

# The Role of Use Cases

- The most widely used approach for capturing requirements
- Input to many subsequent activities and artifacts

# Running example: point-of-sale (POS) system [LAR]

- **Process Sale** use case

  *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.*

# Q1: Who Are the Stakeholders?

- Customer
- Cashier
- Store
- Government tax agencies
- Credit card company

# Terms Relevant to Use Cases

- Actor: Something with behavior
  - Person, computer system, organization
- Scenario (use case instance)
  - 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 the system to support a goal

# Three Kinds of Actors

- Primary actor: uses the system to fulfill goals
  - E.g., cashier
  - Why? To find user goals and drive use cases
- Supporting actor: provides a service to the system
  - E.g., Payment authorization service
  - Why? To clarify external interfaces and protocols
- Offstage actor: has an interest in the behavior
  - E.g., Tax agencies
  - Why? To ensure that all necessary interests are identified and satisfied

# *Handle Returns* use case

- Main Success Scenario: A customer arrives with items to return. The cashier uses the system to record each returned item …
- Alternative Scenarios:
  - If they paid by credit, and the reimbursement transaction to their credit account is rejected, pay by cash
  - If the system detects failure to communicate with the external accounting system, …

# Black-Box Use Cases

- Do NOT describe the internal workings of the system
  - Only system responsibilities
  - Focus on "what" the system should do
  - Good: "The system records the sale"
  - Bad:
    - "The system writes the sale to a database"
    - "The system generates SQL INSERT statement for the sale"

# Levels of Formality

- Brief: one-paragraph, for the main success scenario
  - Process Sale example is brief
- Casual: multiple paragraphs that cover several scenarios
  - Handle Return example is casual
- Fully dressed: all steps and variations
  - Developed iteratively during elaboration; the product of requirement analysis

# Fully Dressed Use Case - Outline

Use Case UC1: Process Sale

**Primary Actor:** Cashier

**Stakeholders and interests:**

E.g., Cashier: want accurate and fast payment

**Preconditions**

**Success guarantee**

**Main success scenario**

**Extensions**

**Special requirements**

**Technology and data variation List**

**Frequency of Occurrence**

# Preconditions

- States what **must always** be true before a scenario is begun in the use case
  - Often the postconditions of another use case
  - Don't bother with it unless you are stating something noteworthy
    - "The system has power" -not interesting

**Preconditions:** Cashier is identified and authenticated

# Success Guarantees (Postconditions)

- State what **must** be true on successful completion of the use case—either the success scenario or alternative ones

> **Success guarantee:** Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions are recorded. Receipt is generated.

# Main success scenario (Basic Flow)

- Defer all conditional and branching statements to the Extension section
- Records three kinds of steps:
  - An interaction between actors
  - A validation (usually by the system)
  - A state change by the system

**Main Success Scenario:**

1. Customer arrives at a POS checkout with items to purchase
2. Cashier starts a new sale
3. Cashier enters item identifier
4. System records the item, presents description and price. Price and total are calculated based on a set of rules.

**Main Success Scenario: (cont'd)**

*Repeat 3-4 until cashier indicates done.*

5. System presents total with tax calculated by an external Tax Calculator system.
6. Cashier asks Customer for payment.
7. Cashier enters cash amount tendered, System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system and external Inventory system.
9. System presents receipt.

# Extensions (or Alternative Flows)

- Often comprise the majority of text
- Indicate all the other scenarios or branches, both success and failure
- Notated with respect to its corresponding steps 1..N in the main success scenario.

**Main Success Scenario:**

… …

3. Cashier enters item identifier

… …

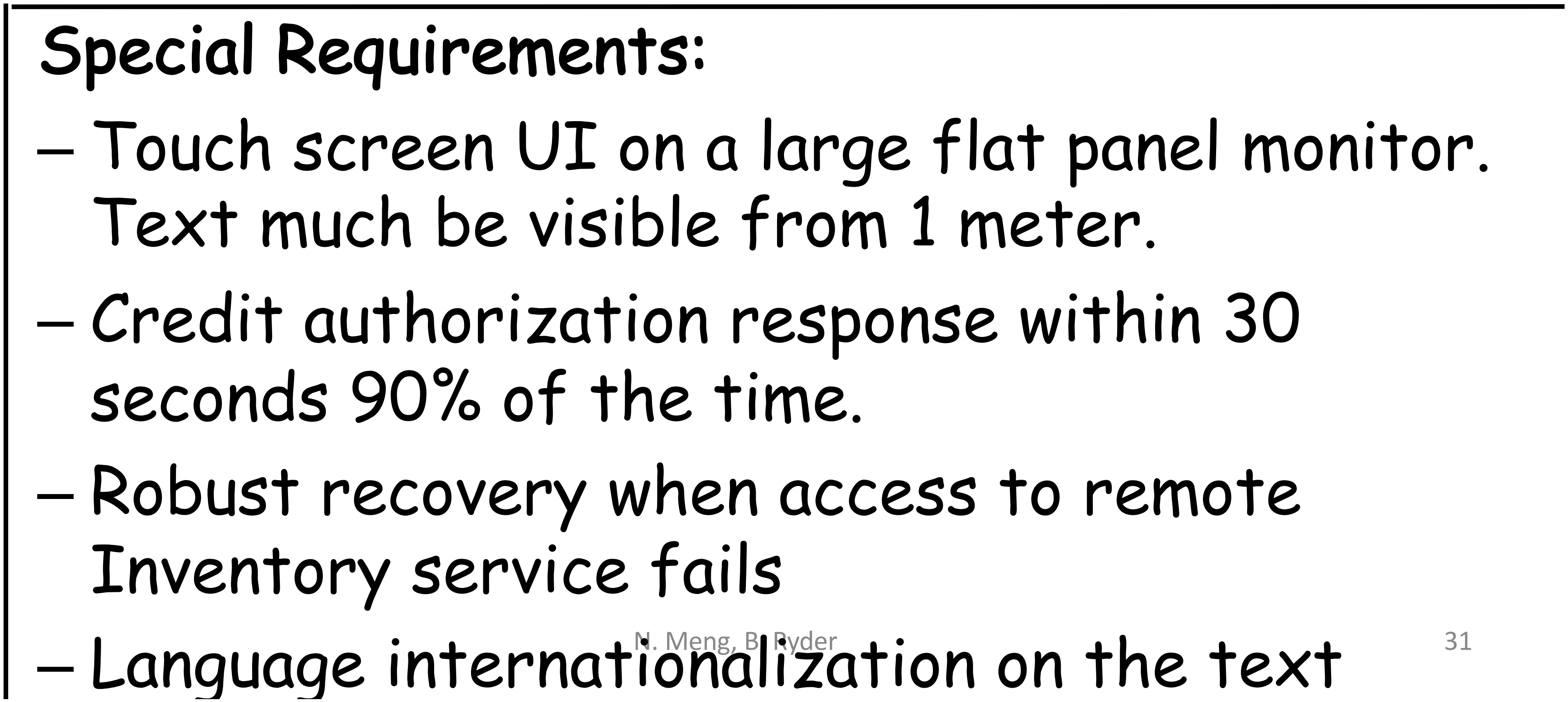
**Extensions:**

3a. Invalid identifier

- 1. System signals errors and rejects entry.
- 2. Cashier responds to the error:
  - 2a. There is a human-readable item ID(e.g., a numeric UPC)
    - 1. Cashier manually enters the item ID.
    - 2. System displays description and price.
      - 2a. Invalid item ID: System signals error. Cashier tries alternative method.
  - 2b. There is no item ID, but there is a price on the tag:
    - 1. Cashier asks Manager to perform an override operation.
    - 2. Manager performs override.
    - 3. Cashier manually enters the price

# Special Requirements

- If a non-functional requirement relates specially to a user case, record it with the use case



**Special Requirements:**

– Touch screen UI on a large flat panel monitor. Text much be visible from 1 meter.

– Credit authorization response within 30 seconds 90% of the time.

– Robust recovery when access to remote Inventory service fails

– Language internationalization on the text

# Technology and Data Variations List

- Technical variations in “how“ something must be done
  - Early design decisions or constraints
    - Technical constraint imposed by stakeholders about input/output technologies.
  - Try to avoid premature design decisions unless they are obvious or unavoidable
- Data scheme variations necessary to understand

**Technology and Data Variations List:**

3a. Item identifier entered by laser scanner or keyboard.

3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.

7a. Credit account information entered by card reader or keyboard.

7b. Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.

# Unified Modeling Language (UML)

- Definition
  - A visual language for specifying, constructing, and documenting the artifacts of systems
  - Standard diagramming notation for drawing pictures related to software
  - Includes 13 types of diagrams

# Two Categories of UML Diagrams

- Structural UML diagrams
  - Class diagram
  - Object diagram

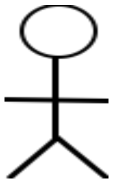
  … …
- Dynamic UML diagrams
  - Use case diagram
  - Sequence diagram

  … …

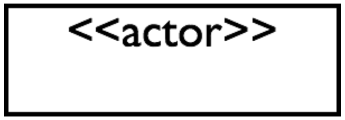
# Use case diagram

- Definition
  - A representation of interactions between actors and the system
- It shows relationship between actors, use cases, and the system
  - the scope of the system
  - the external actors
  - how actors use the system
- It is secondary to text documentation

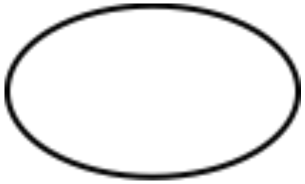
# Legends



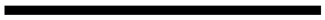
**Actor**: an entity that interacts with the system.



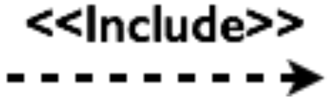
**Actor**: a computer system that interacts with the system under discussion



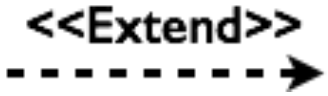
**Use case**: usage of a system



**Association**: relation between an actor and a use case

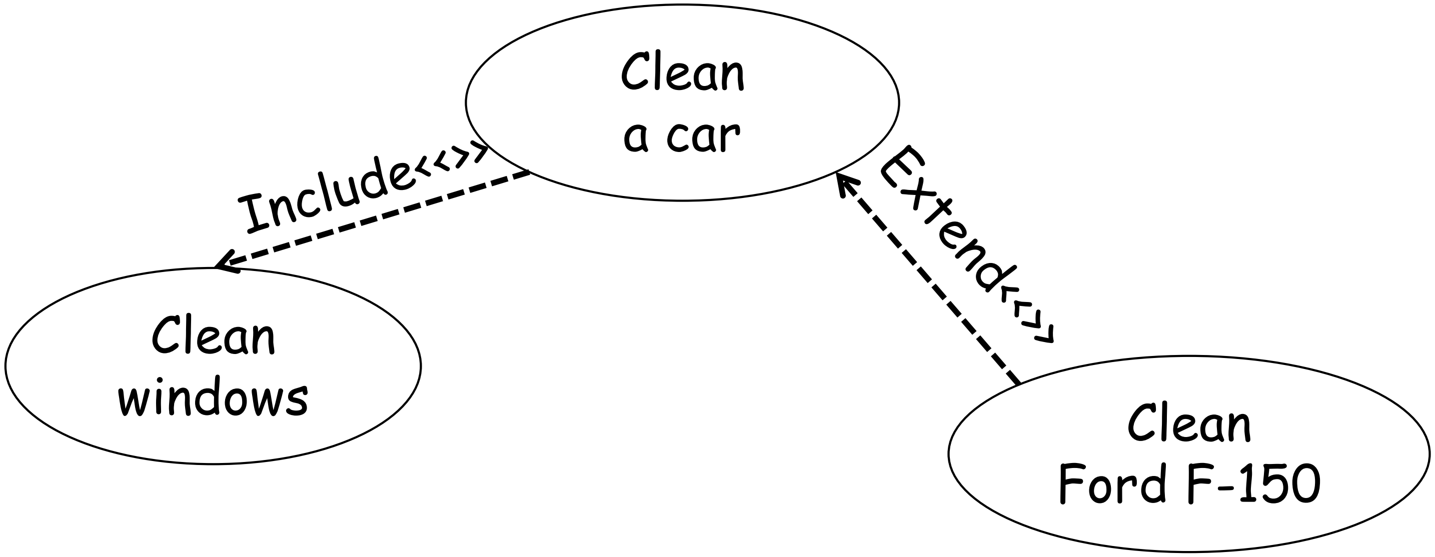


**Includes dependency**: a base use case includes a sub use case as component



**Extends dependency**: a use case extends the behavior of a base use case.

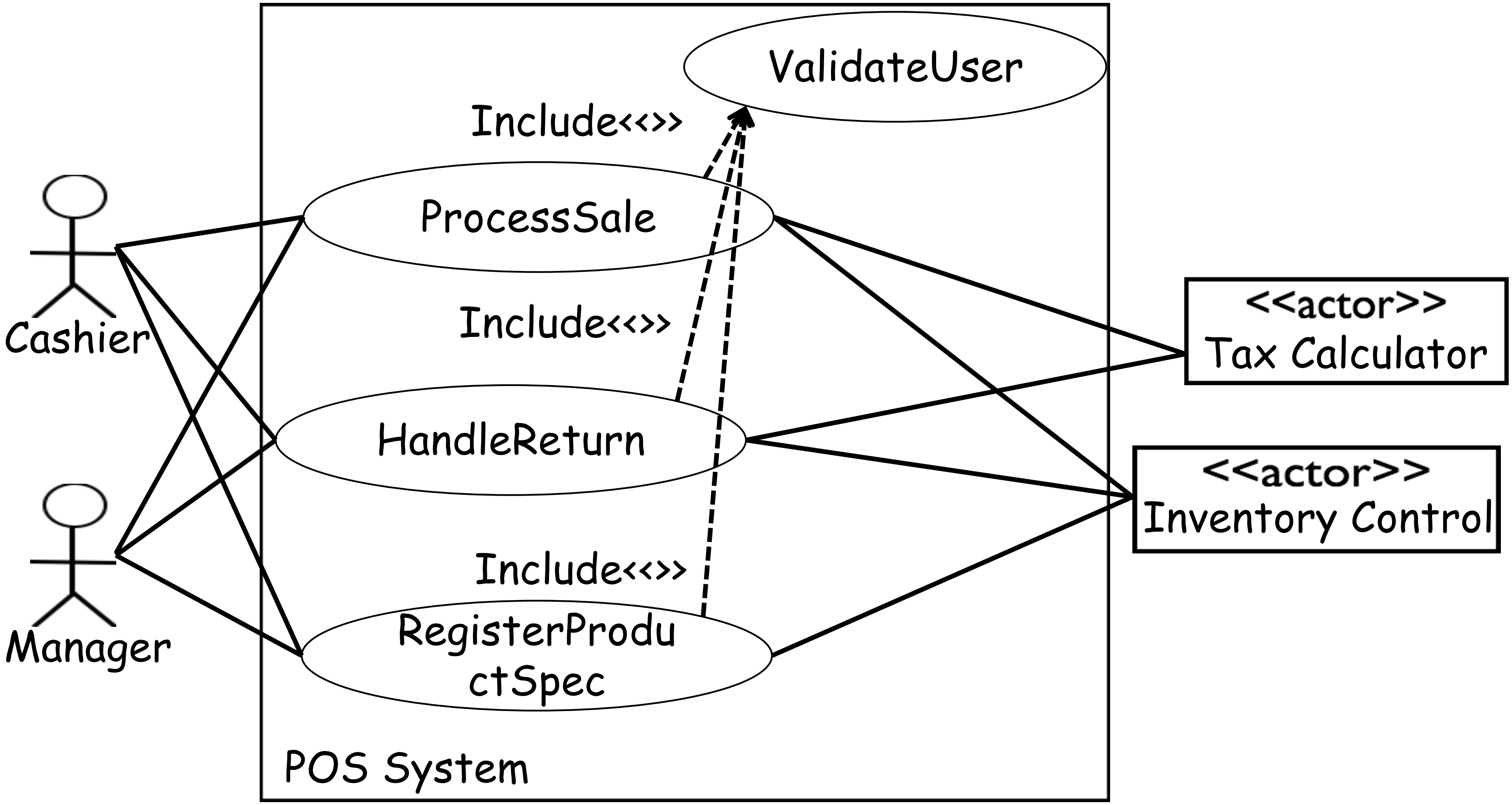
# What are the relations?



# Case study: POS system

- With a POS system,
  - a cashier can perform the following tasks (with help of the manager if necessary):
    - Process sale
    - Handle return
    - Register product specification
  - For each activity, the system may first authenticate the cashier or manager
- The POS system interfaces to third-party tax calculator and inventory control

# Use Case Diagram


ValidateUser
Include<<>>
ProcessSale
Include<<>>
HandleReturn
Include<<>>
RegisterProdu
ctSpec
Cashier
Manager
<<actor>>
Tax Calculator
<<actor>>
Inventory Control
POS System