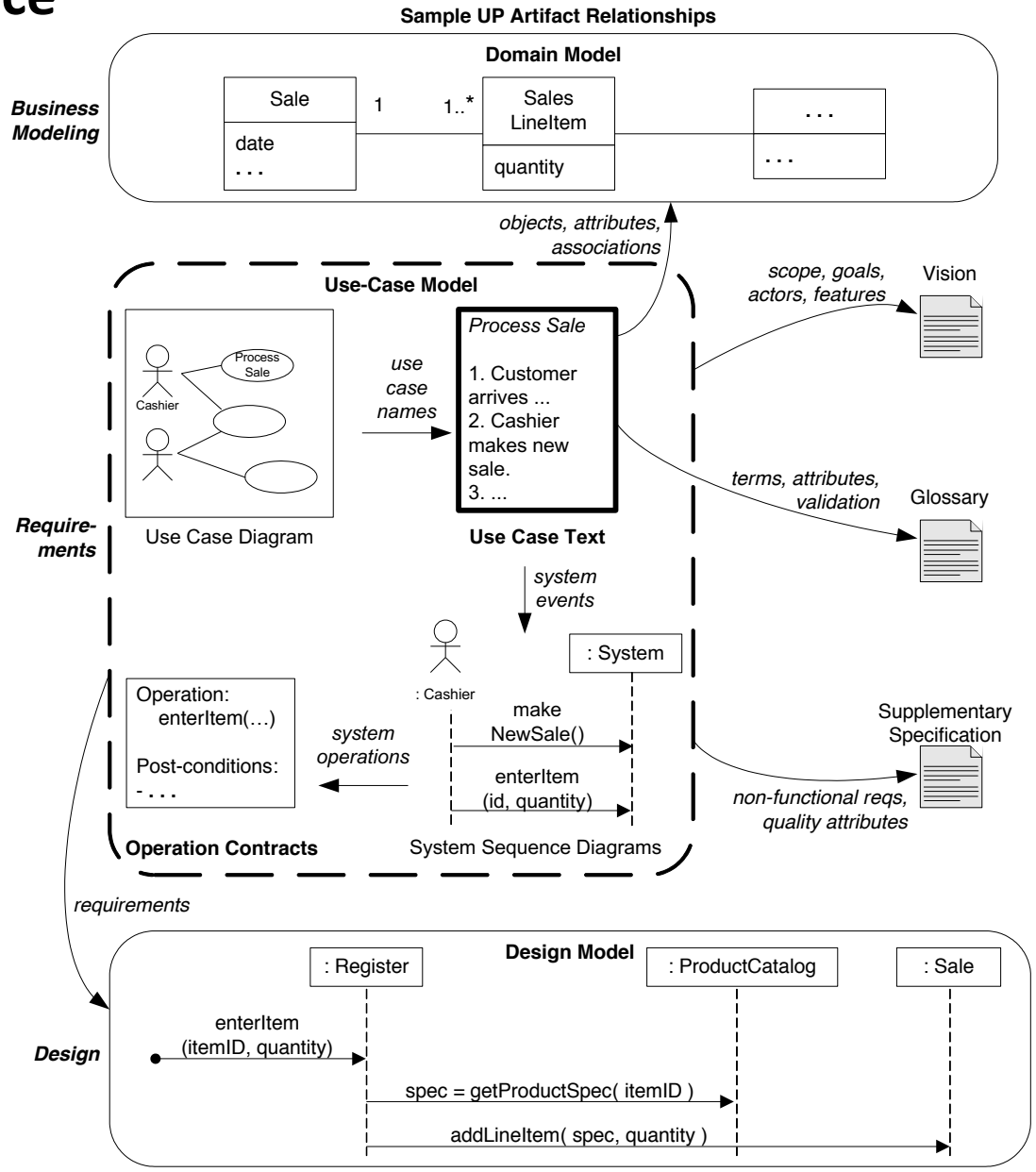
Chapter 6

Use Cases

UP Artifact Influence



Use Cases

- Use Cases:
  - Text stories
    - Some "actor" using system to achieve a goal
  - Used to discover and record requirements
  - Serve as input to later phases of development
  - Emphasizes user goals and perspective
    - Not system implementation choices and details
  - A way of specifying functional requirements
- Example: Process sale (Brief, informal format)

> **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.

Use Cases

- Use cases are text, not diagrams
  - UML use case diagrams are auxiliary
- Definitions:
  - Actor: Something with behavior
    - A person (user)
    - A computer system
    - An organization
  - Scenario: A specific sequence of actions and interactions between actors and the system
    - Also called a “use case instance”
    - One particular story of using a system
    - One path through the use case
  - A use case: A collection of related success and failure scenarios that describe an actor using a system to achieve a goal.

Use Cases

– A use case (Informal format)

Handle Returns

Main Success Scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item ...

Alternate Scenarios:

If the credit authorization is reject, inform the customer and ask for an alternate payment method.

If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).

If the system detects failure to communicate with the external tax calculator system, ...

– Use case model:
Set of all written use cases + UML use case diagrams

Use Cases and the Use-Case Model

- The UP defines the Use-Case Model within the Requirements discipline. Primarily, this is the set of all written use cases; it is a model of the system's functionality and environment
- **Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams.**
- The Use-Case Model is not the only requirement artifact in the UP. There are also the Supplementary Specification, Glossary, Vision, and Business Rules. These are all useful for requirements analysis, but secondary at this point.
- There is nothing object-oriented about use cases; we're not doing OO analysis when writing them.
- Use cases are requirements, primarily functional (They emphasize F in FURPS+)

Motivation: Why Use Cases

- We have goals and want computers to help meet them.
- Capturing goals and involving users
  - Lack of user involvement in software project is among top reasons for project failures
- Emphasizes user goals and perspective
- A good way of keeping it simple

Actors

- An actor anything with behaviour, including SuD (system under discussion)
- Primary Actor – has user goals fulfilled through using services of the SuD, for example cachier
- Supporting Actor – provides service (or information), often a computer system
- Offstage Actor – has an interest in the behaviour, but is not primary, for example tax agency

Use Case Formats

- Three formats
  - **Brief** – one paragraph summary – during early requirement analysis to get a quick sense of subject and scope – takes only a few minutes
  - **Casual** – multiple paragraphs
  - **Fully dressed** (fully detailed, all steps, variations…) – after many uses-cases have been identified and written in brief format, then during the first requirement workshop a few (such as %10) of the critical and high-value uses-cases are written in detail.
- Fully-dressed example follows

Use Case Template (Cockburn)

Use Case Section	Comment
Use Case Name	Start with a verb.
Scope	The system under design.
Level	"user-goal" or "subfunction"
Primary Actor	Calls on the system to deliver its services.
Stakeholders and Interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, *and* worth telling the reader?
Success Guarantee	What must be true on successful completion, *and* worth telling the reader.
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related non-functional requirements.
Technology and Data Variations List	Varying I/O methods and data formats.
Frequency of Occurrence	Influences investigation, testing, and timing of implementation.
Miscellaneous	Such as open issues.

Use Case UC1: Process Sale

Primary Actor: Cashier
Stakeholders and Interests:
- Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer short ages are deducted from his/her salary.
- Salesperson: Wants sales commissions updated.
- Customer: Wants purchase and fast service with minimal effort. Wants proof of pur chase to support returns.
- Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server compo nents (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
- Government Tax Agencies: Want to collect tax from every sale. May be multiple agen cies, such as national, state, and county.
- Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

Main Success Scenario (or Basic Flow):

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

*a. At any time, System fails:

To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

1. Cashier restarts System, logs in, and requests recovery of prior state.
2. System reconstructs prior state.

2a. System detects anomalies preventing recovery:

1. System signals error to the Cashier, records the error, and enters a clean state.
2. Cashier starts a new sale.

3a. Invalid identifier:

1. System signals error and rejects entry. 3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):

1. Cashier can enter item category identifier and the quantity.

3-6a: Customer asks Cashier to remove an item from the purchase:

1. Cashier enters item identifier for removal from sale.
2. System displays updated running total.

Extensions (or Alternative Flows):

3-6b. Customer tells Cashier to cancel sale:

1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:

1. System records sale so that it is available for retrieval on any POS terminal. 4a. The system generated item price is not wanted (e.g., Customer complained about something and is offered a lower price):
   1. Cashier enters override price.
   2. System presents new price.

5a. System detects failure to communicate with external tax calculation system service:

1. System restarts the service on the POS node, and continues. 1a. System detects that the service does not restart.
   1. System signals error.
   2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):

1. Cashier signals discount request.
2. Cashier enters Customer identification.
3. System presents discount total, based on discount rules.

Extensions (or Alternative Flows):

5c. Customer says they have credit in their account, to apply to the sale:
1. Cashier signals credit request.
2. Cashier enters Customer identification.
3. Systems applies credit up to price=0, and reduces remaining credit.

6a. Customer says they intended to pay by cash but don't have enough cash:
1a. Customer uses an alternate payment method.
1b. Customer tells Cashier to cancel sale. Cashier cancels sale on System.

7a. Paying by cash:
1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

Extensions (or Alternative Flows):

7b. Paying by credit:

1. Customer enters their credit account information.
2. System sends payment authorization request to an external Payment Authoriz tion Service System, and requests payment approval.
   2a. System detects failure to collaborate with external system:
      1. System signals error to Cashier.
      2. Cashier asks Customer for alternate payment.
3. System receives payment approval and signals approval to Cashier.
   3a. System receives payment denial:
      1. System signals denial to Cashier.
      2. Cashier asks Customer for alternate payment.
4. System records the credit payment, which includes the payment approval.
5. System presents credit payment signature input mechanism.
6. Cashier asks Customer for a credit payment signature. Customer enters signa ture.

Extensions (or Alternative Flows):

7c. Paying by check...
7d. Paying by debit...
7e. Customer presents coupons:
    1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
    1a. Coupon entered is not for any purchased item:
        1. System signals error to Cashier. 9a.
There are product rebates:
    1. System presents the rebate forms and rebate receipts for each item with a rebate.
9b. Customer requests gift receipt (no prices visible): 1.
    Cashier requests gift receipt and System presents it.

Special Requirements:

- Touch screen UI on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Somehow, we want robust recovery when access to remote services such the inven tory system is failing.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 3 and 7.

Technology and Data Variations List:

3a. Item identifier entered by bar code laser scanner (if bar code is present) or key-board.

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 pre-dict many customers will want digital signature capture.

Frequency of Occurrence: Could be nearly continuous.

Open Issues:

- What are the tax law variations?
- Explore the remote service recovery issue.
- What customization is needed for different businesses?
- Must a cashier take their cash drawer when they log out?
- Can the customer directly use the card reader, or does the cashier have to do it?

Sections of a Use Case

- Scope: Bounds system under design
- Level: User-goal or subfunction
  - User goal: Buy some stuff and pay
  - Subfunction: Cashier authenticates himself
    - Duplicated steps factored out
- Primary actor
- Stakeholders and interests list:
  - Functional and non-functional requirements come from these

Sections of a Use Case

- Pre-conditions:
  - Conditions that must hold before use case is started
  - Noteworthy assumptions
  - NOT TESTED within the use case
  - Example: Logging in successfully completed
- Post-conditions (success guarantees)
  - What is guaranteed to be true upon successful termination of the use case
  - Should meet all needs of all stakeholders

Sections of a Use Case

- Main success scenario
  - Or Basic Flow, Happy path
  - Defer conditional and branching statements to the Extensions section
  - Records steps of success path
  - A step:
    - An interaction between actors
    - A validation (usually by the system)
    - A state change by the system
      - Example: Recording or modifying something
  - Some sequences of steps may be repeated. See POS example.

Sections of a Use Case

- Extensions (Alternate Flows)
  - Normally comprise majority of text
  - Other scenarios or branches
  - Can be both success and failure
  - An extension of Step 3 of basic flow is labeled by 3a, 3b, ...
  - An extension that can take place at any step is labelled *a, *b
  - An extension has two parts
    1. The condition: Something that can be detected by the system or the actor
    2. The handling: Response to the alternative condition
  - At the end of extension, execution merges with basic flow unless indicated otherwise
  - Sometimes one use case scenario can perform a branch and lead to another use case scenario.
    - Show with underlines
- Special requirements
- Technology and data variations list

Some Guidelines

- Keep user interface (UI) details out of the use case
  - Example: Log in vs. authenticate oneself
- Focus on “intent”. Do not specify a mechanism.
- Write terse use cases. Don’t need to use full sentences.
- Write “black-box” use cases. Do not specify internal workings of system, components, design yet.
  - These come later. Do not make design decisions during requirements specification.

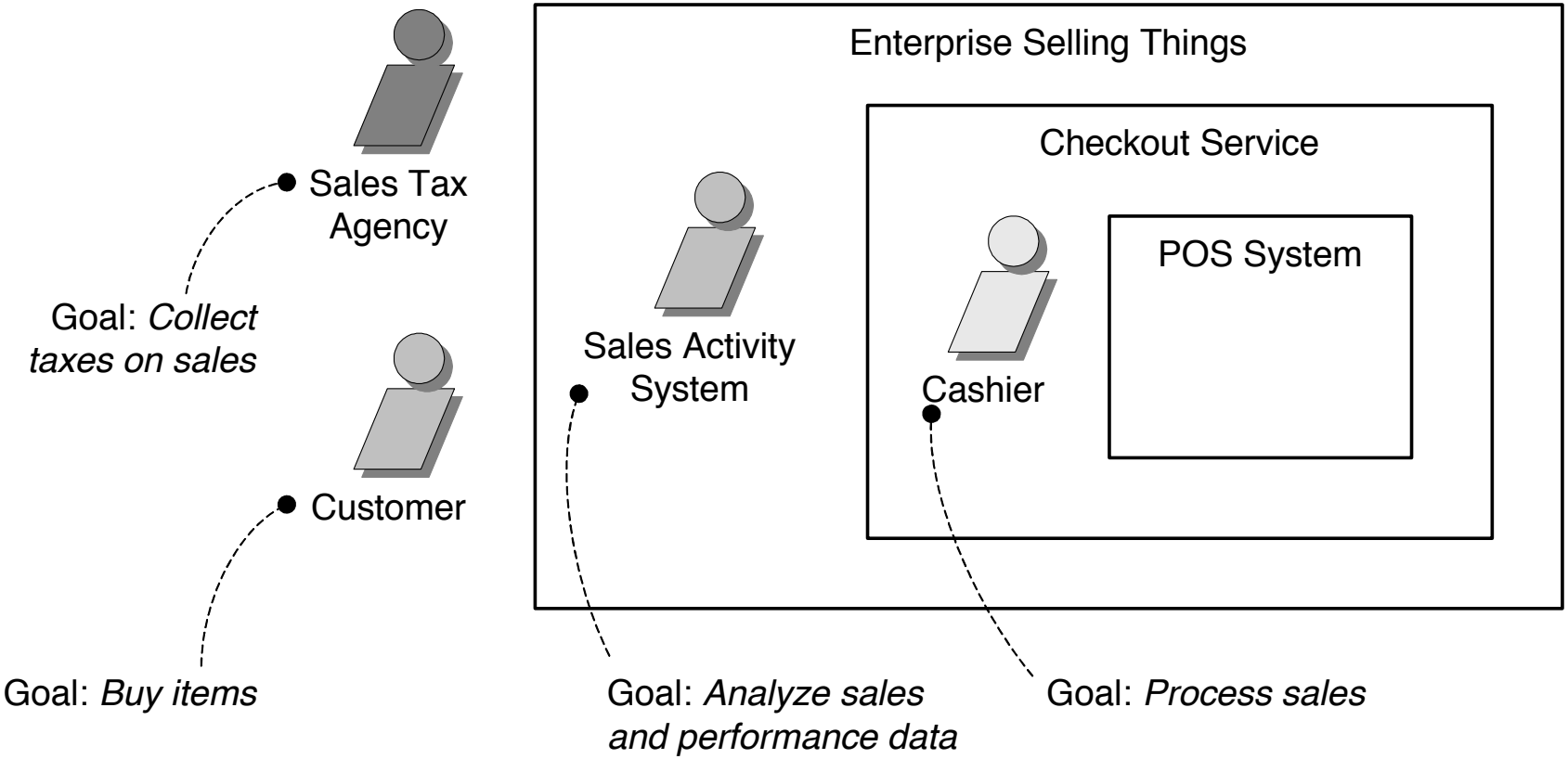
How to find use cases?

- Choose the system boundary
  - Just a software application?
  - Hardware and application as a unit?
  - Entire organization?
- Find primary actors and goals
  - Look for nouns and verbs in informal descriptions

Actor Goals vs Use Cases

- Actors have goals and use applications to satisfy them,
- Rather than asking “what are the tasks”, one starts by asking “who uses the system an what are their goals”
- The name of a use case for a goal should reflect its name

Fig. 6.2


Enterprise Selling Things
Checkout Service
POS System
Sales Tax Agency
Goal: *Collect taxes on sales*
Customer
Goal: *Buy items*
Sales Activity System
Goal: *Analyze sales and performance data*
Cashier
Goal: *Process sales*

Is the cashier or Customer the Primary Actor in the use case Process Sale

- Depends on the system boundary
  - If the enterprise and checkout service is viewed as an aggregate system, the customer is the primary actor
  - However, from the viewpoint of just POS system; the system services the goal of of a chashier to process the sale

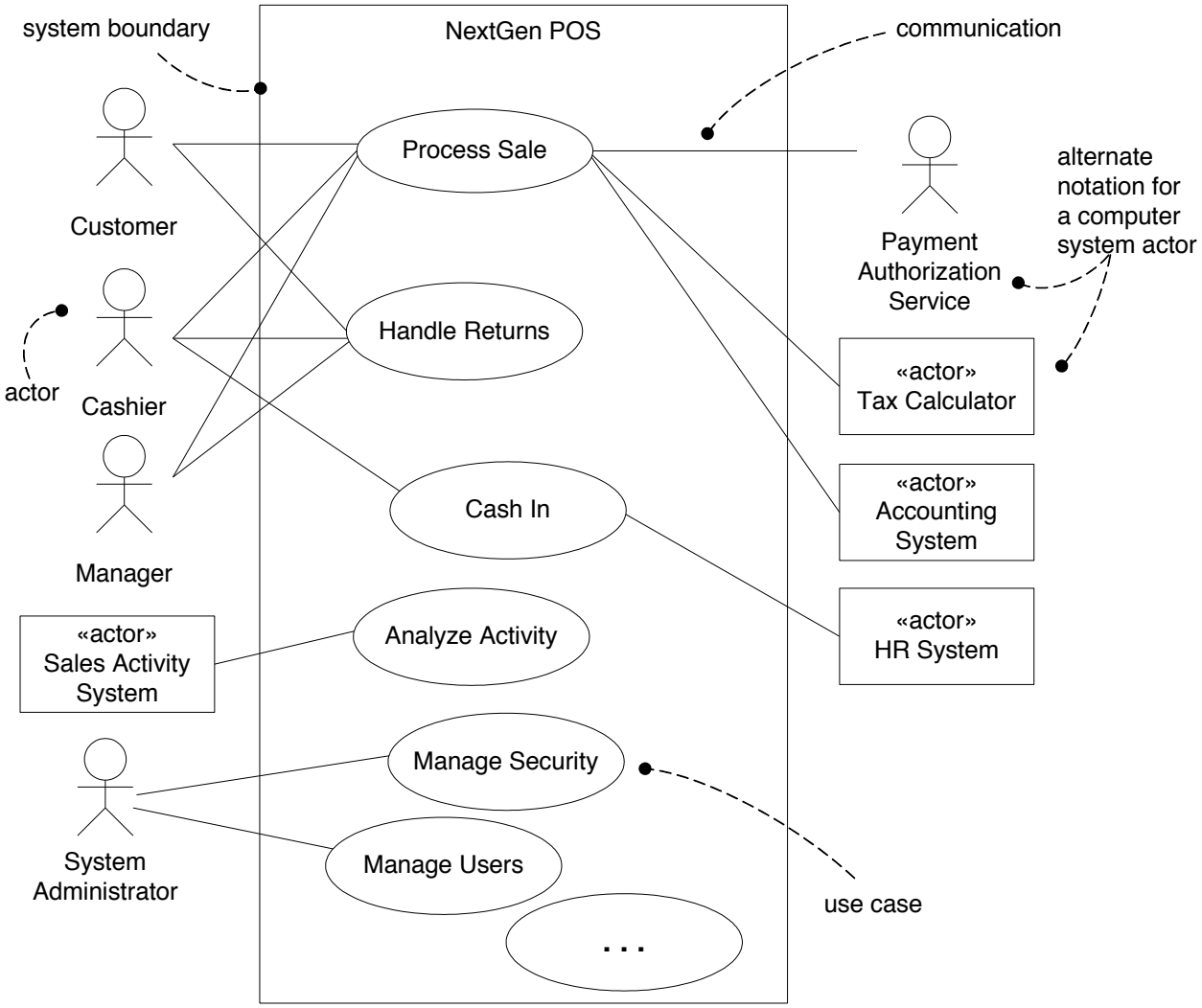
In general

- Define one use case for each user goal
- Name the use case similar to the user goal
- Start the name of use cases with a verb
  - Process Sale
- What tests can help find useful use cases
  - Boss Test: Your boss asks “what have you doing all day?” You reply “Logging in”. Is your boss happy?
  - Size Test: usually contains many steps

Common Mistakes

- Scope of use case too large or too small.
- Examples:
  - Negotiate a supplier contract
  - Handle returns
  - Log in
  - Move piece on game board
- Rules of thumb for deciding use case scope
  - The boss test:
    - Level of task should be an item of business you can report to a boss
  - The elementary business process (EBP) test:
    - A task performed by one person in one place at one time in response to a business event which adds measurable business value and leaves the data in a consistent state.
  - The size test:
    - 3-10 pages
  - Exceptions:
    - Subfunctions, e.g. “paying by credit”

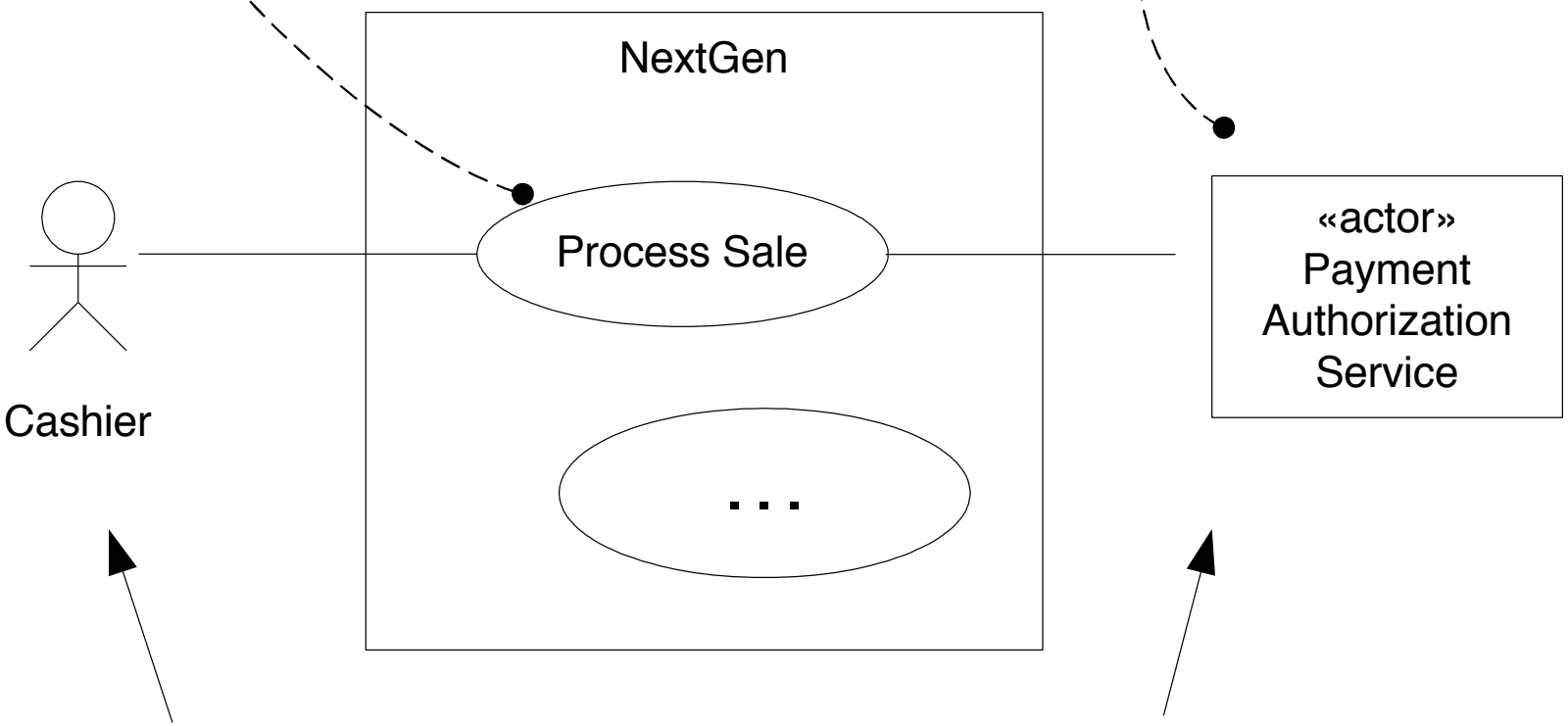
UML Use Case Diagrams



UML Use Case Diagrams

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

UML Use Case Diagrams: Alternative Actor Notations

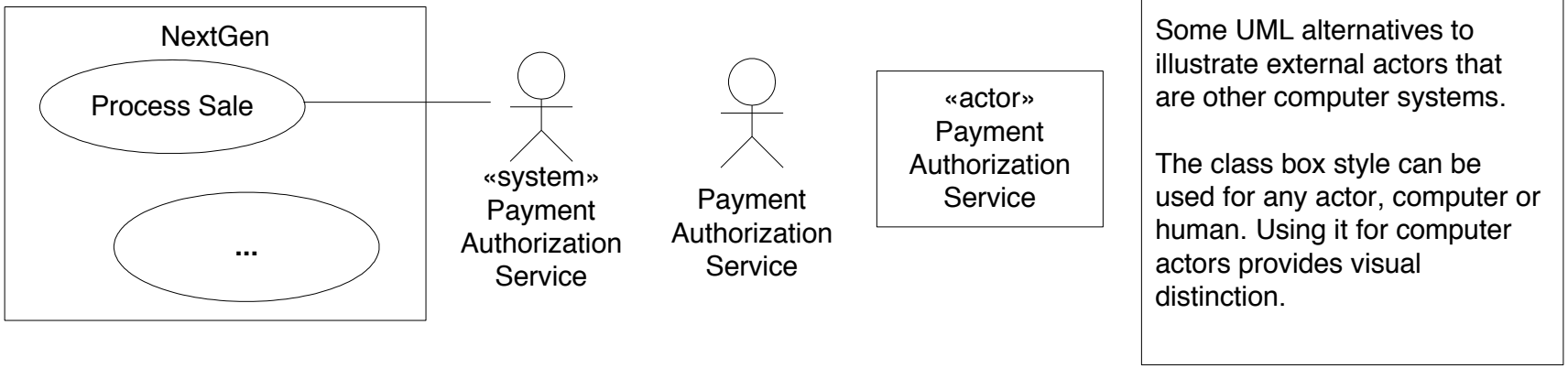


Fig. 6.6

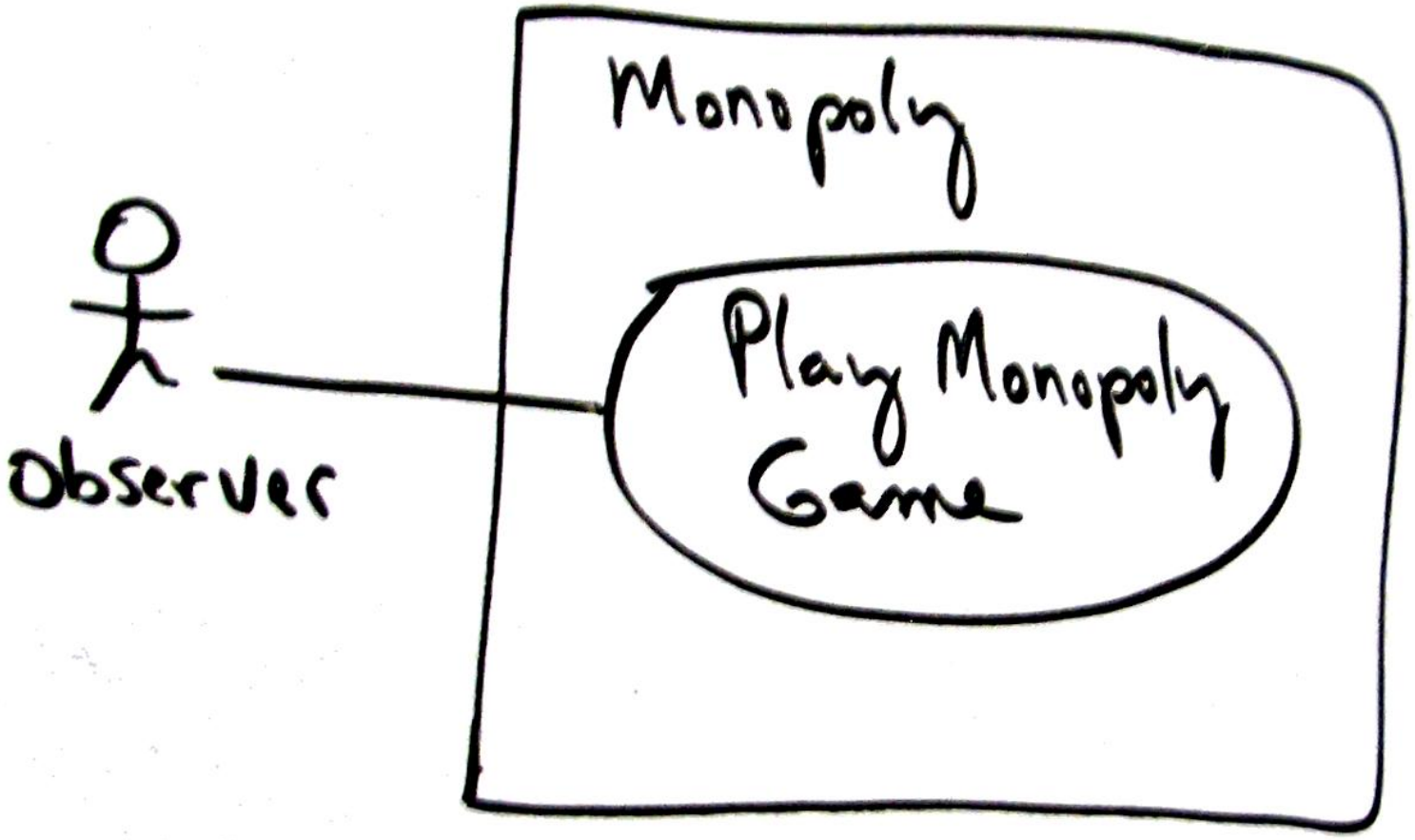
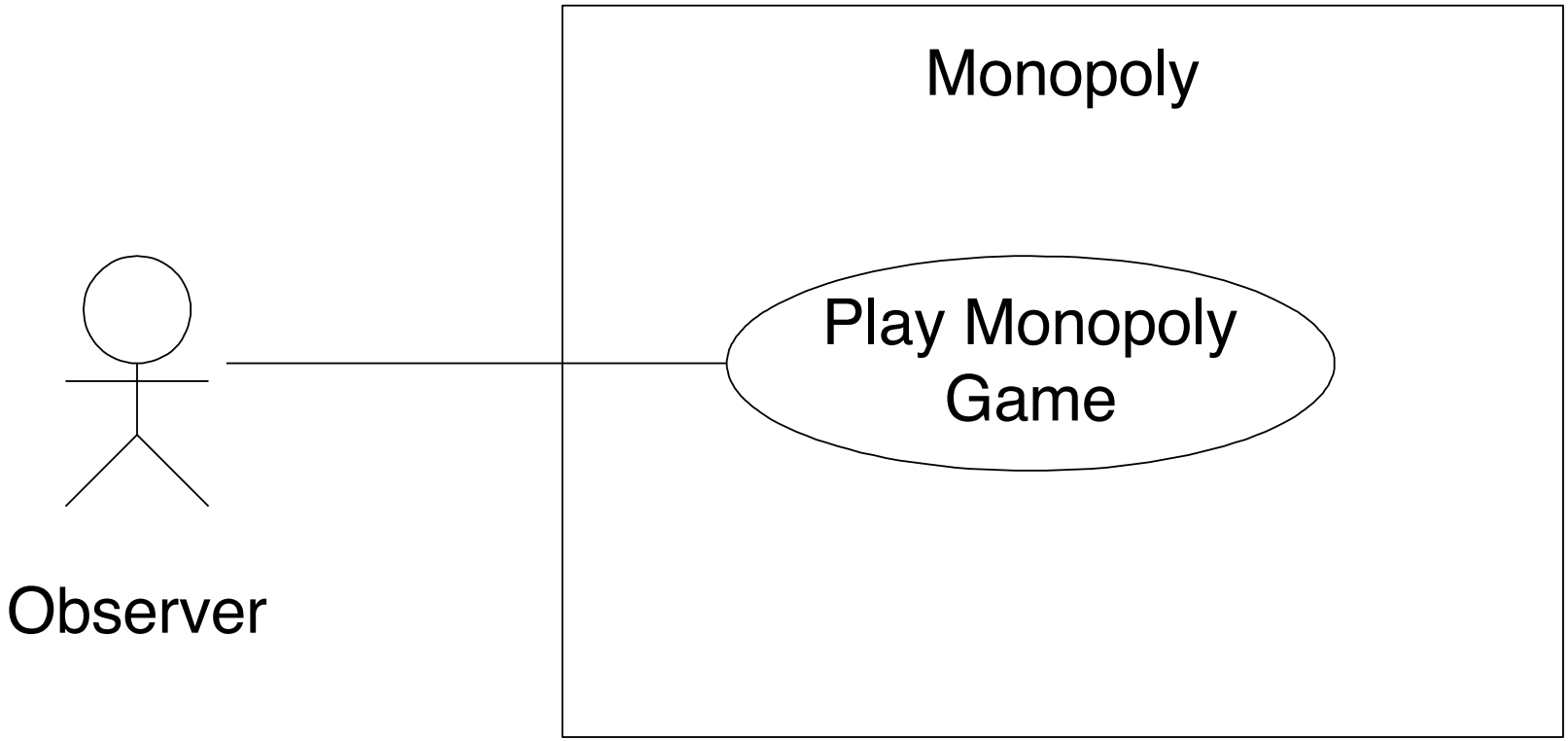


Fig. 6.6



- Observer: Wants to easily observe the output of the game simulation.

Main Success Scenario:

1. Observer requests new game initialization, enters number of players.
2. Observer starts play.
3. System displays game trace for next player move (see domain rules, and "game trace" in glossary for trace details).

Repeat step 3 until a winner or Observer cancels.

Extensions:

*a. At any time, System fails:

(To support recovery, System logs after each completed move)

1. Observer restarts System.

Monopoly Game

- Very simple use case diagram and text
- Most detail is deferred to Supplementary Specification (SS)
  - Domain rules listed in SS

Use Cases in Iterative Methods

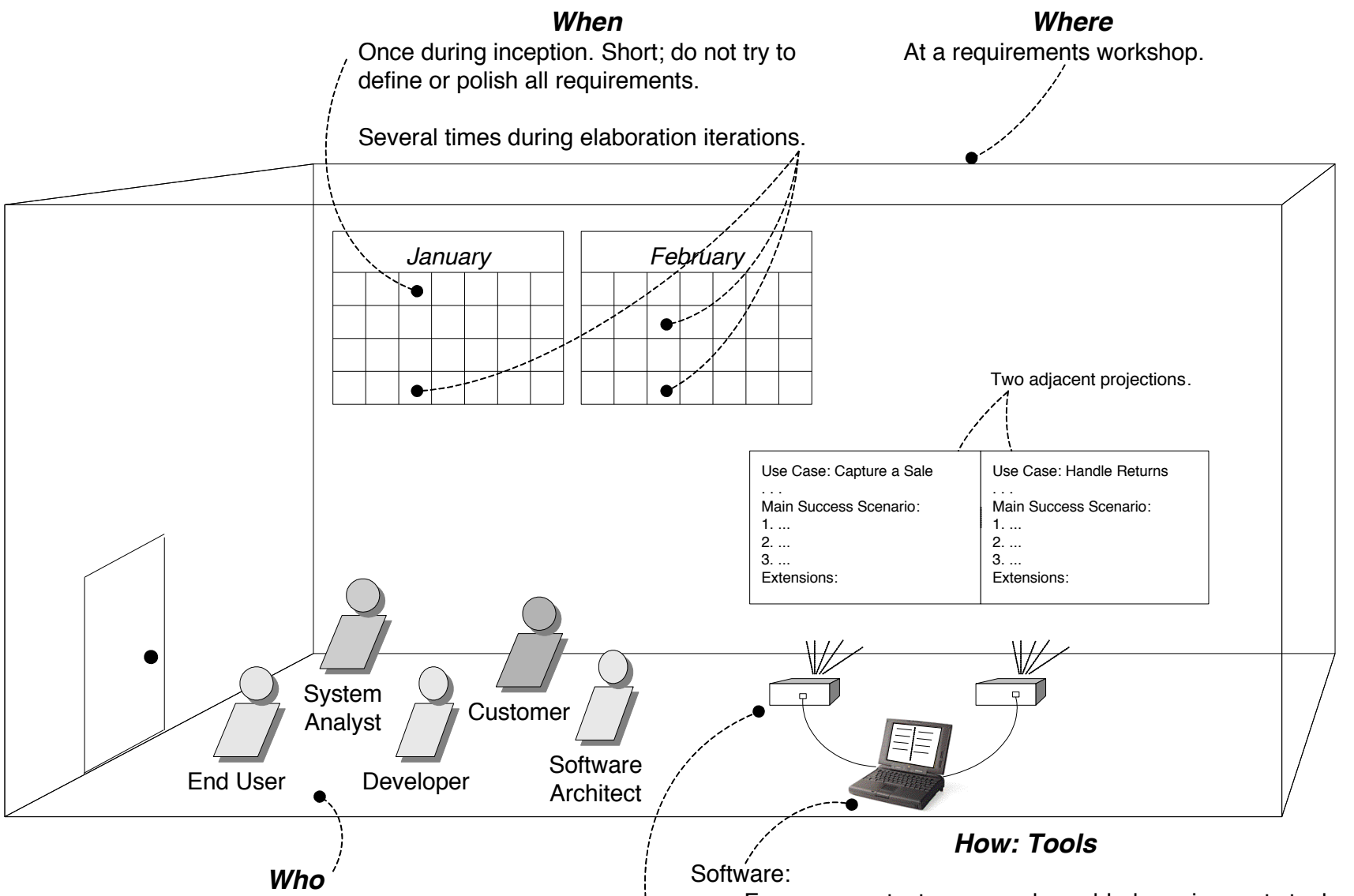
- Functional requirements recorded in use cases
- Important part of iterative planning
  - Work of an iteration decided by choosing use case scenarios or entire use cases
- Use cases influence organization of user manuals
- Testing builds on use case scenarios
- Iterative interplay between
  - Requirements discovery
  - Building parts of system

Table 6.1. Sample requirements effort across the early iterations; this is not a recipe.

Discipline	Artifact	Comments and Level of Requirements Effort				
		Incep 1 week	Elab 1 4 weeks	Elab 2 4 weeks	Elab 3 3 weeks	Elab 4 3 weeks
Requirements	Use-Case Model	2-day requirements workshop. Most use cases identified by name, and summarized in a short paragraph. Pick 10% from the high-level list to analyze and write in detail. This 10% will be the most architecturally important, risky, and high-business value.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 30% of the use cases in detail.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 50% of the use cases in detail.	Repeat, complete 70% of all use cases in detail.	Repeat with the goal of 8090% of the use cases clarified and written in detail. Only a small portion of these have been built in elaboration; the remainder are done in construction.
Design	Design Model	none	Design for a small set of high-risk architecturally significant requirements.	repeat	repeat	Repeat. The high risk and architecturally significant aspects should now be stabilized.
Implementation	Implementation Model (code, etc.)	none	Implement these.	Repeat. 5% of the final system is built.	Repeat. 10% of the final system is built.	Repeat. 15% of the final system is built.
Project Management	SW Development Plan	Very vague estimate of total effort.	Estimate starts to take shape.	a little better...	a little better...	Overall project duration, major milestones, effort, and cost estimates can now be rationally committed to.

Table 6.2. Sample UP artifacts and timing. s - start; r - refine

Discipline	Artifact	Incep.	Elab.	Const.	Trans.
	Iteration →	I1	E1..En	C1..Cn	T1..T2
Business Modeling	Domain Model		s		
Requirements	***Use-Case Model***	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		



Many, including *end users* and *developers,* will play the role of *requirements specifier*, helping to write use cases.

Led by *system analyst* who is responsible for requirements definition.

Software:

For use case text, use a web-enabled requirements tool that integrates with a popular word processor.

For use case diagrams, a UML CASE tool.

Hyperlink the use cases; present them on the project website.

Hardware: Use two projectors attached to dual video cards and set the display width double to improve the spaciousness of the drawing area or display 2 adjacent word processor windows .