An Introduction to Object-Oriented Analysis and Design

PART I: INCEPTION

Dr. 邱 明
Software School of Xiamen University
mingqiu@xmu.edu.cn
Fall, 2009

Chapter 4: Inception is Not the Requirements Phase

Le mieux est l'ennemi du bien
(The best is the enemy of the good).
Voltaire

4.1. What is Inception?

- The initial step of a project
  - The vision and business case for this project
  - Feasible
  - Buy and/or build
  - Rough unreliable range of cost:
    - Is it $10K, 100K or in the millions?
  - Proceed or stop?
- The purpose of the inception phase is
  - not to define all the requirements,
  - or generate a believable estimate or project plan.

4.1. What is Inception?

- Inception in one sentence:
  - Envision the product scope, vision, and business case.
- The main problem solved in one sentence:
  - Do the stakeholders have basic agreement on the vision of the project, and is it worth investing in serious investigation?

4.1. What is Inception?

An analogy In the oil business,

1. Decide if there is enough evidence or a business case to even justify exploratory drilling.

2. If so, do measurements and exploratory drilling.

3. Provide scope and estimate information.

4. Further steps…

4.2. How Long is Inception?

- Especially brief.
  - If the project will definitely be done, and it is clearly feasible
  - It includes
    - first requirements workshop
    - planning for the first iteration

4.3. What Artifacts May Start in Inception?

Artifact	Comment
Vision and Business Case	Describes the high-level goals and constraints, the business case, and provides an executive summary.
Use-Case Model	Describes the functional requirements. During inception, the names of most use cases will be identified, and perhaps 10% of the use cases will be analyzed in detail.
Supplementary Specification	Describes other requirements, mostly non-functional. During inception, it is useful to have some idea of the key non-functional requirements that have will have a major impact on the architecture.
Glossary	Key domain terminology, and data dictionary.
Risk List & Risk Management Plan	Describes the risks (business, technical, resource, schedule) and ideas for their mitigation or response.
Prototypes & proof-of-concepts	To clarify the vision, and validate technical ideas.
Iteration Plan	Describes what to do in the first elaboration iteration.
Phase Plan & Software Development Plan	Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

4.3. What Artifacts May Start in Inception?

- A Lot of Documentation !!!
  - Choose to create only those that really add value for the project,
  - Drop them if their worth is not proved.
  - Since this is evolutionary development
    - Only initial, rough documents is to be created during this phase
    - They are refined during the elaboration iterations

4.3. What Artifacts May Start in Inception?

- Agile Modeling perspective
  - the greatest value of modeling is to improve understanding, rather than to document reliable specifications

In preparing for battle I have always found that plans are useless, but planning indispensable.

General Eisenhower

4.4. You Know You Didn't Understand Inception When...

- More than "a few" weeks long for most projects.
- Attempt to define most of the requirements.
- Estimates or plans are expected to be reliable.
- Define the architecture.
- Believe that the proper sequence of work
    1. define the requirements;
    2. design the architecture;
    3. implement.
- No Business Case or Vision artifact.
- All the use cases were written in detail.
- None of the use cases were written in detail;

4.5. How Much UML During Inception?

- The purpose of inception is
  - to collect just enough information to establish a common vision,
  - to decide
    - if moving forward is feasible,
    - if the project is worth serious investigation in the elaboration phase

Most UML diagramming will occur in the next phase elaboration.

Chapter 5: Evolutionary Requirements

Ours is a world where people don't know what they want and are willing to go through hell to get it.

Don Marquis

Objective

- Motivate doing evolutionary requirements.
- Define the FURPS+ model.
- Define the UP requirements artifacts

5.1. Definition: Requirements

- Requirements
  - capabilities and conditions to which the system/project must conform
- Manage requirements in UP
  - A systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system

5.2. Evolutionary vs. Waterfall Requirements

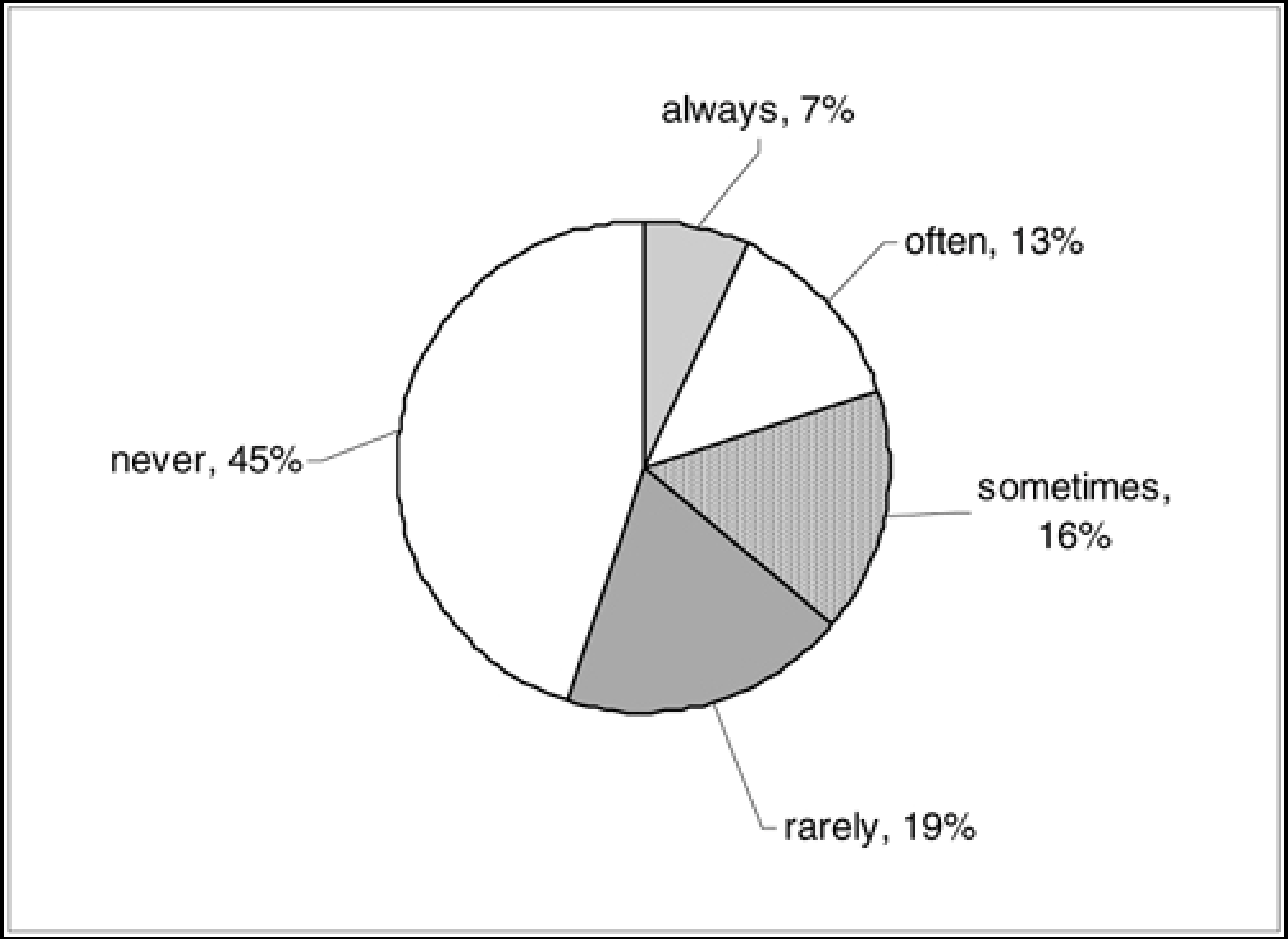
- Unified Process
  - Embracing change in requirements
    - is incredibly important and at the heart of waterfall versus iterative and evolutionary thinking.
- Waterfall practices
  - view a software project as similar to predictable mass manufacturing,
    - low change rates.
- Software is in the domain of new product development,
  - high change ranges
  - high degrees of novelty and discovery.

5.2. Evolutionary vs. Waterfall Requirements

- Some Statistical results
  - Study failure factors on 1027 projects[1]
    - Waterfall practices are the largest contributing factor for failure (82%)

1. Thomas, M. 2001. IT Projects Sink or Swim. British Computer Society Review.

5.2. Evolutionary vs. Waterfall Requirements



Actual use of waterfall-specified features.

Johnson, J. 2002. ROIIt's Your Job, XP 2002, Sardinia, Italy.

5.3. What are Skillful Means to Find Requirements?

- Systematic approach to handling the changing requirements
  - Writing use cases with customers
  - Requirements workshops that include both developers and customers
  - Focus groups with proxy customers
  - A demo of the results of each iteration to solicit customer’s feedback

5.4. What are the Types and Categories of Requirements?

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

5.4. What are the Types and Categories of Requirements?

- "+" in FURPS+
  - 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.

5.5. How are Requirements Organized in UP Artifacts?

- UP offers several requirements artifacts
  - Use-Case Model
    - A set of typical scenarios of using a system. (functional requirements)
  - Supplementary Specification
    - Everything not in the use cases. (non-functional requirements),
  - Glossary
    - defines noteworthy terms.
  - Vision Summarizes
    - high-level requirements, summarizes the business case for the project.
  - Business Rules
    - describe requirements or policies that transcend one software project (ex. government tax laws).

Questions & Answers

Chapter 6: Use Cases

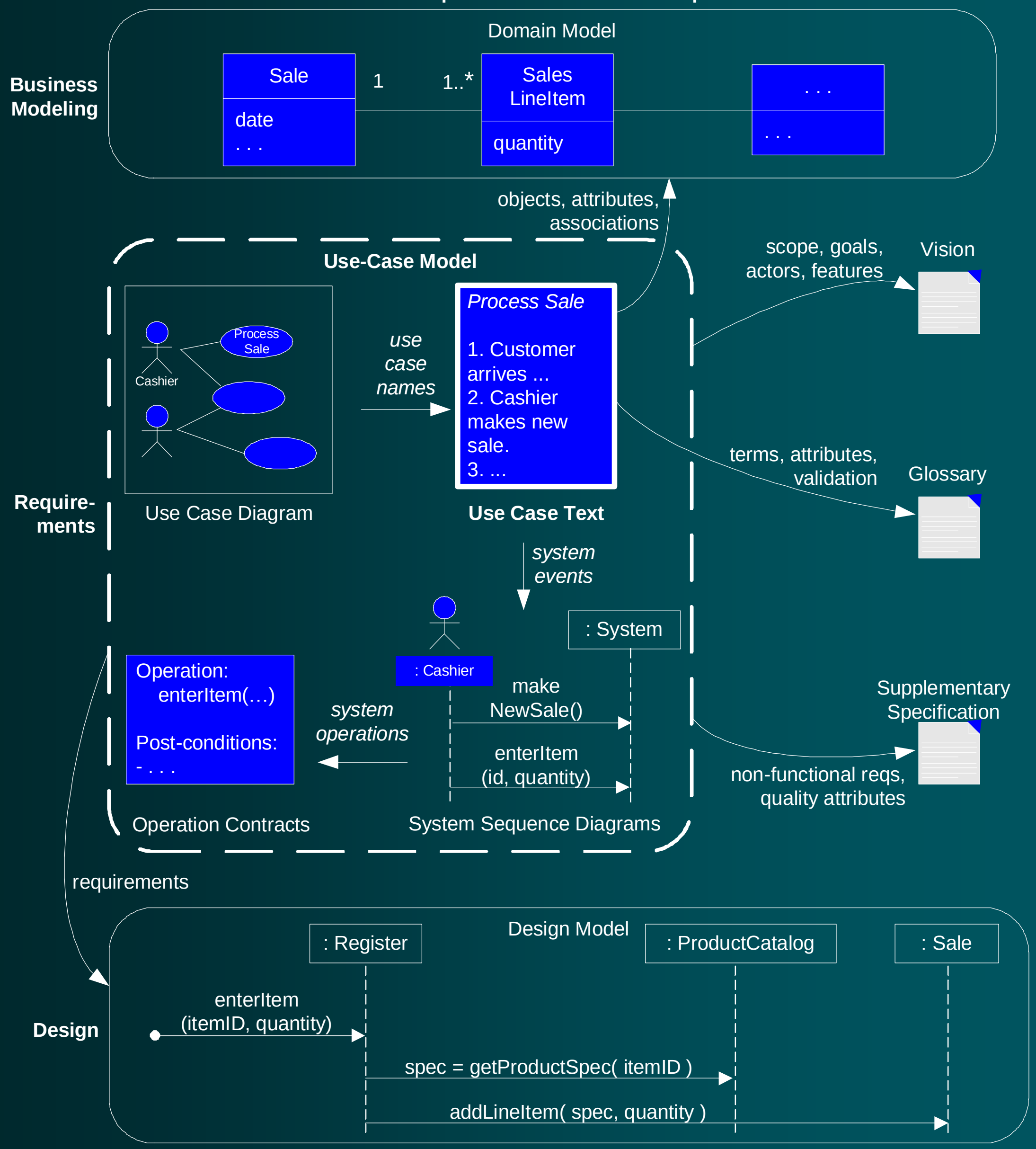
The indispensable first step to getting the things you want out of life: decide what you want.

Ben Stein

Objective

- Identify and write use cases.
- Use the brief, casual, and fully dressed formats, in an essential style.
- Apply tests to identify suitable use cases.
- Relate use case analysis to iterative development

Sample UP Artifact Relationships



6.1 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.

6.2. What are Actors, Scenarios, and Use Cases?

- Actor
  - something with behavior,
    - such as a person, computer system, or organization;
      - for example, a cashier
- Scenario (Use Case Instance)
  - a specific sequence of actions and interactions between actors and the system
    - one particular story of using a system, or one path through the use case
      - for example, the scenario of successfully purchasing items with cash,
      - the scenario of failing to purchase items because of a credit payment denial.

6.2. What are Actors, Scenarios, and Use Cases?

- Use Case
  - a collection of related success and failure scenarios that describe an actor using a system to support a goal.

Scenarios

A set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor

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 customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.

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 accounting system, …

6.3. Use Cases and the Use-Case Model

Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams.

- Use-Case Model may optionally include a UML use case diagram
  - to show the names of use cases and actors, and their relationships

6.4. Motivation: Why Use Cases?

The best are simple and familiar.

- Use cases are a good way to help keep it simple, and make it familiar to domain experts.
  - emphasize the user goals and perspective

6.5. Are Use Cases Functional Requirements?

- Use cases are requirements,
  - primarily functional or behavioral requirements that indicate what the system will do.

6.6. What are Three Kinds of Actors?

- Primary actor
  - has user goals fulfilled through using services of the SuD (System under Discussion)
    - the cashier.
- Supporting actor
  - provides a service to the SuD.
    - the automated payment authorization service
- Offstage actor
  - has an interest in the behavior of the use case, but is not primary or supporting;
    - a government tax agency.

6.7. What are Three Common Use Case Formats?

- Brief
  - terse one-paragraph summary, usually of the main success scenario.
    - Process Sale
- Casual
  - informal paragraph format. Multiple paragraphs that cover various scenarios.
    - Handle Returns
- Fully dressed
  - all steps and variations are written in detail, and there are supporting sections

6.8. Example: Process Sale, Fully Dressed Style

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.

6.9. What do the Sections Mean?

- Preface Elements
  - **Scope: NextGen POS Application**
    - bounds the system (or systems) under design
  - Level: User goal
    - user-goal
      - common kind that describe the scenarios to fulfill the goals of a primary actor to get work done
    - subfunction-level
      - substeps required to support a user goal(duplicate substeps shared by several regular use cases).

6.9. What do the Sections Mean?

- **Primary Actor: Cashier**
  - The principal actor that calls upon system services to fulfill a goal.
- Stakeholders and Interests List
  - remind us what the more detailed responsibilities of the system should be.

Stakeholders and Interests List (Example)

- Cashier
  - Wants accurate, fast entry, and no payment errors.
  - Cash drawer shortages are deducted from his/her salary.
- Salesperson
  - Wants sales commissions updated.
- Customer
  - Wants purchase and fast service with minimal effort.
  - Wants easily visible display of entered items and prices.
  - Wants proof of purchase 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 components are unavailable.
  - Wants automatic and fast update of accounting and inventory.

Stakeholders and Interests List (Example)

- **Manager**
  - **Wants to be able to quickly perform override operations,**
  - **Easily debug Cashier problems.**
- **Government Tax Agencies**
  - **Want to collect tax from every sale.**
- **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.**

6.9. What do the Sections Mean?

- **Preconditions and Success Guarantees**
  - Precondition: Cashier is identified and authenticated
    - **state before a scenario is begun**
  - Success guarantees
    - state on successful completion
    - Examples
      - Sale is saved.
      - Tax is correctly calculated.
      - Accounting and Inventory are updated.
      - Commissions recorded.
      - Receipt is generated.
      - Payment authorization approvals are recorded.

6.9. What do the Sections Mean?

- Main Success Scenario
  - Describes a typical success path that satisfies the interests of the stakeholders.
  - Records three types of steps
    1. An interaction between actors
    2. A validation.
    3. A state change by the system.

Main Success Scenario(Example)

1. Customer arrives at POS checkout with goods (services).
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item, presents item description, price, and running total. Price calculated (price rule).
5. Cashier repeats steps 3-4 until indicates done.
6. System presents total with taxes calculated.
7. Cashier tells Customer the total, asks for payment.
8. Customer pays and System handles payment.
9. System logs completed sale and sends sale&payment information to the Accounting system/Inventory system.
10. System presents receipt.
11. Customer leaves with receipt and goods.

6.9. What do the Sections Mean?

- Extensions
  - indicate all the other scenarios or branches, both success and failure
  - notated with respect to its steps 1…N.
  - two parts: the condition and the handling.
    - condition can be detected by the system or an actor.
  - At the end of extension handling, by default the scenario merges back with the main success scenario

6.9. What do the Sections Mean?

- Special Requirements
  - non-functional requirement, quality attribute, or constraint
  - Example
    - 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.
    - Want robust recovery when access to remote services.
    - Language internationalization on the text displayed.
    - Pluggable business rules to be insertable at steps 3 and 7.

6.9. What do the Sections Mean?

- Technology and Data Variations List
  - Example
    - *a. Manager override entered by swiping an override card through a card reader, or entering an authorization code via the keyboard.
    - 3a. Item identifier entered by bar code 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.

6.10. A Two-Column Variation

- Emphasizes the interaction between the actors and the system.
  - Examples (in book)

6.11-6.14. Guidelines:

- Write in an Essential UI-Free Style
  - Keep the user interface out; focus on intent
    - Example: Log in
- Write Terse Use Cases
- Write Black-Box Use Cases
  - Specify what the system must do without deciding how it will do it.
- Take an Actor and Actor-Goal Perspective
  - Write requirements focusing on the users or actors, asking about their goals and typical situations.
  - Focus on understanding what the actor considers a valuable result

6.15. Guideline: How to Find Use Cases

- Step 1:Choose the System Boundary
  - Define the actor first, then the boundary becomes clearer.
- **Step 2,3: Find Primary Actors and Goals**
  - **Brainstorm the primary actors first, as this sets up the framework for further investigation.**

6.15. Guideline: How to Find Use Cases

Questions to Help Find Actors and Goals

Who starts and stops the system?	Who does system administration?
Who does user and security management?	Is "time" an actor because the system does something in response to a time event?
Is there a monitoring process that restarts the system if it fails?	Who evaluates system activity or performance?
How are software updates handled? Push or pull update?	Who evaluates logs? Are they remotely retrieved?
In addition to human primary actors, are there any external software or robotic systems that call upon services of the system?	Who gets notified when there are errors or failures?

6.15. Guideline: How to Find Use Cases

- How to Organize the Actors and Goals?
  - As you discover the results, draw them in a use case diagram, naming the goals as use cases.
  - Write an actor-goal list first, review and refine it, and then draw the use case diagram.

6.15. Guideline: How to Find Use Cases

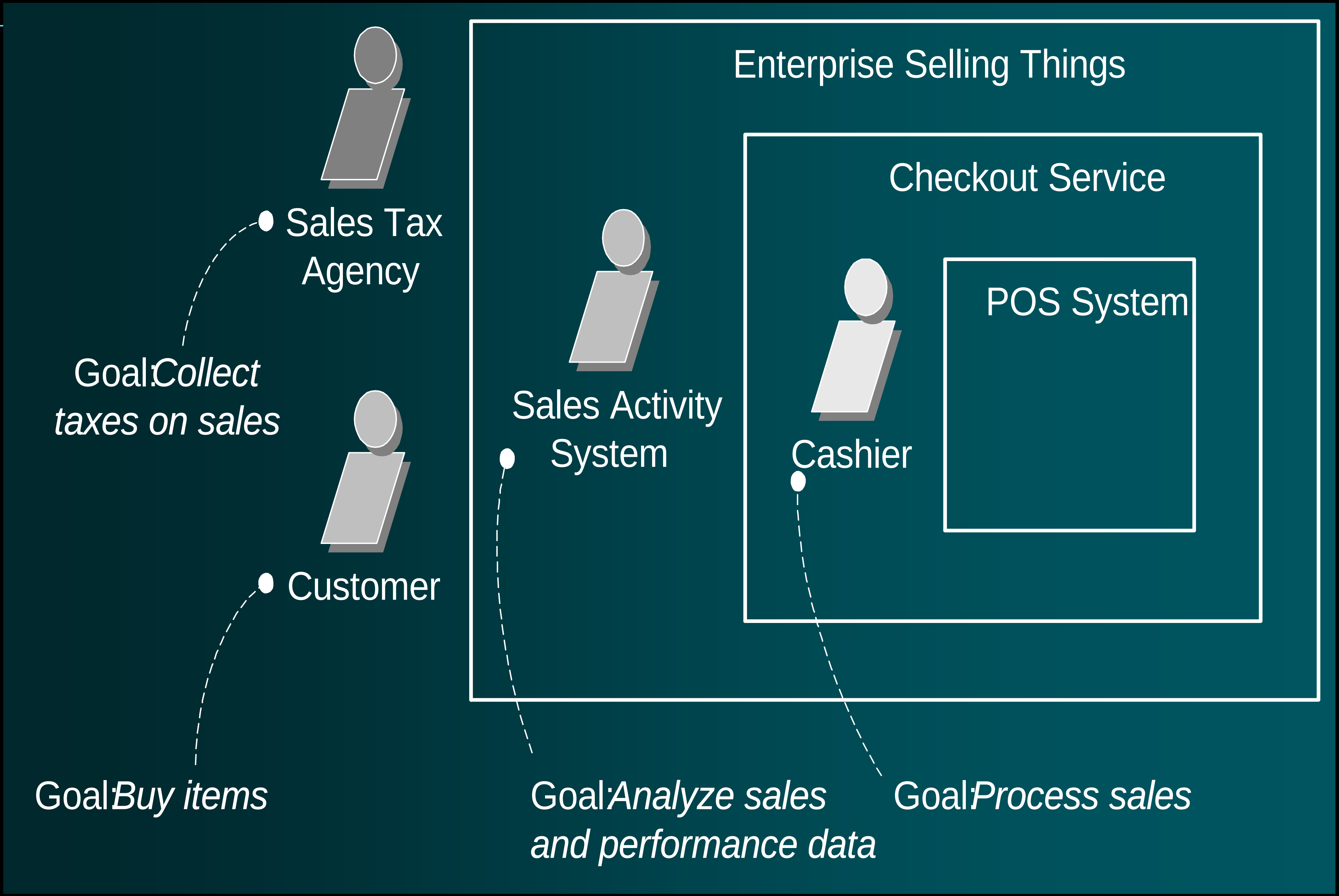
Actor	Goal	Actor	Goal
Cashier	process sale	System Administrator	Add users
	Process rentals		Modify users
	Handle returns		Delete users
	Cash in		Manage security
	Cash out		Manage system tables
	…..		…..

6.15. Guideline: How to Find Use Cases

- Why Ask About Actor Goals Rather Than Use Cases?
  - The viewpoint of use case modeling is
    - to find these actors and their goals,
    - to create solutions that produce a result of value.
  - Two questions in requirements workshop
    - "What do you do?"
    - "What are your goals whose results have measurable value?"

6.15. Guideline: How to Find Use Cases

- Is the Cashier or Customer the Primary Actor?
  - The system was designed for the cashier,
    - to quickly process a sale, look up prices, etc
  - not for the customer.
- Event Analysis -- Other Ways to Find Actors and Goals
  - identify external events.
    - What are they, where from, and why?


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*

6.15. Guideline: How to Find Use Cases

- Step 4: Define Use Cases
  - 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

6.16. What Tests Can Help Find Useful Use Cases?

- Which of these is a valid use case?
  - Negotiate a Supplier Contract
  - Handle Returns
  - Log In
  - Move Piece on Game Board

6.16. What Tests Can Help Find Useful Use Cases?

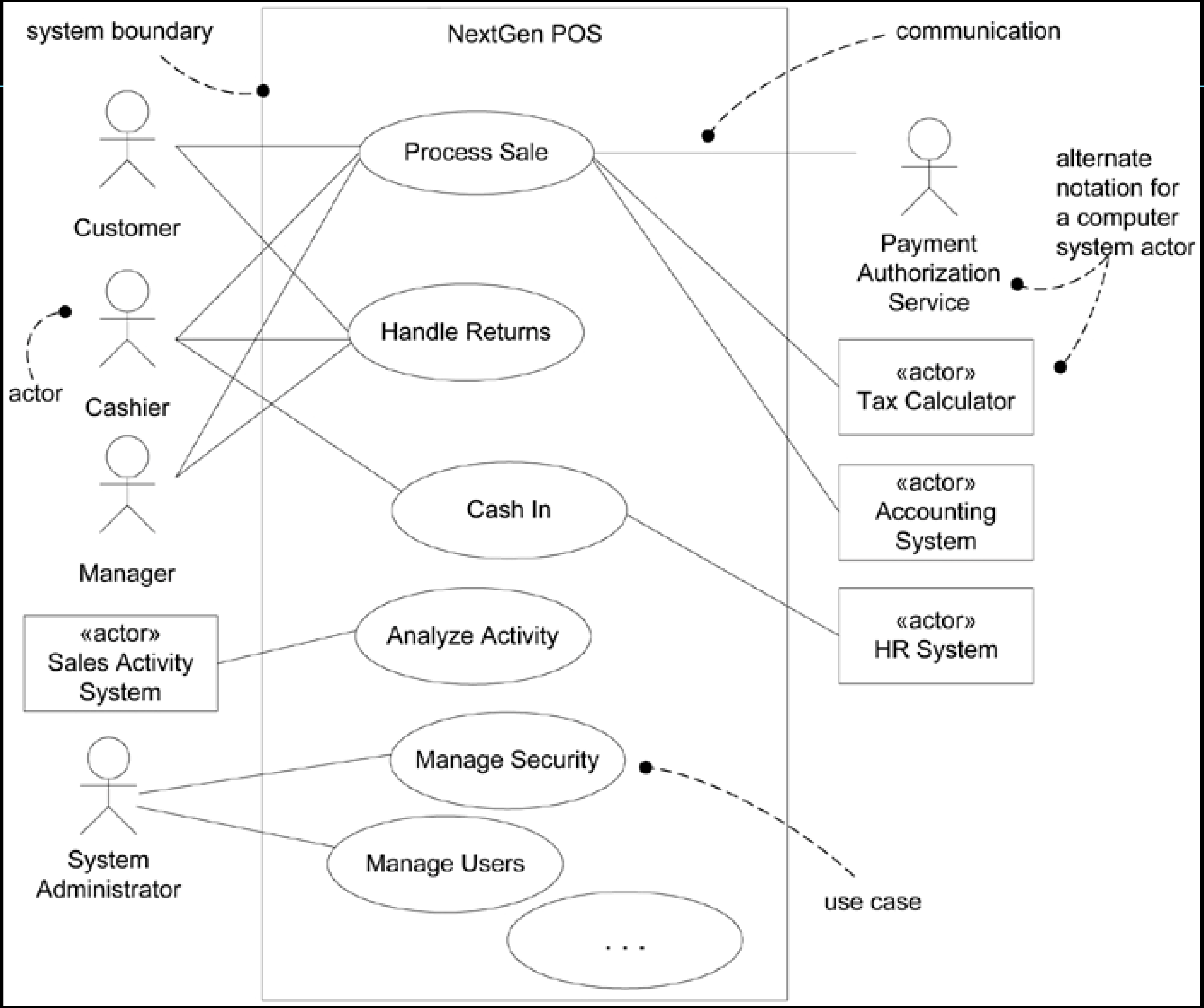
- The Boss Test
  - The use case should be strongly related to achieving results of measurable value
- The Elementary Business Process (EBP) Test
  - EBP
    - A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state.
- The Size Test
  - A use case typically contains many steps

6.16. What Tests Can Help Find Useful Use Cases?

- Applying the Tests
  - Negotiate a Supplier Contract
    - Much broader and longer than an EBP.
  - Handle Returns
    - OK with the boss. Seems like an EBP. Size is good.
  - Log In
    - Boss not happy if this is all you do all day!
  - Move Piece on Game Board
    - Single step. fails the size test.
- Reasonable Violations of the Tests
  - Example: “paying by credit”

6.17. Applying UML: Use Case Diagrams

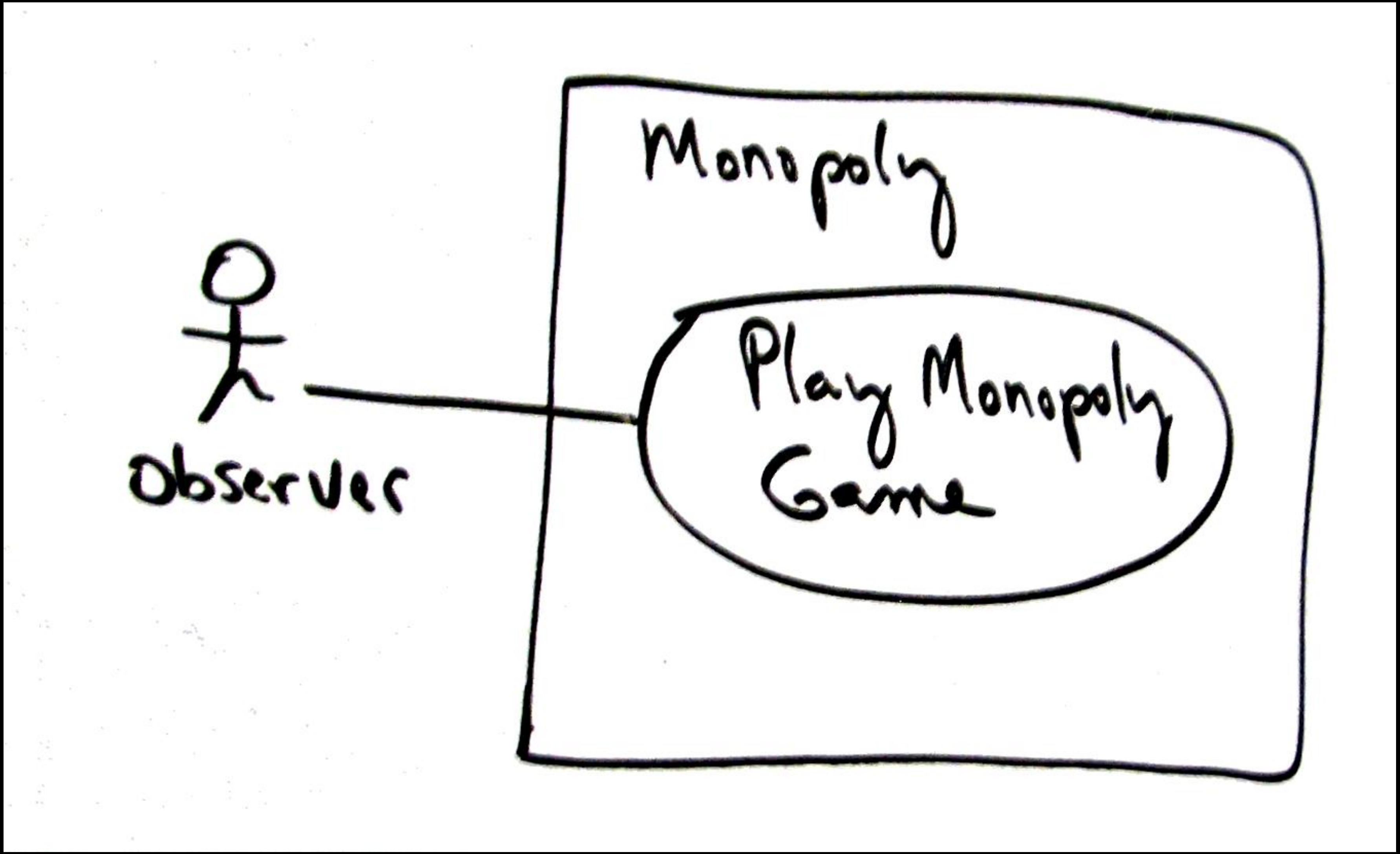
- use case diagram notation
  - illustrate the names of use cases and actors, and the relationships between them.
  - A common sign of a novice use case modeler is
    - a preoccupation with use case diagrams and use case relationships, rather than writing text.
- Downplay Diagramming, Keep it Short and Simple


system boundary
NextGen POS
communication
Customer
Process Sale
Payment Authorization Service
alternate notation for a computer system actor
actor
Cashier
Handle Returns
«actor» Tax Calculator
Manager
Cash In
«actor» Accounting System
«actor» Sales Activity System
Analyze Activity
«actor» HR System
Manage Security
System Administrator
Manage Users
use case
. . .

6.19. Motivation: Requirements in Context

- A motivation for use cases is
  - focusing on who the key actors are, their goals, and common tasks
- Use cases organize a set of requirements in the context of the typical scenarios of using a system.
  - improves cohesion and comprehension
- High-Level System Feature Lists Are Acceptable

6.20. Example: Monopoly Game



6.21. How to Work With Use Cases in Iterative Methods?

- Use-case driven development
  - Functional requirements are primarily recorded in use cases.
  - Use cases are an important part of iterative planning. They are a key input to estimation.
  - Use-case realizations drive the design.
  - Use cases often influence the organization of user manuals.
  - Functional or system testing corresponds to the scenarios of use cases.
  - UI "wizards" or shortcuts may be created for the most common scenarios of important use cases to ease common tasks.

6.21. How to Work With Use Cases in Iterative Methods?

- How to Evolve Use Cases and Other Specifications Across the Iterations?
  - See Table 6.1 in book
- When Should Various UP Artifact be Created?

6.21. How to Work With Use Cases in Iterative Methods?

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		
	Data Model		s	r	

6.21. How to Work With Use Cases in Iterative Methods?

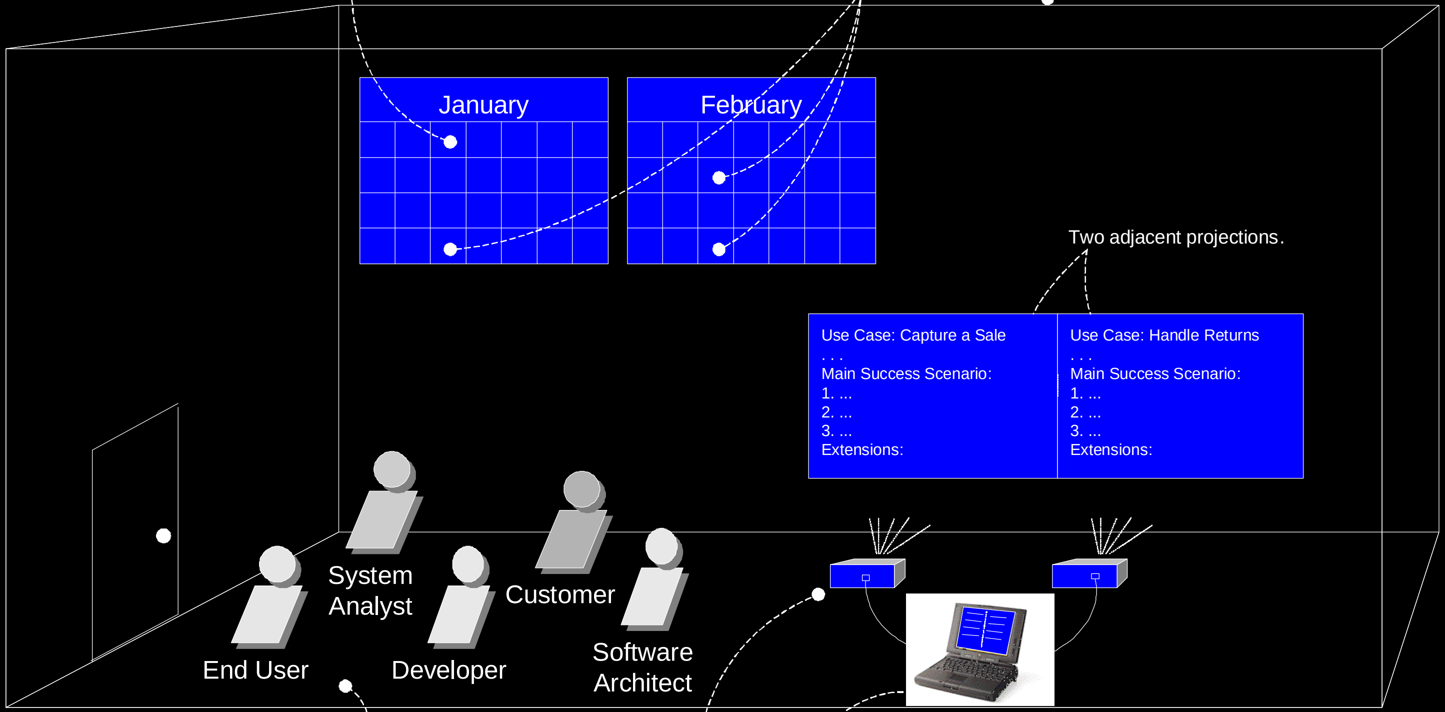
When

Once during inception. Short; do not try to define or polish all requirements.

Several times during elaboration iterations.

Where

At a requirements workshop.



Who

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.

How: Tools

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 .

6.21. How to Work With Use Cases in Iterative Methods?

- How to Write Use Cases in Inception?
  - 2-day req. workshop during the early investigation
    - identify goals and stakeholders,
    - speculate what is in and out of scope of the project
    - Write An actor-goal-use case table
    - Start to draw A use case context diagram
      - use cases are identified by name
    - Most of the interesting, complex, or risky use cases are written in brief format.
    - 10% to 20% of the use cases are rewritten in a fully dressed format

6.21. How to Work With Use Cases in Iterative Methods?

- How to Write Use Cases in Elaboration?
  - Some parts of the system are incrementally built in the timeboxed iterations
  - two-day requirements workshop in each iteration.
    - most important use cases are investigated first
  - More of the use cases are written, and rewritten, in their fully dressed format.

6.21. How to Work With Use Cases in Iterative Methods?

- How to Write Use Cases in Construction?
- Case Study: Use Cases in the NextGen Inception Phase

Fully Dressed	Casual	Brief
Process Sale Handle Returns	Process Rental Analyze Sales Activity Manage Security …	Cash In Cash Out Manage Users Start Up Shut Down Manage System Tables …