So Far …

Part 1: OOAD Intro

- SE
- OOAD
- UML
- Iterative, Evolutionary, and Agile
- Case Studies

Part 2: Inception

- Inception is Not the Requirements Phase
- Evolutionary Requirements
- Use Cases
- Other Requirements

Part 3: Elaboration—Iteration 1

- Iteration 1—Basics
- Domain Models
- ….

UP Main Vocabulary Terms

UP Phases

UP Disciplines

UP Artifacts (Models)

UP Phases

Inception

Approximate vision, business case, scope, vague estimates

Elaboration

Refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates

Construction

Iterative implementation of the remaining lower risk and easier elements, and preparation for deployment

Transition

Beta tests, deployment

Inception

Abdulkareem Alali

ACK Dale Haverstock

Based on Larman’s Applying UML and Patterns Book, 3d

The best (perfect) is the enemy of the good

—Voltaire

Inception —Introduction

Inception is

- Not the requirements phase
- Short initial step to establish a common vision
- and basic scope for the project

Inception —Introduction

Inception will include:

- Analysis of perhaps 10-20% of the use cases
- Analysis of the critical non-functional requirements
- The creation of a business case (justification for a proposed project)
- The preparation of the development environment to allow programming to begin in the elaboration phase

What is Inception?
Answer & explore Questions:

- What is the vision and the business case for the project?
- Is the project feasible?
- Should we buy and/or build?
- What is a rough (and unreliable) order of magnitude range of the cost: Is it $10K, $100K, in the millions, ...?
- Should we proceed or stop?

Some requirements exploration will need to be done to address the Qs

Inception and Requirements

Purpose of Inception is NOT to define all requirements

Elaboration, next phase is where requirements, costs, etc. are flushed out

Only enough investigation should be done so that:

- A rational, justifiable opinion of the overall purpose may be formed
- The feasibility of the potential new system is decided
- Whether or not it is worthwhile to invest in deeper exploration

Inception —Analogy!

In oil business, a new field is being considered, some of the steps would include:

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…

Analogy!

The inception phase is like step one in this analogy

In step one people do not predict how much oil, cost or effort is needed to extract it. Would be nice, won't be realistic

In UP terms, the realistic exploration step is the **Elaboration Phase**

Inception phase is akin to a feasibility study to decide if it is even worth investing in exploratory drilling

An **analogy** is more or less like marriage, it helps you understand life via projection, and once you do, you divorce it, because it won’t make any sense much further!

—AA

Software Engineering is the process of slowing down development, too much is waterfall, a thoughtful slow down is agile!

—AA

Inception Briefly

Inception's purpose in one sentence:

"Envision the product scope, vision, and business case." [Larman]

Inception's 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?" [Larman]

Inception only provides an order-of-magnitude sense of the level of effort, to aid the decision to continue or not

Thus estimates are not to be considered reliable in the Inception phase

How Long is Inception?

Depending on the project Inception may be one week, or a few weeks

The answers to the Inception questions may be clear if similar projects have been done before

Artifacts That 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 and 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.

Inception Artifact Notes

Note that the artifacts are only begun during Inception

Use-case model may list 10-20% use cases at most are deeply investigated

All artifacts are optional, only artifacts that improve understanding or add value should be done

Greatest value of Agile modeling is to improve understanding, not to document reliable specifications

UML During Inception

Typically, there is not much UML used during Inception

There may be some use-case diagrams

Use-cases themselves are text-oriented

Inception has been misunderstood, if:

- There is an attempt to define/analyze most of the requirements
- Many/all the use cases were written in detail
- Estimates or plans are expected to be reliable
- There is no Business Case or Vision artifact

Inception has been misunderstood if:

- For most projects, if it is more than "a few" weeks long
- None of the use cases were written in detail; rather, ~10-20% written in detail to obtain some realistic insight into the scope of the problem

Inception is like a **feasibility** study,

Elaboration is where the more realistic **exploration** is done

Evolutionary Requirements

Abdulkareem Alali

ACK Dale Haverstock

Based on Larman’s Applying UML and Patterns Book, 3d

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

—Don Marquis

Requirements

Capabilities and conditions that the product must **provide** and **meet**

Managing requirements is a best practice for project managers

Requirement issues are the leading cause of project failure

Even if you do a perfect job of building the wrong thing, its no good!

Not Waterfall Requirements

There is an attempt in the waterfall method to describe the requirements fully and accurately and “freeze” them

Unified process realizes that change is constant, so plans for change instead of setting an impossible goal

On average, 25% of the requirements change on software projects

UP embrace change in requirements as a fundamental driver on projects

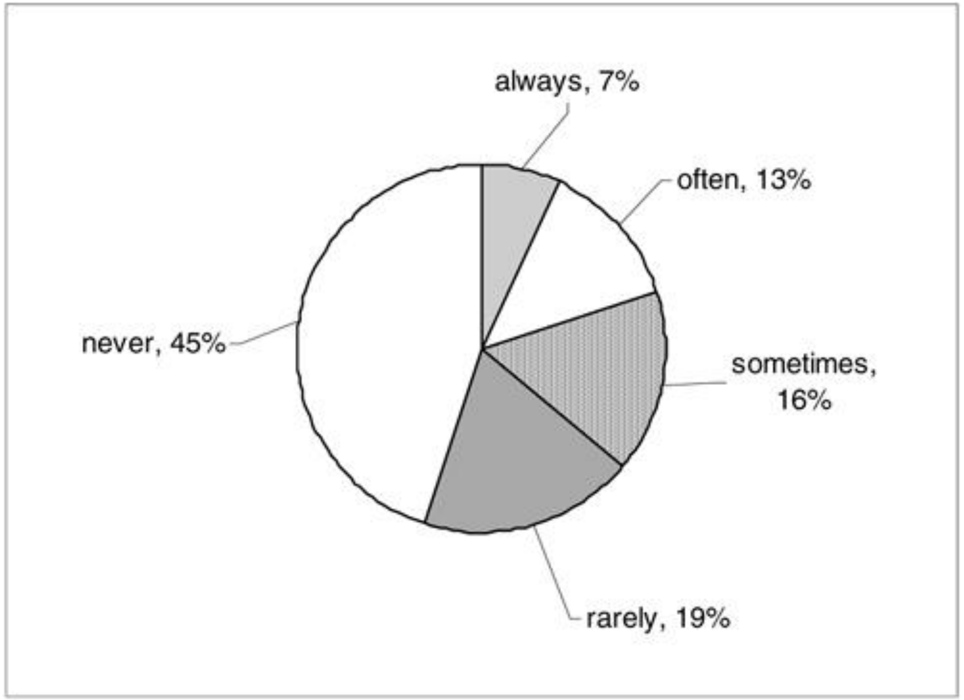
Evolutionary vs. Waterfall Requirements

With UP production-quality programming and testing are begun long before most of the requirements have been analyzed or specified

10% or 20% of the

most architecturally significant, risky,

and high-business-value requirements are specified initially



Actual use of waterfall-specified features

Agile vs. Iterative vs. Evolutionary

Large overlap between agile/iterative/evolutionary processes such that they can sometimes be considered equivalent

Agile is a process that embraces the **12 agile manifesto** .

Iterative is a process **decomposes software development into a series of iterations** for an **incremental** product development and growth .

Evolutionary is a process that **incrementally develops and completes a project**, may not consist of iterations (e.g., spiral models)

Functional & Non-Functional Requirements .

Functional requirement describes what a software system should do (behavior of the system) .

- System must send an email whenever an order is placed, a customer signs up, etc.

Non-functional requirements place constraints on how the system will do so .

- Emails should be sent with a latency of no greater than 12 hours from such an activity

Manage Requirements

Besides **changing**, the word **finding** is important; that is, the UP encourages skillful elicitation via techniques such as:

- Writing use cases with customers,
- Requirements workshops that include both developers and customers,
- Focus groups with proxy customers, and
- A demo of the results of each iteration to the customers, to solicit feedback
- Feedback helps finding and changing requirements

Unified Process —FURPS+

FURPS+ model used by the UP has the categories of requirements given below, developed by HP .

F stands for functional requirements, **URPS** are nonfunctional requirements (quality requirements)

+ group are constraints or pseudo-requirements

FURPS+

Functional (features, capabilities)

Usability (human factors, help, documents)

Reliability (failures, recovery, predictable)

Performance (response, throughput)

Supportability (maintainability, configuration)

+ constraints (next slide)

Constraints

- **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

Terms

Usability: Level of user expertise assumed. User interface standards used. Documentation provided.

Reliability: Safety and security requirements. Availability, robustness, and reliability of the system. Exception handling, mean time between failures, error tolerance, Data loss tolerance

Performance: Number of concurrent users supported, response time, number of transactions per second

Supportability: How will system be extended? Who maintains the system?

Implementation: Platform?

Interfaces: Interfaces to existing systems. Protocols used.

Operation: Who manages the running system?

Packaging: Who installs the system? How many installations are there?

Legal: Licensing? Liability issues? Licensing fees or liabilities incurred from using third-party components or algorithms?

Requirements for Requirements

Requirements, Should be:

Complete— All possible scenarios (for the selected ~10-20%), including exceptions, are described

Consistent— There are no contradictions between any scenarios

Unambiguous— There is only one interpretation for any operation, event, etc.

Correct— The requirements accurately represent the system the client needs

Requirements for Requirements

Desirable:

Realism— The system can be implemented within constraints

Verifiability— It is possible to demonstrate that the system fulfills the requirements

Traceability— Each requirement can be traced through the system models and to the software that implements the requirement

Key UP Artifacts

Key UP artifacts include:

- **Use-Case Model**
- **Supplementary Specification**
- **Glossary**
- **Vision**
- **Business Rules**

All are optional and Create an artifact according to how beneficial it is

The Use-Case Model

The use case model is comprised mainly of use cases

The use case model gives primarily functional requirements,

other requirements may become apparent though

Supplementary Specification

Contains other specification not in the use cases

This artifact is primarily for all non-functional requirements, such as reliability or performance

Other record functional features not expressed (or expressible) as use cases may be included here

Vision

The vision provides a high-level view of the project’s goals

A short executive overview document for quickly learning the project's big ideas

Business Rules

Also called Domain Rules, typically describe requirements or policies that transcend one software project

Are required in the domain or business, and many applications may need to conform to them

An example is government tax laws

Placing them in a central Business Rules artifact makes for better reuse of the analysis effort

Glossary

Communication problems due to **terminology** must be avoided

Everyday word/term has a different or more specific meaning in an application domain

Creating a glossary is a good idea

Data Dictionary— records/requirements related to data, such as validation rules, acceptable values, and so forth.

Glossary: A glossary is an alphabetical list of technical terms in some specialized field of knowledge; usually published as an appendix to a text on that field. [WordNet]

Summary

Requirements are the capabilities and conditions to which the system must conform

Traditional approaches used to come up with feature lists, current approaches utilize **use cases** for requirements

Scenarios and use cases are used to determine the functional behavior of a system and can be validated by the user

Use cases and scenarios represent an external view of the system

Use cases stem from user **goals**. Find a user goal, write the use case

So Far …

Part 1: OOAD Intro

- SE
- OOAD
- UML
- Iterative, Evolutionary, and Agile
- Case Studies

Part 2: Inception

- Inception is Not the Requirements Phase
- Evolutionary Requirements
- Use Cases
- Other Requirements

Part 3: Elaboration—Iteration 1

- Iteration 1—Basics
- Domain Models
- ….

Use Cases

Abdulkareem Alali

ACK Dale Haverstock

Based on Larman’s Applying UML and Patterns Book, 3d

Use Cases

Use cases are **text stories** of some **actor** using a system to meet **goals** (Not diagrams, but text)

A **Use Case** is a collection of related success and failure scenarios to support a goal

A **scenario** is a specific **sequence** of actions and interactions between actors and the system; Use Case Instance

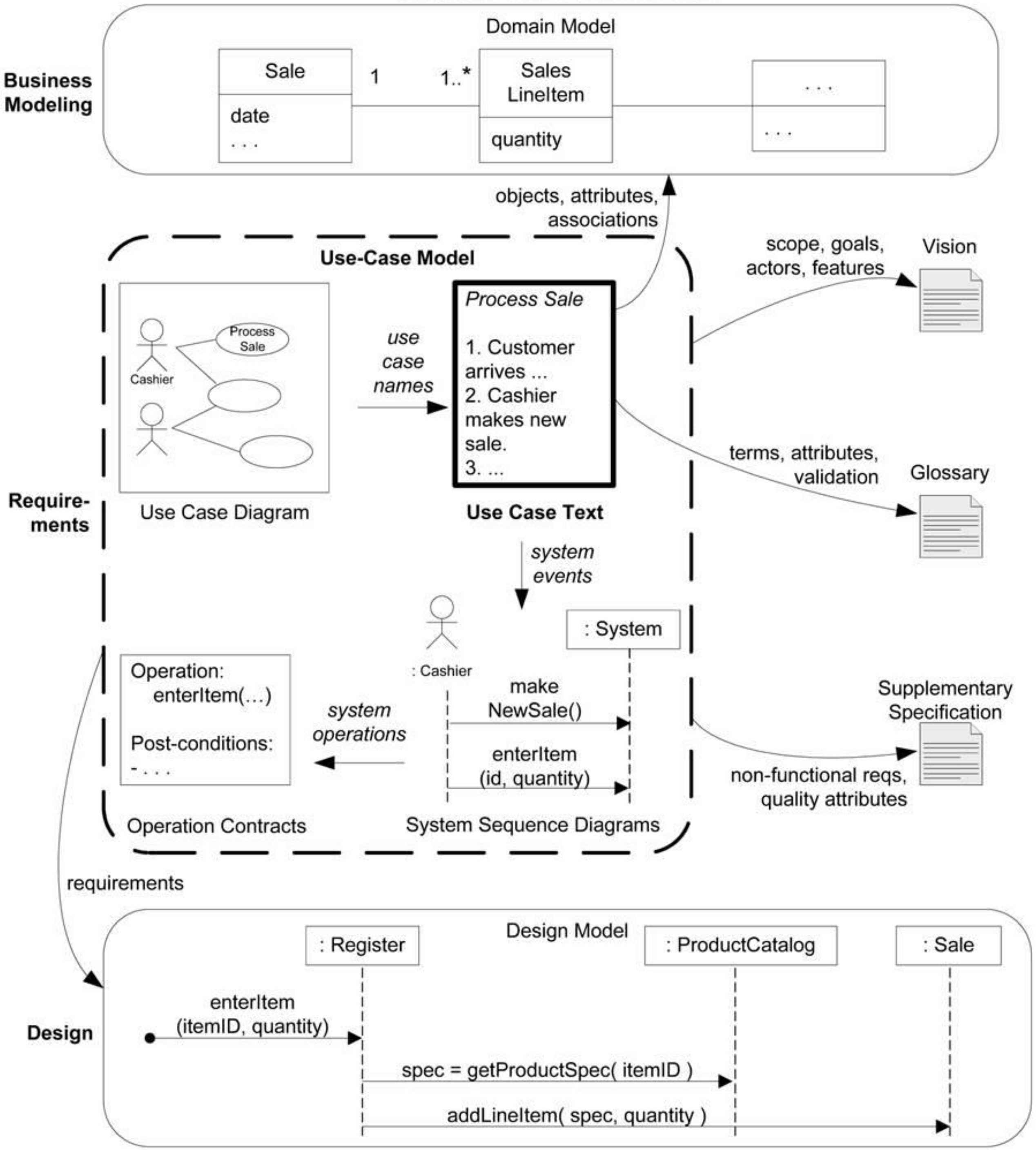
A use case is a generalization of one/more scenarios, like an object to a class

Use Case Influence

Use cases influence many aspects of a project including OOAD, business modeling, glossary, etc.

Constructing the Input

Sample UP Artifact Relationships



Use Case Example, POS

An example from Larman in **brief 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.“

Again note, a **use case is textual, not a diagram.**

UP Artifacts — Use Case Model

UP defines the **Use-Case Model** within the **Requirements Discipline**

Use Case Model is primarily the set of use cases

Use cases are a **key requirements input** to classic **OOAD**

Use-Case Model may include a UML use case diagram

UML Use Case Diagram

A UML use case diagram shows the:

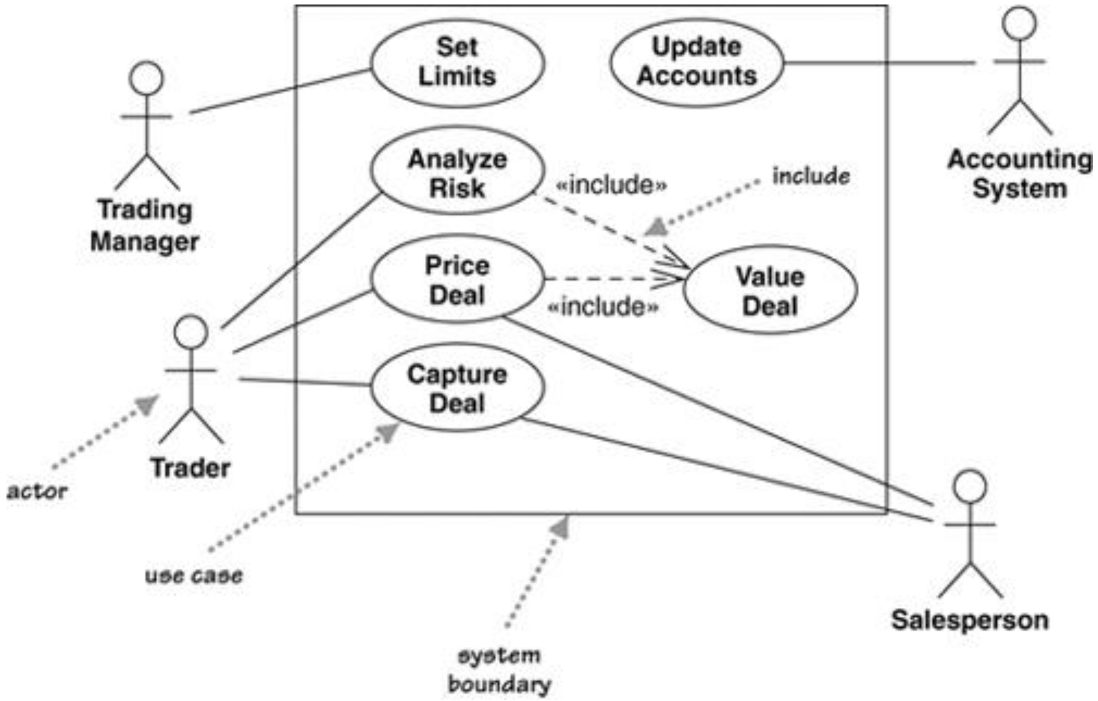
- Names of use cases
- Actors
- Relationships

Use cases are represented by ovals

An actor is represented by a stick person, or not

The diagram provides a quick way to list the use cases by name

Use Case Diagram .



UML Use Case Benefits

Use cases are relatively simple

A use case represents actual uses of the system,
user-centric

Clients can understand use cases and offer feedback

Clients participate in writing vs. list of system features

Actor

Actor is an external entity that needs to exchange information with the system

An actor can represent either a human or an external system

Actors are role abstractions,

a single person may take several roles

Three Types of Actor

Primary Actor– Has user goals fulfilled using services of the System under Discussion (SuD) e.g. A **Cashier**

- Find user goals, drive use cases

Supporting Actor– Provides a service to the SuD, e.g. an inventory system, payment authorization service

- Clarify external interfaces and protocols

Offstage Actor– Has an interest in the behavior of the use case, but is not primary or supporting, e.g. a government tax agency

- All necessary interests are identified and satisfied

Use Case Formats

Brief— Terse one-paragraph summary, usually of the main success scenario

Casual— An informal, sequence of paragraphs, format

Fully Dressed— All steps are written in detail. There are supporting sections, such as alternate flows, preconditions and success guarantees

Fully-Dressed Use Case —Template

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.

Rent Videos —Use Case Example

Use Case UC1: Rent Videos

Primary Actor: Clerk

Stakeholders and Interests:

- Clerk: Wants accurate, fast entry, and no errors.
- Customer: Wants rental and fast service with minimal effort.
- Company: Wants to accurately record transactions and satisfy customers.

Preconditions: Clerk is identified and authenticated.

Postconditions: Rental is recorded. Inventory updated. Receipt is generated.

Summary: Customers brings videos for rental to Clerk, pays Clerk, then receives videos.

Rent Videos —Use Case Example

Basic Flow:

1. Customer brings videos to checkout.
2. Customer presents membership ID to Clerk, who enters it into system.
3. Clerk records item identification for each video.
4. Clerk informs Customer of total charge.
5. Customer pays Clerk with cash or credit.
6. Clerk gives receipt, rental report with return date, and videos to Customer.
7. Customer leaves with videos.

Rent Videos —Use Case Example

Alternate Flows:

3a. System detects an invalid identifier:

- 1. System signals errors and rejects entry.

5a. Customer has insufficient cash:

- 1. Request credit card (CC) payment.
  - 1a. CC unavailable:
    - 1. Clerk cancels transaction.

5b. System reports that Customer has unpaid late charges:

- 1. Customer pays late charges.
  - 1a. Customer fails to pay late charges:
    - 1. Clerk cancels transaction.

5c. System detects failure to contact external CC processor:

- 1. Clerk requests cash payment.
  - 1a. Customer unable to pay cash.
    - 1. Clerk cancels transaction.

Use Case —Name, Scope

Name, a use case has descriptive name, starts with a verb

Scope gives a bound for the system(s) under design

A use case describes use of one software (or hardware plus software) system

Use Case —Level

User-goal level

Scenarios to fulfill the goals of a primary actor to get work done

Sub-function level

Substeps required to support a user goal

Factor out duplicate substeps shared by several regular use cases

- E.g. *Pay by Credit*, which could be shared by many regular use cases

Use Case
—Primary Actor, Stakeholders

Primary Actor is the main actor that uses the system services to accomplish a goal

Stakeholders interests helps:

- Keep in mind all who are affected
- Helps prevent requirements from being overlooked

A use case provides a **contract** for behavior related to satisfying the stakeholders' interests

- Salesperson must have the commission handled properly
- Cashier wants accurate, fast entry and no payment errors, as cash drawer shortages are deducted from his/her salary

Use Case —Pre-Conditions

Conditions that are assumed to be true

Are not tested within the use case

Might imply the scenario of another use case has successfully completed, for example login

Noteworthy assumptions readers should be alerted to. Not worth writing, such as “the system has power”

Use Case —Post-Conditions

True on successful completion of the use case,

main success scenario or some alternate path

The guarantee should meet the needs of all stakeholders

Use Case
—Main Success Scenario

Describes the typical successful completion of a task

Happy Path, Basic Flow, Typical Flow, ..

Does *not* include any conditions or branching

Use Case —Main Success Scenario

Three kinds of **steps** recorded by the scenario:

- An interaction between actors and the system
- A validation, usually by the system
- A state change by the system, e.g., recording or modifying something

Use Case
—Main Success Scenario

First step, **initiate** use case

- Customer arrives at a POS checkout with items to purchase

Idiom to always capitalize the actors' names for ease of identification

- Cashier starts a new sale

Idiom that is used to indicate repetition

- *Cashier repeats steps 3-4 until indicates done*

Use Case —Alternate Flows

Extensions, alternate flows, will comprise much of a use case

Extensions indicate other possible courses that the interaction may take, both successful and failure

Main Success Scenario + Extension Scenario should satisfy nearly all the behavioral? interests of the stakeholders

Some of the stakeholders interests will result in non-functional requirements, Supplementary Specification not a use cases

- Customer's interest for a visible display of descriptions and prices is a usability requirement

Use Case
—Alternate Flows Notation

Branches from the main success scenario, notated with respect to its steps 1 … N

e.g. Step 3 of the main success scenario there may be an invalid item identifier, b/c incorrectly entered or unknown to the system

An extension is labeled "3a", identifies the condition and then the response, alternate extensions at Step 3 are labeled "3b" and so forth

3. Cashier enters item identifier (main)

3a. Invalid identifier:
    1. System signals error and rejects entry.
3b. There are multiple … :
    1. Cashier can ….
        1a. …

Use Case
—Alternate Flows Notation

Range of steps

3-6a: Customer asks Cashier to remove ...:
1. Cashier enters the item identifier for ...

Condition as possible during any steps, the labels *a

*a. At any time, System crashes: ...

Hyperlinks used to link to a different use case, a complex extension might be better as a separate use case

3a. Invalid item ID (not found in system):
1. System signals error and rejects entry.
2a. ...
2c. Cashier performs Find Product Help
to obtain true item ID and price.

Use Case
—Technology & Data Variations List

Specific technologies, might be specified by the client

e.g. input reader or output display

Data schemes

e.g. bar codes, receipts format

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.

Use Case —Open Issues, Note

Open Issues notes items that need clarification

The written use case gives the illusion of completeness and correctness

As modeling of the use case and coding of it progresses, and as feedback is received, flaws will likely be discovered

Use Case Format Example

There are various format templates available for fully-dressed use cases

Template available on alistair.cockburn.us, by Alistair Cockburn

See the example in Larman's, from chapter 6 . (self-study)