The Practical Guide to Use Cases

Learn use cases via practical examples, tips & templates

Michael Shrivathsan

www.accompa.com/use-case-book

First Edition

For my Grandma, who taught me to work hard
and dream big.

Table of Contents

Preface

Who is this book for?

If you participate in software development projects, this book can help you. The most common job roles that this book is targeted at include:

- Business Analysts
- Project/Program Managers
- Programmers
- People in other related roles
- Product Managers
- Engineering Managers
- QA Testers

What is the goal of this book?

My goal for this book is to help you **learn the *practical* fundamentals of "Use Cases"** for use in software development projects.

If you are a beginner to the topic of "Use Cases" – this book can help you learn the basics of this topic and build a strong foundation. If you are an experienced veteran – this book can help you train others about use cases.

How am I qualified to write this book?

I got started writing software requirements in the mid-1990s. Since then I've been writing requirements, managing teams who write them, and/or training teams to write better requirements – including use cases. I've performed these roles for 5 successful high-tech companies in Silicon Valley.

Since 2009, I've been the VP of Product Management for Accompa – we make the popular cloud-based requirements management software. In this role, I've worked with a number of our customers (Fortune-500 companies to growing startups) to help them implement use cases into their requirements management process.

I started writing this book to help Accompa customers. I decided to share it with everyone as I realized more people could benefit from it.

How is this book organized?

We start with a simple definition and an example. We then use a fictional project (online bookstore) to flesh out use cases. Along the way, we'll define and explain various concepts and terms. You can read this book over multiple sessions.

This book does not cover complex theories. Instead, it focuses on how use cases can help you in real-life projects. Just the essentials, ma'am!

1. Introduction

Let us dive right into it. What exactly is a "Use Case" anyhow?

We'll start with a simple definition first. We'll then refine it a few times as we progress through the book and arrive at our final definition in a later chapter.

"Use Case" - Definition #1

Simply put:

```
A "Use Case" describes how a user reaches a specific goal.
```

Sounds very simple - right? Well, not so fast! This is only a preliminary definition, we'll make it more complex soon – I promise!

We'll define the terms italics (i.e. *user* and *goal*) later in this book. We will even define a better term to use in place of *user*.

Now let us take a look at a sample use case.

Use Case Example #1

Throughout this book, we will use a fictional project to demonstrate the various concepts about use cases. This project involves building an "Online Bookstore".

Here's a basic example of a use case for this project:

Buy a book User finds a book he wants and places an order. System accepts the order, and sends it for fulfillment.

As you can see from this example, a use case can be very simple. In the next chapter, we will clarify some terminology many people find confusing. Then we will dive deeper into use cases.

Before we do that, here is a fun exercise for you.

Exercises

1) Write a basic use case for your product. It doesn't have to be "correct" - just write it based on the example above, you will revise it in later exercises.

2. What's in a Name?

Use Cases vs. Requirements
A question that comes up often is "Are use cases requirements?"

Fortunately, the answer is pretty simple. And the answer is: "Yes"!

Use cases are indeed requirements – in fact, they represent a specific type of requirement called "*Functional" Requirement*.

Use cases are actually one of the best ways of documenting the functional requirements (sometimes referred to as *Behavioral Requirements*) of a system.

Please note that in addition to use cases, other types of requirements (such as Non-functional, UI, etc) are also needed to fully document the requirements of a project.

Use Cases vs. User Stories (Agile)
If you are part of a team that follows Agile development process (such as Scrum, XP, etc) , you're probably familiar with *User Stories*.

You might be wondering what the heck is the difference between *Use Cases* and *User Stories*. Hey, after all they both start with the letters "Use" - right?

One of the best explanations of this difference comes from Andrew Stellman in this blog post. Here's a quick summary (I recommend that you read the entire blog post when you have time)…

User Stories describe "user needs" – i.e. something that the user needs to do in his day-to-day job.

- User stories don't contain a lot of details, they are usually just 1 or 2 short sentences. They are not requirements by themselves, but rather "conversation starters" or "pointers" to requirements.

Use Cases, on the other hand, describe the behavior you'll build into the system to meet user needs.

- Use cases describe a complete interaction between the *system* and *user*. As a result, use cases are requirements by themselves.

In summary - *Use Cases* and *User Stories* are different things. That said - I've seen some teams create "lightweight" *use cases* (such as the example in chapter 1 of this book) and use them in agile or hybrid-agile projects in place of *epic user stories*. This is a controversial approach though, and is generally not recommended.

Now that we've clarified these terms, let us look at who reads and writes use cases.

3. The Who

Who reads use cases?

The primary readers of use cases are: People who need to understand what the system being developed must do functionally. This includes roles such as:

- Programmers
- QA testers
- Project/program managers, and
- People in related roles

People who need to review & approve the functionality also read use cases. This includes roles such as:

- Managers & Executives
- Customers
- Partners

Who writes use cases?

The primary writers of use cases are: People who need to communicate to others what the system being developed must do functionally. This includes roles such as:

- Product managers
- Business analysts
- System analysts
- Project/program managers, and
- People in related roles

Ok, now that we've defined who reads and writes use cases - let us consider the benefits of using use cases.

Exercises

1) Write a list of people in your organization who will read use cases.
2) Write a list of people in your organization who will write the use cases.

4. The Why

Benefits of Use Cases

Documenting functional requirements via Use Cases offers key benefits to all stakeholders - as use cases are written using a standard template and process:

Programmers:
- Use cases are easier to read and understand. No long-winded, unstructured text that causes readers to suffer MEGO (My Eyes Glaze Over).

Users, Customers:
- Use cases are easy to review and provide feedback.

Approvers & Executives:
- Use cases make it easier to identify what needs to be removed, and what is missing.

In addition, use cases also form an excellent framework for non-functional requirements of the system – such as performance, scalability, user interface, etc.

Let us now look at a simple example that demonstrates these benefits.

Example: *Unstructured* Text & *Structured* Use Case

BAD – *Unstructured* Text:

Buying a book
Online bookstore has a website that can be accessed over the internet. The online bookstore website offers a list of books. This list can be searched by users. The website processes and ships book to anyone who buys a book from the online bookstore.

GOOD – *Structured* Use Case:

Buy a book
1. User visits online bookstore website.
2. Bookstore website displays a search box.
3. User searches for the book he is looking for: Using author, ISBN and/or title.
4. Bookstore website displays a list of matching books.
5. User places the order for one or more books.
6. Bookstore website processes the order and displays confirmation to user.

As you can see, the use case describes the functional requirements of a system in an easy-to-understand, *structured* text format. It is much easier to read and understand compared to the *unstructured* text.

Use case answers questions like *who*, *when* and *how* much more clearly.

FYI: In a real-world project, this use case should be broken into 3 or 4 separate use cases. We will do just this later on in this book. I've written it as one use case above just to quickly demonstrate the advantages of use cases over a blob of text.

Now we're ready to look at a more complete definition of "Use Case".

Exercises

1) Write a list of benefits your organization can enjoy through use cases.
2) For each benefit, identify who (i.e. which job role) will enjoy that benefit.

5. All the Jargon That's Fit to Print!

"Use Case" - Definition #2

Here's a better definition:

```
A "Use Case" describes how an actor interacts with a system
in order to achieve a goal.
```

Let us now define the terms in *italics* above, as well as a few other key terms frequently used when discussing use cases.

Definitions of Common Terms (Part 1)

System
: A computer, electronic or mechanical system (or a combination thereof) that is capable of interaction.

System Under Design (SUD)
: The *System* we're developing in the project for which we're writing the use case.

External System
: A *System* that is outside of the *SUD*.

Actor
: A human person or an *External System*.

Goal
: The objective the *Actor* has. This is why the *Actor* interacts with the *SUD* in the manner described in a specific use case.

Pre-condition
: The state that the *SUD* was in before the interactions described in the use case start.

Post-condition
: The state that the *SUD* will end up in after the interactions described in the use case are completed.

6. How to Write a Use Case

Revised Example

Now let us revise the use case we wrote earlier to make it more thorough.

Buy a book, using search

SUD: Online Bookstore Website. Referred to as "Bookstore".
Actor: Consumer

1. Consumer visits a page on the Bookstore using a web browser on an internet-connected computer.
2. Bookstore displays the page, which includes a search box.
3. Consumer searches for the book he is looking for: Using author, ISBN and/or title.
4. Bookstore displays a list of matching books.
5. Consumer adds one or more books to the shopping cart.
6. Consumer repeats steps 3-5 until he has added all the books he wants to purchase to the shopping cart.
7. Consumer initiates checkout.
8. Bookstore prompts consumer to enter or select billing & shipping addresses.
9. Consumer enters addresses.
10. Bookstore prompts consumer to enter or select payment information.
11. Consumer enters payment information and places order.
12. Bookstore processes the order and displays confirmation & cross-sell information to consumer.

5 Tips for Writing a Use Case

Here are 5 tips that will help you in writing use cases.

1. Start the title of each use case with an active verb.
    - Notice how in our example above, we started the title with the active verb "Buy".
2. Write using the format of:
    - Actor does A
    - System does B in response
    - User does C
    - System does D in response
    - …
    - Goal achieved

3. Start writing the use case. Even if it is rough and missing key parts, start writing it first. Then you can revise it to improve it and add further details.
4. Remember our end goal in writing use cases: We want to document functional requirements in an easy-to-understand format. As long as we achieve this goal, it is totally fine if our use case does not follow the rigid structure & rules sometimes prescribed by "use case purists".
5. If a use case gets too long (say well more than 10-15 steps), consider breaking it down into two or more use cases. This will keep our use cases easy to read.
    - Different use cases can be connected together using the tips in "Linking Use Cases" later in this book.

Now let us improve our definition of "Use Case" further, and arrive at a final definition.

Exercises

1) Revise the use case you wrote in *exercise 1.1* - using the example and tips in this section. You should end with a use case that looks similar to the revised example in this section.

7. All the Jargon That's Fit to Print – The Sequel

"Use Case" – Final Definition

Here's our final definition of "use case":

```
A "Use Case" describes the story of an actor interacting
with the system under design (SUD) while achieving or
failing to achieve a goal.
```

Thus, a complete use case describes the interaction when the user succeeds in achieving a goal, and when she fails to achieve a goal.

Definitions of Common Terms (Part 2)

Primary Actor
: The *Actor* who initiates an interaction with the *SUD* to achieve the *Goal*.

Scenario
: Sequence of interactions under one condition (such as: User finds the book he's looking for, User is unable to find the book, etc).One or more scenarios (making up *success* as well as *failure* conditions) make up a use case.

Main Success Scenario (MSS)
: The *Scenario* in which nothing goes wrong, and the *Primary Actor* achieves the *Goal*.

Extension
: A *Scenario* during which something goes wrong, or something deviates from the *MSS*.

Use Case ID
: A *unique ID* assigned to a use case. We should assign a unique ID to each use case we write – this makes it easier to communicate about a list of use cases.

8. How to Write "Fully-Dressed" Use Cases ("Naked" is sooo stone age!)

Example – Revised

Now let us revise the use case we wrote earlier to include *Extensions*. We will also make the use case more "fully dressed" (i.e. more structured).

Name:	**Buy a book, using search**
SUD:	Online Bookstore Website. Referred to as "Bookstore".
ID:	UC-1
Primary Actor:	Consumer
Pre-condition:	Consumer is on a computer connected to the internet.
Main Success Scenario:	1. Consumer visits a page on the Bookstore using a web browser on an internet-connected computer. 2. Bookstore displays the page, which includes a search box. 3. Consumer searches for the book he is looking for: Using author, ISBN and/or title. 4. Bookstore displays a list of matching books. 5. Consumer adds one or more books to the shopping cart. 6. Consumer repeats steps 3-5 until he has added all the books he wants to purchase to the shopping cart. 7. Consumer initiates checkout. 8. Bookstore prompts consumer to enter or select billing & shipping addresses. 9. Consumer enters addresses. 10. Bookstore prompts consumer to enter or select payment information. 11. Consumer enters payment information and places order. 12. Bookstore processes the order and displays confirmation & cross-sell information to consumer.

Extensions:	4. (Extensions to search) **A. No matches found** i. Bookstore does not found any matches, and prompts the consumer to revise his search terms. ii. Consumer revises search terms and repeats step 3 of MSS. **B. Match found, but book is out of stock** i. Bookstore informs consumer that the book is out of stock, and is not available for purchase. ii. Consumer adds the book to his "Alert List", so that Bookstore can email him once it is available. 12. (Extensions to order processing) **A. Payment information is invalid** i. Bookstore finds the payment information invalid, and prompts the consumer to re-enter payment information. ii. Consumer re-enters payment information and repeats step 11 of MSS.
Post-condition:	Consumer has successfully placed the order.

Please note that in addition to adding extensions, we've also added more "structured data" to this use case. Specifically:

- ID
- Pre-condition
- Post-condition

5 Tips for Writing Extensions

Here are 5 tips that will help you write *extensions* for a use case.

1. For every step in MSS where behavior can branch due to a *failure* condition – write down the condition and the steps to handle it.
    - Each *extension* is just a very short use case to handle a *failure* condition.
2. Brainstorm all possible *failure* conditions with your team. Then prioritize the extensions to determine which ones you will cover in your use cases.
    - Spend the time to do this step well.
    - For most software systems, documenting the extensions can take as much time (if not more) as documenting the MSS.

    - Missing extensions are one of the top causes of poorly designed products.
    - That said – you must strike a balance between available time and documenting every conceivable extension. Documenting every conceivable extension is often not realistic - as each step in an extension can have its own failure condition, and so forth.
3. Each extension should end by either:
    - Re-merging with MSS successfully, or
    - Primary actor abandoning the goal.
4. Write each extension using the format of:
    - Number the parent bullet of the extension using the step of the MSS (such as "4", "12" in the above example) where behavior branches due to *failure* condition.
    - Give each *failure* condition a unique number (such as "4.A", "4.B" and "12.A" in the above example).
    - Title the extension using a brief description of *failure* condition. See the lines with **bold font** in the above example.
    - For each extension, list numbered steps that describe how SUD will handle the condition.
    - The format described above can be easily achieved with software such as Microsoft Word or Accompa.
5. If an extension gets very complicated, convert that extension into a new, separate use case.
    - You can link the new (extension) use case to the parent use case (MSS) using the tips in "Linking Use Cases" section below.

Exercises

1) Revise the use case you wrote in *exercise 6.1*–to make it **fully-dressed** as in the example this section. You can download the Word template from here.
2) Write 2-3 **extensions** for your revised use case using the example and tips in this section.

9. Relationship Tips

Linking Use Cases - Relationships

In real-world projects, use cases are often related to one another. It is very beneficial – and often absolutely necessary - to create and document these relationships as we write a large number of use cases. This will help us perform impact analysis and tracing of our use cases.

The two most common types of relationships used between use cases are "Extends" and "Includes". Here's a definition of these relationships:

Let us assume we have 3 use cases:

- UC-1: Buy a book
- UC-5: Fix invalid credit card information
- UC-14: Search for a book

Extends

We say "UC-5 **extends** UC-1"when UC-5 describes an *Extension* of UC-1 – i.e. a *failure* condition for one of the steps in the MSS of UC-1.

FYI: The following two descriptions mean the same thing:

- UC-5 **extends** UC-1
- UC-1 **is extended by** UC-5

Includes

We say "UC-1 **includes** UC-14" when UC-14 describes a step of UC-1 in more detail.

FYI: The following two descriptions mean the same thing:

- UC-1 **includes** UC-14
- UC-14 **is included by** UC-1

While it is possible to create and manage relationships using tools such as Microsoft Word or wikis – most teams find dedicated requirements management tools such as Accompa invaluable for this purpose.

Exercises

1) Identify 1-2 use cases that "extend" your use case. You can simply convert the extensions you wrote in *exercise 8.2* into separate use cases.
2) Identify 1-2 use cases that are "included by" your use case.

10. How to Write Use Cases for a Real-World Project

Here are the steps for writing use cases for a real-world project.

Tip: Meet with your team to perform steps 1-4 together as a team. You can then work on step 5.1 by yourself. Then you can meet with your team again to review MSS, and identify all failure conditions (i.e. step 5.2).

1. Create a list of **actors** (people, external systems).
2. Create a list of **goals** for each actor - i.e. what each actor wants to achieve using your system.
3. Now you have the first draft of your **list of use cases**! Each goal becomes a use case.
4. **Prioritize** the use cases in your list. Priority should be assigned based on the business importance of:
   - The actor, and
   - The goal.
5. **Flesh out** each use case:
   5.1. Define the **success condition** for each use case in detail – i.e. Main Success Scenario.
   5.2. Meet with your team to identify failure conditions for each use case. Create **extensions** to address all major **failure conditions**.
   5.3. It is totally fine to cover some extensions, at first, only at a high-level. Later on, you can flesh them out in more detail – and even convert them into separate use cases.
   5.4. Incomplete use cases result frequently from forgetting failure conditions, and can lead to a sub-optimal user experience. Make sure to identify all the key failure conditions.

Sample List of Use Cases for an Online Bookstore

Here are some examples of a use case you might write for a project to build an Online Bookstore. This is just a partial list, to give you an idea – and follows the same steps listed above.

1. Create a list of **actors** for online bookstore:

Actor	Type of Actor
Consumer	People
Bookstore Manager	People
Book Warehouse	External system

3rd Party Book List	External system
Credit Card Processor	External system

2. Create a list of **goals** for each actor:

Actor	**Goal**
Consumer	Browse books
	Search for a book
	Buy book
	Place order
	Track order
	Cancel order
	Write review
Bookstore Manager	Track orders
	Run daily reports
	Provide refunds
	Moderate book reviews by consumers
Book Warehouse	Send book availability information to Bookstore
3rd Party Book List	Send list of books meeting search criteria to Bookstore
	Send details of an individual book to Bookstore
	Send reviews for an individual book to Bookstore
Credit Card Processor	Send credit card validation results to Bookstore
	Send daily transaction report to Bookstore

3. Here's the **list of use cases** – created using the goals listed above:

ID	**Use Case**	**Primary Actor**
UC-1	Browse books	Consumer
UC-2	Search for a book	Consumer
UC-3	Buy book	Consumer
UC-4	Place order	Consumer
UC-5	Track order	Consumer
UC-6	Cancel order	Consumer

UC-7	Write review	Consumer
UC-8	Track orders	Bookstore Manager
UC-9	Run daily reports	Bookstore Manager
UC-10	Provide refunds	Bookstore Manager
UC-11	Moderate book reviews by consumers	Bookstore Manager
UC-12	Send book availability information to Bookstore	Book Warehouse
UC-13	Send list of books meeting search criteria to Bookstore	3rd Party Book List
UC-14	Send details of an individual book to Bookstore	3rd Party Book List
UC-15	Send reviews for an individual book to Bookstore	3rd Party Book List
UC-16	Send credit card validation results to Bookstore	Credit Card Processor
UC-17	Send daily transaction report to Bookstore	Credit Card Processor

4. **Prioritize** the use cases:

ID	Use Case	Primary Actor	Priority of Use Case
UC-1	Browse books	Consumer	P1
UC-2	Search for a book	Consumer	P1
UC-3	Buy book	Consumer	P1
UC-4	Place order	Consumer	P1
UC-5	Track order	Consumer	P1
UC-6	Cancel order	Consumer	P2
UC-7	Write review	Consumer	P2
UC-8	Track orders	Bookstore Manager	P1
UC-9	Run daily reports	Bookstore Manager	P1

UC-10	Provide refunds	Bookstore Manager	P1
UC-11	Moderate book reviews by consumers	Bookstore Manager	P2
UC-12	Send book availability information to Bookstore	Book Warehouse	P1
UC-13	Send list of books meeting search criteria to Bookstore	3rd Party Book List	P1
UC-14	Send details of an individual book to Bookstore	3rd Party Book List	P1
UC-15	Send reviews for an individual book to Bookstore	3rd Party Book List	P2
UC-16	Send credit card validation results to Bookstore	Credit Card Processor	P1
UC-17	Send daily transaction report to Bookstore	Credit Card Processor	P1

5. **Flesh out** each use case:
   We will leave this section as an "Exercise for the reader". Here are some tips:
   a. Take each use case in the table above and flesh it out as a detailed use case. Include as much detail as necessary for your organization to act upon these use cases – but no more.
   b. Identify & document *extensions* for each step of each use case.
   c. Use *includes* & *extends* relationships to link use cases together. Examples from the table above:
      i. UC-3 *includes* UC-1
      ii. UC-3 *includes* UC-2
      iii. UC-3 *includes* UC-4

Exercises

1) Spend 30-60 minutes to recreate steps 1-4 in this section for a specific module of your product/project. You may not be able to do this comprehensively in 30-60 minutes – but you should be able to get a good feel.

11. Templates and Tools

Use Case Template

You can download Microsoft Word template for a use case from the following link:
Download Use Case Template – Microsoft Word

You can choose to use only some of the fields in the template – depending on what works best for your team. To paraphrase the famous expert on use cases, Albert Einstein: "Keep your use case template as simple as possible, but no simpler"! ;)

Software Tools to Manage Use Cases

There are three categories of software tools that are commonly used to manage use cases. Let us take a look at them and their pros/cons.

1. **Desktop Office Suite**
   Many teams use Microsoft Word, Excel and similar tools to create and manage their use cases.

   Pros: These tools are widely available and everyone knows how to use them.

   Cons: It is harder to capture structured data, create relationships, and manage use cases for larger projects.

2. **Online Collaboration Tools**
   With the growing popularity of wikis and online collaboration tools (such as Google Docs), teams at many companies are adapting such tools to document their use cases.

   Pros: These tools are easy to use, and enable online collaboration in real-time.

   Cons: It is harder to capture structured data, create relationships, and manage use cases for larger projects.

3. **Dedicated Requirements Management Tools**
   When teams grow out of tools such as the ones listed above, they start using dedicated requirements management tools like Accompa.

   Pros: These tools make it much easier to capture structured data, create relationships, and manage use cases for larger projects. Web-based tools also enable real-time collaboration over the web.

Cons: These are more expensive than the tools listed above, and there is a learning curve for users to learn and use them.

12. Ten Do's

Here are 10 best practices to follow when you write use cases.

1. **Keep the reader in mind**
   Always remember why you are writing the use cases – i.e. to communicate information (about functional requirements) to your readers.

   Writing clear, easy to understand use cases is our goal – far more important than rigidly adhering to any rules.

2. **Identify all actors**
   Always start by identifying all the *actors*. Doing a good job on this is the first step in ensuring you don't miss any important functional requirements.

3. **Prioritize use cases**
   Always prioritize your use cases. Ideally, you should prioritize them based on their business importance.

4. **Start with the "Satellite view"**
   Online maps such as Google Maps have a "satellite view" and "street view". The "satellite view" provides you a broad view of an area - and as you drill down deeper, you get to the "street view".

   When you start writing use cases for a project, always start with "satellite view" – i.e. with a focus on breadth, not depth. This will get you going quickly, and help you avoid getting mired in details at the outset.

   As you progress, you can focus more and more on depth – i.e. defining the details for each use case. The steps outlined in chapter 10 will help you do this.

5. **Use active voice**
   I've noticed that many people (including me) have a tendency to write in passive voice or from the perspective of the system.

   This results in steps such as "Credit card is verified" – which doesn't clearly define who performed the step. Always write in active voice – for example: "Credit card processing system verifies the credit card".

   Tip: Start each sentence with the name of an actor – this will help ensure an active voice.

6. **Define pre-conditions**
   Always define pre-conditions of each use case. This helps the reader understand the context of the use case. Even when the context is crystal clear to the writer, readers are often confused.

7. **Define post-conditions**
   Always define post-conditions of each use case. This not only helps the readers understand the use case better – it also serves as a "contract" between the project team members.

8. **Define failure conditions**
   Studies by many organizations (such as The Standish Group) have found "Missed requirements" to be a top cause of project failures.

   Missed requirements often result from not fully addressing failure conditions. Work with your team to identify all key failure conditions and define the behavior clearly. For most projects, creating use cases to address failure conditions will take as much time as MSS – and often much longer.

   Well-designed products, such as those from Apple, handle failure conditions in a very user-friendly fashion.

9. **Write textual use cases (Say "No" to stick figures, *aka* UML diagrams)**
   Sometimes you may hear people refer to "UML Diagrams" as use cases – these are diagrams composed of stick figures, ellipses and assorted other hieroglyphics (see this blog post for a discussion on this topic).

   Many readers of use cases- such as end users, engineers & business executives- don't know how to read UML diagrams. There is a reason why humans evolved from hieroglyphics to alphabet text. Text is easier to write & understand!

   Do all your readers a favor, and write text based use cases, in plain English. When it comes to UML diagrams, just say "No". If you're not convinced yet, please reread tip #1 in this chapter.

10. **Keep each use case short**
    Write short use cases, no more than 10-15 steps at the most. If a use case gets longer than this, break it up into multiple use cases – and create relationships (see chapter 9) to tie them together.

    Use cases longer than 15 steps are harder to read - and cause those who read our use cases to suffer the afore-mentioned MEGO (My Eyes Glaze Over) effect!

13. Frequently Asked Questions

1. **Are use cases considered requirements?**
   Yes, use cases are indeed requirements. They are a specific type of requirement – referred to as "Functional Requirements".

   So, use cases are used to document some, but not all, of the requirements for a software project.

2. **Can use cases be used in *agile* projects?**
   Requirements for an Agile project are usually documented via *User Stories.* Please see chapter 2 of this book for a discussion of *Use Cases* vs. *User Stories*.

3. **How do you write and manage large volumes of use cases?**
   Please see chapter 10 of this book for a 5-step process to write and manage large volumes of use cases.

   Requirements management software such as Accompa can be very helpful in doing this as well.

4. **What role do "UML Diagrams" play in use cases?**
   If you're not familiar with UML diagrams (lucky you!) – they are diagrams consisting of stick figures and ellipses, used by some to document use cases. Alistair Cockburn wonders in his popular book "Writing Effective Use Cases":

   > "For reasons that remain a mystery to me, many people have focused on the stick figures and ellipses in use case writing…and have neglected to notice that use cases are fundamentally a text form…"

   I agree with Alistair. I believe the best way to write use cases is using textual format described in this book.

   Just say "No" to UML diagrams – they are very hard to understand for most people who read the use cases we write.

5. **Should use cases contain any information about user interface (UI)?**
   Use cases purists will tell you that use cases should *not* contain any information about user interface at all.

   Well, I'm not a purist when it comes to use cases (or pretty much anything else!). I think it is totally okay to include some information about UI in your use cases. Ultimately, it depends on what makes the most sense for your project team.

   That said, please remember that our primary goal in writing use cases is to

document the functional requirements – not UI.

6. **How do use cases fit into the overall requirements process for a project?**
   In creating and managing requirements for a project – 3 types of artifacts are usually created and managed.
   a. **Features**
      High-level functionality that a system must provide to help the user achieve an objective.
   b. **Use Cases**
      The *functional* requirements needed to implement the feature.
   c. **Requirements**
      The *non-functional* requirements (such as performance, security, UI design, scalability, etc) needed to implement the feature.

   Please see this blog post for a more detailed discussion of of how features, use cases and requirements fit together. You may also want to read the book "Software Requirements" by Karl Wiegers (see link in the "Further Reading" chapter of this book) for an even more detailed look at this.

   In addition, use cases are used as follows:
   a. UI design teams create their UI design specifications based on the use cases
   b. Development teams break down the use cases to create work tasks for developers
   c. QA teams create test cases (for functional testing) based on the use cases

14. Further Reading

I've now shared with you a summarized version of what I've learned about use cases through many years of hands-on experience, and reading some great books. I hope it helps you in the only way that counts – to do your job better and build successful products.

There are many in the industry who know way more about use cases than I do. My reason for writing this book is that they don't seem to have written a short book that helps people quickly learn and benefit from use cases.

However, some of them have written great books on the subject – I've listed them below. I encourage you to buy and read every one of them.

1. **Writing Effective Use Cases - Alistair Cockburn**
   An indispensable book that will help you learn and master use cases. In addition to this book, Alistair has contributed immensely to the subject of use cases.

2. **Software Requirements - Karl Wiegers**
   This is a great book by Karl on managing requirements. This covers the topic of requirements management thoroughly, including use cases.

3. **Mastering the Requirements Process - Suzanne and James Robertson**
   An excellent book that covers the requirements process end-to-end, including a very good topic on use cases.

4. **Applied Software Project Management - Andrew Stellman, Jennifer Greene**
   This book covers many topics on software project management. It is a good to read to understand how use cases fit into overall project management.

15. Glossary

Actor
A human person or an *External System*.

Extends
A type of relationship used to link together use cases for a project. The other type of relationship is "Includes".

Extension
A *Scenario* during which something goes wrong, or something deviates from the *MSS*.

External System
A *System* that is outside of the *SUD*.

Goal
The objective the *Actor* has. This is why the *Actor* interacts with the *SUD* in the manner described in a specific use case.

Includes
A type of relationship used to link together use cases for a project. The other type of relationship is "Extends".

Main Success Scenario (MSS)
The *Scenario* in which nothing goes wrong, and the *Primary Actor* achieves the *Goal*.

Post-condition
The state that the *SUD* will end up in after the interactions described in the use case are completed.

Pre-condition
The state that the *SUD* was in before the interactions described in the use case start.

Primary Actor
The *Actor* who initiates an interaction with the *SUD* to achieve the *Goal*.

Relationship
Used to link together use cases for a project. Two types of relationships are most often used: Extends, Includes.

Scenario

Sequence of interactions under one condition (such as: User finds the book he's looking for, User is unable to find the book, etc.). One or more scenarios (making up *success* as well as *failure* conditions) make up a use case.

System

A computer, electronic or mechanical system (or a combination thereof) that is capable of interaction.

System Under Design (SUD)

The *System* we're developing in the project for which we're writing the use case.

Use Case ID

A *unique ID* assigned to a use case. We should assign a unique ID to each use case we write – this makes it easier to communicate about a list of use cases.

About Accompa - Software to Manage Use Cases

This book is brought to you free by Accompa.

Accompa is a web-based requirements software that helps you create and manage use cases for your projects. You can even define custom templates for your use cases. Here are some ways Accompa can help your team:

- It can save your team a lot of time over managing use cases manually (i.e. using tools such as Word, Excel, Google Docs, wikis, etc).
- It can help your team collaborate on use cases with each other and customers – even when they are located around the world.
- As a result, it can help your team build more successful products and services – and do so more efficiently.

Teams at more than 100 companies of all sizes (from Fortune 500s to growing startups) use Accompa to manage their use cases and requirements over the web. To find out whether Accompa can help your team too, check out one of these links:

- See Product Tour
- Get FREE Trial
- Request Personalized Demo