# Use Case – To Use or Not to Use?

## Introduction

Use cases can be difficult to understand, because they immediately invoke so many different preconceptions and prejudices. English teachers know that some words aren't just words – they are symbolic, and represent ideas. They had us write essays like "Who do I think is a hero" and everyone picks a different person, *for different reasons*.

This can be very powerful as just throwing out a loaded term like *hero* or *use case* communicates a lot more information than the handful of letters would explain. This is also very dangerous, when you throw a different idea than the one that the listener catches.

**If you throw *"brave fireman who saved my cat"*, and your listener catches *"fireman shaved my cat*", you're in trouble**.

Use cases suffer from this symbolic blessing-curse, but added to that are people's past experiences. Executives may have a hazy recollection of "that big project that ran over budget had a bunch of use cases" and be predisposed to not wanting us to invest a lot of time in them.

Or even worse, people are likely thinking of firemen and kittens – everyone seems to have a different definition for *use case*.

## Informal Use Case Template

| Header¤ | |
|---|---|
| Use·Case·ID¤ | UC01¤¤ |
| Use·Case·Version¤ | V.1.0¤¤ |
| Status¤ | <select a status>¤¤ |
| Release¤ | Release·ID¤¤ |
| Author¤ | Your·Name¤¤ |
| **Body¤** | |
| Title¤ | Descriptive·Title·For·The·Use·Case¤ ¤ |
| Actors¤ | Primary·Actor·(Secondary·Actors)¤ ¤ |
| Normal·Flow¤ | The·main·text·that·describes·the·most·common·flow·of·the·use·case.·· This·is·a·paragraph-form,·or·bulleted·list·of·steps.··An·informal·use-case·has·less·structure·than·a·formal·use·case.··An·informal·use·case·is·designed·to·be·rapidly·developed.¶ ¤ ¤ |
| Alternate·Flows¤ | ¤ |
| ¤ | A1.·Alternate·flow¤ ¤ |

*<Press·F1·for·detailed·help·on·any·field>*¶

Free Microsoft Word 2003 template for creating informal use cases. This template is built as a form with guiding text and help text. Read how to use it and download it today.

## Informal Use Cases

Informal use cases are designed to provide as much information as possible without the overhead formal use cases. An informal use case template is much simpler than a formal use case template.

An informal use case includes a minimal set of header information for administrative purposes, and immediately gets to the meat of the use case – describing what the user (also known as the actor) does.

## Informal Use Case Template

Above is a screen shot of the template as you see it when you first open it in MS Word. The template is created as a *form* that you can easily tab from field to field, without having to worry about messing up the formatting. When you want to create an informal use case, just double click on the template file (*.dot). This will create a new word document, Document1.doc. When you save the file, it will be a word document (and not a template).

If you want to modify the template to add header and footer information, for example, you have to open it differently. Open Microsoft Word, and then open the template file. You'll notice that the title bar will say "Informal Use Case.dot" instead of "Document1.doc." That's how you'll know that you are editing the template. All future documents will include any changes you make to the template.

The template is locked, which allows you to quickly tab between the fields (the grey areas in the document) and input information without worrying about changing the formatting of the template. If you want to change the formatting for a single document, just unlock the document that is open. There's an icon that looks like a little lock in the formatting tool bar. Or you can select Tools 2003 or Review in 2007: Unprotect Document.

Note that if you unlock the document, the drop-down control for selecting the use case status will not be available. Just relock the document to re-enable the status drop-down control.

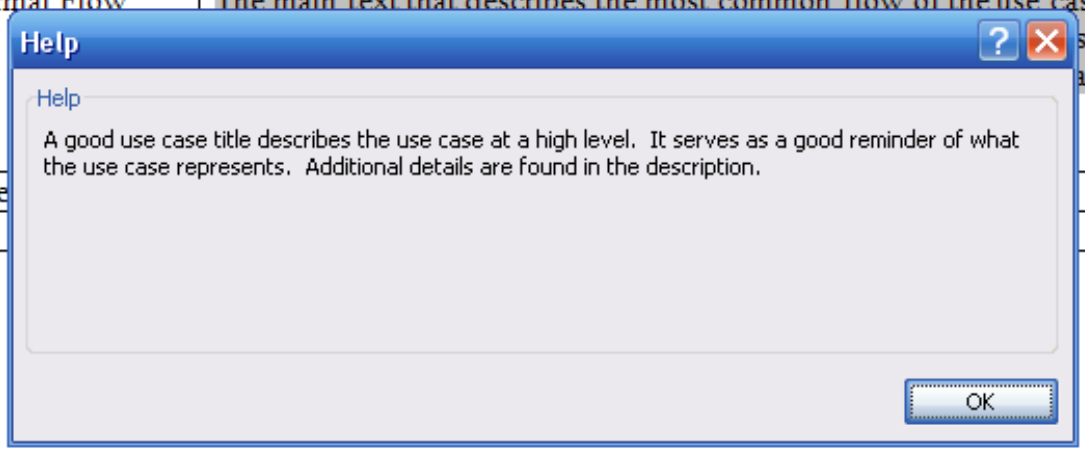
## Using the Informal Use Case Template

1. Download the template file.
2. Double-click on the template file to create a new MS Word document.
3. Enter your data (field details explained below) into the fields.
4. Use the key to quickly navigate to the next field.
5. Save your document.
6. Repeat steps 2 through 5 for additional use cases.

## Template Help feature

There are descriptions in some of the fields (wherever there was room) to remind you what goes in each field. You can also hit F1 to see additional help about any field.

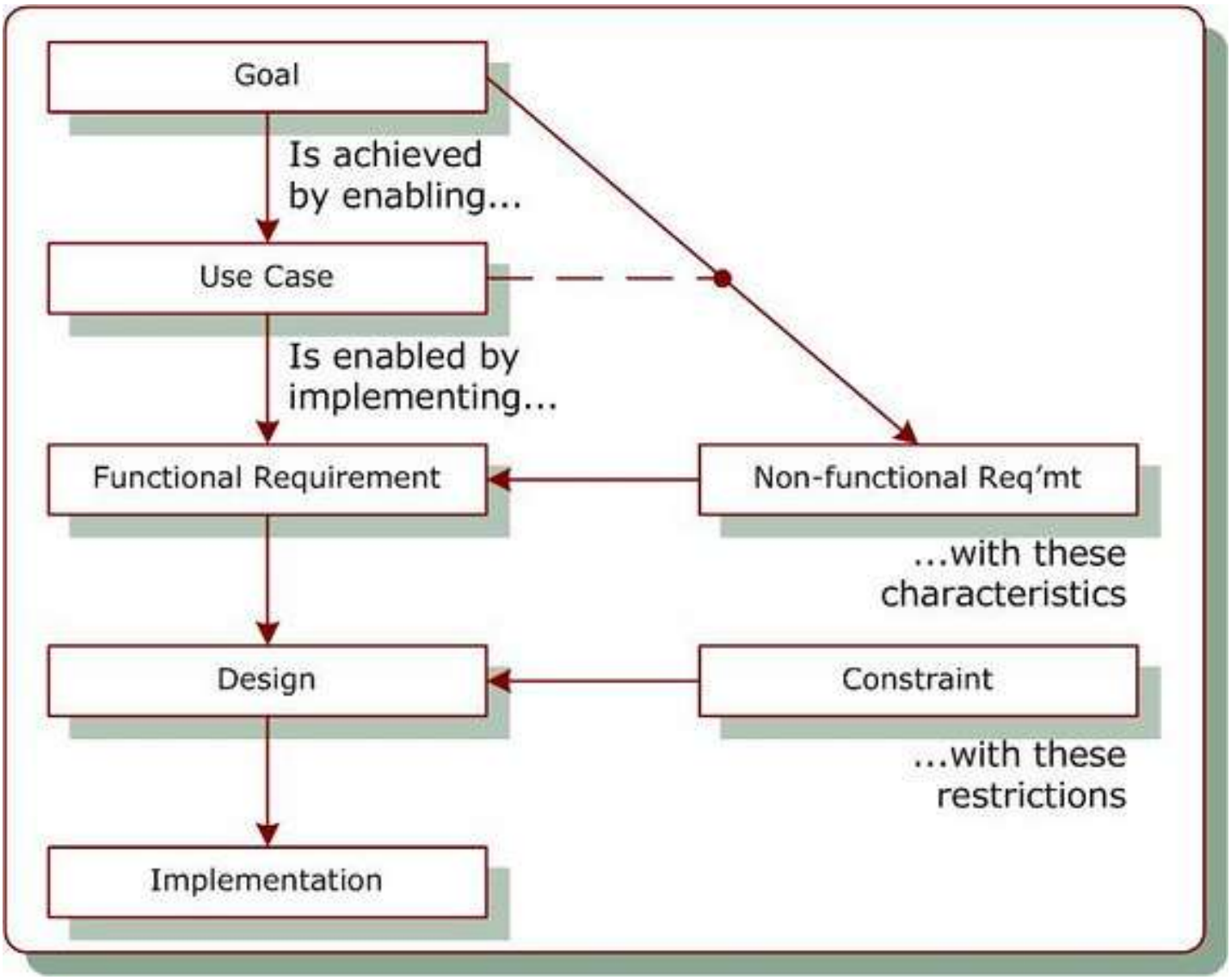
| Header | |
|---|---|
| Use Case ID | UC01 |
| Use Case Version | V.1.0 |
| Status | <select a status> |
| Release | Release ID |
| Author | Your Name |
| **Body** | |
| Title | Descriptive Title For The Use Case |
| Actors | Primary Actor (Secondary Actors) |
| Normal Flow | The main text that describes the most common flow of the use case. |
| Alte | |



## Informal Use Case Fields

The *Header* section of the template organizes the administrative information about the use case.

- **Use Case ID** – This is the unique identifier (***Flow Chart***) that you use to reference the use case from other artifacts that you create as part of developing your software product. In a structured requirements framework, use cases support goals and are supported by functional and non-functional requirements. You will use the use case ID to trace between the use case and the goals it enables. You will also trace between the use case and the functional and non-functional requirements that support it.




***Flow Chart 1 –Use Case ID Identifier***

- **Use Case Version** - The version of the use case (***Chart 1***) can be used to keep track of each draft or revision of the document – but it was intended for something more powerful. We discussed the notion of having *evolutionary* use cases – where each *version* of a use case represents an improvement over the previous version. Version 1.0 might be released for the primary actor, in a particular release. Version 2.0 could be released later, with support for secondary actors. You can track this as the following diagram shows:

| | Actors | | | |
|---|---|---|---|---|
| | Sales Team | | Accounting | |
| | Salesperson | Regional Manager | Corporate Accounting | Collections Processor |
| **Use Cases** | | | | |
| Create Order | Q1 | | | |
| Create Invoice | Q1 | | | |
| Approve Order | | Q1 | | |
| Process Invoice | | | | Q1 |
| Review Regional Orders | | | | |
| Review Regional Orders V1 | | Q2 | | |
| Review Regional Orders V2 | | Q3 | Q3 | |
| Review Corporate Orders | | | Q3 | |

***Chart 1 – Use Case Tracking***

- **Status** – Everyone uses status differently. A good reason to edit this template would be to change the available status options in the template to match what you normally use. The template has four simple status selections.
  1. *Draft* represents an incomplete document.
  2. *Proposed* represents a document that has been completed and is being reviewed.
  3. *Approved* represents a use case that has been approved by all parties.
  4. *Rejected* represents a use case that has been rejected.

- **Release** – Put the name or ID of the release into this field. This represents the release for which the use case is scheduled. It is recommended scheduling releases with complete use cases and not portions of use cases. You can use the versioning approach described above if needed to manage your delivery schedule.

- **Author** – That would be you. Enter the name of the person responsible for authoring the use case.

The *Body* section of the informal use case template includes the primary content of the use case.

- **Title** – the title or name of the use case. This should be a simple sentence that describes the use case. If someone were to only read the title, she should have a good idea of what the use case represents. All your use cases should use a consistent naming convention, but the one you choose is a matter of style. You may want to use the *subject – verb – object* approach ("User Reviews Pending Orders"), or a *verb – object* approach ("Review Pending Orders").

- **Actors** – The people (***Figure 1***) who execute the use case are the actors. Not "Tim" and "Joan," but rather "Office Clerk" and "Department Supervisor." You may want to define an actor hierarchy for organizing the actors that are relevant to this product. If you do – the names in this section should match the ones in the hierarchy.

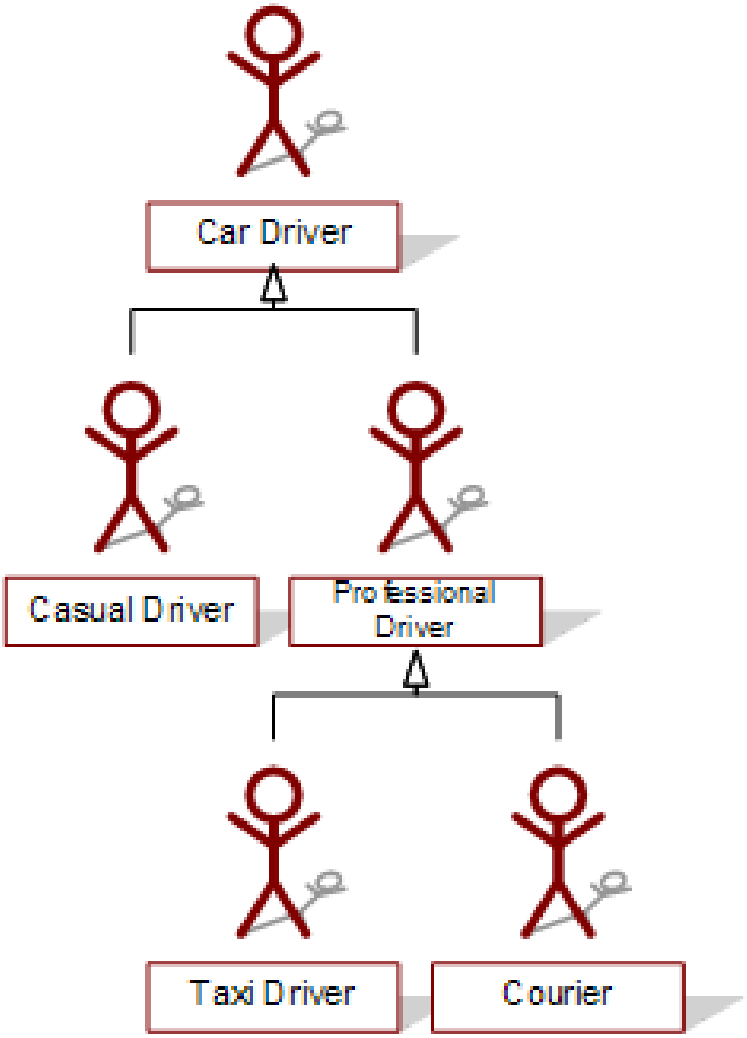



***Figure 1 - Actors***

- **Normal Flow** – This is where the description of your use case goes. The goal here is to write *just enough* to clearly define the use case. The individuals on your team will have the biggest impact on what *enough* is for you. Most use cases involve some branching or decisions. The normal flow should not include these "if…then" constructs. The normal flow should include the most-common or most-valuable path through the use case.

- **Alternate Flows** – This is where those uncommon and lower-value paths are documented. Imagine a use case for processing invoices. The normal course would describe how pending invoices are handled. An alternative flow might handle how past-due invoices are handled. A second alternate flow could handle customers with credit-balances in their accounts.

## Informal Use Case Summary

The informal use case is a great tool for representing requirements without a lot of overhead. Using a template can help your team maintain consistency in documentation.

## Formal Use Cases

This is the classic use case as described by someone who talks about *Software Engineering*. All of the training classes (other than *Agile* classes) that I've been to teach formal use case development as a component in a system of requirements management.

### Pros:

- Detailed information about use cases, making it easy to estimate the cost of implementation.
- Thorough coverage of the use cases is influenced by the use of a template.
- Easy for readers to absorb. The structure of the document makes scanning easy, and also helps targeted *lookups* when a reader needs a specific piece of information.
- Consistency with other use cases is much easier to assure when using a template.

### Cons:

- Expensive to maintain. Mapping a "use case" to the template requires some effort. Since formal use cases contain more (explicit) information than other use cases, there is more content to document, validate and modify. More content equals more cost (of maintenance).

The formal, or classic use case, is a tool used to gather structured information about how users will use the software. Formal use cases are gathered in a template, which structures the information. While this structure can vary from company to company, or even project to project, there are a few common and critical sections to the structure. The two main benefits of having the structure are that it helps with thoroughness (much harder to leave a field blank than it is to forget about it), and it helps with scanning by readers of the document.

Some common elements in a use case template:

- **Actor** – The person or people who will perform the steps of this use case.
- **Preconditions** – A description of the relevant and non-trivial state(s) of the system prior to the use case starting.
- **Normal course** – A description of the use case itself. This description can either be in narrative form, or a numbered list (1..N) of specific user steps. When a use case (such as "User approves/rejects customer requests") has more than one way that a user can accomplish the needed steps, the most common way is shown here – only a single path is shown.

- **Alternate courses** – Descriptions of alternatives to, or deviations from the normal course. For example, the most common course might be to view the oldest unaddressed customer requests. An alternate course may be to view the unaddressed requests from the largest customers.
- **Exception courses** – Descriptions of what the user will experience when something goes wrong.
- **Post-conditions** – Description of the affected portions of the state of the system after the use case has completed.
- **Frequency of use** – An estimate of how often a particular use case will be exercised.
- **Assumptions** – Any assumptions that are implicit in the definition of the use case.

The formal use case can be considered as a contract, in that if the preconditions are met prior to initiating the use case, the post-conditions will be met after its completion. This contract provides a great framework for defining the functional requirements required to support the use case.

## Reading Use Cases

A Use Case represents the activities that people do when interacting with a system to achieve their goals. Use cases are a very effective tool for communicating and documenting what a system is intended to accomplish. Formal use cases are use cases with a specific structure to represent the information. Knowing how to read a formal use case is important.

## Formal Use Case

We wrote previously about formal use cases, covering the pros and cons with a brief overview, as part of our use case series. In this article, we focus on the elements of a formal use case, and how to interpret them.

When using a structured requirements approach to creating software, use cases represent the activities of classes of users, as they relate to the software being developed. Each goal, or market requirement, will have one or more use cases that support it. The easy way to think about it is that for a company to achieve a goal, someone has to do something (Make the Sale; Answer the Support Call, Update Account Information, etc). The use cases capture those *something's*.

Here are the elements of a formal use case, with explanations:

## Meta Data

This is the information that identifies when the use case was first written and by whom. It may also include revision information, who made the changes, what was changed, and when and why it was changed. Revision history is a common *Meta data* element in most business documents – we just normally don't call it "meta data." That's an example of software *jargon*.

## Title

The title of a use case is more than just a heading like "My Pet Spot." The title is intended to provide enough information to identify what the use case represents without reading the use case. This can be useful when getting an overview of a planned project, validating requirements, or when scanning use cases as part of reviewing materials. The title should provide enough information for someone conversant in the domain, while being short. The ideal length is between 4 and 8 words and is a sentence fragment. Anything longer than a sentence is a horrible title. The format is *subject verb object*. Someone does something to or with something else.

**Some examples:**

- Pilot performs pre-flight safety check.
- Author adjusts plot element timelines.
- Accountant reconciles accounts-receivable ledger.

## Description

A brief description of the activity represented in the use case. One to five sentences – no more than a single paragraph – describing the activity or process represented by the use case. The description does not include the full details, but does provide the "next level of insight" into understanding what the use case covers. Someone who is *not* experienced in the domain can get a base understanding of the contents of the use case from reading a description.

**Some examples:**

- A pilot performs an FAA mandated list of equipment and operational inspections prior to every flight. All inspections must pass before the flight is allowed to take off per corporate policy.
- An author will review the timelines, or sequences of events for each plot and subplot within their novel. The author is assuring that the sequence of intended

events is internally consistent, and achieves the desired pacing effects for the book. Any needed changes to the timeline are made.

- Accountants regularly validate that the entries in the accounts-receivable ledger are consistent with the recent sales and current payments. This validation is required for all GAAP public reporting on a quarterly basis, and may be required more frequently for internal management accounting purposes.

## Primary Actor

The primary actor is the person who is the *subject* of the use case, performing the *verb* of the use case on the *object*. A use case may have multiple actors but has one most important person. The term *actor* represents the person who takes action – not someone playing a role. Other actors may be involved, either as participants, or as infrequent or secondary performers of the use case.

**Some examples:**

- Primary Actor: Pilot. Secondary Actors: Flight Crew, Sr. Maintenance Technician
- Primary Actor: Author.
- Primary Actor: Financial Accountant. Secondary Actor: Financial Accounting Manager

## Triggers

A trigger is the outside event that *triggers* the beginning of a use case. This is an event that initiates a use case. Many use cases do not have an obvious trigger – and the trigger will be documented as "User decides to do X." This *invisible* decision represents the trigger.

**Some examples:**

- Pilot receives flight plan.
- Author initiates review of the novel.
- It is the first day of the last month of the quarter.

## Preconditions

Preconditions are the set of things that *must be true* for the use case to occur. These represent a contract of sorts – the use case will not happen unless all of the preconditions are met. If a situation is optional, it is not a precondition.

**Some examples:**

- Flight plan has been approved.

- None. (In our example, the author can review timelines before during or after completing a draft of the novel.)
- Accounts receivable entries have been made in the ledger.

## Post-conditions

Post-conditions represent the other side of the contract. If the normal course (see below) of the use case has been completed, then these things must be true.

**Some examples:**

- The plane has been confirmed to be safe and ready for the approved flight plan.
- The timeline is updated to reflect the author's desired result.
- All accounting entries have been reconciled or identified as being incorrect.

## Normal Course

The preferred, desired or most common sequence of events that represent the use case must be followable in the solution. If no alternate courses are defined, then this sequence of events must be followed (with no alternatives) in the solution. The normal course is documented as a series of *single actor, single action* sentences, each with a number.

**An example:**

1. Accountant accesses current quarter accounts-receivable ledger information.
2. Each of the following steps is repeated for every entry in the ledger.
3. Accountant confirms transaction record matches appropriate other ledger (sales, billing, etc).
4. Accountant updates ledger to indicate that the entry was confirmed.

## Alternate Courses

Alternate course are supported alternatives to the normal course. Each variation is identified as an alternative, and each step that varies from the normal course will be explicitly defined within the alternative course. Each variant step is identified with a number, and all unidentified steps are explicitly defined to be the same as they are in the normal course.

**An example:**

A1: Incorrect ledger entry identified.

A1.3. Accountant confirms that the ledger record does not match the appropriate other ledger.

A1.4. Accountant updates ledger to reflect inconsistency.

A1.5. Accountant identifies proper value to reflect in the ledger.

A1.6. Accountant updates the ledger with the proper value.

A common naming convention is to begin the numbering with the letter "A" to represent that it is an alternate course, followed by a number indicating which alternative it is. In the example above, we have a single alternate course, "A1: Incorrect ledger entry identified". In this example, steps 1 and 2 of the normal course are identical, steps 3 and 4 are replaced with new steps, and steps 5 and 6 have been added.

If another alternate course were defined, it would be numbered "A2", and if it provided an alternate for step 3 in the normal course, that step would be numbered "A2.3."

**[Update]**

Another situation that happens with alternate courses is that a single step in the normal course can be replaced with multiple steps in an alternate course. To represent this, we add a letter, denoting that the steps are all combined to replace a single step. In the example above, we replaced step 4 and added steps 5 and 6. An alternate course may also be documented as follows:

**Another example:**

A1: Incorrect ledger entry identified.

A1.3. Accountant confirms that the ledger record does not match the appropriate other ledger.

A1.4.a. Accountant updates ledger to reflect inconsistency.

A1.4.b. Accountant identifies proper value to reflect in the ledger.

A1.4.c. Accountant updates the ledger with the proper value.

## Exception Courses

An exception course identifies how the system should behave when something goes wrong in the normal course (or an alternate course). If there are multiple exception behaviors, they will be called out as separate exception courses.

**An example:**

E1: Unable to access accounts receivable ledger

E1.1. Accountant fails to access accounts receivable ledger information.

E1.2. System presents error message to accountant explaining access failure and reported cause of failure.

An exception course is identified with the letter "E", and otherwise follows the same numbering conventions as the alternate courses.

## Includes

Sometimes use cases are broken down into smaller, reusable use cases (such as login steps, using a search window to find an account, etc). A reference to another use case from the "*includes"* section indicates that the referenced use case is a subset of this use case.

**An example:**
The use case of "Accountant Prepares Quarterly Report" could include the use case of "Accountant reconciles accounts-receivable ledger".

## References/Traces

Each use case is written to support one or more market requirements (goals). The specific goals that are being supported will be identified as references, or traces.

Each use case is also supported by one or more functional requirements. References or traces to each of these requirements will be identified in this section. These references allow product / program managers to understand the impacts of changes and delays in implementation on the overall product delivery schedule.

## Assumptions

All assumptions that are implicit in the rest of the use case will be explicitly documented here.

**Examples:**

- Pilot is certified to run a safety-check.
- Airport allows pilot to run her own safety-check.

## Notes

Any text that helps the author or potentially helps the reader is documented as notes. These are not *requirement*s documented in the use case, any more than notes in the margin of a book are part of the book.

## Conclusion

With this understanding, reading any formal use case will be a breeze.

## UML Use Case Diagrams

The Unified Modeling Language (UML), the de facto standard visual modeling notation for the analysis and design of software systems, can be used effectively to create such a model.


## The UML Use Cases

### Pros

- Provides a high level view of the use cases in a system, solution, or application.
- Clearly shows which actors perform which use cases, and how use cases combine to form business processes

### Cons

- Presents an "inside-out" view of the system. This description reflects "what it is" not "why it is" – and it is easy to lose sight of why a particular use case is important.
- Poor communication tool when speaking to users and stakeholders about why and when the system will do what it will do.
- Time consuming to create and maintain

Instead of duplicating the explanation and summary work already done, review my Business Use Case Modeling whitepaper.

There are ultimately four pieces of information you want to know about use cases. UML diagrams will show you two of them.

1. **Which actors perform a particular use case?** UML diagrams show this.
2. **Which use cases are combined to create a business process?** UML diagrams show this.
3. **When is a use case scheduled for availability?** UML diagrams do not show this.

4. **Why are we doing a particular use case?** UML diagrams do not show this.

Knowing that we can't answer all 4 questions with a single communication tool, here's what we should do:

(**1&3**) Create a matrix view of use cases versus actors to show which actors perform each use case, and when they will be available.

| | Actors | | | |
|---|---|---|---|---|
| | Sales Team | | Accounting | |
| | Salesperson | Regional Manager | Corporate Accounting | Collections Processor |
| **Use Cases** | | | | |
| Create Order | Q1 | | | |
| Create Invoice | Q1 | | | |
| Approve Order | | Q1 | | |
| Process Invoice | | | | Q1 |
| Review Regional Orders | | | | |
| Review Regional Orders V1 | | Q2 | | |
| Review Regional Orders V2 | | Q3 | Q3 | |
| Review Corporate Orders | | | Q3 | |

(**2**) Create a UML 2.0 use case diagram if you find that the benefits for your communication outweigh the costs of maintaining the diagrams. In projects I've worked on in the past, a simple flow chart with use case names has been used. These simple charts can be made in a fraction of the time, are more easily scannable, and present information more densely. If you are managing requirements with a tool that automatically generates the diagrams, then do it – but don't spend a lot of time on them. A flowchart takes almost no time to draw, and communicates the information just as effectively (and more succinctly). Suggestion – use the flow chart.

(**4**) Ultimately, UML diagrams (often referred to as "use case cartoons") focus your attention on what you are building, at the expense of losing focus on why you are building it. Create a mapping or maintain links (traceability) from use cases to goals.

The *why* of the use case is the most important information! Don't let use case cartoons distract you from it.

# Use Case – To Use or Not to Use?

**Why?**

Stop and think about the best people you work with – developers, project managers, CIOs, you'll find a common pattern – **they ask "why?" all the time**. Somehow, they never stopped asking "why?" Why, to understand, and the ones that don't; keep your eye on them.

Why, is the central theme, the underlying cause, and the most important element to developing a successful product. And it plays an important role in documenting requirements. Without knowing why a product is valuable or why people will use it, or why it needs to be done in 3 months instead of 6, you aren't likely to make the right decisions about what to include, when to include it, or how to market it.

A reader and made a comment about a previous post about requirements testability – he proposed that by rewriting the requirement, we may lose sight of the real requirement. This is really a risk that we lose the underlying "why?" of the requirement. There are ways to prevent disconnect, by keeping requirements synchronized with a product roadmap, repeatedly articulating the strategy, and just keeping your eye on the ball.

Another approach, the one we use, is to structure requirements in the context of goals. We document the goals for a product, we identify the use cases that are required to achieve the goals, and we document the functional requirements required to enable the use cases. Maintaining linkage, or traceability, between the requirements and the goals that they ultimately support is the key to not losing sight of the real requirements.

Asking "why?" can also help in requirement elicitation. "Why do we need 99.99% uptime instead of 99.9%?" "Why is it a requirement that the system work without an internet connection?" "Why is it important to the user to be able to save their work in progress?"

Now you see why!  Also known as Risk Management (RM) or enterprise Risk Management (ERM).