New Mexico
Department of
Information Technology

[INSERT PROJECT NAME]

USE CASE SPECIFICATION AND TEMPLATE

EXECUTIVE SPONSOR – [INSERT NAME]

BUSINESS OWNER - [INSERT NAME]

PROJECT MANAGER – [INSERT NAME]

ORIGINAL PLAN DATE: [INSERT DATE, SPELLED OUT]

REVISION DATE: [INSERT DATE, SPELLED OUT]

Revision: [Insert Number]

ABOUT THIS DOCUMENT

This document provides a background for creating use cases and a specific template for use case development.

REVISION HISTORY

REVISION NUMBER	DATE	COMMENT
1.0	August 14, 2007	Original DoIT PMO Document

Table of Contents

1 DESCRIPTION

This document describes the process for developing use case specifications during requirements gathering. This process needs to be followed when developing functional requirements when those requirements are from a users perspective.

1.1 WHAT IS A USE CASE?

A use case is a kind of story of how a system and its actors collaborate to achieve a specific goal. It is a step-by-step description of a particular way of using a system. The structure of a use case is narrative in nature. The story tells how the system and its actors work together to achieve something of significance to the actors involved.

Each use case expresses a goal of the actors involved and describes a task that the system, with the assistance of the appropriate actors will perform. You can get an idea of a use case's goal simply by observing its name and associations.

When treated formally, the collected set of a system's use cases constitute all the possible ways of using the system.

2 BUSINESS PURPOSE

Use Case Specifications will drive communication between the business and development team. Use Case Specifications provide a way to represent the user requirements that align with the system's business requirements. They are specified in non-technical language and enable ordering, grouping and prioritization techniques. Use Cases should be used to convey user and functional requirements. Also, Use Case Specifications will be the primary input for user acceptance testing as well as the development of test cases.

3 DEVELOPMENT METHOD

3.1 INPUT(S)

← Business/User Requirements Document
← Goal Model

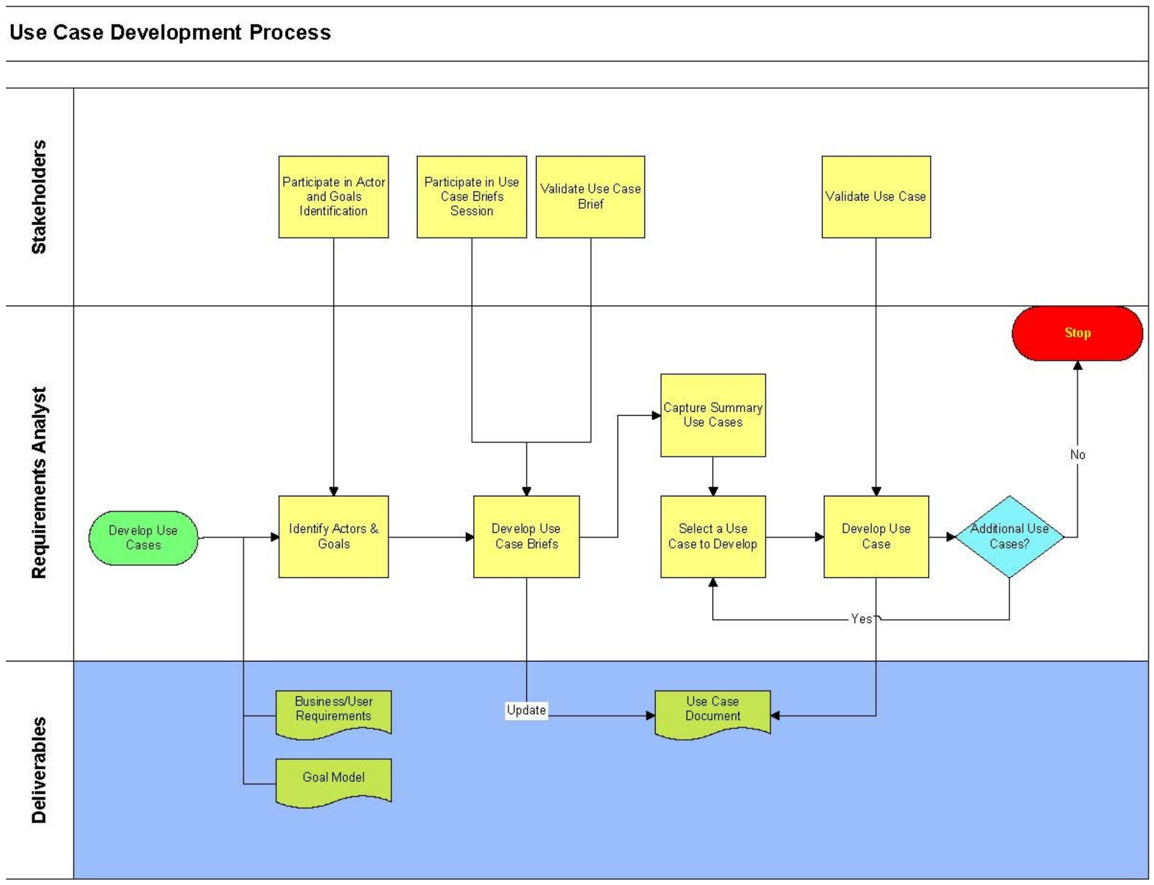
3.2 OUTPUT(S)

• Use Case Document

3.3 DEPENDENCY(IES)

Review Goal Documentation and Business Requirements. Interests should have been documented as Business Requirements and should have been captured in high-level statements that are clear and are typically in the format of 'I/we/Our organization need'.

3.4 Process



3.5 PROCEDURE

3.5.1 Brainstorm Actors & Goals

3.5.1.1 Actors

To fully understand the system's purpose you must know who will be using the system. A facilitated session with the Stakeholders hosted by the Requirements Analyst shall be held to brainstorm all primary and supporting actors (users) along with their goals. Different user types are represented as actors. An actor is anything that exchanges information with the system. An actor can be a user, external hardware, or another system. Each actor must have goals with respect to the use case it is interacting with in the problem domain. Many of the details necessary for Use case modeling are driven from the Primary Actors' point of view.

Primary Actor - The “roles” that will be *using* the system and *initiating* the use case (functionality) and whose responsibility is to complete the use case successfully.

Supporting Actors - "Things" (another system, third party, etc.) that interact with the system. Supporting actors are identified only with respect to use cases. In otherwords, an Actor could be primary while acting as supporting on a different use case. By definition, supporting actors are necessary for the use case to take to its completion.

3.5.1.1.1How to Find Actors

Start by thinking of individuals who will use the system. How can you categorize them? It is often a good habit to keep a few individuals (two or three) in mind and make sure that the actors you identify cover their needs. The following set of questions is useful when you are identifying actors:

1 a. Who will supply, use, or remove information from the system?
2 b. Who will operate the system?
3 c. Who will perform any system maintenance?
4 d. Where will the system be used?
5 e. Where does the system get its information?
6 f. What other external systems will interact with the system?

3.5.1.2 Brainstorm Actors (User) Goals for the System

Assign each goal a priority Goals of an actor are identified only with respect to use cases. Goals must describe the desired state of one or more resources utilized by use cases. The goal that will be attached to a use case must motivate tasks within while referring to the resources used in the use case and not its outside abilities. In other words, balancing the goal with the use case is essential to use case modeling. Do not attach a high-level goal to a lower-level use case or vice versa. The Goal Model shall be used as a starting point for this activity. If all goals have been identified, then this activity shall validate those goals rather than having to develop them. Each goal shall have a priority assigned to them.

3.5.2 Develop Use Case Brief

The Use Case Brief should reflect the use case behavior. As the description is written, refer to the actors involved in the use case, the glossary and, if needed, define new concepts. The purpose of these briefs is to make sure the project team understands the stakeholders needs, or more important, to make sure that the project team and stakeholders agree on the purpose of and the value provided by the use case. Without a use case brief, the project team may *think* they are in agreement on whom or what the actors represent when in fact there may be slightly different conceptions. Without a use case brief, a good understanding of the purpose of the use case may be lost.

The use case brief should be two to six sentences long, but no more than a short paragraph; anything longer is probably overkill. Keep the briefs simple and direct. If a simple and direct use case brief is difficult to develop, then reconsider whether it is needed - if it can't be defined then a clear idea of what is trying to be achieved is not clear.

3.5.2.1.1 Update Use Case Document

Actors, their Goals and the Use Case Briefs should be documented in the appropriate section of the Use Case Document.

3.5.3 Capture Summary Use Cases

Use cases help us focus on what is essential and ultimately create a system that does something useful. Now that the actors, their goals and use case briefs have been identified, representing the text as a

New Mexico
Department of
Information Technology

picture is worth a thousand words.

The actors and use cases can be depicted on a use-case diagram. The use case diagram assists with identifying system boundaries and keeps the project team focused on the main goals of the system. Below is an example of a use case diagram that shows some of the actors and use cases for a very simple telephone system. Actors are represented by stick people and use cases by ellipses. Primary actors (initiators of the use case) are typically represented by the stick figure on the left where as the supporting actors (actors required to help complete the use case) are on the right. Arrows (representing associations) connect the actors and the use cases that interact. The arrowheads help to indicate the initiator of the interaction.

The purpose of the diagram is to summarize what the system will do. The diagram does not really describe the entire system. The diagram provides a summary, but the bulk of the description is held, as text, in the use case specifications. The use-case specifications provide the full story of what happens in the use case. So for every use case in the use-case diagram there will be a specification describing how the actors and the system collaborate together to fulfill the goal represented by the use case. During this step reconsider and revise the summary use cases. Add, subtract, or merge goals.

3.5.4 Select a Use Case to Develop

Before the use cases are described in detail, they should be reviewed to verify that all use cases and actors are identified, and together they can provide what the stakeholder needs. The review shall be facilitated by the Requirements Analyst and include all affected stakeholders.

Based on stakeholder priorities, select a Use Case to expand upon. Choosing one Use Case to expand on will allow the Requirements Analyst to confirm with the stakeholder that their needs are being met early in the process. Also, since use case functionality can stand on its own, this technique allows for iterative development.

3.5.5 Develop the Use Case

The remaining sections of the Use Case shall be expanded upon. The following paragraphs will describe each section of the Use Case Specification:

3.5.5.1 Trigger

An actor must initiate each use case. Actor's behavior to start the use case must be triggered by at least an event. Identify the trigger in terms of business events if the actor is person or an organization. System events trigger use case that are initiated by system actors.

3.5.5.2 Frequency

Express the frequency of initiations of the use case. This will indicate an aspect of performance requirements when the use case is realized. Frequency provides insight into volumes related to the objects and severity of the goal of the use case.

3.5.5.3 Flow of Events

The flow of events of a use case contains the most important information derived from use-case modeling work. It should describe the use case's flow of events clearly enough for an outsider to easily understand it. Remember the flow of events should present what the system does, not how the system is design to perform the required behavior.

The flow of events in a use case should describe the interaction between the actor and the system in a way that clarifies how the actor works in order to achieve a certain goal. Whenever it's not obvious who is doing something it should be stated whether it's the actor or the system. The use case should be described from the actor's point of view.

3.5.5.3.1 Content

Guidelines for the contents of the flow of events are:

- Describe how the use case starts and ends
- Describe what data is exchanged between the actor and the use case

Do not describe the details of the user interface, unless it is necessary to understand the behavior of the system. For example, it is often good to use a limited set of web-specific terminology when it is known beforehand that the application is going to be web-based. Otherwise, the risk that the use-case text is being perceived as too abstract. Words to use could be "navigate", "browse", "hyperlink" "page", "submit", and "browser". However, it is not advisable to include references to "frames" or "web pages" in such a way that assumptions are being made about the boundaries between them - this is a critical design decision.

- Describe the flow of events, not only the functionality. To enforce this, start every action with "When the actor ... "
- Describe only the events that belong to the use case, and not what happens in other use cases or outside of the system
- Avoid vague terminology such as "for example", "etc. " and "information"
- Detail the flow of events—all "what's" should be answered. Remember that test designers are to use this text to identify test cases.

3.5.5.4 Alternative Flows

The alternative flows of a use case cover behavior of optional or exceptional character in relation to the normal behavior. Also, the alternative flows describe errors and exception handling.
However, there is no need to describe all possible error handling, the emphasis should lay on the cases where the business rules forces the flow to diverge from the basic flow.

To clarify where an alternative flow of events fits in the structure, describe where in the basic flow this particular alternate flow is initiated and where the basic flow resumes when the alternate flow ends.

It might be tempting, if the alternative flow of events is very simple, to just describe it in the basic flow of events section (using some informal "if-then-else" construct). This should be avoided. Too many alternatives will make the normal behavior difficult to see. Also, including alternative paths in the basic flow of events section will make the text more "pseudo-code like" and harder to read.

In general, extracting parts of the flow of events and describing these parts separately can increase the readability of the basic flow of events and improve the structure of the use case and the use-case model.

3.5.5.5 Non-Functional Requirements

In the non-functional requirements section of a use case, describe all the requirements of the use case that are not covered by the flow of events. These are requirements that are specific to a use case but are not easily or naturally specified in the text of the use case's event flow. Examples of non-functional requirements include legal and regulatory requirements, application standards, and quality attributes of the system that relates directly to that particular use case.

3.5.5.6 Preconditions

This starting point is represented by the states of the actor(s) and the system at the time the use case is to begin. This statement is known as a precondition. For example, the Place Local Call use case could be given the following pre-condition: The Caller's device has a connection to the system, i.e. the carrier signal is there.

Preconditions are not a description of the event, or trigger, that starts the use case, but rather a statement of the conditions under which the use case is applicable. The precondition is necessary for the use case to be started, but is not sufficient to start the use case. The use case must still be started by an actor but can only be started when the precondition is true.

3.5.5.7 Post Conditions

In addition to using preconditions to clarify when the use case is available, it is often very useful to also specify the state of the system when the use case ends. This is done by the use of post conditions.

A post condition for a use case should be true regardless of which alternative flows were executed; it should not be true only for the basic flow. If something could fail, it would cover that in the post condition by describing the states in which the system can be when the use case is completed. For example, the Place Local Call use case could be given the following post condition: The connection between the Caller and Callee has been terminated and all call details have been recorded.

This may seem trivial, but as alternative flows are added to the use case, they can often lead to the system being left in unacceptable states. For example, a colleague was once phoned by one of his friends who was using a prototype next generation mobile phone. Unfortunately, the new system didn't allow the termination of the connection from the Callee's phone. When the initial call was over and the prototype phone was returned to its owner's pocket, it accidentally redialed the colleague's number leaving his phone unusable for the next few days; the new phone was incapable of terminating the call. Needless to say, the colleague received a rather large surprise on his next phone bill!

3.5.5.8 Extension Points

An extension point opens up the use case to the possibility of an extension to another use case. It has a name, and a list of references to one or more locations within the flow of events of the use case. An extension point may reference a single location between two behavior steps within the use case. It may also reference a set of discrete locations.

To use named extension points will help separate the specification of the behavior of the extending use case from the internal details of the base use case. The base use case can be modified or re-arranged, as long as the names of the extension points remain the same it will not affect the extending use case. At the same time, you are not loading down the text describing the flow of events of the base use case with details of where behavior might be extended into it.

3.5.5.9 Special requirements

It is quite easy to see that use cases are a very good way of capturing functional requirements on a system, but what about the non-functional requirements? What are they and where are they captured?

Many non-functional requirements apply to an individual use case and are captured within the properties of that use case. In that case, they are captured within the flow of events of the use case, or as a special requirement of the use case. Often the non-functional requirements apply to the whole system. Such requirements are captured in the Systems Requirement Specification.

3.6 TEMPLATE(S)

Use Case Document Template

3.7 QUALITY ASSURANCE

Consider the use case "done" when:
1 . All the primary actors and all the user goals with respect to the system have been named.
2 . All trigger conditions to the system either as a use case trigger or as extension conditions have been captured.
3 . All the user-goal use cases have been written, along with supporting use cases.
1 . Each use case is written clearly enough that:
a o The sponsors agree that they will be able to tell whether or not it is actually delivered.
b o The users agree that it is what they want or can accept as the system's behavior.
c o The developers agree that they can actually develop that functionality.
d o The sponsors agree that the use case set covers all they want (for now)

New Mexico
Department of
Information Technology

USE CASE SPECIFICATION TEMPLATE

The next few pages with red text are intended to form the use case template – Copy them to a new document for each use case developed for the project.

ACTORS, GOALS AND USE CASE BRIEFS

Actors	**Task-level Goal**	**Priority**	**Brief**
<Enter a name identifying ALL the categories of actors that will execute the system with the goal in mind>	<Describe the goal in question at the system, not business, level>	<Set a priority for the related requirements: High, medium, or low>	<Describe the interactions that the actor and the system will have to go through in order to achieve the goal. The description shall be two to six sentences>

USE CASE DIAGRAM

<Include use case diagram outlining the actors, system boundary and use cases. See Use Case Development process for details>

USE CASE SPECIFICATIONS

<This template should be copied and completed for each individual use case. If a requirements management tool is available, the requirements management tool should automate the Use Case Identification and History section>

Use Case Identification and History			
Use Case ID:	PROJ.UC.1.1.1 <Assign a unique name to each use case>		
Use Case Name:	<Name – concise results oriented-name with an action verb and a noun. >	**Version No:**	
End Objective:	< The directly observable purpose of this use case >		
Created by:		**On (date):**	
Last Update by:		**On (date):**	
Approved by:		**On (date):**	
User/Actor:	<Description of the person who uses the system to accomplish tasks>		
Business Owner Name:		**Contact Details:**	
Trigger:	<Who (system or user) triggers this use case>		
Frequency of Use:	<How often does this series of activities occurs>		

Preconditions
<What is true of the system state before this flow of actions begins>

Basic Flow <The optimal or normal ("good day") flow of events. The basic flow of events should describe the events that walk through a successful scenario. The basic flow should not include "and/if scenarios">		
Step	**User Actions**	**System Actions**
1	<Phrase saying what user does>	<Description of system response>

New Mexico Department of Information Technology

Basic Flow <The optimal or normal ("good day") flow of events. The basic flow of events should describe the events that walk through a successful scenario. The basic flow should not include "and/if scenarios">		
Step	**User Actions**	**System Actions**
2	<Repeated as needed>	<Repeat…>

Alternate Flow *<may be more than one>*		
Step	**User Actions**	**System Actions**
1	<Phrase saying what user does, identify prior basic step # where alternative flow begins and subsequent basic step # where basic flow continues (if it does) after performing alternate flow> <Example: The User overrides the make, model or year of vehicle (Basic Flow, Step 3)>	<Description of alternative flow step(s) (number each distinct step)> <Example: System requests confirmation by the user>
2	<Repeat as needed>	<Repeat…>

Exception Flow *<identify system and data error conditions that could occur for each step in the normal and alternate flow>*		
1	<Phrase saying what data or system error condition occurred. Identify prior basic or alternative step # where basic/alternative flow begins and subsequent basic/ alternative step # where basic/ alternative flow continues (if it does) after performing exception flow > <Example: The VIN is invalid. (Basic Flow, Step 2)>	<Description of system response to return back to the next step in the flow or state if the condition results in termination of the flow and what the final state of the system is at this point (i.e. redisplay first screen before error condition or something else.) (Number each distinct step). > <Example system response: The system indicates that the VIN format is not valid and asks for reentry of the VIN. The Basic Flow resumes at Step 3. >
2	<Repeat as needed>	<Repeat…>

Post conditions
1. <What is true of the system when the flow of activities finishes>

Includes or Extension Points
1. <Common functionality that appears in multiple use cases can be split out into separate use cases. Provide reference to such of the use cases that are called by the subject use case. >

Special Requirements
1. <Identify any special non-functional requirements such as legal, performance, etc. that need to be considered during design or implementation. These requirements should only be documented here if they are specific to this use case. If the requirements span across multiple use cases, document in the appropriate section of the Systems Requirements Specification>

Business Rules
1. <Identify any business rules or constraints particular to this specific use case. Example of a business rule would be: “When an Account of a subscription has a Credit Card on File, all subscriptions under that account rollover month-to-month.”>

Other Notes (Assumptions, Issues,)
< Any special considerations that need to be kept in mind for this use case only; identify the type of item with a tag like • **Assumptions:** • **Issues:**

USE CASE SPECIFIC CHECKS

USE CASE DIAGRAM

- ❑ The introduction section of the use-case diagram provides a clear, concise overview of the purpose and functionality of the system.
- ❑ The use case diagram clearly presents the behavior of the system; it is easy to understand what the system does by reviewing the diagram.
    - ▪ No long chains of include and extend relationships, such as when an included use case is extended, or when an extended use case includes other use cases. These can obscure comprehensibility.
    - ▪ Minimal cross-dependencies where an included, extending, or specialized use case must know about the structure and content of other included, extending or specialized use cases.
- ❑ All use cases have been identified; the use cases collectively account for all required behavior.
- ❑ All functional requirements are mapped to at least one use case.
- ❑ All non-functional requirements that must be satisfied by specific use cases have been mapped to those use cases.
- ❑ The use-case diagram contains no extra system behavior; all use cases can be justified by tracing them back to a functional requirement.
- ❑ All relationships between use cases are required (i.e. there is justification for all include-, extend-, and generalization-relationships).

ACTORS

- o Have you found all the actors? That is, have you accounted for and diagramed all roles in the system's environment? Although you should check this, you cannot be sure until you have found and described all the use cases.
- o Is each actor involved with at least one use case? Remove any actors not mentioned in the use-case descriptions, or any actors without communicates-associations with a use case. However, an actor mentioned in a use-case description is likely to have a communicates-association with that particular use case.
- o Can you name at least two people who would be able to perform as a particular actor? If not, check if the role the actor diagrams is part of another one. If so, you should merge the actor with another actor.

- Do any actors play similar roles in relation to the system? If so, you should merge them into a single actor. The communicates-associations and use-case descriptions show how the actors and the system interrelate.
- Do two actors play the same role in relation to a use case? If so, you should use actor-generalizations to diagram their shared behavior.
- Will a particular actor use the system in several (completely different) ways or does he have several (completely different) purposes for using the use case? If so, you should probably have more than one actor.
- Do the actors have intuitive and descriptive names? Can both users and customers understand the names? It is important that actor names correspond to their roles. If not, change them.

USE CASE SPECIFICATIONS

- ❑ Is each concrete use case involved with at least one actor? If not, something is wrong; a use case that does not interact with an actor is not required, and you should remove it.
- ❑ For an included use case: does it make assumptions about the use cases that include it? Such assumptions should be avoided, so that the included use case is not affected by changes to the including use cases.
- ❑ Do any use cases have very similar behaviors or flows of events? If so - and if you wish their behavior to be similar in the future - you should merge them into a single use case. This makes it easier to introduce future changes. Note: you must involve the users if you decide to merge use cases, because the users, who interact with the new, merged use case will probably be affected.
- ❑ Has part of the flow of events already been diagramed as another use case? If so, you can have the new use case use the old one.
- ❑ Is some part of the flow of events already part of another use case? If so, you should extract this subflow and have it be used by the use cases in question. Note: you must involve the users if you decide to "reuse" the subflow, because the users of the existing use case will probably be affected.
- ❑ Should the flow of events of one use case be inserted into the flow of events of another? If so, you diagram this with an extend-relationship to the other use case.
- ❑ Do the use cases have unique, intuitive, and explanatory names so that they cannot be mixed up at a later stage? If not, you change their names.
- ❑ Do customers and users alike understand the names and descriptions of the use cases? Each use-case name must describe the behavior the use case supports.
- ❑ Does the use case meet all the requirements that obviously govern its performance? You must include any (nonfunctional) requirements to be handled in the object diagrams in the use-case *Special Requirements.*
- ❑ Does the communication sequence between actor and use case conform to the user's expectations?
- ❑ Is it clear how and when the use case's flow of events starts and ends?
- ❑ Behavior might exist that is activated only when a certain condition is not met. Is there a description of what will happen if a given condition is not met?
- ❑ Are any use cases overly complex? If you want your use-case diagram to be easy to understand, you might have to split up complex use cases.
- ❑ Does a use case contain disparate flows of events? If so, it is best to divide it into two or more separate use cases. A use case that contains disparate flows of events will be very difficult to understand and to maintain.

- Is it clear who wishes to perform a use case? Is the purpose of the use case also clear?
- Are the actor interactions and exchanged information clear?
- Does the brief description give a true picture of the use case?

REFERENCES

- Rational Unified Process Use Case Checkpoints
- Wiegers, K., “Software Requirements”, Microsoft, 1999