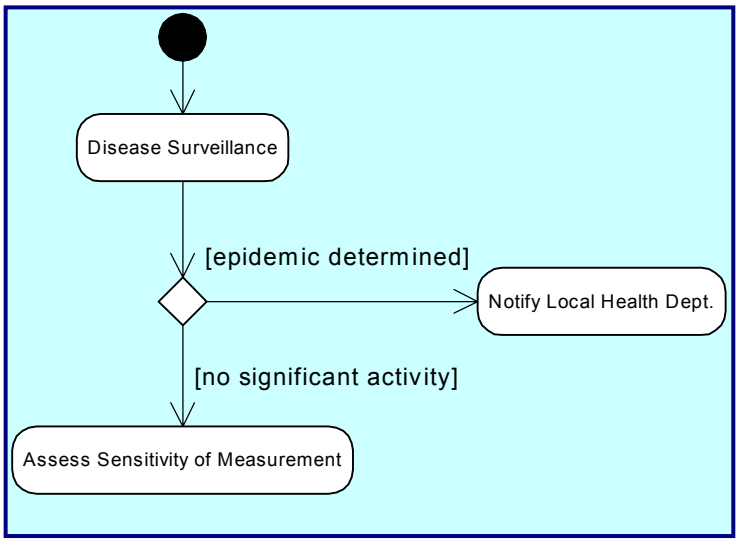
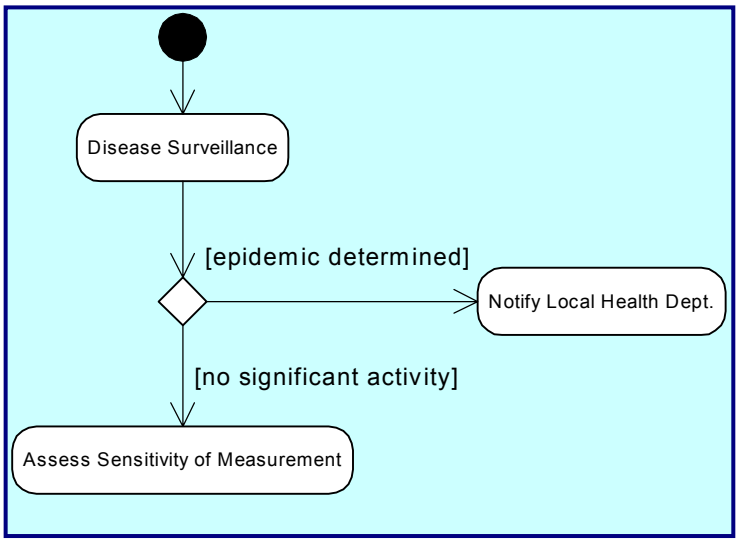
Define your project and system requirements - A guide for program managers and directors



Georgia Department of Human Resources
Division of Public Health
Office of Health Information & Policy

Define your project and system requirements - A guide for program managers and directors



A synopsis of Applying Use Cases,

Prepared by Gordon R. Freymann, MPH
Office of Health Information & Policy
Division of Public Health
Georgia Department of Human Resources
Publication # DPH02.92

Define your project and system requirements - A guide for program managers and directors

Publication # DPH02.92

This guide is a synopsis of Applying Use Cases: A Practical Guide written by Geri Schneider and Jason P. Winters (1998) and published by Addison-Wesley, Reading MA.

Prepared by Gordon R. Freymann, MPH
Director, Data Policy/Analysis Unit
Office of Health Information & Policy
Division of Public Health
Georgia Department of Human Resources

Preliminary release March 14, 2001
Final release 1.0 July 2, 2001

Synopsis of Applying Use Cases

"Use Cases and Activity Diagrams are useful in the Requirements Analysis Phase of a project, and are not about programming language!"
-anonymous business analyst

INCEPTION PHASE
Your Objective: Decide what you're going to do, and answer "does it make sense to continue with this project?"

The inception phase includes the following four steps:

- Project Description
- Market Factors/Risk Analysis
- Risk Factors
- Assumptions

1. Project Description

Begin by writing a description of what you wish to achieve. One paragraph will do. Your description should be brief, and about what you want to accomplish (not how it will get done).

2. Market Factors/Risk Analysis

Begin by writing what market factors, good or bad, will influence your project.

2.1 Consider:

Who or what is the competition
What technologies are being depended upon
Market trends that influence project
Number of expected users
Future trends you are depending on
Number of transactions per time frame
Expected duration of various functionalities
Legacy systems you must interface with

3. Risk Factors

What can go wrong?

3.1 Consider:

Lack of user acceptance
Dependence on a technology that changes
Too many users
Team not experienced enough
Supplier can't or won't deliver product depended upon

4. Assumptions

Assumptions are decisions based on little data. Record decisions made and why you made them.

What you have at this point are two deliverables:

- ✔ Project Description
- ✔ Risk Analysis

5. Identifying System Boundaries

A 'system' is whatever you are planning to create. This could include software, hardware, or your system may only include several processes. Planning, scheduling, and documentation are all included in your project.

Your charge: Identifying system boundaries means defining what's inside your system (i.e. what you have to create) and what's outside your system (you don't have to create, but what you have to interface with).

You find your boundaries by identifying the *actors* and *use cases*.

5.1 Identify Actors

Actors are anything that interfaces with your system; people, software, data stores. One physical person may assume several roles, thereby be represented by several actors.

To find actors in your system, ask:

Who uses the system?
Who installs the system?
Who starts up the system?
Who maintains the system?
Who shuts down the system?
What other systems use this system?
Who gets information from this system?
Who provides information to this system?
Does anything happen automatically at a preset time?

Remember: This does not necessarily refer to software!

For something to be declared an actor – you can not have control over it. For example, an actor might be a customer; you can't control the customer's behavior.

5.1.1 Aggregate entities that interface with you in the same way as one actor.

For example, Federal Express, UPS, and DHL are simply one Shipping Company actor. Or, Vital Records Branch Director, Epidemiology Branch Director, etc. are 'Branch Director.'

5.2. Identify Use Cases

Use Cases describe the things actors want the system to do; such as Track an Order or Cancel an Order. Use Cases are started by an actor. They are a way actors use your system. Ask yourself:

What functions will the actor want from the system?
Does the system store information?
What actors will create, read, update, or delete that information?
Does the system need to notify an actor about changes in its internal state?
Are there any external events the system must know about?
What actor informs the system about those events?

Other use cases to consider are: start up, shut down, installation, training, and maintenance. In public health, some use cases would be Define Goals and Objectives, or Define Data Quality Assurance.

5.2.1. Graphic Representation

Actors are represented by stick figures. Use Cases are represented by ovals. The boundary of your system is represented by a rectangle surrounding your use cases. Actors are outside your rectangle because they are (always) outside your system. Use

cases are (always) inside your system. Solid lines connect actors to their use cases. See Figure 1.

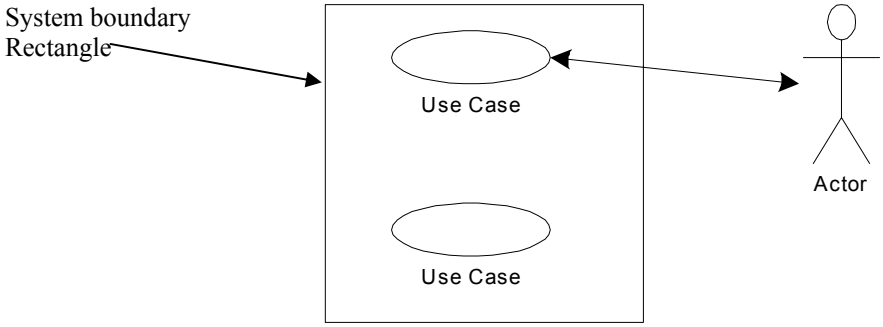


Figure 1

5.3. Describe Actors and Use Cases

Each actor needs a descriptive name and brief description. (1-2 sentences)

Each use case needs a descriptive name and brief description.

Review your system description, market factors, risk factors, assumptions: Are all users represented as actors? Are all system functions represented as use cases?

For example, Order-Processing Uses Cases (your **system**) may include Place Order, Get Status on Order, Send Catalog, Cancel Order, Return Product, Register Complaint, Packages Ready for Delivery, Calculate Postage, Print Mailing Label, Give Product Information, Update Product Quantities, Back-Ordered Items Received, Charge Account, and Credit Account **use cases**, and Customer, Shipping Company, Clerk, Inventory System, and Accounting System **actors**.

5.4. Handling Time

Time could be treated as an actor, or, could be treated as part of your system. If you choose the latter, a use case starts itself at some time ("violating" one of our previous rules – but this can be an exception).

5.5. Boundary Problems

What if some requirements are handled by one or more actors? You will then need to determine if that actor is really part of your system.

5.5.1. Clarify ownership of each requirement

To determine if a requirement is inside or outside your system (such as asking whether Insurance (a requirement) will be provided by your system (and therefore be one of your use cases) or will be provided by a shipper (an actor)), ask:

Are the requirements necessary for this system?
Are these requirements something this system would logically do?
How do the new requirements affect our current risk analysis?
Can these requirements be handled by one of the current actors to our system? I.e. is someone else responsible for them?
Are these requirements something our customers would expect our system to do?

5.6. Scope the project

Categorize your requirements into Must Have, Should Have, and Could Have.

Remember, the hardest part of a project is getting to release 1.0. Going from 1.0 to 1.1 is a lot easier.

What you have at this point are five deliverables:

- Project Description
- Risk Analysis
- ✔ **Use Case Diagram**
- ✔ **Description of Actors and Use Cases**
- ✔ **Project Proposal**

* Congratulations! You have completed the Inception Phase *

ELABORATION PHASE
Answers: Can project be completed technically within time frame?

6. The Complete Use Case – Step 1: Primary Scenarios

A complete use case includes basic functionality, alternatives, error conditions, preconditions, and postconditions. Conditionals, branching and loops may also be part of a complete use case.

6.1. Example 1.0 Place Order use case

Precondition: A valid user has logged into the system

Basic Path
1. The use case begins when the customer selects Place Order
2. The customer enters his or her name and address.
3. The customer enters a product code
4. The system provides a product description and price
5. The customer enters payment information
6. The customer will select <submit>
7. The system will verify the information, save the order as pending, and forward payment information to the accounting system.
8. When payment is confirmed, the order is marked Confirmed, an order ID is returned to the customer, and the use case ends.

Alternate Paths
In step 7, if any information is incorrect, the system will prompt the user to correct the information.

Postcondition: The order has been saved in the system and marked confirmed.

Note that for the use case to have taken place, the postcondition must be true, no matter what errors happen.

6.2. Alternate Paths or IF statements
Either are acceptable. One could write an IF statement in the basic path above (example 1.0) instead of the alternate path (i.e. Branching). Use alternate paths if the branching is complex, or for things that could happen at any time. An example would be a user that decides to cancel the order anywhere during the process. Thus insert:

> Alternate Path
> At any time before selecting <submit>, the customer can select Cancel. The order is not saved and the use case ends.

6.3. Repetition
Use a FOR statement if you find a function must be repeated. In the above example 1.0, one could insert at step 4:

4. For each product code entered
    a. The system will provide a description and price

7. Scenarios

One particular path through a use case is called a **scenario**. Various scenarios are possible, such as an order arriving complete with payment; an order without payment; or an order without a shipping address. These are each different paths. Primary scenarios describe the basic functionality of a use case. Secondary scenarios describe alternates and error conditions for the use case.

7.1. Primary Scenario
Written as if everything goes right. Written from the actor's point of view. Can be thought of as a list of steps to go through to accomplish the use case. Now you get to the how; for example, a scenario describes how an order gets to shipping.

> Task: To validate, take each actor and walk the actor through each step in each use case in the system.

7.2. Guidelines for correctness and completeness
Each step of the scenario should be a simple, declarative statement. By default the steps will be in the order of time. However, sometimes steps could happen in any order. In such a case, make it clear in the description. Resist the temptation to get too detailed just yet.

By definition, scenarios are written from an actor's point of view. Most use cases start and end with an actor.

8. Other Requirements
Some requirements, such as timing and size, may not be visible to the actors. Having a requirement of 'timely service delivery' does not turn into a use case, but affects a lot of use cases. Include these requirements under a **Special Requirements** heading after any **postconditions** in your use case primary scenario steps.

9. Presentation styles
Scenarios can be written as informal text, as numbered steps (e.g. example 1.0), or as psuedocode. Choose one based on your audience. Remember, use case form is: precondition, flow of events, postcondition, special requirements.

At this point your deliverable is:
- ✔ Detailed primary scenario for each use case.

10. The Complete Use Case – Step 2: Secondary Scenarios

Alternate paths and error conditions can be written as secondary scenarios, rather than being imbedded in primary scenarios.

Again, there is no need to write down every thing that could go wrong in detail. For example, in a situation where deposited money doesn't go into your account, the reasons could be that the customer closed the account, there could be insufficient funds, etc. All you need to document is the fact that money didn't go into your account.

To proceed, one could go line by line in each primary scenario, and ask what could go wrong or be done differently. Each time you get a different answer, that's a new scenario.

10.1. Finding Secondary Scenarios

Ask: Is there some other action that can be taken at a particular step (an alternate scenario).
Ask: Is there something that could go wrong at a particular step (an exception scenario).

10.2. Name each secondary scenario

Simply list the name of all secondary scenarios at this point. Such as Payment not there, Order incomplete, Customer can't login, etc.

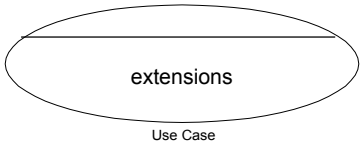
Some significant secondary scenarios should have detailed steps written as was done for the primary scenario.

11. Use Case Description Revisited

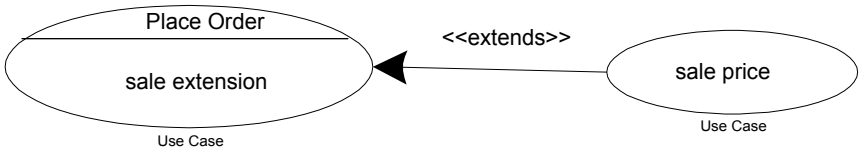
As you write out secondary scenarios, you will find commonality in the various use cases that you want to abstract out into a common place. Or you may want to extend a use case without changing the original description. To take advantage of commonality, you can use some techniques: Extends, Uses, Inheritance, and Interfaces.

11.1. "Extends"

Extends are used when you have an optional sequence of events you want to include in the use case. Using a public health example now, **service data collection** is an extension of **core data requirements**. Or, in the Place Order use case, an extension point might be Sale Item.



So, after getting the price for an item (step 4 in example 1.0), add an extension point for *sale* price. Write a use case describing what will happen at the extension and give it a name. Graphically, this use case is represented as:



Of course there can be more than one extension to a use case. In our place order use case example, there may be a 'seasonal sale price' extension as well as a 'frequent customer discount' extension. It is important to note that the use case must work whether an extension occurs or not; that is to say, a use case does not know if it has been extended.

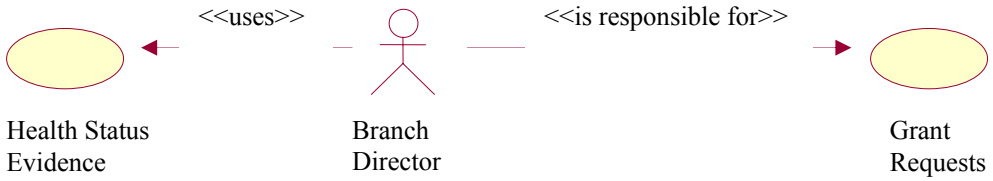
11.2. "Uses"

"Uses. Sounds weird, but it simply is what it is: Uses as in 'I use a hammer to drive a nail.'"

If you find yourself 'copying and pasting' the same information over and over, you probably have something generic you can reuse. Such behavior can be handled with a uses relationship.

For example, a step within an order processing system is to search for an order by customer name or order ID. This searching is done from many use cases (situations) such as Get Status on Order, Cancel Order, and so on. Instead of repeating the steps to search for an order over and over, simply create a use case out of it, and use the (search for order) use case in other use cases (e.g. cancel order) as necessary.

In public health then, data quality policies are created once, and used by many data management activities throughout the Division. Another example is illustrated below:



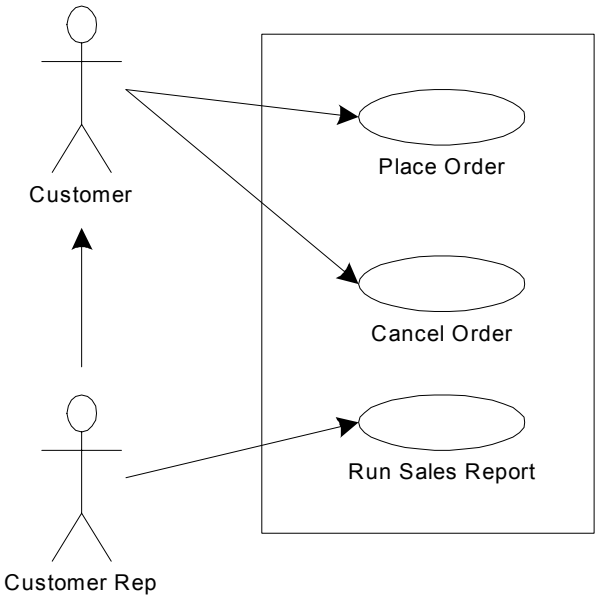
What this says is that the Grant Requests use case is incomplete (unable to be executed) without using health status evidence to drive what type of grant to request.

A use case can use any number of other use cases. You can also have many levels of use cases; Health Status Evidence use case could itself have one or more use cases.

Note: the latest version of the UML replaced the term 'uses' with 'includes.'

11.3. "Inheritance"

Simply, if an actor interacts with the same uses cases in the same way as another actor, then that actor is said to inherit the use cases of the other actor. Inheritance between actors means that one actor fills the same roles as another actor. See the following example:



The customer rep inherits the 2 use cases of the customer (In other words, the customer rep could also be a customer). In addition, the customer rep runs sales reports, which is unique to the customer rep.

11.4. "Interfaces"
Interfaces can be defined for actors or use cases. They tell us what we expect the entity to do.

For example, what behavior do you expect from the Place Order use case? See the following example:

Interface example and expected behavior for Place Order use case

Place Order Get Product Description and Price Return Description, Price Add Price to Total Price, return Total Submit Order, return Order ID.

Similarly, 'what behavior do you expect from your customer?' See the following:

Interface example and expected behavior for Customer Actor use case

Enter Name and Address, return Name, Address Enter Product Code, return Product Code Enter Credit Card Information, return Card Number, Expiration Date.

You may be tempted to aggregate several interfaces that overlap into one large interface. For example, you could add the Order Search interface with the Customer Actor Interface. However, you may want to retain flexibility by creating a large number of small interfaces instead.

At this point, you have the following Elaboration Phase Deliverables:
- Detailed primary scenarios
- ✔ **Secondary scenarios**

ELABORATION PHASE CONTINUED...
Activity Diagrams

12. Diagramming Use Cases
Written descriptions can become hard to understand, especially if there is a lot of branching and exceptions. This section will examine simple ways to diagram the steps in a use case.

12.1. Activity Diagrams
Technical Definition:
"A subset of a state diagram where the states are all action states and the transitions are automatic."

....or a definition for the rest of us:
"...simply a way to graphically show the flow of events that describe the things you want the system to do."

...the things we want the system to do are, of course, use cases.

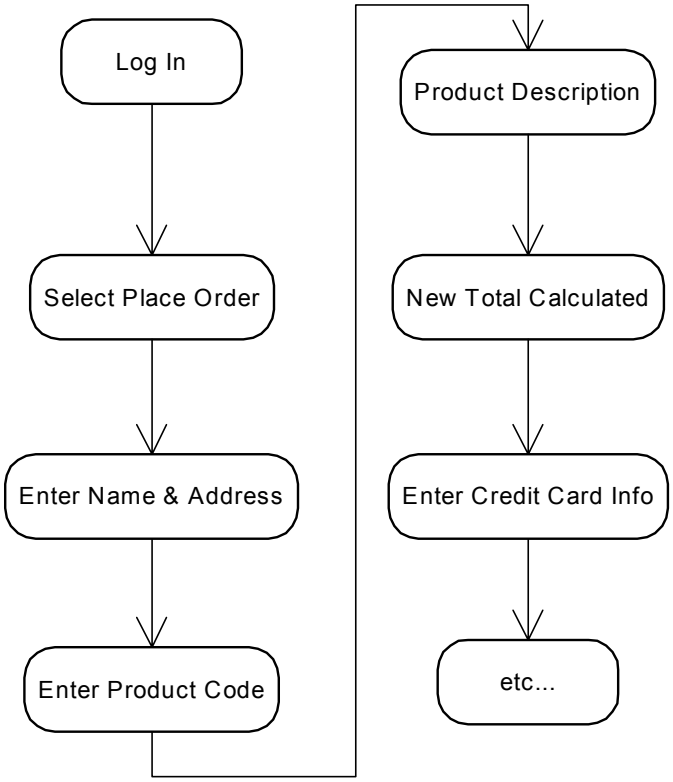
Each activity in a use case is represented by a rounded rectangle: 

. Transitions are represented by arrows ⟶ . Recall the Place Order use case:

Place Order use case

1. Customer selects place order
2. Customer enters name and address
3. Customer enters product codes for products to be ordered
4. System supplies a product description and price
5. etc.....

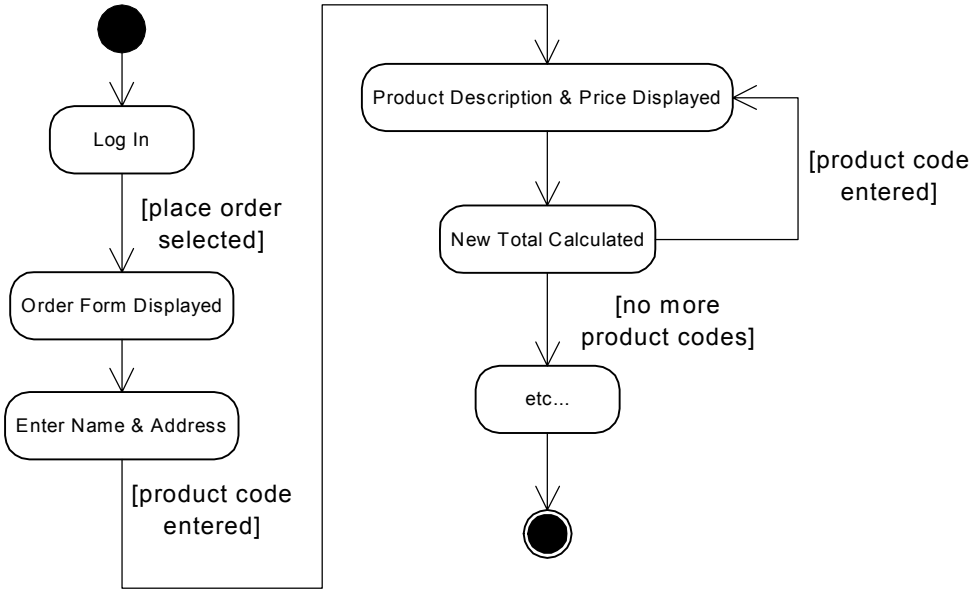
The Place Order use case activity diagram would begin to look like....



12.1.1. <u>You may realize that your diagram doesn't tell the whole story</u> – that is, there may need to be *conditions* added to your diagram, such as showing that a customer can place more than one order, or that the order is confirmed only if accounting approves...

See the next figure, and how conditions are marked in square brackets. A *condition* means that **the path can be taken only if the condition is true**.

A partial activity diagram with Conditional Acts would look like....(see following page)...

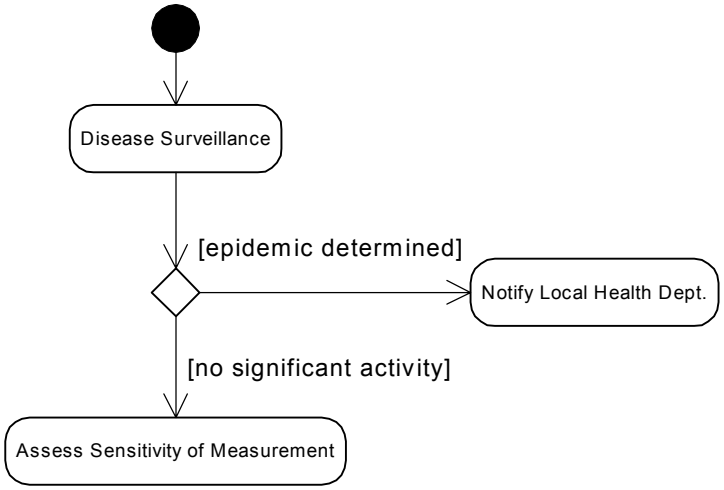


Notice the bullet (starting point) and the bull's-eye (stopping point).

12.1.2. Decision Points

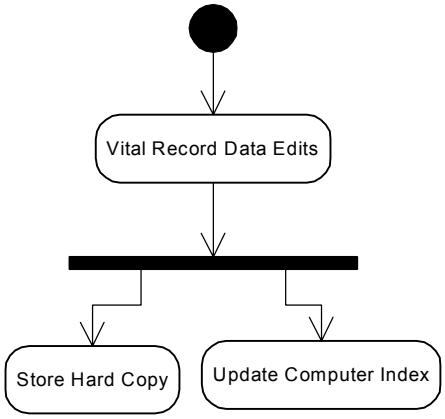
Often, there will be several options a user could take, which require a decision. Decisions are represented by **diamonds**.

You can have as many arrows coming to or from a diamond as necessary, but they all must have conditions.

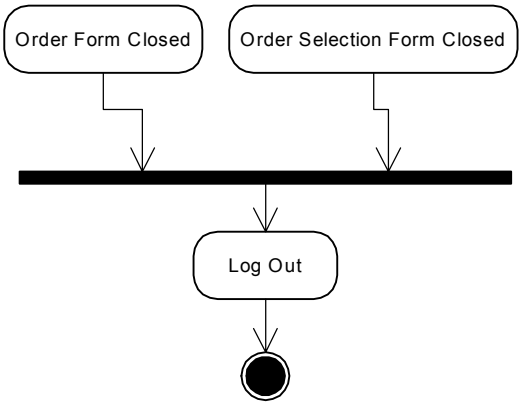


Now, there may be situations where more than one activity takes place at the same time (so far, we've only shown diagrams that allow one thread of activity). This requirement is addressed by a *fork*....(see example on following page)...

Forks (example at right) can have as many paths coming out of the fork as you need.

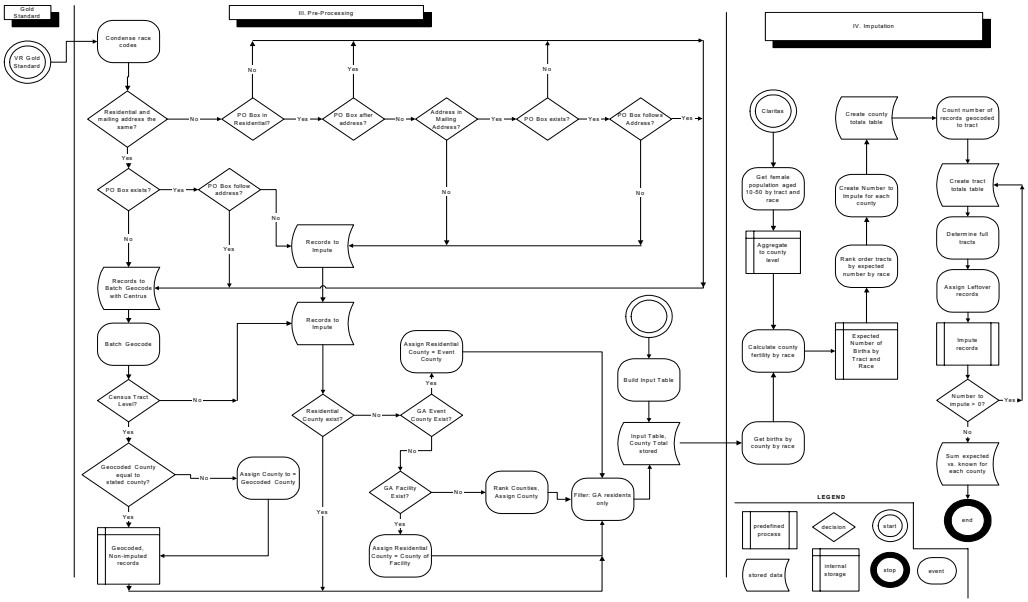


Conversely, a *join* (example below) allows multiple paths to come together.

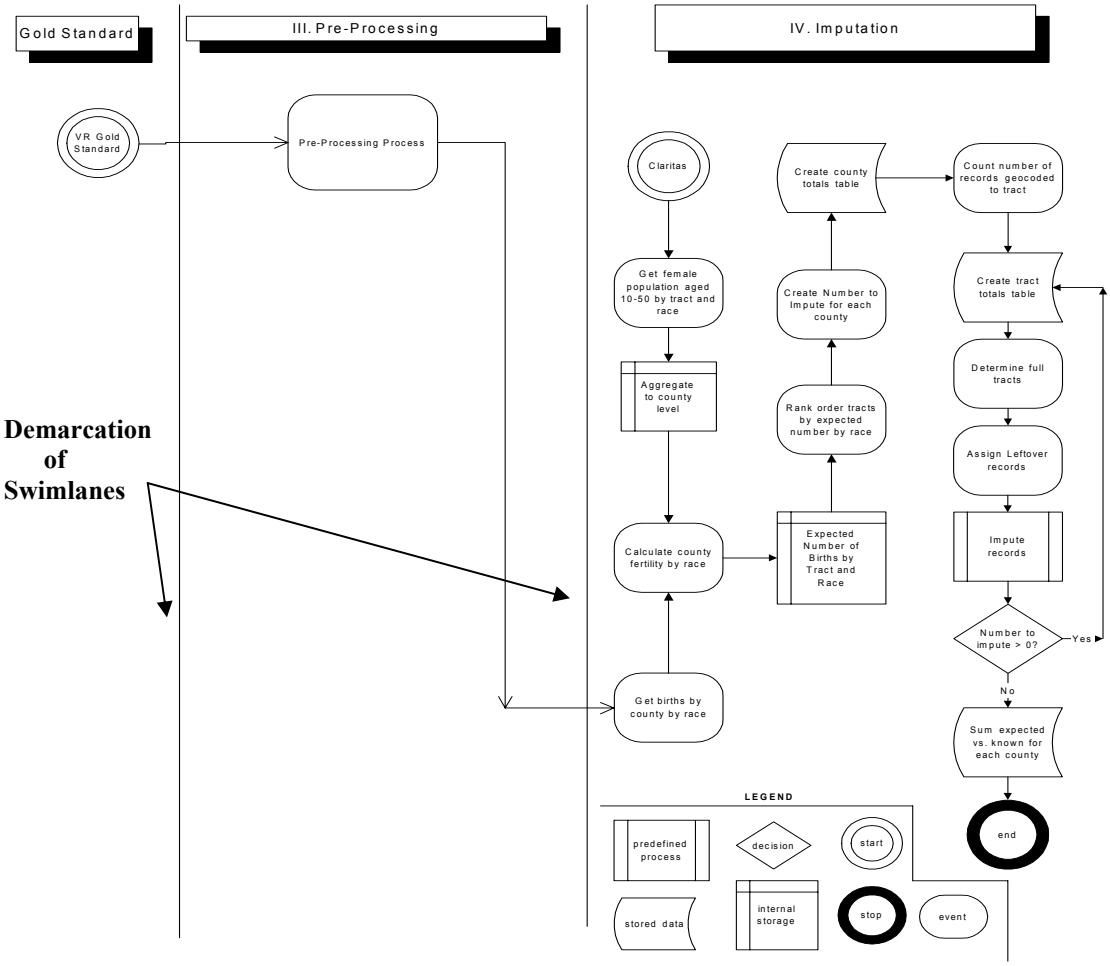


Note: If you find that your diagram is getting really big and confusing, split it into pieces. In fact, the rounded rectangle representing the whole activity could take the place of a **whole diagram**. You might say it depends on the scale at which you examine your business; much like changing the scale on a map shows varying levels of granularity.

The following activity diagram was done by the Office of Health Information and Policy to explain the process we use to geocode vital records data:



As you can see, it seems somewhat complex. It is divided into three sections called *swimlanes*. Swimlanes are indicated by the two vertical lines separating the activity diagram into three sections. As mentioned earlier, a rounded rectangle could be used to represent an entire process rather than just one activity within a process. Compared with the preceding figure, the diagram below uses only one rectangle to represent the entire "pre-processing" diagram in the second swimlane:



At this point you have another Elaboration Phase deliverable:

- Detailed primary scenarios
- Secondary scenarios
- ✔ **Activity diagrams**

Congratulations! You have just completed the basics of working with use cases and activity diagrams!

You may wish to refer to Appendices A and B to help with describing use cases and scenarios.

For further development, there are Architecture and Project Plan deliverables in the Elaboration Phase that remain. Afterwards, there are the Construction (iterative process of analysis, design, implementation, and testing) and Translation (beta testing, performance tuning, plan for rolling out to users) phases. Please refer to Applying Use Cases by Scheider and Winter for further discussion and examples.

This synopsis was prepared from information in

Schneider, G. & Winters, J. P. (1998). Applying Use Cases: A practical guide. Reading, MA: Addison-Wesley

and was prepared by Gordon R. Freymann, MPH, Office of Health Information & Policy, Georgia Division of Public Health.

Appendix A: USE CASE TEMPLATE

Use Case Name

<A brief description of the use case. Usually a paragraph or less.>

Actors

<A list of the actors who communicate with this use case.>

- <Actor Name – brief description.>
- <Actor Name – brief description.>
- <Actor Name – brief description.>

Priority

<How important is this use case to the project? Must have, should have, would like to have, you're dreaming.>

Status

<What is the status of this use case? High level description, detailed description, scenarios complete, coded, tested, etc.>

Pre-Conditions

- <Something that must be true before the use case begins.>
- <Something that must be true before the use case begins.>
- <Something that must be true before the use case begins.>

Flow of Events

Basic Path

1. <Write the first step in the use case.>
2. <Write the next step in the use case.>
3. <Write the next step in the use case.>

Alternative Paths

<Write any alternatives to the basic path.>

Post-Conditions

- <Something that must be true when the use case ends.>
- <Something that must be true when the use case ends.>
- <Something that must be true when the use case ends.>

"Used" Use Cases

- <Use case Name>
- <Use case Name>
- <Use case Name>

Activity Diagram

<An activity diagram of the flow of events, or some significant or complex part of the flow of events.>

User Interface

<For systems that interface with people, include a description of the user interface, possibly using storyboards.>

Scenarios

<List the scenarios of the use case. Include a brief description of each one.>

- <Scenario Name – brief description.>
- <Scenario Name – brief description.>
- <Scenario Name – brief description.>

Sequence Diagrams

<If you do not have separate scenario documents, you may also wish to include a sequence diagram for each Scenario.>

Subordinate Use Cases

<If the use case has subordinate use cases, include a use case diagram of those subordinate use cases here. Or make a list of the subordinate use cases by name, and tell which subsystem is responsible for each one.>

View of Participating Classes

<A collaboration or class diagram showing all the classes whose objects interact to implement this use case. Include the interfaces to the use case and the classes that implement the interfaces.>

Other Artifacts

<Anything else you might want to include. Possibly an analysis model, a design model, or test plans.>

Other Requirements

<This section includes a list of non-functional requirements that affect the use case.>

Appendix B: SCENARIO TEMPLATE

Scenario Name

<A brief description of the scenario. Usually a paragraph or less.>

Use Case

<Use case Name this scenario is part of.>

Priority

<How important is this scenario to the project? Must have, should have, would like to have, you're dreaming.>

Status

<What is the status of this scenario? High level description, detailed description, coded, tested, etc.>

Flow of Events

Basic Path

1. <Write the first step in the scenario.>
2. <Write the next step in the scenario.>
3. <Write the next step in the scenario.>

Alternative Paths

<Write any alternatives to the basic path.>

Activity Diagram

<An activity diagram of the flow of events, or some significant or complex part of the flow of events.>

Sequence Diagrams

<You may also wish to include a sequence diagram for the scenario.>

Other Artifacts

<Anything else you might want to include. Possibly an analysis model, a design model, or test plans.>