BIO

PAPER

F2

November 8, 2002
10:15 AM

TESTING FROM USE CASES USING PATH ANALYSIS TECHNIQUE

Naresh Ahlowalia
Object System Group

International Conference On
Software Testing Analysis & Review
November 4-8, 2002
Anaheim, CA USA

Naresh Ahlowalia

Naresh Ahlowalia has over 20 years of experience in software development, software testing and systems testing. He has worked on many different tools like Rational, Mercury and McCabe. Besides commercial applications he has worked on Military and Avionics systems. He has interests in all aspects of software testing, including automation and performance testing. He is experienced in Waterfall as well as Incremental and Iterative life cycle methodologies.

He has developed an approach for analyzing and testing from use cases, called Use case path analysis. This graphical approach help determine the use cases flow problems and also derive test cases from the use cases. This technique has been effectively used in two projects at Fannie Mae. Both projects were successfully put in production and were finished in time and in budget.

Naresh has a BS degree in Electrical Engineering from Indian Institute of Technology New Delhi, India and a MS degree in Electronic System Design from Cranfield Institute of Technology, Bedford UK. He has been working as a consultant at Fannie Mae, Herndon Virginia and can be contacted by email at nahlowalia@hotmail.com.

Testing from Use Cases, using Path Analysis Technique

Naresh Ahlowalia

Abstract

Use cases have become an industry standard method of specifying user interaction with the system and hence have become part of the requirements definition phase of a software project. Use cases are used to derive, construct and validate interfaces, objects, relationships and processes in an application. They can also be very effectively used for derivation of test cases for the software application. The author has been involved in number of software projects, which were use case driven and were required to come up with test cases to effectively test the application. Out of sheer necessity and in the absence of any defined methodology, author devised a testing methodology called Use Case Path Analysis. This methodology has been successfully applied to test two EJB based projects at Fanniemae. Author has also been involved in teaching this to number of testing professionals and would like to share it with others.

What are Use Cases:

Use case is a description of the interaction between the user and the system. This can also be described as a way of using specific part of application's functionality. A complete collection of use cases describes all the ways that a user can manipulate a software system.
Use cases are a very powerful method of describing requirements. They form basis for

- Software Design
- Test Cases
- User documentation
- GUI design

Use cases are written during the requirement phase of the software project. Use cases are written in a plain English language that user can easily understand. Business analysts that have been specially trained to develop use cases should write them. Rational Rose is a tool that can be effectively used to model and also write use cases. However most use cases are written in MSWord as everyone can easily access them.

Use cases are basically of two types

- Concrete: Concrete use case is the one that stands alone to accomplish a single goal
- Abstract: Abstract use case is used by more than one use cases but may not have a specific user oriented goal

Use case has following elements

- Name
- Description
- Actors/Role(s)
- Preconditions
- Basic Course
- Alternate Courses
- Exception Courses
- Post Conditions
- Issues
- Notes

Example of a simple Use case
A simple example of a use case is given in Appendix A

Path Analysis:
Path analysis is a process specially developed for turning use cases into test cases. This is a tool for the testers, who are required to test applications that have use cases as requirements. Use case can be executed in many possible ways and each thread of execution through a use case is called a path. Each use case path is derived from a possible combination of following use case elements

- Basic Course
- Alternate Courses
- Exception Courses
- Extends and Uses Relationships

Thus any use case has multiple paths and each path through the use case is a potential test case.

Path analysis is a scientific technique to identify all possible test cases without relying on intuition and experience. It reduces the number of test cases and the complexity and bulk of test script by moving the non-path related variability of test cases to test data. It simplifies the coupling of test cases to test data. As an example if we have a use case with 10 paths and each path needs to be tested with 5 sets of data, in traditional methodology we would have written 10x5 = 50 test cases
Using path analysis we write 10 test cases (one per path) and 10 data tables (each with 5 columns). The advantages can be summarized as

- Better coverage through scientific analysis
- Both positive and negative test cases can be easily identified through path analysis
- Easier coupling of path and data
- Reduction of test cases
- Better management of testing risks
- Extremely effective in identifying and correcting use case errors

Path Analysis Process
The path analysis process consists of following 4 major steps

- Draw flow diagram for the use case
- Determine all possible paths
- Analyze and rank the paths
- Decide which paths to use for testing

Draw flow diagram for the use case:
Use case is a textual document and hence cannot be easily analyzed. The first task is to convert this into a graphical format called flow diagram. Let us define some concepts

- Flow Diagram: Graphical depiction of the use case
- Node: The point where flow deviates or converges
- Branch: Connection between two nodes. Each branch has a direction attached to it. Node is also where more than one branch meets.

In order to illustrate this process, we will take the example of the use case that is attached at Appendix A.
Each use case has only one start and can have multiple end points. Using UML terminology the start is indicated by a plain circle and a circle with a dot inside indicates end.

Symbol for Start  Symbol for  :

In order to draw a diagram for the use case following steps should be followed

- First draw the basic flow
  Identify Nodes
  Combine sequential steps into one branch
  Annotate the branches with the text summarizing the action in those branches
  Connect the nodes indicating the direction of flow
- Repeat the step above for each alternate and exception flow

The complete step by step process is illustrated in the attached diagram at Appendix B. The use case example of Appendix A has been used to illustrate the process. As explained above the flow diagram is an excellent tool to identify the use case flow and other problems in the early stages. This feature of the process can save lot of time and effort in the earlier stages of the software process. Appendix B also covers how to identify use case problems and then correcting them early on in the software process.

Determine all possible paths:

As discussed earlier use case path is a single thread of execution through the use case. The path determination process is basically very simple and can be stated as " Beginning from the Start each time, list all possible ways to reach the end, keeping the direction of flow in mind". As you can see from the diagram that there could be a large number of paths through the use cases. In some complex use cases, especially when there is lot of feedback branches, there could potentially be a very large number of paths through the use case. The aim here is to list all these paths. However if the number of paths are too many, it may be necessary to use judgement at this stage itself. The process consists of following steps

Path ID Designation: Each path should be suitably designated as P1, P2 etc.
Path Name: This is the sequence of branch numbers that the path comprises of. For the example given above path ID P2 has a path name of 2,3,4,5.
Path Description: This is a textual description of the complete sequence of user actions and system responses taking place in that path. It is a good practice to describe the path in good plain English that is meaningful. The path description is ultimately nothing but a description of the test case itself. All this information should be combined in a suitable table also called Table of Paths.

Analyze and rank the paths

For simple use cases, which have about 10 separate paths, it is straightforward to select all paths for testing and hence this step can be skipped. However for some complex use cases, the number of possible paths can easily exceed 50, in which case it may be necessary to select only limited paths for testing. Quite frequently testers may be able to use judgement and experience to select those paths. However in lot of cases, we may need some objectivity and Subject Matter Expert's (SME) guidance. Typically SME is business users who have high stakes in the project. We define two attributes for each path called

- Frequency: This attribute can have a value of 1 to 10, with 1 being least frequent and 10 most frequent. This attribute states the likelihood of this path being exercised by the user.
- Criticality: This attribute describes how critical the failure of this path could be with 1 being least and 10 being most.

Having defined these attributes we can compute a Path factor which is = Frequency + Criticality. This is an equal weight path factor. However we can provide a different weights to these attributes to arrive at a proper Path factor.

Selection of Paths for testing

In order to have adequate test coverage, and minimize the risk of not testing while balancing the resources there is a need to follow some guidelines. They are as follows

- Always select basic flow for testing as
  -It is a Critical Functionality
  -If basic flow fails; there may not be much of a point in testing other paths.
  -Basic flow should be included in Sanity test suites and also in Acceptance testing
- Some other paths are too important to be ignored from functionality point of view. These are generally obvious from the table of paths.
- The path factor should be used for selecting a path among several possibilities that do not meet the above criteria.

The aim here is to select a minimum set of paths which would adequately test the critical functionality while ensuring that all branches have been included in at least one path. The path selection process has been adequately explained in the example at Appendix B.

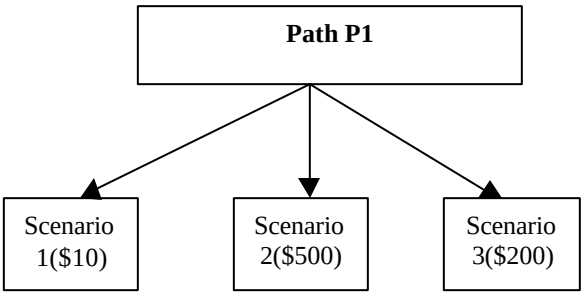
Writing Test Cases from Table of paths

The table of paths provides us with adequate information for testing a use case. The path description is in fact a summary description of the test case itself and even a tester with very little experience can write test cases from that description. Before proceeding further, let me introduce one more concept here called test scenario. Any path with a specific test data becomes a test scenario. Each path is generally associated with number of test scenarios, to adequately test the data width.

Let me take an example of use case called Withdraw money from an ATM machine. One of the paths in the use case which is a basic flow will have path description some thing like this.

"User inserts his card in the machine, system successfully validates the card and prompts for 4 digit pin. User enters a valid pin. System successfully validates the pin and prompts for amount. User enters a valid amount. System ejects user card and correct amount for collection by the user."

Now this is a happy day path but as we know there may be certain minimum and maximum amount that a user can withdraw from the ATM machine. Let us say it is $10 and $500 respectively. In order to adequately test this path using Boundary Value Analysis (BVA), we need to test this withdrawal for $10, $500 and $200(middle). Thus we need to create 3 test scenarios for the same path.



Relationship between path and Scenario

The point to note here is that all three scenarios have the exact same path through the use case. Several times I have been asked the question, why not test < $10 and >$500 withdrawal in the same path. The reason we do not do it is that such tests belong to a different path, where a “user tries to withdraw <$10 or >$500 and he gets an appropriate message and prompts a user to reenter an acceptable amount. User reenters the correct amount and system then lets the user withdraw the money”

The following guidelines should be followed to create test cases

1. Create one test case for each path that has been selected for testing. As explained previously the path description provides enough information to write a proper test case.
2. Create multiple test scenarios within each test case. I recommend using data tables which has data elements as rows and each column is a data set and also a test scenario. Please see Appendix B where this is explained with reference to the example.
3. ADD GUI details in the steps, if you so desire.

Conclusion:

Use case path analysis is an extremely powerful technique to create test cases from the use case. The test cases generated are realistic and the one that business users are going to encounter while using the system. The above paper illustrates the basic technique. Author has a complete methodology that has been designed around this technique and is readily useable in any environments and with any tool set. It not only saves time and money but also helps in controlling risk. Author may be contacted by email at nahlowalia@hotmail.com and by phone at 703 913 2872.

Use Case: Add/Delete/Modify student information
Purpose/Description: This use case describes the way Registrar can maintain student information system, which consists of adding, deleting or modifying students from the system.
Type: Concrete
Actors/Roles: Registrar
Preconditions:
1. The registrar must be logged on the system (Use Case: Login)
2. System offers registrar choices: Add, Delete, Modify

Basic Course:
1. Registrar chooses to add a student in the system.

Alternate Course: Delete Student
Alternate Course: Modify Student
2. Registrar enters following information
    - Name
    - DOB
    - SS#
    - Status
    - Graduation Date
3. Registrar confirms the information
4. System returns a unique ID number for the student

Post Conditions: System displays the list of students with selection of new student that has been added.

Alternate Courses:
Delete Student:
1. At basic course step 1, instead of selecting to Add a student, registrar selects to delete a student.
2. System requests the student ID
3. Registrar enters student ID

Alternate Course: Invalid Student ID
4. System displays the student information.
5. System prompts the registrar to confirm the student deletion

Alternate Course: Cancel Delete
6. Registrar verifies the decision to delete
7. System deletes the student from the system

Post Conditions: Student name and other data no longer displayed to the user.

Modify Student:
1. At basic course step 1, instead of selecting to Add a student, registrar selects to modify a student.
2. System requests the student ID
3. Registrar enters student ID
   Alternate Course: Invalid Student ID
4. System displays the student information.
5. Registrar changes any of the student attributes
   Alternate Course: Cancel Modify
6. Registrar confirms the modification
7. System updates the student information

Post Conditions: Student is not deleted from the system.

Cancel Modify:

1. At step 5 of Alternate course Modify, instead of confirming the deletion, registrar decides not to modify the student.
2. System cancels the modify
3. Return to step 1 of Basic course

Post Condition: Student information is not modified.

Invalid ID

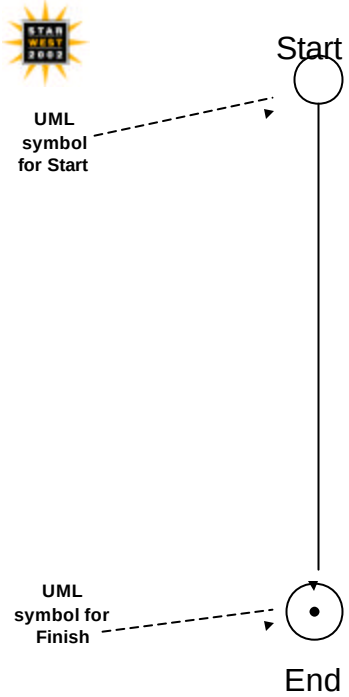
1. At step 3 of Alternate courses (Delete/Modify) system determines that student ID is invalid. (Collaboration case Validate student ID)
2. System displays a message that student ID is invalid.
3. Registrar chooses to reenter student ID.
4. Go to step 1 of Modify/Delete course

Post Condition: Student not deleted/ student information is not modified.

Note: Please see Appendix B which is a Powerpoint presentation file

Appendix B
to
Testing from Use cases using Path Analysis Technique

1



Draw the Basic Flow

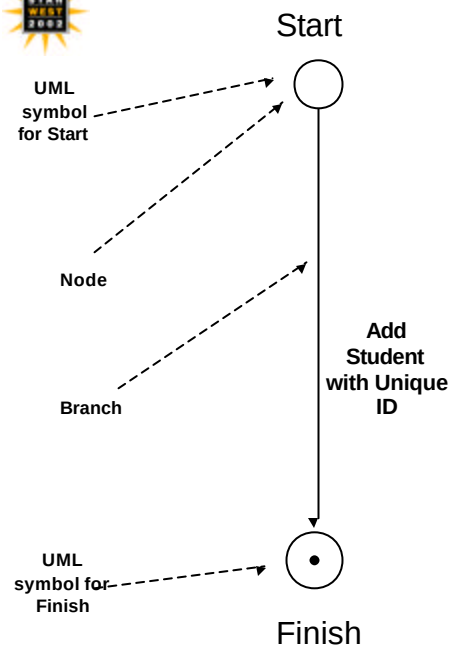
1: Draw Start and End Symbols.

Mark Start and End in the diagram with UML symbols.

2: Connect with a vertical line. This is your Basic Flow.

The Basic Flow steps lie on this line.

2



3: Identify Nodes in the Basic Course.

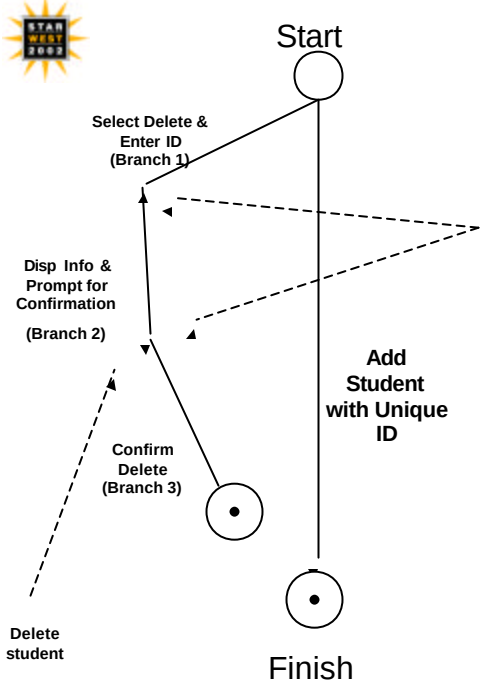
Node is a point where flow deviates. In this case Start is also a Node. **Why?**

If there is more than one Node, identify them.

4: Annotate the Branches. Use text description depending on the contents of the steps.

The Branch is the line/action between the Nodes. It can have multiple steps.

3



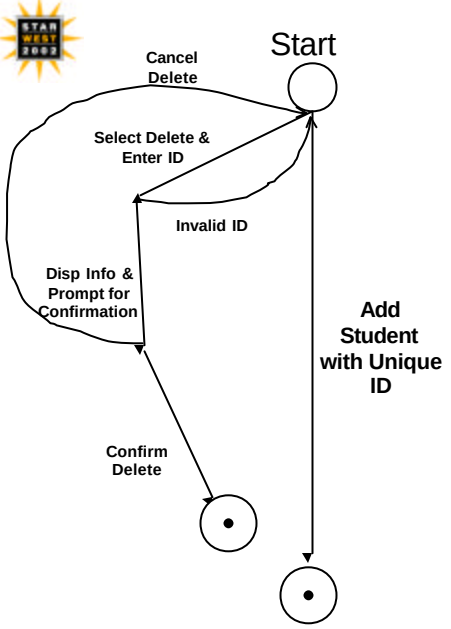
5: Go to first identified Node and start with first Alternate or Exception Course.

6: Identify Nodes within each Alternate Course.

7: Connect the Nodes within each Alternate Course. Arrows should show direction of the flow.

8: Annotate each Branch of the Alternate Course, depending on the contents of the steps.

4

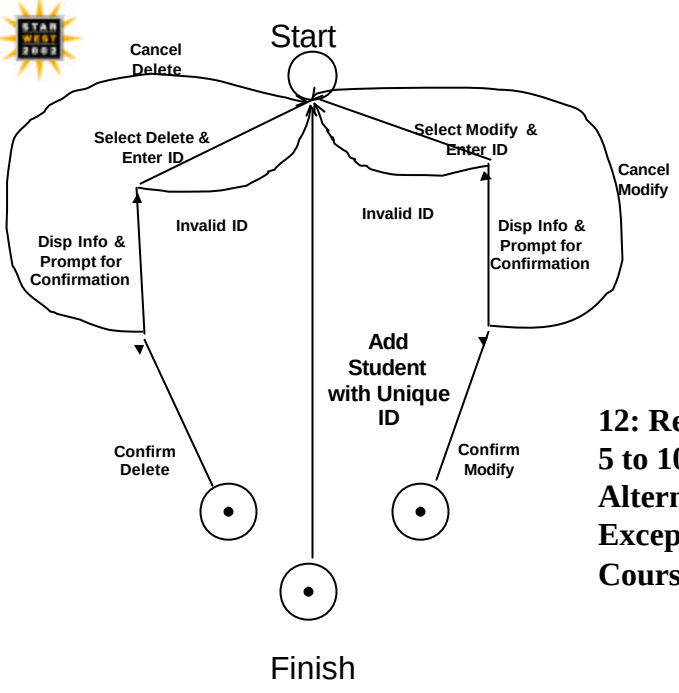


9: Go to each identified Node in the Alternate Course & draw its Alternate & Exception Courses.

10: Connect the Nodes within each Alternate Course. Arrows should show direction of the flow.

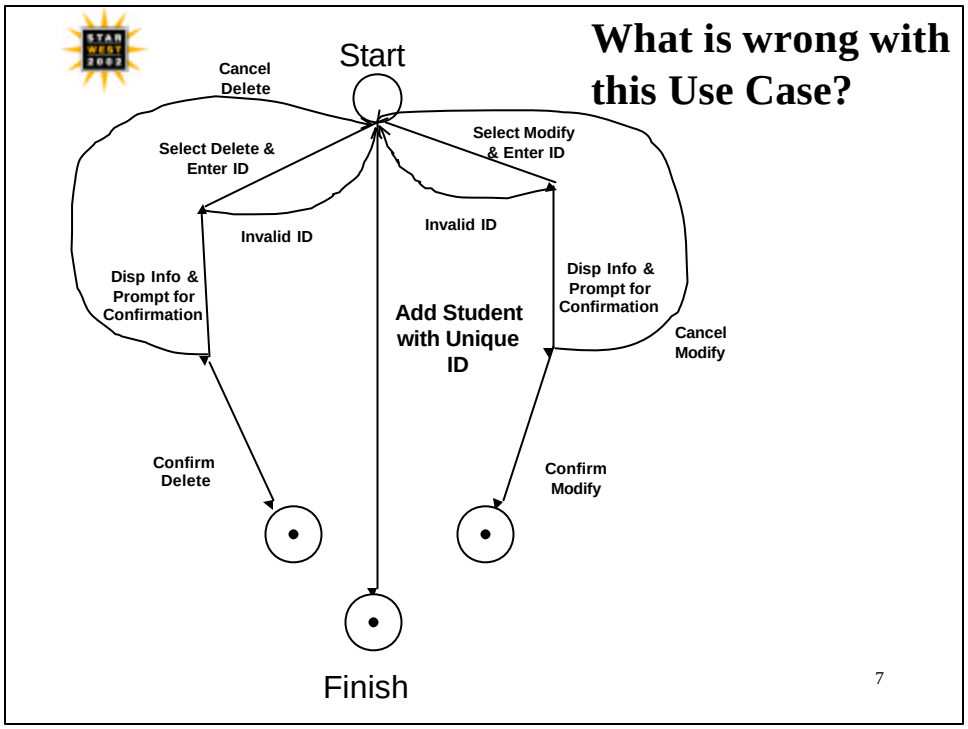
11: Annotate Branches as before.

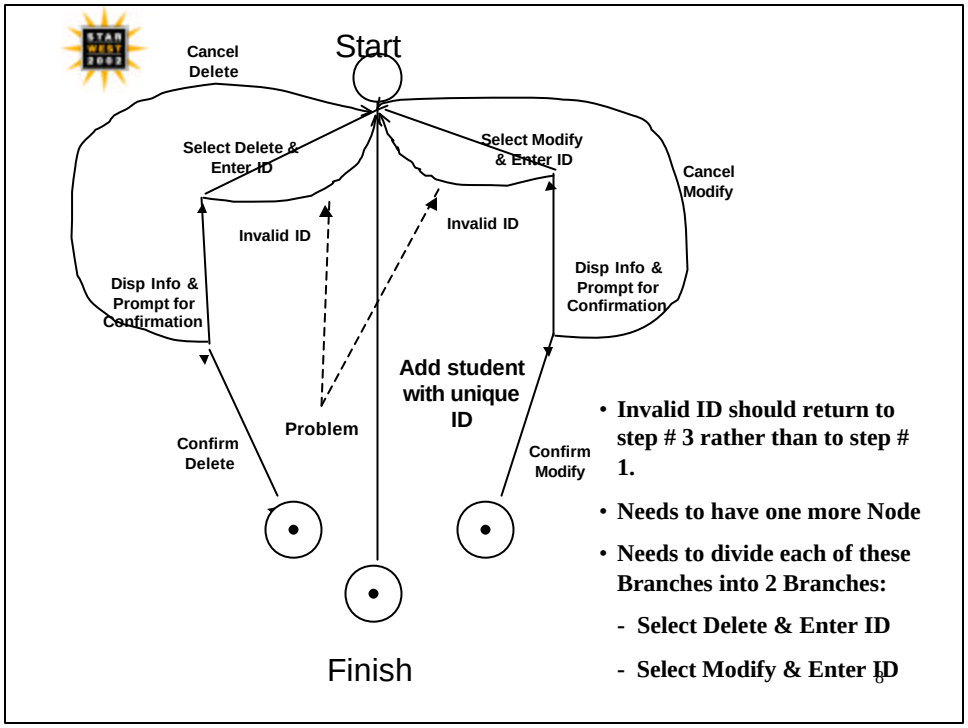
5

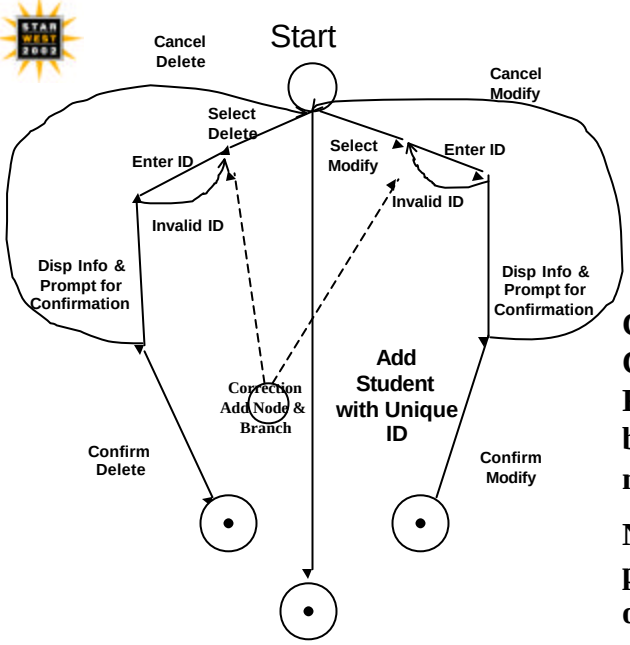


12: Repeat steps 5 to 10 for other Alternate or Exception Courses.

6


What is wrong with this Use Case?
Start
Cancel Delete
Select Delete & Enter ID
Select Modify & Enter ID
Invalid ID
Invalid ID
Disp Info & Prompt for Confirmation
Disp Info & Prompt for Confirmation
Add Student with Unique ID
Cancel Modify
Confirm Delete
Confirm Modify
Finish
7

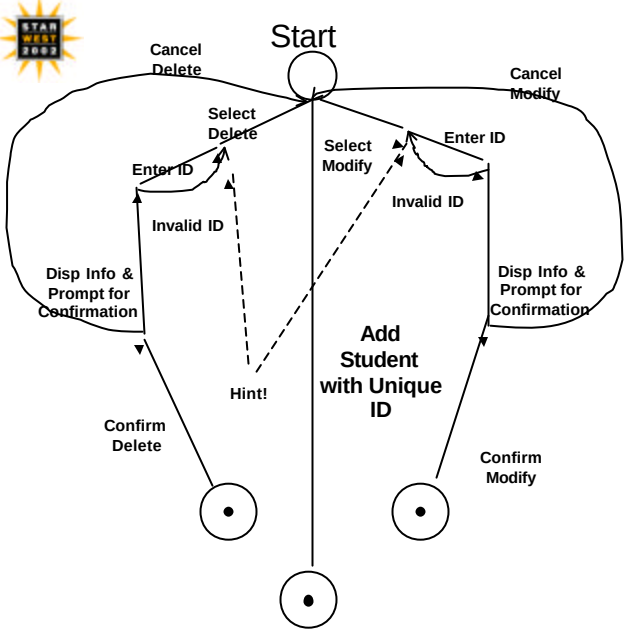

Start
Cancel Delete
Select Delete & Enter ID
Select Modify & Enter ID
Cancel Modify
Invalid ID
Invalid ID
Disp Info & Prompt for Confirmation
Disp Info & Prompt for Confirmation
Add student with unique ID
Problem
Confirm Delete
Confirm Modify
Finish
• Invalid ID should return to step # 3 rather than to step # 1.
• Needs to have one more Node
• Needs to divide each of these Branches into 2 Branches:
- Select Delete & Enter ID
- Select Modify & Enter ID
8



Correct the Use Case and the Flow Diagram by showing two more Nodes.

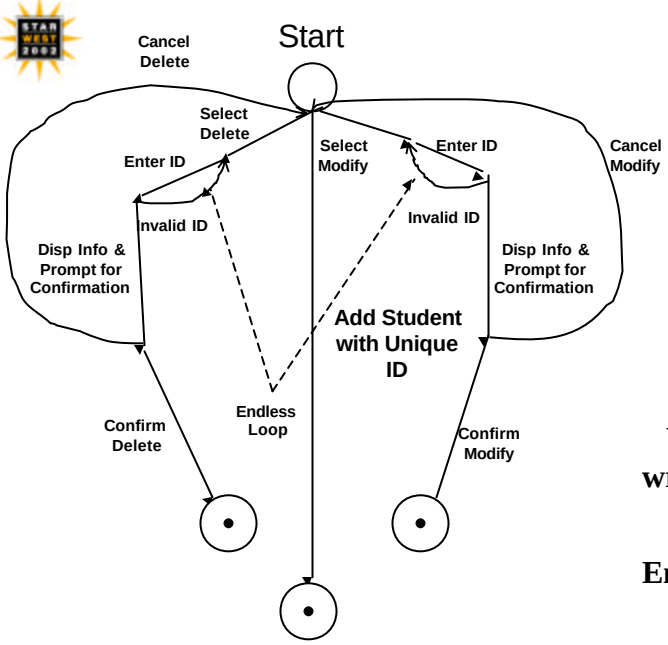
Node is also a point where two or more flows meet.

9



What else is wrong with this Use Case?

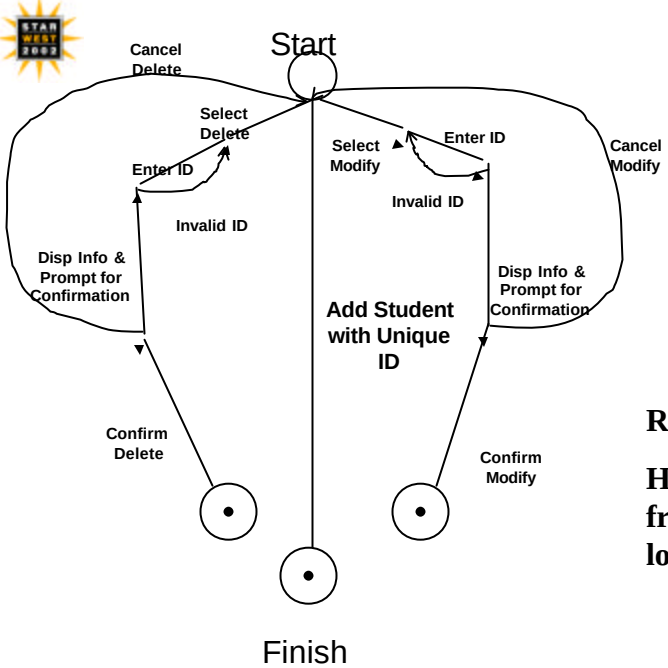
10



What else is wrong with this Use Case?

Endless Loops?

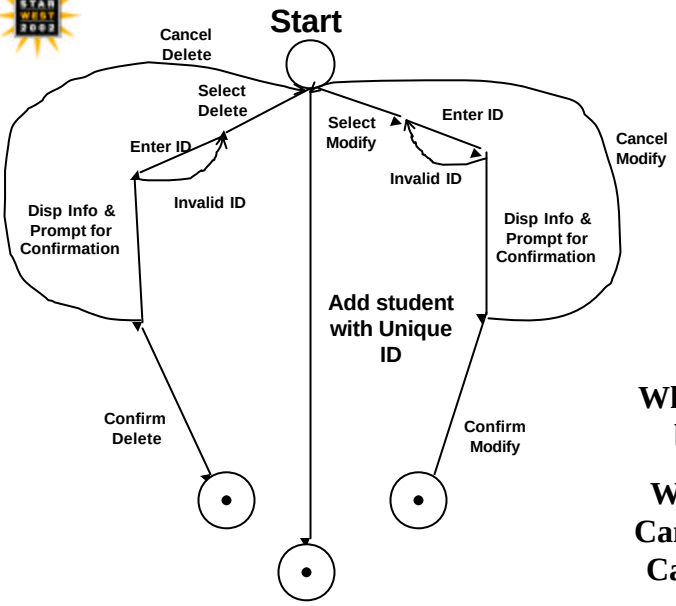
11



Raise an issue:

How to Exit from endless loop?

12

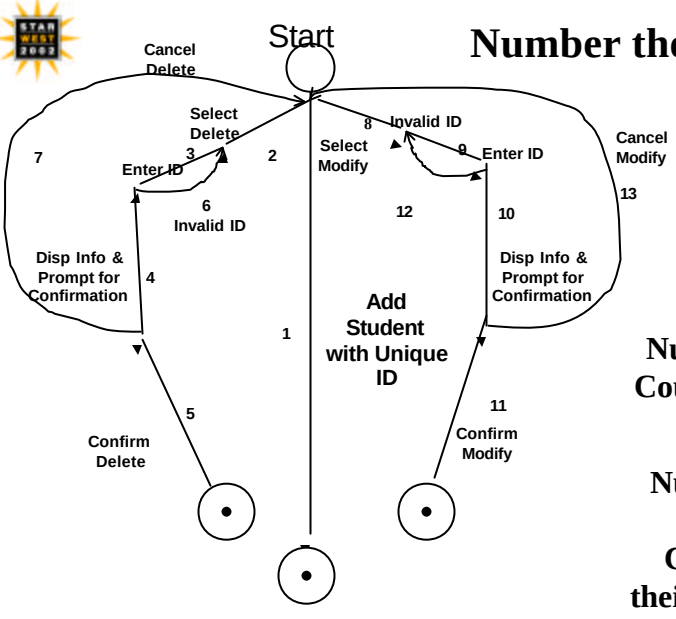


What else could be wrong?

Where should Cancel Delete or Cancel Modify return to?

Raise Issue?

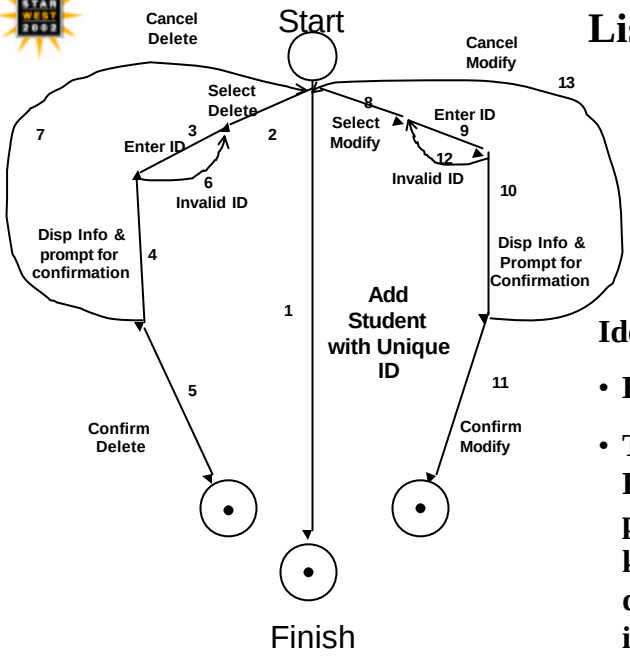
13



Number Basic Course Starting from 1.

Number each Alternate Course and their Exceptions continuously.

14



List All Paths

Identify the Paths:

- **Begin from Start**
- **Try to go to Finish in all possible ways, keeping the direction of flow in mind.**

15

Determine all possible paths

Begin from start and determine all possible ways to reach end keeping the direction of flow in mind.

List all possible paths and for each path

- Designate Path ID
- Designate Path Name
- Designate Path Description
  - -Describe what is taking place in the path
  - -Use Branch designations to come up with the description
  - -This descriptions is a summary of what needs to be tested

16

Partial Table of Paths

#	Criticality	Frequency	Path ID	Path Name	Path Description
1			P1	1	Add a Student
2			P2	2,3,4,5	Delete a student with valid ID
3			P3	2,3,6,3,4,5	Attempt to delete a student with invalid ID. Correct the Invalid ID, and then Delete.
4			P4	2,3,4,7,2,3,4,5	Select Delete with Valid ID, Cancel, and then delete with Valid ID.
5			P5	2,3,4,7,2,3,6,3,4,5	Select Delete with Valid ID, Cancel, and then Attempt to delete a student with invalid ID. Correct the Invalid ID and then Delete.
6			P6	2,3,4,7,2,3,6,3,4,7,1	Select Delete with Valid ID, Cancel, and then attempt to delete a student with invalid ID. Correct the Invalid ID, Cancel Delete again, and then Add a student.
7			P7	2,3,6,3,6,3,6,3,4,5	Attempt to delete a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID, and then Delete.
			etc….		
8			P8	8,9,10,11	Modify a student with valid ID.
9			P9	8,9,12,9,10,11	Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
10			P10	8,9,10,13,8,9,10,11	Select Modify with Valid ID, Cancel, and then Modify with Valid ID.
11			P11	8,9,10,13,9,12,9,10,11	Select Modify with Valid ID, Cancel, and then Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
12			P12	8,9,10,13,8,9,12,9,10,13,1	Select Modify with Valid ID, Cancel and then Attempt to Modify a student with invalid ID. Correct the Invalid ID, Cancel Modify again and then Add a student.
13			P13	8,9,12,9,12,9,12,9,10,11	Attempt to Modify a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID, and then Modify.

17

Analyze and rank paths

Three stage Process

Add Attribute Values-> Consult Business Users

-Criticality

-Frequency

Compute Path Factor

path factor = criticality + frequency (Can Use weighted attributes also)

Sort Paths as per path factor in descending order(Optional)

18

Adding Attributes

#	Criticality	Frequency	Path ID	Path Name	Path Description
1	10	10	P1	1	Add a Student.
2	9	9	P2	2,3,4,5	Delete a student with valid ID.
3	7	5	P3	2,3,6,3,4,5	Attempt to delete a student with invalid ID. Correct the Invalid ID and then Delete.
4	6	6	P4	2,3,4,7,2,3,4,5	Select Delete with Valid ID, Cancel, and then delete with Valid ID.
5	8	2	P5	2,3,4,7,2,3,6,3,4,5 (This includes P3 & P4)	Select Delete with Valid ID, Cancel, and then Attempt to delete a student with invalid ID. Correct the Invalid ID and then Delete.
6	7	2	P6	2,3,4,7,2,3,6,3,4,7,1 (This includes P3, P4 & P1)	Select Delete with Valid ID, Cancel ,and then Attempt to delete a student with invalid ID. Correct the Invalid ID, Cancel Delete again and then Add a student.
7	9	4	P7	2,3,6,3,6,3,6,3,4,5 (This includes P3)	Attempt to delete a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID, and then Delete.
			etc….		
8	9	9	P8	8,9,10,11	Modify a student with valid ID.
9	7	5	P9	8,9,12,9,10,11	Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
10	6	6	P10	8,9,10,13,8,9,10,11	Select Modify with Valid ID, Cancel, and then Modify with Valid ID.
11	8	2	P11	8,9,10,13,9,12,9,10,11 (This includes P9 & P10)	Select Modify with Valid ID, Cancel, and then Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
12	7	2	P12	8,9,10,13,8,9,12,9,10,13,1 (This includes P9, P10 & P1)	Select Modify with Valid ID, Cancel, and then Attempt to Modify a student with invalid ID. Correct the Invalid ID, Cancel Modify again, and then Add a student.
13	9	4	P13	8,9,12,9,12,9,12,9,10,11 (This includes P9)	Attempt to Modify a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID and then Modify.

19

Selection of Paths for testing

Guidelines for minimizing the risk and balancing the resources

- Always test the Basic Flow
  - It is critical.
  - Failure of the Basic Flow may be a show stopper.
  - If the Basic Flow fails test, there may be no need to test other paths for that Use Case in this build.
  - Good tests for Sanity Checking and also for inclusion in Acceptance Testing.

20

Selection of Paths (cont..)

- Although not Basic Flows, some other Paths are too important to be missed. Test them next.
- Determine a set of paths which would
  - Combine other paths, in which case the Path Factor is cumulative.
  - Ensure that all the Branches are tested at least once.
- This set should test all the flows and vital combinations.
  - **Example: Path Selection**

21

How to Select Paths

#	Criticality	Frequency	Path Factor	Path ID	Path Name	Path Description
1	10	10	20	**P1**	1	Add a Student.
2	9	9	18	P2	2,3,4,5	Delete a student with valid ID.
3	7	5	12	P3	2,3,6,3,4,5	Attempt to delete a student with invalid ID. Correct the Invalid ID and then Delete.
4	6	6	12	P4	2,3,4,7,2,3,4,5	Select Delete with Valid ID, Cancel, and then delete with Valid ID.
5	8	2	10	**P5**	2,3,4,7,2,3,6,3,4,5 (This includes P3 & P4)	Select Delete with Valid ID, Cancel, and then Attempt to delete a student with invalid ID. Correct the Invalid ID and then Delete.
6	7	2	9	***P6***	2,3,4,7,2,3,6,3,4,7,1 (This includes P3, P4 & P1)	Select Delete with Valid ID, Cancel ,and then Attempt to delete a student with invalid ID. Correct the Invalid ID, Cancel Delete again and then Add a student.
7	9	4	13	**P7**	2,3,6,3,6,3,6,3,4,5 (This includes P3)	Attempt to delete a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID, and then Delete.
				etc….		
8	9	9	18	**P8**	8,9,10,11	Modify a student with valid ID.
9	7	5	12	P9	8,9,12,9,10,11	Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
10	6	6	12	P10	8,9,10,13,8,9,10,11	Select Modify with Valid ID, Cancel, and then Modify with Valid ID.
11	8	2	10	**P11**	8,9,10,13,9,12,9,10,11 (This includes P9 & P10)	Select Modify with Valid ID, Cancel, and then Attempt to Modify a student with invalid ID. Correct the Invalid ID and then Modify.
12	7	2	9	***P12***	8,9,10,13,8,9,12,9,10,13,1 (This includes P9, P10 & P1)	Select Modify with Valid ID, Cancel, and then Attempt to Modify a student with invalid ID. Correct the Invalid ID, Cancel Modify again, and then Add a student.
13	9	4	13	**P13**	8,9,12,9,12,9,12,9,10,11 (This includes P9)	Attempt to Modify a student with invalid ID. Repeat with 3 invalid IDs, Correct the Invalid ID and then Modify.

22

How to Select Paths

Paths selected :

P1 Basic Flow
P5 or P6 : Choose one of them. I chose P6.
P7 Just to ensure testing of multiple invalid loops
P8
P11 or P12. Choose one of them. I chose P12.
P13

What did we achieve?

- **By testing 6 paths out of 13 identified (may be many more) we have almost tested everything worth testing in this use case.**
- **Have we tested all branches?**

23

Path Selection(Cont..)

Note:
In large number of cases, you will probably be testing all Use case paths. The prioritization process helps you in selecting critical or more important paths for testing.

24

Data Table Example 1

Data Table G(For Path P7)

Attribute Name	#1	#2	#3
ID	*VB4345680* *V234569012* *C45632P235* **VB373890**	*VC245678* *VB789134* *VC340909* **VB789032**	*VA121000* *BV453219* *AV453219* **VA453219**
Remarks	1Valid & 3Invalid IDs. Correct Invalid IDs by Valid ID.	1Valid & 3Invalid IDs. Correct Invalid IDs by Valid ID.	1Valid & 3Invalid IDs. Correct Invalid IDs by Valid ID.
Expected Results	Student with particulars as per Table A1 will be deleted	Student with particulars as per Table A2 will be deleted	Student with particulars as per Table A3 will be deleted

Note: Notice how we show Valid & Invalid data

25

Data Table Example 2

Data Table A (For Path P1)

Attribute Name	#1	#2	#3
Name	Victor Thomson	John Smith	Mary Hopkins
DOB	01/11/75	*02/12/2000*	10/11/1979
SS #	555 44 7777	222 11 7789	543 24 8907
Status	Citizen	Resident Alien	Non Citizen
Graduation Date	02/20/2000	03/15/2001	09/15/2002
Expected Result	System should return a valid ID.	System should return a valid ID.	System should return a valid ID.

26