SOFTWARE ENGINEERING LAB6

Exercise: Draw an activity diagram to graphically represent the following workflow:

Let us consider the development activities of SE Virtual Labs. The process begins by checking out the code from Subversion repository. Necessary modifications are then made to the checked out code (local copy). Once the developer is done with his changes, the application has to be tested to verify whether the new functionality are working fine. This test has to be performed with two of the more popular web browsers: Firefox and Internet Explorer, to support cross-browser accessibility. If testing fails in at least one of the two browser, developer goes back to his code, and fixes it. Only when all the browsers pass the test, a patch is generated from the local copy, and applied to the production code. The local copy is then committed resulting in update of the SVN repository. Note that, if the local copy is committed before generating a patch file, then local changes would get registered, and one won't be further able to generate the patch file.

Note: For further clarification, at any point of time there exists three versions of the source code: Production copy, local copy, and copy in SVN repository.

Think over the following questions:

1. **How would you represent testing of the application with multiple browsers?**

   There are two ways to represent testing of the application with multiple browsers in an activity diagram:

   - Sequential: This approach involves testing the application in one browser at a time. The activity diagram would show two separate test activities, one for Firefox and one for Internet Explorer. The end of the Firefox test activity would be connected to the start of the Internet Explorer test activity.
   - Concurrent: This approach involves testing the application in both browsers simultaneously. The activity diagram would show two separate test activities, one for Firefox and one for Internet Explorer. The start of both test activities would be connected to a fork node, and the end of both test activities would be connected to a join node.

   Which approach is better depends on the specific workflow. If the testing for each browser is independent, then the concurrent approach may be more efficient. However, if the testing for each browser depends on the results of the other browser, then the sequential approach may be necessary.

2. **Can the patch file generation and the Subversion repository update be done concurrently?**

   No, the patch file generation and update of the Subversion repository cannot be done concurrently. This is because the patch file is generated from the local copy of the code. In order to update the Subversion repository, the local copy of the code must be committed first. Therefore, the patch file generation must be completed before the Subversion repository can be updated.

3. **Can patching the production code and updating the Subversion repository be done in parallel?**

   No, patching the production code and updating the Subversion repository cannot be done in parallel. This is because the production code is a shared resource. It is not possible to safely modify the production code and update the Subversion repository at the same time.

Learning Objectives:

1. Identify the basic units of work and visualize the workflow.

The basic units of work in the SE Virtual Labs development workflow are:

1. Checking out the code from the Subversion repository.
2. Making necessary modifications to the local copy of the code.
3. Testing the application in Firefox and Internet Explorer.
4. Generating a patch from the local copy of the code.
5. Applying the patch to the production code.
6. Committing the local copy of the code to the Subversion repository.

The workflow can be visualized as a sequential series of activities, with two parallel activities for testing the application in Firefox and Internet Explorer:

Start --> Check out code from SVN repository --> Make necessary modifications to local copy --> Test application in Firefox and Internet Explorer (parallel) --> Generate patch from local copy --> Apply patch to production code --> Commit local copy to SVN repository --> End

2. Identify activities that could be done in parallel.

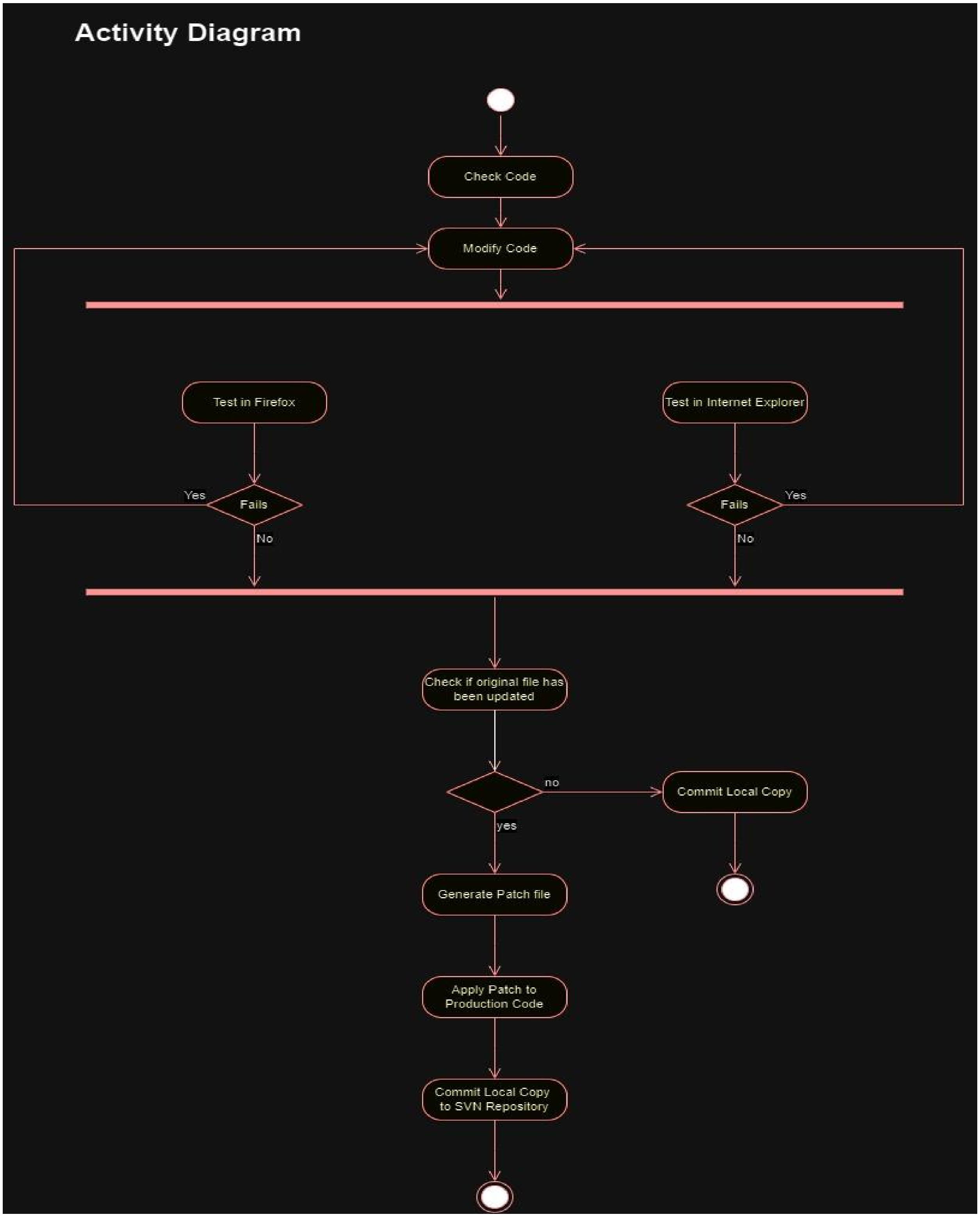
Testing the application in Firefox and Internet Explorer can be done in parallel.

3. Identify stages from where progress could be made only after a list of criteria is satisfied.

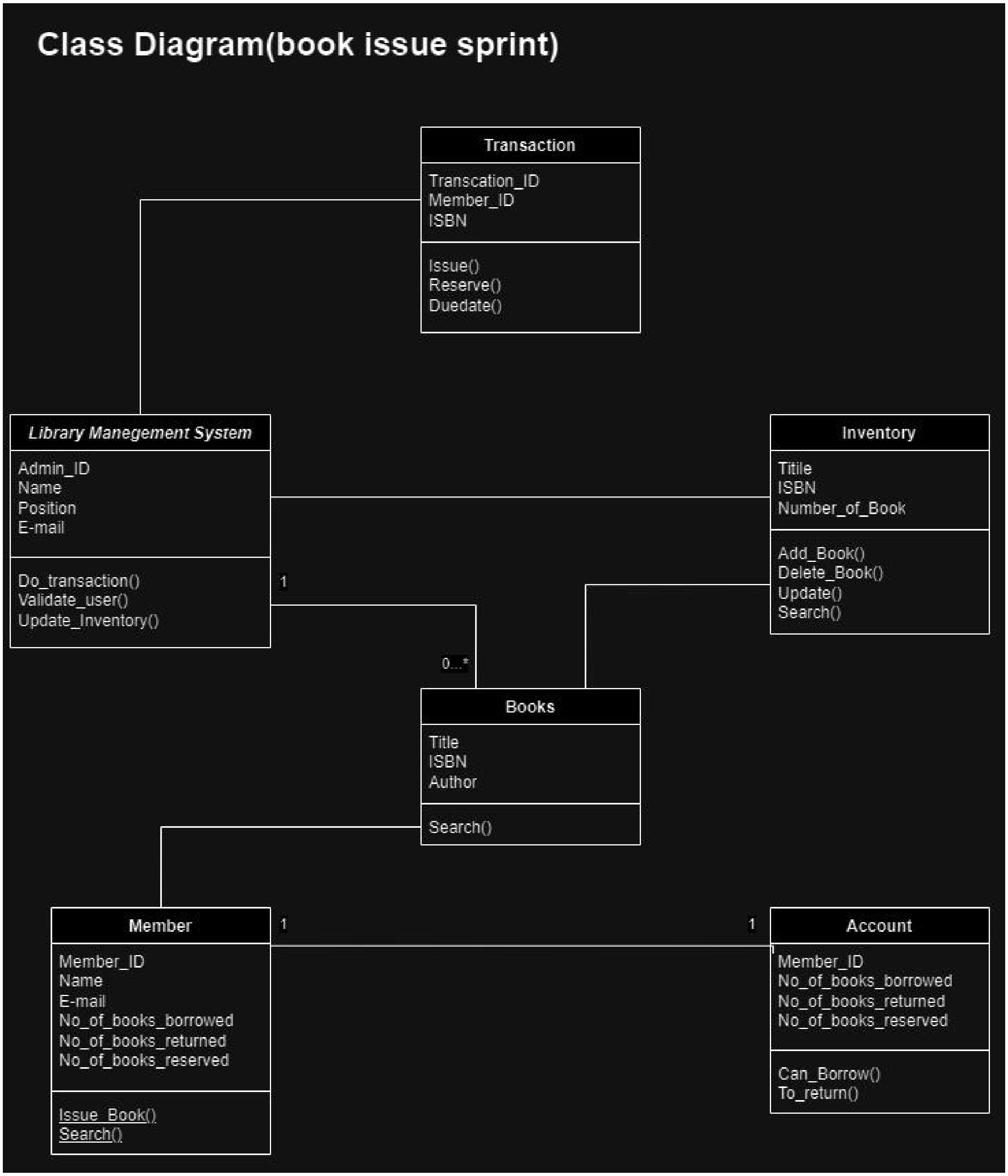
Progress can only be made to the next stage after the criteria for the current stage have been satisfied. The following stages are dependent on the previous stages:

- Generating a patch from the local copy of the code can only be done after the testing has passed in both browsers.
- Applying the patch to the production code can only be done after the patch file has been generated.
- Committing the local copy of the code to the Subversion repository can only be done after the patch has been applied to the production code.

Activity Diagram:


Activity Diagram
Check Code
Modify Code
Test in Firefox
Test in Internet Explorer
Yes
Fails
No
Yes
Fails
No
Check if original file has been updated
no
Commit Local Copy
yes
Generate Patch file
Apply Patch to Production Code
Commit Local Copy to SVN Repository

Class diagram:


Class Diagram(book issue sprint)
Transaction
Transcation_ID
Member_ID
ISBN
Issue()
Reserve()
Duedate()
Library Manegement System
Admin_ID
Name
Position
E-mail
Do_transaction()
Validate_user()
Update_Inventory()
Inventory
Titile
ISBN
Number_of_Book
Add_Book()
Delete_Book()
Update()
Search()
1
0...*
Books
Title
ISBN
Author
Search()
Member
Member_ID
Name
E-mail
No_of_books_borrowed
No_of_books_returned
No_of_books_reserved
Issue_Book()
Search()
1
1
Account
Member_ID
No_of_books_borrowed
No_of_books_returned
No_of_books_reserved
Can_Borrow()
To_return()