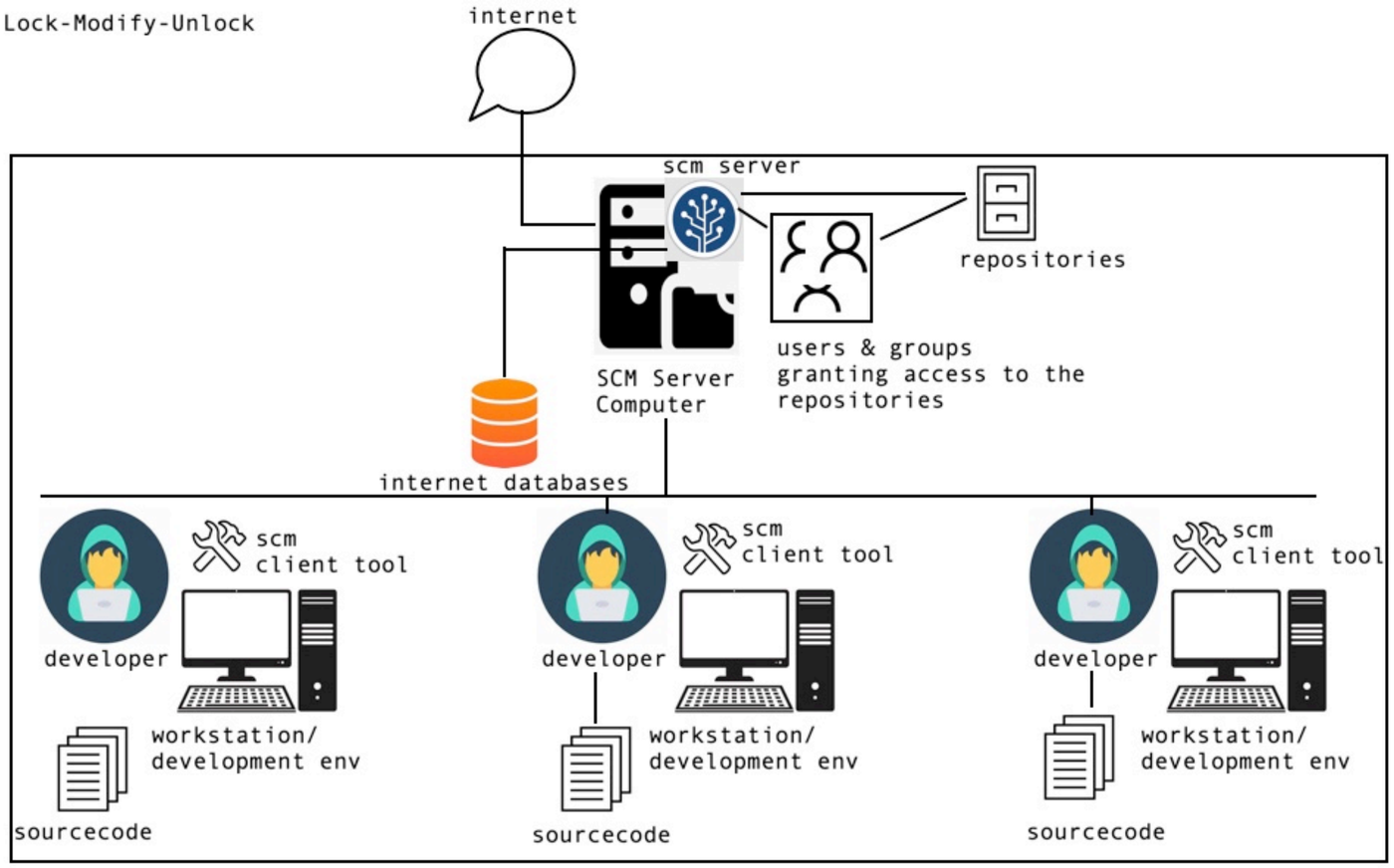
Lock-Modify-Unlock



1. GET
2. CHECKOUT
3. UPDATE
4. COMMIT

dis-advantages:
1. This lock-modify-unlock strategy doesnt support concurrent application development, because at anypoint of time only one developer within the team can acquire a lock on a file and work on it blocking others. Most of the time the developer may not finish the development of the code within a day, aspart of his work, he might begin acquiring the locks on bunch of files over the time and might take couple of days to finish, until then the other developers in the team are blocked and will not be able to work on the sourcecode parallelly

2. deadlock
Each of the developers in team arrive to an situation where a developer while modifying the sourcecode of a file, he might be waiting for another file that is already locked by other developer. even the other developer also waits for the file being modified by developer1, thus causing deadlock.

```
developer#1                developer#2
 - A.java (checkout)       - B.java (checkout)
 - B.java (waiting)        - A.java (waiting)
     ------------ deadlock --------------
```

3. there is always a chance where one of the developer in the team might lock bunch of files aspart of feature he is working on and might go on vacation/un-planned timeoff leaving the entire team blocked for the sourcecode of the project.

How to do the collaborative application development using the source code management repository server?
The initial project will be created by an architect/lead or one of the developer in the team and places this project in the scm server by creating an repository.
For each user in the team, the scm server administrator creates an user/password granting access to the repository allowing them to perform operations like read, write, delete etc

1. Each developer inorder to begin developing the application, he needs to create an working copy of the code locally from the scm server. To create an working copy of the code, he needs to use the client tool and perform “GET” operation on the scm server.
The entire repository on the SCM Server will be downloaded and placed on the developers machine.
At this moment all the files/folders on the developers machine are in read-only state not permitting them to modify.

2. Now the developer to begin working on the sourcecode, he has to checkout the files/folders of the project
  1. if he checkout an folder, then all the files/folders inside it are marked as checked-out
  2. or he can individually goes to an file and can checkout
when the developer marked an file/folder as checked-out, using scm tool the below things happens
  1. The scm client tool takes the updated copy of the file/folder from scm server
  2. makes the file non read-only
  3. marks the file/folder as locked on the scm server, not allowing any other users of the repository to checkout those files/folders

in case of a new file/folder:
the developer can create a new file/folder in his local system within the repository/project directories and needs to add them to the scm server marking them as to be newly created on the server

3. The developer after checkout the files, he can continue to modify them locally, upon completing the code changes, he can commit those changes onto the server
commit = process of placing the modified changes back onto the scm server.
only the files/folders that are checked-out by the developer only will be permitted to commit.
since the developer is the only one who has modified those files as he checked-out, we dont need worry about overwriting the changes of others.

upon committing the files, the local changes will be placed on the server and unlock those files/folders allowing other developers to checkout.

4. At anypoint of time, if the developer wishes to get the changes of made by other developers on the repository he can update the local copy