IT415 Software Testing And Quality Analysis

Prof. Saurabh Tiwari

Subject : Testing Concurrent Software

Group Members :

1. Mohit Savaliya - 201401137
2. Vinaykumar Patel - 201401147
3. Dhaval Panjwani - 201401162

Testing Concurrent Software

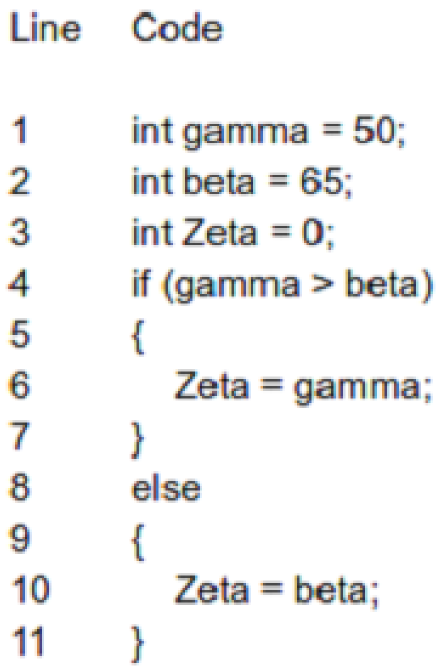
Index :

Abstract

Software Testing for any concurrent software or concurrent programs is very industrially challenging and time consuming activity. In the consideration of current high demand of testing concurrent applications, lots of researches get conducted with time in this field. Our report on the technical end, showcases the current scenario and trends going on, which are being applied around the globe in the vast area of testing concurrent applications. Majorly, that involves the techniques which have been adapted and improved from the core testing tools and frameworks used for sequential code. These sequential code techniques have been reframed in such a manner that they can test multi-threads and make the code industry consistent for establishing a healthy user experience.

What is Sequential Software?

If a programmer wants to write a code, which step by step does some specific set of work then we can call that program to be sequential. So basically if we number the line statements of code in the 1 to N fashion and that code gets executed in the same order of appearance of the increasing number assigned to the statement, then it is said to be a sequential code. And overall, that software is called a sequential software. For example, see the below figure, the code when compiled will execute line by line from 1 to 11 in orderly fashion no matter how many times do we execute.

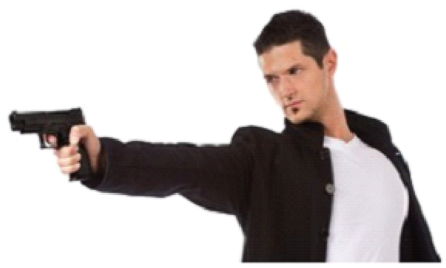


```
Line    Code

1       int gamma = 50;
2       int beta = 65;
3       int Zeta = 0;
4       if (gamma > beta)
5       {
6           Zeta = gamma;
7       }
8       else
9       {
10          Zeta = beta;
11      }
```

What is concurrent software?

As we saw above, sequential code is executing a single buffer of functions. Whereas concurrent code is like having more than buffer of functions which execute simultaneously. Although, these buffers could communicate within themselves while running. So we could take a real time example, imagine James Bond holding one pistol and shooting enemies. This would be sequential execution of pistol firing one bullet at a time. What if there are so many enemies to tackle? At such a time imagine James Bond holding two pistols and shooting enemies simultaneously from both the weapons. That is called execution of concurrent software. See the figure below,


Sequential Software (one pistol)

Concurrent Software (two or more pistols)

Concurrent software Testing

Creating a test plan :

Before testing any concurrent System or Software we will have to make Test Plan for that.

Scope of the Test :

First of all we will have to check the scope of the test and test strategy. That involved about what to be tested and what not to be tested while testing the concurrent software.

Test Strategy :

After our scope is ready for test then we come up with test strategy. There will be different type of testing like Unit Testing, System Testing, Performance Testing and so on. If testing is browser based testing then it would be GUI Testing, Usability Testing and so on.

Depending on the product which is going to be tested, we need to come up with what type of tests that are useful and what type of tests are not useful.

Test Plan would be depending on which type of test strategy will be used, it might be Unit Testing, it might be overall system test plan or it might be a workload to work on performance testing or testing the load of the system. This is how we create the Test Plan.

Resource Requirement :

Once we have done with testing strategy then we can come up with something called The Resource Requirement which is most important requirement.

We have to determine how many QA engineers we will need.

What kind of hardware we need. Example : If we are testing an application which is totally web based app. Then in that case we need Web Server to run on a Linux Hardware or Database Server on the other hardware.

We also come up with software requirement. Like we might need Microsoft word or we need licence for microsoft excel. We might also need some kind of Monitoring Tool so that we can monitor the system resource uses while our tests are running. Thus we can come up with what kind of software we need to execute the testing effort.

Schedule :

After that, Schedule come in the picture which tells us when we are going to finish particular test or when we are going to end up with testing any software.

Building a Quality assurance Plan :

Quality Assurances main agenda is not to Find all of the bugs present in the software. Because its impossible to find all bugs. Quality assurance is for increasing the confidence that the software is working correctly and it has sufficient accessibility. Quality Assurance approaches include education, careful design and training.

Manual Code Review :

We can do Manual Code checking for finding bugs. Review of Experts can be the best way for testing and finding the bugs of concurrency software which is little bit expensive perhaps.

Advantages :

- Manual code review can spot bugs which occur rarely while testing any concurrent software using any framework.
- This technique can find bugs that will not happen on particular specified machine or hardware.
- This technique also improve some of the code content also find errors easily.

This technique is useful for small and separate from the hardware and concurrent components. And this technique is really very hard even for experts. This technique is also very expensive if we use this constantly.

Problems while Testing Concurrent Programs

Unit testing has several advantages as well as some disadvantages also when we are comparing them to traditional testing of acceptance, which will usually happens when the compilation of code will happen.

Different and separate parts such as methods/classes, unit tests is checking all of the segments of codes and also whole code that are little bit smaller than the whole application or software. For a mentioned bunch of classes, there is not any type of inner connection of one method onto another method. So Unit testing will make the detection of bugs and errors easy. Unit testing is generally showing effectiveness in combination with code segments covering tools which is ensuring that the unit testing usually execute every every single line of the written code and code segments too.

Unit tests are programs which can be executed while testing can be conducted so easily and test bugs, errors also can be reported fastly.

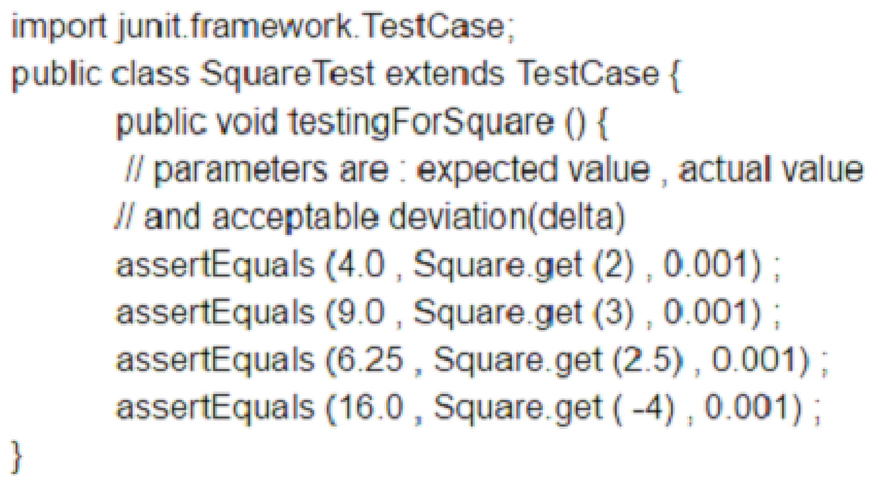
Unit tests will be there and remain unchanged even when the application code or application code segments are changed, the written programme/code can be refactored easily and there will not be any trouble.

What we have to do? We will firstly document a unit test that is the specific bug then we will repeatedly check whether its happening as bug or not - then we will declare it as bug and will be noted down. Only then the bug is fixed, which will be tested further and will be tested with the same test case which are going to be failed. The new unit test will be added to repository of that written code along with the bug fix list and that's why that will prevents the problem from occurring again and again. Unit test also provides us the use of examples of the code segments that they usually test and that can be considered as the main documentation part. Understanding of the unit tests can generally simplify the understanding of our mindset how any program should be used. And we can also find which kind of problems can be happened.

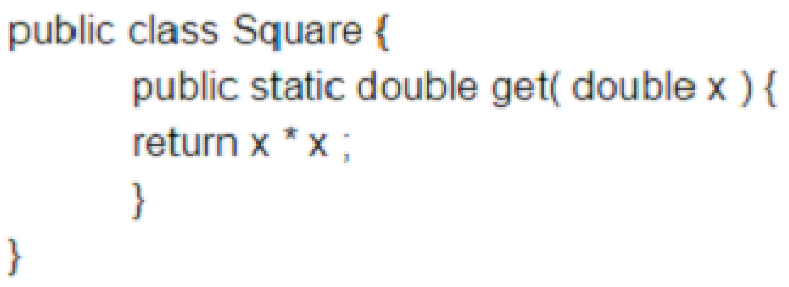
Open-source unit testing frameworks are mostly available for all of the existing languages. Java, JUnit and TestNG are the languages which are being used generally for that.

Here, given the 1st programme below shows us a simple unit test which is using JUnit tests that tests the method named (Square.get) which is provided here in 2nd class Square which is also written below. Unit test class is generally tested by ant tester by asserting according to the output which will be given by that particular method and we will match it with the actual output we were expecting till now. We have tested here all of the possibility like 1) positive numbers, 2) positive decimal number and 3) non-positive number as the input to the given function.

We have given first argument to the given below AssertEquals function is the expected output of that particular function, where the second argument which is also given to the same function is the actual result which is the results of that same method.The third argument is the deviation of real result (actual result) from the expected result list. So if output which we get here is if within this deviation result then that will not be considered as a bug or any fault in the code/programme. And that would be accepted by the written program itself.For the given class named Square, the unit test will pass successfully because all assertions are going to met successfully. In listing 3rd programme, the method square is defined for very small input domain and that domain is restricted there for input and output. The domain is only integers, and that will not be working for any other domain and that will be end up with errors. For an example regarding 3rd programme below, if we put 0.5 as an input and we are expecting as output 0.25 and will not get that result.

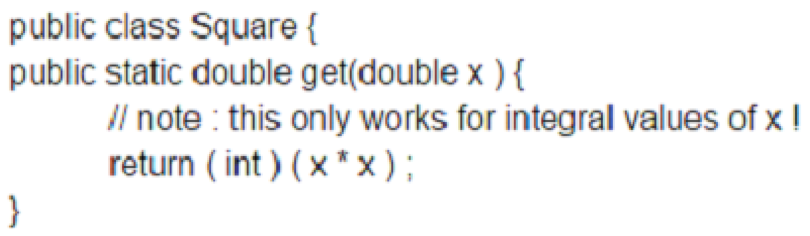


```
import junit.framework.TestCase;
public class SquareTest extends TestCase {
        public void testingForSquare () {
         // parameters are : expected value , actual value
        // and acceptable deviation(delta)
        assertEquals (4.0 , Square.get (2) , 0.001) ;
        assertEquals (9.0 , Square.get (3) , 0.001) ;
        assertEquals (6.25 , Square.get (2.5) , 0.001) ;
        assertEquals (16.0 , Square.get ( -4) , 0.001) ;
}
```



```
public class Square {
        public static double get( double x ) {
        return x * x ;
        }
}
```

Here given below, an incorrect or wrong implementation of square function is used.



```
public class Square {
public static double get(double x ) {
        // note : this only works for integral values of x !
        return ( int ) ( x * x ) ;
}
```

In the above particular example in that class, the function works correctly for integral input values only, and the assertion expecting a result of .25 for the input value .5 will fail.

This is only a simplest unit test example which is easy and it might be unnecessary. So we can say finally that methods for unit testing are not very valuable tool for most complicated and so called more composed systems and methods.

There are some of the restrictions on the developer of the older version of the Junit which are mentioned below :

Example :
1: All methods which are going to be run have to be public void otherwise it wont work.
2: All of the names of the methods Mentioned in the code have to begin with test. Example : @Test.
3: All the methods will be declared in one class and that class must has to be subclass of junit framework test.

Inadequacies of Existing Frameworks :

As shown above, there are many frameworks which are generally used for testing any type of concurrent system or concurrent software. All these given frameworks are mostly designed and aimed particularly for testing code which has a single thread control. There are some extensions. That extensions are using multithreading to a certain level. There is also availability for parallel Junit but that Junit will only work on several running unit tests concurrently. That will not focus on running any concurrent unit test. In order to that, running several tests or test cases in parallel may result in some complicated ways that generally occurs while testing any concurrent system/software. But even with these type of extensions, the existing frameworks are not that much efficient. They can not test concurrent software or system efficiently or perfectly.

Generally unit testing rely on the input and output which are known to us in advance time and then the whole execution of that programme is determine from that advance information. In concurrent systems/software scheduling is very important. Scheduling of threads which will give introduction of an element and predictions of output in advance. The process involved in scheduling must be deterministic. So that unit testing can be valuable and can be useful for us.

The race detection algorithm is used to detect whether the race conditions exists or not. The race detection algorithm runs in parallel and is constantly ensures that programs are not having Race Conditions.

We want to produce meaningful and expected results . For that we can use Schedule Based Execution. For that the framework firstly needs to run and test the programs in a single schedule easily. This seems like easy but it is not an easy task with the existing frameworks. Thats why there will happen exceptions and failed assertions in the execution. Which will be occurred in threads other than the main thread, That will be ignored. The existing frameworks do not give guarantee about child termination before the test is declared a success.

Enforcement of Threading Disciplines :

Developers of Concurrent Software or Concurrent Systems need to obey a certain discipline of locking data structures. They have to also bear in mind while calling methods only from certain threads. And this is necessary because of preventing possible data loss from the content. At this type of critical situations, Developers have to be serious while using locks. It can be also happened in the worst case that some threads have locks and wait for other threads to free their locks. And that are preventing all threads from making progress further. This type of situation is happens and that are called Deadlock.

Javas GUI frameworks, AWT and Swing. All this frameworks require developers to follow complex discipline in order to develop and test the concurrent software. This become necessity because concurrent access to the programme by a general thread may cause data loss and it can be pus us in any unwanted situations. Example : Deadlocked situation. On the other hand, some methods may not be called from within the event thread, because that can create deadlock.

Common Problems in Concurrent Software :

- There can be problems while planning the Test and scheduling any event.
- We can also have to face problems related to test tools or problems related to environment.
- We will also have to face problems which are related to test organization and professionalism problems.
- Problems while testing any process which is involved in concurrent software.
- There will be also general problems related to Management.
- Problems while communication between tests.
- Requirements related testing problems also happens while testing any concurrent software.

Comparing hardenability of Sequential Software testing vs Concurrent Software testing

Sequential code has a working routine to be followed strictly where first a single piece of code runs. Later when that completes, another code instruction is loaded and then run. Hence while doing sequential code testing, we need to look on a single code running at a time. Then after that code is completed, another code has to be looked upon. Testing sequential code is naturally deterministic as we dont have to care about number of different workflows of the same code. A similar type of input would always relatively lead onto the same type of failure.

Test cases designed for testing sequential code

- Should try to achieve most of the part of the code through code coverage techniques like statement coverage, branch coverage, etc
- Might find the combinations and permutations of input and methods or actions which would probably land up in a failure.
- By exercising code with proper asserting constants and post-conditions.
- Should test performance of the system under all the different test scenarios
- Should test the safety and security measures of the system under all the different test scenarios.

Then how should we test concurrent software?

Not the same way, we need something different. Concurrent softwares dont have such low failure modes but very high. They have to deal with more failure states depending upon the situation which generally has more odds in the favour of failure. Concurrent software is no more deterministic like sequential software and non-decidability factor is high.

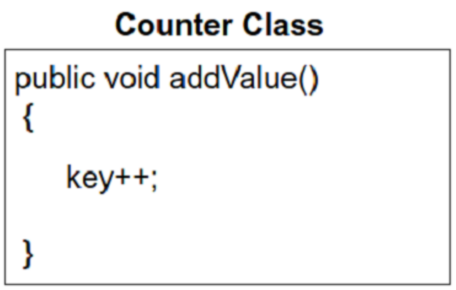
Test cases designed for testing concurrent software

- Executing same test cases multiple times would lead to different outputs every time. And would lead to different results.
- Then using those different results for a single particular test case to debug is hard.
- It is harder to recognize and hardest to reproduce the code.
- Requires large set of configuration files to test the system
- Generally represents and reports only 10
- Later other number of bugs are found very late or by the users/customers but we miss the deadline

- Bugs being found by concurrent software later turns out to be very expensive for the company.
- So the costing effort of the concurrent software by the debuggers at the system level is non-resistable.

Let us consider an example.

Given below is a function of addValue(). It will increases the value of static variable key of the class which works as a counter class.



Counter Class

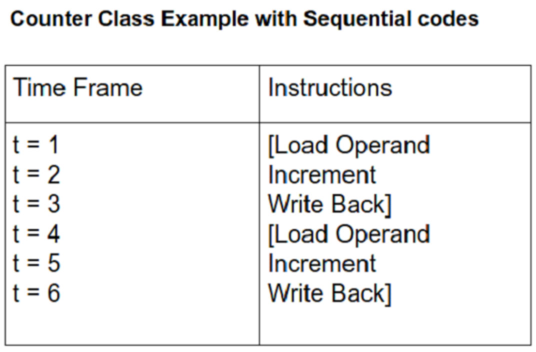
```
public void addValue()
 {

     key++;

 }
```

Even the operator ++ written above works like a single operator, but is a combination of three sub operations. Those operations are instructions namely, (i) Load the present value of the operand in the register (ii) Increment value in the register and (iii) Write back the value of register back to the operand

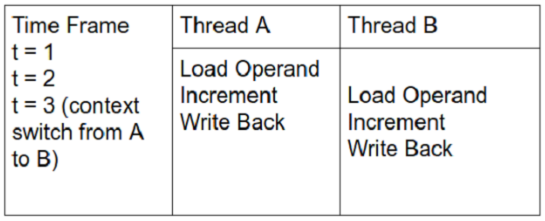
Lets assume initially the value of static variable key is 3. And we perform the below sequential code which calls the addValue() function two time one after the other. This will write value 4 back to the variable key. Similarly, when we will call the function addValue() second time, the value of key changes from 4 to 5 after writing the register value back to the operand. Hence the code works correctly. Code would also give same output for a test case even after using the test case multiple times.



Counter Class Example with Sequential codes

Time Frame	Instructions
t = 1	[Load Operand
t = 2	Increment
t = 3	Write Back]
t = 4	[Load Operand
t = 5	Increment
t = 6	Write Back]

Lets now take the case of same code in the scenario when it runs concurrently. As shown below, there are two threads - A and B. And initial value of key is 3. Initially, we dont know which will run first. So lets say Thread A runs first and loads the value 3 in A and then context switch occurs. Now Thread B loads the same value of key 3, increments it by 1 and writes back value 4 to key. Then again Thread A continues by incrementing same loaded value of 3, not updated value 4 and this results in failure of concurrency in coherence with addValue function.


Counter Class Example with two simultaneous threads

Time Frame	Thread A	Thread B

t = 1
t = 2
t = 3 (context switch from A to B)
	Load Operand
	Increment
	Write Back
		Load Operand
		Increment
		Write Back

In such a system, we cannot be certain which thread will run first or which program will run last. There is no certain order of sequence to be followed. We dont know the order of execution of list of softwares. They may run in any order. These conditions if update a value of the same variable in concurrent time, are called race conditions where each runnable resource races for completing its task altogether. A single test case would hence lead to different - different results based on order of execution of concurrent software. We need a large configuration to debug and reproduce this code. For a company, biggest foes are race conditions when it comes to developing concurrent applications.

Testing Concurrent Software through Unit Testing

It is difficult to develop appropriate multithreaded software as it has complex issues related to it, and many things that are so simple for sequential threaded software are hard for multithreaded software. For example, take a blocking queue code, which blocks when we try to dequeue it when it is empty or when we try to enqueue it when it full with elements. A thread is trying to consume an item from a queue when it is empty, whereas another thread is trying to insert a new item in the queue but is blocked by the initial thread. So writing a test case to test the proper blocking and proper unblocking at needed times and its implementation would be hard. At least it would be hard with testing using general testing frameworks which are used for sequential code. We would rather use a framework built for testing concurrent threaded software. MultithreadedTC framework, which allows us to construct unit test cases which are deterministic and repeatable. This framework isnt meant to test for synchronization errors. This framework will test the concurrent functionality of the code, i.e. will test whether a code gives the proper functionalities of concurrency which a multithreaded software should generally give (For example, a thread which is trying to acquire lock over a resource is blocked if another thread is using the resource and has the lock over the resource).

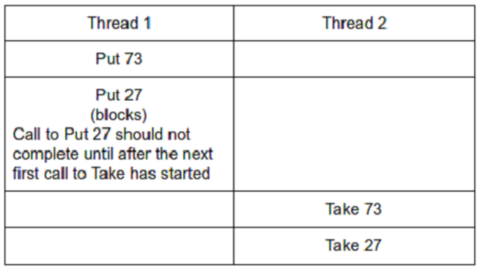
The software developer needs to refer and stick to various protocols for locking, stop race conditions, and allow coordination different associated threads. For a developer, he can manage the above complexities arranging the concurrent logic and business logic separately, by limiting thread concurrency to lower level abstractions like locks, semaphore, latches and bounded buffers. Then independently taking and testing these low level abstractions to validate the concurrent features of the software.

There are two strategies to unit test concurrent components. **1**: First strategy is writing large test cases. These cases are hoped that they contain many possible interleavings. Then running these test cases many number of times. This will hopefully induce much of faulty stuff in concurrent threaded software like faulty interleavings or un-appropriate and unpredicted behaviour of the software. Most of the frameworks to test concurrent components are based on this paradigm and provide features like increasing the probability of diversified interleavings to arrive for us to test adequately. But we cannot totally rely on this strategy, as it relies on probability and may miss interleavings which lead the software into failure mode. **2**: Another strategy is to take a specific set of interleavings or a single interleaving first. Then test those separately. There is presence of blocking in multithreaded software and timing issues. This makes it hard to exercise the specific set of interleavings which occur rarely or occur after many runs on the same test input. So here helps this MultithreadedTC framework. It allows the test case designer to select and exercise interleavings of specific threads in a software. This is fulfilled with the help of a clock, which helps the testers to manage and coordinate with several threads in spite of the presence of timing and blocking problems. Whenever all the threads will block, the clock will advance. Until the clock has reached a specific desired tick, the testers can delay operations within a thread. Moreover, using MultithreadedTC framework, livelock and deadlocks could be detected.

Example -

- Take a bounded buffer which is shared among two concurrent threads as shown in the figure below. A bounded buffer allows following tasks :
- A user can take element that has already been put into the buffer in prior to taking.
- A user can put a new element in the buffer.
- It has a fixed holding capacity for number of elements to hold at a time.
- Put and Get functions of bounded buffer makes the calling threads to block if the get function is called while buffer is empty or the put function is called when the buffer is full

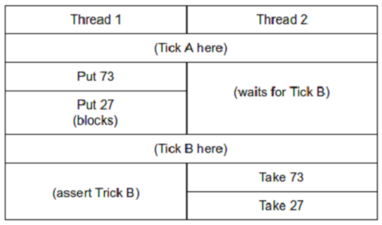
Following are four operations happening on a bounded buffer of capacity 1.



Thread 1	Thread 2
Put 73	
Put 27 (blocks) Call to Put 27 should not complete until after the next first call to Take has started	
	Take 73
	Take 27

Figure - Bounded buffer of capacity 1, hence the operations should occur in particular order

In this buffer, capacity is one. The call to put-27 will be blocked because buffer is already is filled with (73), because of put(73) operation of thread one. This block should be kept on while the space of buffer frees up by the next take(73) call. So how does the tester guarantee that thread 1 will block when put(27) until after the next take(73) call happens ? And how does the tester be sure that after the take(73) call happens, that block over thread 1 will be relieved ? One way to solve this is by putting a sleep to thread 2. And that will be given some time upto to be sure enough that thread 1 has been blocked by own. But this way makes the testing different units timing dependent and it gets difficult to understand, test and debug time dependent code for large test case inputs. Another way to deal with might be to put a latch on thread two that gets released at appropriate time. This wont probably work here at most times as the only other thread available to remove the latch and free the thread two itself is in block state. This would lead to a deadlock. No thread would move forward in this approach.



Thread 1	Thread 2
(Tick A here)	
Put 73	(waits for Tick B)
Put 27 (blocks)	
(Tick B here)	
(assert Trick B)	Take 73
	Take 27

Figure - MultithreadedTC framework makes sure the concurrency of program works properly and guarantees that the order of execution is correct.

Another approach, as used by the MultithreadedTC is shown in the figure above, is to have an external clock working for both the threads. This external clock is a separate thread. The clock will advance to the next desired tick if all of the test threads are in blocking state and only one thread is waiting for the tick. In our figurative example above, our thread 2 blocks initially, then thread 1 calls Put(73) and completes the call. When thread 1 calls Put(27), it blocks. At this stage, all the threads (threads 1 and 2) are in the block state and thread 2 is waiting for a tick B. Hence, the clock seeing all threads blocked, advances clock to tick B, which is our desired next tick. This releases the thread 2 from block state and it is now free to continue its execution. Thread 2 executes Take(73) and takes away element 73 from the bounded buffer. With this assertion, thread 1 is released. And thread 1 can execute normally without any problems or race conditions.

With this test, we are now confident about two things, (i) thread 1 blocked because the clock advanced and (ii) thread 1 was not released until after the execution of first assertion of thread 2 happened. So this clock ticking method always ensures in multithreaded components that a thread is blocked if a clock advances and the block is released from the thread in the next desired tick. This way, we balance the blocking and unblocking and coordinate between threads in a multithreaded environment by testing the test cases.

Restrict concurrent interactions by using immutability

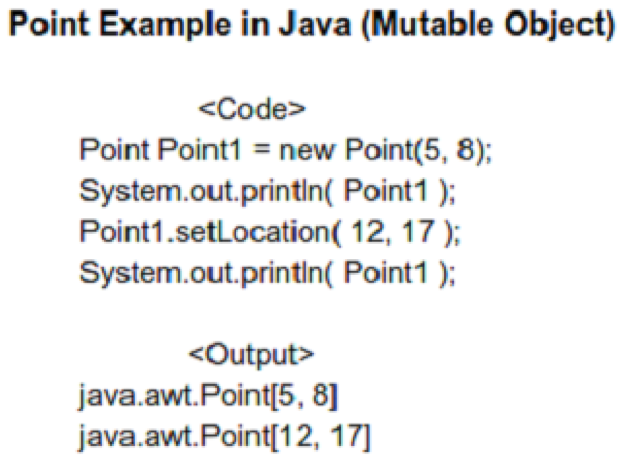
A programmer can write functional programs with public and static variables. It is not a good idea but it can be done. It could be done but it would get very hard to test that code. Without even going through code or understanding the entire piece of code Encapsulation helps analyze a part or portion of code.

In the same way, we could modularize all the concurrent transactional part of our program like workflow managers and resource pools. Rather than looking at the entire program it becomes easy to look at the behaviour of these modules of concurrent codes. We can put our full focus and efforts on testing these concurrent small portions. Mutable objects are objects which can be altered by value. Immutable are those which cannot be altered by value. Greatest example of immutable object is the string class in java which once initialized later cannot be changed.

Objects which are immutable by design, which have a mutable state but other threads cannot change the object value after it is defined are called effectively not mutable objects.A thread needs to create a new object, discard the old one referenced, and assign that reference to this newly created object to change the value of such a reference. The wise use of immutability can restrict the concurrent mechanisms happening and could save us from race conditions. This would ensure that potentially incorrect concurrency never occurs ever again by keeping immutability. These concurrent mechanism methods should be kept in few of the classes only. Rest of the classes should be kept sequential. After then those few numbers of concurrent classes can be unit-tested effectively.

Mutable Objects : If there is a reference to an object, then contents of that object can be altered by that reference.

Immutable Objects :The contents of that object can not be altered by that reference if there is a reference to an object . So see the example below of two classes, Point and String in java.

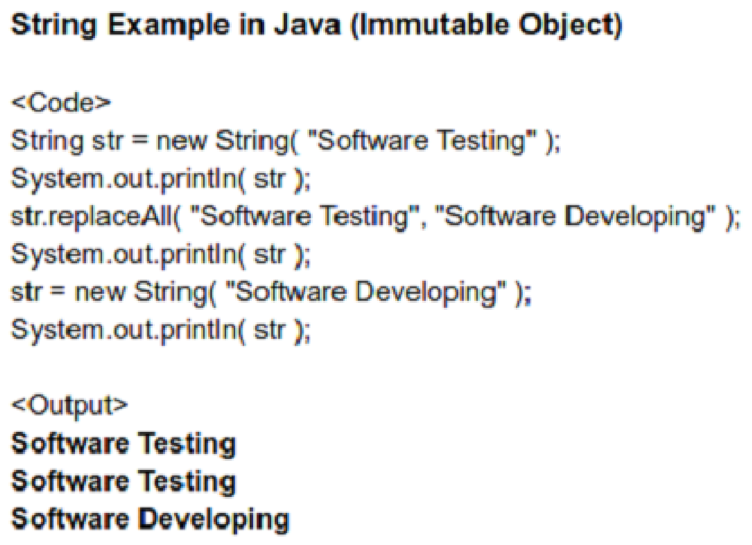


Point Example in Java (Mutable Object)

```
            <Code>
Point Point1 = new Point(5, 8);
System.out.println( Point1 );
Point1.setLocation( 12, 17 );
System.out.println( Point1 );

            <Output>
java.awt.Point[5, 8]
java.awt.Point[12, 17]
```

Point class in Java is mutable. Its methods are public, its variables are public. So any user can use them and change the value of the point being referenced.



String Example in Java (Immutable Object)

```
<Code>
String str = new String( "Software Testing" );
System.out.println( str );
str.replaceAll( "Software Testing", "Software Developing" );
System.out.println( str );
str = new String( "Software Developing" );
System.out.println( str );

<Output>
Software Testing
Software Testing
Software Developing
```

String class in Java is immutable. Its methods are private, its variables are private. So no one could use them to change the value of the string being referenced. This is the reason that replaceAll() method above didnt work to mutate the value of the string. To change the value of an immutable object, we need to create a new object of value we want, discard the old value being referenced and point the pointer pointing to the old object to the new object just created. Just as in the example above, we created a new String object and referenced its with str string pointer in the statement

str = new String(”Software Developing”);

to change the value of string. Usage of immutable objects in concurrent software programming, we can limit the concurrency actions resulting into race conditions or such kinds of synchronization errors. In such cases, every thread would always create a new object with the value it wants to assign and reference it to the pointer concurrently. This way, multiple threads could do their work. Primarily, now our requirement is to build Immutable classes, whose objects if ever get created would never be mutated.

Take the example of below class of a Person wich is shown in the next page while defining rules for immutable classes,

This class above has variables of two types, string and Date. String objects is immutable but date object is mutable. So we can see the date of birth is changed and added by two months. Such a class would not work in concurrent systems. We need to make the above class immutable to make it run in a concurrent system. In the following way we can make it immutable : Every time the date variable is called, create the new object for dob and return that new object. So that even if someone tries to change the value in dob, it would change the value in new object being returned and not the object variable which is in the class.

```
import java.util.Date;
        public final class Employee
        {
                private String Name;
                private String Profession;
                private Date dob;

                public Employee( String Name,
                 String Profession, Date dob)
                {
                        this.Name = Name;
                        this.Profession = Profession;
                        this.dob = dob;
                }
                 public Date getDOB()
                {
                        return this.dob;
                }
                ……..other methods etc
        }
// Driver code below
        Date birthDate = new Date();
        Employee e1 = new Employee( "Harikishan" , "Chaubey" , birthDate );
        System.out.println( e1.getDOB() );
        birthDate.setMonth( birthDate.getMonth() + 2 );
        System.out.println( e1.getDOB() );

<Output>
Tue Mar 14 11:54:10 GMT+11:00 2017
Sun May 14 11:54:10 GMT+11:00 2017
```

```
public Employee( String Name, String Profession, Date dob)
                {
                        this.Name = Name;
                        this.Profession = Profession;
                        this.dob = new Date( dob.getTime() );   // making a new object and
                                                                   then giving the reference
                                                                     To the new object
                }
   public Date getDOB()
                {
                        return new Date( this.dob );            // making a new object of
                                                         class variable dob and returning
                                                          reference to it
                }
```

How to make Immutable Classes?

Now we have a template for creating immutable objects.

- Make all fields private
- Don't provide mutators
- Ensure that methods can't be overridden by either making the class final (Strong Immutability) or making your methods final (Weak Immutability)
- If a field isn't primitive or immutable, make a deep clone on the way in and the way out.

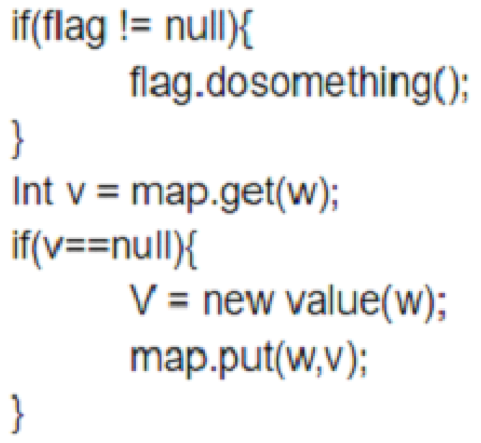
Concurrent Failure Modes

Most features in java language are designed to run repeatedly but many failures in concurrent programs are probabilistic rare events.

Synchronization error:

Suppose one programme has two threads and both them are using same variable. Now at least one of thread is accesses write operation and shared variable is not a volatile field then the accesses must be in synchronization. Otherwise our code is bad.

Now consider the following code:-



```
if(flag != null){
        flag.dosomething();
}
Int v = map.get(w);
if(v==null){
        V = new value(w);
        map.put(w,v);
}
```

Without synchronization errors, we can have some nasty or time dependent concurrency errors. And these are automatic failures.

Suppose in above code, another thread can set foo to null just before executing the programme. And it could also be possible that two thread call map same time and both see null so assign the new value to map then the dosomething() method couldnt be run and both the thread assign different values to V.

Testing for race cases:

We need to find testable properties which fail with high probability if something occurs wrong in our programme. We also not need to introduce additional synchronization.

Errors may be found by test programmes:

- Test programme may be related with timing
- Test programme synchronization may mask data races in our code.
- Delays in test programme may mask race conditions

Performance testing and load testing in concurrent programmes :

Performance testing = how fast our system is?

Load testing = how much volume can our system processes?

One can check the speed at which page renders in any browser, this is called performance testing. Load testing mainly focused on requests hits per second and concurrent users. So the performance testing is more efficient. Performance testing is more concerned with time.And it seems more broad than load testing.And it is also associated with white box testing.

Challenges while performing concurrent software testing

Concurrent users are defined as total number of people using that programme at same time. All online websites are hosted on web servers. These servers have certain number of CPUs which are concurrently in use.

Now we need to determine how many concurrent users are supported by our programme.

There are few approaches. The simplest way is to look at the number of CPUs and CPU cores that web or application server have available in use. Mostly one CPU can support nearly 250 concurrent users at a time. A quad core can handle 1K users at a time. This number can vary with available bandwidth, memory and some of the others factors.

So we have a big number of concurrent users at a same time. Because of that we have to encounter some challenges while performing concurrent software testing

- We first need to test any test case on multiple platforms.
- We required more intensive test cases.
- The flow of programme or any information is not reflected in the call stack.
- The functions use in softwares do not return to the caller immediately, but instead, it can be sent via callback functions or notifications or blocks. Which makes testing more difficult.
- Debugging of the concurrent programmes are difficult.
- The number of execution of any programmes paths can be extremely large as such the processes in a concurrent system can interact with each other when they are executing.
- Concurrent programs have less ratio of success than sequential ones.

Consequences of failure of concurrent software testing for any company

- Effect on the customers.

- Reputation of the company, cost to the company.
- Effect of product sales and prices.
- Effect on competition
- Re-work on the same problem.

Non-determinism in concurrent software/programs

Example of non-determinism in concurrent software/programs:-

Concurrent programs behaves differently than sequential programs. Suppose we are taking fixed inputs and execute it on a multiple concurrent programs and produce different results. One approach to fix such difficulty is non deterministic testing.

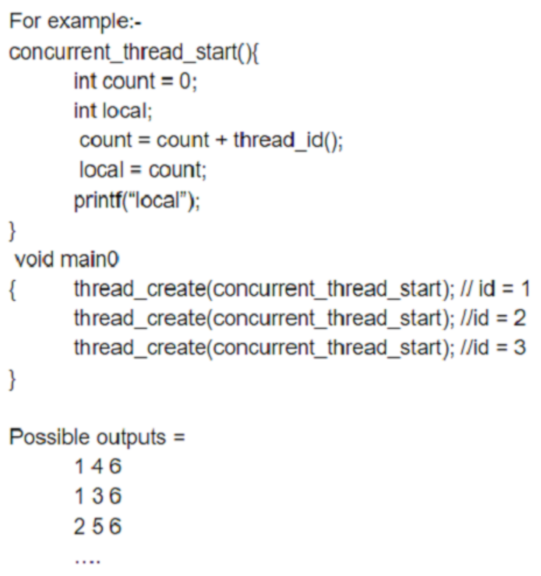
In non deterministic testing, it executes a programme with fixed input many times in hope that faults will be exposed by one of those execution. And its simple so that it is mostly use in practice.

Many factors can contribute to non-deterministic behaviours of any concurrent programs. Example:- variation in message latencies, process scheduling. Race analysis refers to the activity of identifying races. It has found many applications in debugging and testing of concurrent programs. The main observation is that the race analysis can be used to speculate different behaviours.

Suppose in any programme contains three threads T1,T2 and T3. T1 sends(s1) the message to T2 and T3 also sends message (s2) to the T2. T2 receives r1 and r2 messages from both the thread. Now due to same variation in message latencies, it may be possible that the message sent by s2 arrives at T2 before that sent by s1. So, In another execution s2 can be received by r1. So we can say that s2 is in the race set of r1. This is because r1 receives other than s1 message. r2 exits on the how the programme is implemented.

Lets we execute the programme which have multi threads and start test it non deterministically. In the beginning, bugs are everywhere and almost every test case runs fails. Now we need to fix each bug every time and our programme surely needs to be re-tested with new test cases. This process works well but takes too much time.

We can also use mutex or semaphores in these threads, so that the one method will run only after the current method executed successfully.

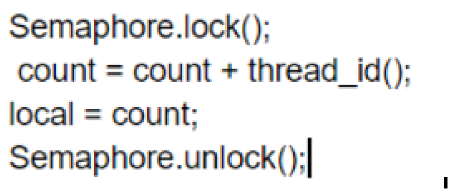


```
For example:-
concurrent_thread_start(){
        int count = 0;
        int local;
         count = count + thread_id();
         local = count;
        printf(“local”);
}
 void main0
{       thread_create(concurrent_thread_start); // id = 1
        thread_create(concurrent_thread_start); //id = 2
        thread_create(concurrent_thread_start); //id = 3
}

Possible outputs =
        1 4 6
        1 3 6
        2 5 6
        ....
```

We can use any semaphore to lock counter and local variables .

So it would be



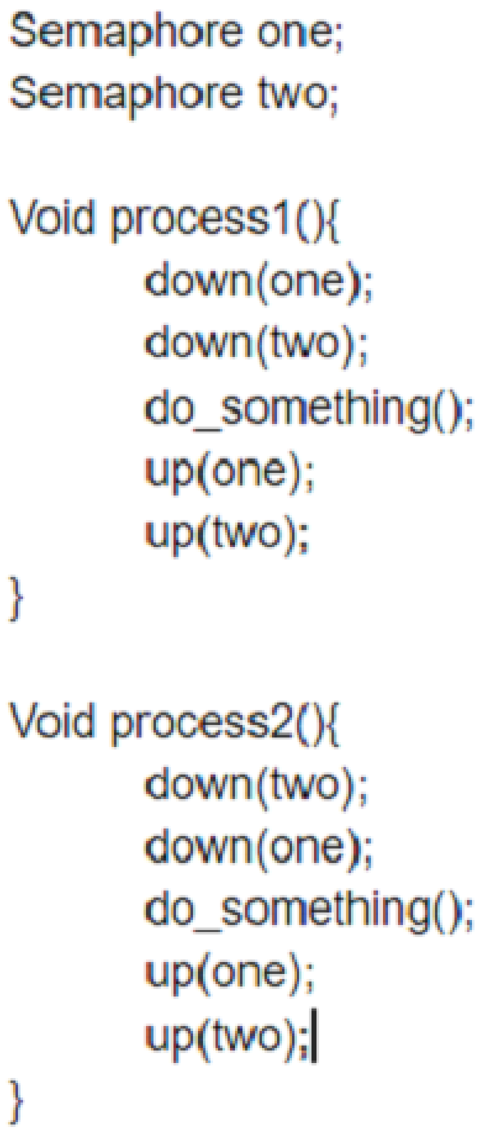
```
Semaphore.lock();
 count = count + thread_id();
local = count;
Semaphore.unlock();
```

This will reduce the number of outputs because another thread can use these variables only after current thread semaphore unlocks it. So by carefully designing a code, we can get upper hand on concurrency bugs.

Sometimes, because using of semaphores, all threads are waiting on an event (like unlocking any semaphore is not done yet) that will never occur, then deadlock occurs.

Deadlock example:

If two processes runs concurrently and simultaneously then if first process has decreased the value of semaphore 1 and second process has decreased the value of semaphore 2, then they both wait to increase value of semaphore but itll never increase. So this is the deadlock example for the concurrent running processes.



```
Semaphore one;
Semaphore two;

Void process1(){
        down(one);
        down(two);
        do_something();
        up(one);
        up(two);
}

Void process2(){
        down(two);
        down(one);
        do_something();
        up(one);
        up(two);
}
```

Functional testing(INPUT/OUTPUT):

Functional testing is that the system is tested against the functional specifications or requirements. The main advantage of functional testing is that it simulates actual system usage and does not make any system structure assumptions.

We have to give input to the functions and examining the output. This testing ensures that the requirements are properly satisfy by application. For the functional testing, we need to use black box testing in which internal logic of the system being tested is not known to the tester.

So the following test involves below steps:-

- We need to identify those functions that the software is expected to perform.
- Create input data for those functions.
- Determine outputs.
- Execute the test cases.
- Last, compare the expected and actual outputs.

Conclusion

After going through methodology and approaches regarding working mechanism of testing a concurrent software, we could say that concurrency testing would help us achieve these set of goals :

- Overall, by keeping the concurrency limited to only few classes or methods, the cost of testing needed will lessen as the scope has now been restricted to a few code elements.
- We can encapsulate the code which we need to review and analyze. Then this portion of software would be only to be reviewed and there isnt any need to analyze the other elements of the software. This saves our testing time drastically.
- It helps in improving the coverage case criterias when it comes to synchronizing the system.
- It allows to vast our domain of compatibility. With new hype of applications running parallelly, widening compatibility of software is right what is needed!
- The cost of testing per test case gets low with concurrency testing.
- Even though this would help us in consistency of concurrency, as and when needed, we can always have an option to test the software sequentially if that is ever required

Contribution

Mohit Savaliya:-

- CONCURRENT FAILURE MODES
- Performance testing and load testing in concurrent programmes
- Challenges while performing concurrent software testing
- Consequences of failure of concurrent software testing for any company
- Non-determinism in concurrent software/programs
- Functional testing(INPUT/OUTPUT)

Dhaval Panjwani:-

- What is sequential software?
- What is concurrent software?
- Comparing hardenability of Sequential Software testing vs Concurrent Software testing
- Testing Concurrent Software through Unit Testing
- Restrict concurrent interactions by using immutability
- Conclusion

Vinay Patel:-

- Abstract
- Creating a Test Plan for Concurrent Software
- Inadequacies of Existing Frameworks
- Enforcement of Threading Disciplines
- Common Problems in Concurrent Software

Bibliography

  - https://www.quora.com/search?q=concurrent+software+testing
  - https://www.cs.umd.edu/class/fall2010/cmsc433/lectures/TestingConcurrentCode$_{J}avaOne$2007$.pdf$
    $//en.wikipedia.org/wiki/Concurrent_{t}esting$

- http://ieeexplore.ieee.org/document/7515488/?reload=true
- http://delivery.acm.org/10.1145/2010000/2002964/p1-souza.pdf
- Distributed reachability testing of concurrent programs by Richard Carver and Yu Lei