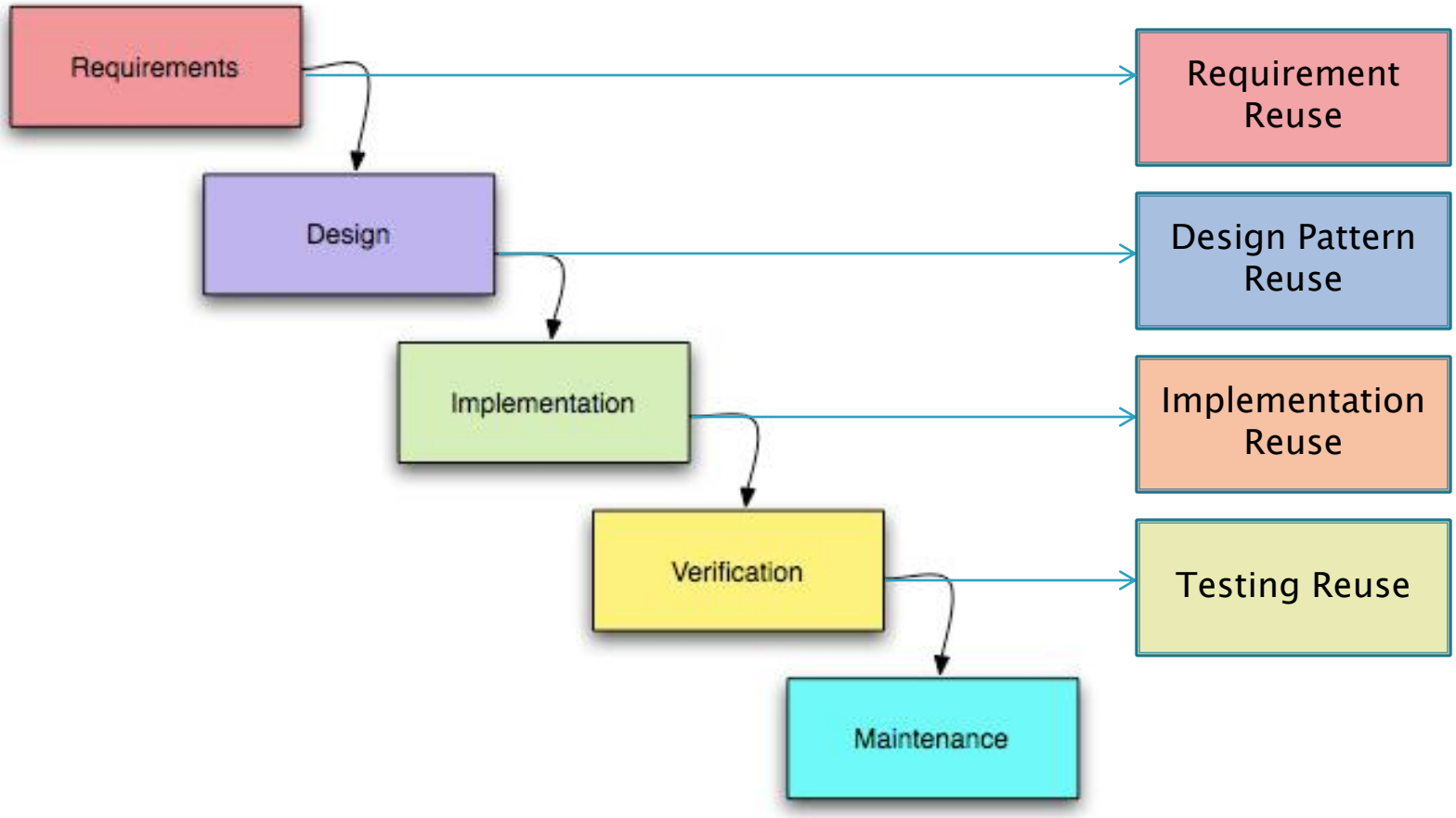
Module 5
Software Testing Reuse (Part 2)

Dr. Shiping Chen

Where are We?



Part of this figure is from: http://en.wikipedia.org/wiki/File:Waterfall_model.png

Outline

- Testing Harness
- iTester – A performance Testing Framework
- Other testing resources for reuse

Test Harness

- A collection of software and test data configured to test a software system by running it under varying conditions and monitoring its behaviour and outputs.
- It includes:
  - Automate the testing process.
  - Execute test suites of test cases.
  - Generate associated test reports.

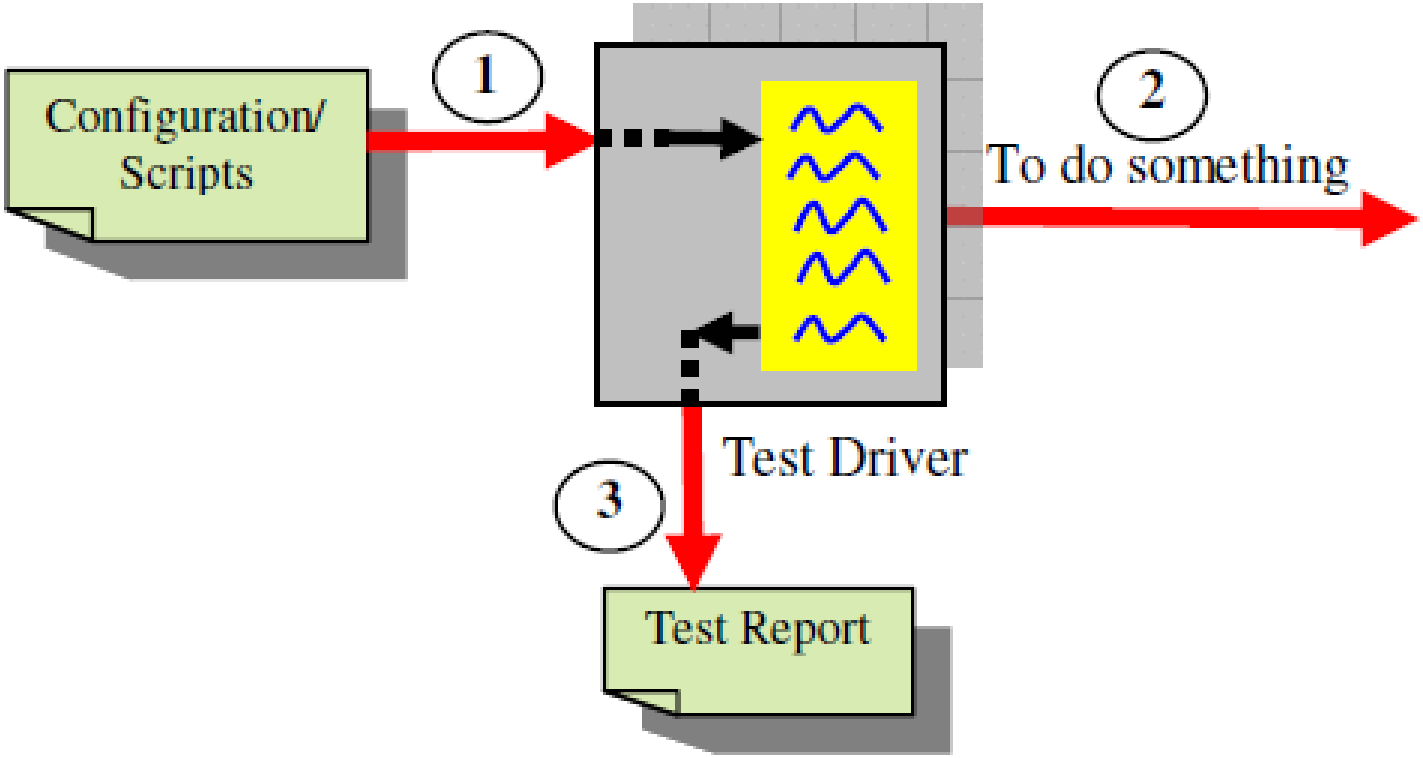
Benefit of reusing test harness

- Increased productivity due to automation of the testing process.
- Increased probability that regression testing will occur.
- Increased quality of software components and application.
- Ensure that subsequent test runs are exact duplicates of previous ones.
- Testing can occur at times that the office is not staffed (ie. at night)
- A test script may include conditions and/or uses that are otherwise difficult to simulate (load, for example)

Key Requirements for Perf-Testing Harness

- Configurable
  - For both testing and application-specific parameters
- Multi-threaded
  - To generate enough loads
- Application-independent
  - To be reused to test different applications
- Automatically generate Testing report
  - To be used for performance evaluation

An Abstract Model for Perf-Testing



From: Shiping Chen *et al*: Yet Another Performance Testing Framework. Australian Software Engineering Conference 2008: 170-179 70-179

Performance Metrics

- **Response Time (RT):** the total time spent by a client in invoking a server function and coming back with a result.
- **Latency:** the total time spent by a message from its sender (source) to its receiver (destination).
- **Throughput:** the total number of jobs (such as packets, messages, and transactions) done within a unit of time (such as second or minute).

Basic Math Required

- $\bar{x}$: The arithmetic mean (average) of the samples measurements, i.e.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (1)$$

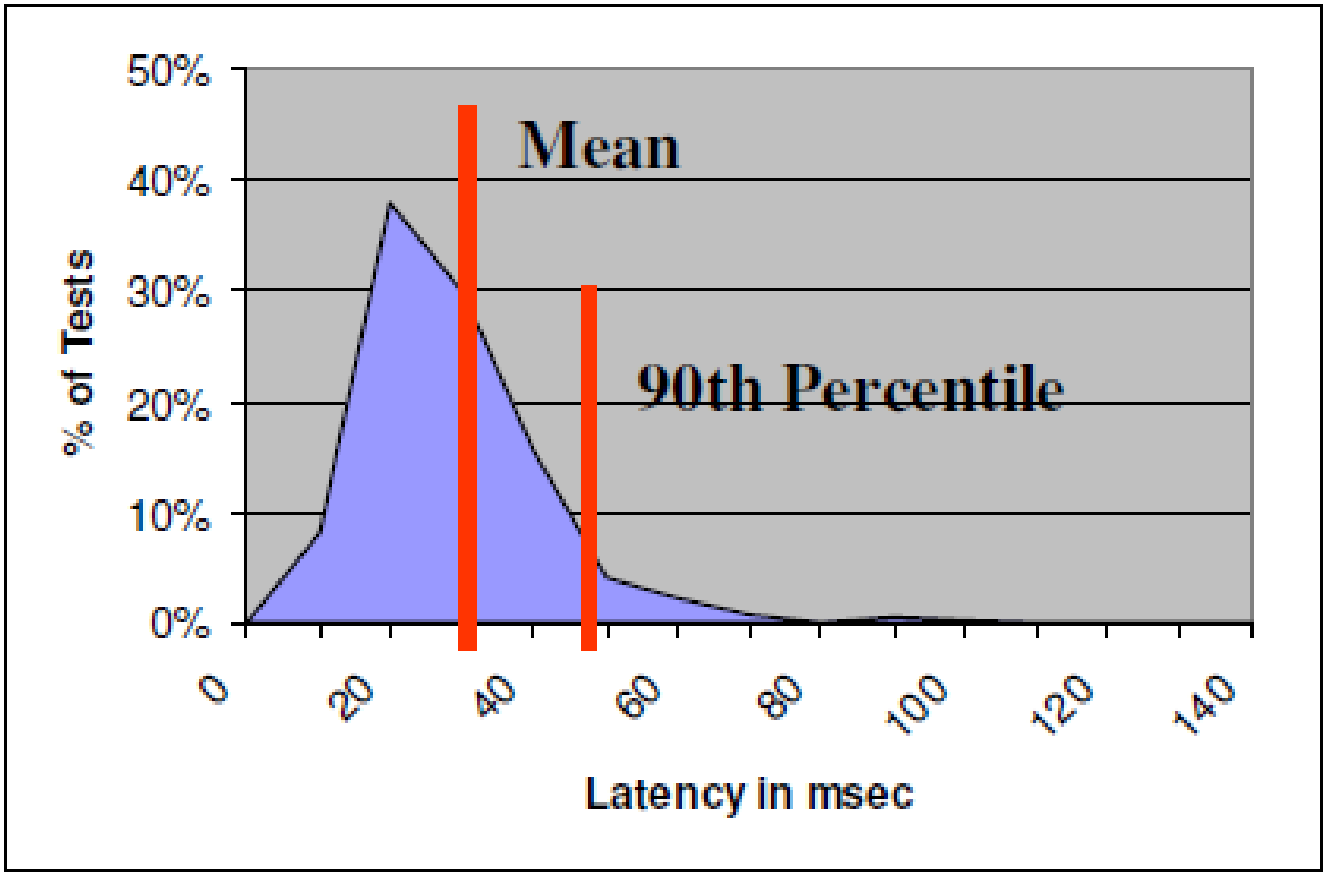
where, x_i is the measurement of sample i and n is the total number of the samples.

- k^{th} percentile of the measurement X: For a certain value x, the probability of a sample of X is less than x is greater than k^{th} %, i.e.

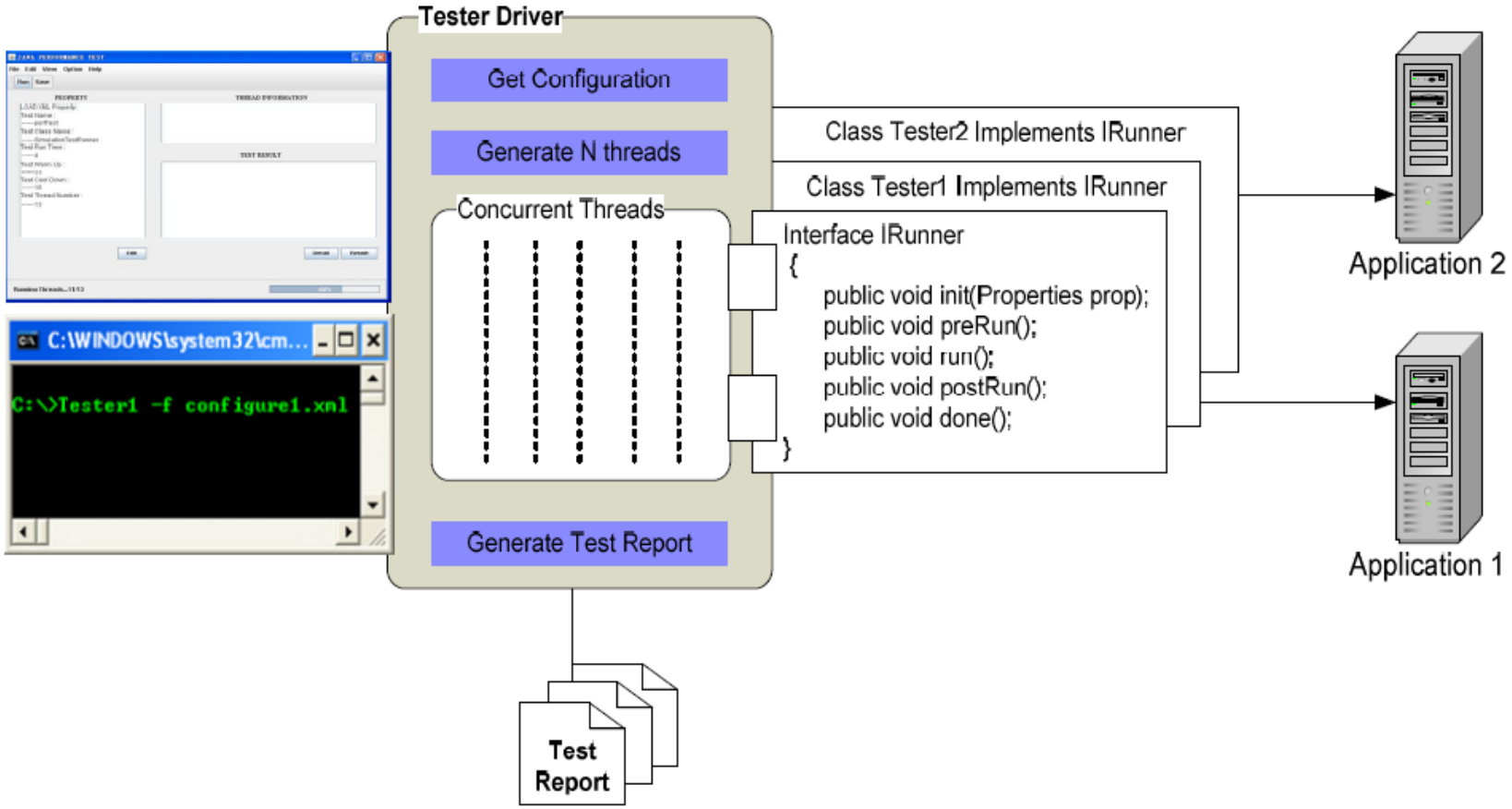
$$P(X \le x) \ge p \qquad (4)$$

where, $p = \frac{k}{100}$

Mean vs. K-th Percetage



Architecture of the test framework



Test.prop

- testName=Hello
- testClassName=HelloTester
- testThreadNmuber=10
- testRuntime=60000 // 60 sec = 1 mins
- testWarmup=10 // 10%
- testCooldown=10 // 10%%

ResultLog.class

```
public class ResultLog
{
    ……
    public void add(long value) {…}
    synchronized public void add(ResultLog rlog) {…}
    public long getAve() {…}
    public void setDistribution(long value)
    {
        int temp = (int) value;

        if(temp<0) temp = 0;
        else if(temp>VALUE_MAX) temp = VALUE_MAX;

        int index = temp/VALUE_UNIT;
        valueDistribution[index]++;
    }

    public void printDistribution()
}
```

Generic Interface: IRunner.class

```
public interface IRunner
{
    public void init(Properties p) throws Exception;
    public void preRun()           throws Exception;
    public void run()              throws Exception;
    public void postRun()          throws Exception;
}
```

Implementation of Irunner: HelloTester.class

```
public class HelloTester implements IRunner  {
   private String ID;
   private Properties prop;
   private Random r = new Random();

   public void init(Properties prop) {
      this.prop = prop;
      this.ID = Thread.currentThread().getName();
   }

   public void preRun() {}

   public void run() throws Exception  {
      try  {
         int t = ((int)(r.nextGaussian()+2.0))*2000;
         System.out.println(" Hello " + ID + " to sleep for " + t + " ms" );
         Thread.currentThread().sleep(t);
      } catch(Exception e) {}
   }

   public void postRun() {}
   public void done() {}
}
```

How to use Runner.class in TestDriver.class

```
public void run()
{
    // instance the common interface IRunner
    IRunner runner = …

    // before test: prepare test
    runner.init(this.config);

    // during test
    for(alarm.start(); !alarm.isTime(); numTx++) {
        runner.preRun();
        timer.start();
        runner.run();   // To run one iteration of your test
        latency = timer.stop();
        if(alarm.isTesting()) localLog.add(latency);
        runner.postRun();
    }

    // after test: add the local log to the global log
    runner.done();
}
```

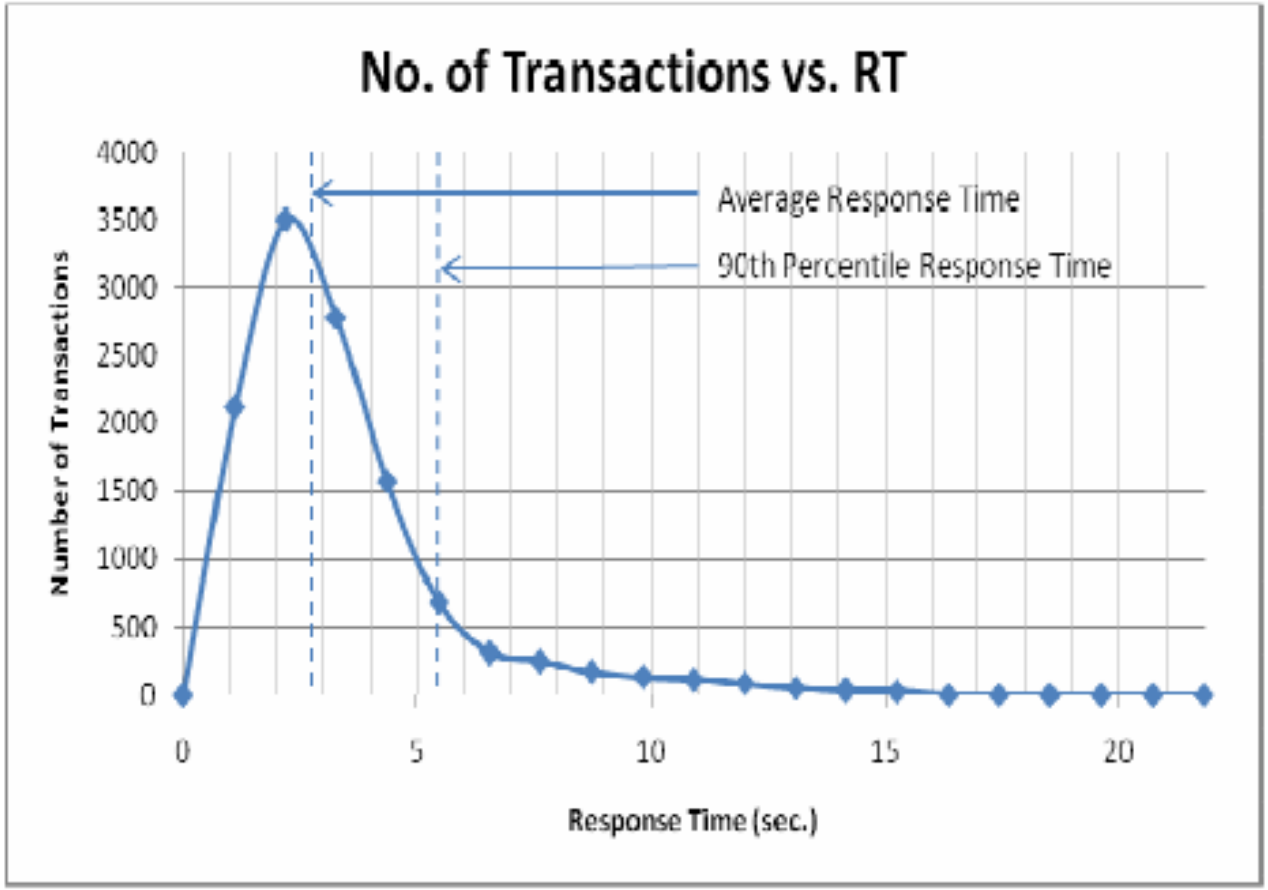
TestManager.class

```
public class TestManager {
    final static boolean debug = true;
    Properties prop = new Properties();
    TestDriver testDriver;
    ……
    public void run() {
        try {
            int testThreadNmuber = Integer.parseInt(prop.getProperty("testThreadNmuber"));
            Thread threadArray[] = new Thread[testThreadNmuber];
            testDriver = new TestDriver(prop);

            for(int i=0; i<testThreadNmuber; i++)  {
                threadArray[i] = new Thread(testDriver);
                threadArray[i].start();
            }

            for(int i=0; i<testThreadNmuber; i++) {
                threadArray[i].join();
                if(debug) System.out.println("Thread-" +i+ " : exited");
            }
        } ……
    }
    ……
}
```

Put it all together

```
class Tester
{
    public static void main(String[] args)
    {
        if (args.length<1)
        {
            System.out.println("Usage: java Tester test.prop");
            System.exit(1);
        }
        System.out.println("test config fileName = " + args[0]);

        TestManager manager = new TestManager(args[0]);
        manager.run();
        manager.report();
    }
}
```

Test Report

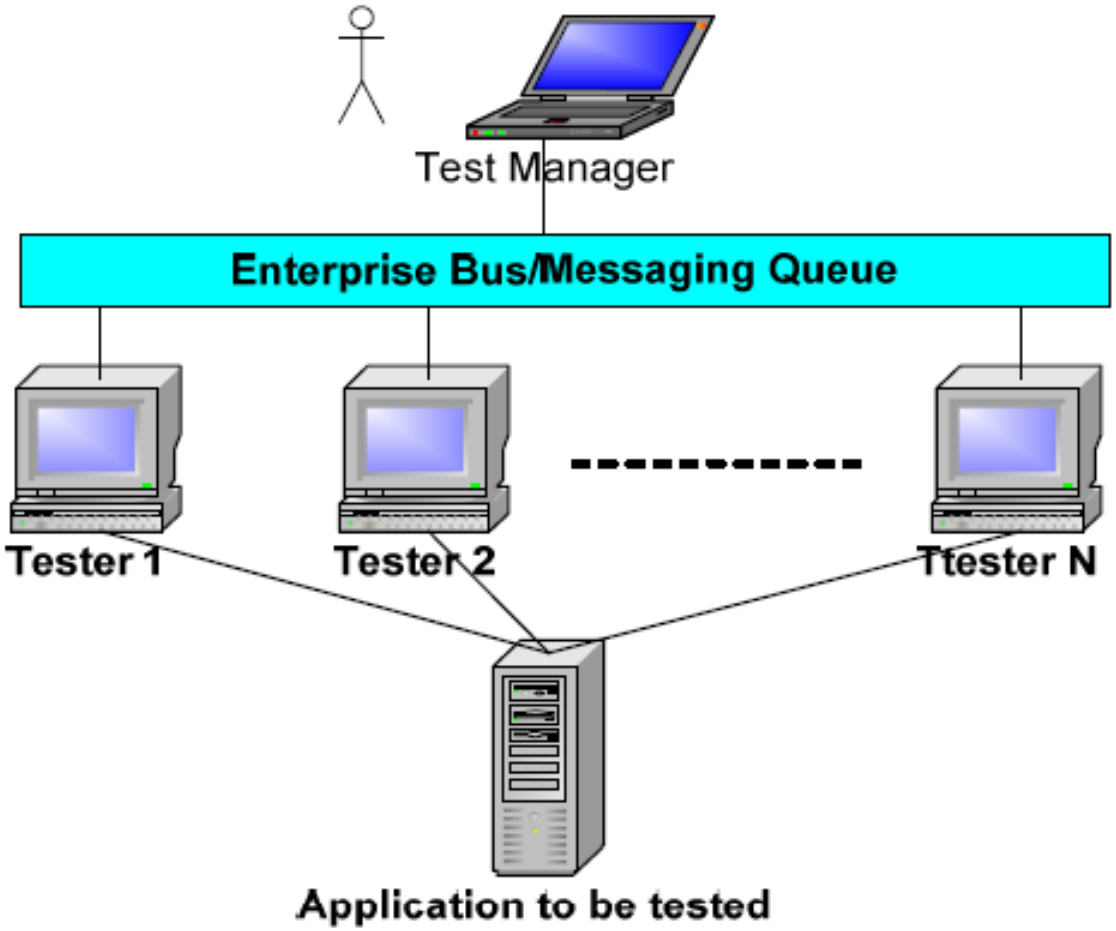


Run it !

C:\Windows\system32\cmd.exe

```
C:\Users\che168\Development\Test\PerfTester>java Tester test.prop
test config fileName = test.prop
Tester.init(test.prop) called
-- listing properties --
testName=Hello
testWarmup=5
testCooldown=5
testClassName=HelloTester
testThreadNmuber=10
testRuntime=10000
Tester.run() called
testThreadNmuber = 10
To instance a test driver
 Hello Thread-4 to sleep for 2000 ms
 Hello Thread-7 to sleep for 2000 ms
 Hello Thread-8 to sleep for 4000 ms
 Hello Thread-6 to sleep for 2000 ms
 Hello Thread-3 to sleep for 0 ms
 Hello Thread-0 to sleep for 0 ms
 Hello Thread-1 to sleep for 2000 ms
 Hello Thread-5 to sleep for 0 ms
 Hello Thread-2 to sleep for 2000 ms
 Hello Thread-5 to sleep for 2000 ms
 Hello Thread-0 to sleep for 4000 ms
 Hello Thread-3 to sleep for 4000 ms
 Hello Thread-9 to sleep for 2000 ms
 Hello Thread-4 to sleep for 0 ms
 Hello Thread-4 to sleep for 6000 ms
 Hello Thread-7 to sleep for 0 ms
 Hello Thread-7 to sleep for 4000 ms
 Hello Thread-6 to sleep for 2000 ms
 Hello Thread-1 to sleep for 2000 ms
 Hello Thread-2 to sleep for 4000 ms
 Hello Thread-5 to sleep for 2000 ms
```

How to test a big app server?



Further Information about iTester

- http://sourceforge.net/projects/itester/

Other Performance Testing Resources

- HP LoadRunner: The best web application Tester
- Grinder: http://grinder.sourceforge.net/
- JMeter – Load and Performance tester
- HammerDB: http://hammerora.sourceforge.net/
- …

