LAB 4: Integration Testing

1. OBJECTIVES

1.1 Integrate subsystems
1.2 Perform system tests

2. INTRODUCTION

2.1 In the last Lab session, the teams built and unit tested the various subsystems.

2.2 In this Lab session, the subsystems shall be integrated in an incremental fashion to form larger aggregates or subsystems. During integration, the focus shifts from individual subsystems to the interfaces between subsystems.

2.3 Verification shall take place as the subsystems are being integrated. Individual subsystems may work in isolation, but may not work correctly when integrated.

2.4 When all the subsystems are successfully integrated, system tests shall be performed to verify functional, as well as, non-functional behaviour.

3. PROCEDURE

3.1 *Integrate subsystems*

3.1.1 From the candidate architecture (Lab #2) and the interaction / coupling between subsystems, determine the integration sequence.

3.1.2 All subsystems are to be reviewed for test / integration readiness before being accepted for integration. Each subsystem must have been unit tested to demonstrate its stability.

3.1.3 Integrate subsystems pair-wise. Drivers and stubs from Lab #3 can be used to simulate other subsystems not yet integrated.

3.1.4 Test the integrated subsystems according to the appropriate segment of the usage scenarios.

3.2 *Perform system tests*

3.2.1 When all the subsystems have been successfully integrated into a single whole, perform functional tests for the entire system, according to usage scenarios. The system's interfaces to external systems and hardware should be carefully tested.

3.2.2 Perform non-functional tests on the system, to verify that the performance requirements can be met. A load test tool should be used for automated testing.

3.2.3 The system test team will receive the binaries, configuration files and scripts from the development team.

4. DELIVERABLES

- Integration test stubs and drivers
- Database scripts for injecting test data
- Test cases for system level functional tests
- Test scripts for performance test / load test
- Discuss the testing strategies, testing techniques and (automatic) testing tools you have adopted in the course.

5. REFERENCES

- Object-Oriented Software Engineering: Using UML, Patterns and Java – B. Bruegge and A. Dutoit
- jMeter Documentation - http://jmeter.apache.org