University of Liverpool

Computing Services Department Testing: Policy and Guidelines

Reference Number	CSD-STR-005
Title	Testing: Policy and Guidelines
Version Number	1.1
Document Status	Open
Document Classification	Public
Effective Date	29 March 2017
Review Date	29 March 2018
Author	Simon Long, Computing Services Department
Comments	29/03/2017 Annual Review

Version Control

Version Number	Comment	Date
0.1	Issued to Application Services manager for review	1st November
0.2	Incorporated further changes from internal CSD review	5th December
0.3	Document updated to reflect 2015 regression testing focus.	8th January
0.4	Updated	6th May
0.5	Incorporated comments from John Cartwright following a review	12th June
1.0	Issued to be published on IT regulations website	2nd July
1.1	Incorporated comments by Steve Tapril	16th July

Introduction

1. Purpose of document

This document is split into two sections and describes the two key elements to the approach and vision of testing within the Computing Services Department:

- Testing Policy
  - *A high level view of the values that underpin the approach to testing*
- Testing guidelines
  - *Provides guidelines and good practice around how testing should be undertaken, what needs to be considered, and the various test stages that can be adopted.*

This document will be reviewed on a regular basis (at least every 6 months). The testing guidelines are intended to have an initial 2 year life span at which point they will be revised in accordance with the maturity of the CSD Testing Service function at that time. However, given the evolving nature of technology, testing techniques and processes, the document will be subject to change based on lessons learned from application to real scenarios.

Testing policy

1. Purpose of policy

The testing policy describes the high level approach that will be undertaken towards System and Software Testing (covering both validation and verification). It describes the test policy objectives, the tangible benefits, performance indicators, quality targets and approach to test process improvement. This policy is intended to provide the foundation on which all subsequent test strategies (including the Testing Strategy section of this document) and low level test plans will be built.

2. Mission statement

"To ensure the quality characteristics of mission critical existing functionality offered to both students and staff is not compromised by any remediation or development activities"

"To verify, validate and measure the quality of software or systems being delivered and to ensure that testing in all forms and all stages is performed effectively and passionately to demonstrate that the product satisfies the end-user requirements resulting in client satisfaction"

3. Benefits of policy

The aim of this testing policy is to deliver the following tangible benefits to the University:

- Creation of an approach for regression testing focussing on ensuring existing functions and processes are validated during an upgrade / development project;
- Testing that is directly linked back to the higher level functional and non-functional requirements;
- Testing that is a consistent and repeatable activity rather than a bespoke activity for each project/change, i.e. that we are not re-inventing the wheel but equally not constrained by a singular restrictive method. Encouraging a 'freedom to test' mind-set is the ethos that this policy and subsequent strategy will encourage;
- Effective team-working through the adoption of standardised principles, processes and deliverables across the organisation;
- An efficient, lean and responsive testing organisation influencing other areas by embedding best practices;
- A decision making framework under which testing (planning, design, execution & closure) can take place;
- A test resource that documents and captures the key elements of the testing process and make accessible those testing deliverables which form part of the wider project delivery.

4. Test policy objectives

The following objectives are intentionally high level in this section but are expanded further in the later section: *Testing Strategy: Testing Objectives.*

- To plan testing early in the project so that testing activities may be started as soon as possible in the project life cycle, and resources planned accordingly;
- To identify and engage key stakeholders (both academic and technical) in the testing process so that testing priorities and risks may be understood from all aspects of the business and technology;
- To adopt a risk based approach to testing to help target, focus and prioritise testing with the added benefit of making efficient use of resources;
- To focus on ensuring defect discovery (be they code defects, requirement defects etc.) takes place as soon as possible to prevent the escalating costs of identifying and fixing bugs later on in the project/software lifecycle. - This will include (but is not limited to):
    - *Document reviews, e.g. Opportunity Assessment Documents, Business Requirements, Functional and Technical Specifications, Test Plans. This will also involve engaging stakeholders to confirm key Business Acceptance Criteria is defined and approved;*
    - *Reviewing of Prototypes produced by the agile development team.*

- Ensuring full traceability of testing coverage back to original business requirements to ensure key business requirements are covered by testing;

- To also focus on testing the 'unwritten requirements' of a system to identify any defects that users may find. Unwritten requirements cover areas such as user behaviour, using the system in the context of the end users rather than how it has been designed / built:
  - *"I know the requirement says the user should perform x function to generate their desired result, but what if a typical users decided to perform y function instead"*

- Testing to adhere to guidelines, standards and processes described in an overarching Testing Strategy and associated Test Plans which:
  - *Identifies and addresses risks for systems and the way testing mitigates those risks;*
  - *Describes in detail the testing types, test process and methods to be adopted throughout the project*

- To document all testing activities to an agreed, appropriate level, in accordance with the Testing Policy
  - *The intention is to create an environment where key testing deliverables / test ware are captured, audited, and stored in a way they can be re-used for future projects;*
  - *Where possible all participants in the testing effort should look to reuse existing test documentation.*

- Carry out testing activities making the most efficient use of available resources (CSD for system testing and wider University resources for Acceptance testing). We will do this by:
  - *Increasing the understanding of the importance and criticality of testing in the Software Development Lifecycle with the use of mechanisms such as workshops, roadshows, 1-2-1 training sessions;*
  - *Communicating clearly and in good time with our stakeholders exactly what the requirement is for specific resources;*
  - *Appropriate informal and formal training for all involved in Testing;*
  - *Actively encourage participation from University Staff and Students in the testing process, providing a platform to allow users to develop skills and knowledge in the subject of Testing whilst at the same time making the most of any available resources for testing projects;*
  - *Encouraging open communication especially around issues, risks, concerns;*
  - *Adopting a "No Blame" culture;*
  - *Developing a better-rounded testing team to enable working across CSD departments, teams, and the wider University business areas.*

5. Evaluation of testing (Key Performance Indicators)

We will measure the quality of testing at each stage of the software development lifecycle including the transition to live. The measurement of testing will use various methods such as monitoring of live services for trend analysis. Further details will be specified in the Test Strategy.

6. Approach to test process improvement - “Measure > Review > Improve”

We will constantly measure, review and improve the quality of the testing process. The quality of the testing process will determine the success of the testing effort therefore it is vitally important to have a clearly defined, robust set of testing processes/standards. This will be achieved by mechanisms such as workshops being held at the end of projects to understand how the process can be improved.

Testing guidelines

1. Purpose of guidelines

The Testing Guidelines describe suitable approaches for all testing activities required as part of the CSD development life cycle. The intention is to provide a generic framework for testing to be followed across all Projects / Changes that CSD manage. This document is not intended to address individual testing requirements for a specific project. In these instances a project level test strategy and/or test plan should be created where the finer details of the approach will be defined.

These guidelines inform staff such as Project Managers, Team Leaders, Testers, and Developers of the fundamental structure of the testing process. This includes the testing objectives, methods of testing new and existing functions, testing roles and responsibilities, which types of tests should be performed, and which entry and exit criteria apply. It is a requirement that the testing process is applied for all projects where testing is required, unless it is specially stated to the contrary. In this instance the deviation should be documented in the lower level test plan and will be required to be approved by the CSD Test Manager.

2. Definition of testing

The definition of testing within The University of Liverpool is described below.

- To validate and improve the end user experience (student and staff) of the product, ensuring it is fit for purpose;

- To prevent the migration of defects throughout the software development lifecycle by testing as early in the lifecycle as possible (e.g. in the requirements definition stage) using different techniques (e.g. testing unwritten requirements such as user behaviour of the system). It is not satisfactory simply to discover any defects; the aim is to prevent defects happening in the first instance;

- Ensuring that a product satisfies its core requirements. Verifying and validating this statement by constantly asking the question “is this what the customer requires?”

  - *This may involve early prototyping sessions / wireframe demonstrations during development;*
  - *Tests designed (where possible) directly from the requirements documentation;*
  - *Involvement from the senior business partners in the User Acceptance Stage and possibly earlier if deemed appropriate.*

3. Risk based approach to testing

It is recommended that a Risk Based approach should be undertaken in all aspects of the testing lifecycle. Depending on project objectives, testing should always be undertaken with a view of identifying and mitigating any risks the Project / Change may introduce. Obtaining a view of any such risks will shape the testing required. In simple terms: if one part of a system is deemed to have a potentially high-risk impact if it fails then this area should be the initial focus for the test effort.

The designated test lead for a project would select the degree of quality risk mitigation desired. The level of risk will determine the test effort and test sequencing. Test results using this method can be reported in terms of mitigated and unmitigated risks. The testing scope for a Project / Change should take into account various factors. A mixture of those described below may be utilised depending on the size and scope of a particular project:

- Review the **potential product risks** and focus testing on the high risk areas (known as Risk Based Testing);
- If the system under test is already in existence and therefore a change is being implemented, a review of the most **frequently used functions** can be undertaken and testing could be focussed at testing these areas:
  - *Defects in these areas are more likely to cause bad feedback from customers if they were to go undiscovered than bugs in non-frequently used areas;*
  - *Note: this is not to say testing should not focus on areas that are infrequently used, rather it is a case of priorities testing effort to key areas initially;*
- **Project criticality based test effort.** The levels of testing will be based on the criticality of the project. This rating is directly correlated to the rating given to the project at the Opportunity Assessment Document stage of the CSD Software Development Lifecycle.

4. Testing objectives

The following objectives for testing describe the key principles which should be used to formulate approaches for all testing activities

- Plan testing early in the project so that testing activities may be started as early as possible in the project life cycle, and resources planned accordingly. This will be achieved by:
  - *Ensuring every change to university critical software and systems are validated by careful test design before being transitioned into Live Service;*

- *Commencing test design early on in the project so that any omissions to requirements can be detected and ambiguity in requirements can be clarified;*
- *Checking at every stage that the product satisfies its requirements. Verifying and validating this statement by constantly asking the question "is this what the customer requires?"*

➢ Identify, and engage key stakeholders (both academic and technical) in the testing process so that testing priorities and risks may be understood from all aspects of the business and technology.

➢ Regularly communicate with relevant stakeholders providing updates whilst also requesting participation at key checkpoints where clarification is required on such areas as requirements.

➢ Adopt a risk based approach to testing to help target, focus and prioritise testing with the added benefit of making efficient use of resources.

➢ Focus on ensuring the defect discovery element of testing takes place as soon as practicably possible to prevent the escalating costs of identifying and fixing bugs later in in the project/software lifecycle. This will include (but is not limited to):

- *Document reviews, e.g. Opportunity Assessment Documents, Business Requirements, Functional and Technical Specifications, Test Plans. This will also involve engaging stakeholders to confirm key Business Acceptance Criteria is defined and approved;*
- *Reviewing of Prototypes produced by agile development teams;*
- *Ensuring defects identified during any stage of the testing process are documented and tracked effectively to closure, ensuring all parties required to assist in the triage of the issue are involved from the very outset of the defect resolution cycle.*

➢ Ensuring full traceability of test coverage back to original business requirements to ensure key business requirements are covered by testing. *Note: Where business requirements cannot be met it is highly recommended these limitations are documented in the relevant Test Plan and wording provided to explain the limitation*

➢ Document all testing activities to an agreed, appropriate level, in accordance with the Testing Policy:

- *The intention is to create an environment where key testing deliverables / test ware are captured, audited, stored and created in a way they can be re-used for future projects;*
- *Where possible all participants in the testing effort should look to reuse existing test documentation from previous testing projects. Please contact CSD Testing Services for further information.*

➢ Carry out testing activities making the most efficient use of Computing Services Department resources (for system testing) and wider University resources (for User Acceptance). We will do this by:

- *Increasing the understanding of the importance and criticality of testing in the Software Development Lifecycle with the use of mechanisms such as workshops, roadshows, 1-2-1 training sessions;*

- *Communicating clearly and in good time with our stakeholders exactly what the requirement is for specific resources;*
- *Appropriate informal and formal training for all involved in Testing;*
- *Ensuring a 'open door' policy into testing i.e. encourage anyone who expresses an interest to be involved in any level and provide a way of involving such volunteers in the testing process;*
- *Encouraging open communication especially around issues, risks, concerns;*
- *Adopting a "No Blame" culture;*
- *Creating a testing culture of a "test to break" attitude, with the aim to find the existence of defects. Defects will only be discovered if you make it your objective to find them;*
- *Developing a more well-rounded testing team to enable working across CSD departments, teams, and the wider University business areas.*

5. Test levels

The use of test levels (or more commonly known *test phases*) promotes mitigation of quality risk as early as possible and to the highest practical extent. The table below describes the various test levels that unless stated in the relevant Test Plan should be undertaken for all Projects / Changes.

Note: each Test Level should have a dedicated representative responsible for performing / overseeing the activity

The notable exception to this is the *System Integration Test* Level whereby this will only be required where the Project / Change introduces changes that affect multiple systems or a systems interface to another system (e.g. changes made in Spider that would impact information flowing to VITAL would require a level of System Integration Testing)

An example would be where a new screen is added to VITAL only but the information entered on this screen is only written locally to the VITAL database. System Integration would not be required in this instance

Level	Owner	Objective	Key areas of testing
Unit	Development	• Detect defective code in units • Reduce risk of unit failure in Live Service	• Functionality • Resource utilization
System	Development	• Detect defects in end-to-end scenarios concentrating on systematically validating whether each function performs as expected or not • Assist in mitigating risk of unmet business requirements	• Functionality • Performance o Reliability • Usability • Resource utilization, • Maintainability • Installability, portability interoperability
System Integration	Development	• Detect defects in unit interfaces • Reduce risk of dataflow and workflow failures in Production	• Functionality • Data quality • Unit interoperability • Compatibility • Performance
Acceptance (User, Business, Operational)	Senior Business Partner (who requested the deliverable of the Project / Change	• Demonstrate the product works as expected in a real life environment using real life operational scenarios • Detect defects in user workflows • Ensure key business acceptance criteria are met	• Functionality in context of normal usage • Operational Processes • Pre-agreed acceptance criteria

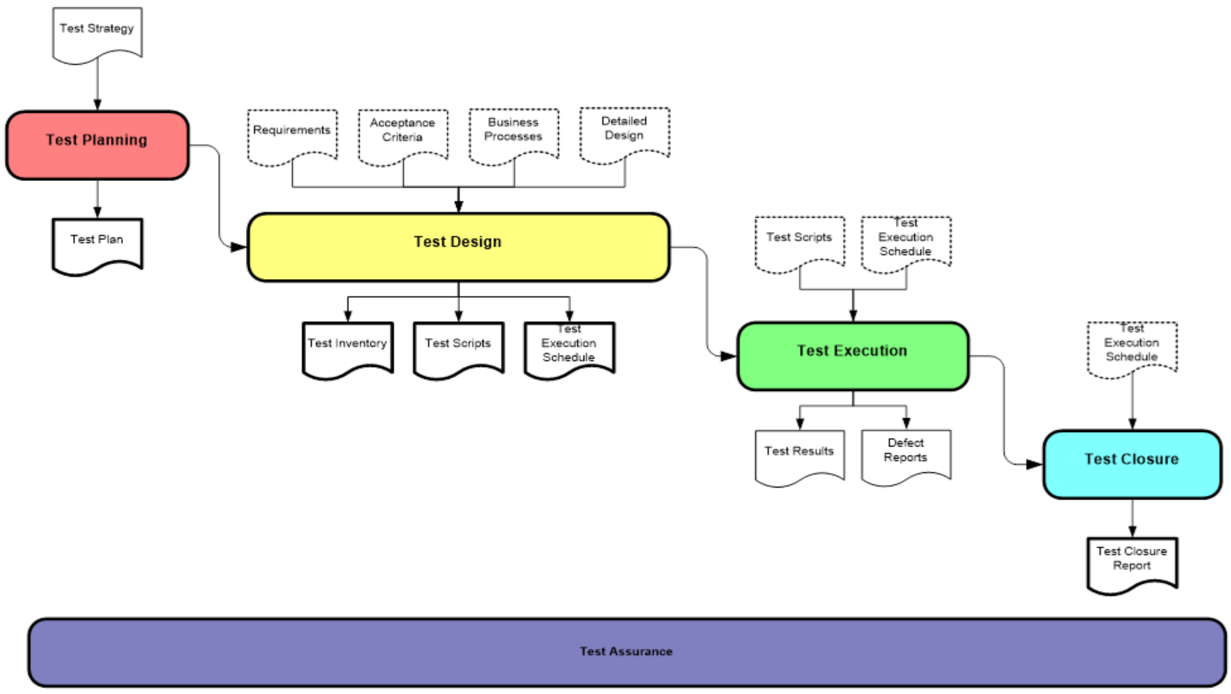
6. The fundamental test process

The Fundamental Testing Process comprises four stages which are required to be followed in sequential order. Regardless of the Test Level (Unit, System, UAT etc.) the stages should be followed in this logical order for any Test Level to provide maximum benefit to the testing process and quality assurance overall.

Whilst at first glance using the process may appear excessive for some projects, the stages can be as formal or informal as required. This underpins the Context-Driven-Testing philosophy which states *"The value of any practice depends on its context"*. As long as the basic principles of the process are being followed then the testing process will produce the required result. The test process may vary in terms of duration and activities involved depending on the development methodology being used by the development teams. For example, Application Development are very much Agile in their development methodology whilst other areas such as Student Systems are very much traditionally waterfall / V Model in their development methodology.

Below is an overview of each stage of the Fundamental Testing Process and also the test deliverables that should be produced as the outputs of the stage. For further definitions of each stage please refer to the CSD Testing Services website: CSD Testing Services

1. **Test Planning**
2. **Test Design**
3. **Test Execution**
4. **Test Closure**



7. Guiding principles

To aid and assist in each of the test stages, the following sections provide some guiding principles on how to ensure each stage is performed effectively and to obtain the optimum benefits (both to testing and the wider project delivery).

Test planning

- Ensure resources required to support the Test Design and Test Execution stages of the test process are agreed and assigned early enough to provide adequate test preparation.
  - *This can be done by holding a Test Initiation Meeting at the very start of the Project Change*
  - *Please refer to the Master Test Plan Template for details on the Test Initiation Meeting*
- Where possible (and indeed practical), a Test Plan for each Project / Change should be created which should be reviewed with end-users / user groups and all affected stakeholders
  - *A Template has been created to aid test representatives in the creation of such test plans. Please contact CSD Testing Services to obtain a copy of the Master Test Plan Template*
  - *The plan should not be onerous in completing but rather should allow the test lead / representative to capture the key points. Try and keep a test plan to on average 5 pages.*
  - *CSD Testing Services can provide assistance in completing a Test Plan for the relevant test level.*

Test design

- Different projects may require different levels of test documentation. Ensure any test cases that are created are done so only if they are going to add value.
    - *Not all projects will need reams and reams of test cases creating, however almost all projects will require some form of documented tests.*
- Ensure tests are designed, reviewed and approved before testing begins. Test Cases should contain:
    - *If applicable, the reference to the requirement tested.*
    - *Description of the function or feature being tested.*
    - *Name of person who designed the test & date.*
    - *Test data & instructions to run the test e.g. "for this this you need a windows 7 laptop with a student type of post graduate"*
    - *Expected results.*
- When designing and executing tests, testers should approach all testing with a "test to break" attitude, with the <u>aim to find the existence of defects</u>
    - *It is important to remember that as well as testing the written requirements, tests should be designed to test the unwritten requirements (such as typical user behaviour, random scenarios that may push the system to its limit)*
- Test Scenarios should be the first part of the test design process. Scenarios should be captured in a "Test Scenario Inventory".
    - *Scenarios should be kept very short and used a as a high level view describe the function or process to be tested.*
    - *Scenarios should be written so that they can be easily reviewed by anyone wanting to understand what is being tested. E.g. "To test that a member of staff can approve a course module abc on Tulip"*
    - *A template can be found on the CSD Testing Services website*

Test execution

- Prior to commencing any Test Execution phases where testers may be unfamiliar with the testing process, it is worth hosting an informal meeting beforehand running through the key aspects of what is expected i.e.:
    - *How the testers will record the results?*
    - *Walkthrough of some example test cases to make them feel more comfortable. This will avoid a lot of downtime during the testing window with testers asking questions.*
    - *How will testers record defects / observations? Walk through the process of how issues will be reported and what information is required.*
    - *How long do the testers have to complete the testing? By communicating the timescales this will focus testers minds ensuring they have an understanding they only have a finite time to complete their assigned tests*

- Ensure the development team have been contacted to provide support in case of any queries found during testing. This also applies to the business area that requested the project i.e. the subject matter expert.
- Try and avoid (if possible) testing across multiple buildings or locations. Attempt to bring the testers together in one room or location.
    - *Where this is not possible, use the new VOIP system to communicate between key contacts regularly i.e. have an open conference call where people can dial into and raise any issues.*
    - *Also avoid cramming testers in to a small room. This can stifle test progress and would put people off volunteering for testing in the future*
- Where possible reward any volunteer testers for their efforts.
    - *If the testers are students, such rewards may take the form of print vouchers, an opportunity to update their HEAR with the relevant skills gained during their testing efforts*
    - *Other rewards may simple include the sustenance during the testing required (e.g. provide lunches / refreshments etc.)*
    - *Each test phase will analyse if rewards are needed to incentivise any required testers and submit this for approval to the head of application services.*
- In advance of producing a Test Progress report for the first time, ensure you produce a draft and possibly send out a copy to Project / Change stakeholders in advance to ensure the information to be captured is what they need.
    - *There is no point in producing multiple metrics if the audience are only interested in certain elements. This **WILL** save you time!*

Test closure

- Ensure that all tests that can be executed are executed.
    - *Review the status of any tests that cannot be run and ask the question "can these really not be executed? If not why?"*
- Ensure you have <u>documented reasons</u> why any tests cannot be completed
    - *e.g. "5 tests could not be started before the server abc has been decommissioned"*
- In advance of any Go/No Go quality gates at the end of testing, take time to review results on your own and ensure you are confident what message you will be communicating.
- Chase down any defects that remain open after testing has finished.
    - *If any defects are open then reasons for defects being open will need to be given.*
    - *Stakeholders will understand these points as long as you have documented reasons.*

8. Approach to manual testing

Manual testing within Computing Services Department is an essential part of the testing approach, given the volume and diversity of the applications being developed, supported and utilised by the population on and off campus.

There are two test techniques to undertake manual testing which have been detailed below. Each technique can be utilised appropriate; however the points below give a standardised suggestion of which types of Projects / Changes these can apply to. Referring again to the philosophy of Context-Driven Testing, this is not meant to be a prescriptive approach to testing, moreover giving test representatives ideas which they can apply to their own project using one technique or a combination.

Script based testing

- Typically used when testing Web Based Applications and /or systems where there is a set of clearly defined and document requirements and associated design specification
- This is the traditional method of designing test scenarios and test cases in advance of performing tests, ensuring an expected behaviour is defined in the test scenario and / or the test case.
- During the execution of each test, the defined test steps (*what the tester is required to follow to perform the test*) would be followed systematically without diverging away from what the test is asking the tester to perform.
- If the expected result matches the actual result (*what the tester has observed*) then the test would be marked as Passed.
- If there is a difference between the expected result in the test and the actual result, then the test would be marked as Failed (*regardless of suspected root cause*).
- Once a test is marked as Failed a defect would then need to be recorded and passed to the development team for triage and fixing.

Exploratory testing

- This is a relatively modern technique which is used when the system being tested can be accessed in multiple ways on multiple systems by multiple devices, or simply has too many functions to be systematically tested given the time and resources constraints that may apply to the project.
- As part of these guidelines, it is recommended this technique is adopted particularly for any Projects / Changes involving Mobile development.
- Exploratory testing removes the dependency to create a list of cases in advance of performing tests, rather the testers explore and navigate around the system / application and :
    - *Record what actions they have taken*
    - *Record any observations they have uncovered*

- Given the Mobile market and the proliferation of devices and operating system available, exploratory testing is the most pragmatic way of testing any mobile developments in the most efficient way.
- It should be noted however that where possible *script based testing* should be used in conjunction with *exploratory testing* i.e.
    - *Script based testing would aim to test key requirements*
    - *Exploratory based testing would test the app to a greater depth across a larger geography of the device / browser population*

9. Approach to automated testing

The focus of these guidelines and the initial remit of CSD Testing Services is aimed at introducing and embedding the key principles of manual testing to CSD and associated business areas. Once the principles and associated processes of manual testing have been implemented and suitably matured the aspirational goal is then to look at how automated testing can complement and replace existing manual testing processes.

However whilst automation of testing activities will not formally be part of the day to day processes, CSD Testing Services will always be reviewing ways to improve the testing framework. Further regular reviews of this policy are planned and the approach to automation may be updated to reflect the approach at that time.

Technical Security Testing

When considering testing for any Projects or Changes, Test Managers / Test Leads / Team Leaders also need to take into consideration any technical security implications the Project / Change will introduce.

Where activities are undertaken with University critical systems and sensitive data, it is mandatory to undertake technical security reviews and assessments. Security Review(s) and Assessment(s) are required to ensure all vulnerabilities are detected **prior, during and on a periodic basis**. A request should be made to undertake a Technical Security Review as part of the overall plan of testing.

Note: **Security Testing is required to be conducted separately and independent. Please contact CSD Testing Services if you have any queries in the first instance. For more information please refer to the Information Security Review Policy.**

If such a test is required, a Security Review and Assessment (SR&A) form needs to be completed and submitted thereon.

10. Roles and responsibilities

The table below describes the three aspects to roles and responsibilities of all personnel that are typically required to be involved with testing for Projects / Changes. The following defines the three aspects:

Standard Testing Role

- This is an industry standard testing role that a member of staff will directly or indirectly inherit as part of the Project / Change.
- Note: typically roles and responsibilities should be agreed at the very start of a project and documented. It is recommended a 'Test Initiation Meeting' is held by the Project Manager / Team Leader to kick off discussions about planning testing

UOL Mapped Role

- Given the statement above, this describes a role which currently exists in The University of Liverpool which will typically take on the relevant Testing Role (s) for that project.
- Whilst not exhaustive this is used to illustrate the type of roles in The University of Liverpool that may map to a Standard Testing Role for a project.
- It may well be that **one UOL person** may undertake **multiple Testing Roles** and therefore assigning 'real life' roles to testing roles is **not a 1-2-1 mutually exclusive scenario.**
    - *An example of a Test Lead for System Testing could be an Application Development Services Subject Matter Expert (SME)*
    - *This person would intimately know what is being delivered and is familiar with the project therefore is best placed to co-ordinate the testing activities.*
    - *Using the above example, on smaller projects the SME may also perform test execution therefore they would be taking on the testing role of "Test Analyst" also.*

Responsibilities

- Key testing tasks and activities that the testing role incumbent would perform.

Standard Testing Role	UOL Mapped Role	Responsibilities
Testing Manager	CSD Test Manager CSD Test Lead Project Manager	• To provide guidance to the test leads when engaged , providing a framework, approach and processes the test lead requires • Monitoring of ongoing projects and milestones, proving input and approval at quality gates. • Assisting in creation and production of test plans • Providing ongoing testing related training
Test Lead	**Unit & System Testing** - SME / Team Leader **User Acceptance Testing :** Senior Business Partner - representatives	• Day to Day management of test activities required for that test phase / project • Recruitment of test analysts to support the design and execution stages of that test phase. • Responsible for ensuring FTP (Test Process) deliverables (e.g. test scripts) are produced within the agreed project timescales • Regular reporting of progress during all stages of the FTP (Test Process) including Planning, Design, Execution and Closure
Test Analyst	One of: *(where deemed appropriate)* • Developer • UX team member • Business Analyst • University Staff • Students	• Designing Test Scenarios and/or Test Cases as part of the Test Design Phase • Identifying Test Data required to support the tests during execution and working with support teams to produce the data • Execution of the tests and providing results to the test lead • Raising any defects found during execution
Defect Manager	• Project Manager • SME / Team Leader • CSD Test Lead	• Responsible for the tracking of defects ensuring… o Produce key metrics on open and closed defects o Arranging triage meetings between developers and testers when required to assist in resolving high priority defects. o Obtaining approval for deferral of any open defects found to exist prior to 'Go-Live'

11. Environment requirements

All testing whether it is at Unit or User Acceptance Testing level is required to be conducted on test environments (non-live). Testing should also be undertaken on an environment that mimics the set-up/configuration of the live system as closely as possible.

At the outset of every Project / Change, considerations need to be given to the creation for a test environment in order to validate the delivery in a non-production environment. If there are any issues setting up a robust test environment then then the following must occur at the earliest opportunity:

- A Issue is raised on the relevant RAID log stating the lack of test environment
- A hard incoming Dependency should be logged on the RAID log for that project stating that testing (unless approval is gained from Director Level) cannot commence.
- In the absence of any formal Risk Management logs mentioned above please escalate the issue your Senior Manager and CSD Testing Manager.

If a test environment does exist then it is recommended the following health checks are carried out before testing begins and also on a 6 monthly basis:

- Quality / validity of test data on the environment.
    - *Is a refresh required from the live system to keep the test bed and environment up to date?*
- Pipe-cleaning tests.
    - *These types of checks effectively check the environment is still operational and can still successfully connect to other test systems (and can therefore pull / push data) as expected.*

Each project / system may have lower level environmental requirements such as required operating systems and service packs, browsers, devices, database versions etc. These specific requirements should be described in the relevant test level plan / approach document.

12. Testing risks and mitigation

Any risks that will affect the testing process during a Project / Change should be listed along with the mitigation. By documenting a risk, its occurrence can be anticipated well ahead of time and therefore proactive action may be taken to prevent it from occurring, or to mitigate its damage.

An example of a risk to testing could be that a particular testing tool required for that project may not being available. Any such risks to testing should be documented in the Test Plan for that test level / project

13. Test schedule

All Projects / Changes planning for testing should make an estimation of how long it will take to complete the testing phase. The estimations should be captured in the Test Plan and form the Test Schedule for the specific test level. There are many requirements to complete testing phases:

- Firstly, testers have to execute all test cases (typically covering new functionality and regression testing) at least once.
- Furthermore, if a defect is found, the developers will need to fix the problem. The testers should then re-test the failed test case until it is functioning correctly.
- Last but not the least, the tester needs to conduct further regression testing towards the end of the testing cycle to make sure the other parts of the software have not been

compromised whilst fixing a specific area. This can occur on test cases that were previously functioning properly.

If possible, assign test cases to each tester before test execution begins. This will not only give the tester(s) a view of what they will be testing, but will also provide a view to the test lead who will be testing what functions, requirements etc. and when.

When creating a test schedule, test representatives should take into account the extra time needed to accommodate contingent issues.

- One way to make this approximation is to look at the time needed by the previous releases of the software (or similar projects)
- If the software is new, multiplying the initial testing schedule approximation **by two is a good way to start.**

14. Test priorities

As part of a risk based approach to testing, when designing test cases it is important to assign priorities for each test or group of tests. When testing is required on projects, certain test cases will be treated as the most critical tests and if they fail, the product cannot be released. Some other test cases may be deemed less critical if they were to fail. As part of the Fundamental Test Process, the test design activity will involve creation of test cases that will form 3 test methods:

- Verify the new functionality / changes being introduced (verification testing)
  - *Known as testing the written requirements*
- Attempt to find any unexpected behaviours in functionality / changes being introduced (defect discovery testing)
  - *Known as testing the unwritten requirements*
- Verify that existing functionality continues to behave and perform as previously (regression testing)

For every test that is created each test should have a priority weighting attached to it:

- *High* – Any tests deemed to be in this category mean they must be run initially as the functionality being tested is critical and if failed would have a severe impact
- *Medium* - Any tests deemed to be in this category mean they must be run only after the *High* category tests. The functionality being tested is important and if failed would have a high but not severe impact
- *Low* - Any tests deemed to be in this category mean they must be run only after the *Medium category* tests. These are effectively 'nice to have' tests. The functionality being tested is not critical important and if failed would only impact a small volume of users whilst at the same

It is worth mentioning that the weightings above must, where possible, be agreed with key stakeholders, especially the stakeholder who has requested the Project / Change.
Whilst this may seem like overkill, this is introducing a risk based view to test coverage and if the time allocated for testing is reduced the high priorities areas would be able to be covered first.

15. Regression testing

Regression testing of existing functions is an undervalued part of verifying whether a system is fit for purpose. The purpose of regression testing is to verify that one fix or change does inadvertently impact other areas in that program or in any other interface.

When considering any form of testing it is strongly recommended that regression testing and what requires regression testing is always considered when planning the scope of the test level. It is the intention of these guidelines to put regression testing at the heart of the testing design and execution process.

The following statements describe the approach to regression testing that should be undertaken:

- Regression testing of a <u>brand new system</u> is not possible initially. However once a new system has a defect and associated fix applied, a set of focussed regression tests should be performed not only on the failed area but also areas surrounding that function.
- If the Project / Change is <u>amending or modifying an existing system</u> then a high degree of regression testing should be performed.
- Regression testing should not be the last element of any testing that is required. If the requirement for regression is high, then as well as initially focussing on the new functionality during test execution, a focussed set of regression tests should be planned.
- Where possible, a set of regression tests should be designed and maintained for <u>each mission critical system.</u> As per the previous *Test Priorities* section the regression tests should each be prioritised using the following weightings:
    - ***High*** *– Any regression tests assigned to this category are required to be executed as early on in the testing window as possible*
        - The functionality being tested is critical and if it does not perform as required would have a severe impact.
    - ***Medium*** *- Any regression tests assigned to this category are required to be executed only after the High category tests.*
        - The functionality being tested is important and if failed would have a high but not severe impact
    - ***Low*** *- Any regression tests assigned to this category mean they must be ran only after the Medium category tests.*
        - These are effectively ‘nice to have’ tests.
        - The functionality being tested is not critical important and if failed would only impact a small volume of users

16. Evaluating test results

When a tester records whether a test has successfully passed or has failed, the following basic testing design concept always applies and therefore is a <u>critical element when designing any tests:</u>

- A test can only be deemed as Passed if the Expected Result matches the Actual Result.

There are two methods which can be used to measure the success of a test:

- **Pass/ Fail method**
  - *Used as the standard measure of success of an individual functional test.*
  - *The number of specific failed tests at the end of each test phase can be used to determine whether the specific quality gate can be passed.*
    - For example there may be criteria that have been agreed prior to the start of test that an acceptable testing success rate would be 97%.
    - In the instance of 96% of tests being passed a decision would need to be made by the Project / Change stakeholders whether this is acceptable or whether the Project / Change cannot proceed
    - Note: The actual metric to determine the % of acceptable failures / required completion % should l be defined in the relevant Test Plan
- **Metric method**
  - *For <u>non-functional</u> tests such as performance of a product, the load / stress levels of a product, the associated tests will be deemed a success by way of the metrics they produce and subsequent acceptance or rejection of these outputs by key stakeholders*
    - *i.* An example of this is “The screen should load in 5 seconds”. If the screen took 8 seconds then rather than failing the test and potentially endangering the project, this result should be used to generate discussion on whether metric produced is acceptable.

It is worth noting here that measuring of quality of a system / product should be a continuous process. Following the transition Project / Change to live, it is suggested that periodic monitoring of incidents raised via the CSD Service desk should take place to review the volume and nature of incidents

17. Test status and reporting

Following the successful completion of agreed test cases the test lead, test manager and project manager must know exactly where the Project / Change stands in terms of testing activities.

Recording of test results should be completed by using the ‘CSD Test Execution Inventory’ which is an Excel based tool. The template can be obtained from CSD Testing Services and can be used by University Staff / Students who are taking on the ‘test analyst’ role (see *Roles and Responsibilities* section). Each test should be given one of the following statuses:

- Not Started - The test has not started and should be the default status for planned tests at the very start of test execution.
- Passed - The expected result defined in the test step **matches the actual result** experienced by the tester
- Failed - The expect result defined in the test step does **not match the actual result** experienced by the tester
  - *It is important to state that even though the failure of the test step may be suspected at that moment in time (e.g. a suspicion a requirement has been changed), whilst the root cause of the unexpected behaviour is still unknown the test step / test must be marked as failed.*

By using the 'CSD Test Execution Inventory' this will allow testers to send their results to the nominated test lead.

It is the responsibility of the test lead (with the assistance of the testing manager) for that test phase to communicate the test execution summary on a regular basis throughout the project. Details on how regular and what test metrics will be reported should be described in the lower level test plan as different Projects / Changes may require differing levels of reporting.

Key metrics that should be reported include:

- Volume of tests that have been executed that day / week / custom period;
- Volume of tests successfully passed;
- Volume of tests failed;
- Volume of tests that could not be started or completed i.e. blocked by an existing issue;
- Details of defects discovered including:
  - *Volume of defects found;*
  - *Volume of defects closed;*
  - *Volume of defects currently open;*
  - *Defect should be categorised by severity (High Impact, Medium Impact, Low Impact) and types of defects (hardware, code, configuration, requirements).*

As at November 2014 all testing to be undertaken will be manually recorded using templates provided by the CSD Testing Service Team and named accordingly to the Project / Change. However as part of the ongoing strategy of introducing and maturing the testing process, steps will be taken to look into the feasibility of recording test results and defects in a testing tool such as JIRA.

18. Requirements traceability matrix

The Project / Change being delivered must completely satisfy the set of client / business requirements. From the initial design stage, each requirement should be addressed by every single deliverable produced by the development process.

The deliverables that cover the requirements include but are not strictly limited to:

- Functional design documents;
- Technical design documents;

- Source code;
- Unit test cases;
- System test cases;
- User Acceptance test cases.

It is recommended for each project where defined requirements exists that a mapping is produced to show how tests trace back to business and technical requirements and vice versa. To help achieve this view (known as *Requirements Traceability*) CSD Testing Services have designed a template entitled 'Requirements Traceability Matrix'. The requirements traceability matrix document is structured as follows

- Each horizontal row will represent an individual requirement;
- Each vertical column represents a document / test level (unit, system, UAT) in scope for that particular project;
- Intersecting cells are marked when a document or Test Level addresses a particular requirement;
- Ideally if every requirement is addressed in every single document / test level, all the individual cells are marked as populated.

19. Evaluation of testing and test process improvement

This section suggests activities to be undertaken to evaluate current testing practices and to ensure the testing process, associated test procedures and guidelines are regularly reviewed, refined and continually improved.

By identifying areas for improvement and measuring results from the testing effort this will allow testing to be conducted in the most efficient way possible at that point in time. The points below can be undertaken by any member of a testing project however CSD Testing Services will be reviewing any outcomes from the tasks and assist with implementation of any improvements.

- Project Lesson Learned workshops
  - *Ensure there is a testing representative involved in any planned overall project lessons learnt meetings.*
  - *Identify and review any areas of improvement that have been raised during the test process (test planning, test design, test execution and test closure)*
  - *Raising any recommendations to the wider project that may fall outside the testing subject but that as a whole may improve the delivery of future Project / Change delivery.*
- Testing Lessons Learned workshops
  - *Arrange lessons learned workshop focussing solely on testing. This should take place at the end of a Project / Change delivery.*
  - *The outputs should be shared and reviewed by the CSD Testing Services and where appropriate:*
    - Test Process and low levels procedures will be updated with the actions arising from the workshop

        - Non-testing relating actions will be communicated to the relevant Project Manager / Team Leader for future projects

- Regular meetings held with all key testing stakeholders to agree areas for improvement
- Post "Go-Live" support requests / incidents monitoring:
    - *Monitoring of volume and types of incidents to CSD Service desk in first 3 months following transition to live (trend analysis)*
    - *The purpose of this is to identify incidents which could have been prevented during the development and testing effort and to refine future procedures accordingly*
- Quality assurance of each test stage:
    - *Monitoring of volumes of defects found during early test phases (e.g. System Testing) vs later phases (User Acceptance testing)*
        - The key objective here being to identify any defects found in UAT that should have been discovered earlier, and putting in place mechanism to reduce future occurrences of such defects being found at this late stage.
- Coverage of business requirements in testing (spanning all phases)
    - *100 % coverage of agreed base lined business requirements across the whole testing process unless there is a specific reason why a requirement cannot be tested.*
    - *Please refer to Requirements traceability section for further information*
- Testing Maturity Level 2 to be attained by end of 2016
    - *Please Appendix B – Testing Maturity Model for details on the testing maturity model*

20. Defect management

Defect management is the term given to the processing and management of any defects found during any part of the testing lifecycle. A defect can be discovered during initial reviews of requirement documentation or observed during the test execution phase (typically when the system has been built and is under test).

The way in which defects are handled is arguably equally as important as running the tests themselves. The detailed approach of how such defects should be handled and processed is documented in *Computing Services – Defect Management Procedure.*

The basic process of defect management is described below and should be followed whenever there is a test phase being undertaken.

- Step 1: Defect observed
    - *Tester observes a difference between the expected result defined in the test and the actual result demonstrated by the system*
        - Tester needs to ensure the requirements have not changed. If they have then this means the test is factually incorrect and requires to be updated. **No defect required if this is the case**

        - Tester also needs to check if the defect has been previously reported by another tester or if the developer is aware if this. **No defect required if this is the case** however the existing defect requires to be updated explaining there is a further test that is now being affected

- Step 2: Defect logged

    - *If the actual result has not already been reported and is not connected to a requirements change i.e. the system is doing something it shouldn't be or vice versa, **then a defect needs to be logged** using the agreed defect management tracking system*
    - *Please refer to the Computing Services – Defect Management Procedure for details on the system which the defect needs to be logged. Alternatively please contact CSD Testing services using e-mail address: CSDTestingServices@liverpool.ac.uk*

- Step 3: Defect assigned for triage

    - *Developer **reviews the defect** and investigates the Root Cause*
    - *Developer **details Root Cause** in the defect and if applicable releases the fix*

- Step 4: Defect assigned for re-test

    - *The Defect is now assigned back to the tester to re-test. The same test that failed during Step 1 of this process would be executed again. The tester would then either:*

        - **Close the defect** after successfully re-testing, ensuring comments are added stating the test was successful. The test status would now also be updated to state: Passed
        - **Re-assign the defect** back to the development team for further investigation if the re-test was not successful. The test status would continue to have the status of : Failed

Appendix A – Glossary of terms

Term	Definition
Acceptance Criteria	The exit criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorised entity
Actual Result	The behaviour produced/observed by the tester when a component or system is tested.
Defect	Where the system under test behaves differently when compared to a pre-defined expected behaviour in a test case or test scenario.
Defect Management	The process of recognising, investigating, taking action and disposing of Defects. It involves recording defects, classifying them and identifying the impact.
Defect Management Tool	A tool that facilitates the recording and status tracking of defects and changes.
Deliverable	Any product that must be delivered to someone other than the product's author. An example of a deliverable produced by testing is a Test Plan.
Entry Criteria	A set of criteria which has been agreed upon for the test phase. The criteria must be met before the test phase can commence. One example of a particular criteria for User Acceptance Testing would be "Test Scripts for UAT have been signed off by key stakeholders"
Exit Criteria	A set of criteria which has been agreed upon for the test phase. These criteria must be met before the test phase can officially close. One example of a particular exit criterion for User Acceptance Testing would be "There are no high severity defects currently open".
Expected Result	The behaviour predicted by the test scenario, test step of the component or system under specified conditions.
Load Testing	A type of performance test conducted to evaluate the behaviour of a component or system with increasing load, e.g. numbers of parallel users and/or numbers of transactions, to determine what acceptable load can be handled by the component or system
Performance Testing	The process of testing to determine the performance of a software or Product against initial baselines agreed in advance of the test.
Quality Gate	A special milestone in a project. Quality gates are located between those phases of a project strongly depending on the outcome of a previous phase. A quality gate includes a formal check of the documents of the previous phase
Regression Testing	Testing of a previously tested function following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is usually performed when the software or its environment is changed
Requirements traceability matrix	A two-dimensional table, which correlates requirements and test cases). The table allows tracing back and forth the links of one entity to the other, thus enabling the determination of coverage achieved and the assessment of impact of proposed changes
Risk	A factor that could result in future negative consequences; usually expressed as impact and likelihood.
Stress Testing	A type of performance test conducted to evaluate a system or component at or beyond the limits of its anticipated or specified workloads, or with reduced availability of resources such as access to memory or servers
System Integration Testing (SIT)	A test level which the primary objective is to Test the integration of systems and application and testing interfaces to external systems
System Testing	A test level which the primary objective is to detect defects in end-to-end scenarios concentrating on systematically validating whether each function performs as expected or not
Technical Security Testing	Testing to determine the security of the software product
Test Case	A collection of test steps (1,2,3,4 etc.) which go into low level detail instructing the tester what they specially need to do when running the test. Every step must have an expected result
Test Data	Data that exists (for example, in a database) before a test is executed, and that affects or is affected by the component or system under test. A set of test cases created by testers should define a core set of test data requirements required to be made available in order for the test to be executed.
Test Design	Part of The Fundamental Testing Process. This stage involves reviewing what needs to be tested and defining the tests in the form of deliverables such as test scenarios, test cases, test steps, test data requirements.

Test Execution	The process of running a test on the component or system under test, and producing actual result(s).
Test Initiation Meeting	This is typically held during the Test Planning phase and is usually led by the Test Lead. The following people would be invited: Developers, Business Analysts, Project Manager / Team Leader, UAT representatives. The intention is to discuss the key aspects of the Test Plan and agree role and responsibilities for all testing activities.
Test Level	There are generally four recognised levels of tests: unit testing, integration testing, system testing, and acceptance testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.
Test Plan	A document produced during the Test Planning phase, typically by the test lead for the test phase. This documents how testing will be conducted, who by, what the testing will cover
Test Process	The fundamental test process comprises test planning, test design, test execution, evaluating exit criteria and reporting, and test closure activities.
Test Process Improvement	A series of activities designed to improve the performance and maturity of the test processes
Test Progress Report	A document summarising the status of the current in-progress testing activities and results. Produced at regular intervals to report progress of testing activities against a baseline (such as the original test plan) and to communicate any risks and alternatives
Test Result	The consequence/outcome of the execution of a test. It includes outputs to screens, changes to data, reports.
Test Scenario	A high level statement, kept very brief describing a function / process / feature to be tested e.g.: "To test that a user can logon to TULIP and submit a holiday request for next week"
Testing Maturity	A framework for test process improvement that describes the key elements of an effective test process
Testware	Artefacts produced during the test process required to plan, design, and execute tests, such as documentation, scripts, inputs, expected results.
User Acceptance Testing (UAT)	Testing with respect to user needs, requirements, and business processes. Conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorised entity to determine whether or not to accept the system

Appendix B – Testing maturity model

Maturity Level	Description	Focus
1. Initial	• No standard processes. • Ad hoc approaches, disjointed, manual. • Evidence that issues are recognised and need to be addressed. • Reactive (firefighting) • Best efforts, relies on individuals • Some limited documentation or no documentation • User driven ('who shouts loudest')	Avoid downtime
2. Repeatable	• Stable repeatable procedures • Staff have some awareness of processes • Co-ordinated manual process • High reliance on individual knowledge and skill level (knowledge silos) • Largely reactive with some limited planning • Little or no formal training or communication	Get control
3. Defined	• Stable processes documented and standardised • Full adherence to procedures by staff • Knowledge transfer and training completed • Some performance monitoring • Some integration with other processes • Request driven	Adopt standards and best practice
4. Managed	• Processes fully adopted and embedded in work structure • Processes measured, controlled and reported • Proactive and accountable • Service driven	Efficiency
5. Optimized	• Processes are best practice and have continual improvement • Full integration with other processes • Increased productivity • Proactive and continuous • Quantitative feedback of process	Enable Innovation

Appendix C – Testing templates

To assist in making the testing process as efficient as possible, Testing Services provides a selection of templates used for the key elements of the testing process. This section describes examples of templates which you may choose to use for own test project. Please contact CSD Testing Services directly who can provide them to you.

Test planning templates

Test plan - template

- ➢ A document produced during the Test Planning phase, typically by the test lead.
- ➢ This documents how testing will be conducted, who by, what the testing will cover.

Test design templates

Test scenario inventory - template

- ➢ An inventory of high level statements, kept very brief describing a function / process / feature to be tested e.g.:
    - o *"To test that a user can logon to TULIP and submit a holiday request for next week".*

Test case inventory - template

- ➢ An inventory of test cases which includes a number of test steps (1,2,3,4 etc.) for each test. The test cases should go into low level detail instructing the tester what they specially need to do when running the test.
- ➢ Every step must have an expected result e.g.:
    - o *"Step 1 – Logon to system ABC"*
    - o *"Step 2 – Enter the word text and click go"*
    - o *"Step 3 – Click cancel"*

Test execution templates

Test progress reporting - template

- ➢ Used for reporting on progress during Test Execution phases (when the tests are actually performed).
- ➢ Depending on the size and nature of the testing activity this can be issued daily, weekly or any other intervals you choose.
- ➢ The audience is typically the people who have most interest in the testing e.g. development managers, departmental managers etc. and is used to produce a view of metrics such as volumes of tests passed, failed, functionality covered so far etc.

Test closure templates

Test closure report - template

- ➢ This is used when the planned testing is completed and is subsequently issued to project stakeholders to summarise the testing that has taken place.

- This describes all the testing that has taken place, any defects that were uncovered and the current status of any open defects.
- The main objective of this document is to provide the project a view of test results which can then be used to inform decisions on whether to proceed or not

End of document