Introduction

For the evolutionary development of our comprehensive solution to be successful it is important that we have a robust and achievable testing strategy to outline our testing timescales, the methods of testing defined, the types of tests defined, our objectives for this testing and some idea of the necessary resource.

At the high level we know that there are a number of processes that we need to be able to achieve for example we know that we need to be able to pay people, we need to be able to hire new members of staff, we know that we need to have a means of paying our suppliers and to invoice other organisations. The testing strategy seeks to bring some clarity around the approach for our testing, what we want to achieve and why. It aims to inform all relevant parties to ensure a common understanding of the scope of testing, what, how, when, why, who to involve and the likely issues and risks involved.

This will be a living document as the granular levels for individual tests will not be understood until our precise specification and requirements are defined.

Categories of test/objectives/risks

Unit testing – This will occur throughout the build of each module and can be seen as testing of the smallest testable aspects of the system (be that a piece of code, the functionality of a field, a single operational procedure etc) to ensure these are fit for purpose. This is the first element of
testing and arguably the most important as if we get this right we put ourselves in a far stronger position for the testing that follows.

**Risks;** missing something small that will have an impact further down the line, necessary change requests as a result of incorrect testing, testing in isolation resulting in issues for the solution once combined.

**Migration testing** – UBW provides us with a number of means for importing/migrating data from our legacy systems and into the new solution such as file upload, web services, using code etc. Some of our data will remain as is and other data will require mapping or data conversion. The migration test seeks to confirm that our data has been successfully migrated into the new solution and tests may take a variety of forms including visual spot checks and programmatic checking for automated comparison. We will also need to test volume as well as quality of migration as performance and also quality may be affected when a bulk data load is migrated.

**Risks;** duplicate data, special characters and null values causing errors, data elements missing in the new system, incompatibility of systems (code for example), importing data which has not been cleansed sufficiently, unknown errors which may take time to detect.

**IST (integrated system testing)** – We will have multiple integrations with UBW and these will take different forms. This testing seeks to prove the successful cohesion and coexistence of these systems when integrations have been implemented for the exchange of data. We need to ensure the successful coexistence of the system as ‘a whole’ once the various constituent parts are assembled and ‘talking to each other’. This is a collaborative effort and likely to involve technical resource and also end users seeking to prove that a change in one system has the expected result in the next. We will be cross-checking against business and technical specifications, investigating logs (so requires Unit4 technical involvement), identifying sticking points, correcting mapping etc.

**Risks;** for the project to be considered a success it is vital that we do not achieve reduced integration to that currently enjoyed by the College, timeliness of the exchange of data – i.e. is it an error or a delay, data volumes causing unexpected error, the technical lack of understanding in house of the new solution.

**Parallel Payroll testing** – This is a particularly critical aspect of testing and requires a large amount of time, resource and effort to achieve. Our aim will be to prove that the payroll calculations are being accurately calculated in the new system over an iterative testing process which seeks to identify and correct errors early. We shall be making comparisons with calculations in our legacy system with those made in UBW to ascertain that the calculations are correct and that there are no inconsistencies. As part of this testing we will run a range of specific payment adjustments such as increments, deductions for pensions etc to ensure that changes are accurately reflected in the new systems.

The success or failure of this testing should determine whether we can go-live or must delay release of the new system. All relevant parties must have high confidence levels by the close of the parallel payroll test cycle.

- **Payroll Parallel Run Month 1 (Testing Migration)** - In December – migrate November (post-payroll) into UBW from Alta HR. Crank lever for Payroll and identify/rectify errors
- **Payroll Parallel Run Month 2 (Testing Payroll Engine)** - In January – migrate December (post-payroll) into UBW from Alta HR. Again, no transactions. Just crank lever and hope for fewer errors.
- **Payroll Parallel Run Month 3 (Testing Payroll Engine)** – In early February – migrate from January (post pay) into UBW from Alta HR and crank again – hopefully no errors now.
- **Payroll Parallel Run Month 4 (Testing Data Input & Payroll Engine)** – Catch up with the live transactions made since the run in UBW to ensure these are the same as live Alta HR and continue then to apply (true parallel) in both systems. Payroll run in both and reconcile.
- **Payroll Parallel Run Month 5 (Testing Data Input & Payroll Engine)** – Catch up with the live transactions made since the February payroll run in UBW to ensure these are the same as live Alta HR and continue then to apply (true parallel) in both systems. Payroll run in both and reconcile.
- **Go/No go decision re Go-Live**

**Risks:** High dependence on successful data migration, volume of data and therefore difficulty spotting errors, importance of timely migration and data input, resource heavy testing, assumption that the legacy system is correct - discrepancy is due to error in the legacy system not in UBW, mistakes made during data input into two separate systems.

---

**Figure 2 - Parallel Payroll Cycle**

**Regression testing** – This form of testing may not be necessary within the implementation of this project but it’s purpose is to confirm that the software that has been previously developed and successfully tested still performs correctly after it has been changed or interfaced with other software.

Changes may include software enhancements, patches, configuration and other alterations and it may be the case that these are necessarily made in other systems within our technical architecture and thus we need to ensure that they have not had an adverse effect on the integration with UBW.
**Risks:** Complexity of integration meaning errors and bugs are difficult to detect, resource and time heavy, being aware that a change has been made to another system.

**UAT (user acceptance testing)** – This is the final phase of our testing (as most issues will have been picked up in unit testing) and is to be completed by the intended audience for each module of the system. This will be completed against provided specification and in the form of a test script and or user story. The UAT testing will include end to end process testing and will require role specific testing as the system, views, access and workflow will be role-driven.

**Risks:** Dissatisfaction from end users regarding process or aesthetic of solution, lack of familiarisation misconstrued as error, incorrect interpretation of Birkbeck requirements, lack of time for change, potential cost or other impact of making a change.

**Format for testing**

The vast majority of UBW testing will take the form of test scripts and test scenarios many of which will be derived from the requirements information and user stories which make up the overall solution design.

Example test script;

As an HR end user

I can indicate that a member(s) of staff should appear on the establishment report

Expected result 1 can tick or untick ‘Establishment’ indicator TRUE/FALSE

Expected result 2 person does appear on establishment report if indicator set TRUE/FALSE

Expected result 3 person does not appear on establishment report if indicator not set TRUE/FALSE

During the process of following these scripts we will hope to drive to the surface any bugs or errors and to validate the development work and confirm that it meets our requirements and is fit for purpose.

For some testing we are likely to utilise automated tests (for example to compare sets of data for integrated systems testing and data migration. This will take the form of coded scripts to be run in or across systems.

**Location for testing**

Testing will primarily take place within the recently renovated EGM1 which has been re-cabled, equipped with new PCs and rearranged to be conducive to testing, training and build work.

**Testing Manager**

The Testing Manager will be responsible for a full understanding of the test schedule and the
coordination of testing, allocation of resources, supervision, some test-script preparation (in conjunction with others), collating of results and for progressing the correction of bugs and errors to the relevant responsible persons or escalating risk to the Project Manager.

Testing Manager - TBC

**Test schedule**

<table>
<thead>
<tr>
<th>Go-Live</th>
<th>When</th>
<th>What</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2017</td>
<td>Early September</td>
<td>Human Resources Unit Testing</td>
<td>HR Build Representative HR End Users</td>
</tr>
<tr>
<td></td>
<td>(Exact dates dependant on build progress)</td>
<td>Payroll Unit Testing</td>
<td>Payroll Build Representative Payroll End Users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Financials Unit Testing (essential for first go-live)</td>
<td>Finance Build Representative Finance End Users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Research Grants Unit Testing (essential for first go-live)</td>
<td>RG Build Representative RG End Users</td>
</tr>
<tr>
<td></td>
<td>October/November</td>
<td>Data Migration Testing</td>
<td>Technical Users</td>
</tr>
<tr>
<td></td>
<td>November</td>
<td>Integrated Systems Testing</td>
<td>Technical Users</td>
</tr>
<tr>
<td></td>
<td>As required</td>
<td>Regression Testing</td>
<td>TBC</td>
</tr>
<tr>
<td>December – March</td>
<td>Parallel Payroll Testing</td>
<td>Payroll Build Representative Payroll End Users</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>User Acceptance Testing</td>
<td>HR, Payroll, Finance, Research Grants End Users</td>
</tr>
<tr>
<td>August 2017</td>
<td>January-July</td>
<td>Financials Unit Testing</td>
<td>Finance Build Representative Finance End Users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Research Grants Unit Testing</td>
<td>RG Build Representative RG End Users</td>
</tr>
<tr>
<td></td>
<td>??</td>
<td>Data Migration Testing</td>
<td>Technical Users</td>
</tr>
<tr>
<td></td>
<td>??</td>
<td>Integrated Systems Testing</td>
<td>Technical Users</td>
</tr>
<tr>
<td></td>
<td>As required</td>
<td>Regression Testing</td>
<td>TBC</td>
</tr>
<tr>
<td>May-July</td>
<td>User Acceptance Testing</td>
<td>HR, Payroll, Finance, Research Grants End Users</td>
<td></td>
</tr>
</tbody>
</table>