# User Acceptance Testing (UAT)

## Testing Aims

For UAT testing (also known as beta testing) you are attempting to achieve 2 things:

### Functional Testing

To determine if all of the known issues and change requests addressed by a new release of the tool have indeed been fixed or the functionality enhanced in the way they were proposed. If the issue/change has been applied but not in the way proposed that is not necessarily a problem but it should still be reported. These tests should focus specifically on the functionality relating to each known issue or change request so you need to be familiar with what the issue/change is and how it is proposed to be resolved/applied.

### Business Testing

To determine if all of the business functions/features of the tool still work as they should/did. In such a complex tool it is possible that changes to the code to fix a known issue or implement a new feature may cause unwanted behaviour in other areas. The best way to determine this is to perform a series of repeatable tests that use as many of the features of the tool as possible to establish that they still behave as expected.

## Test Planning

### Test Plan

When you receive and have installed a new release of the tool, rather than jumping straight in trying a few things out it is a really good idea to write a test plan outlining what you are going to test in this release of the tool. Some aspects of the test plan will be common to all releases of the tool, others will be specific this this release.

### Test Scripts

The test plan should contain a set of test scripts that you or a colleague can follow. Using test scripts will help focus your efforts and can be repeated if required for future releases. Each test script should focus on one aspect of the tool's functionality (e.g. splitting polygons, applying attributes, selecting attributes, etc) and will contain a series of test cases (steps) that will be followed in execute a specific function/process.

### Test Cases

When designing test scripts you need to clearly document what function/process you are trying test and what steps (test cases) you will follow to achieve it. Try and mix up the test cases so that sometimes you use the same data record all the way though a typical process (e.g. physical split, then logical split, then attribute update) - this may highlight where one step causes issues in a later step. In other cases use different records for each test case to ensure that the tool will works with existing data.

## Test Data

### Selecting Test Data

To save time when testing try and select a geographic area in GIS that represents a good selection of data, for example, an area containing a small town or village with surrounding arable/farm land, woodland and 'richer' habitats (likely to contain some BAP priority habitat). This will give you a good variety of data and will save you needing to move around in GIS too much.

You will most likely need to search for records in this area than meet the needs of your test cases (e.g. an incid with a single polygon from a single toid, an incid with a single polygon from a toid that also has fragments belong to other incids, an incid with multiple polygons, etc). Some of the time you can find this test data by selecting polygons in GIS and querying them with the tool. In other cases you may need to run a query in GIS or the SQL Server/Access database to find specific data records.

You will need to identify records that are suitable for each test case. For example, if you are aiming to test how the tool splits a single polygon with toid\_fragment\_id = '00001' into 2 fragments when there is already another polygon with toid\_fragment\_id = '00002' for that toid (so that one of the new fragments becomes toid\_fragment\_id = '00003') then you will need to identify records that meet this criteria. Try and identify several suitable records for each test case so that you can repeat the test on a different record if you experience a problem or unexpected behaviour on the first record.

### Backups and Restores

Take a backup copy of your 'live' data or a set of data that you know to be of good quality. You will need to backup both the SQL Server (or Access) database and the GIS layer(s).

If you can, the best approach when it comes to testing each new release of the tool is to restore from the same backup before you start testing so that you are always starting with the same known, stable point and you can use the same records (incids, toids and toid\_fragments) each time you run your test scripts.

If you experience any errors during testing it might be a good idea to restore the data before you continue testing to ensure that any potential corruption in the data caused by an error doesn't affect subsequent testing.

## Issues and Issue Logging

If you experience an issue try and select a different set of records and repeat the test to see if the problem re-occurs. You may need to check your test data to ensure that it was not corrupt before you ran the test, or to see if it has been corrupted during the test (and hence may need to be restored).

Before logging an issue check if it has already been identified as a known issue on GitHub. If your experience of the issue is slightly different (e.g. using a different GIS application or version) then add a comment regarding this to the issue on GitHub (this may help narrow down the causes of some issues). If you have more information to explain how an existing known issue occurred (sometimes they can be unpredictable), or an idea of how to resolve it, then please add a comment on GitHub.

If you discover that an existing known issue addressed in the latest release that you are test has not been resolved then add a comment to that issue on GitHub with your findings. Similarly, if you find that an enhancement relating a change request (addressed in the latest release) has not been made, or does not work as proposed, then add a comment to that change request on GitHub.

When raising a new issue or adding a comment to an existing issue try and include as much information as possible regarding what you were doing, what applications and versions you are using (including the tool version) and if the issue is repeatable.

## Key Steps

### Phase 1 : Planning the tests

1. Prepare a test plan
2. Prepare the test scripts (or test scenarios)
3. Prepare the tests cases

### Phase 2 : Preparing for the tests

1. Find/prepare the test data
2. Conduct training of unfamiliar functions of the tool
3. Establish the test environment and test data (separate from the live environment)

### Phase 3 : Executing the tests

1. Run the tests (test scripts and test cases)
2. Record the results (passes and failures)
3. Log issues and comments on GutHub