

Test Automation as a Development Requirement

With the latest test automation frameworks and tools it is possible to achieve significant benefits and return-on-investment from automated software testing.

However, on many projects further savings in time and effort could be made if the software design had the requirements for test automation built-in.



Sam Warwick
Director, Odin
Technology Ltd

Automated testers regularly deal with a lot of issues that needn't exist at all if they got together with Development early on in the project. There are often cases where a few very simple changes in the System Under Test (SUT) can make the task of building and maintaining automated test systems much easier. If the development team know about these requirements early then they can often be accommodated. This article will illustrate this by using some typical real-world examples.

Designing for Test Automation

The average software developer usually has very little knowledge of how automated testing tools work and neither should they be expected to. On a typical project developers may make design decisions that can actually make getting the test tools to work effectively much harder. They also typically 'break' the automation with new releases or builds of the software. When confronted by such a situation the typical developer response may be "Oh, I didn't know the tool worked that way, sorry", or "If you had told us that sooner we wouldn't have written it that way. It's too late to change it now."

There are two key points here:

1. If you are planning to automate some or all of your testing then get the automation team involved early. This will help influence the design and educate the developers on the automation requirements.

2. You need to have a good idea of what elements of the software design can help or hinder the automation effort.

Before we look in more detail at why we should influence Development it will help to have a brief overview of how most tools work and some example problems.

The Test Tools in Action

All of the leading functional GUI test automation tools rely on the ability to locate and uniquely identify each testable window and object of the SUT. This process is typically referred to as 'object mapping' and the tool will normally provide a repository for all these objects and their identification properties (the 'object map'). Often the biggest hindrances to most automation projects are either difficulties in identifying the objects in the first place or frequent changes to object properties.



Fig.1 GUI Object Identification

- ① Login:


```
{
  class: window,
  MSW_classhtml_frame
  html_nameLogin
}
```
- ② Login.Username


```
{
  class: edit,
  MSW_classhtml_edit
  html_nameedtUsername
}
```
- ③ Login.SignIn


```
{
  class:push_button
  MSW_classhtml_push_button
  html_name"Sign In"
}
```

Figure 1 shows an example of a typical screen in a GUI application. On the right can be seen some typical sets of properties used by the tool to identify the object at run time.

Example Scenarios

The examples listed below have been made as generic as possible and the points illustrated could easily apply to many software architectures and tools. It is not the intention here to write a complete set of guidelines for one particular technology such as web page automation. This article should give the reader enough ideas to go away and think about this in the context of their own application and environment.

Example 1 Changing Object Identification Property

Problem:

The application has an Edit Box on the Logon screen for entering the 'Username'. The test tool identifies this object by an 'html_name' property of 'edtUsername'. On a later build of the software the developer decides to change 'html_name' to 'txtUserID'.

```

Login.Username
{
  class: edit,
  MSW_class: html_edit,
  html_name : edtUsername
}

```

Built 1

```

Login.Username
{
  class: edit,
  MSW_class: html_edit,
  html_name : txtUserID
}

```

Built 2

Fig. 2 Changes in object property between builds

Impact:

No automated tests that use the Logon screen can run until the test team has updated the 'name' entry for the 'User name' object in the Object Map.

Action:

- Get Development to standardise object naming conventions early to reduce the likelihood of having to change them in a later build.
- Educate Development early regarding the impact of changing object properties.

Example 2 Inadequate Page/Window Identification

Problem:

A web application has 100 pages that all have the title 'Pro Web Portal'. The URL for each page is not guaranteed to be unique either.

Impact:

The tool does not have a reliable and consistent way of recognising each page. An automation technician has to write custom automation code to deal with identifying and synchronising all affected pages. Object Map maintenance is also made more difficult.

Action:

- Suggest unique titles for each page e.g. 'Pro Web Portal Logon', 'Pro Web Portal Home'. Identify this requirement before any web pages are delivered to the test team.

Example 3 Unreliable Object Identification Property

Problem:

The automated tests need to validate the contents of an 'Account Details' table on a web page. The tool cannot find a meaningful property for the table so it uses the 'location' property. This is a numeric index that identifies the nth instance of a table on the page. The page contains many other tables that are used for layout and formatting.

Impact:

Extra tables are constantly being added and removed across builds as the cosmetic appearance of the website is refined. Every time this happens the table 'location' changes and the automation breaks until

the Object Map is updated.

Action:

- Get the developers to add a 'name' property to all table objects that require testing

Early Influence

It can be seen from the examples above that the impact of each of these problems could result in many extra days being spent on the automated test system. However, the proposed solutions would require minimal effort if implemented at the initial development phase of the project. Every situation is different but there are some common messages which should apply to most projects.

- Get input from the test automation team early. Ideally this should involve someone who has a good knowledge of the test tool most likely to be used. They should also have a basic understanding of the software architecture being tested.
- Work with development to produce some brief guidelines for building and maintaining the software to support automated testing frameworks. E.g. standardisation of object identification properties, understanding the effects of changing properties between builds, etc.

It can be seen that a little time spent with development early on will save a lot of unnecessary time and effort in every automated testing cycle. This will result in an even better ROI on automated testing and significant savings and benefits to the project overall ■

sam@odin.co.uk

Subscribe to
Professional Tester Magazine