

Automatic Object Map generation - Teaching a new dog an old trick

By looking back at some old techniques and applying them to today's systems we can shave weeks off the time taken to build an automated test system.

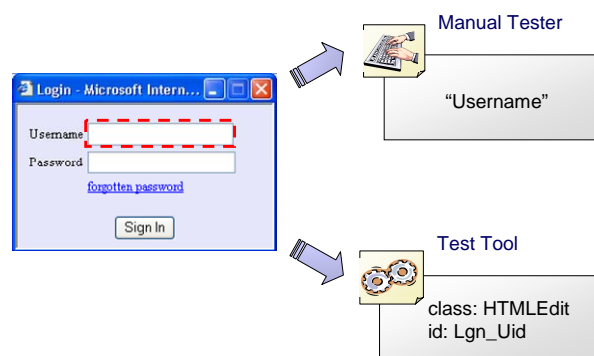
Duncan Briggins
Director, Odin Technology Ltd.

Introduction

Automating the testing of a typical application involves familiar tasks to many of us, defining test data, writing scripts and building object maps. For some modern applications however some of these tasks can be automated too! By borrowing some tricks first used back in the days of green screen applications we can make the path to automation a less painful one. First let us look at what we mean by an object map.

Object maps...de-mystified

Without trying to point out the obvious, testers and automation tools are very different and they see the world in different ways. Most people would refer to the visual elements of an application using simple terms related to the business at hand e.g. "The login window" or the "OK button". Test tools on the other hand use a variety of technical properties to identify and relate to the same things. Fig 1 shows a simple example of the way a tester would describe a UI element and the terms the average test tool would use.



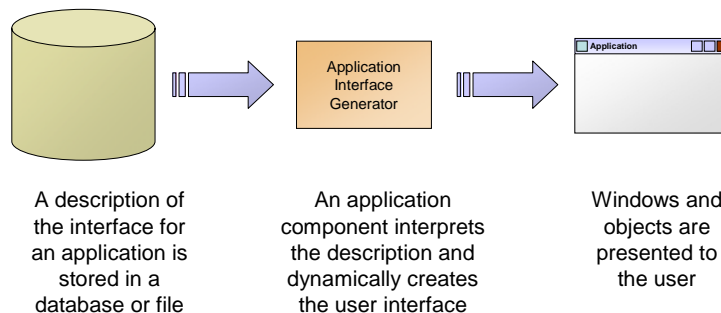
Simply put, an Object Map provides the mapping between the human terms used to describe the elements of an applications interface and the test tools' terms. Most modern test tools provide some kind of object map facility, usually referred to in slightly different terms.

Building an object map can be a laborious process. With the test tool and the application open, a tester would use the mouse to point to an element of the application. The tool would read the properties it requires to refer to the object, then prompting the user to type in a sensible business name for the same object. This would be stored and the user would move to the next object and so on.

Introducing the old trick – self describing applications

Some of us still reminisce about using dumb terminals and how things were simpler back then. Not surprisingly, similar application design to that used in the days of central servers and dumb terminals is back and being used in modern client server applications today. Your average “green screen” user interface was made up from a set of regular sized pages, with fixed rows and columns of text. Descriptions of these pages were held on the server and when required would be sent to the dumb terminal where they would be interpreted and presented in all their monochromatic glory.

Some of today’s bigger enterprise applications are borrowing these techniques, storing descriptions of the windows that make up the interface in a database. The description is interpreted and presented dynamically to the user when required, as shown in Fig.2.



An example of the kind of description stored in the database is shown in Fig. 3.

WINDOW: CUSTOMERDET	
FIELD	CUST_SNAME
TYPE	INPUT
LABEL	Surname
LENGTH	20
SEQ	1
FIELD	CUST_GEN
TYPE	OPTION
LABEL	Gender
LENGTH	1
SEQ	2

More and more applications are using similar ideas with different non-database architectures, opening themselves up through APIs and other self-describing techniques. The advent of Microsoft .NET has taken this to a new level with the availability of a technique called “Reflection” to provide information about an application. This however is a topic for an entirely different article...

Using the old design techniques to our advantage we can make the path to automation a much easier one, let us explore how.

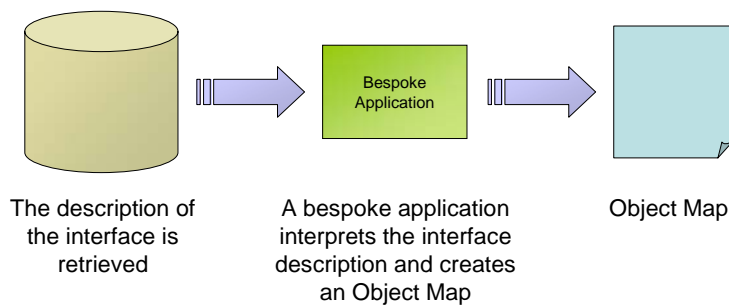
Creating object maps automatically

We can see from the examples above that there are two important pieces of information stored in the application description held in the database:

1. A “LABEL” that your average manual tester could use to describe the application elements i.e. Surname or Gender,
2. The technical properties that the test tool might use to refer to the same element, i.e. “TYPE” and “SEQ”

We nearly have all the information we need to create an Object Map, giving both the human description of the application interface and the tools requirements.

In practice a bespoke application to do the translation would need to be created. Fig 4 shows where this would fit.



As with all automation efforts, an investigation of the time taken to create this bespoke application versus the time taken to manually build the object map would be sensible before diving in and creating it. If it will only take two days to build and save the team two weeks of object mapping time then it is the right thing to do. (Remember to factor in the time taken to navigate to the objects as well!).

Can this work in practice?

This all sounds excellent in theory but will it work in practice? We have used this technique on two automation projects in the last year to vastly improve the time to automation. The figures stand up for themselves.

Project 1 – 1812 Object Map entries created automatically.

By using a conservative figure of a minute to find and map a single object, nearly a weeks worth of solid mapping was completed in less than a day.

Project 2 – 7537 Object Map entries created automatically.

Again using a minute per object, over four weeks solid of object mapping time was saved using the object map generation techniques.

Summary

In many areas of Information Technology, application design techniques are being recycled and re-applied in new and highly effective ways. When embarking upon a new automation effort, it is often worth taking a look back at the past and remembering some of the old tricks, as they may apply to even greater effect today.

duncan@odin.co.uk