

A Comparison Of CRM Testing Techniques For Merthyr Valleys Homes

By Kristian Cole
1810705

Abstract

The Aims of the project is to provide a case study that analyses:

- A testing methodology that can be utilized in an on-premise/Cloud hybrid infrastructure.
- An implementation of a chosen framework that reduces not only time and cost but provides a means of testing on premise before cloud deployment.
 - The research method to come to this conclusion.

The outcome of this project:

- Implementation of testing framework.
- Deliverable of framework to real life development at Merthyr Valley Homes
- Concluding that the replacement of the current code testing process utilizing a framework can reduce time and cost overall.

Introduction



Merthyr Valley Homes is a social housing sector organisation that provides housing to the residents of Merthyr Tydfil who are in need of housing support. The organisation resides within the housing sector and has been operating since 2008 serving the people of Merthyr Tydfil. The organisation currently handles 4500 properties, 4000 properties are a variation of houses/bungalows/flats and 400 are garages and is steadily growing into creating new estates and homes going forward into 2022/23. Each of these departments utilizes the current on-premise software Microsoft Dynamics CRM. Merthyr Valleys Homes needed to change mobile working standards due to the Covid pandemic of 2020 which has forced the use of meeting based software such as Microsoft Teams and further shifts into working from home and mobile working.

The aim of this project is to evaluate various software testing approaches for deploying on-premise software to the cloud as Software As A Service(SaaS):

1.1 Objective 1

Analyse software testing techniques that could be used to deploy an application to the cloud.

1.2 Objective 2

To determine the effectiveness of a new approach to porting an on-premise application to the cloud by analysing the required testing frameworks available on the market that would facilitate the most benefits and reduce overall costs and time.

1.3 Objective 3

To Evaluate the practicality of testing frameworks in the software development working practices at Merthyr Valley Homes.

FAKEXRMEASY

FakeXRMEasy is a Mocking Framework for testing Microsoft Dynamics Plugins in C#. It is designed to work on pre-cloud based on-premise CRM system but as the current CRM 2016 system is codenamed Microsoft Dynamics 365 On Premise It is equal to the Microsoft Dynamics 365 version of CRM.

The plugin structure derives from an interface called Iplugin as referred to in 2.3 and as a limitation when testing as there is no connection to the host system. The connection to the CRM system is provided by the IOrganization service also referred to in 2.3 and as a limitation when testing. Without the IOrganization connection to any of the CRM systems there is little chance of successfully running the code to connect to the database back end.

FakeXRMEasy solves this problem by providing a fake Iplugin interface and IOrganization service to communicate to. The idea of faking in the Microsoft world derives from the need to test code in a program that requires database access and this generally means that the code needs to be executed within a live program to achieve data results. The developer can reference and make connections to any database within scope but would struggle from a code angle to view the data behind the connections. Only within functional testing does the data present itself for view.

XRMPage-Mock

XRMPage-Mock is a mocking framework for the XRM Javascript language.

XRMPage-Mock carries out the task of faking entities within Microsoft Dynamics CRM and allowing a fake connection within the Visual Studio IDE. The test of the framework was to reference to already existing functionality within the live system of Microsoft Dynamics CRM mocking the test that would take place prior to the cloud platform but within an on-premise infrastructure.

The kind of scenarios this fixes:

- Testing XRM Javascript functionality on premise before have to deploy to the cloud and risk the failure of the code in the centre of a DevOps process.
 - Highlighting issues with code referencing entities that may be misspelled, mis referenced or not exist at all.
 - Catching issues early rather than testing within the cloud dev environments

Results

	Microsoft Dynamics On Premise	Microsoft Dynamics 365
Time	JavaScript/C# Plugin Development/Testing time: 30 minutes	JavaScript/C# Plugin Development/Testing time: 30 minutes
Cost Of Time	JavaScript/C# Plugin Development/Testing: £18.66 an hour	JavaScript/C# Plugin Development/Testing: £12.75 every 40 minutes approximately.
Procedure	Debug time: 5 minutes Deployment Time: 5 minutes five Testing Time: 20 Minutes Total: 60 minutes	Debug time: 1 minutes Deployment Time: 0 seconds Testing Time: 10 Minutes Total: 41 minutes
Benefits	Average working day: 7.5 hours Plugins/Scripts developed: 7	Average working day: 7.5 hours Plugins/Scripts developed: 10

Conclusion

Following the case study on XRMPage-Mock and FAKEXRMEasy there are technical advancements in the on-premise testing prior to the inability to obtain full testing control in a cloud environment. The cloud environment certainly has major advantages in terms of being to host code offsite and to have multiple developers working on the same code. This leads to faster development times and with the advent of the Covid 19 pandemic the need to work in silo and remotely now more than ever is a plus in terms of cloud hosting code solutions. The additional benefit to testing with the two aforementioned frameworks provides the developer a way of faking the Dynamics CRM 365 cloud structure on premise and through this brings down the time of testing in a live environment considerably. Where the time is deducted from the overall development time, without the on-premise frameworks the time would still exist from the configuration, deployment and unit testing in the cloud that would have taken place. In faking the connections and the Dynamics CRM 365 backend overall onsite there is a large amount of testing that can be conducted prior to deployment and testing in the cloud.