A Process for Improving Software Testability

Troy Lamoreaux
MediServe Information Systems, Inc.
troyl@mediserve.com

Mark Ofori-Kyei
MediServe Information Systems, Inc.
mofori-kyei@mediserve.com

Mark Pinone
Segue Software, Inc.
mpinone@segue.com

Abstract

The process of software engineering creates a unique problem for testability. MediServe Information Systems has developed a process for creating test harnesses in an automated testing environment. The process is allowing MediServe to build maintainable products that contain hooks for testability at the time of the product's release.

Introduction

MediServe Information Systems develops software products for hospital respiratory, rehabilitation, and emergency departments. The company's MediLinks® SQL based product for respiratory and rehabilitation departments was introduced in 2000. Since its release, MediServe's customers have requested numerous changes to the product.

Problem

In order to ensure the quality of MediServe's products, regression testing is executed for all major software releases. Prior to November of 2003, regression testing was conducted manually. Manual software testing increases product cycle times and development costs. MediServe enhanced its software maintenance capability by purchasing an automated regression testing tool in November, 2003.

During our evaluation of regression testing tools we determined that most mature software products such as MediLinks® are not designed for testability. For example a certain Outlook Bar control only exposed the class name and nothing else. This left the Test Engineer with the option to use x and y coordinates or replace the control with a new one that exposed its methods and properties. Neither option was acceptable.

Certain controls in the applications are not friendly to automation tools. With some controls the only area exposed to automation tools is the class name. There is no exposure to methods or properties. Why are some controls unfriendly to automation? Most automation tools target web and more recent development environments. Mature software products with third party and native controls most likely will not support automation testing out of the box.

Solution

It was imperative for MediServe to develop and implement a process that would increase the testability of its MediLinks® product. A simple eight step process was adopted.

- Determine what is testable
- Determine what objects or classes the automation tool can identify
- Determine what can be manually scripted
- Document non-testable components
- Determine if a re-usable function or method can be created to test a specific control that is used multiple times in the application
- Ensure that the solution will not interfere with the applications functionality
- Develop solution
- Validate solution

To implement the functions developed there needs to be some type of communication between the Application and the Automation testing tool. In the Windows environment this can be implemented via a custom .dll. Some automation testing tools come bundled already with this solution. MediServe currently uses Segue's SilkTest for regression testing. Segue provides a SilkTest Extension kit which we used to implement our process in a more expedient manner.

Conclusion

The team is following this process during the development of new products so that testability can be built into the product during the product development cycle. This allows MediServe to build maintainable products that contain the hooks for testability at the time of the product's release.