**TESTING**

**On**

**Automated Patch Mechanism for MR Software Components**

**SUBMITTED BY**

**PRAVEEN K 1MS12CS075**



**M. S. Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

**Department of Computer Science & Engineering**

***Under the guidance of***

|  |  |  |
| --- | --- | --- |
| **Sriranga Natha**  **System Architect, PIC ADI Magnetic Resonance**  **Philips Innovation Campus, Bangalore** | **&** | **Sini Anna Alex**  **Assistant Professor**  **Department of Computer Science & Engineering** |

* 1. **Introduction**

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product meets the business and technical requirements that guided its design and development.

* 1. **Testing Environment**

The application is tested on MR Environment configured test systems.

* 1. **Test cases**
     1. *Command line arguments check*

Steps:

1. User runs the application by specifying the modified files list and the patch location.
2. The existence of patch directory is checked.
3. If patch directory is present, use it and continue execution of the program.

Exception: Patch directory does not exist.

Solution: A default patch location is used.

* + 1. *Building dlls*

Precondition: The command line arguments are validated.

Steps:

1. The dlls affected by the patch are found.
2. All the found dlls are built.
3. The build log of the dlls is checked to determine if the build was successful.
4. If the build is successful, continue execution of the program.

Exception: Build of one of the dlls failed.

Solution: Print a fatal error message and exit the program.

* + 1. *Stopping and restarting processes*

Precondition: All the patch affected dlls are built successfully and patch directory exists.

Steps:

1. The processes that use the built dlls are found using build metadata.
2. Check if the patch impacted processes are currently running.
3. If the processes are running, stop them and goto step (e).
4. If the processes are not running, goto step (e).
5. Copy the built dlls to the patch directory.
6. Restart the patch impacted processes.

Exception: Restarting of patch impacted processes failed.

Solution: Print a message to indicate to the user that the restart of the process has failed.

* 1. **Defect Distribution in module wise**

1. *Identify and build the dependent DLLs on the developer’s system:*

During unit testing of this module, it was found that the module was not handling the build failure case. So the defect status was changed to Re-open and once fixing was done, the status was changed to Closed.

1. *Identify and stop the dependent processes on the remote test system:*

No defects were found in this module during testing.

1. *Target all the DLLs to their respective locations on the remote test system:*

No defects were found in this module during testing.

1. *Start only the impacted processes:*

During testing, it was found that module does not handle the scenario where the restarting of processes has failed. So, it was reopened for fixing. Later it was fixed and the status was changed to Closed.

It was also found that module does not check if any other processes are affected by stopping a patch impacted process. This issue is not yet solved and so the status is Re-open.

* 1. **Number of defects identified and their status and severity**

|  |  |  |
| --- | --- | --- |
| **DEFECT** | **STATUS** | **SEVERITY** |
| 1. Application does not handle the build failure scenario. | CLOSED | S1 |
| 1. Application does not handle the scenario where the restarting of processes has failed. | CLOSED | S2 |
| 1. Application does not check if any other processes are affected by stopping a patch impacted process. | RE-OPEN | S1 |

**STATUS**

- **Closed**: Indicates that the code change is made and issue has been verified as fixed by developer but tester is yet to test the same and conclude defect is indeed fixed.  
- **Re-open**: Indicates defect fix was retested by tester and found as not fixed and status of the defect is changed to reopen.

**SEVERITY**

* **S1** = Critical
* **S2** = Major
* **S3** = Minor
* **S4** = Trivial
  1. **Types of testing performed**



**Unit Testing-** The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Typically it is done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.

**Unit Testing Techniques:****-**

* **Local data structures:** These are tested to inquiry if the local data within the module is stored properly or not.
* **Independent paths:** All independent paths are tested to see that they are properly executing their task and terminating at the end of the program.
* **Error handling paths:** These are tested to review if errors are handled properly by them or not.

**Integration Testing-** In Integration Testing, many unit tested modules are combined into subsystems, which are then tested. The goal here is to see if the modules can be integrated properly.

**Regression Testing-** Regression Testing is done when some changes are made to an existing system. When modifications are made to an existing system, testing also has to be done to make sure that the modification has not had any undesired side effect of making some of the earlier services faulty.

**Performance Testing-** Performance testing is the process of determining the speed or effectiveness of a software program.  It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

**Performance Testing Technique:**

* **Load testing -**It is the simplest form of testing conducted to understand the behavior of the system under a specific load.