**Literature Survey**

**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**

Philips Medical Systems, a division of Royal Philips Electronics, is the leading supplier of medical imaging equipment and related healthcare services. Magnetic Resonance Imaging (MRI) is one of the youngest and most innovative imaging technologies and is growing in applications but also in complexity of the system. MR has a large software presence in the Philips Innovation Campus, Bangalore, producing state of the art software driving MR workflow, post-processing, reconstruction and serviceability.

Figure 1 shows the complexity involved, both in the MRI system as well as in the organization (Callo Arias, van der Spek, Avgeriou, 2011).



Figure 1: Overview of Philips MRI system size and complexity

The whole MRI system is controlled by a software system that runs in several computers. The software consists of several million lines of code written in nine different programming languages. Also, the software has a long history of being exposed to numerous changes that need to be patched regularly. Patches are additional code to replace logic flaws in existing software.

An essential part of the Philips development framework is its association with Research hospital sites. To facilitate co-development in such sites, MR offers a software patching mechanism that enables the sites to explore and test new pulse sequence techniques. As a continuous improvement, MR is building an intuitive and fast Software patching framework with a long term view of single-click enabling of patches.

The aim of this project is to successfully develop an automated patch mechanism for MR software components. This project concerns itself with the workflow aspect of the complex MR software.

1. **Main Body**

In the MR systems being used currently, the main activity involves identification of the MR applications that need to be restarted to activate a patch. In these systems, if an issue is detected with a particular DLL, the user, after rectifying the issue, needs to manually search all over the remote repository for dependent DLLs, check for the compatibility, and then rebuild the entire system, which usually takes around 6 hours, and then install the system again, which takes some more hours. In addition, all system processes need to be suspended in order to check for the dependent processes (executables). Hence, the manual patching process has a significant impact on business operations, which includes the direct cost of patch management software as well as the indirect costs associated with taking IT staff away from other tasks and downtime.

To resolve the above mentioned problem, an automated patch mechanism is proposed whose function will be to fix the logical issues, automatically search for dependent DLLs and ensure compatibility is maintained, and then patch the fix to the system without the need to build the entire system again, or to stop all system processes, except for the dependent ones. Implementation of this patch mechanism shall aid in improving the efficiency of the system along with a reduction in the user effort.

The most significant benefit of the automated patch mechanism is reduced administrator effort. The automated solution takes over the labor-intensive functions and offers greater efficiency of operations.

1. **Conclusion**

This study highlights the details of the MR system and brings out the benefits of automated patching when compared to manual patching. By implementing an effective automated patch mechanism, the following business benefits can be achieved:

* Reduced downtime and costs associated with non-availability of systems and applications.
* Reduced labor costs.

1. **References**
2. <https://www.open.collab.net/media/pdfs/philips_medical_systems.pdf>
3. <http://www.cs.rug.nl/paris/papers/ESE11.pdf>
4. <https://en.wikipedia.org/wiki/Patch_(computing)>
5. <https://www.novell.com/docrep/documents/c4pzmecjku/automated_patch_management_impressive_return_on_investment.pdf>