*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. VI (Mar – Apr. 2015), PP 41-43*
*www.iosrjournals.org*

# Restoration and Degeneration of the Applications

Rajesh Venkatesan, Ankit Jain and Prateek Arora

**Abstract:** *This study is to analyze and identify solutions for software degeneration lifecycle.*
*A software runs on multiple platforms (Operating Systems) these days and makes use of their resources which communicate directly with the hardware. It is, at the time of installation, in its most critical and vulnerable state because at that moment, it is dealing the system files which were created or configured with latter. Hence, a software is likely and most probable to degenerate at this stage.*
*So, in a mechanism where the application state can be saved during its update and installation would act as a rescue measure if the software degeneration happens during the same.*
*One crossroad we may come across in the discussion is that the framework that deals with saving the application state shall be embedded along with the application or can it be made a cross platform extension or shall it be a facility in the operating system.*
*Also, from a user point of view, Application Installation, an Upgrade or a Critical Software Patch are always vulnerable. This not only involves unexpected functioning of the software but also affects the utilization of resources in the operating system.*
*This seeks for a restoration framework at the application level. Yes, the operating system for its own self does it today using checkpoints however it is not for the softwares running over it. Restoring applications specifically across platforms is required today for clear software installations and upgrades.*

## I. Experiments and Data analysis:

Observation of the software failures leads to discovery of an important fact that the softwares are generally in the high risk of failure during their update/upgrade cycle. For a software, when it is in its useful life as described in the graph below, it undergoes its upgradation many a times and for each single time, there is dramatic increase in its failure rate.

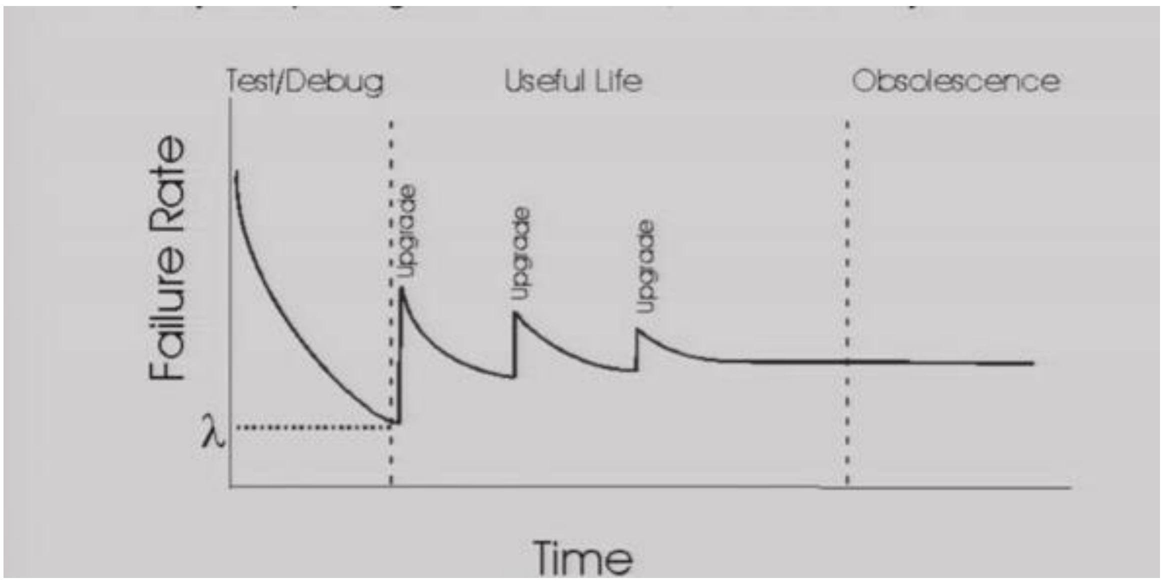

**Fig 1.0**

These failure rates makes the software less immune to unstability and thus risk of degeneration in each upgrade rises as a spike. This graph can help us determine the checkpoints of the restoration that we need to create in a software’s lifecycle and dispose off when not required.

Now, lets have a look at what the idealized software failure rate curve should look like :-

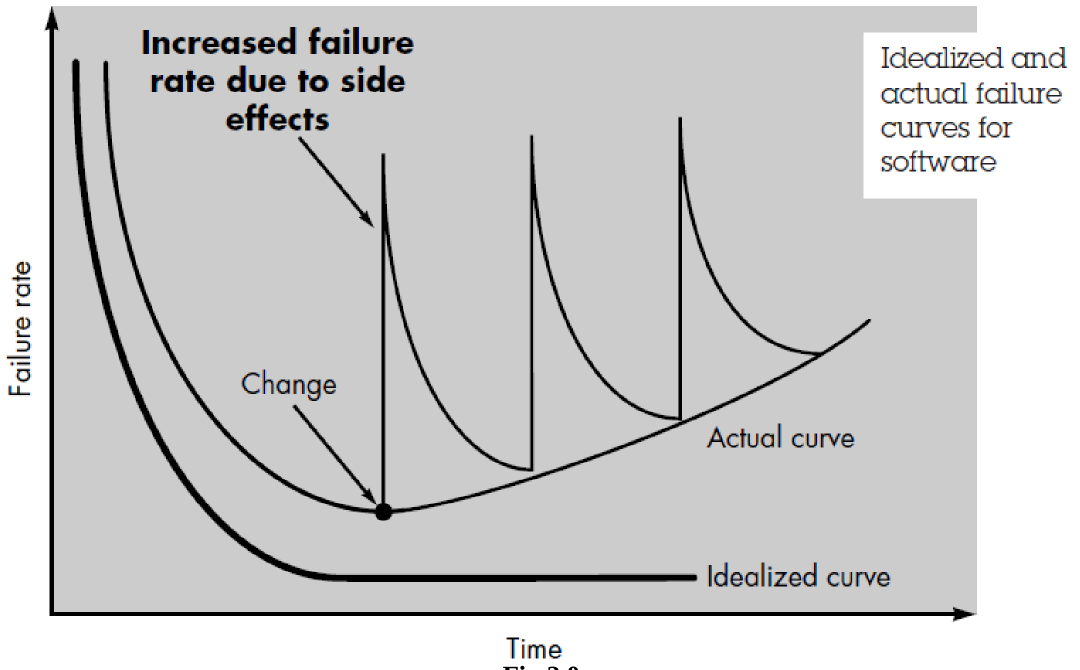

**Fig 2.0**

If we compare the idealized curve to the actual curve, we can easily figure out the drastic increase in failure rates of the software. The software becomes more and more unstable with each upgrade/change instead of becoming more stable. Idealized curve shows a consistent drop in slope of the failure rate, but in reality, on the contrary, upgrades tend to come in each software's lifetime which leads to the creation of spikes depicting unstability in the softwares configuration and lifecycle. Each of these spikes calls for the creation of checkpoint in the underlying system's permanent memory. As a result, we can come closer to the idealized curve, which is beneficial for the software as well as the operating system it is running upon.

## II. Discussions & Results:

The restoration of the application state can take place by the following three methods:
1) OS to handle application restoration.
2) Standalone Cross Platform Agent.
3) Application level restoration.

**OS:-**

For the current research and discussion let us take ".dat" files for consideration as they can appropriately help to understand the whole motive of this research in a nutshell.

The ".dat" files are the ones which provide the information to OS to protect against the pre-existing virus (if any) in it. A minor change in the configuration data in these files can make them act as their counterparts i.e. Virus. This has happened in the past in many of the similar software updates. However, if a healthy state for the application exists, we can make it roll back to that state. For the determination of creation of checkpoint following procedure can be followed:

An application when installs maintains its entries in the file system and registry. If the file system changes go smooth during update, then the registry entries are made. So prior to the file system changes, the current configuration for the application can be saved in the already created file system in some other location in permanent memory. As a result of this, a disturbed update for the application, as soon as it suspected, can be rolled back using the data which was stored prior to the update.

There is a drawback in the inbuilt-OS level restoration framework. It will cyreate restoration framework for all the applications making use of OS resources and user will have no choice to select only a few particular application which he wants to stored. Also, storing status of every application will use a lot of memory in the system. For this purpose, we can have an idea of a standalone restoration agent which overcomes the mentioned drawbacks.

**Application Level restoration:**

Application level restoration deals with the restoration of the application data using the framework associated with the application itself. The application level framework will keep check on the upgradation cycles and maintain the record of the application data by itself. In working, the application level restoration framework will work in similar fashion as OS one.

**Standalone Cross Platform Agent:**

Working of the standalone agent will be similar to the the one as mentioned above in the OS part but this agent will have the provision of selecting the applications to create the restoration files for. This may be a standalone application having a UI for selecting the application to create restoration check points.

The agent shall be a cross platform application like as that of JVM which can be made available to all the OS currently in existence.

## III. Conclusion

With all the options considered, it appears that the efficiency of these three restoration methods is still an open ended question but one thing for sure is that, the applications if restored at certain milestones can help improve the lifetime of a software. The methods which are mentioned above can precisely help in determining the checkpoints/milestones for the restoration. In the plot against time and efficiency, it is quite possible to move the curve more to the ideal line as we restrain the application from withering away its life span everytime it receives an update.

## References:

[1]. https://www.tele-task.de/archive/podcast/9186/
[2]. Operating System Concepts: by Avi Silberschatz, Peter Galvin, Greg Gagne
[3]. Guide to Software Development: Designing and Managing the Life Cycle by Arthur M. Langer
[4]. Operating System: A Design-oriented Approach by Charles Crowley
[5]. A Brief Introduction to Software Development and Quality Assurance Management
[6]. by Steven C. Shaffer