FS InfoCat

Comprehensive Summary

Leonard T. Erwine

Western Governors University

000356334

Comprehensive Summary

Our local church was experiencing problems managing and finding files that were being stored on many different storage devices. As their storage needs increased, the cost-effectiveness of file storage subscriptions decreased. Although they did have written policy and governance on file storage practices, it was difficult to verify and enforce without causing disproportionate degradation of productivity. This was largely due to the inherent transiency of the mostly volunteer and part-time staff. There were many instances where someone had to re-create something that someone had already done before, simply because nobody knew where to find the files. They also found themselves periodically running out of storage space. The prospect of manually cleaning up the common storage areas was at least as costly as the price of increasing storage capacity. Due to the uncertainty of the actual content within the shared drives, it was not possible to accurately assess the cost in man hours, should they decide to go through each file to decide what can be consolidated or discarded. It is also conceivable that the pace of unmanaged data contributions could outpace the mitigation efforts. Even if they did reduce the storage footprint and reduce the cost of file storage services, that would not have solved the problem of recalling specific stored files when they are needed in the future.

FS InfoCat was designed to create a database catalog of meta data from files within multiple, disparate file systems, including network drives, local hard drives and removable media. Basic information, such as file location, size and modification dates would be gathered and stored in the database. Additionally, the Windows Property System will be leveraged to collect and store other meta data from files, such as author name, audio titles and video frame rates. Since not all devices support the Windows Property System, the application allows people to manually provide relevant information for each file.

One of the concerns that were expressed with this type of a solution was the possibility of the confidentiality of certain files being compromised through the publication of its meta data to the database. The mitigate this risk, the application initially store information to a local database within the user’s windows profile. It is only published to the centralized database when the user explicitly opts to do so. Additionally, the software can be configured to skip specific file and folders.

To assist with the mitigation of excessive storage utilization, this software checks for redundancies. It does this by creating a cryptographic hash code of each file that has the same length as another file. Files which share the same length and cryptographic hash are highly likely to be duplicates of each other. If deemed appropriate, files can be compared byte-by-byte to determine whether they truly are duplicates. In addition to the cryptographic hashing, which only indicates possible binary data duplication, other meta data can be search and compared, such as title and author, for instances where 2 files may have the same content but are stored with different quality standards.

The original intent was to produce an application in the first iteration which would be able to synchronize with a centralized database. However, since the shared drives are once again close to capacity, they would like to be able to utilize the software sooner. Therefore, the first release does not include this capability. In this state, it can still be used to gather and analyze information locally.

This software was designed using an amalgamation of test-driven and Agile development principals. Only the most critical components or those which are difficult to validate using event tracing or other debugging means were validated using unit tests. Due the complexity of the technology implementation required in order to meet the basic need of gathering and storing file meta data information, using true Agile development was not feasible, because the result of the first sprint would not have been usable. Therefore, the software was developed using a GitHub project where use cases and story boards were represented and organized using issue cards.

References