|  |
| --- |
|  |
| Java GlusterFS |
| Requirements Document |
|  |
| **CIS 4911 Senior Capstone Project**  **Section U01**  **Team Members: Maylem Gonzalez** |
| **Ian Herbig**  **Mentor: Louis Zuckerman**  **Instructor: Masoud Sadjadi**  **12/9/2014** |
|  |

*Abstract*

*This document details a high-level overview of the current state of GlusterFS and explores the requirements of the system for this project as provided by our mentor. It also contains the project schedule relevant to the timeframe in which this document was created and a complete work breakdown. The requirements description and analysis pays special attention to the use cases as the project’s implementation is scenario-based.*

Table of Contents

[1: Introduction 3](#_Toc406068759)

[1.1: Problem Definition 3](#_Toc406068760)

[1.2: Scope of System 4](#_Toc406068761)

[1.3: Definitions, Acronyms, and Abbreviations 5](#_Toc406068762)

[1.4: Overview of Document 6](#_Toc406068763)

[2: Current System 7](#_Toc406068764)

[3: Project Plan 9](#_Toc406068765)

[3.1: Project Organization 9](#_Toc406068766)

[3.2: Identification of Tasks, Milestones, and Deliverables 10](#_Toc406068767)

[3.3 Cost Estimate 11](#_Toc406068768)

[4: Proposed System Requirements 12](#_Toc406068769)

[4.1: Functional Requirements 12](#_Toc406068770)

[4.2: Analysis of System Requirements 13](#_Toc406068771)

[4.2.1: Scenarios 13](#_Toc406068772)

[4.2.2: Use Case Model 13](#_Toc406068773)

[4.2.3: Class Diagram 14](#_Toc406068774)

[4.2.4: Sequence Diagrams 15](#_Toc406068775)

[5: Glossary 16](#_Toc406068776)

[6: Appendix 17](#_Toc406068777)

[6.1: Appendix A – Use Cases 18](#_Toc406068778)

[6.2: Appendix B – Use Case Diagram 24](#_Toc406068779)

[6.4: Appendix D – Sequence Diagrams 26](#_Toc406068780)

[6.5: Appendix E – Diary of Meetings and Tasks 30](#_Toc406068781)

[7: References 36](#_Toc406068782)

# 1: Introduction

GlusterFS is a network-attached, virtual file system especially used to handle and process Big Data.The Java GlusterFS project is, in its current state, an incomplete implementation of Java 7’s NIO.2 file system API backed by GlusterFS via libgfapi-jni. This project seeks to make headway towards a more complete implementation of Java’s FileSystemProvider API.

## 1.1: Problem Definition

When GlusterFS was first introduced, applications could only connect to a Gluster volume through a FUSE mount. This was slow, but the only alternative was for developers to write their own API for their application. Apache’s Hadoop project did just this, at much cost.

Eventually, the GlusterFS project was extended with the introduction of libgfapi: an official API to allow applications to connect and communicate directly with a Gluster server, removing the need for a slow FUSE mount. This was much faster, but the API was written in C. Any projects that wished to use GlusterFS had to either write their application in C, or write some sort of interface between libgfapi and their platform of choice.

This problem has been solved for several platforms where developers created platform-specific bindings for libgfapi: eg. libgfapi-python is a Python binding for libgfapi, enabling Python developers to use a Gluster volume without writing a single line of C code.

This project seeks to do the same for the Java platform, but utilizing the NIO.2 API introduced in Java 7 to make developing an application that uses a Gluster volume as painless as manipulating files on any ordinary file system.

## 1.2: Scope of System

The Java GlusterFS binding’s extension can be described by the functionality that the system will contain after the project’s completion.

* Create directories
* Delete files and directories
* Copy files and directories
* Move files and directories
* Determine whether two files are the same
* Determine whether a directory is empty
* Retrieve the file store associated with a file
* Set and change owner/group names
* Advanced synchronous file I/O
  + Seeking & reading/writing a portion of a file

## 1.3: Definitions, Acronyms, and Abbreviations

* **API**: Application programming interface
* **File store**: The partition, drive, volume or other implementation-specific storage pool that a file resides in. For example, the C drive in Windows and the root directory in Linux are both file stores.
* **File system**: A system used to control how data is stored and retrieved. Focuses on logical units of storage rather than physically contiguous units when applied to a computer’s method of data storage.
* **FOSS**: Free, open-source software.
* **Horizontal scaling**: A form of scaling a system which involves adding more nodes to a system. Contrast with vertical scaling, where the size of each node in a system is larger.
* **I/O**: Input-output
* **Native applications**: Programs specific to hardware and operating system platforms
* **Network-attached storage**: file-level computer data storage connected to a computer network providing data access to a heterogeneous group of clients.
* **Scale-out**: see “horizontal scaling”
* **Synchronous I/O**: Input and output which blocks the progress of a program until its completion; also called blocking I/O
* **OS**: Operating system

## 1.4: Overview of Document

This document is broken up into 7 main chapters, and each chapter is further broken up into sections and subsections. This section marks the end of the first chapter. Chapter 2 consists of a description of the current system.

Chapter 3 describes the plan for the Java GlusterFS project related to both the requirements elicitation and analysis parts of the project’s lifecycle as well as the breakdown of the project as a whole. Section 3.1 describes the individual project roles assigned for this segment of the project. Section 3.2 contains a list of the tasks, milestones and deliverables to be completed in the duration of this project. While section 3.3 features our cost estimate for the project’s development.

Chapter 4 is the most important part of the document as it handles the proposed system requirements. Section 4.1 involves the functional requirements described on a high level. Section 4.2 analyzes those system requirements by breaking them into scenarios (subsection 4.2.1), a use case model (subsection 4.2.2), a static model (subsection 4.2.3), and a dynamic model (subsection 4.2.4).

Chapters 5, 6, and 7 are the metachapters in that they concern themselves with the content in the previous chapters. Chapter 5 is a glossary, chapter 6 contains the appendices, and Chapter 7 contains any references we made throughout the document.

# 2: Current System

The state of the Java GlusterFS binding can be described in three forms: its functionality, the limitations on that functionality, and the system’s constraints. They are as follows:

**Functionality**:

* Able to connect to a GlusterFS volume using the NIO.2 API
* Able to perform basic synchronous file I/O
  + Reading the contents of a file all at once
  + Writing a chunk of bytes to a file all at once
* Able to handle file attributes
  + See owner/group id (as a number), size, permissions, and last modified timestamp on files and directories
  + Set file permissions
* Able to view file system/volume statistics
  + See the total, free, and usable bytes in a volume
* Directory listing (with filtering) functionality
* Able to rename files
* Able to watch files for changes
* Able to publish test coverage and code quality reports to SonarQube

**Limitations**:

* Unable to perform asynchronous I/O
  + Cannot seek and read/write portions of a file
* Basic error reporting and handling
* Unable to copy files
* Unable to move files
* Unable to delete files and directories
* Unable to determine whether two files are the same
* Unable to retrieve the file store that a file is associated with
* Unable to create directories
* Unable to determine whether a directory is empty
* Watch system is slow and blocking, utilizing polling
* Attribute support limited
  + Owner/group identification limited to numbers, no name support
  + Cannot change owner/group

**Constraints**:

* System is written in Java
* System is only usable on a 64-bit Linux OS

# 3: Project Plan

This chapter focuses on the details of project management. We specifically focus on the project organization for this segment of the project’s lifecycle and the identification of tasks, milestones, and deliverables throughout.

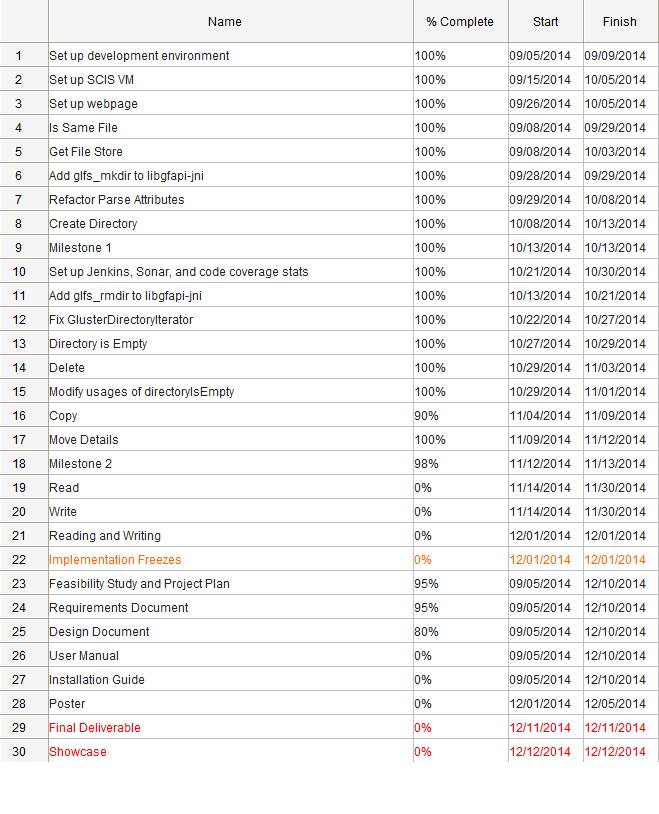
## 3.1: Project Organization

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Description** |
| **Ian Herbig** | Project Manager/Document Editor/Document Writer/Modeler | Serves as the project manager for the other team member. Edits and writes sections of the document. Creates required models. |
| **Maylem Gonzalez** | Project Manager/Requirements Elicitor/Document Writer | Serves as the project manager for the other team member. Gathers the requirements from the mentor. Writes the content for the document. |

## 

## 3.2: Identification of Tasks, Milestones, and Deliverables

The following schedule is an incomplete schedule subject to revision.



## 3.3 Cost Estimate

As GlusterFS is licensed under the GNU General Public License, it is FOSS. This means that all source necessary for the project must be free and open source. We are also not required to purchase any hardware for the project’s completion. There is no personnel cost since we are working for free, therefore all costs associated with the personnel (training and testing) are also null.

Thus, we can sum up the costs associated with the project in the following table:

|  |  |
| --- | --- |
| **Project Component** | **Monetary Cost** |
| **Hardware** | $0 |
| **Software** | $0 |
| **Personnel** | $0 |
| **Training** | $0 |
| **Testing** | $0 |
| **Total** | $0 |

# 4: Proposed System Requirements

This chapter details the tentative functional and non-functional requirements attained by analyzing the high-level user requirements. Section 4.1 describes the functionality on a high level by enumerating functionality the system shall have and the associated non-functional requirements for each element of that set as required.

Following this, we analyze these system requirements in section 4.2 with an emphasis on generating scenarios from which we can draw specific use cases to be implemented. Models and diagrams follow naturally.

## 4.1: Functional Requirements

* *The system shall allow a user to delete a file from a GlusterFS volume*
* *The system shall allow a user to copy a file within a GlusterFS volume*
* *The system shall allow a user to move a file within a GlusterFS volume*
* *The system shall allow a user to read a portion of a file*
* *The system shall allow a user to write to a portion of a file*
* *The system shall allow a user to create a directory*
* *The system shall allow a user to determine whether two files are the same*
* *The system shall allow a user to retrieve the file store that a file is associated with*

The above functional requirements do not have non-functional requirements that could be considered under the umbrella of usability, except those specified by the Java 7 NIO.2 API. They must work in 90% of the cases in which they are used, barring exceptional circumstances. And copying, moving, reading, and writing, are variables in terms of performance as they are dependent on the size of the file. A “small” file should not take a noticeable amount of time.

All other non-functional requirements of the system are specified by the non-functional concerns of the depending systems (libgfapi and glusterfs).

## 4.2: Analysis of System Requirements

This section contains an explanation of the diagrams displayed in the appendices and the scenarios that helped build them.

### 4.2.1: Scenarios

Maylem wants to delete a file from a directory in her GlusterFS volume. She knows the path of the file, but she isn’t sure if the file exists, so she calls the Java GlusterFS method *deleteIfExists* on the file path to remove the file from the GlusterFS volume. Maylem knows that the method returns true if the file was deleted and false if the file did not exist, so she stores the return value and prints it to know the status of the file.

Ian wants to copy a file from a directory in his GlusterFS volume to a local directory. He knows the path of the file in the Gluster volume, as well as the target path in his directory. He calls the Java GlusterFS method *copy* with the source path and the target path. He knows that he can add any number of additional copy options to the method call, but he only wants to do a simple copy so he does not specify any. He does not receive a *FileAlreadyExistsException*, so he knows that he did not already have the file copied in his local directory.

Maylem wants to determine if two files in her GlusterFS volume are the same. She has the paths to the two files, but is not sure if the files are the same because the two paths are not identical. She calls the Java GlusterFS method *isSameFile* with the paths to the two files to determine this. She knows that this method returns true if the two paths locate the same file, so she stores the return value, prints it, and sees that *false* is printed.

Ian wants to create a new directory in his GlusterFS volume. He knows where he wants the new directory to be created, so he calls the Java GlusterFS method *createDirectory* with his desired path. He knows that the method returns the directory of the path, so he stores the return value and prints the result to see the path to his newly created directory. He sees that the directory is printed, and does not receive an exception, so he knows that the operation was successful.

### 4.2.2: Use Case Model

The diagram in appendix B provides a limited overview of the use cases we have determined given the aforementioned scenarios (appendix A) and how users of our system may interact with the use cases.

We specify two types of users in the diagram: the User is someone who wishes to utilize GlusterFS from within a Java program; the System is involved in the calling of private methods involved in the execution of system methods.

Some of the use cases interact with some other functionality that we describe, but as this process is iterative, the diagram and use cases will likely be expanded upon in the future as progress on the system is made and insights are had into what may be necessary for the completion of future milestones.

### 4.2.3: Class Diagram

The first class diagram visible in appendix C (Figure 1) is the minimal class diagram for the Java GlusterFS system, while the second class diagram in appendix C (Figure 2) shows the parents of each of the classes. Though they are not broken up as such, almost every class in the class diagrams can be seen to make up a subsystem in itself. Exception to this are classes like the GlusterDirectoryStream and GlusterDirectoryIterator which both make up the subsystem responsible for parsing the contents of a directory.

The core classes, GlusterFileSystemProvider, GlusterFileSystem, and GlusterPath, are the classes in which most user-driven logic is performed and are the focus of our work on the project (excluding our extension of the libgfapi-jni project, which is merely creating JNI bindings to the libgfapi project). Users do not typically see the aforementioned classes directly, preferring instead to go through the NIO.2 Files API which enables generic typing.

### 4.2.4: Sequence Diagrams

The sequence diagrams contained within appendix D are based on our work on the current system and reflect a general idea for the flow of control throughout the system. Generally, when a function is user-driven, the sequence diagram will reflect that the user goes through the Files class rather than handling the Java GlusterFS library itself. Pay special attention to notes within the diagrams as the notes indicate where functionality may be lacking, or assumptions that are made.

For example, in the isDirectoryEmpty sequence diagram, a note is made that the directory entries parsed by the GlusterDirectoryIterator are checked against the special directory entries of “.” and “..” which represent “this” and “parent” respectively, and discarded if they match. This functionality is something we aim to implement, and the diagram reflects the current state of the function.

# 5: Glossary

* **GlusterFS**: A scale-out, networked-attached storage platform
* **Java Native Interface (JNI)**: A programming framework which allows Java code to call and be called by native applications and libraries written in other languages
* **libgfapi**: A library for accessing data in GlusterFS
* **libgfapi-jni**: Java Native Interface bindings for the libgfapi
* **NIO.2**: Java 7’s enhanced file package which provides file system APIs
* **NIO**: Non-blocking I/O; also called asynchronous I/O

# 6: Appendix

This chapter consists of the following appendices:

* Appendix A – Use Cases
* Appendix B – Use Case Diagram
* Appendix C – Class Diagram
* Appendix D – Sequence Diagram
* Appendix E – Diary of Meetings and Tasks

## 6.1: Appendix A – Use Cases

The use cases described herein represent the functionality involved up to the second milestone of the project. Use cases will be elaborated on as the milestones are elaborated on throughout the project and as time allows for expansion of functionality.

|  |  |
| --- | --- |
| **Use Case ID** | **Copy (java-glusterfs-02)** |
| **Description** | Allows a user to copy a file or directory. |
| **Actor** | All Users |
| **Preconditions** | * User has a path to a file or directory to be copied. * User has a set of “copy options” that they desire. * User has a path to the target directory. * User has connected to a GlusterFS volume in their program. |
| **Details** | 1. The use case begins when the user calls the copy method in their program. 2. The system shall check if the desired copy options are supported. 3. The system shall check if the destination path exists. 4. The system shall check if the destination path is an empty directory. 5. The system shall copy the source file to the destination path based on the desired copy options. |
| **Postconditions** | The file or directory has been copied to the specified target location. |
| **Exceptions** | * The file already exists in the target location. * The directory already exists in the target location and it is not empty. * An I/O exception occurs. |

|  |  |
| --- | --- |
| **Use Case ID** | **Create Directory (java-glusterfs-03)** |
| **Description** | Allows a user to create a new directory. |
| **Actor** | All Users |
| **Preconditions** | * User has a path to the desired directory location. * User has connected to a GlusterFS volume in their program. |
| **Details** | 1. The use case begins when the user calls the createDirectory method in their program. 2. The system shall check if the path exists. 3. The system shall check if the parent of the path exists. 4. The system shall create a new directory at the location specified by the supplied path with a default set of permissions. |
| **Postconditions** | The directory has been created at the location specified by the path supplied by the user. |
| **Exceptions** | * A file with the same name requested by the user for the directory already exists. * The parent directory does not exist. * An I/O exception occurs. |

|  |  |
| --- | --- |
| **Use Case ID** | **Is Same File (java-glusterfs-04)** |
| **Description** | Allows a user to check if two files are the same. |
| **Actor** | All Users |
| **Preconditions** | * User has the paths to two files. * User has connected to a GlusterFS volume in their program. |
| **Details** | 1. The use case begins when the user calls the isSameFile method from within their program. 2. The system shall check if the two paths are the same. 3. The system shall check if the file systems of the two paths are the different. 4. The system shall check if both paths exist. 5. The system shall retrieve the inode number of both files. 6. The system shall check whether the inode numbers of both files are the same. 7. The system shall return whether or not the two inode numbers are the same. |
| **Postconditions** | The user knows if the two files are the same. |
| **Exceptions** | * An I/O exception occurs. * Either file does not exist. |

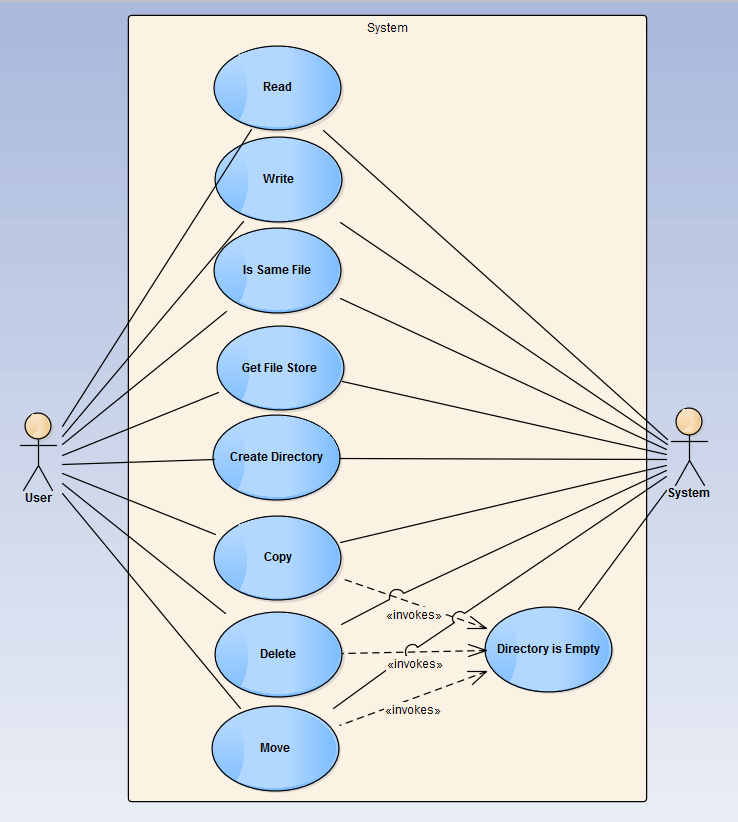
|  |  |
| --- | --- |
| **Use Case ID** | **Directory Is Empty (java-glusterfs-05)** |
| **Description** | Allows an actor to check if a directory is empty. |
| **Actor** | System |
| **Preconditions** | * Actor has a path to a directory. * Actor has access to a GlusterFS volume. |
| **Details** | 1. The use case begins when the actor calls the directoryIsEmpty method. 2. The system shall check if the path is a directory. 3. The system shall check if the directory has any entries. 4. The system shall return whether or not the directory has any entries. |
| **Postconditions** | The actor knows if the directory is empty. |
| **Exceptions** | * An I/O exception occurs. * The specified path does not exist. * The specified path is not a directory. |

|  |  |
| --- | --- |
| **Use Case ID** | **Delete (java-glusterfs-01)** |
| **Description** | Allows a user to delete a file or directory. |
| **Actor** | All Users |
| **Preconditions** | * User has a path to a file or directory to be deleted. * User has connected to a GlusterFS volume in their program. |
| **Details** | 1. The use case begins when the user calls the delete method in their program. 2. The system shall check if the file is a directory. 3. The system shall check if the directory is empty. 4. The system shall delete the file or directory from the volume specified by the path provided. |
| **Postconditions** | The file has been deleted. |
| **Exceptions** | * The file does not exist. * The path specifies a directory and it is not empty. * An I/O exception occurs. |

|  |  |
| --- | --- |
| **Use Case ID** | **Get File Store (java-glusterfs-06)** |
| **Description** | Allows a user to retrieve the file store of a file or directory. |
| **Actor** | All Users |
| **Preconditions** | * User has a path to a file or directory. * User has connected to a GlusterFS volume in their program. |
| **Details** | 1. The use case begins when the user calls the getFileStore method. 2. The system shall check if the file exists. 3. The system shall get the file system associated with the file. 4. The system shall get all of the file stores associated with the file system. 5. The system shall return a file store in that set. |
| **Postconditions** | The user has the file store associated with the file. |
| **Exceptions** | * The file does not exist. * An I/O exception occurs. |

## 6.2: Appendix B – Use Case Diagram

**Figure 1: Use Case Diagram**

****

6.3: Appendix C – Class Diagrams

Figure 1: Minimal Class Diagram

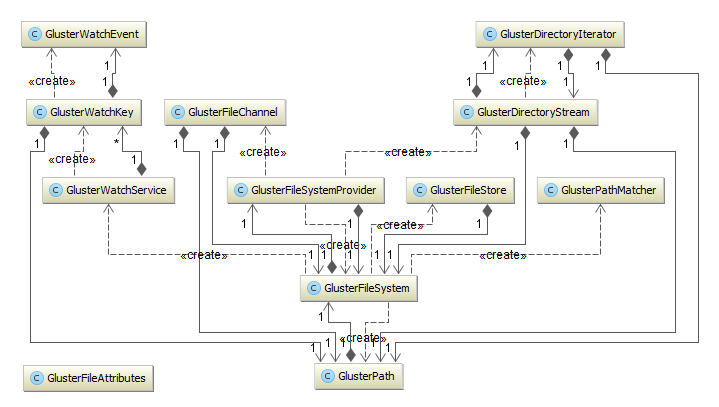
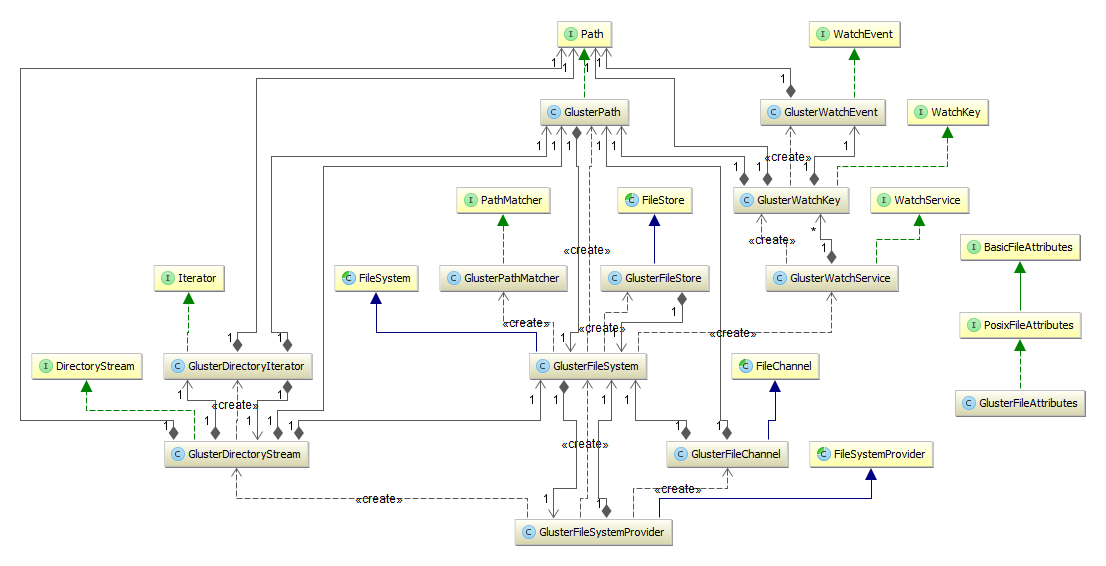


Figure 2: Minimal Class Diagram with Parents



## 6.4: Appendix D – Sequence Diagrams

Figure 1: Is Same File

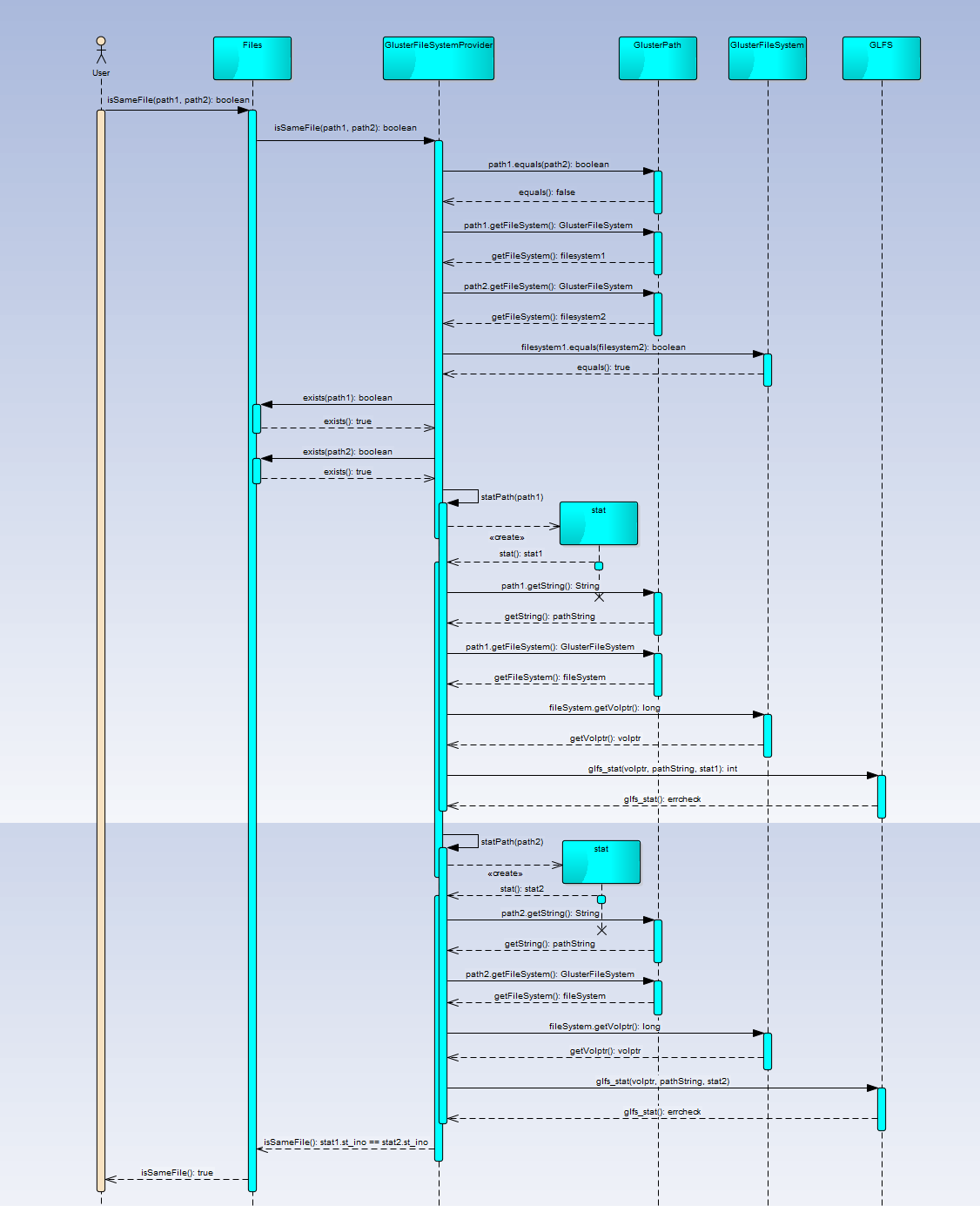


Figure 2: Get File Store

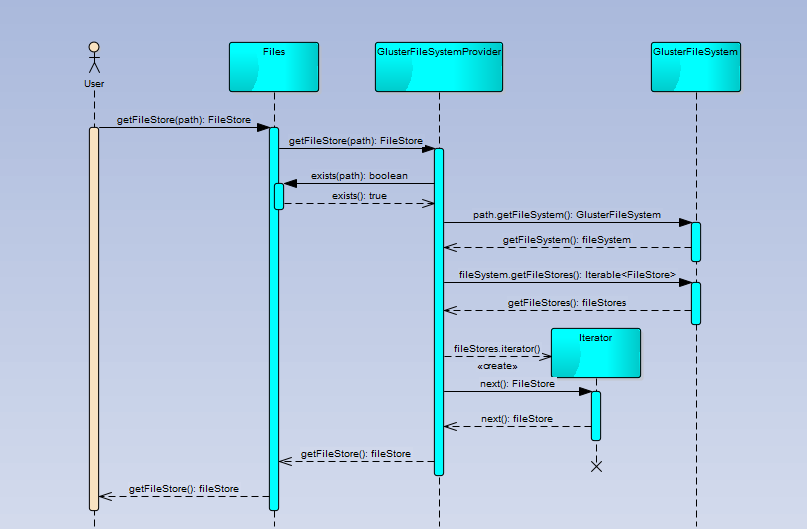


Figure 3: Create Directory

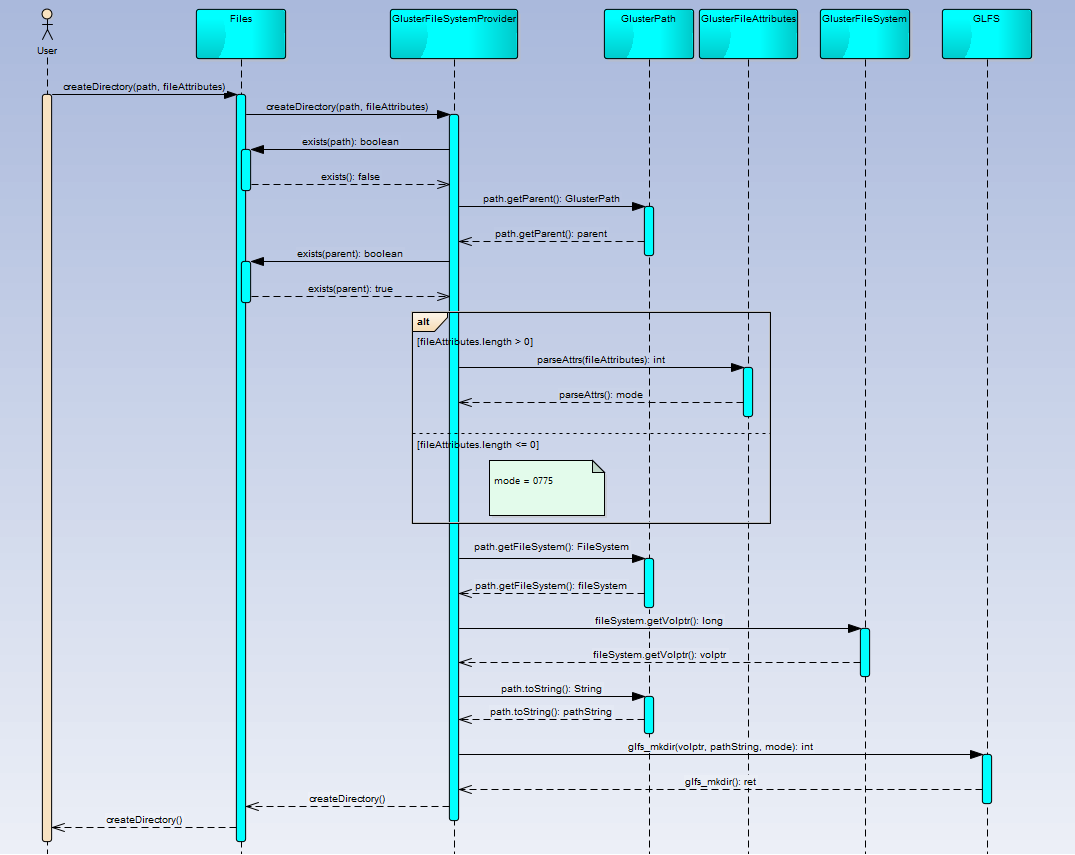
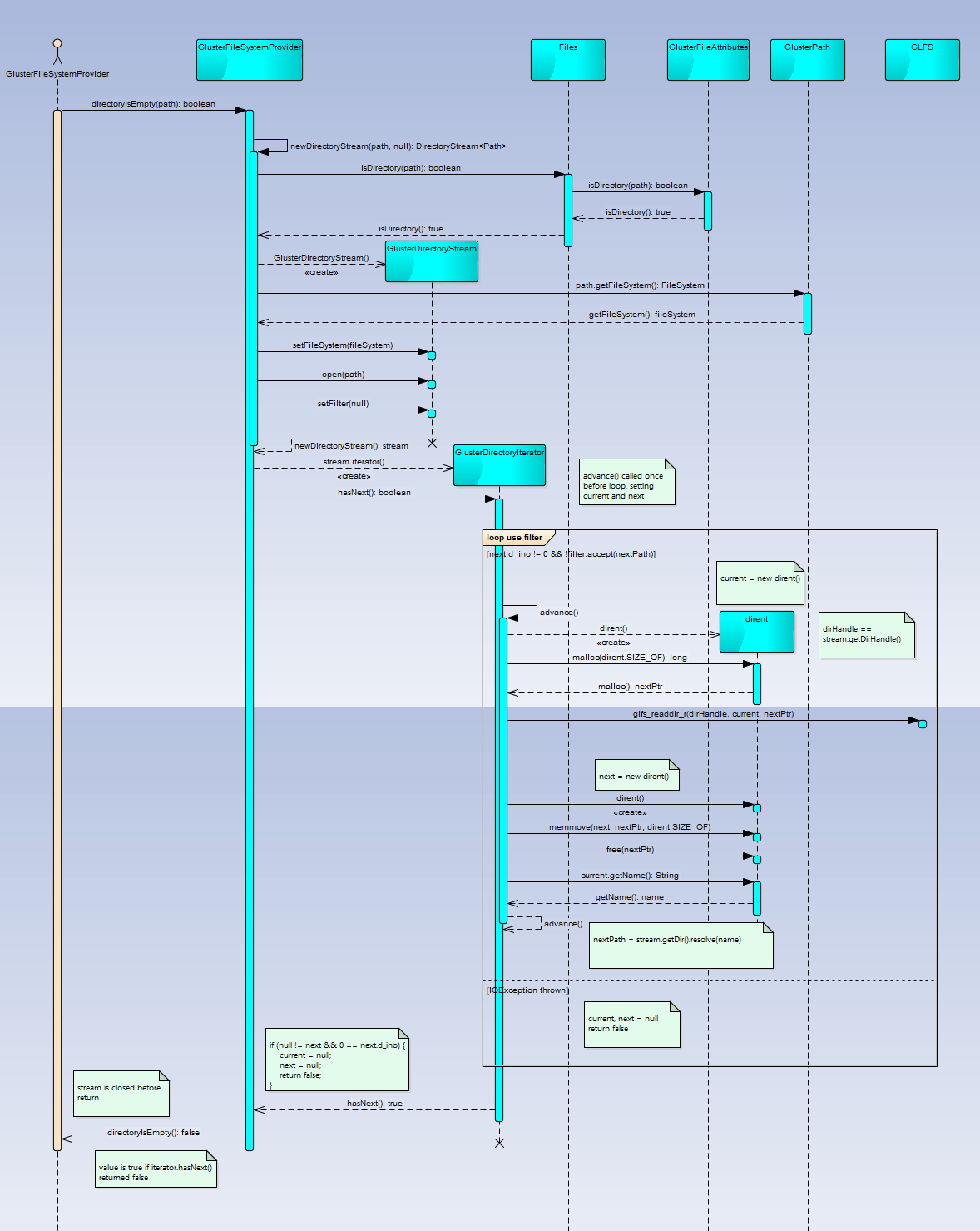
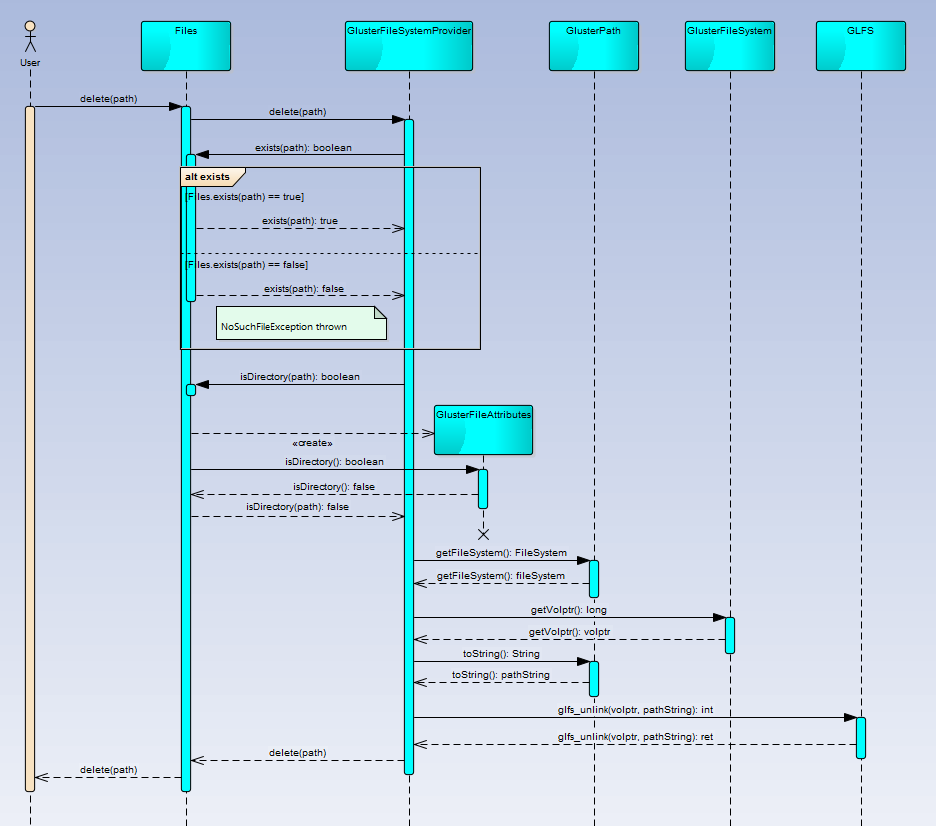


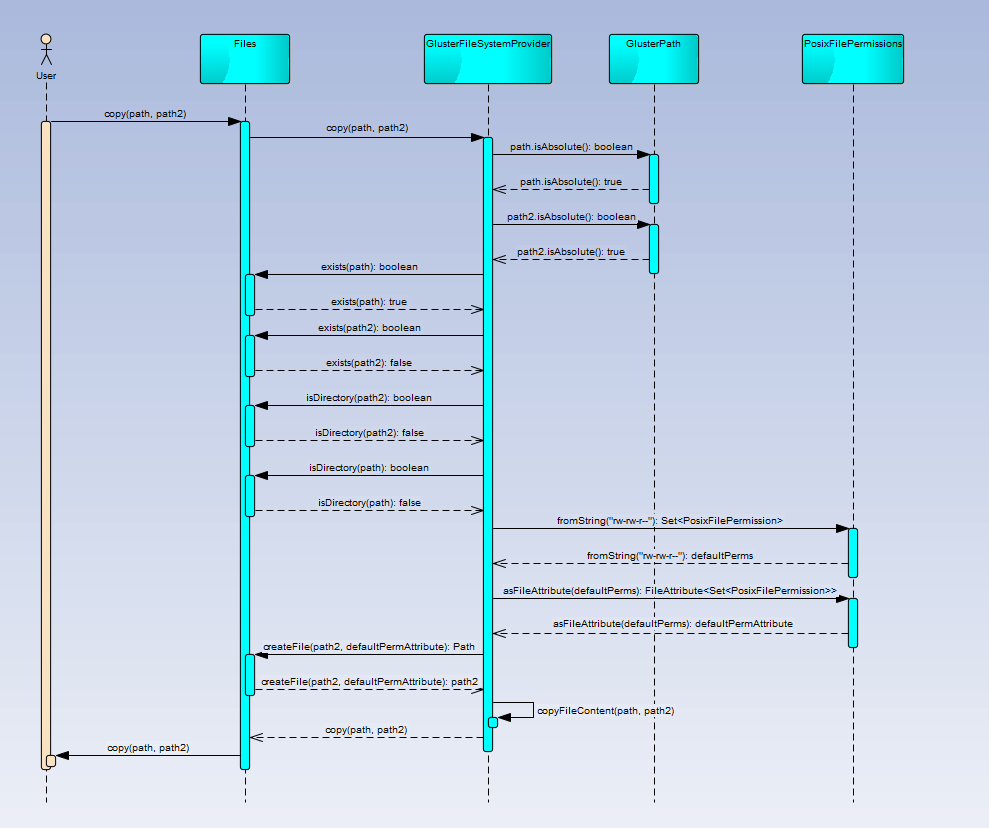
Figure 4: Directory is Empty



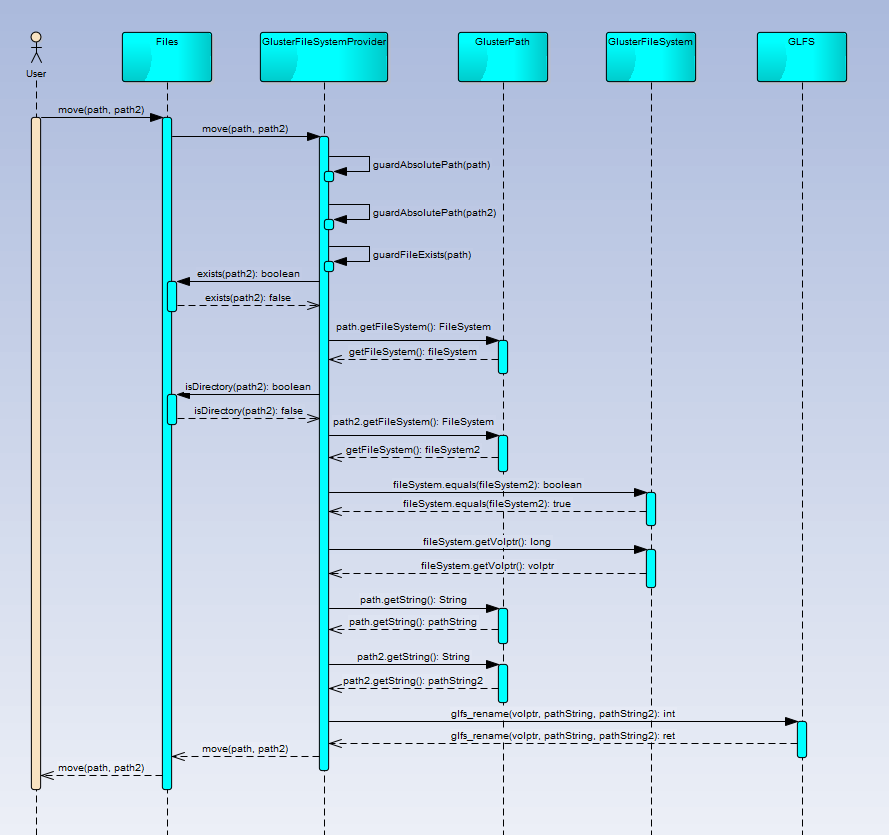
**Figure 5: Delete**

****

**Figure 6: Copy**

****

**Figure 7: Move**

****

6.5: Appendix E – Diary of Meetings and Tasks

**Begin meeting**: 9/3/14 7:15PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Personal introductions
* Overview of GlusterFS
* Overview of development environment
* Current state of glusterfs-java-filesystem
* Demos detailing functionality and unit tests
* Discuss extent of project (resulting in timeline)

**End meeting**: 9:00PM

**Next meeting date**: 9/7/14 3PM

**Next meeting agenda**: Set up development environment, sort out development logistics (physical and management perspectives)

**Start meeting**: 9/7/14 1:55PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Forked GlusterFS-Java-Filesystem repository into FIU SCIS organization
* Overview of Maven commands
* Set up development environment
  + Made shell script to run IntelliJ with correct Java 7 JDK
  + Built libgfapi-jni with Maven
  + Built glusterfs-java-filesystem with Maven
  + Ran glusterfs-java-filesystem-example with Maven
  + Ran/Debugged glusterfs-java-filesystem tests with IDEA
  + Ran/Debugged glusterfs-java-filesystem-example with IDEA
* Broke up milestones into tasks
* Discussed class diagrams and use cases

**End meeting**: 5:10PM

**Next meeting date**: 9/14/14 3PM

**Next meeting agenda**: Resolve issues with boot testing VM with Vagrant, discuss progress and position in timeline, and identify further tasks.

**Start meeting**: 9/14/14 3:40PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed shortcuts to make the development process more efficient in IntelliJ IDEA
* Concluded that toFile() should not be implemented in the project
* Discussed how relativize and resolve work in the library
* Overview of Mockito, Lombok plugin, and PowerMock syntax for JUnit testing
* Discussed the acceptance process to be used in the project

**End meeting**: 6:10PM

**Next meeting date**: 9/21/14 4PM

**Next meeting agenda**: Identify further tasks, discuss any issues with JUnit testing, and discuss the acceptance of any implemented and tested features.

**Start meeting**: 9/17/14 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Addressed questions concerning the creation of JUnit tests
* Overview of GlusterFS and how it works
* Decided on the structure of the git repository
* Discussed whether the SCIS virtual machine would suit our needs
* Overview of Jenkins and continuous integration

**End meeting**: 8:30PM

**Next meeting date**: 9/21/14 4PM

**Next meeting agenda**: Discuss the acceptance of the isSameFile and getFileStore features along with any questions concerning JUnit testing.

**Start meeting**: 9/21/14 3:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the acceptance of the two completed features (issamefile and getfilestore)
* Overview of pull requests
* Discussed changes needed to be implemented in libgfapi-jni project for the upcoming createDirectory and directoryIsEmpty features
* Discussed integration testing in the libgfapi-jni project
* Overview of file attributes
* Discussed preferred programming practices for this project
* Overview of HawtJNI

**End meeting**: 5:30PM

**Next meeting date**: 9/28/14 4PM

**Next meeting agenda**: Discuss the next round of features set for acceptance along with any questions concerning testing in the libgfapi-jni project.

**Start meeting**: 9/28/14 7:00PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to parse file attributes and how to refactor our initial implementation to best accomplish this
* Solved the issue of glfs\_mkdir not being recognized in GlusterFileSystemProvider
* Discussed a procedure to follow for the next time that we encounter an issue with the snapshot and missing dependencies
* Discussed how to best demo our work and agreed to use Jenkins and Sonar
* Discussed how to fix issues encountered in getting our development environment to run in FIU's SCIS virtual machine

**End meeting**: 8:15PM

**Next meeting date**: 10/5/14 4PM

**Next meeting agenda**: Discuss the next round of features set for acceptance, and determine how to tie Jenkins and Sonar to our project.

**Start meeting**: 10/13/14 6:50PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to use and set up Jenkins with the Java GlusterFS project
* Discussed issue where createDirectory was resulting in an error on some machines
* Discussed how Java handles the default permissions for a new file when no file attributes are provided
* Discussed the acceptance of the createDirectory feature

**End meeting**: 8:50PM

**Next meeting date**: 10/20/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents.

**Start meeting**: 10/20/14 7:10PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed a potential issue with the current GlusterDirectoryIterator implementation where the special entries "." and ".." were being returned (contrary to the behavior of the default provider)
* Determined the best fix for the GlusterDirectoryIterator issue
* Discussed the current implementation of isDirectoryEmpty
* Discussed a potential issue with calls to verifyStatic() in the JUnit tests
* Discussed how to set up Jenkins on the FIU VM

**End meeting**: 8:30PM

**Next meeting date**: 10/26/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents. Additionally, possible modifications to the Jenkins and Sonar profiles will be addressed.

**Start meeting**: 10/29/14 7:10PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to set up the Atlassian Clover code coverage tool with Jenkins and Sonar
* Discussed possible tools and locations for filming our videos
* Discussed the current state of the documentation, including class diagrams and sequence diagrams
* Discussed the extent to which security and file permissions will be implemented and enforced

**End meeting**: 8:10PM

**Next meeting date**: 11/3/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents and videos. Additionally, brainstorm real-world scenarios where our Java library can be used.

**Start meeting**: 11/12/14 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the pull request for dirIsEmpty branch and any necessary changes to the current implementation or tests. Decided to modify if statements in advanceHelper method to switch cases.
* Discussed the move feature (specifically, how its implementation differs from spec) and decided that it is only missing a throws clause to be complete.
* Discussed the copy feature and how file attributes would be copied over (using chmod, chown).
* Discussed utilizing the platform-specific umask value in the logic of create directory. Decided that it would be best to not do this.
* Changed the priority of (1) user and group name and (2) reading and writing features. Reading/writing has a higher priority now, and user and group name feature will be done only if time permits.
* Discussed what aspects of the reading and writing features will need to be prioritized.
* Discussed how the user and group names feature would be implemented using a Linux UserPrincipleLookupService.

**End meeting**: 8:25PM

**Next meeting date**: 11/17/14 7PM

**Next meeting agenda**: Discuss the implementations and tests of delete, copy and move, as well as discuss the current status of the project documents and videos.

**Start meeting**: 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the architectural and design patterns found in the Java-GlusterFS project. Determined that the following six patterns apply: Service Provider Interface, Abstract Factory, Object Pool, Singleton, Iterator, Fluent Interface, and Prototype.
* Determined that time should currently not be spent on trying to get Cobertura to work with our project.
* Discussed the use of isSameFile() in the move feature and how it results in a difference from specs. Decided that a note should be added to detail this change in behavior for future consideration.
* Discussed the use of ByteBuffer in reading/writing bytes in the project.
* Discussed the implementation of write(ByteBuffer src) and determined that the copy feature needs to be put on hold until the write implementation is modified to spec and bugs with the current implementation are fixed.
* Decided that UtilJNI should be made public instead of package protected.

**End meeting**: 8:15PM

# 7: References

|  |  |
| --- | --- |
| [1] | L. Zuckerman, "GitHub Repository for Java GlusterFS," [Online]. Available: https://github.com/semiosis/glusterfs-java-filesystem. [Accessed 5 9 2014]. |
| [2] | J. Darcy, "GlusterFS Internals and Directions," in *Red Hat Summit*, Boston, 2013. |
| [3] | "File I/O (Featuring NIO.2)," Oracle, [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/fileio.html. [Accessed 5 9 2014]. |
| [4] | "Java SE 7 Java Native Interface-related APIs and Developer Guides," Oracle, [Online]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/jni/. [Accessed 5 9 2014]. |
| [5] | "java.nio.file (Java Platform SE 7)," Oracle, [Online]. Available: http://docs.oracle.com/javase/7/docs/api/java/nio/file/package-summary.html. [Accessed 7 9 2014]. |