SOFTWARE COLLABORATION FEDERATION

Architectural Concept Document

CSE-681 Software Modeling and Analysis
Fall 2016 – Final Project
Instructor – Dr. Jim Fawcett

Venkata Santhosh Piduri
vepiduri@syr.edu
SUID:944740835
Date: December 07,2016

Table of Contents

1 Executive Summary

To develop software projects from scratch, organization follows one of the Software Development Life Cycle methods to get fruitful results. This Architectural Document considers one of the most famous SDLC i.e. Agile Development and also describes how Software Collaboration Federation supports SDLC for developing software projects.

Software Collaboration Federation (SCF) is Federation of Collaboration Server, Repository Server, Build Server, Test Harness and Client. Main goal of SCF is to support continuous test and integration using Test Harness server in turn reduces integration time in Agile development. SCF achieves this goal with the help of Repository server and build server.

Primary users of SCF are Software Project Manager, Software Architect, Software Development team, Quality Assurance team.

Each Server is designed in loosely coupled fashion, so that there won't be any dependencies between servers.

Repository Server – Provides open check-in, version control features, helps in tracking versions of source code, specification documents, test suits etc. and uses storage management system to store test log results, meta-data files to maintain baseline code repository.

Build Server – This server is flexible in setting up its own environment automatically for projects using configurations file. Contains package manager which is like Maven in java, can parse configuration files to download and install packages required, to setup environment. After setting environment, based on request, build server will download source code files from repository and creates executable images or libraries.

Test Harness – This server performs continuous tests and integration for all test suits and source code present in repository.

Collaboration Server – This server provides all features related to collaboration in agile development cycles. Such as creating project team structure, creating specification documents, scheduling meetings etc.

Client – Has tabbed fashion GUI, uses Single Sign-on concept to login into the servers.

Following are some of the critical issues that should be considered during development of Software Collaboration Federation.

1. Current file access issues
2. Maintenance issue (GUI)
3. Data integrity issue

2 Federation

In Initial stage of project development, Software Project Manager and Software Architect uses Collaboration Server to document gathered requirements or stories from customers. Later they use collaboration server for creating project teams, organizing sprints, scheduling scrum meetings and to disclose project statuses.

When project developers started working on project, they use Repository Server for storing specification documents, source code, test suits, test log results and other documents needed for project. Repository server also have check-in and version control features which helps developers to check-in their code and versioning them respectively and also maintain baseline code.

2.1 Introduction

Software Collaboration Federation is group of things, which includes both clients and servers and associated software designed to support SDLC activities of development team.

2.1.1 Obligations

1. Software collaboration federation should support continuous test and integration using Test Harness server and repository server.
2. Application should support automatic testing for running many tests efficiently.
3. While testing application should be able to provide isolated environment for each test request.
4. Should provide ITest interface for all users, helps user to writer test cases of an input application.
5. All servers should be able to expose their services using WCF, so that client can able to communicate with servers.
6. All servers should be able to accept multiple requests, and run concurrently.
7. Client user interface should be designed and implemented using Windows presentation Foundation (WPF)
8. SCF should support collaboration, scheduling meetings, storing project management information.

2.1.2 Organizing Principles and Key Architecture Ideas

1. We will use .Net Framework-4 features and Visual Studio 15 for developing Software Collaboration Federation application.
2. Entire system is resided in Test Harness server, Repository server, Collaboration server, Virtual display server and client machine. All automatic testing related functionality will be there in Test Harness server, Assemblies, Storage related functionality will be resided in Repository server, User Interface for Test Harness will be there in client machine, Project management information we be there in collaboration server, and scheduled collaborated information will be available in Virtual display server.
3. Followed peer to peer communication architecture to communicate between each system mentioned above using Windows Communication foundation.
4. Entire architecture is designed using Model View Controller design pattern where model is separated from view and controller, view is separated from model and controller, in the

same way controller is separated from view and model, so that any changes to any module will have very minimal code changes in other modules.

5. Test Harness, Repository, Collaboration, Build server structure is designed in a way that application can be able to handle multiple requests at the same time. In order to implement this feature, one Producer-multiple Consumer thread pool design pattern is adapted.
6. Make use of AppDomain feature in .Net Framework that allows us to create isolated environment for each test request.
7. Make use of LINQ XML for reading and parsing of test requests and use of .Net I/O features to search dynamic linked libraries and copy them to temporary directory.
8. Blocking Queue will be used to hold test requests from user.
9. MongoDB resided in Repository server is used to store test request results. With this data will be available for querying about test requests results and analyzing project stability.
10. Connection Pool concept is used to get MongoDB connection instance, so that we can ensure that all the connection instances are used efficiently.
11. Logging mechanism implementation helps in debugging application easily.
12. User Interface is developed by following Command Pattern, where user click on tabs will be considered as command.
13. User Interface is designed by using Windows Presentation Foundation provided by .Net Framework.
14. For developing maintainable code, complete user interface is developed by following MVVM pattern.
15. Command pattern is used for all the buttons in GUI.

2.2 Uses

2.2.1 How users use the Software Collaboration Federation?

Primary stakeholders of systems are Software Project Manager, Software Architect, Software Development Team, Quality Assurance Team.

2.2.1.1 Software Project Manager

Software Project Manager uses collaboration server tool to organize project team structure, creates or edit sprint work packages, schedule weekly meetings to know about project status, view Test log results from repository server which helps to know the stability of project, schedule collaborative meetings with off-shore/on-shore teams, gathering project requirements from customer and documenting them.

Impact on Design:

As this is one of the important works of Software development in Agile SDLC and most of the work is related to collaboration server, it provides collaboration tool for clients with rich user interface which helps in creating team structure, documenting, scheduling meetings, discloses project status.

2.2.1.2 Software Architect

Software Architect is responsible for the following tasks.

1. Designing whole project design and user interfaces

2. Breaking large requirement stories into modules and assigning them to software development individuals.
3. Suggesting solutions for business problems with in project.
4. Preparing Architectural documents for projects.
5. Creating sprints based on requirements.

Impact on Design:

Collaboration server provides sprint work package tool which helps in breaking requirements and assigning them to developers.

2.2.1.3 Software Development Team

Following are some responsibilities of Software developers in team.

1. Responsible to write code for modules.
2. Responsible to write test suits.
3. Uses Test Harness to verify code as part of check-in process and defect analysis.
4. Can be a part of collaborative discussions

Impact on Design:

Federation is providing rich user interface, which makes developer to interact with repository server, test harness server and collaboration server.

2.2.1.4 Quality Assurance Team

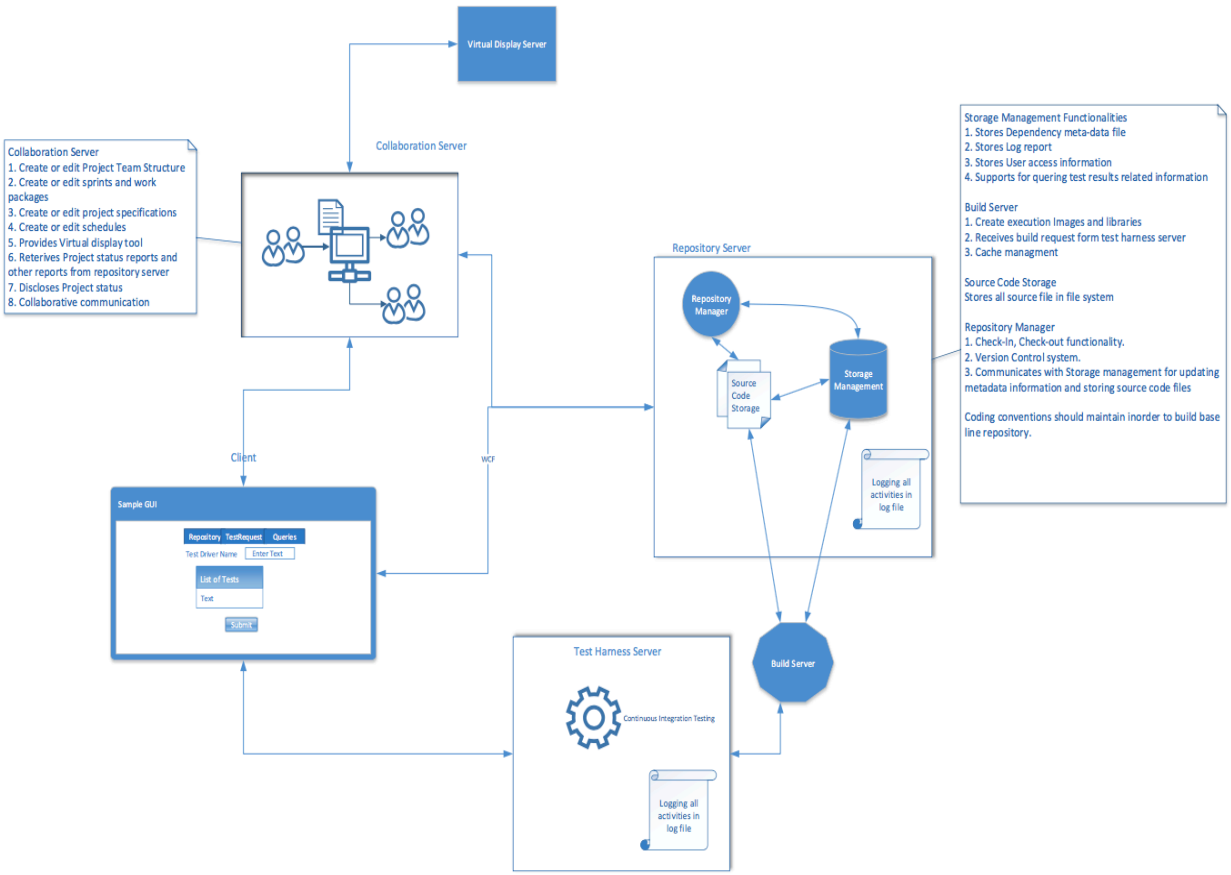
This team focus on the testing process in preparation for acceptance testing and on maintaining code quality.

Impact on Design:

This team will be submitting test request messages for large bodies of code. It is important that collection of dependent code packages be automated and that the TestHarness operates as efficiently as possible. It is also important that we be able to scale out by adding new TestHarness servers without a lot of rework to the existing Federation infrastructure.

2.3 Structure

2.3.1 Overall System Architecture



Software Collaboration Federation is a federation of Test Harness server, Repository server, Build server, Collaboration server, Visual Display server, and client. Each and every server and client communicates with each other using Windows Communication Foundation. Mostly useful for SDLC activities of software development team.

1. **Repository Server:**

   Main function of Repository server in SCF is to store source code metadata information, source code file, specification documents by managing versions. Following are some of import concepts for repository server.

   Single Ownership policy: When work package is assigned to responsible individual developer, he is the only responsible person who has complete rights in developing, modifying, and deleting that package. Other developers will have only view right.

   - Responsible Individual can able to check-in modified package, check-out for modifications.
   - Other developers have only extract operation, where they can only view and edit package but don't have permission to check-in that package.

   Versioning: Version controller package in repository manages versioning. Following steps describes how versioning takes place when there is request for check-in file.

- Creates or update metadata information in storage management system. when there is check-in new file or modified file.
- Creates directory with version# as name.

Metadata information will be as follows.

```
{
    "Project-Solution": "SCF",
    "List-OF-Sub-Systems": [{
            "Sub-System": "Test Harness",
            "List-Of-Modules": [{
                    "Module-Name": "TestHarness",
                    "Packages": [{
                            "Package-Name": "Controller",
                            "Current-Version": "V1.5",
                            "ListOfVersions": ["V1.0", "V1.1", "V1.2", "V1.3", "V1.4"],
                            "Dependency-Packages": [{
                                    "Module-Name": "Utilities",
                                    "Package-Name": "BlockingQueue",
                                    “location-”
                                    "Version": "V1.5"
                            }]
                    }],
                    "Test-Suits": ["TestDriver1.cs", "TestDriver2.cs"]
            }, {
                    "Module-Name": "Utilities",
                    "Packages": [{
                            "Package-Name": "BlockingQueue",
                            "Current-Version": "V1.2",
                            "ListOfVersions": ["V1.0", "V1.1"],
                            "Dependency-Packages": []
                    }]
            }]
    }, {
            "Sub-System": "Repository",
            "List-Of-Modules": [{
                    "Module-Name": ""

            }]
    }]
}
```

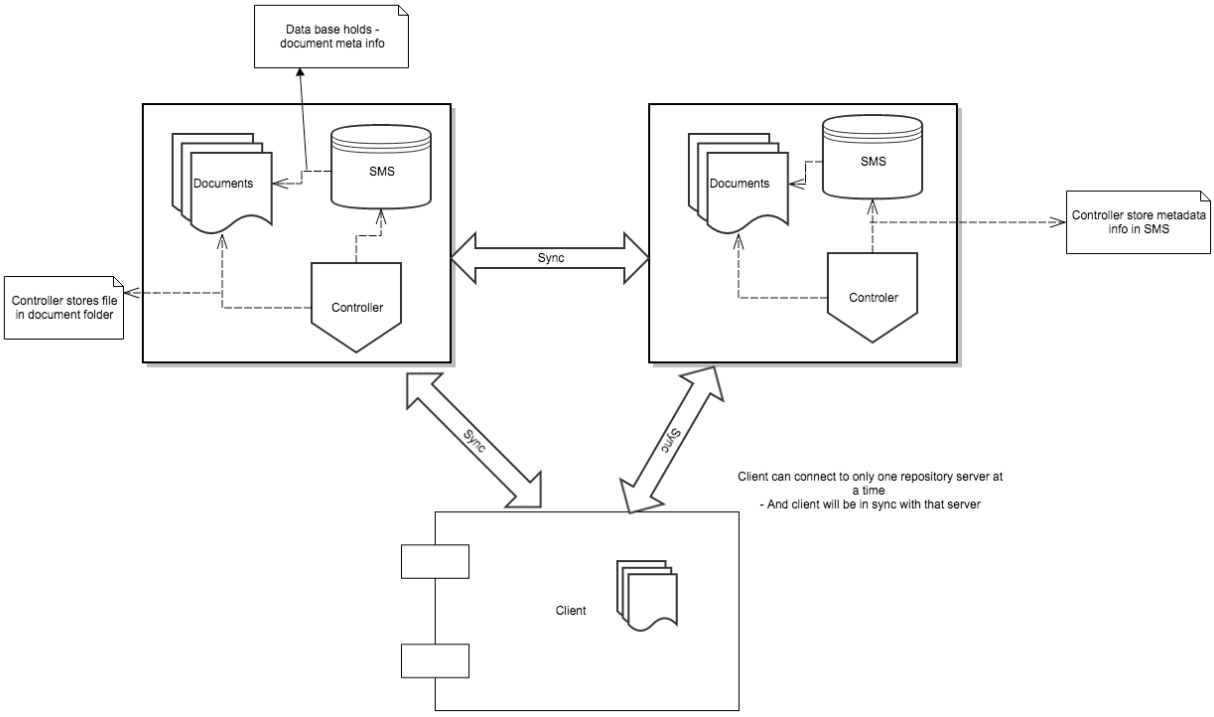
Repository server accepts two types of check-in, one is normal check-in and other is open check-in.

Check-In: Pushing your source code into repository by associating your work package number. Pushed package goes through all the steps given by version controller and store it under director with version # as name.

Open Check-In: when user is performing open check-in operation, that means file is not ready for working or not ready for use. In this case version controller place file in default directory without going through versioning process.

Repository server make use of Analysis management tool to check coding conventions of our code.

Repository server is designed as distributed server, when there is any catastrophic, there won’t be any loss of data. All the servers are in synchronization, and also client also have same copy of source code files locally.



Branch: Is a pointer to which code has to push.

Coming to baseline code management, repository maintains two branches of codes one is dev branch, and other is baseline (master) branch. Software developer check-in their code to dev branch. After continuous testing by Test Harness server, if all the test suits associated with the module are successfully executed, then repository server copies code present in dev branch to baseline branch.

2. Build Server

Software collaboration federation uses Build server is used executable images or libraries. Build Server based on requests it downloads all source code files, meta-data information from repository server and then start building executable images or libraries.

Build server follows following steps for building executable images or libraries.

1. Build server checks presence of requested version of modules in build server.
2. If it is not in build server at first it will start building all module dependencies and builds the requested module. So that there won't be any build errors while building.

Build server also maintains baseline code executable images or libraries. This helps managed services team in deploying project to development server, stage server and production servers.

Once Test harness completes its automated continuous test an integration, Test Harness stores test log results in repository server. If every test driver executes successfully then Repository server moves all source code in dev branch to baseline branch.

Build Server contains package manager which is used to download and install required packages that are needed for building images or libraries. Package Manager by default it take environment setup information from build configuration file. There is also option of manually configuring build configuration file based on requirement through administrative console.

3. Test Harness Server

Software collaboration federation uses Test Harness Server to achieve Continuous test and integration. Test Harness have batch job like a scheduler in Linux, which runs periodically to perform automated continuous integration testing. And also have option of accepting test request from clients to perform testing.

Test Harness stores test log results in repository server, and send those to requested peer. For security purpose test harness performs testing on test drivers in an isolated environment called child application domain.

Based on the request load Test Harness system spans instances of Test Harness servers. Load balancer present in system balance the request load by delegating carefully. If Test Harness system finds there are less number of requests, systems reduce few test harness server instances.

4. Collaboration Server

Generally, in initial stage of software project development in agile methodology, Software project manager gathers requirements from customer and document them. And sends copy of requirement document to customer through e-mails. If there are any changes in that document customer informs to manager. Manager modify document and send it back to customer again for review. This process keeps on going until all the requirements are fixed. The problem with the described method is it takes too much time, and communication between manager and customer.

Collaborative server provides collaborative environment which helps manager to collaborate with customer or any other stakeholders in organization regarding projects.

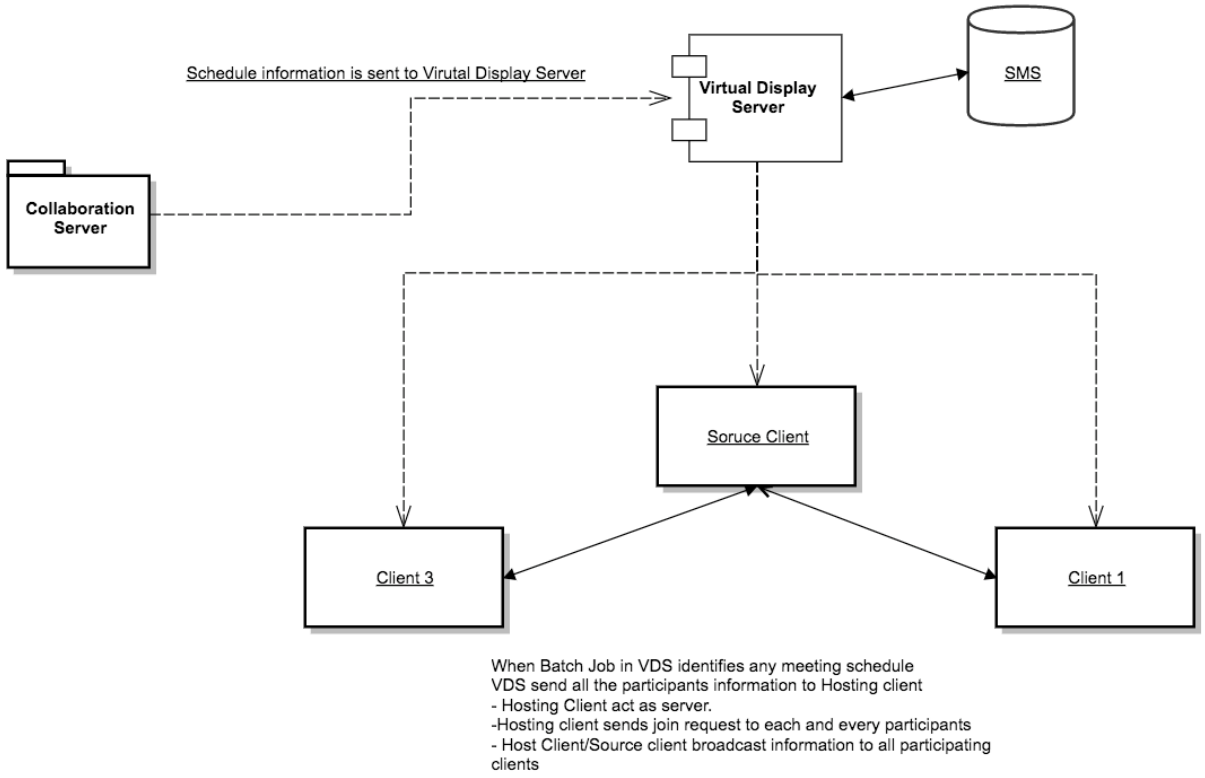
Primary function of collaborative server is to support project management activities like managing project team structure, describing and assigning work packages from requirements, scheduling meetings, sharing documents among project team etc.

5. Virtual Display Server

Virtual Display server is used for collaboration of remotely located project teams. VDS provides the following functionalities.

1. Sharing multimedia content.
2. Chatting with project teammates.

Virtual display server takes very less amount of burden by allowing clients to act as servers for broadcasting contents. When user schedules meeting using collaboration server tool, collaboration server sends schedule information to VDS and gives full control to handle scheduling. Following figure describes how VDS schedules meetings.



VDS stores scheduling information in mongo database using storage management package. Batch Job present in VDS server keeps on listening to all scheduling start times. One hour before start time, batch job will send connection information to client who is hosting meet. Hosting client application acts as server and sends join request to all the participants. Hosting client broadcast messages to all the clients who are joined.

If we design in above way, Virtual display can host multiple number of meetings.
Feasible if Source Client (hosting client) system requirements matches following requirements.

1. System should have 4gb ram.
2. System should have high internet speed.
3. System should support multimedia content.
4. Operating system can be CentOs, Mac, Windows Server 8, Windows 7

6. Client

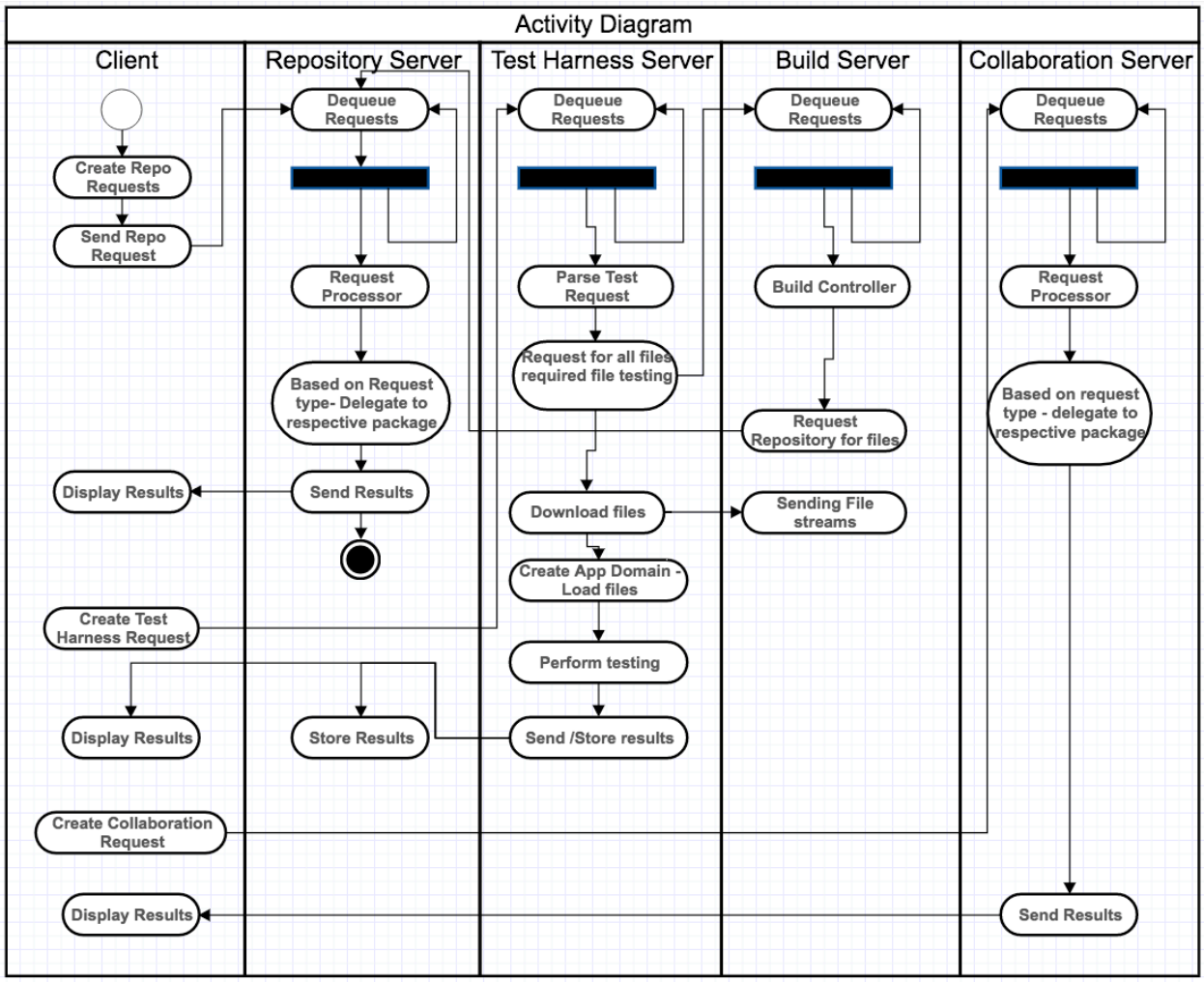
Software Collaboration Federation integrated all server tools in single client. User should be authenticated in order to access all the functions provided by SCF. Client GUI is developed using Windows Presentation Foundation. Designed in tabbed fashion so that it becomes very easy for user to navigate across all the SCF functions. There are four tabs. They are

1. Repository tab
2. Test Harness tab
3. Build Server tab
4. Collaboration tab

Each server has several functionalities all the functions are displayed as side menus under each tab.
GUI is designed by using MVVM pattern, which is model-view-view-model pattern makes us to write maintainable code.

2.3.2 Activity Diagram

Whole federation activity diagram is presented as swimlane activity diagram. Each column represents one server. This diagram shows basic interaction operations between clients and servers.



Detailed explanation of server's activity diagram will be able under respective servers. Above activity diagram represents following steps.

1. All servers are in running state.
2. Client creates respective message request and send to server by using proxy objects.
3. Server processes that request and send back results to client or other peers.

3 Repository Server

3.1 Summary

Primary functions of repository server are to store source code files, specification documents, metadata information in SMS and to manage baseline code repository.

3.2 Structure

3.2.1 User Interfaces

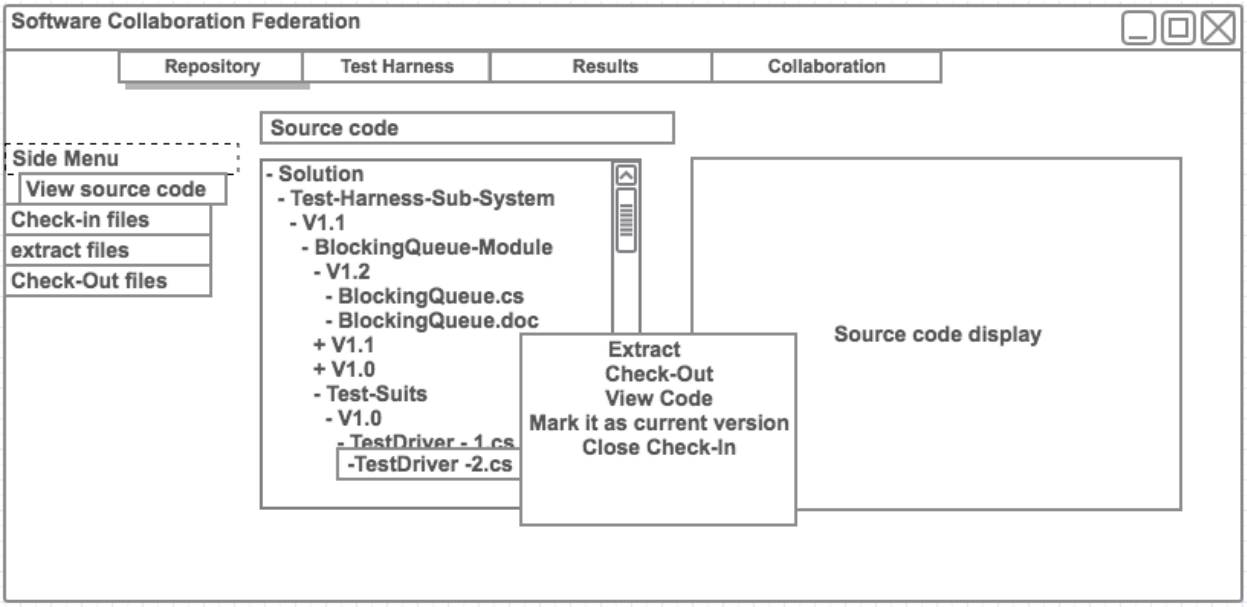
Software Collaboration Federation provides GUI for clients to interact with Repository server. Sub menus under repository tab shows all repository features that are available to client.

3.2.1.1 View metadata properties and source file.

GUI is using three column grid to lay out all the features, and stack panel in vertical orientation for side menu display (list view). When client clicks on View Source code submenu SCF client request repository server to give full information about metadata and source code. All metadata information is displayed using tree view. Client select on file to view content. Also when user right clicks on any selected file a popup appears that contains operations to Extract, Check-Out, View Code, Mark It as current version, Close Check-In menus.

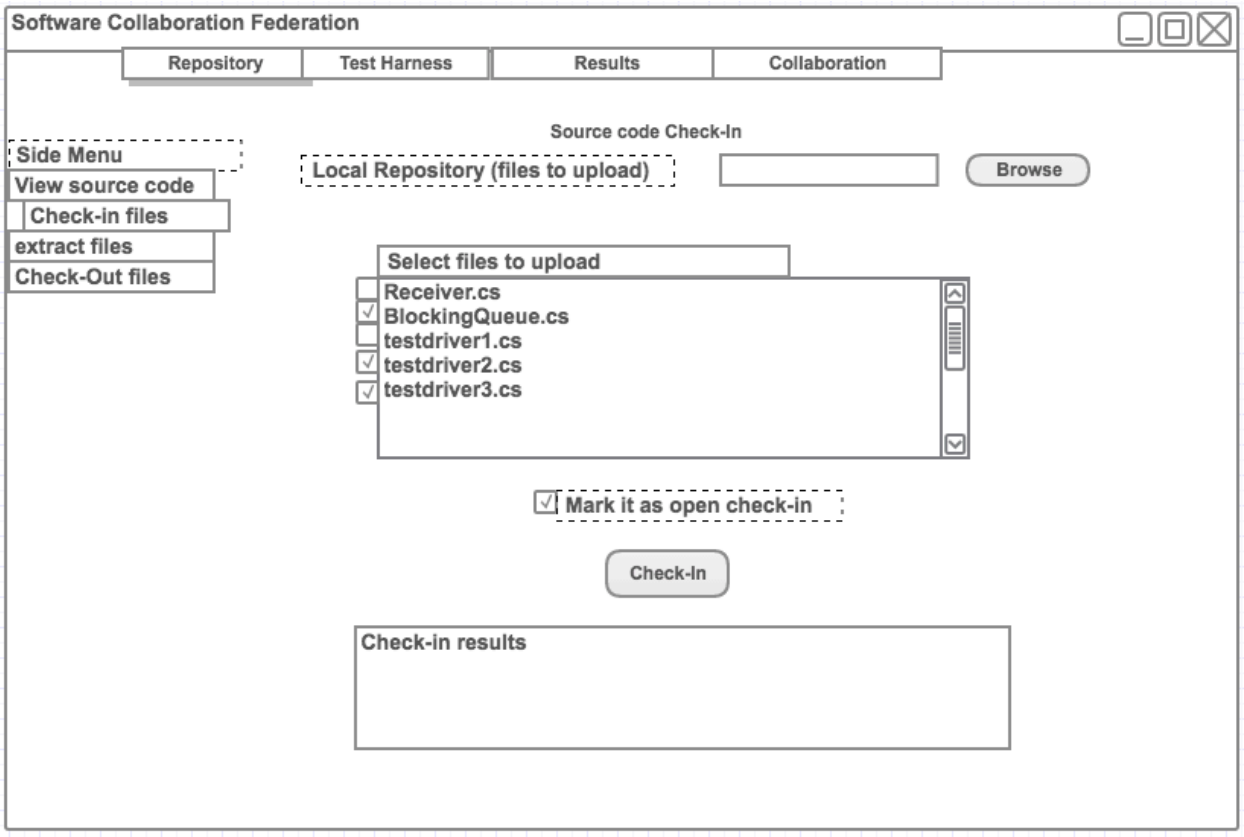
Operations:

1. Extract – Downloads selected file to local desktop.
2. Check-Out – To request for file modifications.
3. View code – Displays content of the file on right corner panel.
4. Mark it as current version – makes selected file as working version.
5. Close Check-In – Closes open check-in files.



3.2.1.2 Check-In View

This view allows client to check-in their source codes by browsing local directory and selecting files. There is also one option called mark it as open check-in, meaning files are not ready for use. Repository server places all open check-in files in default directory.



3.2.2 Repository Architecture

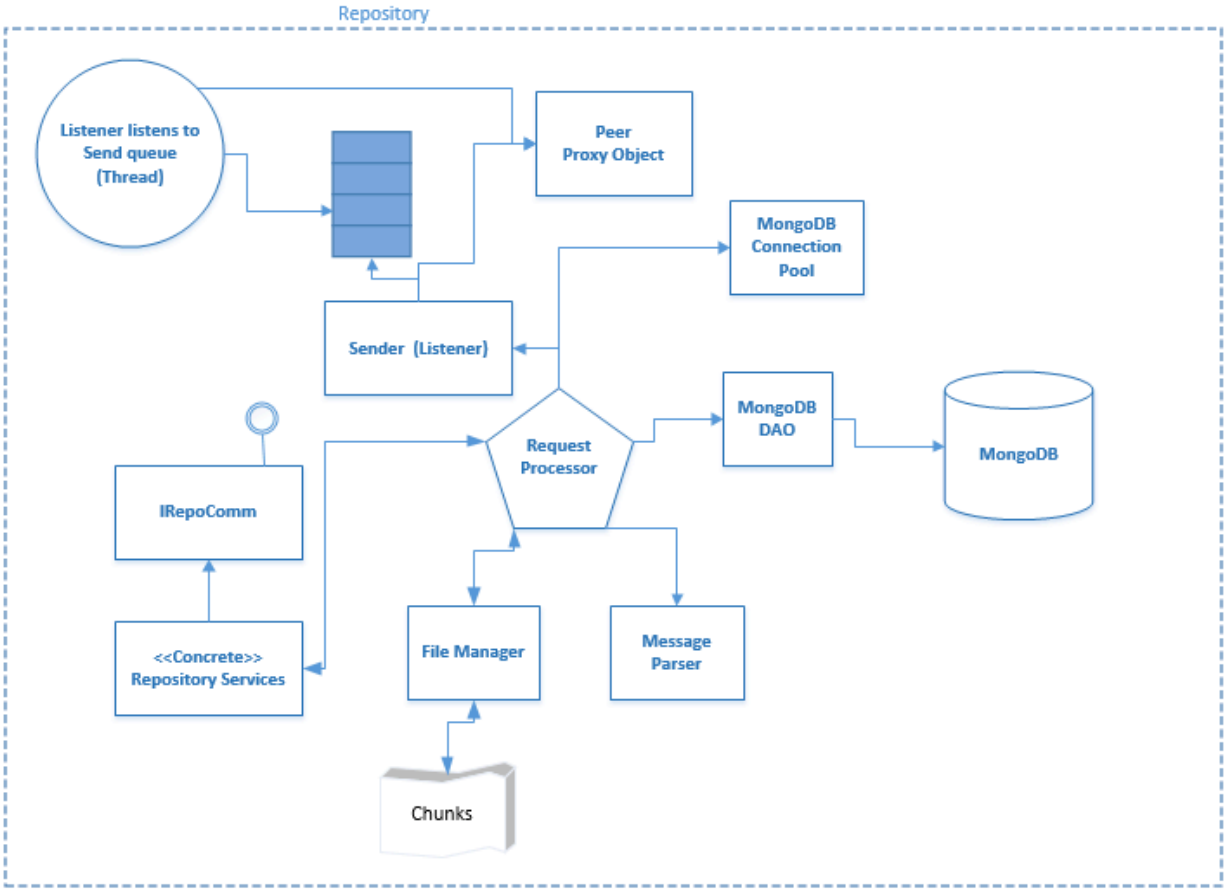
Repository server is designed as distributed server (i.e. clone of repository server is available on network and they are in synchronized). When there is any catastrophic, there won't be any loss of data. Clients interacts with repository server using SCF GUI. All the repository features are published as secure web services, so that organization can also able to design separate GUI to access all features if they are not comfortable with SCF GUI. Whole repository server is designed as one peer, acts as both server and client. Which contains receiver blocking queue and sender blocking queue. Since there will be different types of requests such as query request, test log storage request, view source code request etc. Request processor package in repository server uses front controller pattern to handle all types of requests and delegates to respective packages. Code snippet will be as follows.

```
try
 {
 //Front Controller pattern
 // this code at runtime creates object based on type of request
 ObjectHandle handle = Activator.CreateInstance("Repository",
"Repository.RequestProcessor." + act.type);
```

```
Object p = handle.Unwrap();
Type type = p.GetType();
MethodInfo method = type.GetMethod("processRequest");

method.Invoke(p, new object[] { act.body,act.from });
}catch(Exception e)
{
   Console.Write(e);
}
```

Repository server make use of Storage Management System services to store all information. SMS is nothing but Mongo database. Repository server also maintains set of MongoDB connection objects in a pool called connection pool, helps in managing connection instances.



3.2.3 Package diagrams and interactions

a. Blocking Queue:

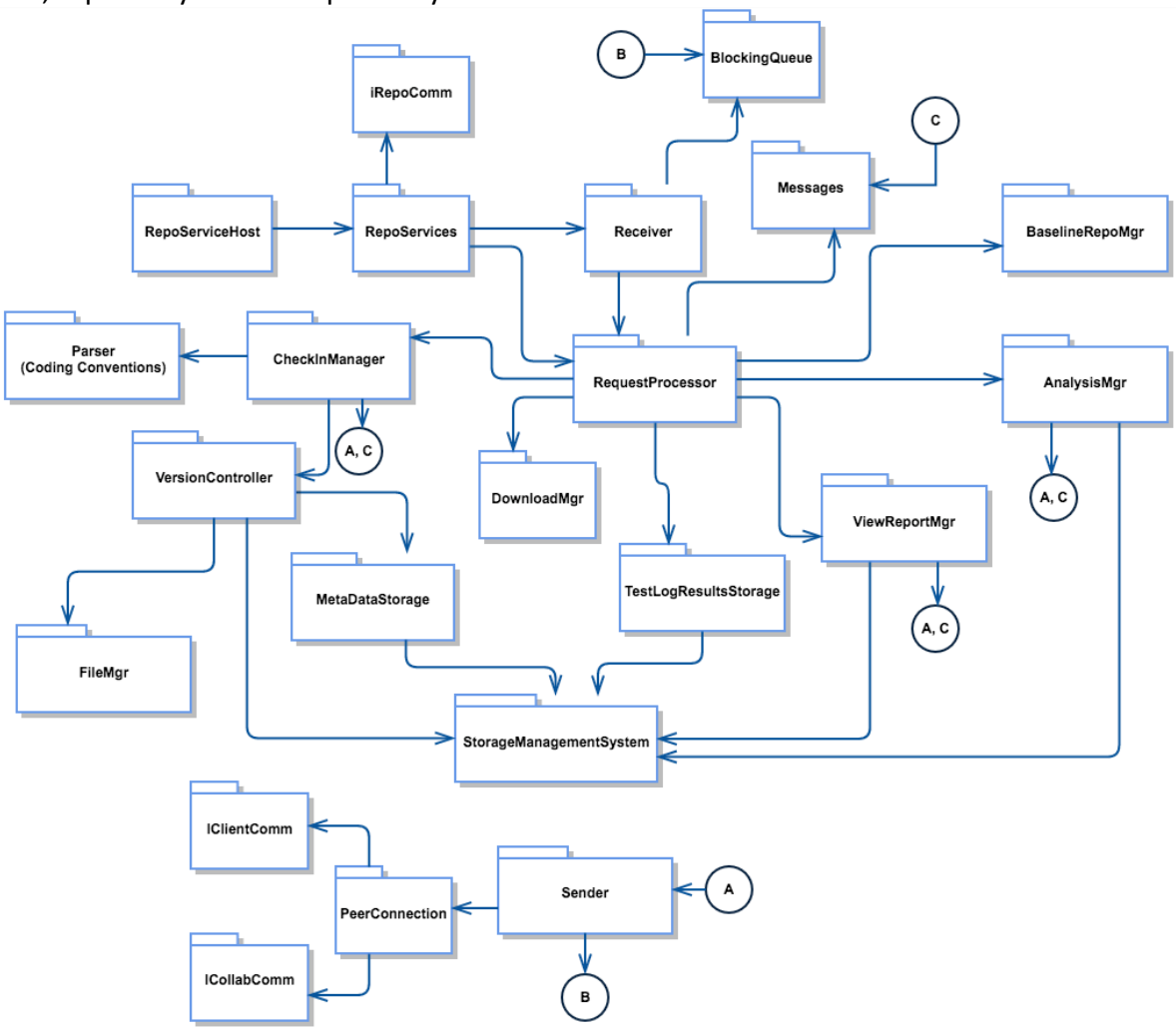
Blocking Queue package implements a generic thread safe queue. Used for communication between peers.

b. Peer Connection:

Peer Connection is a utility package to create channels and to create proxy objects for particular services using WCF facilities.
For creating channel, channel utility method accepts URL, binding type, and Service Contract Implementation type.
For creating proxy objects, proxy utility method accepts URL and binding type.

c. IClientComm, ICollabComm, IRepoComm:

IClientComm, ICollabComm, IRepoComm are service contracts for client, collaboration server, repository server respectively.



d. Repo Service Host:

Repo Service Host is used to create channel for Repository server. Other Peers (Clients) communicate with server by creating proxy object for this services.
Interactions: Repository service host uses Peer Connection package utility to create channel using URL (http://localhost:8090/RepoServices).

e. Repo Services:

Repository Services package implements IRepoComm service contract. This provides upload, download and post message operations.

Interactions: Post message operation enqueues all messages into blocking queue with the help of Receiver Package.

f. Check-In Manager

Following steps get processed when modified file is checked into repository.

1. Check-In Manager checks whether request is made by responsible individual or not.
2. If yes, with the help of parser, and version controller packages this package update metadata file in SMS with version.
3. New folder is created with version as name.
4. Stores files in that folder.

Following steps get processed when it is open check –in request.

1. Stores files in default folder. (when there is close check-in request, this file goes through all the pervious above steps)

Interactions: This package interacts with Parser, Version Controller and Sender packages.

g. Storage Management System:

This package contains all CURD (Create, Update, Retrieve, Delete) related operations, which are performed on Mongo database. Also contains connection pool module used to maintain MongoDB connection instances. Connection pool is nothing but maintaining list of connection instances as like thread pool. Following are some advantages of using connection pool.

a. Minimize creation effort of new connection instances
b. Effective usage of connection objects
c. Reduce of number of stale connection objects

h. Message:

This package is used to parse requested message for further processing. Package contains extension methods used to de-serialize request into respective objects.

i. Metadata storage:

This package is used for storing metadata information in mongo database with the help of Storage Management System. Metadata looks as follow.

```
{
    "Project-Solution": "SCF",
    "List-OF-Sub-Systems": [{
            "Sub-System": "Test Harness",
            "List-Of-Modules": [{
                    "Module-Name": "TestHarness",
                    "Packages": [{
                            "Package-Name": "Controller",
                            "Current-Version": "V1.5",
```

```
                        "ListOfVersions": ["V1.0", "V1.1", "V1.2", "V1.3", "V1.4"],
                        "Dependency-Packages": [{
                                "Module-Name": "Utilities",
                                "Package-Name": "BlockingQueue",
                                “location-”
                                "Version": "V1.5"
                        }]
                }],
                "Test-Suits": ["TestDriver1.cs", "TestDriver2.cs"]
        }, {
                "Module-Name": "Utilities",
                "Packages": [{
                        "Package-Name": "BlockingQueue",
                        "Current-Version": "V1.2",
                        "ListOfVersions": ["V1.0", "V1.1"],
                        "Dependency-Packages": []
                }]
        }]
    }, {
        "Sub-System": "Repository",
        "List-Of-Modules": [{
                "Module-Name": "repo"

        }]
    }]
}
```


j. Test Log Results storage:

This package is used for storing test log results sent by Test Harness server with the help of Storage management system. Test Log result looks a follow.


```
{
    "author": "Piduri Santhosh",
    "listOfTestDriverResult": [{
        "listOfTestCaseResults": [{
                "logs": "\u000a entering into TestDriver5 - > test() method\u000a divides by zero but does not try to catch exception\u000a Result :- False",
                "testName": "TestDriver.TestDriver5",
                "testResult": "False"
        }, {
                "logs": "\u000aResult :- True",
                "testName": "TestDriver.TestDriver11",
                "testResult": "True"
        }, {
```

```
                    "logs": "\u000a entering into TestDriver6 - > test() method\u000a calls
thread.abort() and tries to catch exception",
                    "testName": "TestDriver.TestDriver6",
                    "testResult": "False"
                }],
                "testDriverName": "TestDriver2, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null"
    }],
    "testRequestName": "Piduri Santhosh131201970635793646",
    "timeStamp": "2016-10-05 23:11:03"
}
```


k. Download Manager:

This package helps other peers in downloading files present in repository server.

l. View Report Manager:

Repository server uses this package to process view report requests from other peers. There are two types of report requests, one is simple test logs query and other is detailed test log results.

m. Analysis Manager:

Analysis manager gives code analysis tools which analysis driver source code to improve stability and reliability of driver. For example, Code Analysis tools for drivers by visual studio.

n. Version Controller:

Version controller update metadata information in storage management system with the help of Metadata Storage package by creating folder with version# as name and place files in that folder.

o. Request Processor:

This package handles all the incoming requests, and delegates to respective packages to perform operation.

p. Sender:

Repository server uses Sender package to send responses for all requests. This package includes thread pool, when application is started this module creates required number of threads to process responses that are enqueued in sender blocking queue by repository server. Advantages of using this module.

1. Thread creation time will get reduced.
2. Thread management will become easy.

q. Receiver:

Receiver package is used by repository services package. All the incoming requests to repository services gets enqueued into receiver blocking queue. Receiver package dequeues all the requests and starts processing by delegating to Request Processor package.

3.2.4 Activity diagrams

Following steps describes activity diagram:

1. All the repository services are exposed to other peers by creating channel by using service host package.
2. When clients are invoking post message operation contract using proxy object, following activities will take place.
    a. Post Message operation behavior enqueues messages into receiver blocking queue using receiver package.
    b. Receiver package dequeues each and every messages from queue and start processing them by delegating messages to request processor.
    c. Request processor identifies type of message, and delegates to respective package for further processing.
    d. There are six types of messages. They are
        i. Check-In Request
        ii. Query Test Result Request
        iii. Test Log results storage request
        iv. Metadata storage request
        v. Analysis manager request
        vi. View report
    e. Repository server try to get mongo database connection instance from connection pool and perform CURD operation on database.
    f. All responses for will be enqueued into sender blocking queue with the help of receiver package.
    g. Sender package dequeues and send responses to respective peers.
3. When clients invoking download operation contract with module name as parameter, download manager searches that file and returns that file in terms of streams or chunks.

3.3 Critical Issues

Issue #1

Security: In SCF, Developers will upload project files into the repository server to perform testing. How data integrity of those files will be maintained?

Solution: To maintain data integrity of files, there should some security while sending files from client to server. To achieve this secure communication between client and server, we are implementing WSHTTP binding provided by “Windows Communication Foundation”. By using this feature, files will get encrypted before sending from client to server.

Issue #2

Performance: Test Harness is using Mongo Database to store testing results. In order to communicate with database, application have to create mongo database connection instance. Since too many users use Test Harness, there will be too many connection instances. Obviously there will be performance issue when creating new connection instances when there are too many connections and communicating with database, how application is going to handle this scenario?

Solution: Application is handling the above scenario by using MongoDB Module, where pool of connection instances and pool configuration will be maintained. Following are some advantages with connection pool.

a. Reduces connection instance creation time
b. Reduces number of stale objects.

c. Efficient use of resources.

Issue#3

Concurrent access to files in Repository: may cause exception when there is opening conflicts
Solution: Try to access file for MAXIMUM number of times, report error if it exceeded

4 Build Server

4.1 Summary

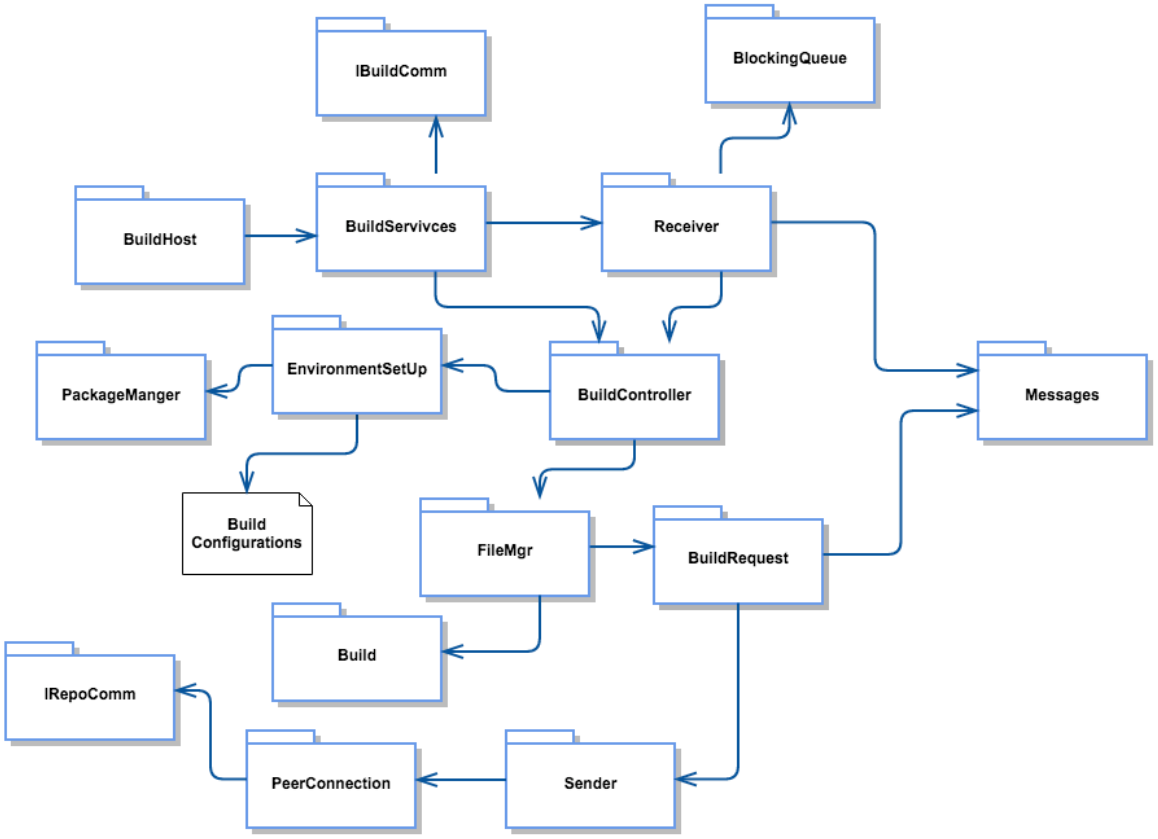
Build Server builds executable images or libraries based on requests from Test Harness.

4.2 Structure

4.2.1 User Interfaces

Build Server is providing administrative user interface, used by administrators to modify or change build configuration file.

4.2.2 Package diagrams and interactions



a. IRepoComm, IBuildComm

IRepoComm, IBuildComm are service contracts for Repository server, build server respectively.

b. Peer Connection:

Peer Connection is a utility package to create channels and to create proxy objects for particular services using WCF facilities.

For creating channel, channel utility method accepts URL, binding type and Service Contract Implementation type.
For creating proxy objects, proxy utility method accepts URL and binding type.

c. Blocking Queue:

Blocking Queue package implements a generic thread safe queue. Used for communication between peers.

d. FileMgr:

File manager is a utility package provides all the operations related to file such as opening, reading, writing, closing files etc.

e. Sender:

Build server uses Sender package to send responses for all requests. This package includes thread pool, when application is started this module creates required number of threads to process responses that are enqueued in sender blocking queue by Build server.
Advantages: Thread creation time will get reduced and Thread management will become easy.

f. Messages:

This package is used to parse requested message for further processing. Package contains extension methods used to de-serialize request into respective objects.

g. Build Controller:

Build controller acts as a controller for whole build server application part. Can be able to handle requests and delegates them to respective package for further processing.

h. Environment setup:

In reality, build server environment should be exactly same as dev, stage, and production environment in which project is going to deploy. If an organization dealing with multiple number of projects, then build server should be flexible enough to setup environment according to projects. For that this package parse configuration file and also invokes package manager if there is any need for new installations for the setup. Sample configuration file looks as follows.
Build Configuration file

```
<configuration>
    <environments>
        <dev>
            <required-packages>
                <package>BJson</package>
            </required-packages>
            <remove-packages>
                <package/>
            </remove-packages>
        </dev>
    </environments>
</configuration>
```

There is also option of sending configuration file from client, in that case environment setup package takes that configuration file as default one and will start processing it.

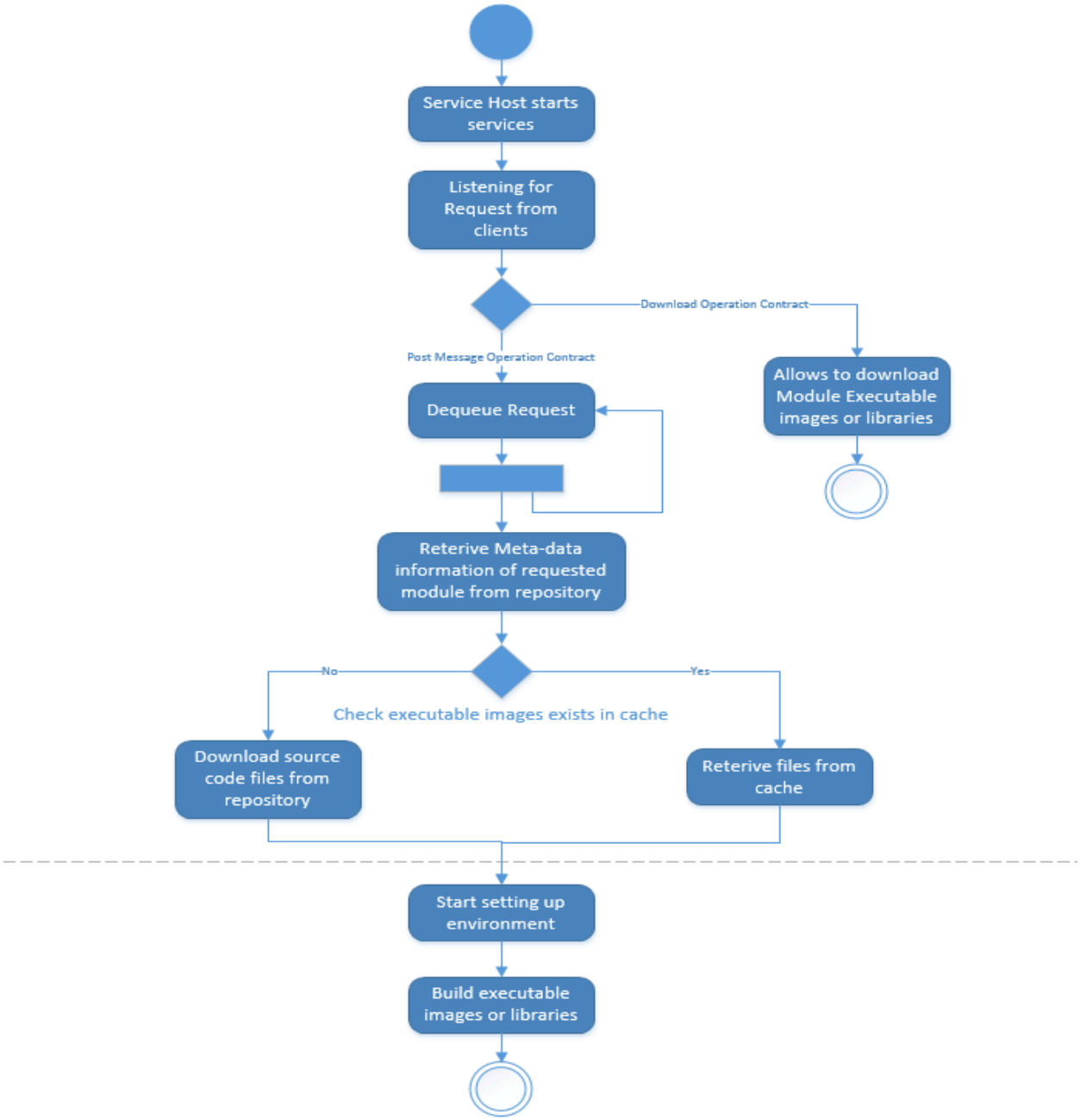
i. Package Manager:

Environment setup package invokes package manager if there is any need of package installation. Package Manager is like Nuget, Maven etc. This package installs required packages and un-installs remove packages in configuration file.

j. Build

This package is used for building executable images or libraries by Process class available in System.Diagonistics.Process package. This also can build CPP related files using Diagonistics managed code. So that Test Harness can able to perform testing both C#, Cpp related test drivers.

4.2.3 Activity diagrams



1. All the build services are exposed to other peers by creating channel by service host package.
2. When a client invokes post message operation contract, based on request type receiver package delegates to build controller.
3. If it is build request, build server retrieves metadata information from repository server
4. Before downloading required source code files, build server checks whether source code exists in cache or not. If not server downloads source code files from repository server. Otherwise build server uses source code files from cache for building executable images.
5. Environment package in build server parses configuration file to setup environment.

6. Build server starts executing dependency source code files before executing source code file. So that there won't be any build errors.
7. When client invokes download operation contract, build server allows client to download module executable image or library.

4.3 Critical Issues

Issue #1

How build server try to build executable images or libraries if C# code is dependent on some C++ libraries?

Solution: Build server should also able to compile and build CPP modules using Diagnostics namespace.

5 Test Harness Server

5.1 Summary

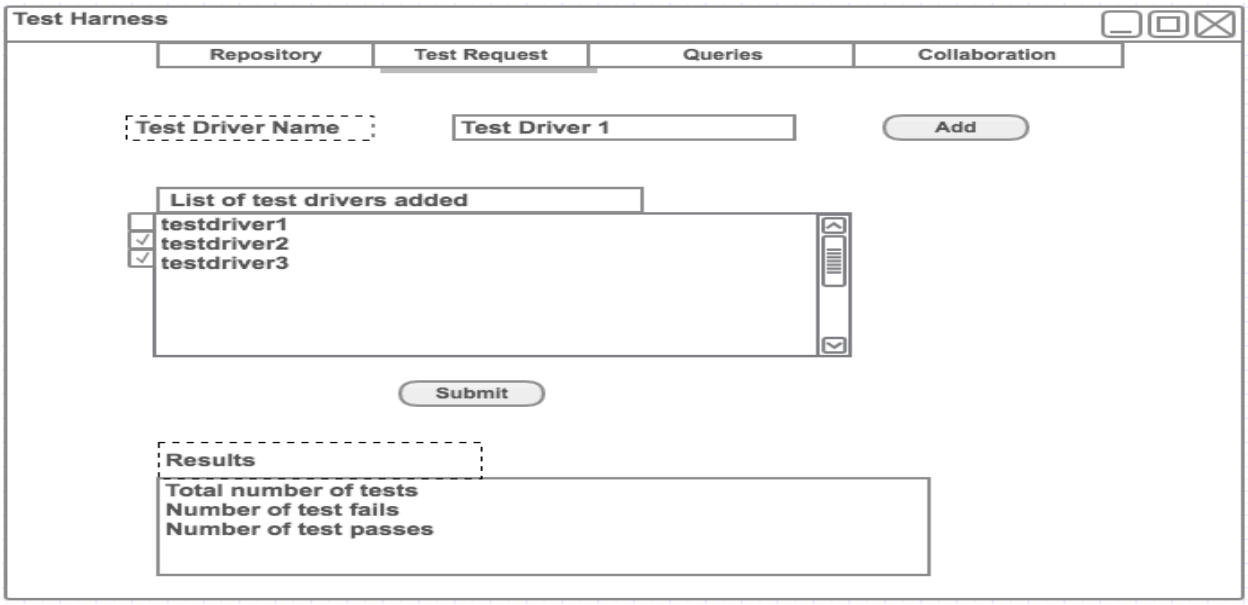
Generally, development of Large Software Systems needs a large set of developers and Quality Assurance members. They use various collaboration tools during their development and production release life cycles. They involve in a lot of collaborative activities to manage the versions of code changes and finalizing a production ready code by having various testing activities on master code branch. Some of environments that help in controlling versions of code are CVS, RTC, GitHub, etc. Before code going to production environment, members in Quality Assurance and Developer team starts conducting various types of test on master code repository. Based on results correctness they take necessary actions to make it stable. During this process if there is a testing application, which perform various types of test automatically, it will reduce a lot of time spent on manual effort.

The purpose of this document is to provide implementation details of automatic testing application for developers, which is called as **Test Harness**.

5.2 Structure

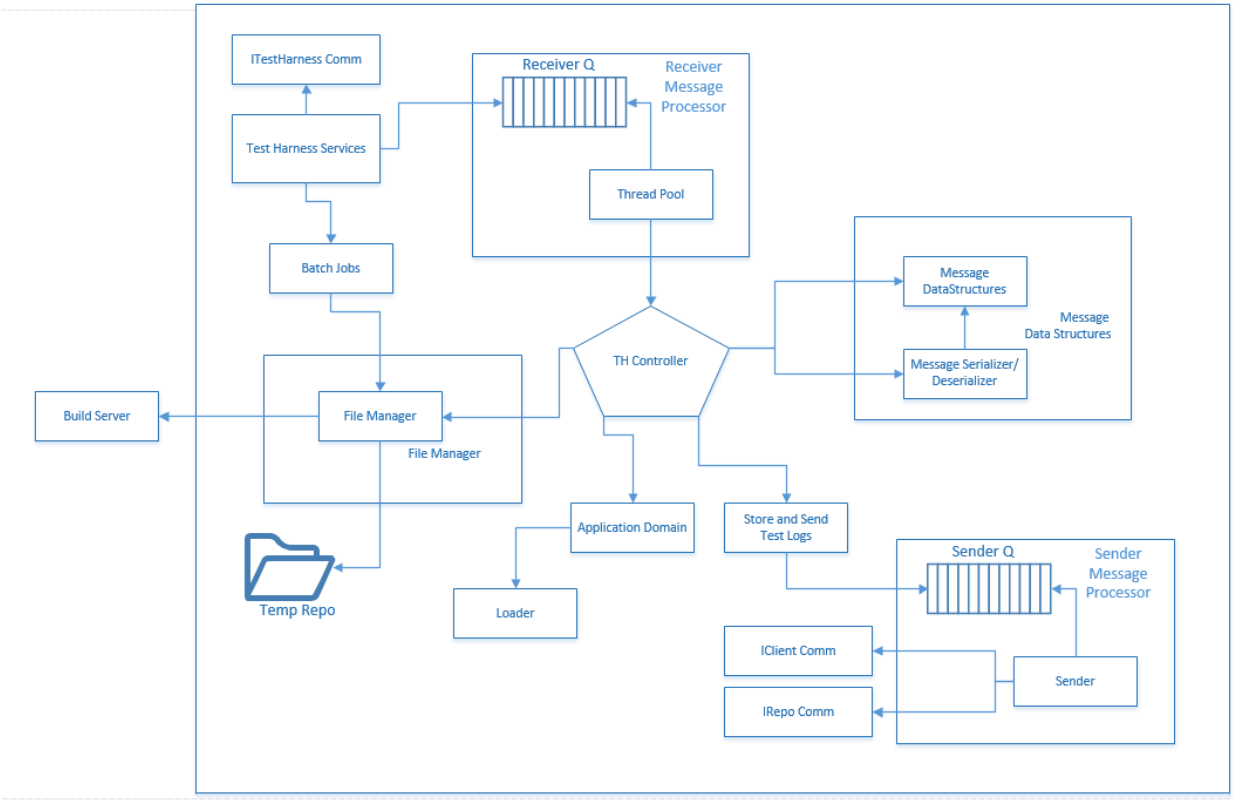
5.2.1 User Interfaces

Developers or QAs who want to perform testing have to provide test driver name, and click on Add button to add test driver to "list of test drivers added". Up on clicking Submit button test request will be sent to Test Harness server.
Testing results will get displayed in Results section.



5.2.2 Test Harness Architecture

Test Harness server is designed as distributed server. Server consists of two blocking queues, one for receiving messages and other is to send responses to peers. Uses reflection and Application domain concepts for identifying ITest Concreate implementation classes and to perform testing. To achieve automated testing, test harness implements batch jobs which runs periodically.

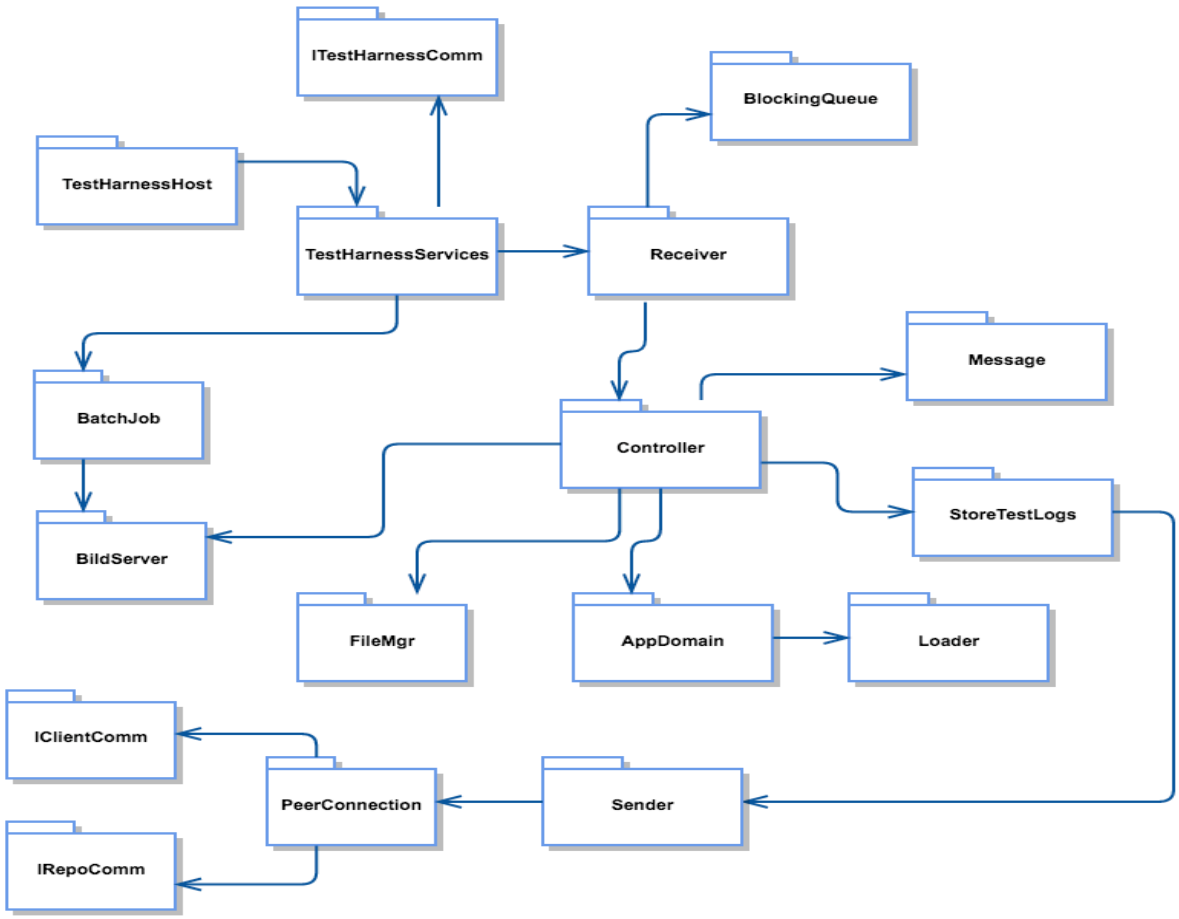


5.2.3 Package diagrams and interactions

a. FileMgr:
File manager is a utility package provides all the operations related to file operations such as opening, reading, writing, closing files etc.

b. Peer Connection:
Peer Connection is a utility package to create channels and to create proxy objects for particular services using WCF facilities.
For creating channel, channel utility method accepts URL, binding type, and Service Contract Implementation type.
For creating proxy objects, proxy utility method accepts URL, binding type.

c. Blocking Queue:
Blocking Queue package implements a generic thread safe queue. Used for communication between peers.



d. Application Domain:

Application Domain module creates child application domains of each test requests. And also used to create child application domain proxy object. Following are some advantages we will get by running DLL files in AppDomain.

- AppDomain creates isolation process for each DLL so that it will continue processing even when there is failure in code execution.
- Child processes are created with secure environment by AppDomain.

Interaction with other modules:

Application domain module will get called by Test Harness controller to create child application domain.

e. Loader:

This module loads all the execution libraries required for the test request and also it starts running test drivers with help of reflection feature provided by C#. Net

f. Store Test logs:

All performing all the test on test request using Loader package, results will get stored in software management system present in repository server. And also sent to client who requested to testing.

g. Sender:

Test Harness server uses Sender package to send responses for all requests. This package includes thread pool, when application is started this module creates required number of threads to process responses that are enqueued in sender blocking queue by Test Harness server.

Advantages: Thread creation time will get reduced and Thread management will become easy.

h. Receiver:

Receiver package is used by test harness services package. All the incoming requests to test harness services gets enqueued into receiver blocking queue. Receiver package dequeues all the requests and starts processing by delegating to Controller package

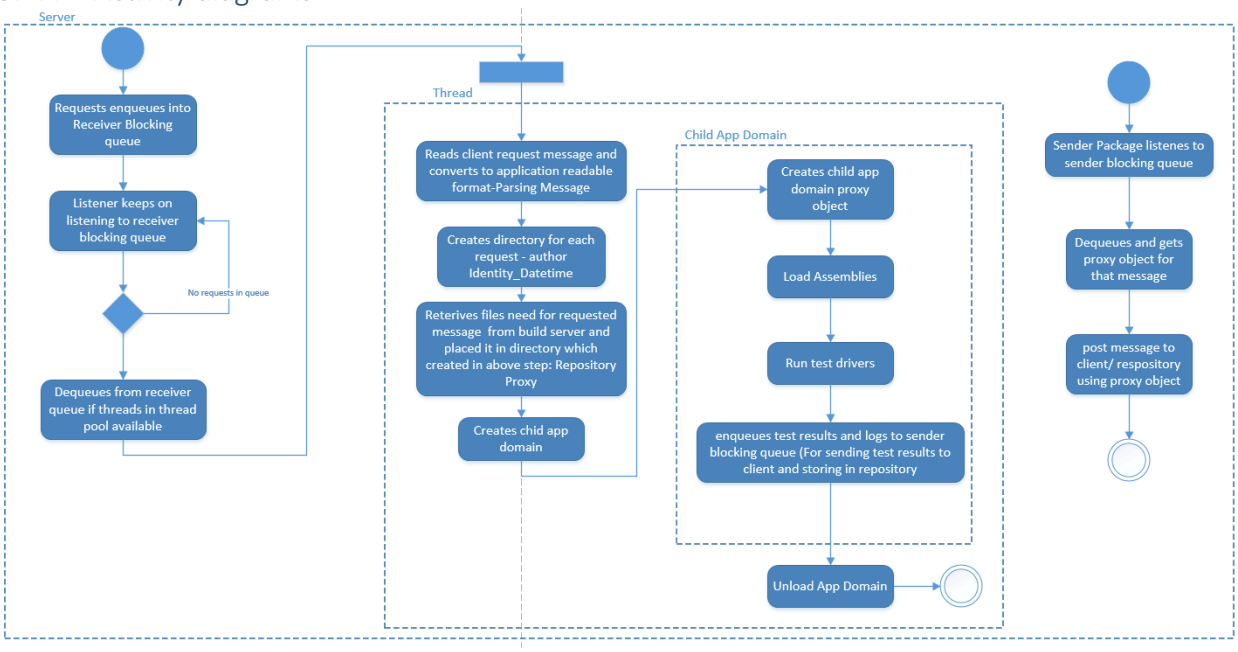
i. Batch Job:

In order to achieve continuous test and integration, test harness performs integration testing at the end to the day. Batch job is like a scheduler in Linux environment, it keeps on listening to time and starts testing all the test drivers automatically. To get executable images, batch job communicates with build server package.

j. Message:

This package is used to parse requested message for further processing. Package contains extension methods used to de-serialize request into respective objects.

5.2.4 Activity diagrams



Following steps describes above activity diagram.

1. All the client requests are enqueued into receiver blocking queue by Test Harness Service module.
2. Message handling listener dequeues each request and allocate a thread from thread pool. If threads are busy in thread pool listener won't dequeue requests and it waits for a thread.
3. Message parser is used to parse request messages and convert to application readable format.
4. Creates separate directory inside application base path for each test requests. Directory name will be author followed by timestamp.
5. File Manager download files that are required for a request into directory that was created. These files are downloaded from repository server using proxy object.
6. Controller creates child application domain to perform testing.
7. Creates child application domain proxy object.
8. Child application domain load all assemblies required for the request.
9. Run test drivers using reflection concepts.
10. Child application domain send results to sender blocking queue, so message handling listener dequeues and sends to respective clients
11. Finally, all the test results are stored in MongoDB in repository server and logs are logged in Test Harness server.
12. Main application domain will unload child application domain and deletes the directory which is used by child application domain.

How Test Harness server sends request results to other peers?

Following steps describes sending messages to other peers from test harness.

1. Message Handling listener listens to send blocking queue.
2. Listener dequeues the messages and get proxy object for that message.
3. Test Harness posts message to client using that proxy object.

5.3 Critical Issues

Issue #1

How test harness able to perform testing on C# code which is dependent on CPP libraries?

Solution: There should be some adapter, which uses extern modifier with DllImport attribute.

Issue #2

Test Harness has to support multiple client requests; Application creates multiple threads in order to process client requests. Though CLR is providing thread pool, it will be difficult to know how many threads are running at a time and maintaining them. Most of the times Test Harness will be very busy with many client requests, if we allow application to create threads by itself using CLR there will performance issues. For instance, If Test Harness take 5 seconds to process a simple request, among those 5 seconds application will take 3secs to create thread. So how can we easily keep track, maintain threads and increase performance of Test Harness considering the above situation?

Solution: Test Harness also should maintain thread pool. When an application starts, it will also start a set of threads and keep it in a pool. Threads in a thread pool will be waiting or sleeping until requests comes to blocking queue. Listener notifies threads when a request is enqueued in the blocking queue. By maintaining thread pool in this application, thread creation time will be reduced, which improves performance of an application.

Thread Pool code will be as follows

```
public THConsumerThreadPool(int noOfThreads)
            {

                //this.blockingQueue = blockingQueue;
                threads = new ThreadPool[noOfThreads];
                for (int i = 0; i < threads.Length; i++)
                {
                    threads[i] = new ThreadPool();//blockingQueue);
                    ThreadStart threadStart = threads[i].threadFunc;
                    Thread thread = new Thread(threadStart);
                    thread.Start();

                }
    }

public void threadFunc()
            {
                while (true)
                {

                    lock (locker_)
                    {
                        while (blockingQueue.size() == 0)
                        {
                            Console.Write("\n waiting");
                            Monitor.Wait(locker_);
```

```
                        }
                        Action act = blockingQueue.deQ();
                        if (act != null)
                        {
                            Console.Write("\n          Thread"          +
Thread.CurrentThread.ManagedThreadId);
                            act.Invoke();
                        }
                    }
                   // Thread.Sleep(5000);
                }
            }
```

Issue #3

How Test Harness application handles when multiple requests want to make use of same DLL file?

Solution: For each test request, application will create and load all files in a directory which is particularly created for that request. And then child application domain will run tests on the files in that directory. In this way application will maintain a separate copy of DLL files for each requests. By this way multiple requests that are referring to the same DLL file can be handled.

6 Collaboration Server

6.1 Summary

Primary function of collaborative server is to support project management activities like managing project team structure, describing and assigning work packages from requirements, scheduling meetings, sharing documents among project team etc.

6.2 Structure

6.2.1 User Interfaces

6.2.1.1 Project status view

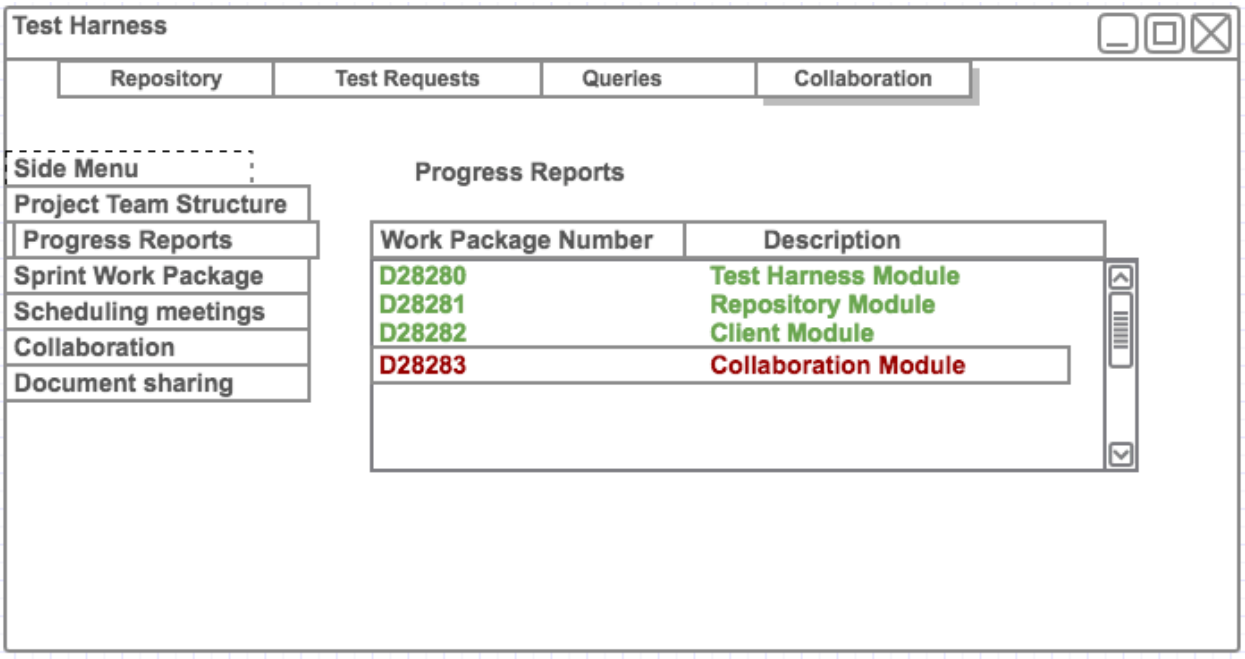
All the collaboration server functions are available under collaboration tab in software collaboration federation client GUI. Collaboration server function are available as side menus. There are six functions, they are

1. Project Team Structure – This allows software project manager to create team structure by providing each and every team member details.
2. Progress Reports – This view show progress or status of the project.
3. Spring work package – assigning work packages to responsible individuals.
4. Scheduling meetings
5. Document sharing
6. Collaboration settings menu.

Following GUI shows progress reports view. Each work package is assigned with work package number and associated with brief description.

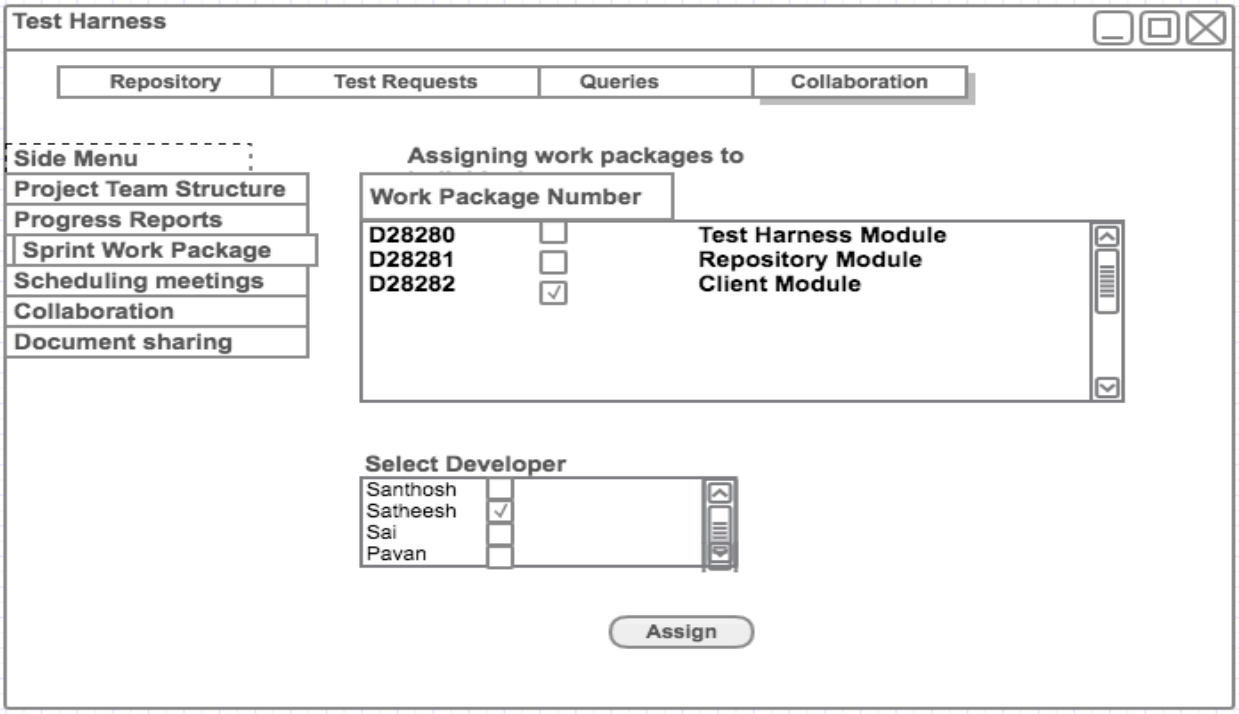
Green – completed work packages

Red – In progress work packages.



6.2.1.2 Assigning work packages to responsible individual view

Software Architect go through all the requirements and forms into stories. If story is to large architect breaks story into smaller modules and make them as work packages. Work packages are nothing but description of unit independent module work. Software architect will assign work packages to responsible individual by using following view in SCF GUI.



6.2.2 Package diagrams and interactions

a. Peer Connection:

Peer Connection is a utility package to create channels and to create proxy objects for particular services using WCF facilities.
For creating channel, channel utility method accepts URL, binding type, and Service Contract Implementation type.
For creating proxy objects, proxy utility method accepts URL, binding type.

b. Blocking Queue:

Blocking Queue package implements a generic thread safe queue. Used for communication between peers.

c. IClientComm, IRepoComm, ICollabComm

IClientComm, IRepoComm, ICollabComm are service contracts of client, repository, collaboration server respectively.

d. Receiver:

Receiver package is used by Collaboration services package. All the incoming requests to collaboration services gets enqueued into receiver blocking queue. Receiver package dequeues all the requests and starts processing by delegating to Request processor package

e. Messages:

This package is used to parse requested message for further processing. Package contains extension methods used to de-serialize request into respective objects.

f. Progress Report Mgr:

This package is used to get to know the status of the project, simply by getting all the test related logs, and information about work package status.

g. Sprint Work Mgr:

Sprint Work manager is used for store and retrieve work package stores, and for storing who is responsible for work package. Stores all information in mongo database with the help of storage management system.

h. Schedule Mgr

Schedule Manager package is used for scheduling customer meetings, project review meetings. Sends notifications for scheduling meeting before the meeting date. Displays schedule timings.
If request is for virtual display scheduler, information is sent to virtual display server.

i. Specification Mgr

Create or edit specification documents, and to share documents to project team members.

j. Team Mgr

When request type is related to team management, Request Processor package delegates messages to this package. Team Mgr has following functions

1. Stores team details in mongo database with the help of Storage Management System.
2. Gives options for creating or editing team structure.
3. To retrieve information about any team member.

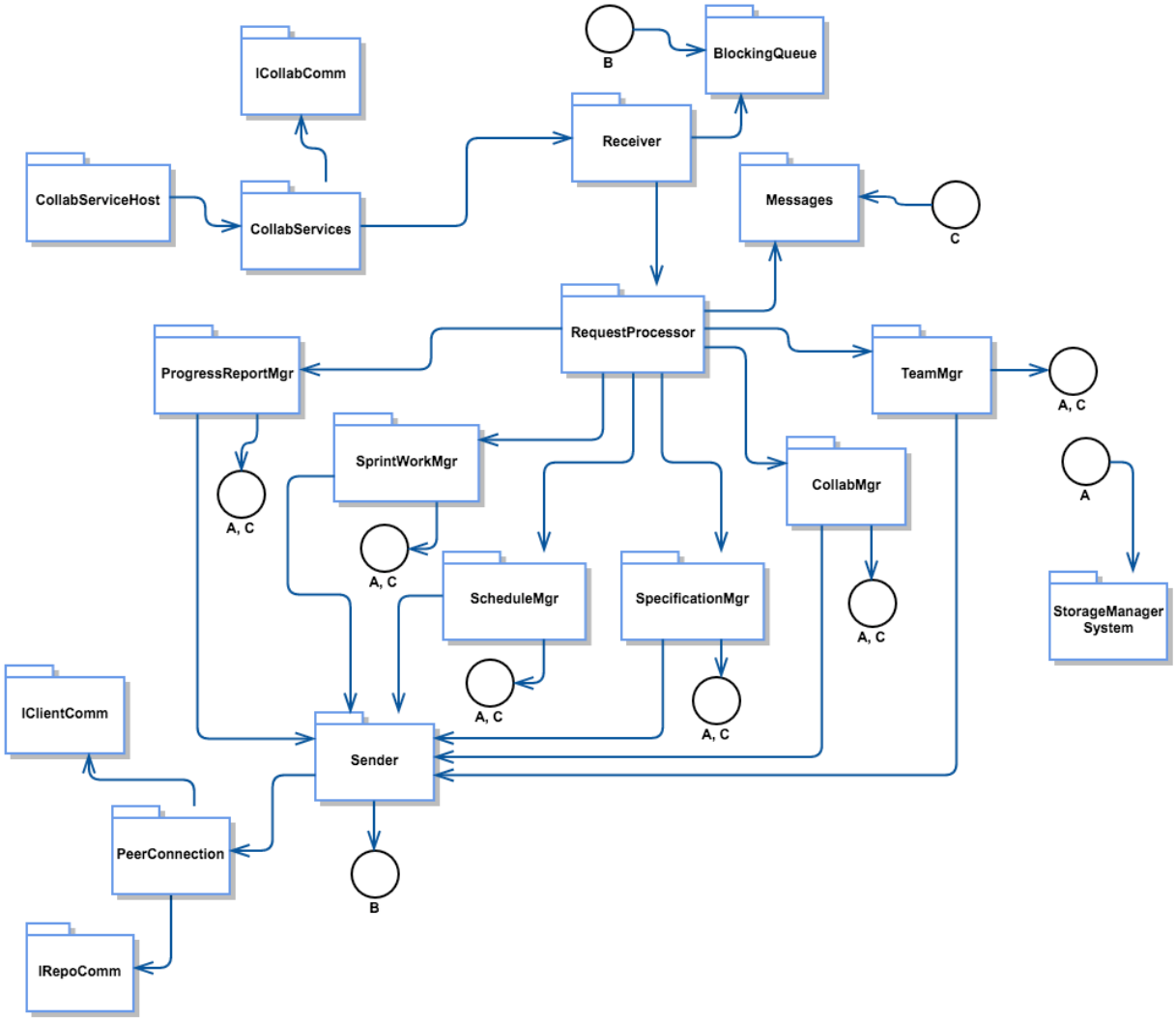
k. Storage Management system

This package contains all CURD (Create, Update, Retrieve, Delete) related operations, which are performed on Mongo database. Also contains connection pool module used to maintain MongoDB connection instances. Connection pool is nothing but maintaining list of connection instances as like thread pool. Following are some advantages of using connection pool.

a. Minimize creation of new connection instances
b. Effective usage of connection objects
c. Reduce of number of stale connection objects

l. Request Processor

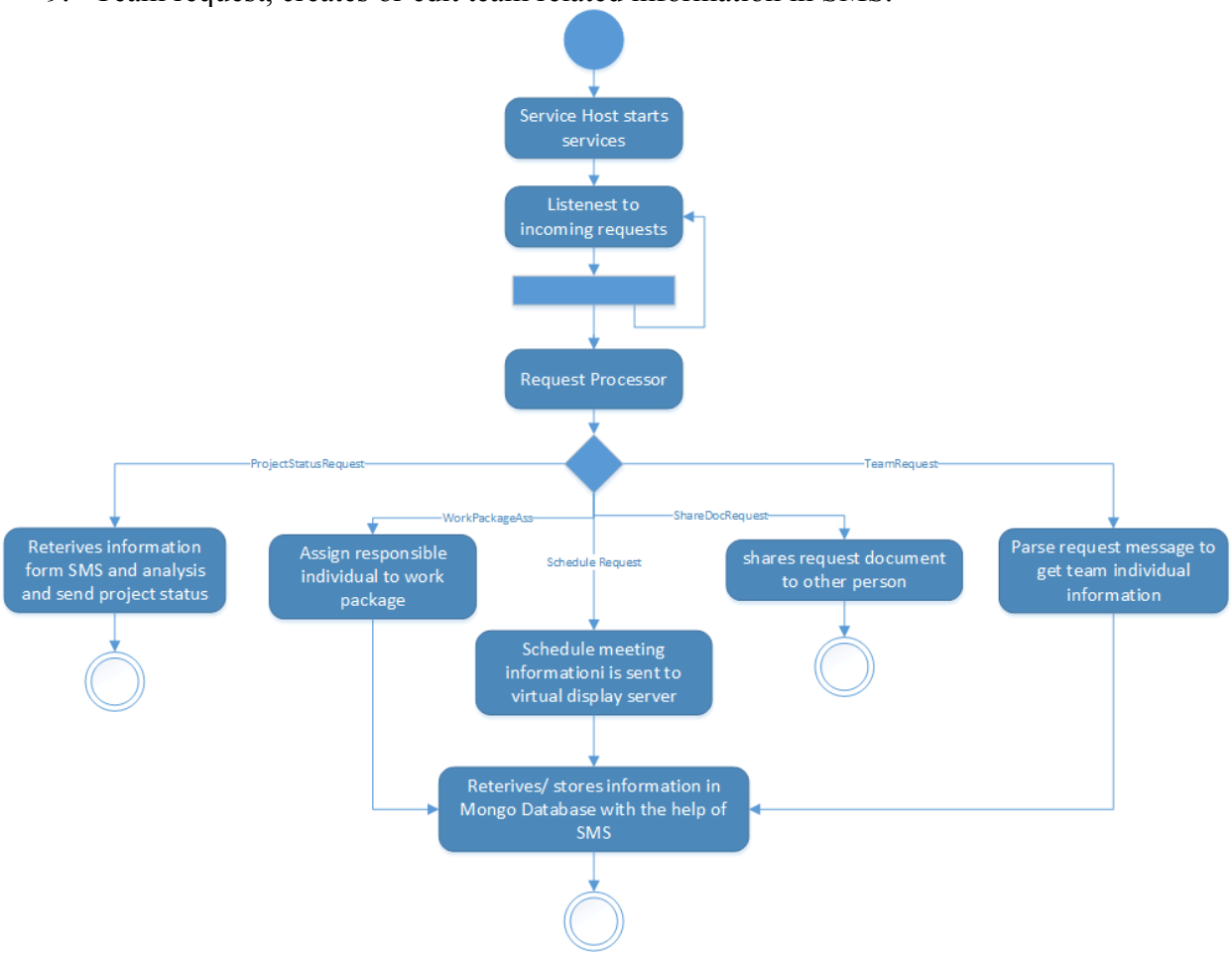
This package delegates all messages to respective packages for further processing.



6.2.3 Activity diagrams

Following steps describes activity diagram.

1. All Collaboration services are exposed to other peers by creating channel by service host package
2. Server listens to incoming request, if there is any request service enqueues that request into receiver blocking queue.
3. Receiver packages dequeues messages from blocking queue and delegate to request processor.
4. Request processor gets request type and call request related package for further processing.
5. If request type is project status, project status related packages retrieves information from SMS, Process them and send it to client.
6. Work package related request, insert or update related information in SMS
7. Scheduling meetings, sends information to VD server.
8. Share document request, allows user to share document with others.
9. Team request, creates or edit team related information in SMS.



6.3 Critical Issues

Issue #1

Concurrent access to files in Repository: may cause exception when there is opening conflicts

Solution: Try to access file for MAXIMUM number of times, report error if it exceeded.

7 Virtual Display Server and Client

7.1 Summary

While developing software projects in large multinational organization, project teams are located remotely. For example, there will be offshore team and onshore team. At the end of the day every team member should demonstrate his work by sharing his desktop screen to offshore or onshore team members.

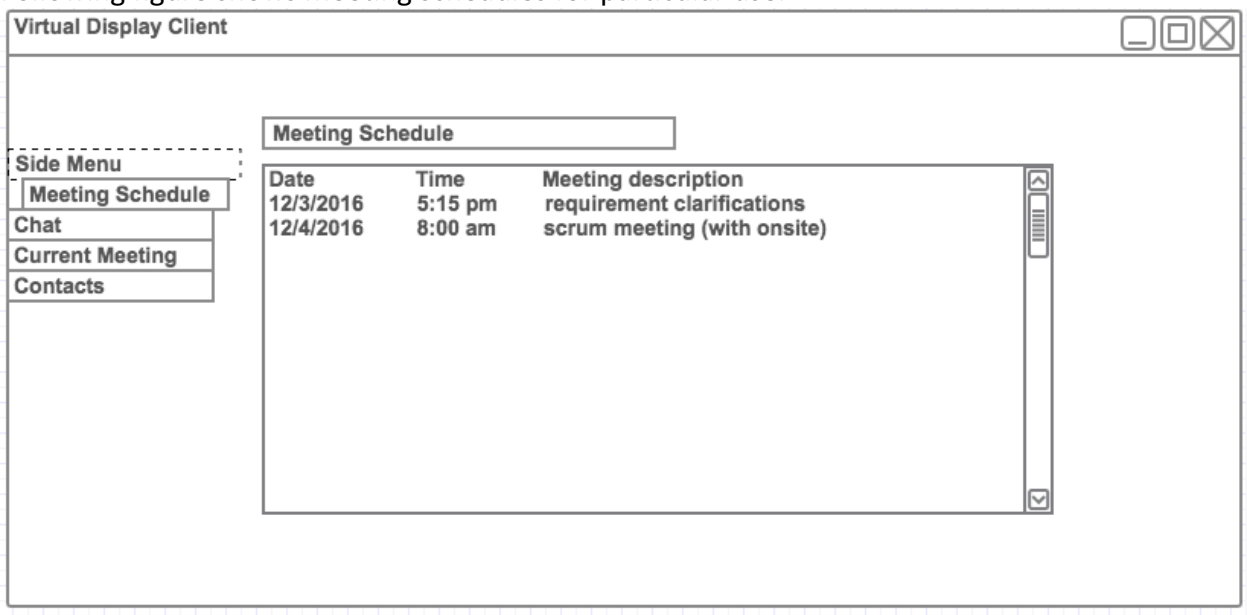
There should some application which helps in sharing media content collaboratively. Virtual Display Server provides collaborative environment to share media content, chatting etc.

7.2 Structure

7.2.1 User Interfaces

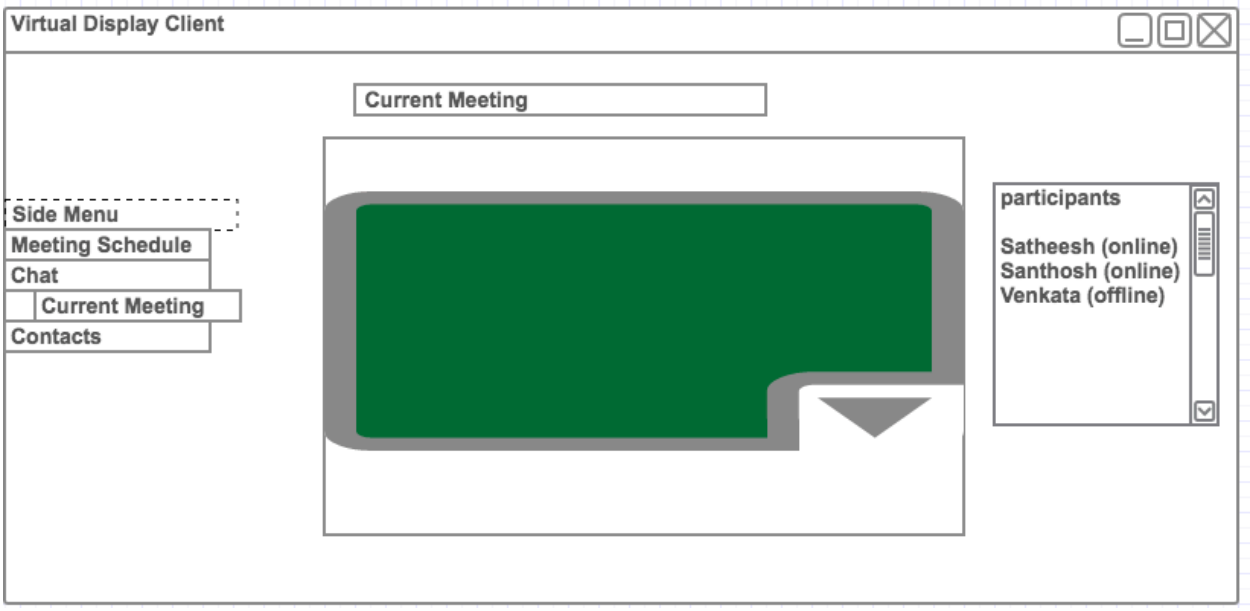
Virtual Display provides GUI to access all the functionalities of Virtual display. Using this GUI client can able to chat, view scheduled meetings, current meetings, he can search for contacts. User can access all functions from side menu.

Following figure shows meeting schedules for particular user

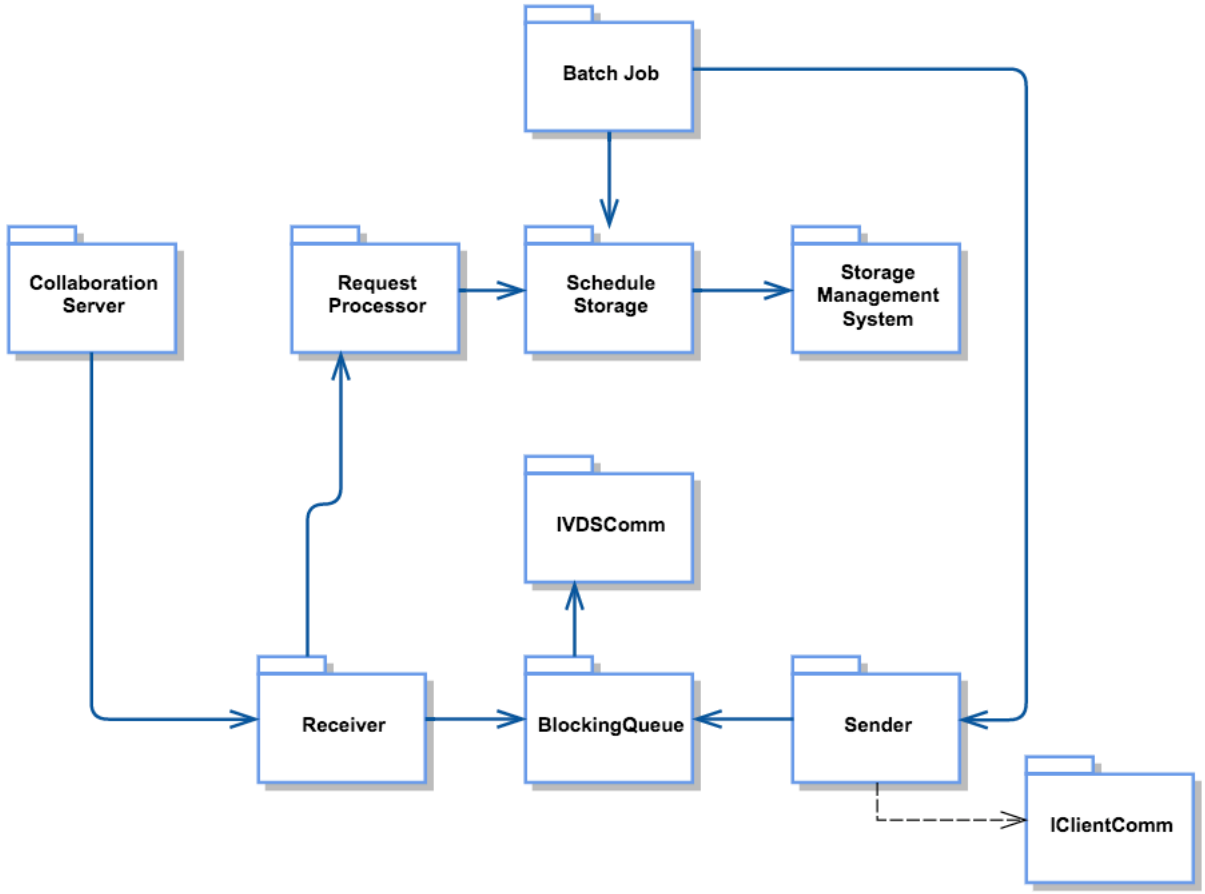


Following figure shows how user can able to chat with his colleagues.

Following figure shows how user can view shared screen of hosting client.



7.2.2 Package diagram and interactions



a. Receiver:

Receiver package is used by virtual display services package. All the incoming requests to services gets enqueued into receiver blocking queue. Receiver package dequeues all the requests and starts processing by delegating to Request Processor package.

b. Sender:

VD server uses Sender package to send responses for all requests. This package includes thread pool, when application is started this module creates required number of threads to process responses that are enqueued in sender blocking queue by VD server. Advantages of using this module.

1. Thread creation time will get reduced.
2. Thread management will become easy.

c. Blocking Queue:

Blocking Queue package implements a generic thread safe queue. Used for communication between peers.

d. Request Processor:

This package handles all the incoming requests, and delegates to respective packages to perform operation.

e. Batch Job:

Batch job in VDS is like scheduler in Linux, runs periodically, keeps on listening to all scheduling start times. One hour before start time, batch job will send connection information to client who is hosting meet.

f. Storage Management System:

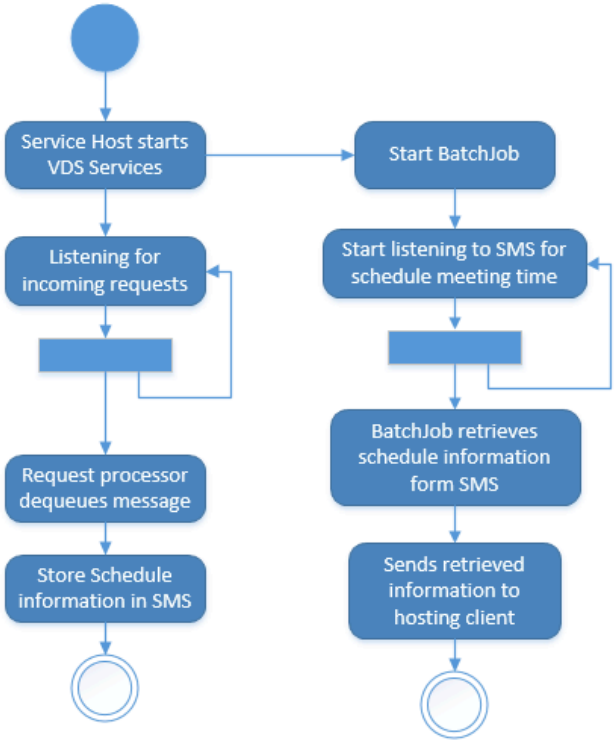
This package contains all CURD (Create, Update, Retrieve, Delete) related operations, which are performed on Mongo database. Also contains connection pool module used to maintain MongoDB connection instances. Connection pool is nothing but maintaining list of connection instances as like thread pool. Following are some advantages of using connection pool.

a. Minimize creation effort of new connection instances
b. Effective usage of connection objects
c. Reduce of number of stale connection objects

g. Collaboration Server

Collaboration server is used for scheduling multimedia meetings.

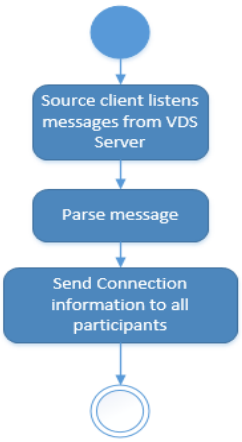
7.2.3 Activity Diagrams



Following activities describes above activity diagram.

1. All Virtual Display services are exposed to other peers by creating channel by service host package, and start batch job.
2. Services keeps on listening for incoming requests, once received request is enqueued into receiver blocking queue.
3. Receiver package dequeues form blocking queue and start processing.
4. If request is of schedule meet type, then request processor stores schedule information in storage management system.

5. Batch job that is started, keeps on listening for start time of schedules in SMS.
6. Batch job retrieves that schedule information and send to hosting client.



7. Hosting client listens to vds server for message. When it receives schedule information message.
8. Hosting client parses the message.
9. Send connection information to all the participants.

7.3 Critical Issues

Issue #1

How Virtual display server handles delay in transferring multimedia content?

Solution: Virtual display makes host client as server, so in this way virtual display server handles delay in transferring content. Host client have only few load so there won't be any delay in transferring content.

8 Client

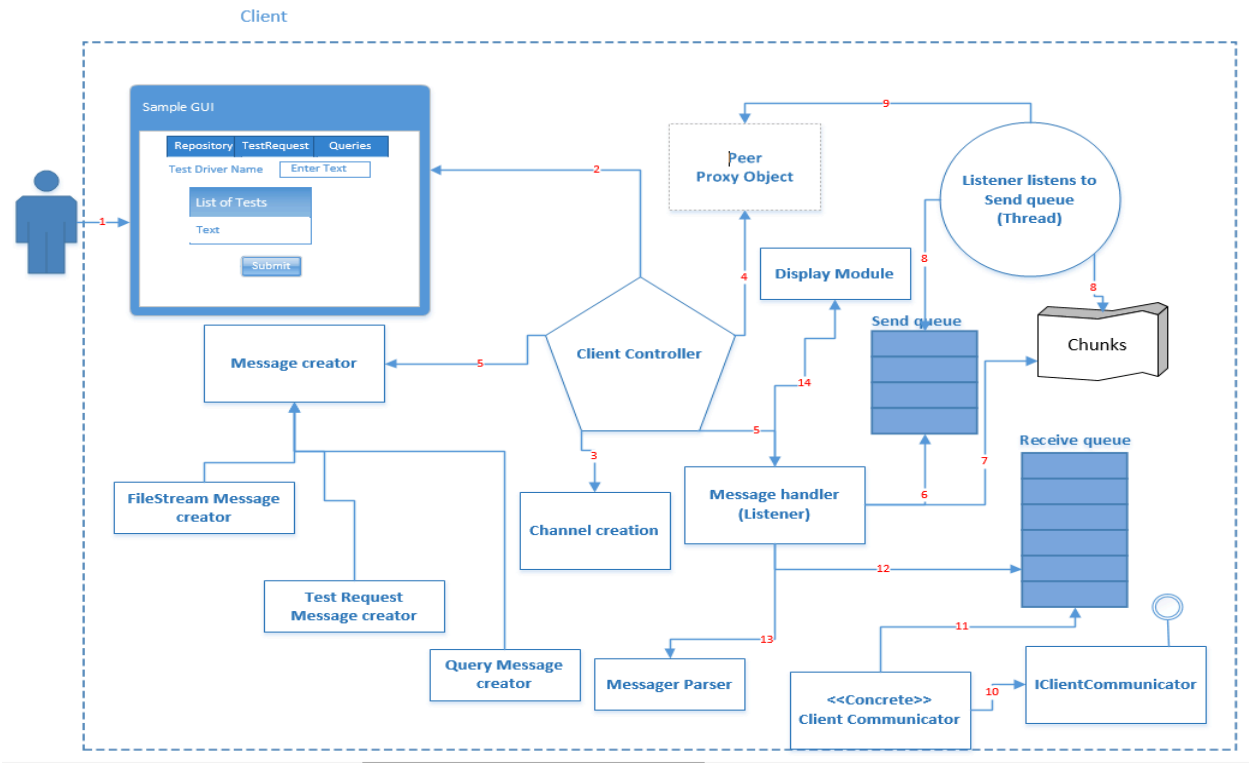
8.1 Summary

Software Collaboration Federation integrated all server tools in single client. User should be authenticated in order to access all the functions provided by SCF. Client GUI is developed using Windows Presentation Foundation.

8.2 Structure

8.2.1 Architecture diagram

Whole client GUI is developed by adapting MVVM pattern, so that there won't be any maintenance issues and code will be well organized. Client consists of two blocking queues. One is sender blocking queue, and other is receiver blocking queue. Both queues are used for communicating with other peers.



1. User interacts with Test Harness through GUI through GUI Module.
2. Client controller acts as controller for whole client application. Main functionality of controller is to delegate calls from one module to other.
3. Client controller creates WCF channel, which helps in communication with test harness server.
4. Creates required proxy objects.
5. Based on user input client controller create message request in XML format. If client is querying, query message is created, similarly if client is asking for file upload request then file stream message is created.
6. With the help of message handler, controller will enqueues incoming requests to “send blocking queue”.
7. For file upload request, file chunks are created by client controller with help of file manager.
8. 9. Listener listens to “send queue” and for “file chunks”, and identifies the type of message and finally post that to server by using proxy object.
10. 11. If server wants to communicate with client, then server creates client proxy object and calls client communicator methods. All the results will be en-queued in client receive queue
13. 14.Message Handler listener listens to receive queue. By de-queueing it will parse the message using message parser. And send back to display module in client.

8.2.2 User Interfaces

User interacts with Software Collaboration Federation using GUI interface. Layout of GUI is designed in tabbed fashion. Following are few tabs.

h. Repository tab
i. Test Harness tab
j. Build tab

k. Collaboration tab

Every tab is illustrated under user interfaces section in each server.

8.2.3 Package diagrams and interactions

a. FileMgr:

File manager is a utility package provides all the operations related to file operations such as opening, reading, writing, closing files etc.

b. Peer Connection:

Peer Connection is a utility package to create channels and to create proxy objects for particular services.

c. Blocking Queue:

Blocking Queue package implements a generic thread safe queue. Used for communication between peers.

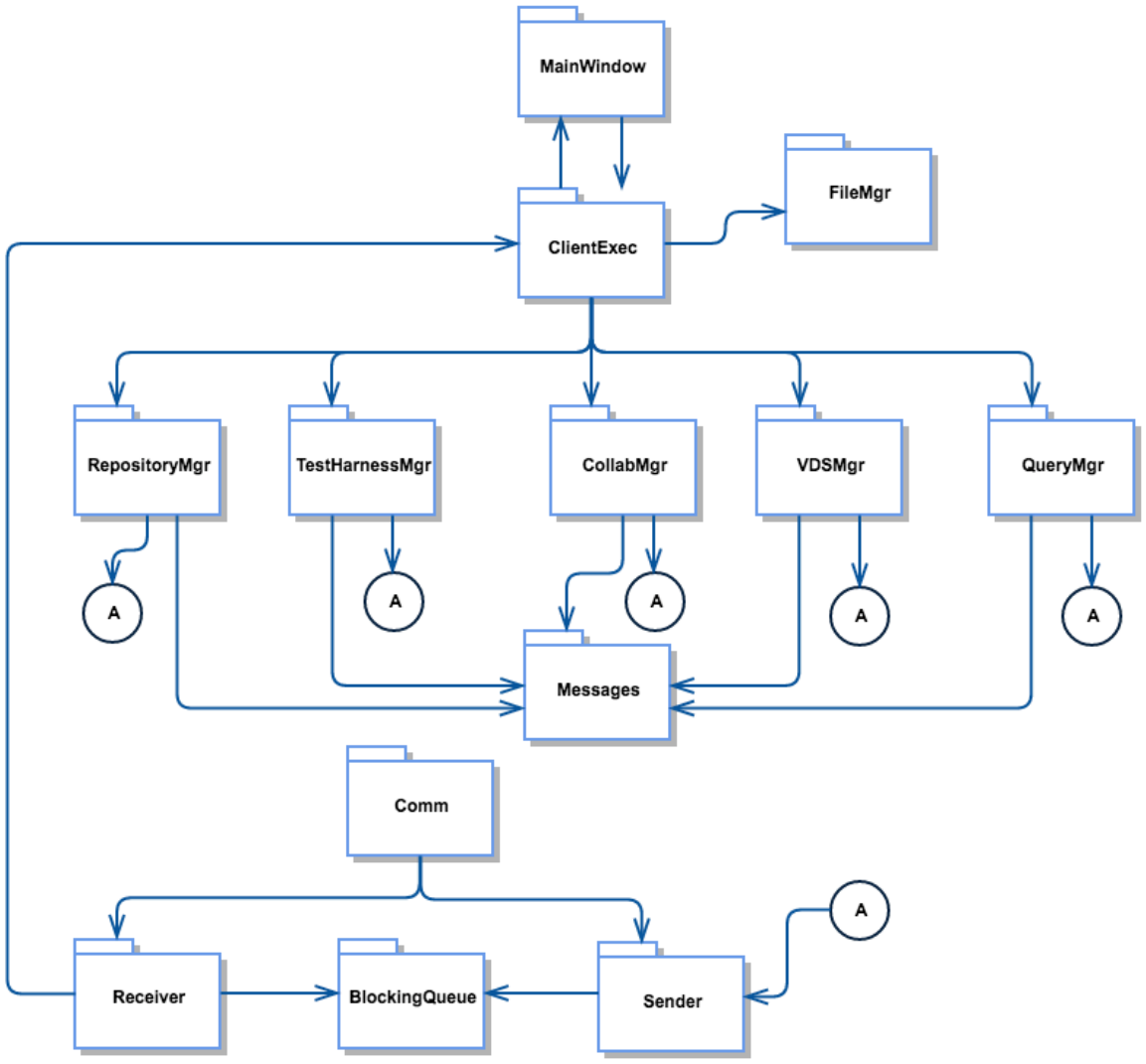
d. Receiver

Receiver package is used by Client services. All the incoming requests to client services gets enqueued into receiver blocking queue. Receiver package dequeues all the requests and starts processing.

e. Sender

Client uses Sender package to send responses for all requests. This package includes thread pool, when application is started this module creates required number of threads to process responses that are enqueued in sender blocking queue by client. Advantages of using this module.

1. Thread creation time will get reduced.
2. Thread management will become easy.



f. Repository Manager

This package contains logic related to repository client request. Input details are converted to messages. And those messages are enqueued into sender blocking queue.

g. Test Harness Manager

This package is used to create test request messages with the help of message package. And used for de-serializing messages to display results.

h. Collaboration Manager

This package is used to create collaboration request messages with the help of message. And used for de-serializing messages to display results.

i. Virtual Display Manager

This package is used to create Virtual display request messages with the help of message. And used for de-serializing messages to display results.

j. Query Manager

This package is used to create Query request messages with the help of message. And used for de-serializing messages to display results.

k. Messages

This package is used to parse requested message for further processing. Package contains extension methods used to de-serialize request into respective objects.

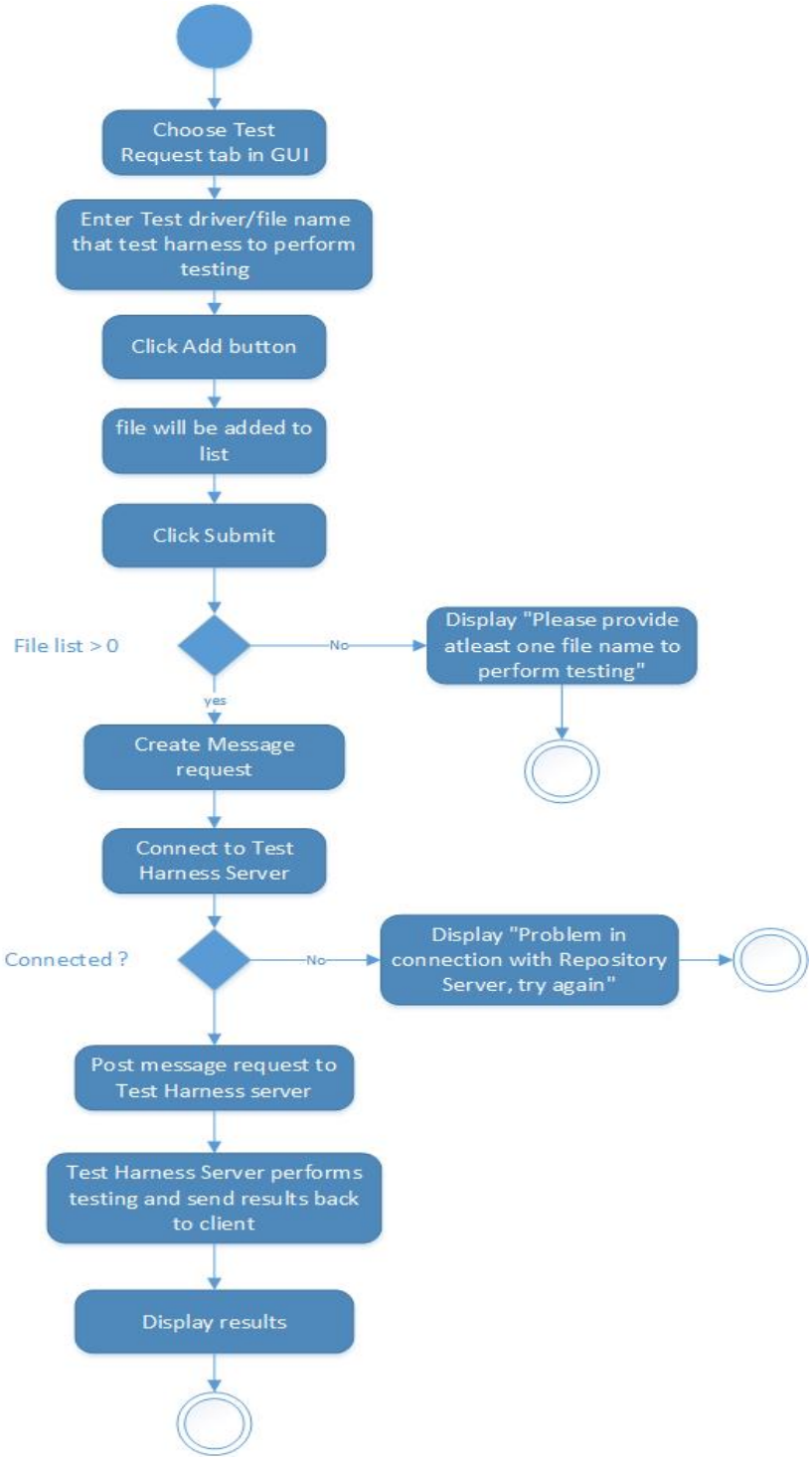
l. Main Window

This package contains display related logic, xmal, view models, models, user controllers.

8.2.4 Activity diagrams

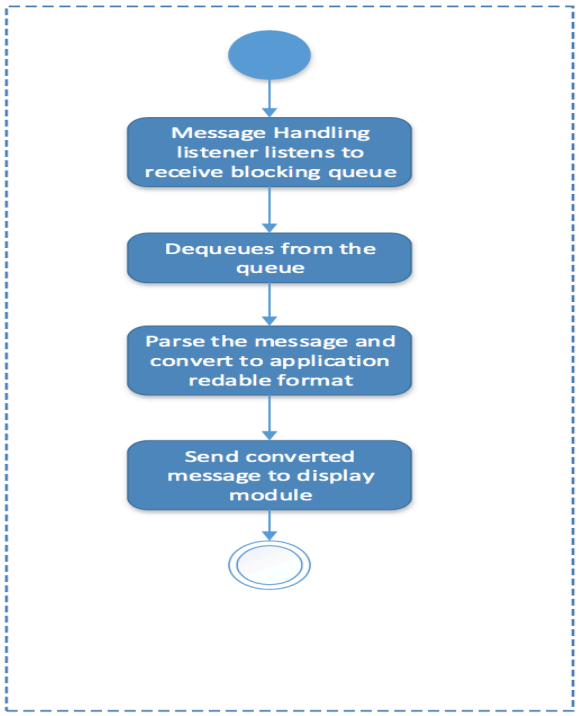
Following steps describes Test Request activity diagram.

1. User should select Test Request tab to perform test request.
2. User should provide name of the test driver on which test harness should perform testing
3. User should click on add button to add test driver name to list. This steps repeats
4. All files will be added to file list.
5. Click on submit button.
6. Client application converts request into xml format
7. Connects to Test Harness server.
8. Posts request to Test Harness server by using proxy object.
9. Test Harness starts processing this request and sends back the results to client.



Following activity diagram describes how Client accepts and display results coming from other peers?

1. Message Handling listener listens to receive blocking queue in client.
2. Dequeues from the queue.
3. Parse the message and convert to application readable format.
4. Sends converted message to client display module.



8.3 Critical Issues

Issue #1

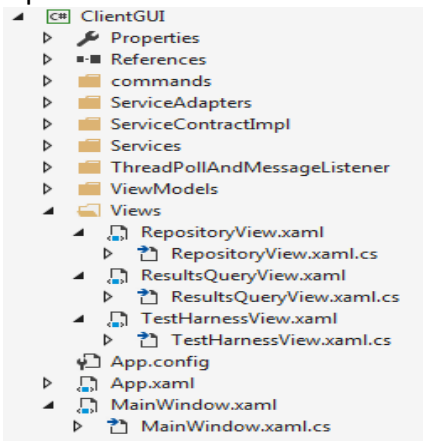
GUI design may be complex due to too many functionalities provided by Software Collaboration Federation.

Solution: Design GUI in tabbed fashion, each tab represents each server. Under each tab there will be side menu which helps to go through all the functions. In this way we can reduce complexity in showcasing all functionalities.

Issue #2

Maintenance Issue: If we code each and every GUI component in single MainWindow.xaml, there will be issue in understanding code in future if we are making any modifications.

Solution: Each tab is designed using UserControl concepts in WPF. So that all related functionalities will be there in single xaml file. For example, Repository tab, Result Query tab, and Test Harness tab is designed in separate xaml files.



We just plugin these user controls in MainWindow.xaml as following.

```
<TabControl>
    <TabItem Header="RepositoryTab">
        <!--Repository user control will be displayed in this tab-->
        <UserControls:RepositoryView x:Name="RepoView"/>
    </TabItem>
    <TabItem Header="TestHarnessTab">
        <!--Test Harness user control will be displayed in this tab-->
        <UserControls:TestHarnessView x:Name="testHarnessView"/>
    </TabItem>
    <TabItem Header="ResultQueryTab">
        <!--Query results user control will be displayed in this tab-->
        <UserControls:ResultsQueryView x:Name="QueryView"/>
    </TabItem>
</TabControl>
```

9 Messages

All systems in Software Collaboration Federation communicate with each other using messages send through windows communication foundation. Some of the important messages are described in this section.

Test Request Message:

Developers or QA team send Test Request Message to Test Harness server to perform testing on their test drivers. Test Request consists of List of test driver names.

<TestRequest>
    <TestDriverName>TestDriver1</TestDriverName>
    <TestDriverName>TestDriver2 </TestDriverName>
    <TestDriverName>TestDriver3</TestDriverName>
</TestRequest>
<!—test harness will get all the test driver and its dependencies from build server-- >

Whole test request message is wrapped with message and send to test harness server. Wrapper looks as follows.

<Message>
    <Author>Santhosh</Author>
    <To> {Test Harness Address} </To>
    <From> {Client Address} </From>
    <Type> TestRequest </Type>
    <username>Venkata</username>
    <password>****</password>
    <Body>
        // Above Test Request Message
    </Body>
</Message>

Sender package parses this message to get “To Address” and send that message to that address.

TestRequest Results Message:
After performing testing test harness sends testing results to repository and client for storing results and to display respectively. Test Harness results message will look as follows.

```
<TestResults>
        <Author>Santhosh</Author>
        <Results>
                <TestResult>
                        <TestName>Test1</TestName>
                        <passed>true</passed>
                        <log>test always passes </log>
                </TestResult>
        </Results>
</TestResults>
```

This result is wrapped with message and send it to repository for storing and client for displaying.

```
<Message>
        <Author>Santhosh</Author>
        <To> {Client Address, Repository Address } </To>
        <From> {Test Haress Address} </From>
        <Type> StoreTestResults/DisplayTestResults </Type>
        <username>Venkata</username>
        <password>****</password>
        <Body>
                // Above Test Result Message
        </Body>
</Message>
```

Check-In Request Message:
Developer check-in his/her code to repository, first developer sends check-in request message which includes details of files that needs to check-in and whether check-in is open check-in or not. Check-In Request message is wrapped with message. Message looks as follows.

```
<Message>
        <Author>Santhosh</Author>
        <To> {Repository Address} </To>
        <From> {Client Address} </From>
        <Type> CheckInRequest </Type>
        <username>Venkata</username>
        <password>****</password>
        <Body>
                <CheckInRequest>
                        <OpenCheckIn>true</OpenCheckIn>
```

```
                    <FileNames>
                        <FileName>BlockingQueue.cs</FileName>
                    </FileNames>
                </CheckInRequest>
        </Body>
</Message>
```

Repository server parses the above message and checks whether same file and version exists or not if not pulls all the files from client using client proxy object.

Check-Out Request Message:

Client can extract or check out file by requesting file name. If client have permission on the requested file. Repository allows client to download file. Message will be as follows

```
<Message>
        <Author>Santhosh</Author>
        <To> {Repository Address} </To>
        <From> {Client Address} </From>
        <Type> CheckOutRequest </Type>
        <username>Venkata</username>
        <password>****</password>
        <Body>
                <CheckOutRequest>
                        <FileName>BlockingQueue.cs</FileName>
                </CheckOutRequest>
        </Body>
</Message>
```

Schedule Meeting Request:

Client sends schedule meeting request to schedule collaborative meetings. Request will be as follows.

```
<Message>
        <Author>Santhosh</Author>
        <To> {Repository Address} </To>
        <From> {Client Address} </From>
        <Type> ScheduleMeetingRequest </Type>
        <username>Venkata</username>
        <password>****</password>
        <Body>
                <Meeting>
                        <title>Project discussion </title>
                        <hostingclient>Address</hostingclient>
                        <hostname>Sai</hostname>
                        <participants>
                                <name>Santhosh</name>
                                <name>Satheesh</name>
```

```
                    <name>Shishir</name>
                </participants>
            <Meeting>
        </Body>
</Message>
```

Work Package assignment request:

Software Architect assigns work package to each responsible individual. He will send work package assignment request to collaboration server. Message will look as follows.

```
<Message>
        <Author>Santhosh</Author>
        <To> {Collaboration Address} </To>
        <From> {Client Address} </From>
        <Type> WorkPackageAssignRequest </Type>
        <username>Venkata</username>
        <password>****</password>
        <Body>
            <WorkPackageAssign>
                <WorkPackages>
                    <WPId>DS2019</WPId>
                <WorkPackages>
                <DeveloperId>Santhosh</DeveloperId>
            <WorkPackageAssign>
        </Body>
</Message>
```

View Source Code response:

Client has to know how project layout looks like in repository for that client request for layoutRequest. Repository send response to client as follows.

```
<ViewSourceCodeResponse>
  <Response>
    <Solution name ="Solution">
      <SubSystem name="Test-Harness-Sub-System">
        <Versions>
          <Version name="V1.1">
            <Modules>
              <Module name="BlockingQueue">
                <Versions>
                  <Version name="V1.2">
                    <FileNames>
                      <FileName>BlockingQueue.cs</FileName>
                      <FileName>BlockingQueue.doc</FileName>
                    </FileNames>
                  </Version>
                  <Version name="V1.1">...</Version>
                  <Version name="V1.0">...</Version>
                </Versions>
              </Module>
            </Modules>
          </Version>
        </Versions>
      </SubSystem>
    </Solution>
  </Response>
</ViewSourceCodeResponse>
```

10 Prototypes

Prototype code for Repository critical issue #3 - Concurrent access to files in Repository

```
public int numRetries { get; set; } = 10;
public int waitMilliSec { get; set; } = 50;
public bool show = true;
```

```
public FileStream doWithReTry(string fileName)
{
    FileStream stream = null; ;
    for (int i = 0; i < numRetries; ++i)
    {
        try
        {
            if (i > 0)
                Console.Write("\n retrying to open file ");
        stream = new FileStream(fileName, FileMode.Open);
    }
    catch
        {
            Thread.Sleep(waitMilliSec);
        }
    }
    return stream;
}
```

Above doWithReTry function accepts file name as input, if it finds exception in accessing that file, then function will try to access again for maximum number of times. Report if still it is giving exception after maximum retry attempts.

Test Harness Prototype:

To illustrate how the test harness will help Software Collaboration Federation to support CTAI. Following steps will be performed by test harness to achieve CTAI.

1. Whenever there is a request for performing test on test drivers, test harness asks build server for all executable files related to project.
2. Test Harness loads all the files into child application domain
3. With the help of reflection, test harness incepts each and every file to identify test driver and perform testing.
4. Communication between servers, clients is done using WCF.

Following code demonstrate how it works

Application domain setup is used to create configuration information for new child domain. Configuration information contains application base path, private bin path, private bin probe path, and application name information. Below is snippet of code show of application domain setup class.

```
AppDomainSetup domaininfo = new AppDomainSetup();
```

```
     domaininfo.ApplicationBase = AppDomain.CurrentDomain.BaseDirectory;
    // domaininfo.PrivateBinPath =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "dl\\");
           // domaininfo.PrivateBinPathProbe =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "dl\\");
```

Evidence is used to set information that constitutes input to security policy decisions.

```
            //Create evidence for the new AppDomain from evidence of current
            Evidence adevidence = AppDomain.CurrentDomain.Evidence;
```

Child application domain is created by using application domain CreateDomain static method.

```
            // Create Child AppDomain
            AppDomain domain = AppDomain.CreateDomain("ChildAppDomain",
adevidence, domaininfo);

            // creates instance
```

CreateInstanceAndUnwrap method is used to create instance and assembly is used to load dll files using loadfrom method. And with the help of reflection test method is executed.

```
       AssemblyProxy proxyInstance =
(AssemblyProxy)domain.CreateInstanceAndUnwrap(
                typeof(AssemblyProxy).Assembly.FullName,
                typeof(AssemblyProxy).FullName);
                  string[] files =
      fileManager.getFilesInSpecifiedPath(AppDomain.CurrentDomain.BaseDirectory,
      "*.dll");
            //fileManager.displayFileNames(files);

            foreach (string file in files)
            {
                //Loads DLL files and returns Assembly object reference
                      Assembly assembly =
      proxyInstance.getLoadedAssembly(System.IO.Path.GetFileName(
                    file));

                // AssemblyName[] arrayOfAssems =
      assembly.GetReferencedAssemblies();
                //with the help of Reflextions program can able to go through all
      types
                //if any class implements ITest interface it will create instance
      and call test method

                Type[] types = assembly.GetExportedTypes();
                foreach (Type t in types)
```

```
    {
        ITest.ITest
            lib = null;
        if (t.IsClass && typeof(ITest.ITest).IsAssignableFrom(t))
            lib = (ITest.ITest)Activator.CreateInstance(t);
        else
            continue;
        Console.WriteLine(lib.test());

    }
}

//used to unload domain
AppDomain.Unload(domain);
```