# CSE 681 Software Modeling Analysis

Instructor

Dr. Jim Fawcett

# Build Server OCD

SHUBHAM RAMESH JIWTODE
#870427143

# Contents

## Figures:

# 1. Executive Summary

Advancement in technology has led to increase in size and complexity of a system. The developers are expected to work efficiently, correctly so that the product can be brought to market at the earliest.

Each developer develops his own part of the final product code which is then merged with the code from the other developers or the other teams. While developing such a big product, maintaining the sync between developers or in between teams, maintaining the quality, and keeping note of the pace of development is a difficult task. Also, everyone involved in the product development must know about its status so that they can make further plans. This project mainly focuses on building a build server. The Build Server is an automated tool that builds test libraries. First, a client sends test request to repository. Then, build server accepts the file and test request from repository, builds it and sends the build libraries for further testing when requested. The tests are carried by the test harness, which sends all the logs to repository and a notification to the client describing the results. Project presents a way in which numerous codes can be handled, tested, examined.

This project is meant for Teaching assistant (TA), Professor, Developer, Quality Assurance (QA), Manager, Customer.

- The TA and professor will examine document to check whether the concept covers all the necessary points.
- The developer will refer this project in future during implementation.
- QA to test the quality of the product.
- Manager can check log to get an idea about the pace of the work.
- Customer also refers to the log to get an idea about the product handover date.

The project may also encounter some limitation due to the following issues:

- Longer builds – If the building process takes longer time than normal. Then, the whole development process during high time may slow down and the developer may have to sit idle.
- More workload – This may happen when lots of projects are sent to build server from different teams.
- Complex builds- When different teams use different components and libraries in their code. This leads to complex building process. Possible solutions are discussed further in the document.
- Inability of the build server to support code built in multiple languages and code built on multiple platforms such an Windows and Linux.

# 2. Introduction

In this project, we emphasize on building a build server. Build server facilitates remote building, unbiased with the environments it's running on. The big companies working on projects have to deal with lots of codes, testing of those codes, merging of those codes, and retesting them before inserting them to the baseline. With lots of developing and testing tasks, companies need something accurate and fast to release a quality product within the time limit, this is when build server comes into picture. In this project, we will build a build server module, a mock repository module, a mock test harness module, and a mock client-side module. We need mock modules as this project mainly focuses on Build server, the other modules are just to replicate the whole process where the test is generated by client and at the end the results are displayed to the client.

## 2.1 Objective and Key Idea

The objective of the project is to fasten the product development phase without compromising the quality of product. To achieve this, the key idea is to build a build server. When there are large number of code build server will perform task like building the code, sending build logs, integration of codes from multiple sources, sending code for testing. Build server, ensures the product quality and automation of processes with high throughput.

## 2.2 Obligations

The primary obligations of build server are

- Accepting Build request Command from Client.
- Fetching the files and request (single or multiple) from the repository to initiate build process.
- Generating build logs and notifying client about the result of build.
- Sending the lib to test harness and commanding test harness to start the testing.

## 2.3 Organizing Principles

- The whole project is distributed in 4 different modules i.e. Client module, Repository module, Build server module, Test harness module. Each module work independently but in sync to avoid any loss in data or information.
- Build server comprises of child builder which accepts test request on first come first serve basis. This way we can build several build requests in parallel. The number of child process is decided by the Client.

## 2.4 Key Architectural Ideas

- The key idea is to build a build server module along with a mock repository module, A mock Test Harness module, Mock client module.
- The communication between the modules will be established using windows communication foundation (WCF). WCF will responsible to send files, test request, logs, notification on the communication channel.

- The Graphical User Interface has been developed using Windows Presentation Foundation. All the users can access the Build server from client using GUI.
- The whole system will be developed in C# .NET framework in Visual Studio 2017.

### 2.4.1 Message Passing Communication

Message passing is a form of communication between objects, processes. To communicate, a channel is established, and processes communicate using messages. Message passing can be **synchronous** or **asynchronous**. Synchronous message passing systems require the sender and receiver to wait for each other while transferring the message. In asynchronous communication the sender and receiver do not wait for each other and can carry on their own computations while transfer of messages is being done.

#### 2.4.1.1 Windows Communication Foundation

WCF is designed to support distributed computing where services have remote clients.one can send Asynchronous messages from one endpoint to another. A WCF client connects to WCF service via an endpoint. Each service exposes its contract via one or more endpoints. An endpoint has an address (which is a URL specifying where the endpoint can be accessed) and binding properties that specify how the data will be transferred. Each message in WCF consist of a Simple Object Access Protocol (SOAP) wrapper around a serialized instance of a data class that defines the request, to and from addresses, and any parameters needed to execute the request.

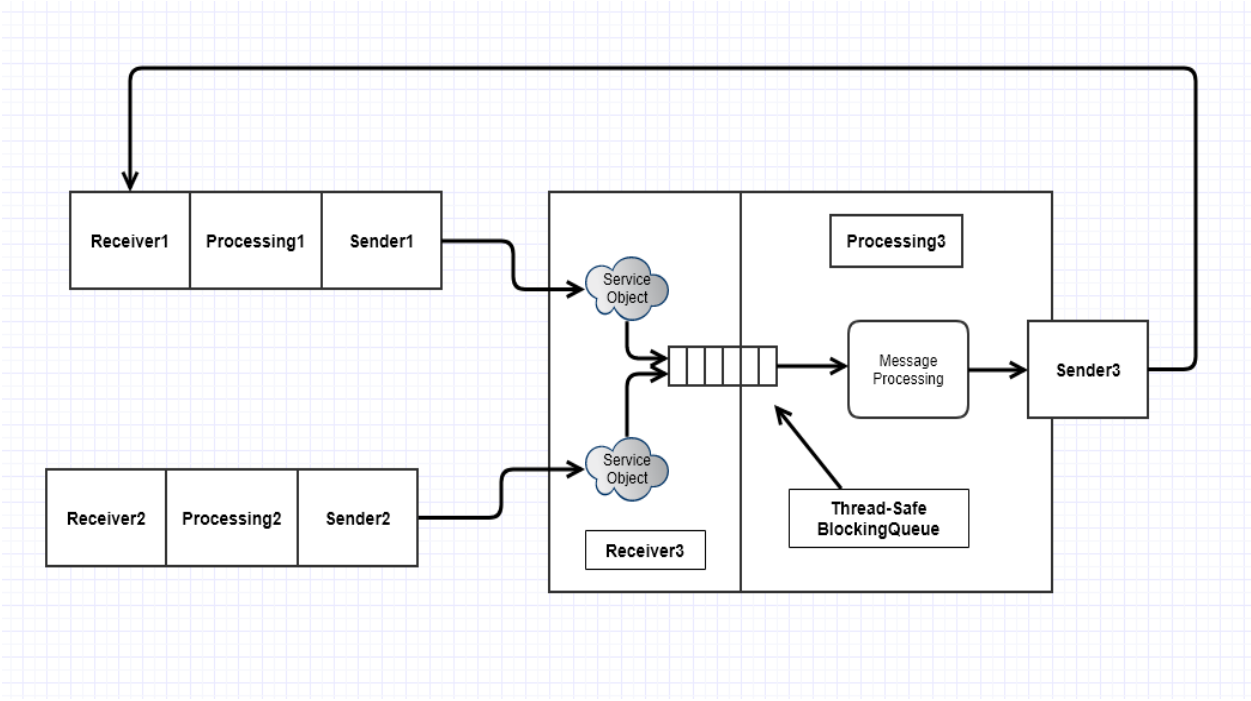

Fig1.WCF Message-Passing Communication

### 2.4.1.2 Message Structure

Message structure used in message passing is as shown below.

```
Message Type : Request, Connect, Reply, Close sender, close receiver
to           : Address from where message is been sent
from         : Address where message is been sent
author       : Name of author of message
command      : To indicate the operation being done
arguments    : To send any addition info
```

- Request type is used when we must send a message.
- Connect type is used to establish a channel.
- Reply type is send in response to a message.
- Close sender and close receiver are sent to inform that the sender or receiver has to be closed.
- The command attribute is used to command the receiver process to perform a operation. The receiver process reads the command attribute from the message and invokes appropriate function to complete the operation mentioned in the command.
- The arguments attribute is used to hold various parameters such as file names which one process may need from another process.
- The to and from parameters specify the source of message and destination of message respectively.

## 2.4.2 GUI using WPF

The GUI is developed for the user to interact with the entire system. The user interface provides the facility to add multiple tests a test request, provides a list box to view the received test logs and build logs and send the created test request to the build server. The user interface is developed using Windows Presentation Framework. The appearance and the behavior of the user interface is well separated in WPF. A XAML which an XML based language defines the appearance while the code defines the behavior of the user interface.

# 3. Use Cases

## 3.1 Developer

- Developers are the primary users of this project. This project not only builds the code but also generates build logs, generates test logs. The build server makes smart use of child builder to reduces the build time thus developer don't have to wait longer for the results.
- Cross platform testing. If the developer environment and target environment are different, build server provides cross platform testing. This helps developer to test the application in target environment during developing phase.
- Thus, developer can get an idea about his current work and make the changes if required.

## 3.2 TA/Professor

TA and professor will examine the product to check whether the project works according to the requirements of the problem statement. They will also check whether the project issues are discussed, and satisfactory solutions are proposed.

## 3.3 Quality Assurance

QA's are responsible for finding out any bugs or defect in the working of code developed by the developer so that if there is any bug or defect it can be eradicated as soon as possible. QA ensure the quality by making some changes to the process or repeating the process without repeating the mistake. QA can refer to build logs and test logs by simply accessing the GUI to get an idea about the current work. also perform testing on the latest and trusted build to ensure the requirements and quality of the product.

## 3.4 Manager

Manager has to keep account of the time, cost of the project, also coordinating the people, resources and processes required to deliver product. To complete the work in speculated time and cost manager can refer to logs to check for any defects or requirement, to plan and reschedule the developer, other resources if required.

## 3.5 Client

In case if the customer company sends a representative to get an acknowledgement about the product being developed or wants to test whether product is committed to the quality promised. He/she can check the logs and conclude whether the development team is lagging behind or are going good. This helps the customer company to plan their further events like product launch etc.

# 4. Activity Diagram

The activity diagram describes the process flow involved in the system. This diagram show some high level process of client module, repository, test harness and build server. A more comprehensive activity diagram of build server is shown later in this section.

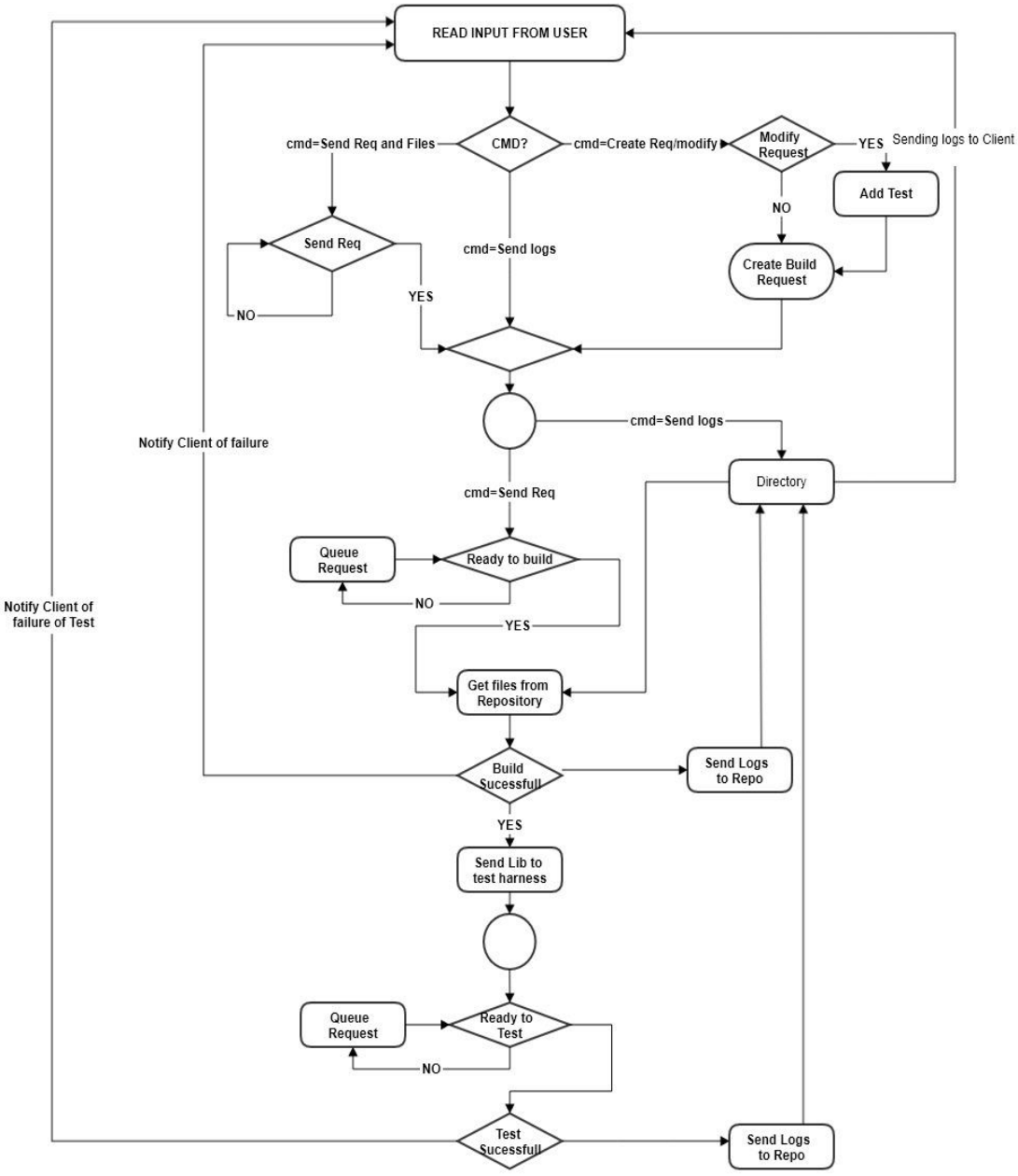


**Fig 2. Activity diagram of overall system**

## 4.1 Client

### 4.1.1 Read the User Input /Command

Client will receive Command through graphical user interface. User can use button, text box, and list box to input a particular command. Depending on the input from user client does further processing. Following are some of the commands at user end.

- **Create Test Request**- User can select multiple files from test files and test driver list box to create a Test req.
- **Send Logs**- This is used to get logs from repository and display them on GUI.
- **Add Test**- One can add multiple test to an already existing test.
- **Send Request**- It is used to send the build request to Build server.

### 4.1.2 Display logs and result

GUI Comprises of two list boxes for build logs and test logs resp. All the list boxes are updated using Dispatcher. Both contains logs, which provide a more comprehensive view of the build and test process. Using those logs one can locate error or defects in building and testing process.

## 4.2 Repository

### 4.2.1 Storing Build request, Files and Logs

On user command (send request) client sends build request and related files to repository. Repository stores the files until requested by the build server. Repository also stores build logs and test logs received from Build server and test harness respectively.

### 4.2.2 Sending files to Build Server

When a particular build request is to be build, On request of build server, repository parses that particular build request and sends the required files to builder.

### 4.2.3 Sending Build logs and Test logs

When requested by user repository sends build logs and test logs to client.

## 4.3 Builder

Below is a comprehensive activity diagram of builder.

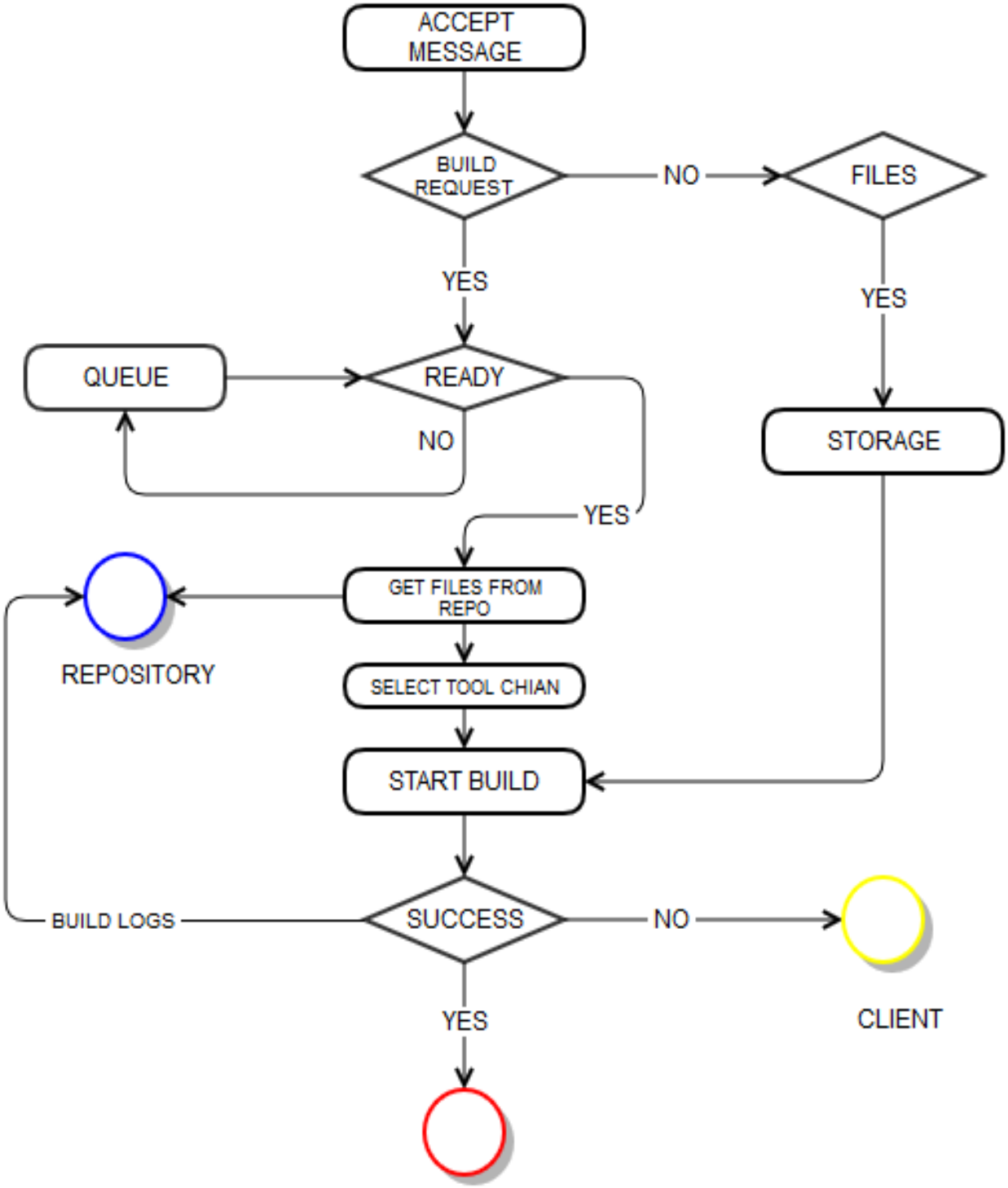


**Fig 3. Comprehensive activity diagram of Build Server**

### 4.3.1 Accept Messages

Build server accepts the message sent over the communication channel. Incoming messages are queued to a blocking queue. Once builder is free message is de queued.

### 4.3.2 Build Request

If the message received is build request it is passed on to start building process, else if the message has files along with it, they are stored in local storage.

### 4.3.3 Ready

If child builder is not ready the build request is queued , once the builder becomes free the build request is passed to builder to initiate build process.

### 4.3.4 Get Files from Repo

After receiving the build request builder parses the request and request repository for all required files.

### 4.3.5 Select Tool Chain

- A general tool chain may have a compiler/cross compiler, assembler, linker.
- Compiler – It is a program written in an native language which turns a code written in high level language into an object code. We will be using csc.exe as the compiler in our project.
- Cross compiler – These are used when your target environment is different from the developed environment. For example a developer who has written code on MAC OS can test the same code in build server running on Windows OS.
- Linker- As the name implies the role of linker is to link object code, resources in an single file (a library file or executable). A working prototype has been added in section 7.

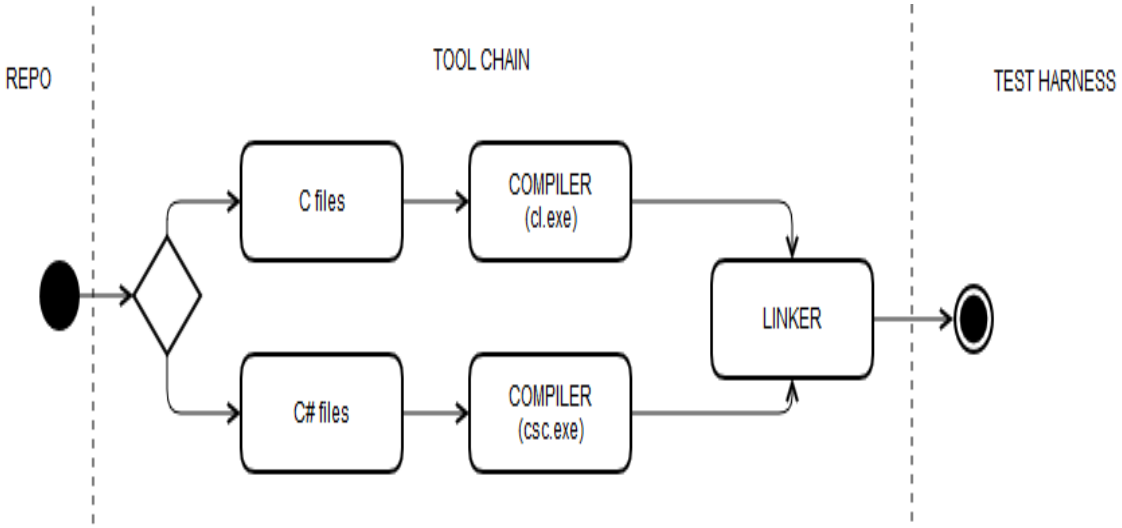


**Fig 4. Tool Chain sequence diagram**

### 4.3.6 Start Build

After selecting the tool chain builder will get the files from local storage and start building the files. If success libraries are sent to test harness else user is notified.

In both the cases logs are sent to repository.

## 4.4 Test Harness

### 4.4.1 Accept test request

The test log received from builder is parsed the required file are fetch into local storage. If loader is busy the request is queued till the loader becomes free.

### 4.4.2 Load libraries

Once the loader is free the request is d-queued and loaded and the required tests are performed.

### 4.4.3 Send logs and notify client

After the test is completed the test logs are sent to repository and client is notified about the result.

# 5. Partitions

Package Diagram show packages involved in the system. Packages perform a specific task assigned to them.

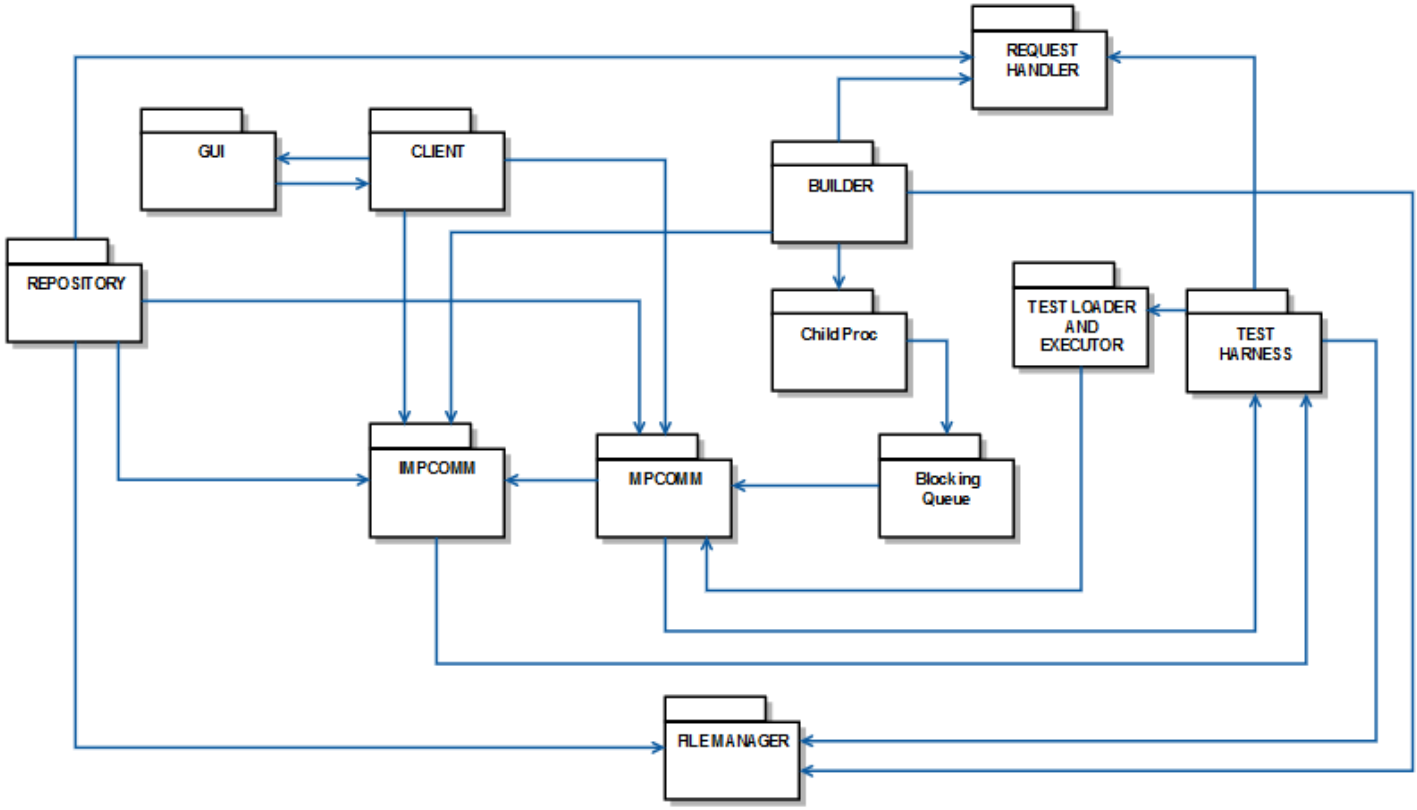


**Fig 5. Activity diagram for entire system**

## 5.1 Client

Client package is responsible for communication between GUI and rest of the project, any command send by the user on GUI are passed to client and in return client sends that particular message or command over channel. Client is also responsible for keeping the content of GUI (list boxes) up to date by continuous monitoring any change in repository contents.

## 5.2 Repository

Repository package is responsible for maintenance of files for whole system, it's a huge directory with some sub directory. It also send files to them when requested for, Repository receives files from Build server and test harness, build logs and test logs that are sent after completion of build and testing process.

## 5.3 Request Handler

This package is responsible for creation of request, manipulation to it and parsing it when required. Package handler also save the final request file and load it to a document file when asked to. A sample XML request is shown in appendix.

## 5.4 Test Harness

Test harness accepts libraries from Build server after successful build. Test Harness then loads those libraries in loader (one by one if there are multiple libraries), notifies about the result and sends relevant logs to the repository for further reference.

## 5.5 Builder

Builder package is responsible for spawning several child builder and scheduling them in accordance to availability of test request. To achieve this builder maintains two queue one for test request and another for available child process.

## 5.6 Child Process

Child process is nothing but child builder where actual code is build. After been spawned child builder waits for test request (to be sent by the mother builder). After receiving the request it fetches the required files from repository builds those files and sends logs back to repository. On successful build it sends the libraries to test harness for further testing.

## 5.7 GUI

Graphical User Interface provides an interactive end to project. GUI provides the provision of creating , manipulating the test request, staring the process, it also displays logs and notifications.

## 5.8 Test loader and Executer

Test loader load the libraries provided by the build server on successful loading tester perform some test over the loaded libraries and respective results are stored in log file.

## 5.9 Blocking queue

This package implements generic blocking queue. All the modules i.e. repository, build server, client, test harness all use instance of blocking queue to manage all the incoming messages over the channel.

## 5.10 MPComm

The MP Comm package is responsible to take care of the communication part of the system. It initializes the sender and receiver so that the work load is shed off from the Executive. It also keeps the track of the last channel that is connected to and opens a new channel if the last Url is not equal to present to filed in the address. The MP Comm Service Package uses messages to communicate between the sender and the receiver. The MP Comm Service package defines three classes sender, receiver and the Comm Classes. The Sender class implements public methods such as Connect, Close, PostMessage and PostFile. The Receiver Class implements methods such as OpenFileFor Write, WriteFileBlock etc. The Comm Class implements using Sender and Receiver instances with public methods such as CompareFilesBytes and CompareMessages.

## 5.11 IMPComm

The IMP Comm Service package provides interface used for message passing and file transfer. It contains Comm Message class which is used to define the message structure and represents serializable messages, it also initializes the client storage and service storage environments for each module.

## 5.12 File Manager

It manages receiving and sending of files for each modules.

# 6. Builder Package Diagram

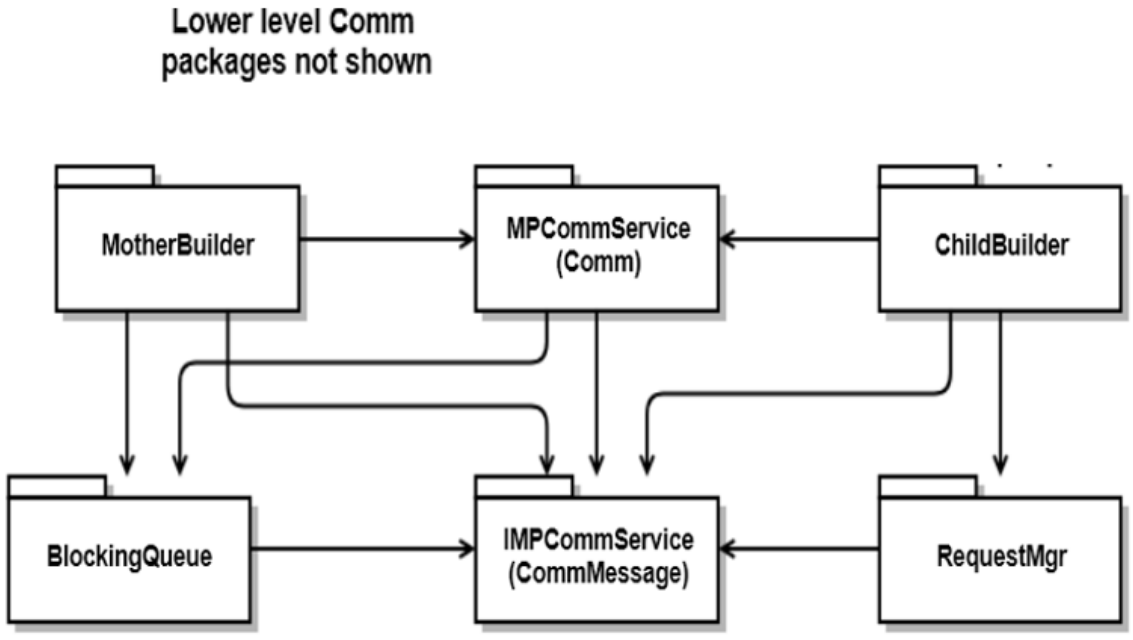


**Fig 6. Activity diagram for Builder**

## 6.1 Mother Builder

The Mother Builder package is responsible for managing the Child Builder Processes.

## 6.2 Child Builder

The child builder package builds test libraries. It is also responsible to send build logs to the repository and send the test libraries to the test harness to execute the test.

## 6.3 Request Manager

The Request Manager is responsible for creating the test request xml once a build request is initiated by the GUI. It also implements the methods to parse and load the Test request created.

## 6.4 MP CommService

This package provides communication services used by the Mother Builder and Child Builders. The MP Comm Service Package uses messages to communicate between the sender and the receiver.

## 6.5 IMP CommService

The IMP CommService defines the CommMessage structure used by the communicators.

# 7. Message Flow Diagram

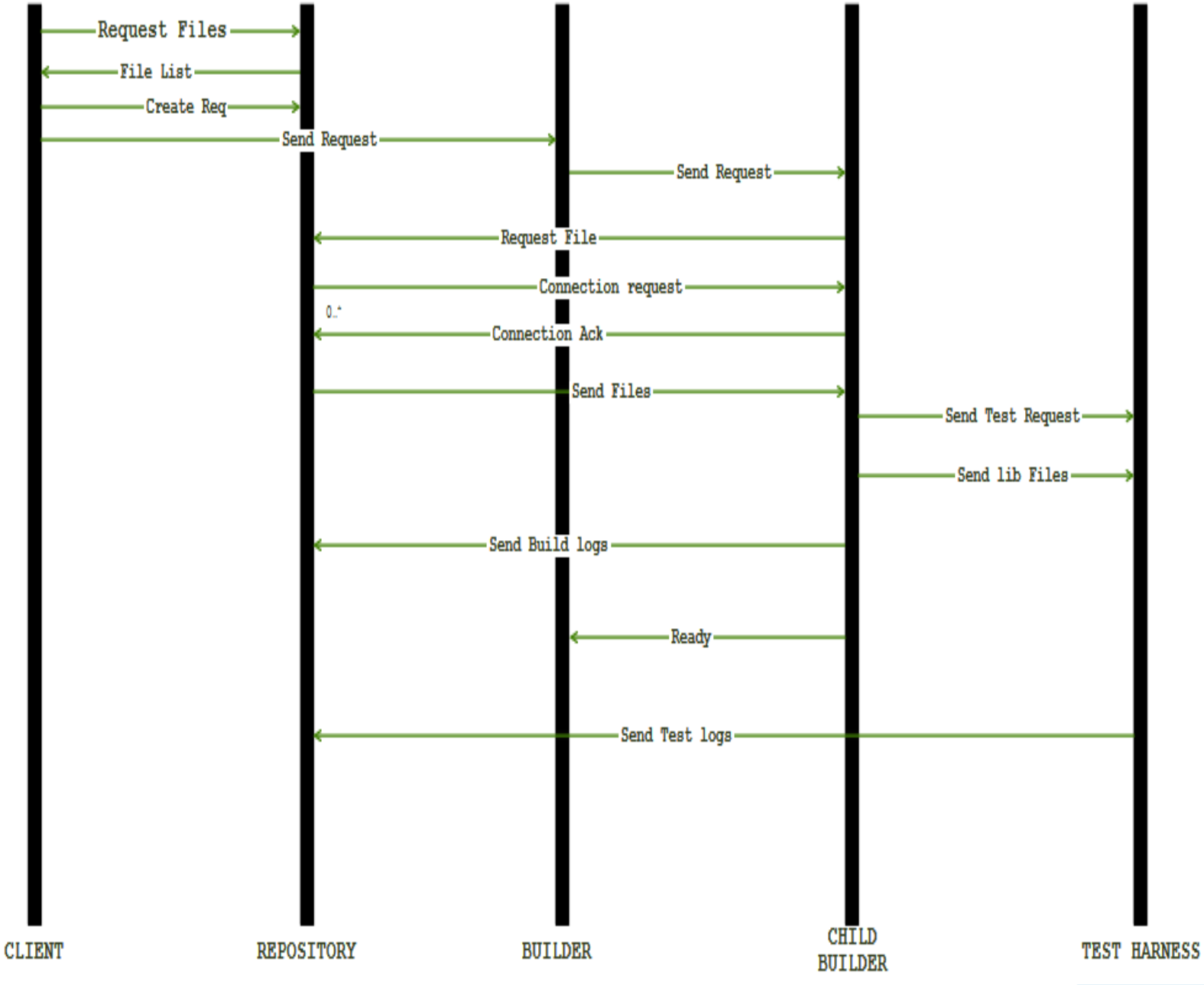


**Fig. 7 Message flow diagram**

- **Request Files**-Client request repository for update on the file it has.
- **File List** –In response to request files Repository send list o different files available in its storage directory.
- **Create Request**- Client creates the request and sends it to repository.
- **Get Files**- Repository parse the request and fetched the required files.
- **Send Request to mother**- This message send the selected request to mother builder.
- **Send Request to child**- Mother send this message to child proc long with the build request if child proc is free.
- **Request Files**- after receiving build request child proc request repository for files.
- **Connection Request**- Repo sends a request to child proc to establish a channel to send files.
- **Connection Ack**- Child Proc acknowledges the repo that the channel is established.
- **Send Files**- Repo send all the request files to Child proc.
- **Send Test Request**- After successful build child proc sends the message to test harness, Also sends test request along with message.
- **Request lib files**- Test harness request child proc for the files in test request.
- **Send Lib Files**- Builder then sends the requested files to test harness.
- **Send Build Logs**- Child proc send the build logs to repository.
- **Ready**- Child proc sends this message to mother builder to notify builder about its availability.
- **Send Test Logs**- Upon completion of Testing test harness send the logs to repository.

# 8. Class Diagram of Builder

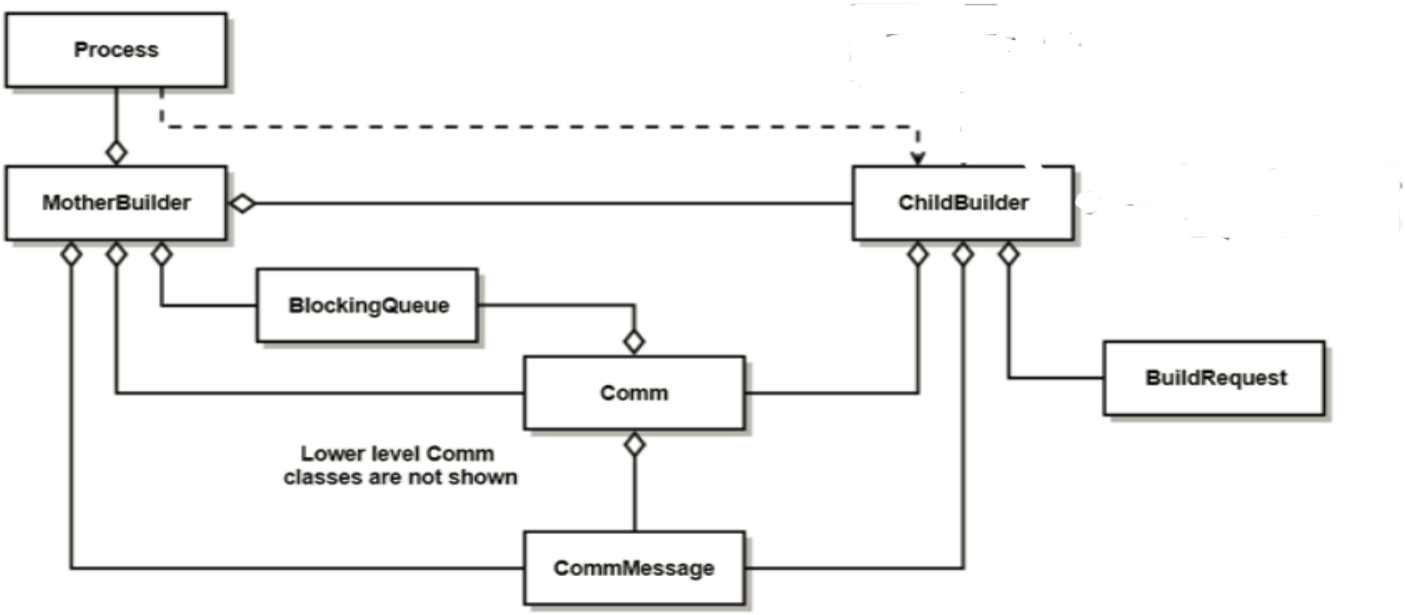


**Fig 8. Class diagram of Build server**

**8.1 Mother Builder** Spawns child processes and receives message and build request from client queues it to the queue and sends them to child process if available. Once the child builder is free the message is DE queued and the message is fed to builder.

**8.2 Child Builder**- Actual build take place in child builder. On builder command child process takes the build request, fetches the file and start building. After completion it notifies builder about its availability.

**8.3 Blocking Queue**- It a generic blocking queue it holds the messages coming over the channel in a queue.Builder and child process implements blocking queue for incoming message.

**8.4 Comm**- It performs task like posting a file, posting message, writing to a file.

**8.5 CommMessage**-This describes the structure of the message and also holds the operation contract

.

**8.6 Build Request**- It manages the build request until the child builder is free. Once child is free a particular test is passed to child on builders command.

# 9. Critical Issues

## 9.1 Longer build

Due to large number of developers and several versions many build requests can stack up due to which following scenarios can occur.

- Developer must sit idle waiting for his build request to build so that he can troubleshoot if any error occurs.
- If developer starts working on another project while his previous code still in build server, at that time if any error occurs the developer has to go through the previous work again to solve the error.
- During high time if any build request takes longer time than it should normally take, all other code will have to queue up which will be a bottleneck situation.
- Due to huge build time the amount of bugs fixed can be less and due to time constraints less number of test may be performed which will hamper the quality of product.

### Solution-

- One solution to this problem is by using parallel builds on same machine (process pool), thus as there will be two or more build running in same time span. Average time take by each build will be less than that in single build operation.
- Precompiled libraries can be created for portions of the code that change less frequently.

## 9.2 Heavy Workload

Build server are also used at the large companies where there are tons of developer, each developer is working on some project. There can be a time when company plan to launch or release most of their work at a particular day. During this time the build server can receive heavy workload. Thus even if per build time is less, due to large number of build the developer will have to wait till the build server fetches his code. The total time in this kind of situation is very large as compared to time taken in single build.

### Solution-

- This problem can be solved by using distributed build or using parallel build. In distributed, the code is run over the cluster of machine while in parallel same machine runs two or more build process.
- Another way may be by prioritizing the work so that the more important build doesn't have to wait longer.
- Also by upgrading the processing power of server one can serve the high demands.

## 9.3 Complex build

At time even if the components in the build are less in number and there is no load on build server, Complex build can be a headache. Complex builds may have multiple libraries, framework, third party resources. Complex build can not only take long time but also are difficult to build, one might also need human intervention, there is also a possibility that the build might break in between due to distributed dependencies. In such case only increasing the build speed is not a solution.
Due to huge build time the amount of bugs fixed can be less and due to time constraints less number of test may be performed which will hamper the quality of product

### Solution-

- One way do address this issue is by optimizing the make file. Make file contains headers, source file info and also last done updates. There are two ways to do the optimization peephole optimization and structural optimization.
- Second may be building some similar files together so that parsing of header, resources are done for few times which lowers the load on build server.
- Third Precompiled libraries can be created for portions of the code.

## 9.4. Multi-platform and multi-language support

Build Sever should support performing the build process on tests developed using more than one language such as C#, Java, C++, etc. Moreover, it should also support tests developed on multiple platforms such as Windows and Linux.

### Solution:

The build server needs to identify the language in which the tests are developed and then invoke the appropriate toolchain and compiler by setting appropriate environment variables. The build Server can do so by reading environment information and toolchain to be invoked from a XML file. This way, less time will be required for adding support for additional languages and platforms. Moreover, having a file which contains necessary information for setting

environment variables and invoking toolchain ensures that the developer of the build server will not have to make any changes to the code. This approach helps in maintaining the build server.

# 10. Drawbacks:

## 10.1. Limited language support

The build server developed in this project has support for performing build on code developed in C#. However, it does not support build operation on C++ and Java code. Multi-language support can be provided by configuring the build process to appropriate environment and invoking appropriate compiler to perform build operation.

## 10.2. Limited platform support

The build server does not support code built on platforms other than Windows. The build server does not have the ability to identify the platform on which the code is developed. It can do so if the build server is further enhanced by providing facility to identify the platform in which the code is built and by adding additional environment and compilation options.

# 11. References


- https://ecs.syr.edu/faculty/fawcett/handouts/webpages/fawcettHome.htm
- https://electric-cloud.com/plugins/build-automation
- https://msdn.microsoft.com/en-us/library/0k6kkbsd.aspx