



---

## Table of Contents

---

<b>1</b>	<b>Introduction.....</b>	<b>3</b>
1.1	Purpose .....	3
1.2	Scope .....	3
1.3	Definitions, Acronyms, Abbreviations .....	3
1.4	Design Goals .....	5
<b>2</b>	<b>References.....</b>	<b>6</b>
<b>3</b>	<b>Decomposition Description.....</b>	<b>7</b>
3.1	Module Decomposition .....	7
3.2	Concurrent Process .....	8
3.3	Data Decomposition .....	9
3.4	STATES .....	10
<b>4</b>	<b>Dependency Description.....</b>	<b>13</b>
4.1	Intermodule Dependencies .....	13
4.2	InterProcess Dependencies .....	14
4.3	Data Dependencies .....	15
<b>5</b>	<b>Interface Description .....</b>	<b>16</b>
5.1	Module Interface.....	16
5.2	Process Interface.....	21
<b>6</b>	<b>Detailed Design.....</b>	<b>23</b>
<b>7</b>	<b>Design Rationale.....</b>	<b>24</b>
7.1	Design Issues .....	24
<b>8</b>	<b>Traceability.....</b>	<b>26</b>

# 1 Introduction

## 1.1 PURPOSE

The purpose of this document is to describe how the CyChat project will be developed. This document specifies the different sub-systems of the project and the interfaces between them in order to concurrently develop different parts of the project. The data in this document is intended for the developers so that they know what exactly needs to be done for successful completion of the project.

## 1.2 SCOPE

This document defines the sub-systems of the project and outlines the services of each. Detailed information is given on the different processes, groupings and states of the project. The interfaces between the different sub-systems are exhaustively defined. Exact implementation of the sub-systems and their components is not given, but each sub-system's required behaviors, actions, and interfaces are specified.

## 1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description
Action area	An area inside of a room that has a server-side script associated with it. Instead of moving to that spot when the user clicks inside the action area, the server executes the script. The user has no way to tell if an area is an action area unless the background or images displayed by the server indicate that it is one.
Avatar	The visual appearance of a user. This can be changed by the user and is then made visible to all other users in the same room.
Bookmark	A saved configuration of a server that has been previously visited. A bookmark consists of server name, network address, and port.
Client	The program used by the user to connect to a server. It has a graphical interface to display information and the room in which the user is.

Closet	The local collection of Avatars available for wearing by a user. Avatars can be imported into the closet. This Closet is stored locally on the user's computer.
Dewear	To remove an avatar from use.
Image item	An image that can be seen by all users in a room that is not static like the room's background and is not user-controlled like an Avatar. They can be modified by action areas.
Room	A specific "channel" in a chat server. A user is in one room at one time on a server. Normally, messages sent go to all people in the same room as the sender (with the exception of whispers and global messages). Rooms are visually represented as a rectangle with a background image downloaded from the server.
Server	The program used by the developer to set up a common area for users to chat in. It manages all of the rooms and clients connected to it. "Server" may also be used as a general term for the collection of rooms and scripts that the server program is composed of.
Wear	To use an avatar. A user is said to be wearing an avatar when it is in use as the user's visual representation.
World	The collection of rooms hosted on a server

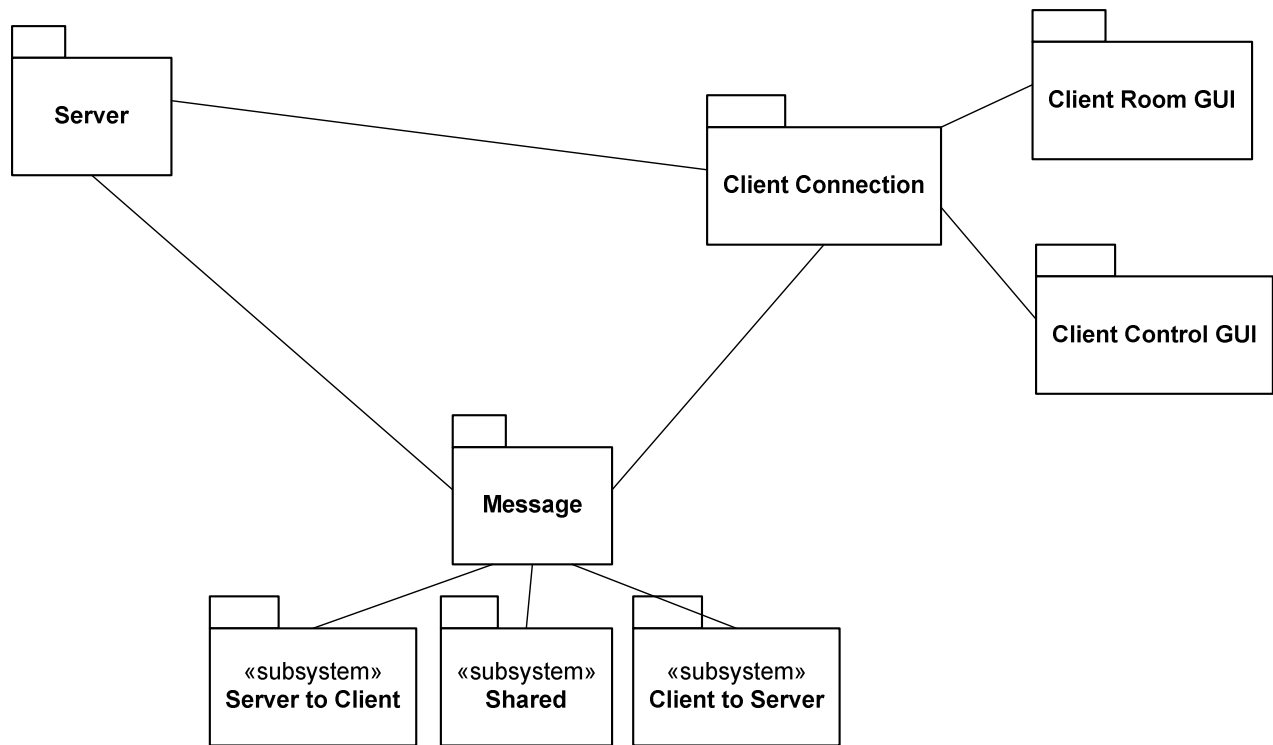
## 1.4 DESIGN GOALS

1. Extensibility – CyChat is intended to be a very versatile product. With the ability for people to develop their own servers and define everything from Room features to action areas, it is very important that the System Admin’s creativity is not limited.
2. Response Time – As this is a chat program, response time is obviously a very important issue. In development, lag and response time should be a primary concern. This refers to the loading time of rooms as well as response time for chat messages between users.
3. Reliability – The final product should be a well running product with no unexpected errors. Extensive testing, in collaboration with clean coding standards, should be employed to ensure this.
4. Maintainability – CyChat should be developed in order that it may be updated or expanded later on. New features may be desired at a later date, so it’s important to use good coding standards and sufficient documentation so that maintenance is possible.

## 2 References

## 3 Decomposition Description

### 3.1 MODULE DECOMPOSITION



#### 3.1.1 <Server>

##### *the server program*

- Server authenticates users
- Server manages world state data such as room creation and action areas
- Server relays messages to users
- Server activates scripts
- Server provides status information to Administrator

### 3.1.2 <Messages>

*contains objects that both the client and server use*

- Defines data types sent between server and client as messages such as Avatars, Image Items, ect.

### 3.1.3 <Client Connection>

*Part of the client that is connected to the server*

- Connects to server
- Relays messages from user to server
- Determines action based on messages from server

### 3.1.4 <Client Control GUI>

*GUI that handles inputs and outputs from the user except in the room window.*

- Reads user settings file
- Generates and displays menus and frames (including Room GUI)
- Takes user input (connect, room change, user list request, etc) and activates appropriate client connection methods
- Accepts user chat input

### 3.1.5 <Client Room GUI>

*GUI that handles the room window and all associated inputs*

- Displays state of the room including user avatars
- Displays chat bubbles
- Takes user click events and relays data to Client Connection
- Caches image items, avatars from other users, and backgrounds

## 3.2 CONCURRENT PROCESS

### 3.2.1 <Server >

*Communicates with the client and processes data*

- Main thread : waits for a client to connect
- Client Connection thread: runs when a client connects and waits for messages

### 3.2.2 <Client>

*Communicates with sever and interprets data sent*

- Main thread: launches GUI
- Server Connection Thread: run when the client connects to server and waits for server messages

## 3.3 DATA DECOMPOSITION

### 3.3.1 <Settings file>

*The settings file is used to keep track of information that the client needs to retain between instances.*

- The settings file is stored locally by the client
- Filename is cychatsettings
- If the file doesn't exist, it is created.
- File stores the last used server, port, and username
- File stores server bookmarks (name, address, port)
- The SettingsHandler class controls the settings file.
- File is read when the client opens, and is written when the client closes.

### 3.3.2 <Server World Data>

*The server's world state data will be defined in a composite class encapsulated by the ServerState class. The data will be populated by the server's configuration script file on server startup.*

- Server Name
- Server Message of the Day
- Maximum Server Capacity
- Room List

*Each room (defined by a RoomState object) will contain additional world data, also created by the configuration script:*

- Room Name
- Maximum Room Capacity
- Background File Name
- Action Area List

### 3.3.3 <Avatar Data>

*The Avatar data is all of the necessary data that makes up a users avatar in the world and also data to identify different avatars as unique.*

- Avatar Image
- Avatar Checksum
- Avatar Filename
- Avatar name

### 3.3.4 <Localization File>

*This file is an XML file containing translation definitions for GUI messages and titles. Users can create these XML files to create translations for messages and titles in CyChat.*

- Defines UTF format of XML to define Unicode format.
- Root node
- Name of language translation
- Specific translations of messages and titles in GUI

## 3.4 STATES

#### 3.4.1 <Logged Off >

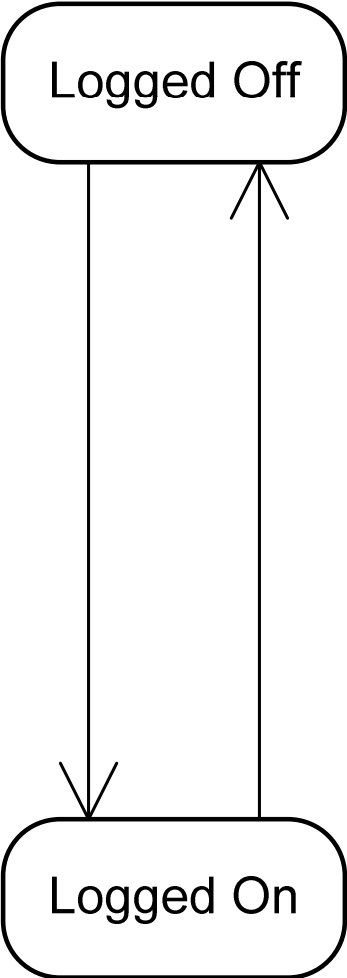
When a user is logged off, the only available option is to log on to a server. While logged off the user may still change languages, change or update avatars, and view debug logs but no chat or interactive functions are available.

#### 3.4.2 <Logged On>

When a user is logged on all functions are available to the user except changing server in which case they must first log off.

#### 3.4.3 <State Diagram>

///(!!! We could only think of 2 states for our program and asked professor Mitra if he thought it was ok and he agreed that we would really only have the 2.)



## 4 Dependency Description

### 4.1 INTERMODULE DEPENDENCIES

#### 4.1.1 Server

Uses data types defined in the Common package

Uses messages from Client Connection.

Provides server information to Client Connection.

Provides world state data to Client Connection.

Provides messages to Client Connection.

#### 4.1.2 Message

Provides data types to Server.

Provides data types to Client Connection.

#### 4.1.3 Client Connection

Uses data types defined in the Common package

Uses messages from the Server to determine appropriate action.

Provides state changes to Client Room GUI.

Provides messages to Client Room GUI.

Provides server information to Client Control GUI.

Uses user input from Client Control GUI.

Uses user input from Client Room GUI.

Uses messages from Client Control GUI.

Provides user input to Server

Provides messages to Server

#### 4.1.4 Client Control GUI

Uses Client Connection to retrieve server data such as room list, user list, etc.

Uses Client Room GUI to display the world.

Provides user input (connect, room change, user list request, etc.) to Client Connection.

Provides messages to Client Connection.

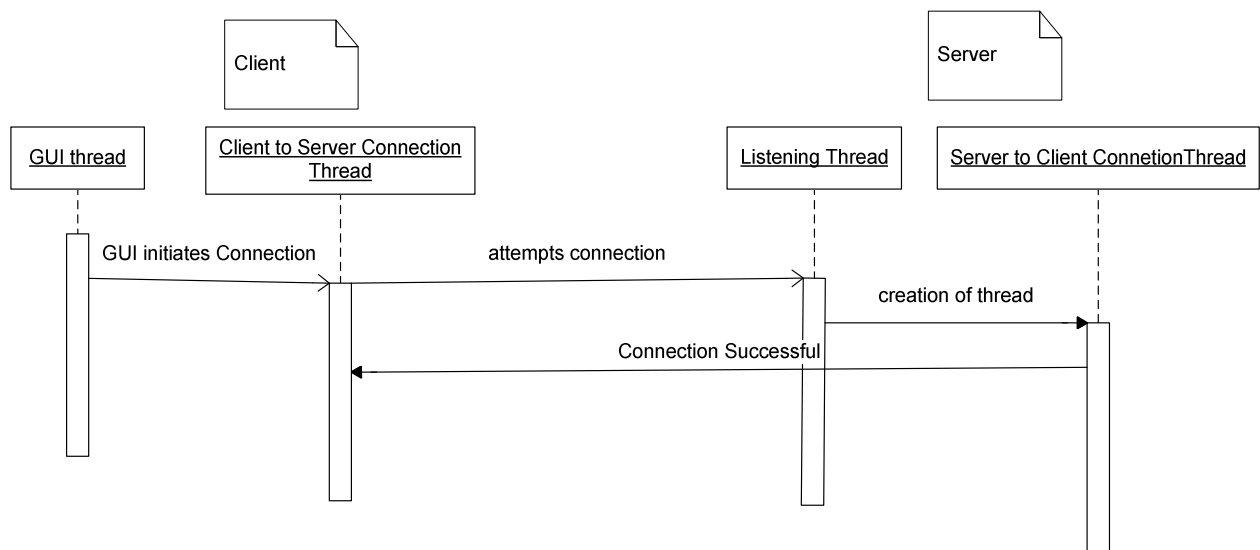
#### 4.1.5 Client Room GUI

Uses state data from Client Connection to display world.

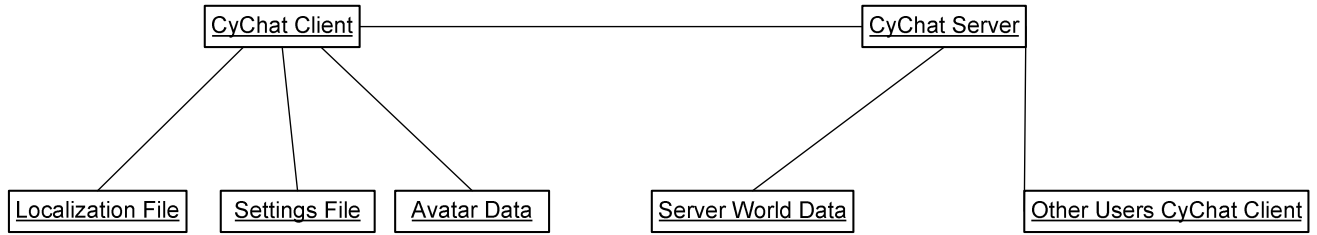
Uses messages from Client Connection to display chat messages.

Provides user click event data to Client Connection.

## 4.2 INTERPROCESS DEPENDENCIES



### 4.3 DATA DEPENDENCIES



## 5 Interface Description

### 5.1 MODULE INTERFACE

#### 5.1.1 <Server> Interface

- Server ↔ Client Connection

TCP/IP

Java Sockets

Object Serialization

#### **Protocol Description:**

*The server interface operates by sending java objects stored in the messages package through the connection between the server and client connection. These objects act as messages between the two and contain all the necessary information to operate. A description of what messages and how they are sent follows.*

(All points marked with bullets are objects stored in the messages package.)

To logon, client sends...

- ServerRxLogonInfo

Server replies with (if successful)...

- ClientRxServerInfo
- ClientRxRoomInfo
- ClientRxUserInfo

Or replies with (if unsuccessful)...

- ClientRxError

To send globals, whispers, and room chat, client can send...

- ServerRxChat

Server replies with...

- ClientRxChat (to all users intended to get the message and always to the sender)

To update the user's avatar, the client can send...

- ServerRxUserInfo

Server replies with...

- ClientRxUserInfo (to all users in room)

To get a current user listing, client can send...

- ServerRxUserListRequest

Server replies with...

- ClientRxUserList

To get a current room listing, client can send...

- ServerRxRoomListRequest

Server replies with...

- ClientRxRoomList

To change rooms, client can send...

- ServerRxRoomChange

Server replies with...

- ClientRxRoomInfo (to the moving user)
- ClientRxUserLeft (to all users in the old room)
- ClientRxUserInfo (to all users in the new room and the moving user)

To get uncached backgrounds and image items, client can send...

- ServerRxFileRequest

Server replies with...

- ClientRxFile

To perform moderator actions, client can send...

- ServerRxModAction

Server replies with (only when unsuccessful)...

- ClientRxError

To get uncached remote user avatars, client can send...

- AvatarRequest

Server replies with...

- Avatar

To get and distribute new user avatars, server can send...

- AvatarRequest

Client replies with...

- Avatar

- Server  $\longleftrightarrow$  Message

Creates Objects: the server creates instances of objects stored in the Message module

### 5.1.2 <Client Connection> Interface

- Client Connection  $\longleftrightarrow$  Server

TCP/IP

Java Sockets

Object Serialization

- Client Connection → Message

Uses data types: the client uses data types from the Message module

- Client Connection → Client Room GUI

Clear room: clears the room of all users

Set room name: sets the room name

Load room image: loads the background image of the room

Update user location: updates where the user's avatar is in the room

Update user avatar: updates any change in the users avatar

Remove user: removes the user from the room

Show room chat

Show global chat → (Currently show bubble): shows chat in a bubble on screen

Show whisper chat →

- Client Connection → Client Control GUI

Fatal server error: Server suffers a fatal error and has to shut down

Server error: server suffers an error

Update room list: updates the list of rooms in the world

Update user list: updates the list of users in the world

### 5.1.3 <Client room GUI>

- Client Room GUI → Client Connection

Send location: sends information about user's avatar's location

Request avatar: sends request for an avatar image that is not locally stored

Request file: sends request for a file

Request image item: send request for an image item that is not locally stored

#### 5.1.4 <Client Control GUI>

- Client Control GUI  $\longrightarrow$  Client Connection

Activates start: starts the client program

Activates stop: stops the client program

Change room: changes the room the user is in

Request room list: sends request for a list of rooms in the world

Request user list: sends request for a list of users in the world

Send chat: sends chat

#### 5.1.5 <Message>

- Server  $\longleftrightarrow$  Message

Creates Objects: the server creates instances of objects stored in the Message module

## 5.2 PROCESS INTERFACE

### 5.2.1 <Client>

*Implemented through subclassing of the built-in java.lang.Thread class.*

- GUI Thread (Event Dispatch Thread)

For the client's GUI, the Java Swing library will be used. This means that all actions that the client takes are run on the Swing Event Dispatch Thread. It is automatically started and stopped when the main program window is closed, which corresponds to the start and stop of the entire client program.

- Client-Server Connection Thread

Inside the Client Connection package, there will be a thread called "Server Connection Thread" that listens for messages from the server. It will be started by the Server Connection Handler in the Connection package whenever a connection is initiated by the Client Control GUI module. If the server requests a disconnect or there is a connection error, the thread will deactivate itself and inform the Server Connection Handler that the connection has terminated. If the Server Connection Handler requires a disconnect (when requested from the Control GUI module), it will close the connection stream and the thread receives a disconnect exception. When this occurs, the thread then ends itself.

Technically, the client has a third thread: the main Java thread. This thread is blocked throughout the course of the program because it blocks on the GUI Thread.

### 5.2.2 <Server>

*Implemented through subclassing of the built-in java.lang.Thread class.*

- Listening Thread

This is the server's primary thread, which will run throughout the course of the program (i.e. it will be started when the program is launched and end when terminated). It will listen for incoming client connection requests and, when it receives one, it will create a new Server-Client Connection Thread.

- Server-Client Connection Threads

These threads, defined in the Server package, receive and process incoming messages from connected clients. A new one is created by the Listening Thread for every client that connects to the server. It is ended similarly to the Client's analogous connection thread: it will terminate itself when such a request is made (when the server shuts down or due to a kick/ban command initiated by a different Connection thread) or when a disconnect condition (I/O exception) arises.

## 6 Detailed Design

JAVA DOCS will be used in our program to give a finalized detailed design

## 7 Design Rationale

### 7.1 DESIGN ISSUES

#### 7.1.1 <Avatars>

Issue: way to have user uploadable avatars.

Design issues were encountered with the passing and storing of avatars. These were encountered as a result of our design decision to allow users to use their own custom avatars, and the decision to not store them on the server. This makes it difficult as a user adding an avatar has to send it to the server and then distribute it to all clients that both are in the same room and do not already have the file locally cached on their machine. Once a client receives the file they then must cache it locally to be stored if they chat with the same person again. This requires use of a list of the checksums of the avatar images to track which ones are already owned and which are needed.

An alternative design to this would be to allow the avatars to be cached on the server but that would ruin our design of a highly mobile and easy to set up server.

#### 7.1.2 <Defining Interactive World Elements>

Issue: There needs to be a way to define world content.

Factors affecting issue:

Need to define action areas

Ease of use and Readability and Maintainability

Needs an existing extensive package for java

One way to implement world content is to use definition files such as XML files. Such a file would allow the server administrator to define rooms, room names, room backgrounds, etc. but not the interactive content. For such content (i.e. action areas), to fulfill the requirements of the Action Area to allow modification of “state variables”, it would seem natural to use a scripting language, which has variables, as opposed to XML. This allows the server administrator to combine the static and dynamic portions of the world into one configuration file. Additionally, existing functionalities

can be imported and reused. Thus we decided to use a scripting package for world definition. Jython. Jython is almost nearly completely implements python which means python scriptwriters are able to easily create Worlds and because python is popular, it is easy to find reference material for it. Python is very simple. Jython allows scriptwriters to directly use existing java classes within python code and allow CyChat to provide java objects for the scriptwriter to work with. LuaJava is an alternative to Jython with similar capabilities, however, it would require the CyChat developers to learn Lua as opposed to using our existing knowledge of python. Additionally, the developers are more familiar with using Jython. One advantage to Lua is that it is used widely in the gaming industry (eg. World of Warcraft) so that many scriptwriters are already familiar with it.

We picked Jython, it fits our requirements. It is able to define action areas naturally using object oriented features. It is easy to use, and widely known with many available resources. Finally, Jython is an extensive and well tested implementation.

## 8 Traceability

### 8.1.1

No	Use Case/ Non-functional Description	Subsystem/Module/classes that handles it
1		
2		

FEEL FREE TO ADD APPENDICES AS NEEDED. UPDATE TOC BEFORE SUBMITTING