Assignment-4 (Object-Oriented Structure Measurement)

Course Code:SE3209

Software Metrics

03/05/2023

Team Members:

1.	Ikra chowdhury Nowkshi	BFH1925019F
2.	Ishrat Jahan Rintu	BFH1925002F
3.	Roichuddin Rana	ASH1925003M
4.	Dhruba Kanti Bakshi	ASH1825018M

Noakhali Science and Technology University (NSTU)

Table of Contents

1. Measuring Coupling in Object-Oriented System

Coupling is a term used in software engineering to describe the degree to which different components or modules of a software system are interconnected or dependent on each other. In other words, coupling measures how much one part of the system depends on another part of the system to function correctly.

We can measure coupling in object-oriented system by

1. Coupling Between Object Class (CBO)
2. Response For Class (RFC)
3. Message Passing Coupling (MPC)

1.1 Coupling between Object Class

In object-oriented programming, coupling refers to the degree of interdependence between classes or objects. There are several types of coupling that can occur between object classes.

Measurement Process: We consider here

1. Methods Call
2. Class Extends

Table 1: Coupling between Object

Class	Number	Coupled WIth
userstructure	0	
accountdata	1	userstructure
chathistory	0	userstructure
objectread	1	userstructure
FileShow	0	
ObjectFileCreate	0	
createaccountWindow	4	ObjectFileCreate ButtonSound accountdata userstructure loginwindow
loginwindow	4	objectread personalchatroomui aaccountdata createaccountWindow

personalchatroomui	6	Chathistory Clintthread FileShow TictacMainFrame DictionaryMainFrame
serverroom	0	
TicTacToy	0	
TictacMainFrame	1	TicTacToy
clintsocket	0	
clintthread	0	
file_name_transfer	0	
clinthandaler2	1	serversocket
serversocket	1	clinthandaler2
AgeCal	0	
Converter	0	
DeleteListedWord	0	
Dictionary	0	
DictionaryMainFrame	5	ImplementWord DeleteListedWord WordSearch DisplayWord Dictionary
DisplayWord	0	
ImplementWord	0	
WordSearch	0	

1.2 Response for Class

In object-oriented programming, the response for a class refers to the actions or behavior that a class can perform in response to inputs or events. The level of coupling between a class and other classes in the system can affect its response, with low coupling promoting modularity, reusability, and maintainability. Good software design aims for low coupling and clear separation of concerns between classes.

Measurement Process:

we count RFC by counting the number of local methods plus the number of external methods called by local methods.

RFC = number of methods in the class **+** number of distinct method calls made by the methods.

accountdata: 10

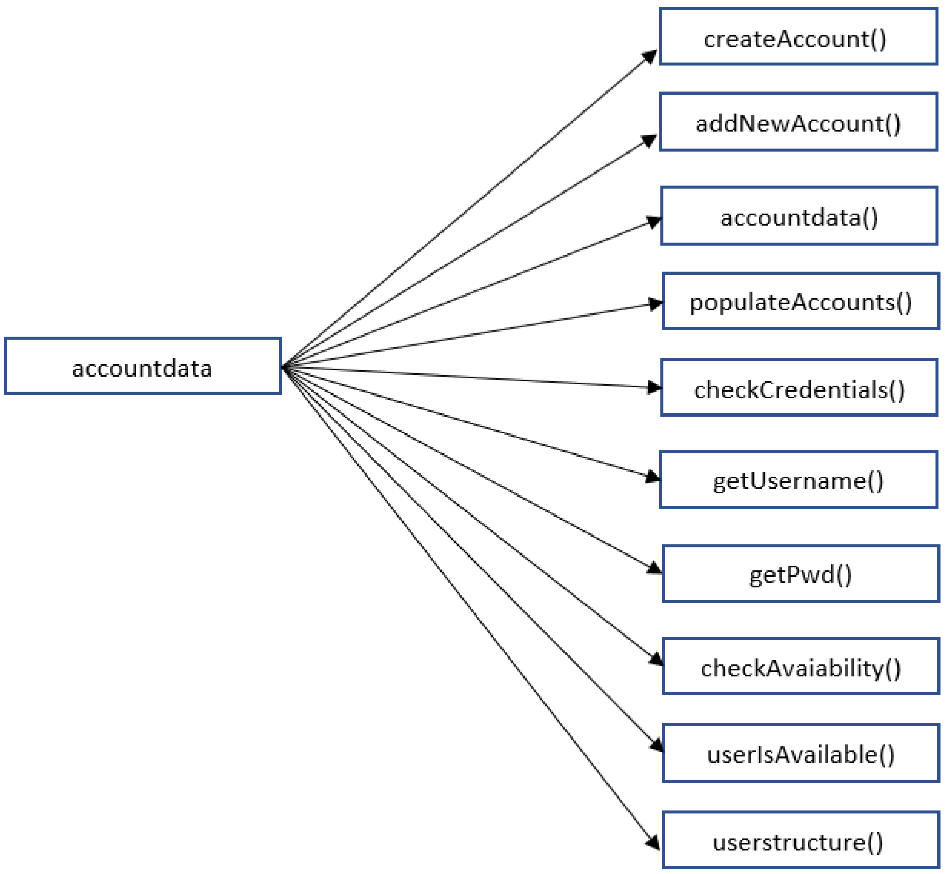

Figure 1: RFC (accountdata)

chatHistory: 4

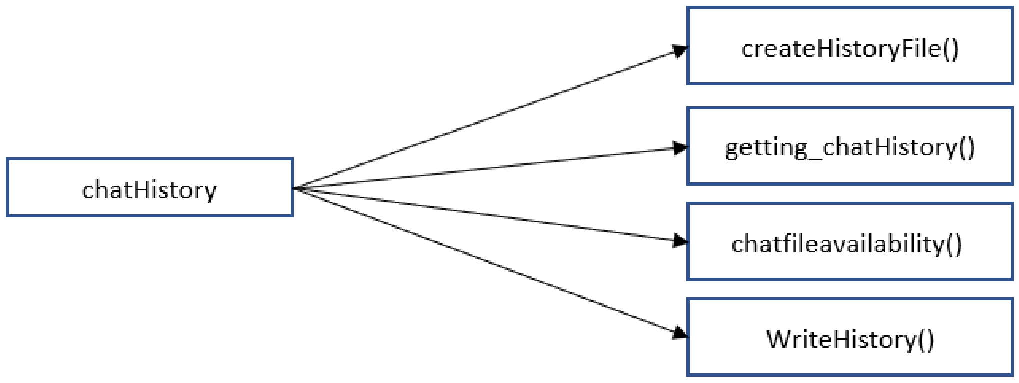

Figure 2: RFC (chatHistory)

objectread: 5

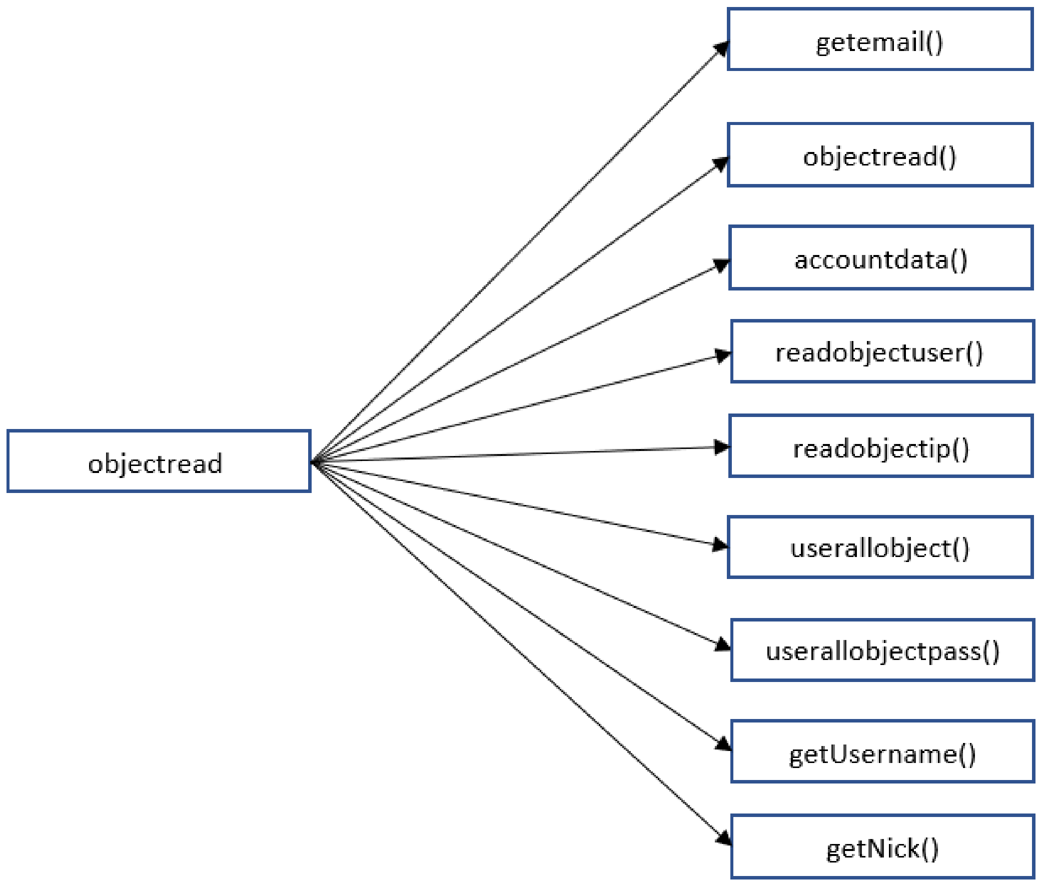

Figure 3: RFC (objectread)

createaccountWindow: 8

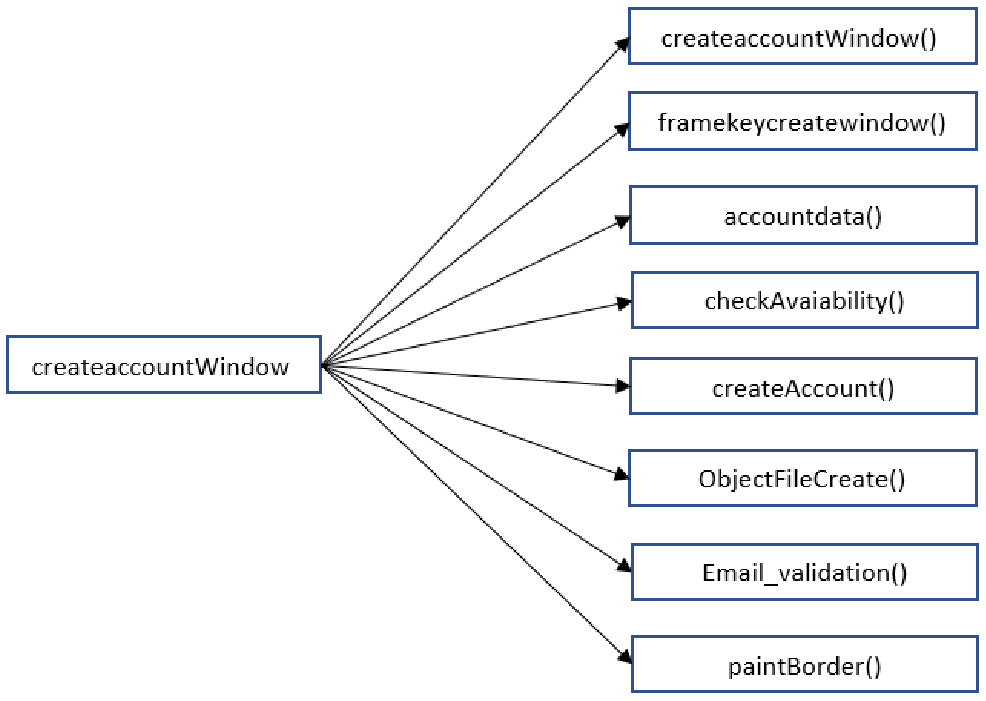

Figure 4: RFC (createaccountWindow)

FileShow: 3

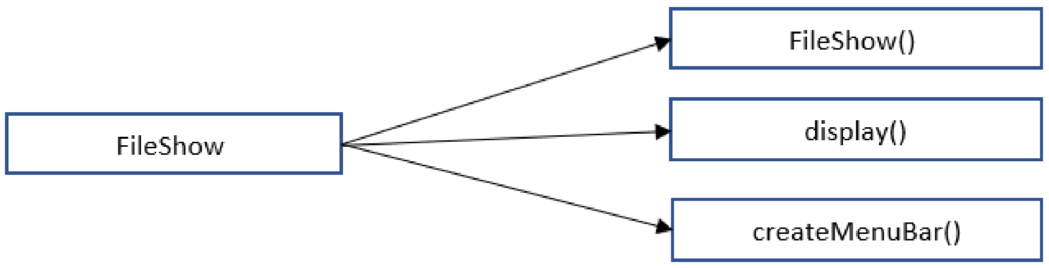

Figure 5: RFC (FileShow)

loginwindow: 6

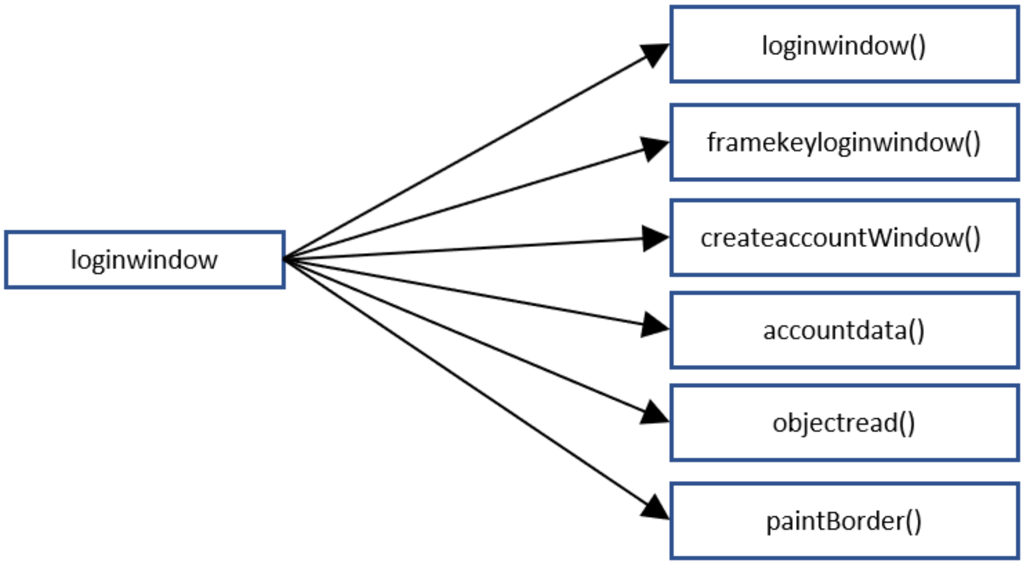

Figure 6: RFC (loginwindow)

ObjectFileCreate: 2

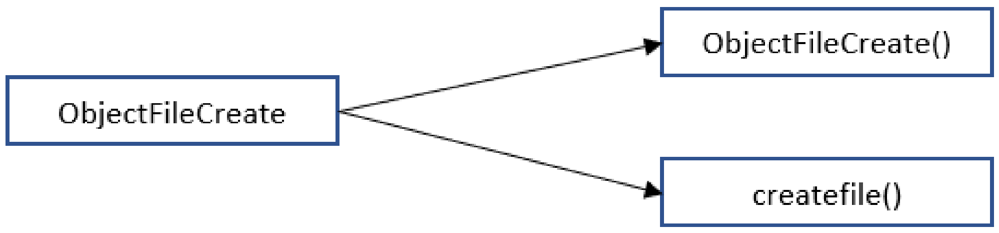

Figure 7: RFC (ObjectFileCreate)

personalchatroomui: 13

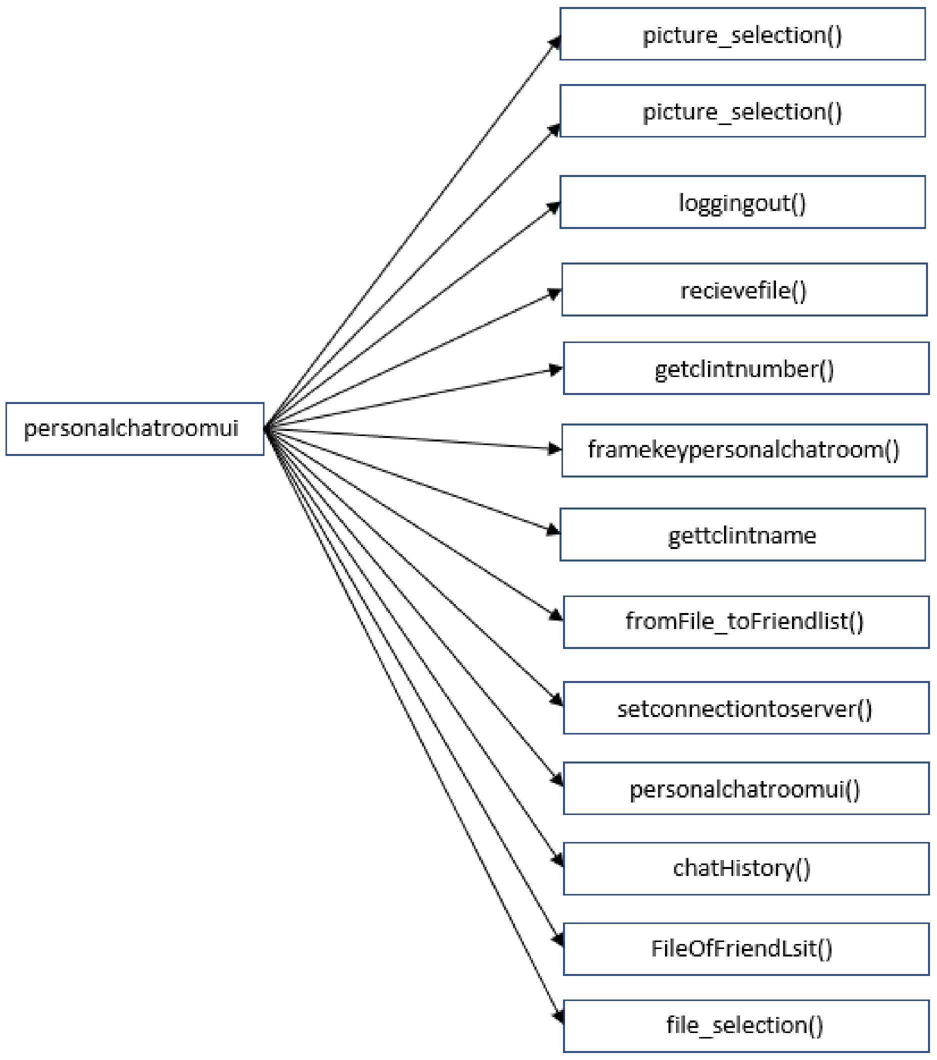

Figure 8: RFC (personalchatroomui)

DeleteListedWord: 1

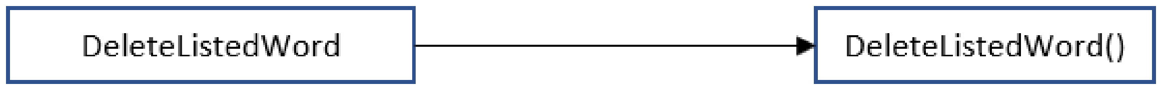

Figure 9: RFC (DeleteListedWord)

Dictionary: 8

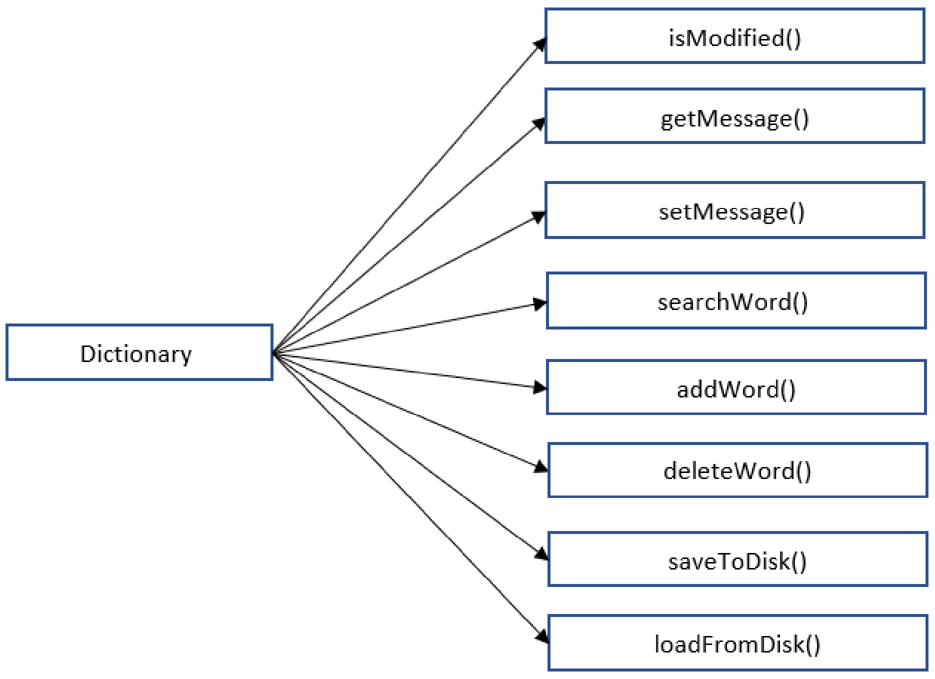

Figure10: RFC (Dictionary)

DictionaryMainFrame: 13

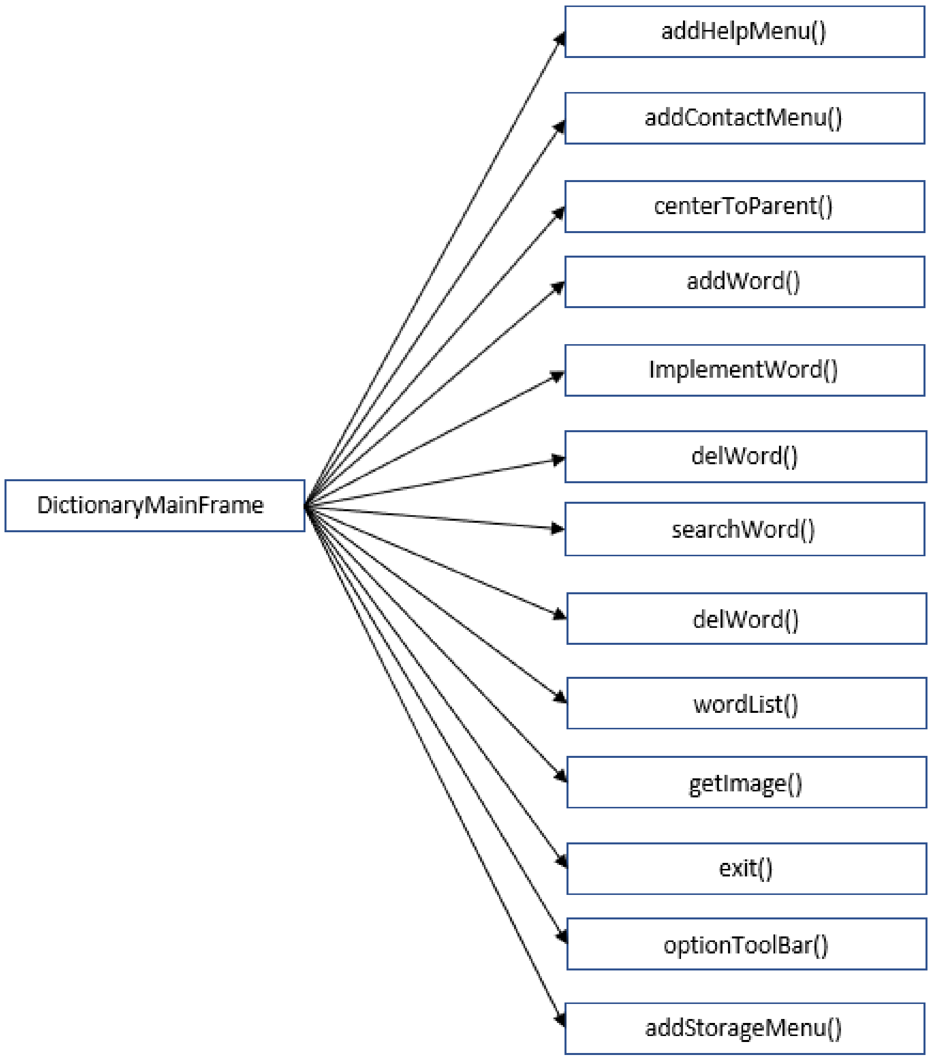

Figure 10:RFC (DictionaryMainFrame)

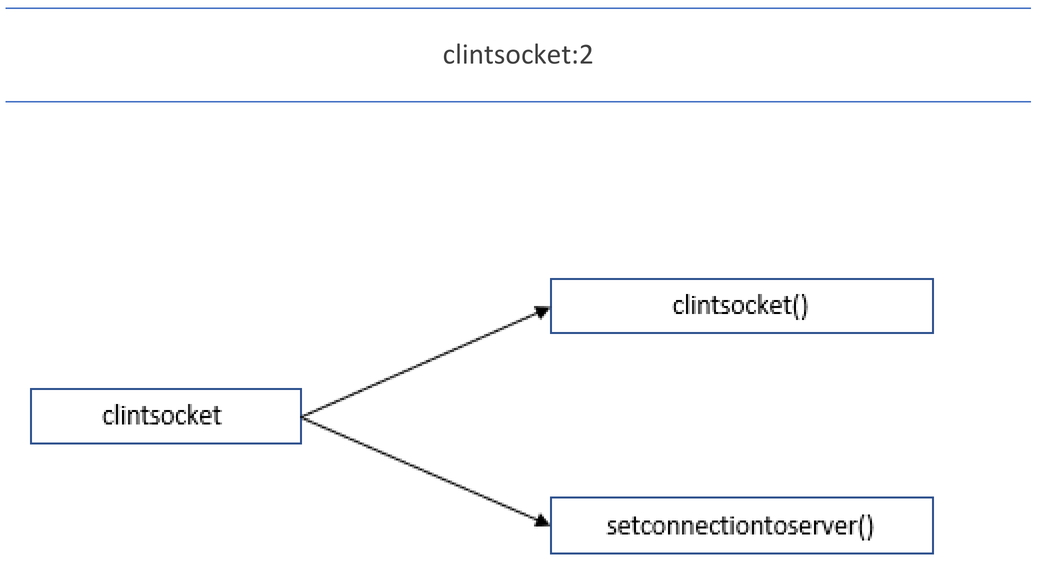

Figure 11:RFC(clintsocket)

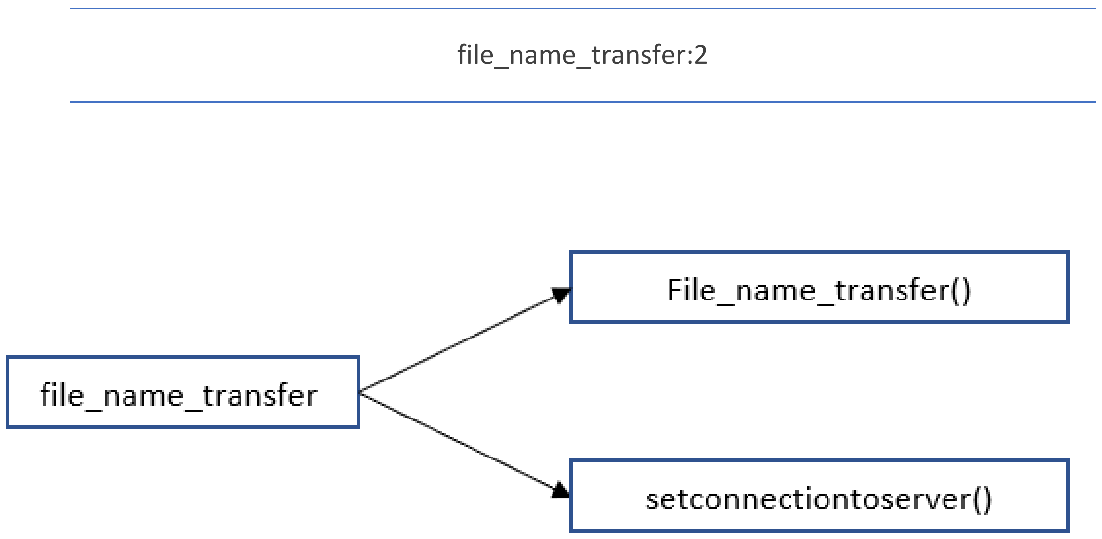

Figure 12:RFC(file_name_transfer)

clinthandler2:13

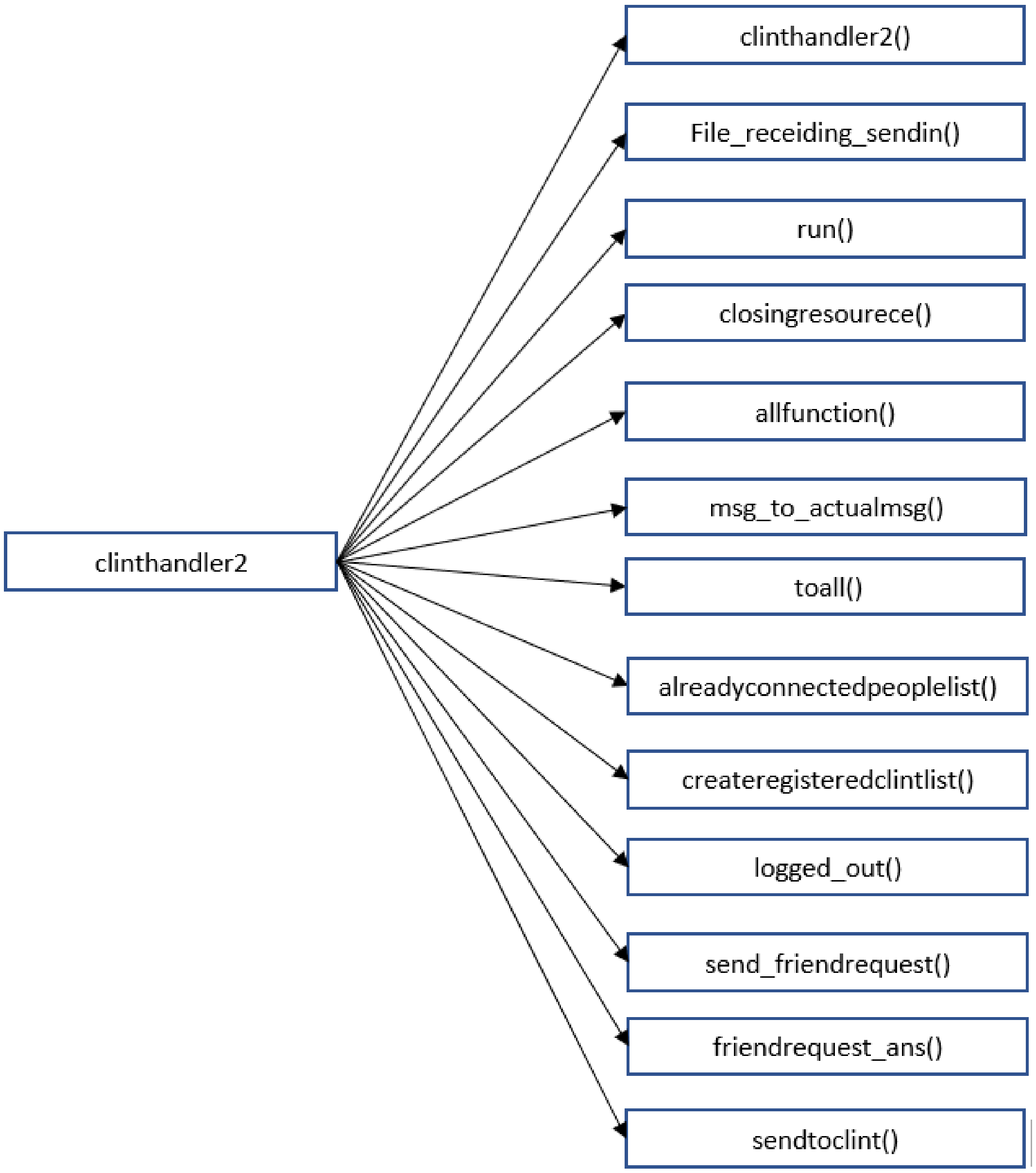

Figure 13:RFC(clinthandler2)

serversocket:2

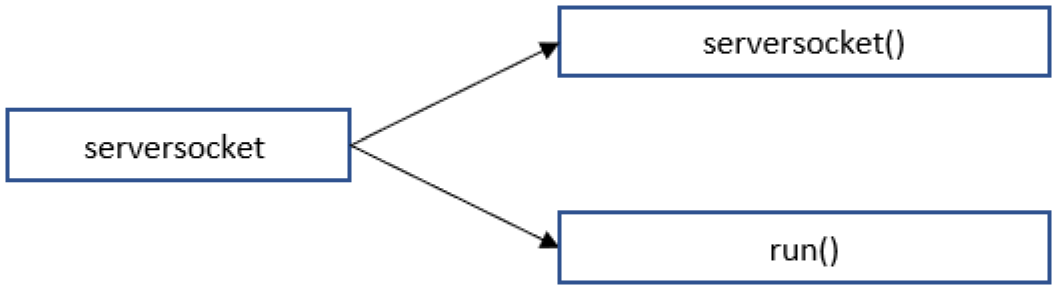

Figure 14:RFC(serversocket)

TictacMainFrame:4

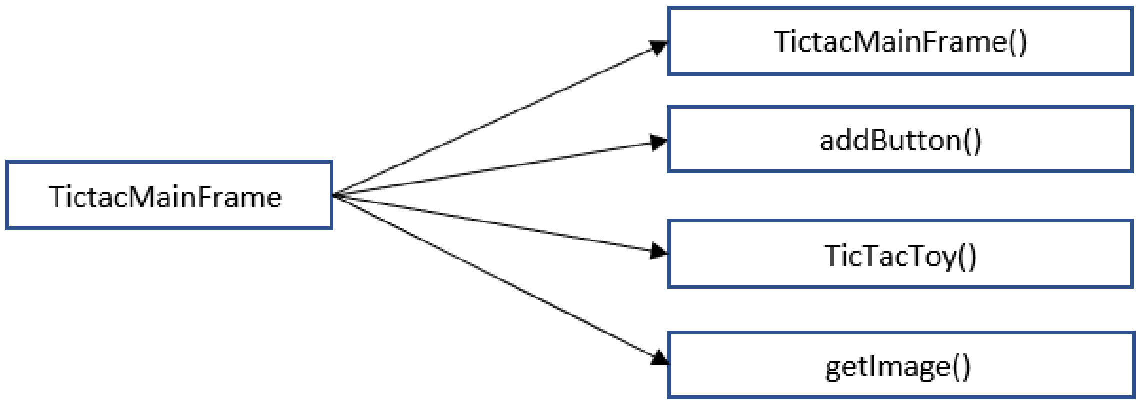

Figure 15:RFC(TictacMainFrame)

TicTacToy:5

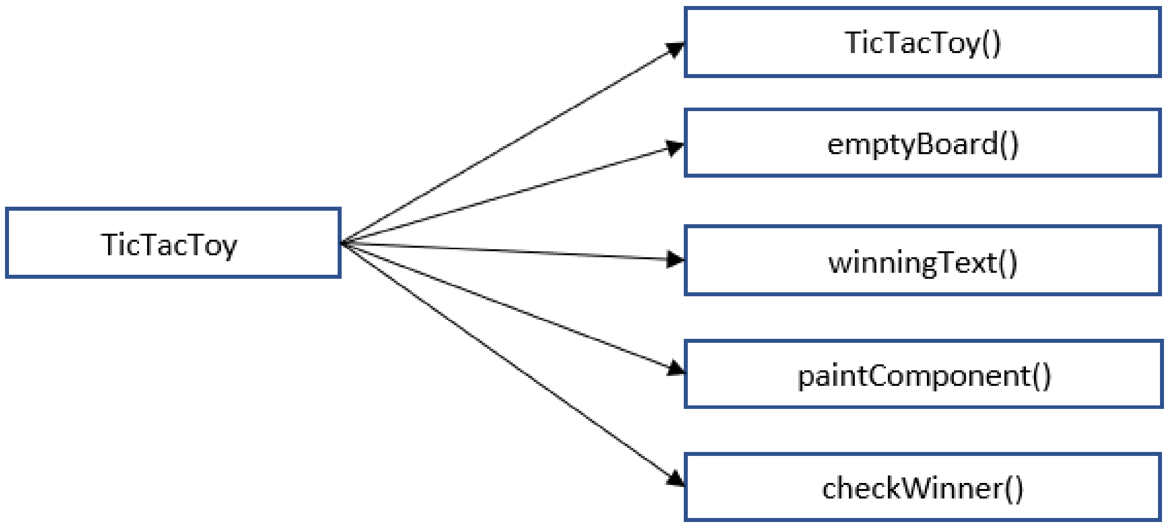

Figure 16:RFC(TicTacToy)

1.3 Message Passing Coupling

The MPC Metric measures the number of messages passed between modules, as well as the number of different message types that are used. This metric can be used to identify modules that are highly dependent on message passing and therefore may be more difficult to maintain or modify.

Measurement Process:

The MPC of each class is determined by adding up the total number of methods that are called within local methods of that class, but are not actually defined in that class.

- accountdata: 4

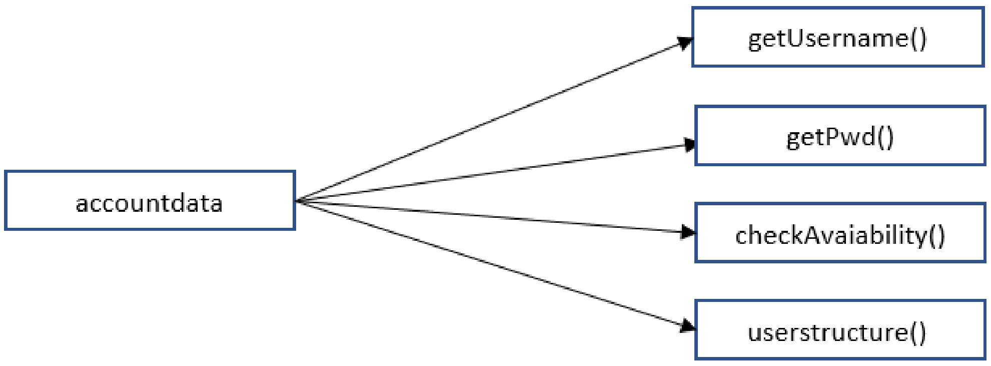

Figure 17: MPC (accountdata)

- chatHistory: 0
- objectread: 4

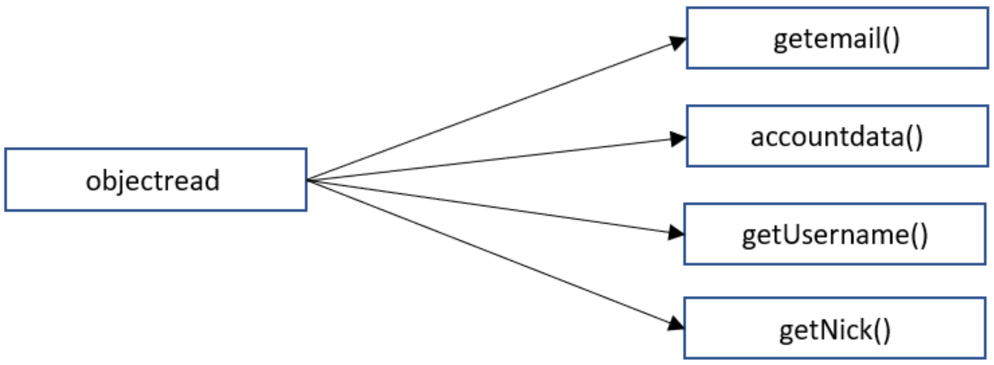

Figure 18: MPC (objectread)

- createaccountWindow: 3

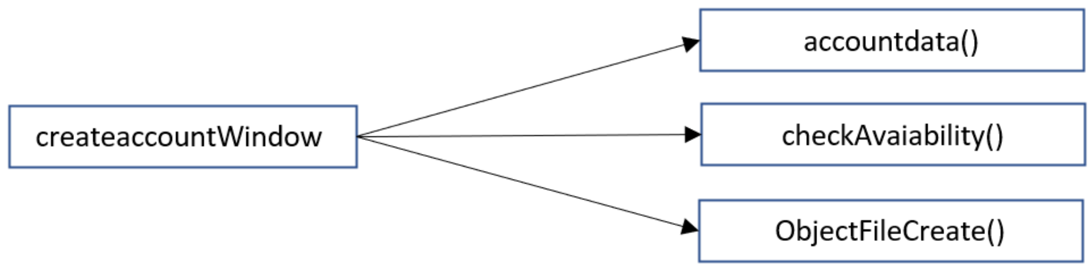

Figure 19: MPC (createaccountWindow)

- FileShow: 0
- loginwindow: 2

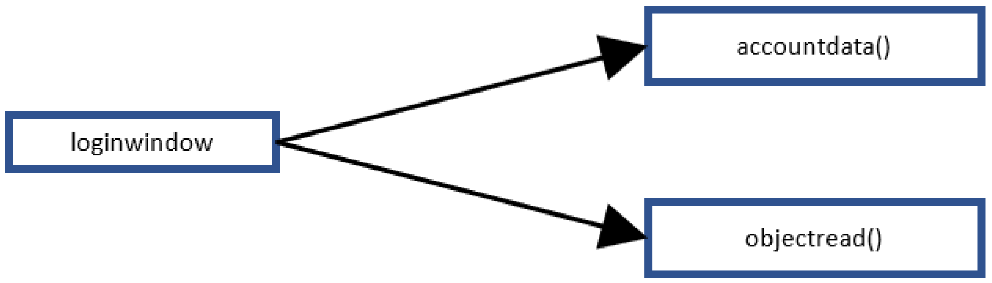

Figure 20: MPC (loginwindow)

- personalchatroomui: 1

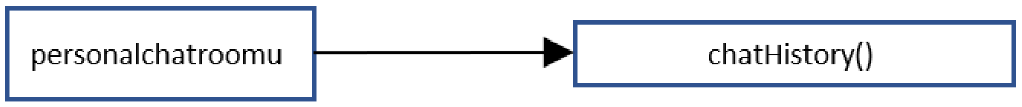

Figure 21: MPC (personalchatroomui)

- ObjectFileCreate: **0**
- DeleteListedWord: **0**
- Dictionary: **0**

- DictionaryMainFrame: **1**

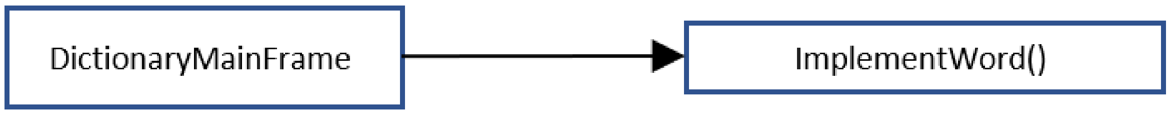

Figure 22:MPC(DictionaryMainFrame)

- Clintsocket:0

- file_name_transfer:1

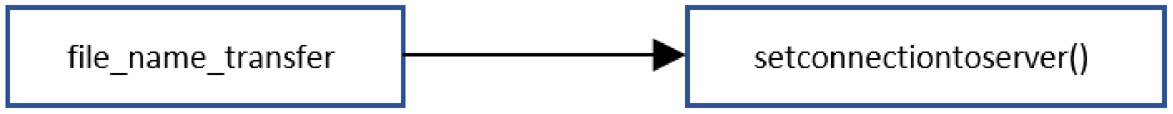

Figure 23:RFC(file_name_transfer)

- clinthandler2:0
- TictacMainFrame:0
- TicTacToy:0

2. Measuring Cohesion in Object-Oriented System

Cohesion in object-oriented programming refers to the degree to which the methods and properties of a class are related to each other and contribute to a single, well-defined purpose. Cohesion metrics measure how well the methods of a class are related to each other. cohesion refers to the degree to which the elements inside a module belong together. We can measure Cohesion in Object-Oriented System by LCOM (Lack of Cohesion Metric)

Measurement Process:

LCOM measures the degree of cohesion within a class by calculating the ratio of the number of methods that access the same instance fields to the total number of possible method pairs. A low LCOM value indicates high cohesion, meaning that the methods in the class work well together and share common instance fields. A high LCOM value suggests low cohesion, which may indicate that the class can be broken down into smaller, more cohesive classes.

LCOM = 1 – (sum(MF) / M*F)

Where,

- M is the number of methods in class
- F is the number of instance fields in the class.
- MF is the number of methods of the class accessing a particular instance field.
- Sum (MF) is the sum of MF over all instance fields of the class.

Table 2:Measuring Cohesion (LCOM)

Class	LCOM
userstructure	0.123
accountdata	0
chathistory	1
objectread	1
FileShow	0
ObjectFileCreate	0
createaccountWindow	0
loginwindow	0.5
personalchatroomui	0.192308
serverroom	0
TicTacToy	0.333
TictacMainFrame	0.4
clintsocket	1
clintthread	1
file_name_transfer	-1
clinthandaler2	0
serversocket	0
AgeCal	0
Converter	0
DeleteListedWord	0
Dictionary	0
DictionaryMainFrame	-1

DisplayWord	0
ImplementWord	0
WordSearch	0

3. Object-Oriented Length Measure

In general, length measurements show the separation between two elements. Distances in object-oriented systems depend on the viewpoint and the model used to describe the proper view of the system. We can gauge it using DIT (Depth Inheritance Tree)

Measurement Process:

DIT (Depth Inheritance Tree) can be measured by counting the number of classes in the inheritance hierarchy of a given class, excluding the starting class itself, and adding 1 to the count. We calculated it by using OpenStaticAnalyze tool.

Table 3:Object Oriented Length Measure

Class	DIT
userstructure	0
accountdata	0
chathistory	0
objectread	1
FileShow	0
ObjectFileCreate	0
createaccountWindow	0
loginwindow	0
personalchatroomui	1
serverroom	0
TicTacToy	0
TictacMainFrame	0
clintsocket	0
clintthread	1
file_name_transfer	1
clinthandaler2	0

serversocket	0
AgeCal	0
Converter	0
DeleteListedWord	0
Dictionary	0
DictionaryMainFrame	0
DisplayWord	0
ImplementWord	0
WordSearch	0

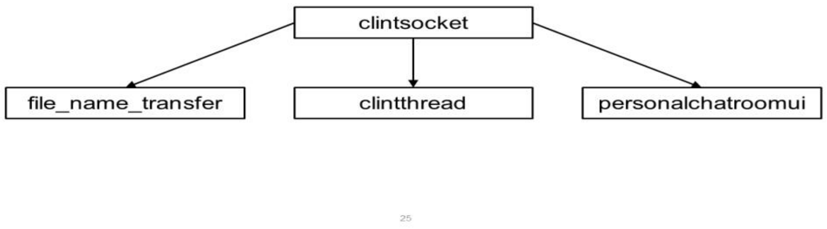

Figure 24: Depth of Inheritance

4. Object-Oriented Reuse Measurement

The NOC (Number of Children) metric is a measure of object-oriented reuse that can be used to assess the reusability of a class. A high NOC value can indicate that the class is highly reusable and serves as a useful abstraction for other classes. On the other hand, a low NOC value may indicate that the class is not as reusable or generic, and may be more specific to its current use case.

Measurement Process:

To measure NOC, you need to identify the class for which you want to calculate the metric and count the number of direct subclasses that inherit from this class. The number of direct subclasses is the NOC for that class.

Table 4: Number of Children

Class	NOC
userstructure	0
accountdata	0
chathistory	0
objectread	1
FileShow	0
ObjectFileCreate	0
createaccountWindow	0
loginwindow	0
personalchatroomui	0
serverroom	0
TicTacToy	0
TictacMainFrame	0
clintsocket	3
clintthread	0
file_name_transfer	0
clinthandaler2	0
serversocket	0
AgeCal	0
Converter	0
DeleteListedWord	0
Dictionary	0
DictionaryMainFrame	0

DisplayWord	0
ImplementWord	0
WordSearch	0