Unified Process Document Version Control System

Design Model 1.0

August 16, 2004

1 Deployment Model

As you can see in Figure 1 there is a server Odin on which the server part of the program is running. It is also possible that there is a client running on Odin too. For the server it makes no difference if the client is on that machine or any other. As you can see it should be possible to connect to the server by ≪LAN≫ or ≪Internet≫. For both ways we consider a Remote Message Invocation ≪rmi≫ for the communication. In this case we do not think about this and think that the client and the server are running in one. This diagram only shows the way it is going to work in the future.

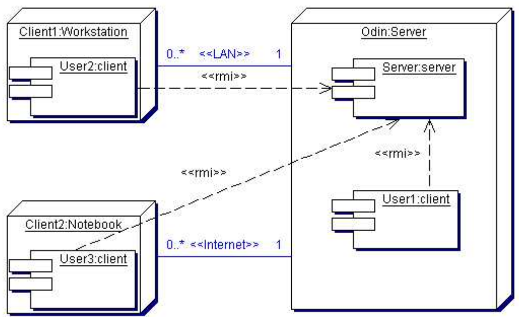

Figure 1: Deployment Diagram

2 Class Model

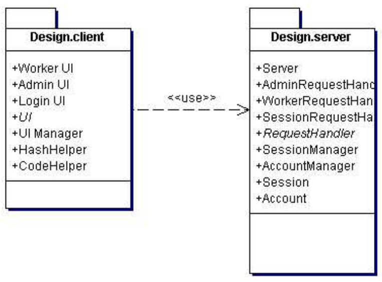

Figure 2: Package Diagram

2.1 Client Package

`Class Admin_UI`

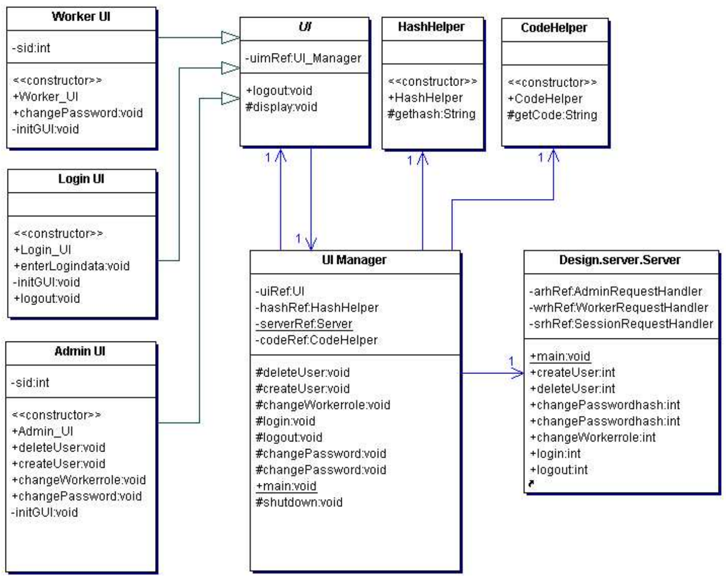

Figure 3: Client Class Diagram

package: client

```
client.UI
  |
  +-client.Admin_UI
```

public class Admin_UI
Extends:
 client.UI

This class is the interface for the Admin. Here the Admin can perform all of his activities.

Field Detail

sid

private int sid

This attribute is the session ID which is generated during the login procedure. The ID is transmitted to the server with every request.

Constructor Detail

Admin_UI

public Admin_UI(int sid)

This operation is the constructor of the Admin_UI class.

Stereotype:

```
  constructor


Method Detail


changePassword

public void changePassword(String username, String newpassword)

This operation changes a password of an existing account.

mod_Visibility:
  $public

mod__returnType:
  void


changeWorkerrole

public void changeWorkerrole(String username, Vector newroles)

This operation changes workerroles in an existing account.

mod_Visibility:
  $public

mod__returnType:
  void


createUser

public void createUser(String username, String password, Vector workerrole)

This operation is used to start the create user process.
There for it needs a username, a password, and the workerroles of the user who
is going to be created.

mod_Visibility:
  $public

mod__returnType:
  void


deleteUser

public void deleteUser(String username)

This operation starts the delete user process.
It needs the username of the user, who is going to be deleted.

mod_Visibility:
  $public

mod__returnType:
  void
```

```
initGUI

private void initGUI(int sid)

This operation initializes the Admin interface.
 It is called by the constructor.

mod__returnType:
  void

mod_Visibility:
  $private

Class CodeHelper

package: client

public class CodeHelper

The Server returns always a code.
This class translates the int return value into the appropriate message.


Constructor Detail


CodeHelper

public CodeHelper()

 Stereotype:
  constructor


Method Detail


getCode

protected String getCode(int var)

This operation is called by the UI_Manager class and returns the appropriate
message as a String.

Class HashHelper

package: client

public class HashHelper

The password is transferred as a Hash.
Because of this the UI_Manager class asks this HashHelper to generate the Hash
from the entered password.


Constructor Detail


HashHelper

public HashHelper()
```

```
 Stereotype:
  constructor


Method Detail


gethash

protected String gethash(String password)

This operation returns a hash from a given string.

mod__static

mod_Visibility:
  $protected

mod__returnType:
  String

Class Login_UI

package: client

client.UI
  |
  +-client.Login_UI


public class Login_UI
Extends:
 client.UI

This class represents the login interface of the software.
If you are or a Worker or a Admin you need to login yourself over this interface.


Constructor Detail


Login_UI

public Login_UI()

This is the constructor of the Login_UI class.

Stereotype:
  constructor


Method Detail


enterLogindata

public void enterLogindata(String username, String password)

 This operation is called after the user enters his logindata.
 This operation starts the login process.
```

```
mod_Visibility:
  $public

mod__returnType:
  void


initGUI

private void initGUI()

This operation initializes the interface.

mod__returnType:
  void

mod_Visibility:
  $private


logout

public void logout()

This operation overrides the operation which is inherited by the implemented UI class.
This is caused by the fact, that the login interface needs no possibility to logout.
This function starts the shutdown process on the client.

 mod__returnType:
  void

Class UI

package: client

public abstract class UI

This class is the basis for all user interfaces.
It provides the basic functions.

mod__abstract


Field Detail


uimRef

private UI_Manager uimRef

This reference is to the manager of the client package.
It is used to call the requested operations.


Method Detail


display

protected void display(String whattodisplay)
```

```
This operation displays any String message on an interface.

mod_Visibility:
  $protected

mod__returnType:
  void


logout

public void logout()

This operation calls the logout procedure.
It is used by all user interfaces.

mod_Visibility:
  $public

mod__returnType:
  void

Class UI_Manager

package: client

public class UI_Manager

This class is the manager of the client package.
It calls the server when needed and knows when to start which user interface.


Field Detail


codeRef

private CodeHelper codeRef

This attribute is a reference to the CodeHelper class of this client.


hashRef

private HashHelper hashRef

This attribute is a reference to the HashHelper class of this client.


serverRef

private static Server serverRef

This attribute is the reference to the server.


uiRef

private UI uiRef
```

```
This attribute is a reference to the actual user interface.


Method Detail


changePassword

protected void changePassword(String username, String newpassword, int sid)

This is one of two change password operations.
This one is called by the admin ui.
It needs the username of the account where the password is going to be changed,
the new password and the admin's sid. The password will be hashed before it
will be transmitted.

mod_Visibility:
  $protected

mod__returnType:
  void


changePassword

protected void changePassword(String newpassword, int sid)

This is one of two change password functions.
This one is called by the worker ui.
It only needs the new password and the sid.
The password will be hashed before it will be transmitted.

mod_Visibility:
  $protected

mod__returnType:
  void


changeWorkerrole

protected void changeWorkerrole(String username, Vector newroles, int sid)

This operation is called from the user interface.
It performs the rolechange of all role from one worker.
At the end of the process the display operation of the user interface will be called
and an appropriate message will be shown.

mod__returnType:
  void

mod_Visibility:
  $protected


createUser

protected void createUser(String username, String password, Vector workerrole, int sid)

This operation is called by the user interface and performs a create user process.
At the end of the process the display operation of the user interface will be called
```

```
and an appropriate message will be shown.

mod__returnType:
  void

mod_Visibility:
  $protected


deleteUser

protected void deleteUser(String username, int sid)

This operation is called by the user interface and performs a delete user
process. At the end of the process the display operation of the user interface
will be called and an appropriate message will be shown.

mod__returnType:
  void

mod_Visibility:
  $protected


login

protected void login(String username, String password)

This operation performs the login request. It hashes the password and calls
the server.

mod_Visibility:
  $protected

mod__returnType:
  void


logout

protected void logout(int sid)

This operation calls the server and shuts down the worker or admin ui and starts
the login ui.

mod__returnType:
  void

mod_Visibility:
  $protected


main

public static void main()

This is the main function which starts the client.

mod__static

mod__returnType:
  void


shutdown

protected void shutdown()

this function shuts the client down.

mod__returnType:
  void

mod_Visibility:
  $protected

Class Worker_UI

package: client

client.UI
  |
  +-client.Worker_UI


public class Worker_UI
Extends:
 client.UI

This class is the interface for the Worker. Here the Worker can perform all of
his activities.

Author:
Christian

Field Detail

sid

private int sid

This attribute is the session ID which is generated during the login procedure
and is transferred with every request to the server.


Constructor Detail


Worker_UI

public Worker_UI(int sid)

This constructor starts the initGUI() operation.

Stereotype:
  constructor


Method Detail


changePassword
```

```
public void changePassword(String newpassword)

This operation allows the Worker to change his password.

mod_Visibility:
  $public

mod__returnType:
  void


initGUI

private void initGUI(int sid)

This operation initializes the Worker interface. It also sets the sid.

mod__returnType:
  void

mod_Visibility:
  $private
```

2.2 Server Package

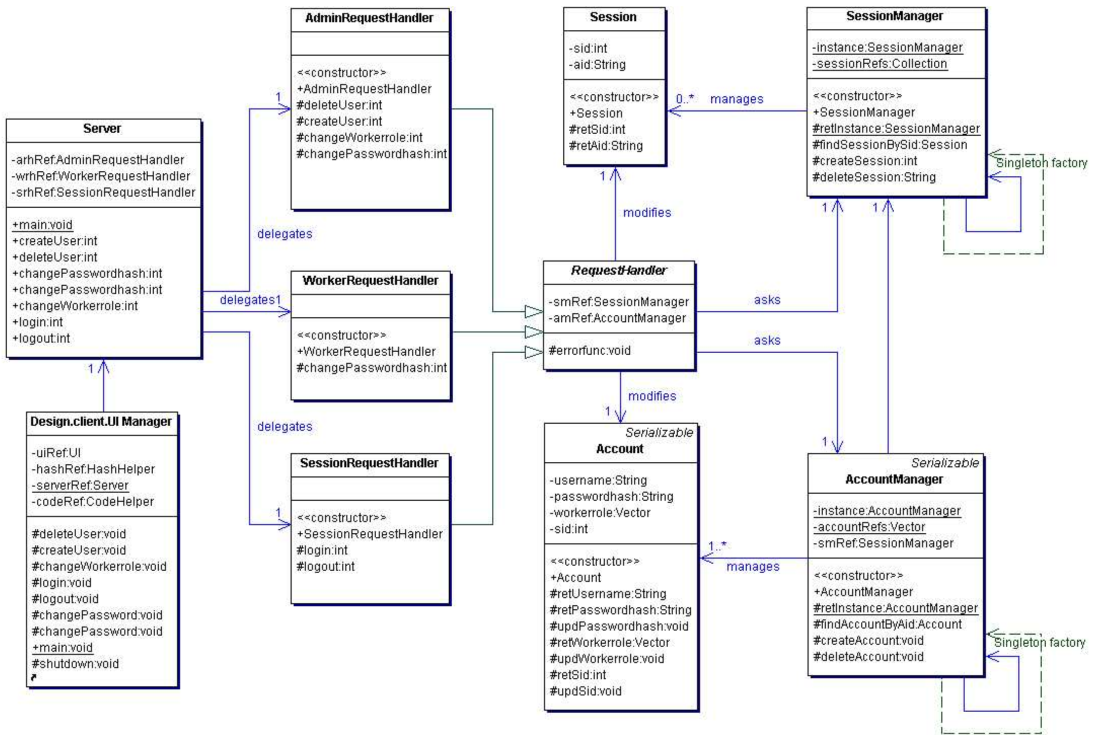

Figure 4: Server Class Diagram

```
Class Account

package: server

public class Account
```

```
Implements:
 java.io.Serializable

This is the Account class. For every valid user the system has one Account instance.


Field Detail


passwordhash

private String passwordhash

 This is the passwordhash, which belongs to the user.


sid

private int sid

 This is the sid. This field is only used, if the user is logged into the system.
 If the user is not logged in the value is null


username

private String username


workerrole

private Vector workerrole


Constructor Detail


Account

public Account(String username, String passwordhash, Vector workerrole, int sid)

 Stereotype:
  constructor


Method Detail


retPasswordhash

protected String retPasswordhash()

 This operation returns the Passwordhash to the caller.


retSid

protected int retSid()

 This operation returns the Sid stored in the Account.
```

```
retUsername

protected String retUsername()

 This operation returns the Username to the caller.


retWorkerrole

protected Vector retWorkerrole()

 This operation returns the Workerroles to the caller.


updPasswordhash

protected void updPasswordhash(String passwordhash)


updSid

protected void updSid(int sid)

 This operation sets a sid to the Account.


updWorkerrole

protected void updWorkerrole(Vector workerrole)

 This operation sets a new Workerroles to the Account.

Class AccountManager

package: server

public class AccountManager
Implements:
 java.io.Serializable

This AccountManager is managing the Accounts. Therefore it has a vector of
references to the Accounts.


Field Detail


accountRefs

private static Vector accountRefs

 This is a vector which contains all references to valid Accounts.


instance

private static AccountManager instance = null

 Because the Account Manager is a Singleton it has a reference to him.
```

lnkAccount

```
private Account lnkAccount
```

smRef

```
private SessionManager smRef
```

```
 This is a reference to the SessionManager class, because in some cases it can
 happen, that the AccountManager needs one of the functionalities of the
 SessionManager.
```

Constructor Detail

AccountManager

```
public AccountManager()
```

```
 Stereotype:
  constructor
```

Method Detail

createAccount

```
protected void createAccount(String username, String passwordhash, Vector workerrole, int sid)
```

```
 This operation is called to create a new account. At first it checks if
 the account does exist at all and if the account does not exist the function
 calls the constructor of the account class. The sid value is set to null at
 the creation.
```

deleteAccount

```
protected void deleteAccount(String username)
```

```
This operation is called whenever any class wants to delete an existing account.
At first it searches for the account and after that it checks if the account is
active at the moment. If the account is active it deletes the account but before
it calls the SessionManager to delete the Session of the user.
```

findAccountByAid

```
protected Account findAccountByAid(String username)
```

```
 This operation returns a reference to an Account.
 It finds the wanted Account by the give username.
```

retInstance

```
protected static AccountManager retInstance()
```

```
 Whenever any class needs a reference of this class this operation is called.
 It returns a reference of the existing object.

Class AdminRequestHandler

package: server

server.RequestHandler
  |
  +-server.AdminRequestHandler


public class AdminRequestHandler
Extends:
 server.RequestHandler

This class performs all requests regarding the admin. It is a realization of
the UseCase manage account.


Constructor Detail


AdminRequestHandler

public AdminRequestHandler()

 Stereotype:
  constructor


Method Detail


changePasswordhash

protected int changePasswordhash(String username, String newpasswordhash, int sid)

 This operation performs the change password request. At first it checks if the
 requesting user is the admin. When this is done it asks the AccountManager for
 the Account and modifies the password. At the end it returns an info code to
 the calling class.


changeWorkerrole

protected int changeWorkerrole(String username, Vector newroles, int sid)


createUser

protected int createUser(String username, String passwordhash, Vector workerrole, int sid)

 This operation performs the create user request. Like with all requests it checks first;
  if the requesting user is the admin. After this it asks the AccountManager to create
  a new Account. At the end it returns an info message.


deleteUser

protected int deleteUser(String username, int sid)
```

This operation performs the delete user request. At first it checks if the requesting user is the admin. After that it asks the AccountManager to delete an Account.

Class RequestHandler

package: server

public abstract class RequestHandler

This Class is the mother of all RequestHandlers. They all need a Reference to the SessionManager, the AccountManager. And if the Managers return a type they also need a possibility to handle the returned Account or Session.

Field Detail

amRef

private AccountManager amRef

This is a reference to the AccountManager

aRef

private Account aRef

smRef

private SessionManager smRef

This

sRef

private Session sRef

Method Detail

errorfunc

protected void errorfunc(int errorcode)

This error function is called, whenever something is going wrong. It generates an error code which will be transferred to the calling client.

Class Server

package: server

public class Server

This class is a kind of interface for the package server. Every request has to pass this class. It decides which of the request handlers is going to process

```
the incoming request.


Field Detail


arhRef

private AdminRequestHandler arhRef

 This attribute is a reference to the AdminRequestHandler.


srhRef

private SessionRequestHandler srhRef

 This attribute is a reference to the SessionRequestHandler.


wrhRef

private WorkerRequestHandler wrhRef

 This attribute is a reference to the WorkerRequestHandler.


Method Detail


changePasswordhash

public int changePasswordhash(String username, String newpasswordhash, int sid)

 This is an overloaded operation with which the admin can change a user’s password.
 This will be directed to the AdminRequestHandler.


changePasswordhash

public int changePasswordhash(String newpasswordhash, int sid)

 This is an overloaded operation, with which a worker can change his password.
 This will be directed to the WorkerRequestHandler.


changeWorkerrole

public int changeWorkerrole(String username, Vector newroles, int sid)

 This changeWorker() call will be directed to the AdminRequestHandler.


createUser

public int createUser(String username, String passwordhash, Vector workerrole, int sid)

 The incoming call createUser() will be directed to the AdminRequestHandler.


deleteUser
```

public int deleteUser(String username, int sid)

The incoming call deleteUser() will be directed to the AdminRequestHandler.

login

public int login(String username, String passwordhash)

This login() call will be directed to the SessionRequestHandler.

logout

public int logout(int sid)

This logout() call will be directed to the SessionRequestHandler.

main

public static void main(String argv)

This is the main function of the server. This function "starts" the server.

Class Session

package: server

public class Session

This is the Session Class. Whenever a user logs him in a session object is
created from this class. And whenever any user logs him out the session must
be destroyed.

Field Detail

aid

private String aid

This aid is another name for the username, which belongs to the session.
It is needed; if a user wants to change his password the session check returns
the aid to find his Account.

sid

private int sid

This is the sid. It is a number with which the calling user is identified in
the server.

Constructor Detail

Session

```
public Session(int sid, String aid)

 Stereotype:
  constructor


Method Detail


retAid

protected String retAid()

 This operation returns the aid to the calling user.


retSid

protected int retSid()

 This operation returns the sid to the caller

Class SessionManager

package: server

public class SessionManager

This is the SessionManager. As the name says it is the class which manages the
sessions. They are stored in a collection and the manager only knows the
possibilities to find, destroy, or create a session.


Field Detail


instance

private static SessionManager instance = null

 The SessionManager is a Singleton factory and because of this he has a
 reference to himself.


lnkSession

private Session lnkSession


sessionRefs

private static Collection sessionRefs

 This is the collection, which contains all valid sessions in the system.


Constructor Detail


SessionManager
```

```
public SessionManager()

 Stereotype:
  constructor


Method Detail


createSession

protected int createSession(String username)

 This operation performs a create session request. It generates a sid,
 calls the constructor of the session class, and returns the sid to the caller.


deleteSession

protected String deleteSession(int sid)

 This operation is called whenever a session has to be deleted.
 It searches the session, asks it for the aid, and deletes the reference
 from the collection that the garbage collector can delete the object.


findSessionBySid

protected Session findSessionBySid(int sid)

 This operation is called when any class needs the reference to a session.
 It searches the session by her sid and returns a reference.


retInstance

protected static SessionManager retInstance()

 Whenever any class needs a reference of this class this operation is called.
 It returns a reference of the existing object.

Class SessionRequestHandler

package: server

server.RequestHandler
  |
  +-server.SessionRequestHandler


public class SessionRequestHandler
Extends:
 server.RequestHandler

This class gets all requests regarding to the session handling.


Constructor Detail
```

```
SessionRequestHandler

public SessionRequestHandler()

 Stereotype:
  constructor


Method Detail


login

protected int login(String username, String passwordhash)

 This operation gets all login calls from the users.
 After the call is checks the account, compares the password, and asks
 the SessionManager to create a new session objec. At the end the generated
 session id is transferred to the calling class.


logout

protected int logout(int sid)

 This operation gets all logout calls from the users of the system.
 Then it asks the SessionManager to delete the session and after that it sets
 the sid, which is stored in the account to null. At the end it transfers a
 message code to the calling class.

Class WorkerRequestHandler

package: server

server.RequestHandler
  |
  +-server.WorkerRequestHandler


public class WorkerRequestHandler
Extends:
 server.RequestHandler

This class receives all requests regarding the Worker.


Constructor Detail


WorkerRequestHandler

public WorkerRequestHandler()

 Stereotype:
  constructor


Method Detail


changePasswordhash
```

```
protected int changePasswordhash(String newpasswordhash, int sid)

 This operation performs the change password request from a user.
 It does not check is the requesting user is the calling user.
 It just changes the password in the account of the calling user, which is
 identified by his sid.
```

3 Use-Case Realisation

In this part are all sequences shown which must be possible regarding the Use- Cases.

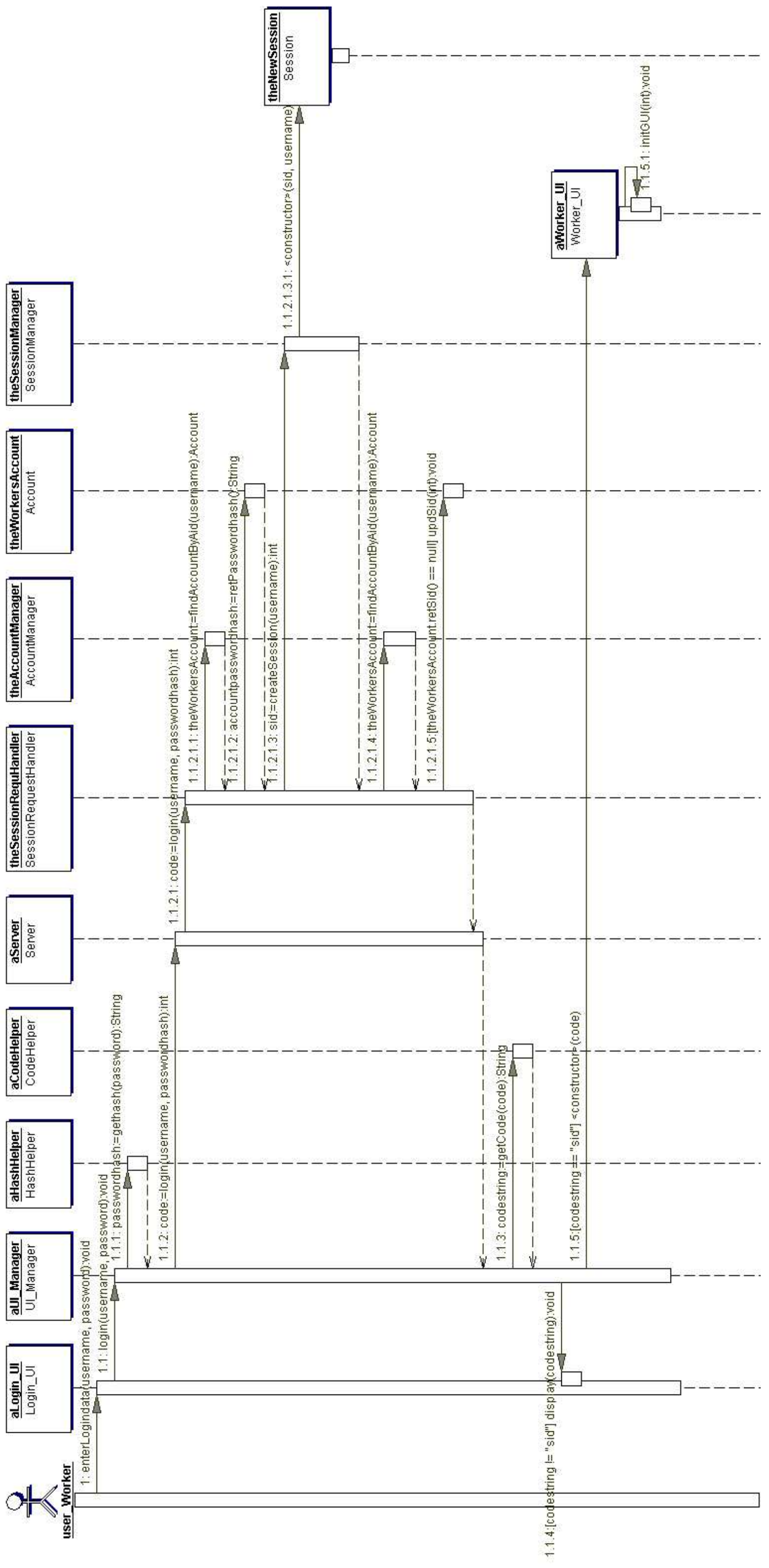
user_Worker
aLogin_UI
Login_UI
aUI_Manager
UI_Manager
aHashHelper
HashHelper
aCodeHelper
CodeHelper
aServer
Server
theSessionRequHandler
SessionRequestHandler
theAccountManager
AccountManager
theWorkersAccount
Account
theSessionManager
SessionManager
theNewSession
Session
aWorker_UI
Worker_UI
1: enterLogindata(username, password):void
1.1: login(username, password):void
1.1.1: passwordhash:=gethash(password):String
1.1.2: code:=login(username, passwordhash):int
1.1.2.1: code:=login(username, passwordhash):int
1.1.2.1.1: theWorkersAccount:=findAccountByAid(username):Account
1.1.2.1.2: accountpasswordhash:=retPasswordhash():String
1.1.2.1.3: sid:=createSession(username):int
1.1.2.1.3.1: «constructor»(sid, username)
1.1.2.1.4: theWorkersAccount:=findAccountByAid(username):Account
1.1.2.1.5:[theWorkersAccount.retSid() == null] updSid(int):void
1.1.3: codestring:=getCode(code):String
1.1.4:[codestring != "sid"] display(codestring):void
1.1.5:[codestring == "sid"] «constructor»(code)
1.1.5.1: initGUI(int):void

Figure 5: Login Successful Sequence Diagram

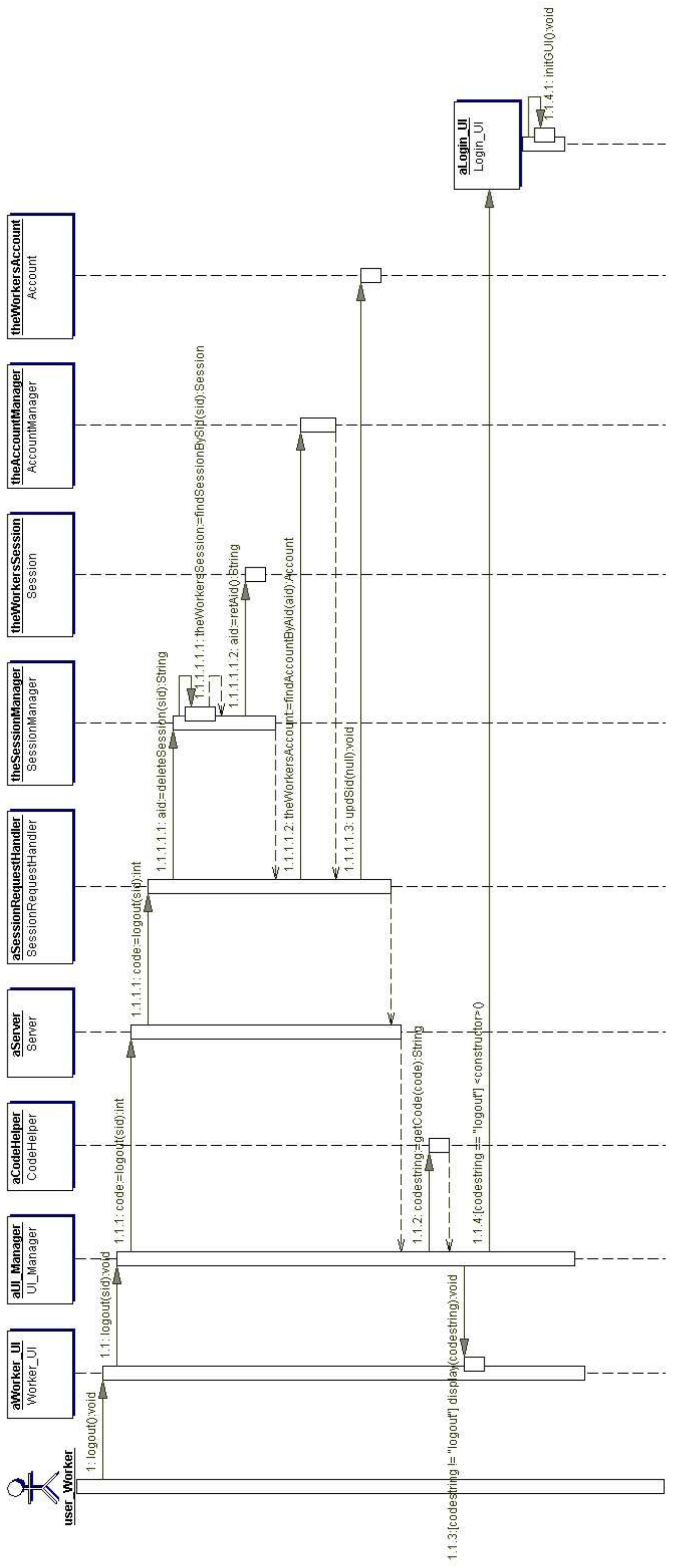
user_Worker
aWorker_UI
Worker_UI
aUI_Manager
UI_Manager
aCodeHelper
CodeHelper
aServer
Server
aSessionRequestHandler
SessionRequestHandler
theSessionManager
SessionManager
theWorkersSession
Session
theAccountManager
AccountManager
theWorkersAccount
Account
aLogin_UI
Login_UI
1: logout():void
1.1: logout(sid):void
1.1.1: code=logout(sid):int
1.1.1.1: code=logout(sid):int
1.1.1.1.1: aid:=deleteSession(sid):String
1.1.1.1.1.1: theWorkersSession:=findSessionBySid(sid):Session
1.1.1.1.1.2: aid:=retAid():String
1.1.1.1.2: theWorkersAccount:=findAccountByAid(aid):Account
1.1.1.1.3: updSid(null):void
1.1.2: codestring:=getCode(code):String
1.1.3:[codestring != "logout"] display(codestring):void
1.1.4:[codestring == "logout"] <constructor>()
1.1.4.1: initGUI():void

Figure 6: Logout Successful Sequence Diagram

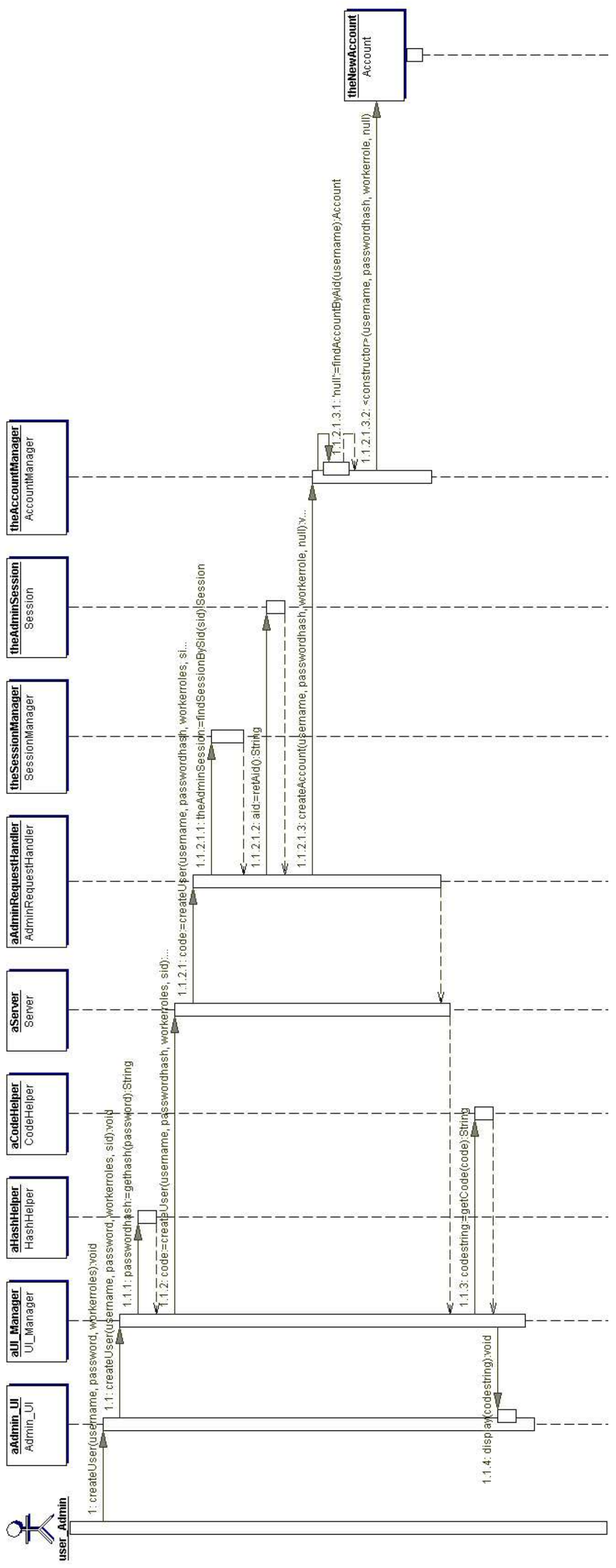
theNewAccount
Account
theAccountManager
AccountManager
theAdminSession
Session
theSessionManager
SessionManager
aAdminRequestHandler
AdminRequestHandler
aServer
Server
aCodeHelper
CodeHelper
aHashHelper
HashHelper
aUI_Manager
UI_Manager
aAdmin_UI
Admin_UI
user_Admin
1: createUser(username, password, workerroles):void
1.1: createUser(username, password, workerroles, sid):void
1.1.1: passwordhash:=gethash(password):String
1.1.2: code:=createUser(username, passwordhash, workerroles, sid):...
1.1.2.1: code:=createUser(username, passwordhash, workerroles, si...
1.1.2.1.1: theAdminSession:=findSessionBySid(sid):Session
1.1.2.1.2: aid:=retAid():String
1.1.2.1.3: createAccount(username, passwordhash, workerrole, null):v...
1.1.2.1.3.1: 'null':=findAccountByAid(username):Account
1.1.2.1.3.2: «constructor»(username, passwordhash, workerrole, null)
1.1.3: codestring:=getCode(code):String
1.1.4: display(codestring):void

Figure 7: Create User Successful Sequence Diagram

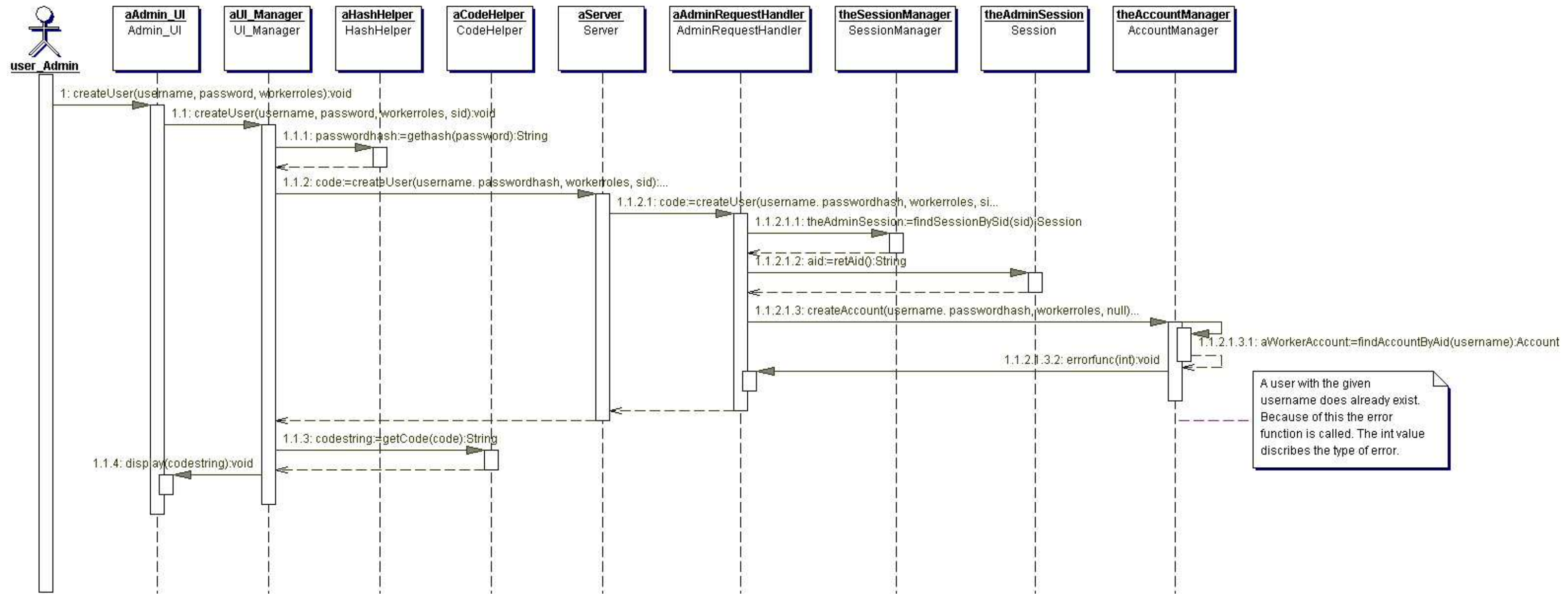
user_Admin
aAdmin_UI
Admin_UI
aUI_Manager
UI_Manager
aHashHelper
HashHelper
aCodeHelper
CodeHelper
aServer
Server
aAdminRequestHandler
AdminRequestHandler
theSessionManager
SessionManager
theAdminSession
Session
theAccountManager
AccountManager
1: createUser(username, password, workerroles):void
1.1: createUser(username, password, workerroles, sid):void
1.1.1: passwordhash:=gethash(password):String
1.1.2: code:=createUser(username. passwordhash, workerroles, sid):...
1.1.2.1: code:=createUser(username. passwordhash, workerroles, si...
1.1.2.1.1: theAdminSession:=findSessionBySid(sid):Session
1.1.2.1.2: aid:=retAid():String
1.1.2.1.3: createAccount(username. passwordhash, workerroles, null)...
1.1.2.1.3.1: aWorkerAccount:=findAccountByAid(username):Account
1.1.2.1.3.2: errorfunc(int):void
A user with the given username does already exist. Because of this the error function is called. The int value discribes the type of error.
1.1.3: codestring:=getCode(code):String
1.1.4: display(codestring):void

Figure 8: Create User Failed Sequence Diagram

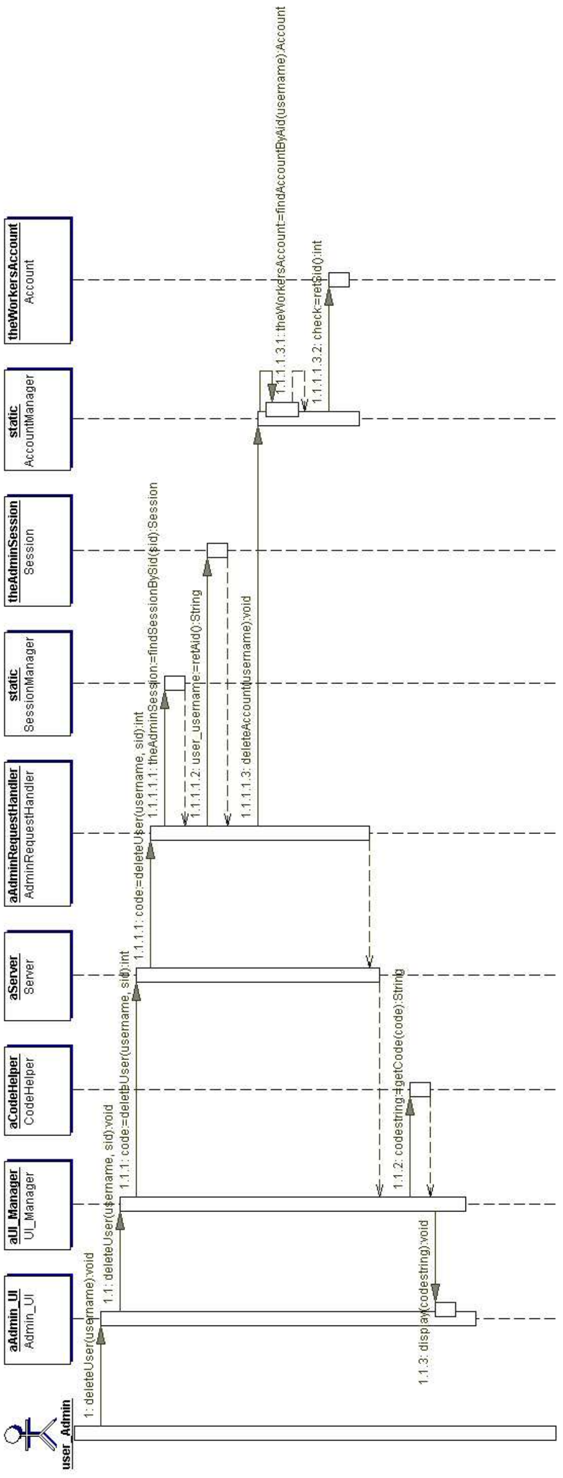
user_Admin
aAdmin_UI
Admin_UI
aUI_Manager
UI_Manager
aCodeHelper
CodeHelper
aServer
Server
aAdminRequestHandler
AdminRequestHandler
static
SessionManager
theAdminSession
Session
static
AccountManager
theWorkersAccount
Account
1: deleteUser(username):void
1.1: deleteUser(username, sid):void
1.1.1: code:=deleteUser(username, sid):int
1.1.1.1: code:=deleteUser(username, sid):int
1.1.1.1.1: theAdminSession:=findSessionBySid(sid):Session
1.1.1.1.2: user_username:=retAid():String
1.1.1.1.3: deleteAccount(username):void
1.1.1.1.3.1: theWorkersAccount:=findAccountByAid(username):Account
1.1.1.1.3.2: check:=retSid():int
1.1.2: codestring:=getCode(code):String
1.1.3: display(codestring):void

Figure 9: Delete User Successful Sequence Diagram

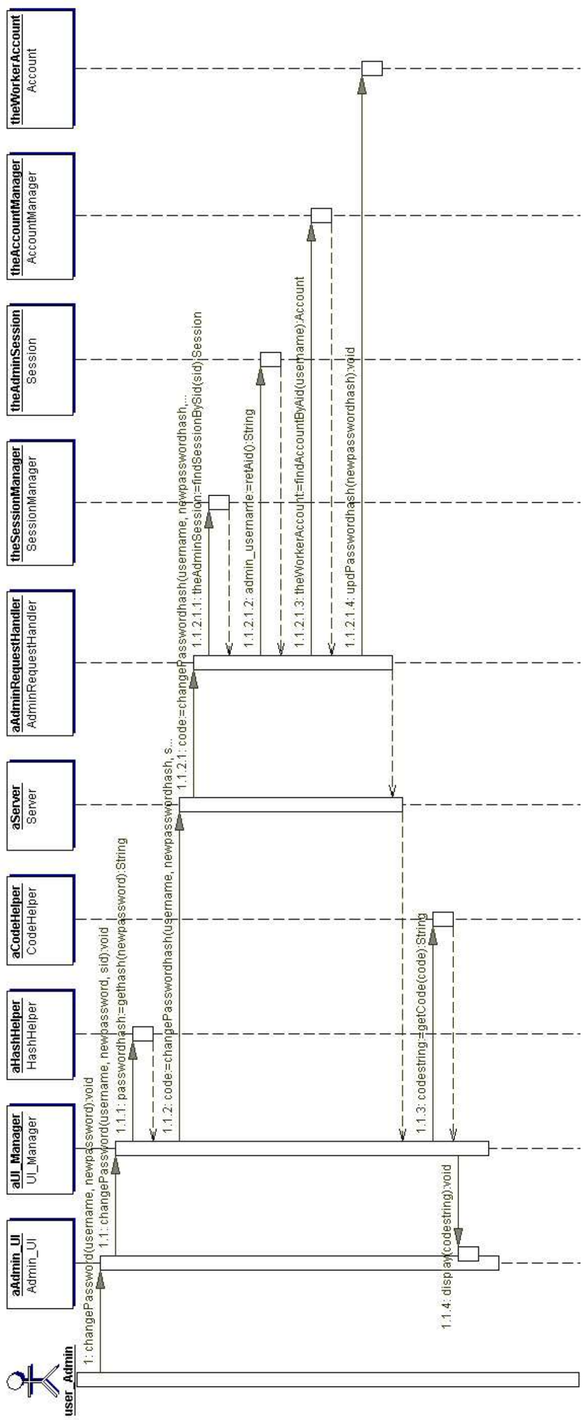
user_Admin
aAdmin_UI
Admin_UI
aUI_Manager
UI_Manager
aHashHelper
HashHelper
aCodeHelper
CodeHelper
aServer
Server
aAdminRequestHandler
AdminRequestHandler
theSessionManager
SessionManager
theAdminSession
Session
theAccountManager
AccountManager
theWorkerAccount
Account
1: changePassword(username, newpassword):void
1.1: changePassword(username, newpassword, sid):void
1.1.1: passwordhash:=gethash(newpassword):String
1.1.2: code:=changePasswordhash(username, newpasswordhash, s...
1.1.2.1: code:=changePasswordhash(username, newpasswordhash,...
1.1.2.1.1: theAdminSession:=findSessionBySid(sid):Session
1.1.2.1.2: admin_username:=retAid():String
1.1.2.1.3: theWorkerAccount:=findAccountByAid(username):Account
1.1.2.1.4: updPasswordhash(newpasswordhash):void
1.1.3: codestring:=getCode(code):String
1.1.4: display(codestring):void

Figure 10: Admin Change Password Successful Sequence Diagram

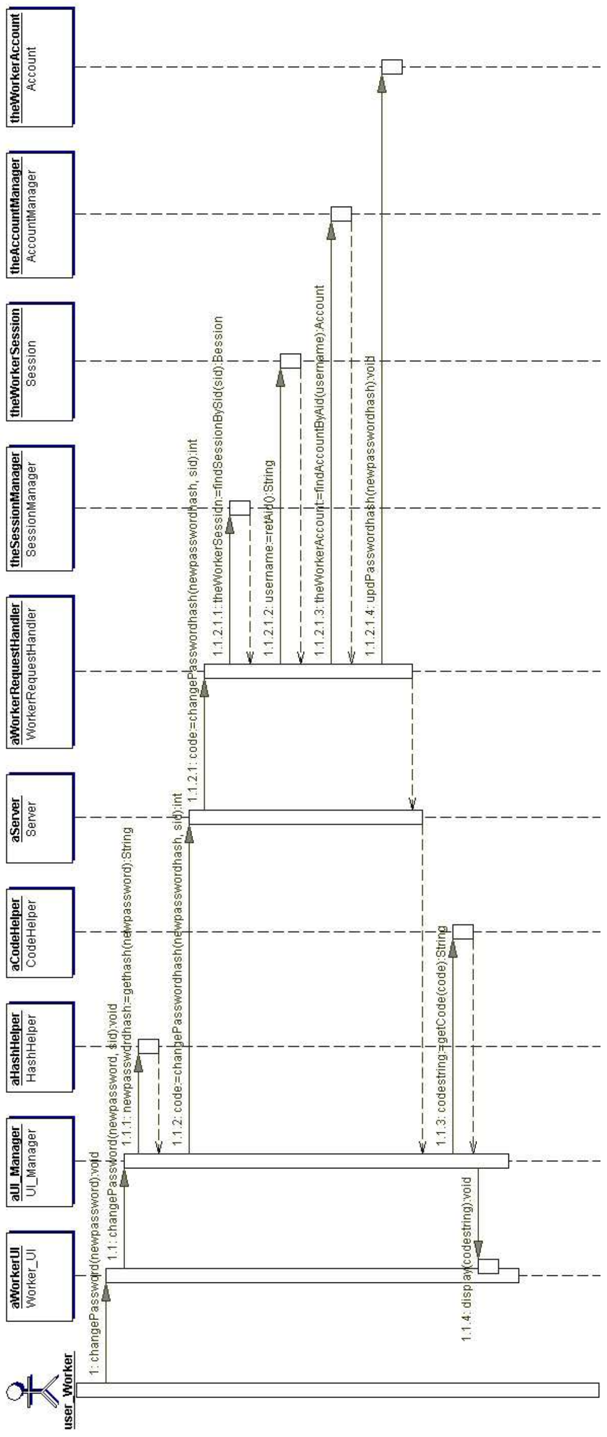
theWorkerAccount
Account
theAccountManager
AccountManager
theWorkerSession
Session
theSessionManager
SessionManager
aWorkerRequestHandler
WorkerRequestHandler
aServer
Server
aCodeHelper
CodeHelper
aHashHelper
HashHelper
aUI_Manager
UI_Manager
aWorkerUI
Worker_UI
user_Worker
1: changePassword(newpassword):void
1.1: changePassword(newpassword, sid):void
1.1.1: newpasswordhash:=gethash(newpassword):String
1.1.2: code:=changePasswordhash(newpasswordhash, sid):int
1.1.2.1: code:=changePasswordhash(newpasswordhash, sid):int
1.1.2.1.1: theWorkerSession:=findSessionBySid(sid):Session
1.1.2.1.2: username:=retAid():String
1.1.2.1.3: theWorkerAccount:=findAccountByAid(username):Account
1.1.2.1.4: updPasswordhash(newpasswordhash):void
1.1.3: codestring:=getCode(code):String
1.1.4: display(codestring):void

Figure 11: Worker Change Password Successful Sequence Diagram

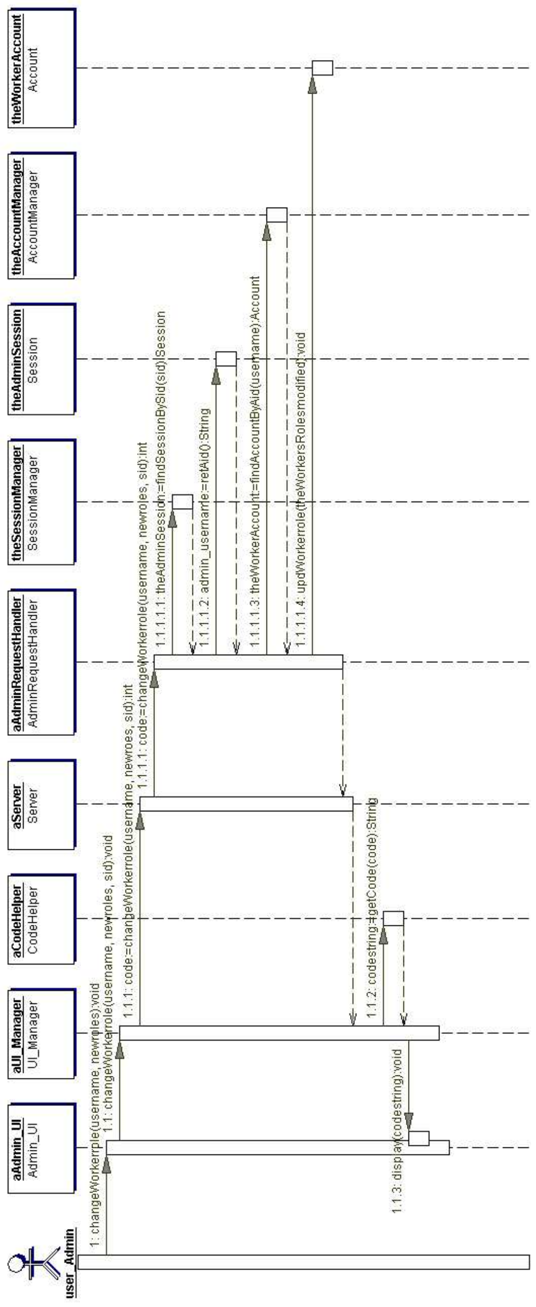
theWorkerAccount
Account
theAccountManager
AccountManager
theAdminSession
Session
theSessionManager
SessionManager
aAdminRequestHandler
AdminRequestHandler
aServer
Server
aCodeHelper
CodeHelper
aUI_Manager
UI_Manager
aAdmin_UI
Admin_UI
user_Admin
1: changeWorkerrole(username, newroles):void
1.1: changeWorkerrole(username, newroles, sid):void
1.1.1: code:=changeWorkerrole(username, newroes, sid):int
1.1.1.1: code:=changeWorkerrole(username, newroles, sid):int
1.1.1.1.1: theAdminSession:=findSessionBySid(sid):Session
1.1.1.1.2: admin_username:=retAid():String
1.1.1.1.3: theWorkerAccount:=findAccountByAid(username):Account
1.1.1.1.4: updWorkerrole(theWorkersRolesmodified):void
1.1.2: codestring:=getCode(code):String
1.1.3: display(codestring):void

Figure 12: Change Workerrole Successful Sequence Diagram