ID2207- Modern Methods in Software Engineering

Homework 4

Course Coordinator:

Mihhail Matskin

Course Assistant:

Shatha Jaradat

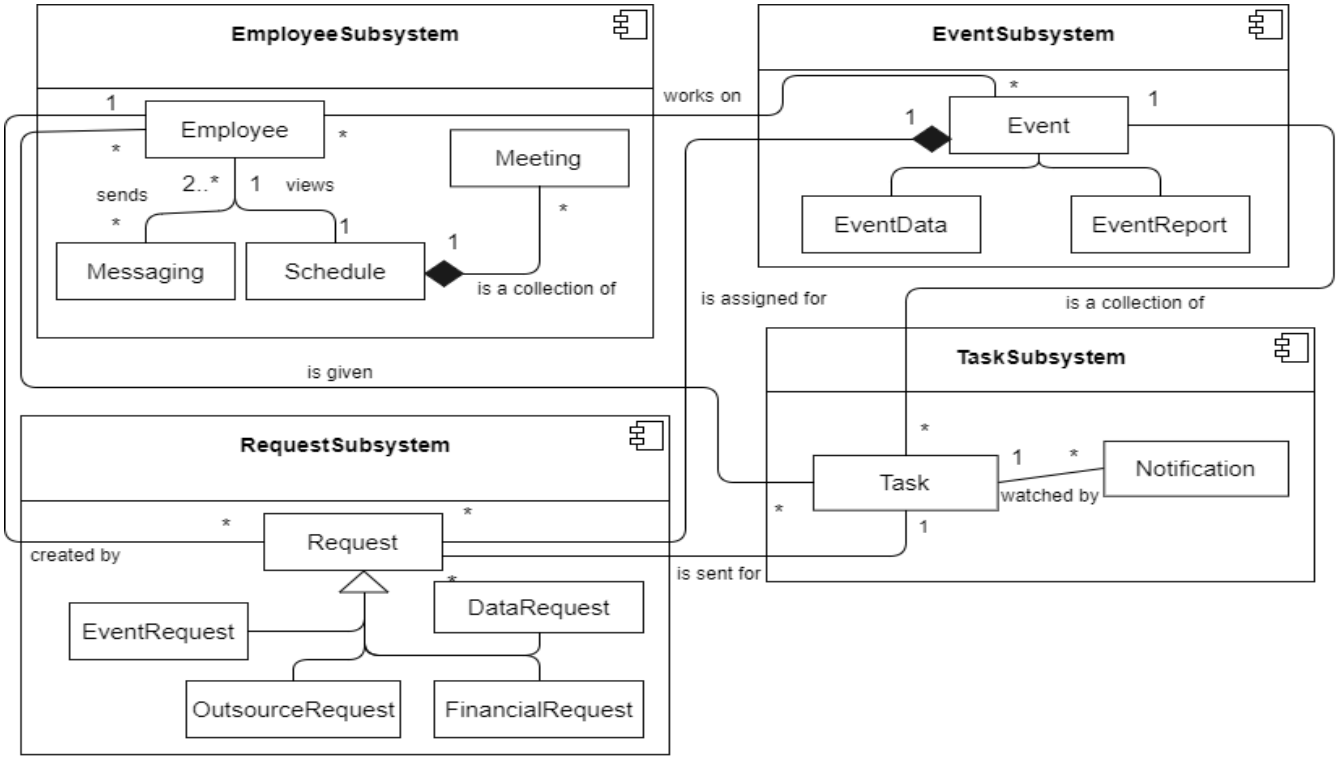
Óttar Guðmundsson and Örn Arnar Karlsson

Group 17

2017.10.1

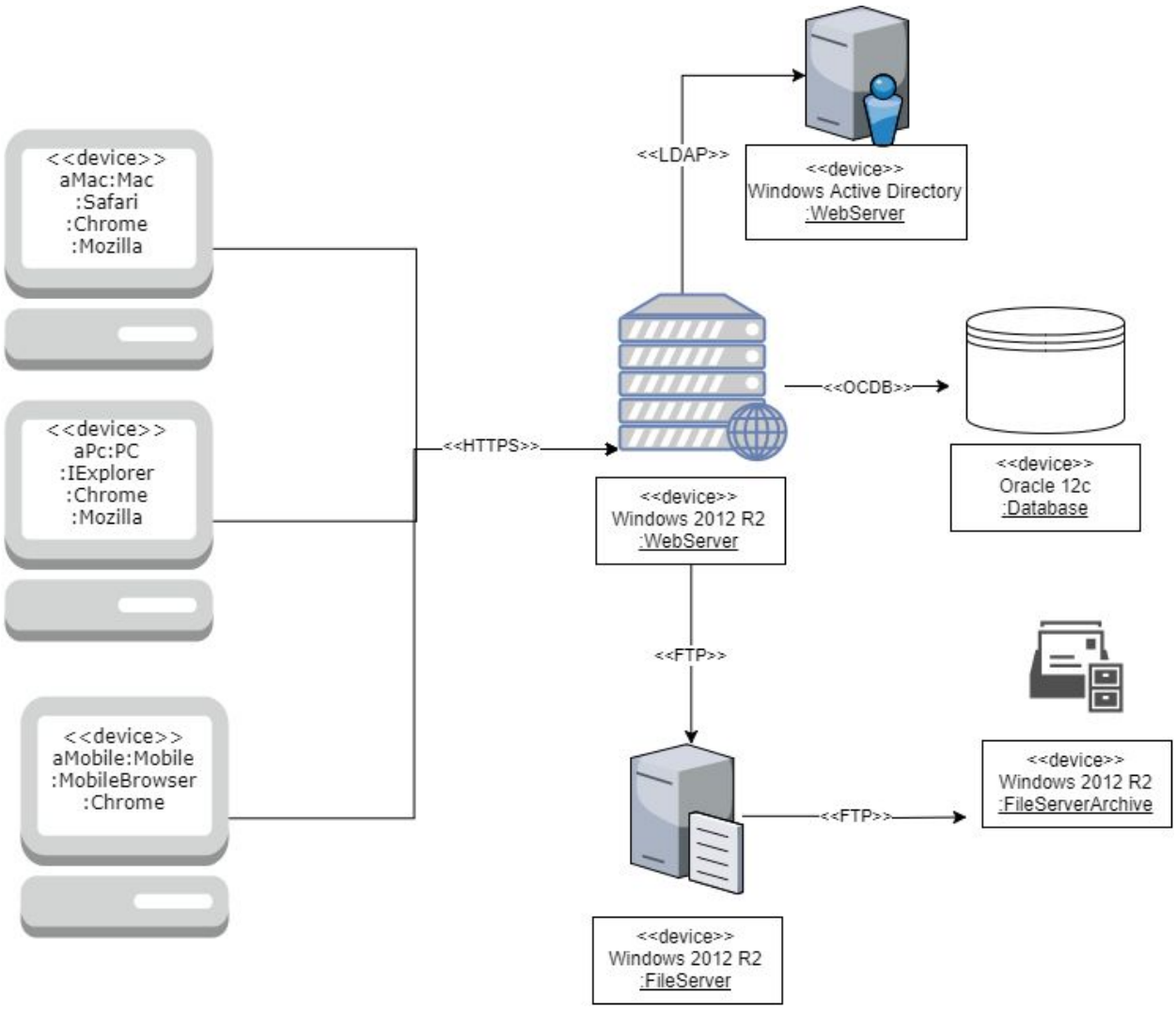
1. Subsystem decomposing structure (using class diagrams)

A class diagram showing the decomposition.



2. Mapping subsystems to processors and components (hardware/software mapping) using UML deployment diagrams.

1. UML deployment diagram



2. Brief description about the selected technologies and software components.

The reason why we suggest this implementation is that both of us have several years of experience in chosen platforms. By our judgment, they also provide stable and secure methods to quickly regain control of crashed servers or corrupted data. To start off with the client, the frontend should be implemented as a browser based solution with a HTTPS connection to the server. A browser runs on all modern devices including mobiles and tablets. The downside of this method is that the CSS of the interface needs to be coded with scalability solutions but that is easier and more cost-efficient than writing apps for different operating systems.

The server itself should run on Windows server 2012 R2 with IIS7.5 or higher since the system does both include role delegation and access control, modules and other components. The service can then be extended to use Windows Active directory if the corporate environment and network runs on Windows to implement Impersonation passthrough for authentication. The connection used will be of LDAP to manage users later on if the corporation grows global in upcoming years. For mobile devices, two step authentication is provided.

The database chosen will be Oracle 12c, both for scalability of data and usage of transactions to prevent database corruption on server crash. It will hold data about the corporation's financial information, former and upcoming events, client details and more information that can be stored and fetched when creating objects for client.

Finally, the file server should also be implemented on Windows since all of our other systems runs on Windows. This simply makes things easier. The server will hold data about event reports in pdf formats, images related to events and excel / word files generated by users. The file server runs on a external server so users can get their files across different devices. We prefer keeping the file server for files that have proper extensions that can be viewed so we do not have to keep it as a BLOB in Oracle database which needs encoding / decoding every time it has to be opened. With these technical setups we are confident in saying that the system we are going to develop is a three tier layer architecture.

3. Persistent storage solution for the problem

1. List of persistent objects

Object	**Storage management**
EventRequest	Relational Database
FinancialRequest	Relational Database
OutsourceReqeust	Relational Database
Notification	Relational Database
Task	Relational Database
User	Relational Database
UserInfo	Relational Database
UserRoles	Relational Database
Event	Relational Database
EventData	Flat file and Relational Database
EventReport	Flat file
EventImages	Flat file

2. Brief description of the selected storage management strategy

The data strategy used will be split into two separate servers, both a file server and an Oracle database. The database itself will store tables that represent the structured objects mentioned above. Whenever an user needs to generate a object with information related to it, a simple procedure can be called to transfer data which the server can send back to the user. For example, if user wants to see all events that have been planned but are yet to be held a query such as

*select * from DB_Events where DateFrom >= Sysdate*

Which will return datatable with all the information. If user wants to view a fresh EventReport a call will be made to the database

select DServer, DPath, DFieId, DExtension from DB_GeneratedData where DEventId = :pID and EType='EVENTREPORT'

Which might return us something like 'https://SED.freshfileserver.se', '2017/10/events/report/', 'EVENTREPORT_FINAL' and '.pdf'. This can be concatted together to create a simple url that the user can view through his browser or the fileserver can feed through so the user can download it.

The reason we want to do it this way is because it allows the server to be able to hold multiple files of different extensions that can easily be fetched by a query. Another great reason for doing it this way is that whenever we want to migrate old data to the archived server, we can simply move a folder between servers and simply update the *EventServer* column in the table. That way we keep the new fileserver fresh with only 5 year old data. Other benefits are also that we can restrict certain users by joining roles to urls or extensions and many more.

4. Access control, global control flow, and boundary conditions.

1. Access Control:

1. Access Matrix

Actor/Object	Event	EventData	EventReport	EventRequest	FinancialReque	OutsourceRequ	Task	User	Schedule
ProductionManager	View		View		CreateRequest	CreateRequest	Create Edit	View	view
SeniorCustomerService	CreateEvent		View	Create AcceptNewRequest RejectEvent			View	View	view
CustomerService	AddReport	Create	Create EditInfo SendForApproval	Create SendRequest			View	View	view
FinancialManager	View	AllocateData	View	WriteFeedback	AcceptRequest RejectRequest		View	View	view
AdministrationManager	View		ApproveAndSave	AcceptRequest RejectRequest			Create Edit	View	view
HumanResources	View		View			Accept Reject	View	CreateUser	view
SubTeams	View		View				Edit	View	view
VicePresident	View	ModifyData	View				View	View	view
ServiceDepartmentMana	View		View		CreateRequest	CreateRequest	Create Edit	View	view

2. Brief description about security, authentication/authorization strategy, confidentiality of data, and network/infrastructure security.

We prefer to keep all text messages send in a chatbox between users to be encrypted to ISO27001 standard since users might be talking about sensitive information regarding financial information and/or client details regarding the event. The messages sent from server to client should also be sent through a HTTPS connection to prevent common attacks such as man in the middle.

Authentication can be achieved through three different ways. Support for a two step authentication will be implemented if users are trying to log on through an untrusted or unrecognised device to prevent a malicious user to enter the system. Electronic ID's will also be supported for devices that are not connected to the company's intranet if user does not want to have the two step authentication. The last option is that we will allow authentication if user has successfully logged on his Windows account in a Windows operating system on the intranet. Impersonation passthrough will be enabled to utilize a single sign on factor for the system.

The network/infrastructure should be done through Windows Firewall since all of our system recommendations run on Windows. These settings will be implemented by the IT / Tech support of the company but can be outsourced by a third party company if advanced settings in the firewall are required.

2. Global control flow:

1. Brief description of the selected control flow for the system

For our system, an event driven architecture (EDA) would be the perfect fit since the system is constantly waiting for a series of events, generated by individual users and then waiting for reactions. The objects used in the system often undergo a difference in changes of their states, such as the EventRequest going from “Waiting for FM Feedback” and “Waiting for AM Approval”. By changing states of objects, other interfaces of the application have no problem knowing the state of objects when they have been changed by someone else. The systems interface is made out of individual boundary objects that all have similar og related components and functionality. They can all be bound to an event or an action listener to propagate actions by the system in the correct order they are performed. A similar platforms that use these techniques are Asp.NET and Java Swing where classes are most often away of a event which is called by a listener. The classes can override inherited methods and interfaces so we could create distinctive *View, Get, Accept* etc for all objects. Then we could simply add these classes to the listener. For a short C# example

```
void SendRequestForm(object sender, CustomEventArgs a)
{
        /*
                Code here...
        */
}
EventHomePage.SendFormButton += new CustomEventHandler(SendRequestForm);
```

3. (Boundary conditions):

1. Boundary use cases

Use case name	CreateUserTimeout
Entry condition	HumanResources has logged into the EventSystem
Flow of events	1. On successful login, the HumanResources chooses to create a new user. 2. A new form appears where HumanResources can fill in the required info regarding a new user, such as name, email, phone and responsibilities. 3. He clicks the “Confirm new user” and wait for a confirmation from the system. 4. After the systems maximum request timeout, HumanResources is notified that the user was not created because of a *TimeoutException*
Exit condition	HumanResources accepts the prompt message notifies the IT department about the faulty creation.

Use case name	CheckDataValidity
Entry condition	VicePresident has successfully logged on into the EventSystem
Flow of events	1. When VicePresident has the objective of modifying data in the EventReport, he opens up the raw EventData 2. After a successful HTTPS call from the server, the client performs a checksum on the data that it compares to the checksum in the header. 3. If the checksum does not match, data is marked as corrupted as is not displayed. 4. VicePresident is notified with a warning and has an option to report this to the IT department.
Exit condition	VicePresident notifies IT department about possible data corruption or a bad connection.

Use case name	RollbackQueryOnCrash
Entry condition	SeniorCustomerService has successfully logged into the system AND FirstBusinessMeeting is completed.

Flow of events	1. After a successful business meeting the SeniorCustomerService takes the information provided by the client and chooses to create a new Event in EventSystem. 2. When SeniorCustomerService presses the CreateEventButton an Oracle transaction is executed with several different queries to modify the needed tables in Database. 3. An unexpected error hits the server because of power outage and server cannot finish the transaction. 4. By default, the Oracle database rollback the inserted statements since they are bound to an Oracle transaction to prevent database corruption.
Exit condition	The client is responded with a NoServerFoundException.

Use case name	ServerOutOfMemory
Entry condition	EventSystem is up and running
Flow of events	1. Any user within the EventSystem is either answering a task or replying to a request. 2. In an unforeseen edge case, when many concurrent users have been using the EventSystem a connection might not be properly disconnected or handled which might be related to a memory leak in software. 3. Server gets unresponsive and Windows 2012R2 handles the IIS7.5 OutOfMemory exception. 4. Windows displays the configuration error to client.
Exit condition	User can see the reason why server is not responsive, based on the web config variable system.web - customErrors is set to *On*

5. Applying design patterns to designing object model for the problem

Brief description about the selected design patterns and why/how you are going to use them

We'll be using a *Facade design pattern*. It provides a simple interface which makes the rules and boundaries clear. That makes it easier for multiple programmers to work together and minimizes the risks of bugs.

It is convenient because the client's interface is quite simple, there are not many options. All data flows in a pre-designed path and because of that it makes sense to provide the interface designers with a simple facade to talk to. To summarise: The options the client has are limited even though the system behind it might be quite complicated so a simple facade for it to talk to is suitable.

6. Writing contracts for noteworthy classes

Contracts described in OCL

```
context Event inv:
        self.hasBeenApproved == true && self.formerEventRequest != null

context Event::setEventDates(dStart:DateTime, dEnd:Datetime) pre:
        dEnd > dStart

context Event::setExpectedGuests(g:int) pre:
        g > 0

context Event::addTaskToEvent(t:Task) pre:
        self.dStart > DateTime.Today && !self.cancelled

context Event::viewEventReport(e:EventReport) pre:
        e.Approved == true
```

```
context FirstBusinessMeeting inv:
        EventAcceptedByAm() == true

context FirstBusinessMeeting::inviteUsers(e:Employee, d:Date) pre:
        isUserAvailable(e,d)

context FirstBusinessMeeting::saveClientDetails(m:MeetingForm) pre:
        m.isValid()
```

```
context EventReport inv:
        self.Event.getDateTo() < DateTime.Today

context EventReport::modifyData(d:DataTable) pre:
        d != null

context EventReport::requestData(f:FinancialManager) pre:
        self.ValidReportDescription()

context EventReport::saveToFileserver(a:AdministrationManager) post:
        a.Approved(self) == true
```

```
context EventRequest inv:
        self.seniorCustomerServApprove == true

context EventRequest ::setFinancialFeedback(f:FinancialFeedbackForm) pre:
        f.estimatedBudget > 0 && f.isValid()

context EventRequest ::reviewFeedback(f:FinancialFeedbackForm) pre:
        self.getFinancialFeedBackForm() != null

context EventRequest ::bookFirstBusinessMeeting(f:FirstBusinessMeeting) post:
        self.admittedByAm = true && f.isValidDate() && f.isWorkDay()
```

```
context Task inv:
        EventAccepted == true

context Task::sendToSubTeam() pre:
        self.getSubteam.PositionsFilled == true

context Task::sendToSubTeams(s:Subteam) pre:
       self.EventHasBeenAccepted == true

context Task::sendToSubTeams(t:Task,s:Subteam) post:
       t.count < s.Maximum

context Task::sendToSubTeams(c:ClientDetails) pre:
       self.clientDetailsValid() == true

context Task::assignEmployeeToSubteam(e:Employee) pre:
        e.IsFreeOn(self.EventDate) == true
```