
KTH

MODERN METHODS IN SOFTWARE ENGINEERING

(ID2207)

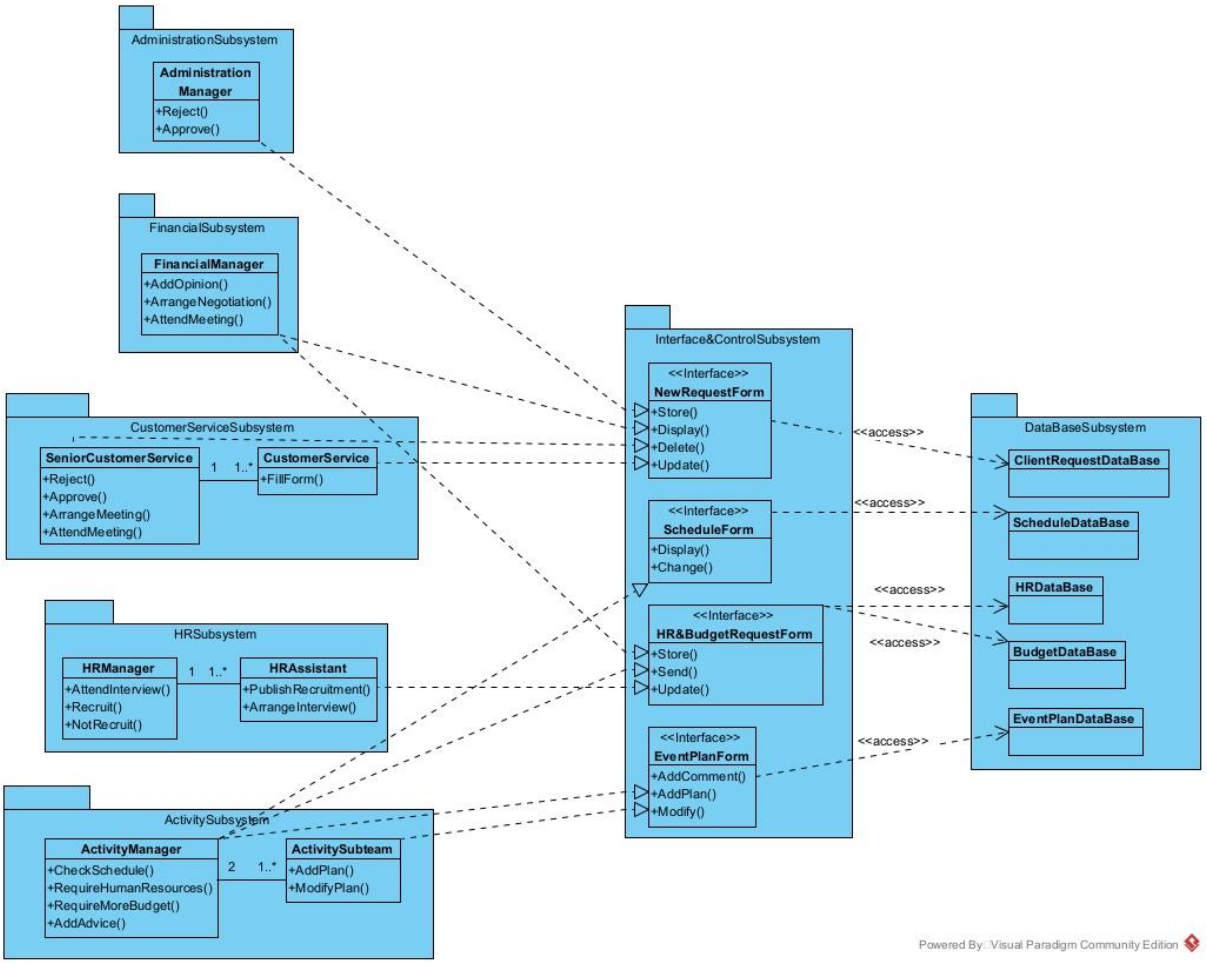
HOMEWORK 4

GROUP 6

Name: **Yuchen Gao**
Name: **Weikai Zhou**

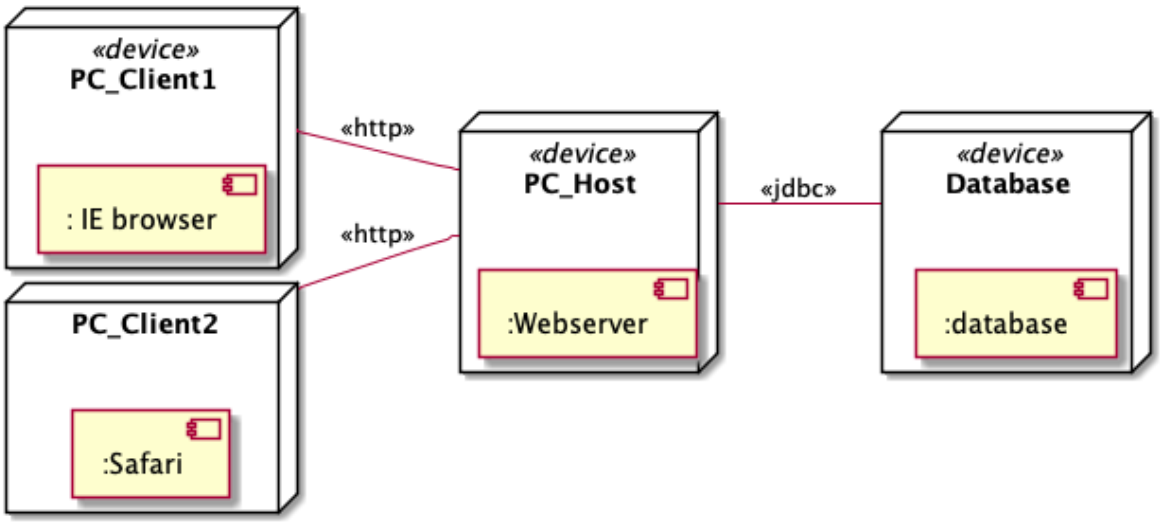
Date: **3 October, 2021**

Subsystem decomposition structure



Deployment Diagram

PC Clients	Client that uses a personal computer to log in and operate.
PC_Host	The virtual host machine that supports forward and backward proxy.
Database	The physical location of data.
http	An application layer protocol in the Internet protocol suite model for systems.
jdbc	A standard application programming interface (API) that allows Java programs to access database management systems



Persistent Objects:
Data of Client Requests
Data of Staff Schedule
Data of Human Resources Log
Data of Budget Log
Data of Event Plan

Storage Management Strategy:
Object-oriented database:We have extensive use of association to retrieve data, and irregular associations among objects. The data should be stored even if there is a crash in the system. And multiple users will access the data and change the data. And generally, the data is of medium size since the SEP is not a large scale company. Therefore, the data should be stored in an object-oriented database.

Access Matrix:

\ Object Actors	ClientRequest	HRLog	EventPlan	BudgetLog	StaffSchedule
Customer Service	RecordRequest();				
Senior Customer Service	ReviewRequest(); ApproveRequest(); RejectRequest();				Review(); AddSchedule();
Financial Manager	ReviewRequest(); AddOpinion();			Review(); ChangeBudget();	AddSchedule();
Senior HR Manager		Review(); ApproveRecruitment(); RejectRecruitment();			AddSchedule();
Administration Manager	ReviewRequest(); ApproveRequest(); RejectRequest():			Review();	AddSchedule();
HR Team		Review(); ArrangeRecruitment();			
Activity Manager	ReviewRequest();	RequestHR();	Initiate(); AddAdvice();	RequestBudget();	Review(); AddSchedule();
Sub-team			MakePlan(); ModifyPlan(); CommentOnResources();		AddSchedule();

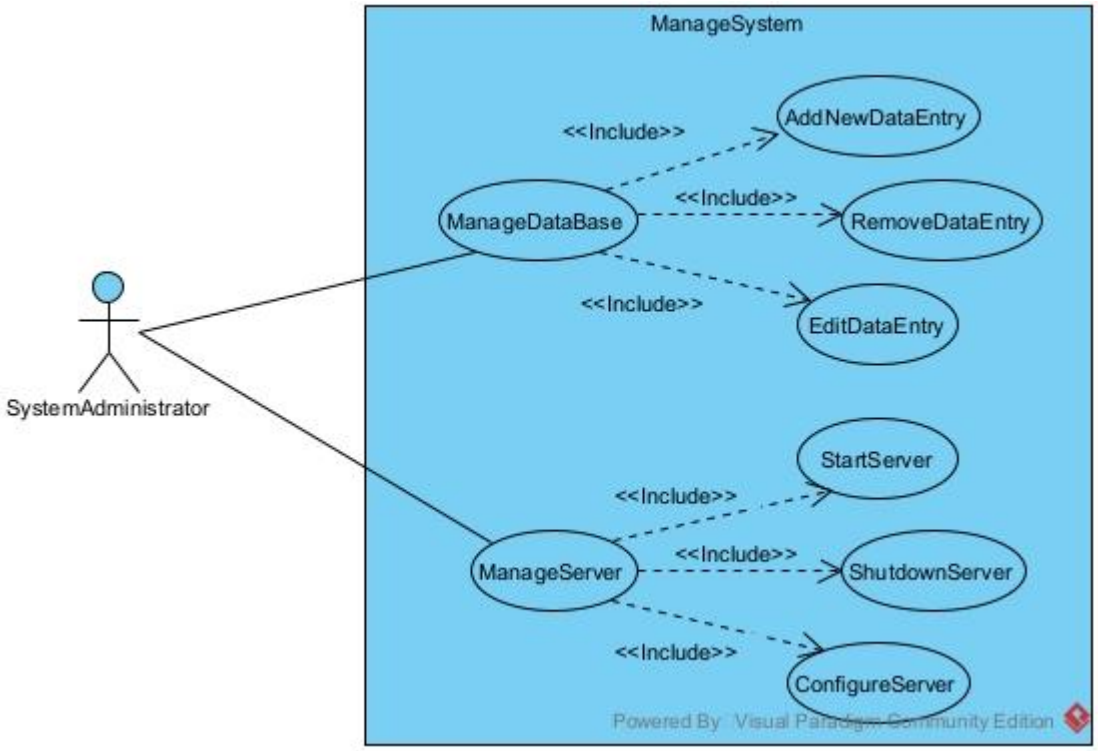
Staff has to log in and identify their position before accessing functions, after the verification, they can operate the functions only restricted to them in the Matrix. Taking the Activity Manager for example, they can only check the plan for activity and view the status of human resources, instead of reading the budget log.

Global Control Flow:

Event-driven control

Because procedure-driven control should be avoided in the final system, and event-driven control is more mature than the threads. We want our system to be stable and we do not have the multi-task requirement, we want a centralized procedure and wait for other branch tasks to terminate so the event driven control is the best choice.

Boundary Conditions



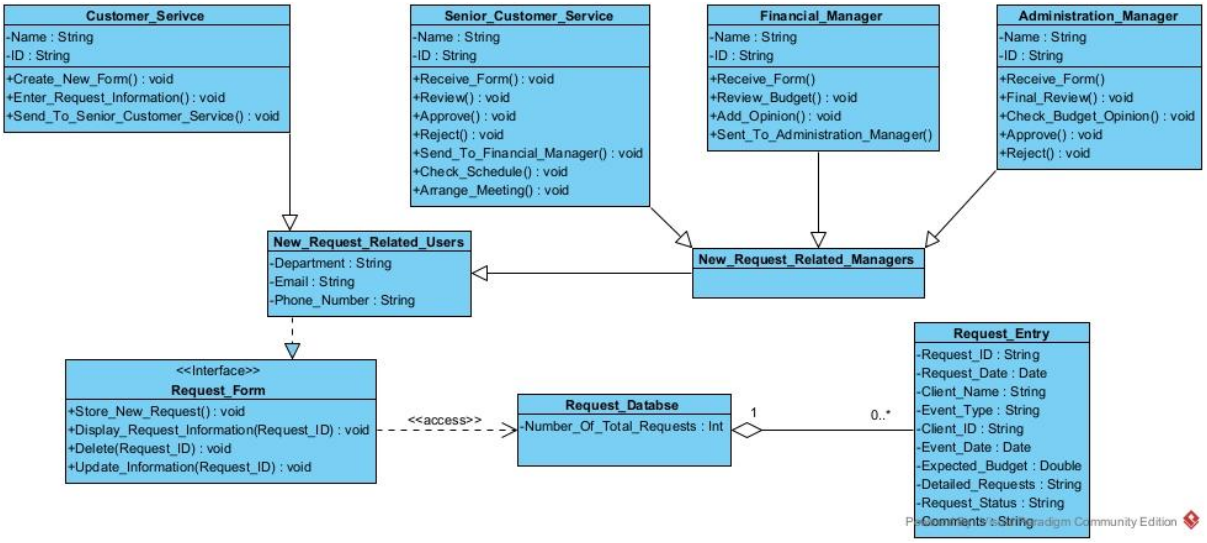
ManageDataBase	A use case that includes AddNewDataEntry, RemoveDataEntry and EditDataEntry.
AddNewDataEntry	The system administrator adds new data entries manually to the database.
RemoveDataEntry	The system administrator removes data entries manually from the database.
EditDataEntry	The system administrator edits the data entries manually from the database.
ManageServer	A use case that includes StartServer, ShutdownServer and ConfigureServer.
StartServer	The system administrator starts the server.
ShutdownServer	The system administrator stops the server.
ConfigureServer	The system administrator performs routine configuration of the server.

Design Pattern:

Proxy Design Pattern:

The proxy design pattern uses another object ("the proxy") that acts as a stand-in for the real object. In our system, the proxy is the virtual request forms. We use the virtual request form to represent the physical form, and normally we do not need the real request form during the communication of the company. The Proxy Object implements the same operations as the Real Objects. We can use the Java interface to implement the proxy pattern easily. And it provides the access control to the real object. For example, HRLog is only shared by Senior HR Manager, HR Team and Activity Manager.

OCL



context Customer_Service::Send_To_Senior_Customer_Service()

post: Senior_Customer_Service.Receive_Form()

context Senior_Customer_Service::Send_To_Financial_Manager()

post: Financial_Manager.Receive_Form()

context Financial_Manager::Send_To_Administration_Manager()

post: Administration_Manager.Receive_Form()

context Senior_Customer_Service::Arrange_Meeting()

pre: Administration_Manager.Approve()

context Request_Form::Store_New_Request()

post: Request_Database.Number_Of_Total_Requests =

Request_Database.Number_Of_Total_Requests + 1

context Request_Form::Display_Request(String: Request_ID)

pre: Request_Database.Number_Of_Total_Requests > 0 and
Request_Entry.Request_ID.isExist() == True

context Request_Form::Delete(String: Request_ID)

pre: Request_Database.Number_Of_Total_Requests > 0 and
Request_Entry.Request_ID.isExist() == True

context Request_Form::Delete(String: Request_ID)

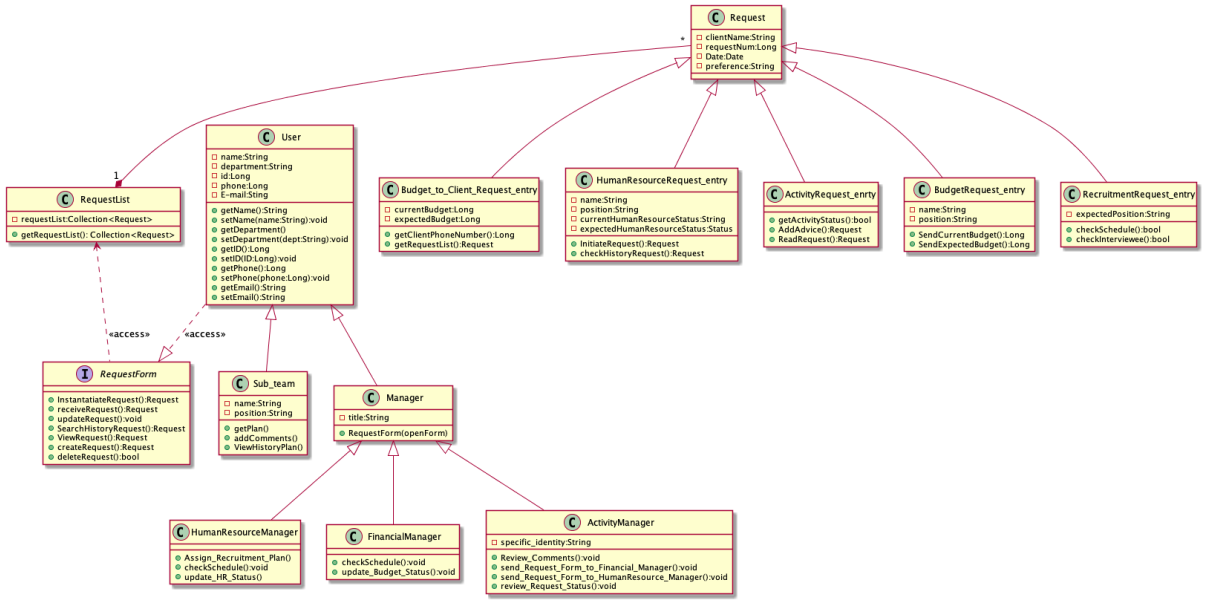
post: Request_Database.Number_Of_Total_Requests =
Request_Database.Number_Of_Total_Requests - 1

context Request_Database

inv: Number_Of_Total_Request >= 0

context Request_Entry

inv: Request_Data.before(Event_Date)



context Manager::RequestForm():

post: RequestList.getRequsestList->size = RequestList.getRequsestList->size + 1;

context Sub_team::getPlan()

pre: ActivityManager.RequestForm();

context ActivityManager::ReviewComments()
pre: Sub_team.addComments();

context Sub_team::ViewHistoryPlan()
pre: (x|x.SearchHistoryRequest->size) >= 1;

context FinancialManager::RequestForm()
pre: ActivityManager.send_Requesr_Form_to_Financial_Manager();

context FinancialManager::RequestForm()
post: RequestList.getRequsestList->size = RequestList.getRequsestList->size + 1;

context FinancialManager::checkSchedule()
pre: FinancialManager.RequestForm();

context BudgetRequest_entry
inv: self.sendCurrentBudget() >= 0;

context BudgetRequest_entry
inv: self.sendExpectBudget() > self.sendCurrentBudget();

context ActivityManager::review_Request_Status()
pre: FinancialManager.update_Budget_Status();

context HumanResourceManager::AssignRecruitmentPlan()
pre: ActivityManager.send_Request_Form_to_HumanResource_Manager();

context ActivityManager::send_Request_Form_to_HumanResource_Manager()
post RequestList.getRequsestList->size = RequestList.getRequsestList->size + 1;

context ActivityManager::send_Request_Form_to_HumanResource_Manager()
post HumanResourceRequest_entry->currentHumanResourceStatus **is**
UnderRecruitment;

context RecruitmentRequest_entry::check_interviewee()
pre HumanResourceManager.AssignRecruitmentPlan();