CENG 491

D&D SOFTWARE

INITIAL DESIGN REPORT

Prepared by:
Firat Alpergin
Dogan Yazar
Tuncay Namli
Mehmet R. Dogar

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROBLEM DEFINITION

The resulting product of this development process is a web-based project management tool (DProject). Dproject lets its users define new companies in the system, manage the projects of a company, perform task management operations, perform user operations, perform resource management, perform notification operations within the system, create and export/import files & statistics, perform meeting arrangement operations, use project planning facilities, view forums and compose forum threads. The clients of the project also have the opportunity to view the overall progress of the project they are purchasing. The tool's ultimate aim is to ease the development of a project by all means.

The main functionalities, goals and objectives of DProject can be found in the section that follows.

1.2 GOALS & OBJECTIVES

DProject sets its limits to the level where the aim of easing the management of projects can be fully satisfied. The main goals and objectives of DProject are as follows:

- To provide easy and secure access to its users. The ease of access is accomplished by the web-based nature of DProject. To be able to provide enough security to its users, DProject will have additional security issues that will provide the secure environment to any of its users.

- To provide consistency into the system among the members. In the real world projects, there is hierarchical decomposition among the project team (and generally in the company). This should also appear in a project management tool and DProject accomplishes that by defining different levels of access rights that can simulate the real world hierarchy (e.g. administrator rights, project manager rights, ordinary user rights, etc.).

- To provide efficient task management operations. Task management is one of the most important features of a project management tool and DProject offers advanced task management features to its users. Users, depending on their access rights, can create tasks, assign users to tasks, assign reviewers to tasks, can view task history trails, monitor task progress, perform critical path management, work on tasks, etc.

- To provide features for efficiently managing meetings. Meeting management is one of the most problematic issues of a typical project development process, especially in major ones. DProject uses a special system, in which the arranger of the meeting provides options for the meeting and notifies them. Then according to the feedback from the potential attendants, DProject lets the meeting arranger choose the optimum meeting details, also taking the preferences of the arranger into account.

- To provide communication means among the users. Communication is very important in large scale projects and DProject provides notifications within the system to satisfy

the communication needs of its users. Another important communication feature is forums, which can be used for any purpose among the members of a company.

- To provide human management features. Human factor is an important variable projects so they are treated separately in DProject. The users working in a project, the amount of work done by each member, the payment information of members, and many other features can be monitored and controlled in DProject.

- To provide resource management features. Resources of a management are very important entities and efficient ways should be developed for handling the management of them. In DProject, different resources can be attached to different projects (or companies, more generally), their necessary information (e.g. unit price, seller address, etc.) are kept, resources can be attached to tasks, budget information of a project is kept and updated accordingly, etc.

- To provide features for report & statistics generation and their exportation/importation. Reports and statistics are vital for any project development because they are useful both within the project and also among different projects because they are used for various purposes including efficient project planning, user capability analyses, etc. DProject has a number of important features for efficient report & statistics operations. These include the importation/exportation of reports from/to different formats, the importation/exportation of a project as a whole from one system to another, statistic generations for specific subset of tasks, for the overall project tasks, for user teams, for individual members, for a duration of time, for the whole project life span, etc.

- To provide efficient means of project scheduling. Scheduling is one of the most problematic issues of a project development process that can occur in serious conflicts between the developer site and the client side. To be able to prevent such inconsistencies, DProject offers sophisticated features for project scheduling. The users can see task creation times, the estimated hours spent on tasks or the whole project, Gantt charts created automatically, etc.

- To provide features for the clients to follow the progress of the project. The clients naturally want to view the project they are purchasing, so DProject lets its clients see the necessary information for them to understand that whether the project is progressing as they wish or not.

1.3 STATEMENT OF SCOPE

The following general requirements apply to DProject:

- A way to define new company and set up new company information
- A way to add new users to the system
- A way to define new projects and set up new project information
- A way to define tasks, assign users to tasks, assign reviewers to tasks, work on tasks, attach resources to tasks, review tasks, confirm/reject tasks, view tasks
- A way to handle critical path management
- A way to arrange meetings
- A way to handle communication among users

- A way to handle human management
- A way to handle resource management
- A way to create, import/export statistics & reports
- A way to perform project planning
- A way to perform project progress monitoring for clients

1.4 DESIGN CONSTRAINTS

To be able to work efficiently, satisfying the requirements imposed, DProject should be carefully designed. However, there are some design constraints which should be taken into account while designing the system.

DProject is a web-based system and that adds an overhead because of the possible problems with the Internet connection. To be able to minimize the effect of this overhead, the communication within the system modules should be minimized avoiding the unnecessary interactions that can further delay processing.

DProject is a system that heavily interacts with the database behind it. Nearly all the necessary information for processing is maintained in the database. There is a heavy load of fetching/storing data from/to the database. This makes the efficiency of the DBMS an important constraint that must be taken into account seriously. An efficient DBMS should be used and the database should be carefully designed, preventing any unnecessary burden put on the DBMS. Also the queries should be designed efficiently to minimize the cost of database operations. By this way, the overhead caused by the DBMS can be minimized.

1.5 WORK BREAKDOWN STRUCTURE

The work breakdown structure for DProject is as follows. Note that the work-package definitions for 'implementation' and 'test & debugging' sub-projects are preliminary, and will be revised in each successive document. Also note that in the Gannt chart, the corresponding numbering for the work-packages will be used, and work-package names will not be rephrased.

Work Package Name	Numbering
Project: DProject	01-00-00
Sub-project: Detailed Design	01-01-00
Work-package: Database Tables Design	01-01-01
Work-package: Design of Database Interface Classes	01-01-02
Work-package: Design of Procedural Classes	01-01-03
Work-package: Design of Control Architecture	01-01-04
Work-package: Design of JSP Architecture	01-01-05
Work-package: Design of Visual Interface	01-01-06
Work-package: Design of Visual Classes	01-01-07
Sub-Project: Prototype Production	01-02-00
Work-package: Creation of Database Tables (Limited for prototype, including: User Account Tables, Project Tables; Company Tables, Task Tables)	01-02-01

Work-package: Implementation of Database Interface Classes 01-02-02
Work-package: Implementation of Visual Classes 01-02-03
(Limited for prototype, including: Upper Menu, Right Menu, Header, Footer)
Work-package: Implementation of Procedural Classes 01-02-04
(Limited for prototype, including: Session class(limited), Initializer class (limited), SqlConnection class, Project class(limited), User class (limited), Task class(limited))
Work-package: Implementation of JSP architecture 01-02-05
(Limited for prototype, including: login screen, projects screen, users screen, tasks screen)

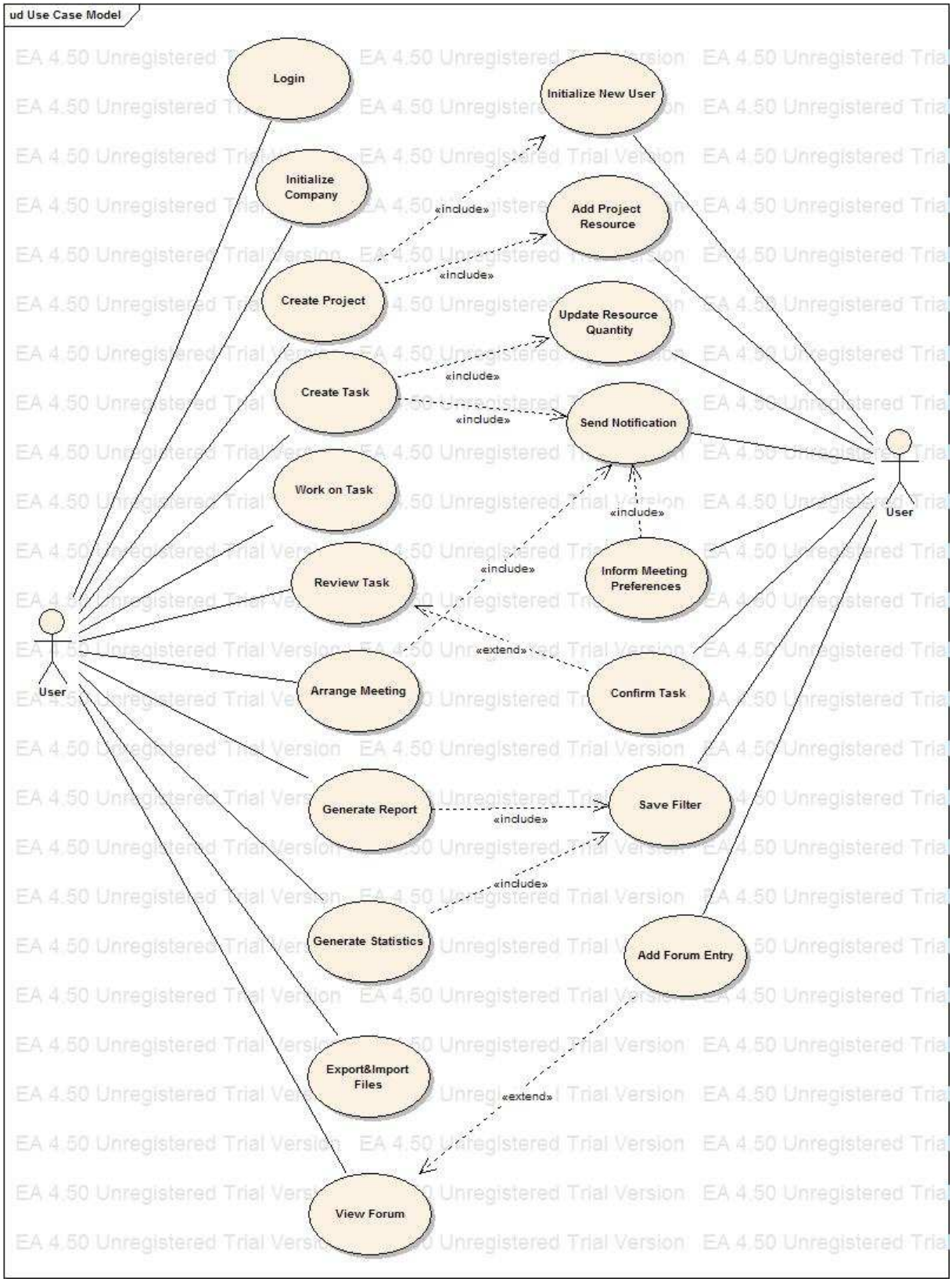
Sub-project: Implementation 01-03-00
Work-package: Implementation of Database 01-03-01
Work-package: Implementation for First Phase 01-03-02
(including Project, Task and Meeting Management)
Work-package: Implementation for Second Phase 01-03-03
(including Notifications, Reports, Statistics, and Forum)
Work-package: Implementation of Visual Classes 01-03-04
Work-package: Implementation of JSP pages 01-03-05

Sub-project: Documentation 01-04-00
Work-package: Preparation of User's manual 01-04-01
Work-package: Preparation of Help pages 01-04-02

Sub-project: Testing & Debugging 01-05-00
Work-package: Determination of Test-cases 01-05-01
Work-package: Application of Test-cases 01-05-02
Work-package: Debugging 01-05-03

2. MODELING

2.1 USE-CASE DIAGRAM



Flow of events for the Login use case	
Objective	To log in the system
Precondition	None
Main Flow	1 – The user enters his login id 2- The user enters his password 3 – The entered id and password are checked for validity 3 – The system creates a new session for the user and displays the main screen of the new user
Alternative Flows	At 3, if the entered id or password is invalid, the user is prompted to enter a new id or password
Post Condition	A new session is created for the user

Flow of events for the Initialize Company use case	
Objective	To set up a new company account in the system
Precondition	The user should have administrator access rights
Main Flow	1 – The user enters new company information 2 – The user selects an id and password for the company 3 – Access rights of the user are checked to see if they are enough or not 4– The entered information is checked for validity (i.e. non-existing company name, non-existing company id) 5 – The main screen of the user is displayed
Alternative Flows	At 2, if the user does not have enough access rights, he is not allowed to set up new company account At 3, if there is a conflict, the user is prompted to enter valid information into the conflicting fields
Post Condition	A new company information is saved in the database

Flow of events for the Initialize New User use case

Flow of events for the Initialize New User use case	
Objective	To set up a new user account into the system
Precondition	The user setting up the new account should have administrator access rights
Main Flow	1 – The user enters new account information 2 – The user enters an id and password for the new account 3 – Access rights of the user is checked whether they are enough or not 4 – The entered information for the new user is checked for validity (e.g. non-existing id) 5 - The user is assigned to projects, if necessary 6 – The main screen of the user is displayed
Alternative Flows	At 3, if the access rights of the user are not enough, the user is prompted stating that the intended operation can not be carried on At 4, if the information for the new user is invalid, the user is prompted to enter valid information
Post Condition	A new user account information is saved in the database

Flow of events for the Create Project use case	
Objective	To create a new project
Precondition	A company should be already selected and the user should have enough access rights
Main Flow	1 – The user sets up the information for the new project 2 – The access rights of the user is checked to see whether they are enough or not 3 – The entered information is checked for validity (e.g. non-existing project name) 4 – Existing users are assigned to the new project, if necessary 5 – Task groups, task types and task priorities are set up for the new project, if necessary 6 – Resource information is set up for the new project, if necessary 7 – The main screen of the user is displayed

Flow of events for the Create Project use case	
Alternative Flows	At 2, if the user does not have the necessary access rights, he is prompted stating that the operation can not be carried out At 3, if the entered information is not valid, the user is prompted to enter valid information to the invalid fields At 4, if a new user should be assigned to the project, a new user account is created
Post Condition	A new project is created and saved in the database

Flow of events for the Create Task use case	
Objective	To create a new task in a project
Precondition	A project should be selected already and the user should have enough access rights
Main Flow	1 – The user enters information for the new task 2 – The user assigns reviewers to the new task 3 – The user assigns user to the new task 4 – The user assigns resources to the new task, if necessary 5 – Files are attached to the new task by the user, if necessary 4 – A unique identifier is created and saved for the new task
Alternative Flows	None
Post Condition	A new task is saved in the database

Flow of events for the Work on Task use case	
Objective	To work on a particular task in a project
Precondition	A task should be selected already and the user should be assigned to the selected task
Main Flow	1 – The user opens the IN/OUT item to start working on a task 2 – The user selects preferences for the current IN/OUT item 3 – The user adds comments on the work done, if necessary 4 – The user closes the IN/OUT item when the work is completed 5 – The user sends the task to reviewers, if necessary
Alternative Flows	None

Flow of events for the Work on Task use case	
Post Condition	The new progress status of the task is saved and the task history is updated, working history of the user is updated

Flow of events for the Review Task use case	
Objective	To review the work done on task
Precondition	A task should be already selected, the user should be assigned as reviewer to the task and should be notified for review
Main Flow	1 – The user reviews the work done on task 2 – The user either accepts or rejects the work done 3 – The user that sent the task for review is notified on the reaction of the reviewer
Alternative Flows	None
Post Condition	Depending on the reaction of the reviewer, the work done is accepted or the user is obliged to do the work again, the status of the task is updated accordingly

Flow of events for the Confirm Task use case	
Objective	To confirm the task as completed or not
Precondition	A task should be already selected, the user should be assigned to the task as reviewer and should be notified for review
Main Flow	1 – The user reviews the work done on task 2 – The user either selects the task as completed or not 3 – The user that sent the task for review is notified depending on the reaction of the reviewer
Alternative Flows	None
Post Condition	Depending on the reaction of the reviewer, the task is marked as completed or not, and the status of the task is updated accordingly

Flow of events for the Generate Report use case	
Objective	To create and view a time report or task report

Flow of events for the Generate Report use case	
Precondition	None
Main Flow	1 – The user selects the type of the report to be created 2 – The user selects the filter to generate the report 3 – The user saves the filter, if necessary 4 – The report is generated depending on the filter 5 – The report is displayed
Alternative Flows	At 2, if the user makes invalid selections (e.g. non-existing date), the user is prompted to change the selections At 3, if there is a conflict in saving the filter (e.g. existing filter name), the user is prompted to remove the conflict
Post Condition	The report is generated and the filter is saved, if selected

Flow of events for the Generate Statistics use case	
Objective	To create and view statistics of a project
Precondition	A project should be already selected
Main Flow	1 – The user selects the filter to generate the statistics 2 – The user saves the filter, if necessary 3 – The statistics are generated depending on the filter 4 – The statistics are displayed
Alternative Flows	At 1, if the user makes invalid selections (e.g. non-existing date), the user is prompted to change the selections At 2, if there is a conflict in saving the filter (e.g. existing filter name), the user is prompted to remove the conflict
Post Condition	The statistics are generated and the filter is saved, if selected

Flow of events for the Save Filter use case	
Objective	To save a filter for later use
Precondition	None

Flow of events for the Save Filter use case	
Main Flow	1 – The user makes the selections for the different fields of the filter 2 – The user selects a name for the filter 3 – The user saves the filter
Alternative Flows	At 1, if the user makes an invalid selection (e.g. non-existing date), the user is prompted to change the selection At 2, if the user selects and existing date, he is prompted to change the name
Post Condition	A filter is saved in the system

Flow of events for the Arrange Meeting use case	
Objective	To arrange a meeting
Precondition	The user should have necessary access rights to arrange a meeting
Main Flow	1 – The user selects potential dates for the meeting 2 – The user selects the potential attendants of the meeting 3 – The user notifies the potential attendants on the potential dates 4 – Depending on the selections of the potential attendants, the user fixes the details of the meeting 5 – The user notifies the user stating the meeting details and attendants
Alternative Flows	None
Post Condition	A new meeting is created and its details are saved

Flow of events for the Inform Meeting Preference use case	
Objective	To inform the arranger about the selections about a meeting
Precondition	The user should have been notified by the arranger
Main Flow	1 – The user views the potential dates sent by the arranger 2 – The user notifies the arranger stating the dates suitable for him
Alternative Flows	None
Post Condition	The user preferences are sent to the arranger for further processing

Flow of events for the Export & Import Files use case	
Objective	To export & import files from/to the system
Precondition	A project or a report should be already selected
Main Flow	1 – The user selects whether to import/export a report or a whole project 2 – Depending on the selection of the user, either a report is imported/exported in the specified format, or the whole project is imported/exported as SQL statements
Alternative Flows	At 2, if the file to be imported/exported is invalid, the user is prompted stating that the file is invalid
Post Condition	Depending on the exported/imported file, either a new report file, or a new project is saved/opened

Flow of events for the View Forum use case	
Objective	To view forum threads
Precondition	None
Main Flow	1 – The user selects the forum he wants to view 2 – The user selects the thread to be viewed 3 – The thread that the user selected is displayed
Alternative Flows	None
Post Condition	A forum thread is displayed

Flow of events for the Add Forum Entry use case	
Objective	To add a new forum entry
Precondition	None
Main Flow	1 – The user selects the forum to which he wants to add a new entry 2 – The user selects the thread under which he wants to add a new entry 3 – The user adds the entry to the forum thread 4 – The thread is displayed with the new entry added

Flow of events for the Add Forum Entry use case	
Alternative Flows	None
Post Condition	A new entry is added to the forum

Flow of events for the Add Project Resource use case	
Objective	To add a new resource information to a project
Precondition	A project should be selected and the user should have enough access rights
Main Flow	1 – The user enters the information of the new resource 2 – The user enters the quantity of the new resource 3 – The user enters the unit price of the new resource
Alternative Flows	At 1, if one of the fields is conflicting (e.g. existing resource name), the user is prompted to change the conflicting field
Post Condition	New resource type and information is saved

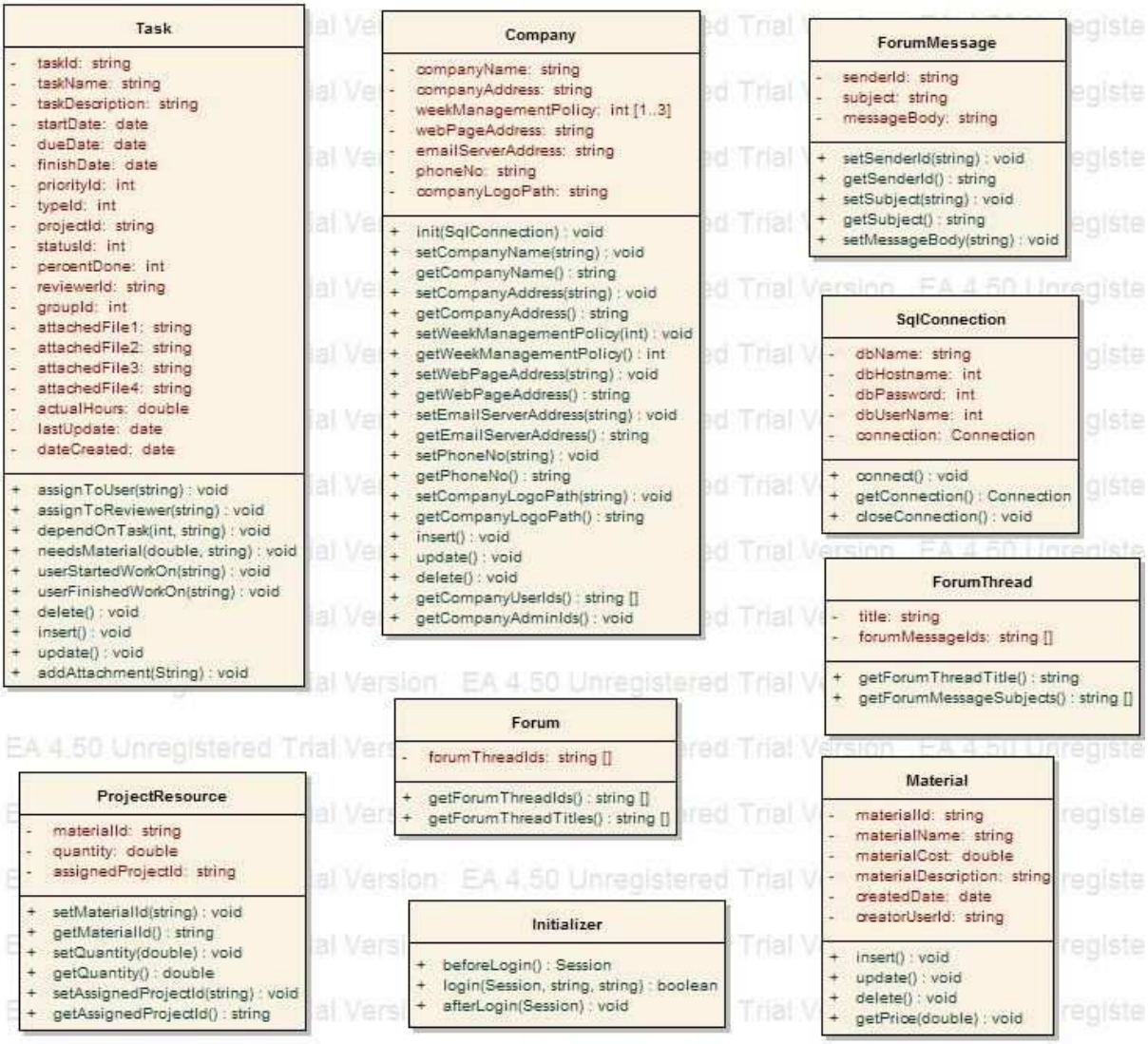
Flow of events for the Update Project Resource use case	
Objective	To update the information & quantity of a resource
Precondition	A project should be selected and the user should have enough access rights to make the update
Main Flow	1 – The user selects the resource to be updated 2 – The user selects the fields of the resource that are to be updated 3 – User updates the fields accordingly
Alternative Flows	At 3, if there is an invalid selection (e.g. resource quantity below zero), the user is prompted to change the selection
Post Condition	The information of the resource is updated and saved

Flow of events for the Send Notification use case	
Objective	To send notifications to other users in the system
Precondition	None

Flow of events for the Send Notification use case	
Main Flow	1 – The user enters the subject of the notification, if desired 2 – The user writes the main body of the notification, if desired 3 – Files are attached to the notification by the user, if desired 4 – The users selects the users to send the notification 5 – The user sends the notification
Alternative Flows	At 3, if the user tries to attach an invalid file (e.g. excess file size, corrupted file), the user is prompted about the error At 4, if the user tries to send the notification to a non-existing user, he is prompted about the error
Post Condition	A notification is sent to other users in the system

2.2 CLASS DIAGRAM

2.2.1 CLASSES


Task
- taskId: string
- taskName: string
- taskDescription: string
- startDate: date
- dueDate: date
- finishDate: date
- priorityId: int
- typeId: int
- projectId: string
- statusId: int
- percentDone: int
- reviewerId: string
- groupId: int
- attachedFile1: string
- attachedFile2: string
- attachedFile3: string
- attachedFile4: string
- actualHours: double
- lastUpdate: date
- dateCreated: date
+ assignToUser(string) : void
+ assignToReviewer(string) : void
+ dependOnTask(int, string) : void
+ needsMaterial(double, string) : void
+ userStartedWorkOn(string) : void
+ userFinishedWorkOn(string) : void
+ delete() : void
+ insert() : void
+ update() : void
+ addAttachment(String) : void
Company
- companyName: string
- companyAddress: string
- weekManagementPolicy: int [1..3]
- webPageAddress: string
- emailServerAddress: string
- phoneNo: string
- companyLogoPath: string
+ init(SqlConnection) : void
+ setCompanyName(string) : void
+ getCompanyName() : string
+ setCompanyAddress(string) : void
+ getCompanyAddress() : string
+ setWeekManagementPolicy(int) : void
+ getWeekManagementPolicy() : int
+ setWebPageAddress(string) : void
+ getWebPageAddress() : string
+ setEmailServerAddress(string) : void
+ getEmailServerAddress() : string
+ setPhoneNo(string) : void
+ getPhoneNo() : string
+ setCompanyLogoPath(string) : void
+ getCompanyLogoPath() : string
+ insert() : void
+ update() : void
+ delete() : void
+ getCompanyUserIds() : string []
+ getCompanyAdminIds() : void
ForumMessage
- senderId: string
- subject: string
- messageBody: string
+ setSenderId(string) : void
+ getSenderId() : string
+ setSubject(string) : void
+ getSubject() : string
+ setMessageBody(string) : void
SqlConnection
- dbName: string
- dbHostname: int
- dbPassword: int
- dbUserName: int
- connection: Connection
+ connect() : void
+ getConnection() : Connection
+ closeConnection() : void
ForumThread
- title: string
- forumMessageIds: string []
+ getForumThreadTitle() : string
+ getForumMessageSubjects() : string []
Forum
- forumThreadIds: string []
+ getForumThreadIds() : string []
+ getForumThreadTitles() : string []
ProjectResource
- materialId: string
- quantity: double
- assignedProjectId: string
+ setMaterialId(string) : void
+ getMaterialId() : string
+ setQuantity(double) : void
+ getQuantity() : double
+ setAssignedProjectId(string) : void
+ getAssignedProjectId() : string
Initializer
+ beforeLogin() : Session
+ login(Session, string, string) : boolean
+ afterLogin(Session) : void
Material
- materialId: string
- materialName: string
- materialCost: double
- materialDescription: string
- createdDate: date
- creatorUserId: string
+ insert() : void
+ update() : void
+ delete() : void
+ getPrice(double) : void


Project
- projectName: string
- projectId: string
- startDate: string
- estimatedDuration: integer
- budget: double
- clientId: string
- isProjectManager: boolean
- canApproveTime: boolean
- canSeeProjectDetails: boolean
- taskEditingLevel: int [1..9]
+ init(SqlConnection) : void
+ setProjectName(string) : void
+ getProjectName() : string
+ setProjectId(string) : void
+ getProjectId() : string
+ setStartDate(string) : void
+ getStartDate() : string
+ setEstimatedDuration(int) : void
+ getEstimatedDuration() : int
+ setBudget(double) : void
+ getBudget() : double
+ setClientId(string) : void
+ getClientId() : string
+ insert() : void
+ update() : void
+ delete() : void
+ assignToUser(string) : void
+ hasMaterial(double, string) : void
+ getProjectResources() : ProjectResource []
+ getProjectTaskIds() : string []
+ getProjectTaskNames() : string []
+ getProjectUserIds() : string []
+ getProjectUserNames() : string []
+ addTaskType(String) : void
+ addTaskPriority(String) : void
+ addTaskSatatus(String) : void
Session
- date: float = <current date>
- currentCompany: Company
- currentUser: User
- currentProject: Project
- currentNotifications: Notification [] = null
- dbConnection: SqlConnection
- currentMaterial: Material
- currentTask: int
- loggedInUser: User
- currentForumThread: ForumThread
- currentForumMessage: ForumMessage
+ init() : void
+ setDate(float) : void
+ getDate() : float
+ setCurrentCompany(Company) : void
+ getCurrentCompany() : Company
+ setCurrentUser(User) : void
+ getCurrentUser() : User
+ destroySession() : void
+ exportProject(int, string) : void
+ importProject(int, string) : void
+ retrieveProject(string) : Project
+ retrieveTask(string) : Task
+ retrieveMeeting(string) : Meeting
+ retrieveNotification(string) : Notification
+ retrieveUser(string) : User
+ retrieveFilter(string) : Filter
+ retrieveForumThreadTitles() : string []
+ retrieveForumMessagesInThread(string) : string []
+ retrieveMaterial(string) : Material
Filter
- filterId: string
- selectedUserId: string
- selectedProjectId: string
- selectedPaymentPolicyId: string
- selectedSalaryComparison: int [0..3]
- selectedSalaryQuantity: double
- selectedAgeComparison: int
- selectedAge: int
- selectedGender: int
- selectedGlobalProfile: int
- selectedTimeEntryMode: int
- selectedStartDateComparison: int
- selectedStartDate: date
- selectedFinishDateComp: int
- selectedFinishDate: date
- selectedDueDateComp: int
- selectedDueDate: date
- selectedPriorityId: string
- selectedTypeId: string
- selectedStatusId: string
- selectedPercentDoneComp: int
- selectedPercentDoneQuantity: double
- selectedReviewerId: string
- selectedGroupId: string
- selectedActualHoursComparison: int
- selectedActualHours: int
- selectedProjectStartDateComparison: int
- selectedProjectStartDate: date
- selectedProjectFinishDateComparison: int
- selectedProjectFinishDate: date
- selectedProjectDueDateComparison: int
- selectedPojectDueDate: date
- selectedProjectManagerId: string
+ insert() : void
+ update() : void
+ delete() : void
Meeting
- meetingId: string
- finalMeetingDate: date
- dateOption1: date
- dateOption2: date
- dateOption3: date
- dateOption4: date
- dateOption5: date
- creatorUserId: string
- creationDate: date
- attachement1: string
- attachement2: string
- attachement3: string
- lastReplyDate: date
+ insert() : void
+ update() : void
+ delete() : void
+ isLastReplyDatePassed() : boolean
+ isMeetingSettled() : boolean
Notification
- notificationId: string
- notificatedUser: string
- notificationType: string
- ownerOfAction: string
- dateOfAction: date
- attachedFile1: string
- attachedFile2: int
- attachedFile3: string
+ setNotificationId(string) : void
+ getNotificationId() : string
+ setNotificatedUser(string) : void
+ getNotificatedUser() : string
+ setNotificationType(int) : void
+ getNotificationType() : int
+ setOwnerOfAction(string) : void
+ getOwnerOfAction() : string
+ setDateOfAction(date) : void
+ getDateOfAction() : date
+ setAttachedFile1(string) : void
+ getAttachedFile1() : string
+ setAttachedFile2(string) : void
+ getAttachedFile2() : string
+ setAttachedFile3(string) : void
+ getAttachedFile() : string

AccessRights

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 6:23:50 PM. Modified on 12/6/2004 2:58:21 PM.

AccessRights Attributes

Attribute	Type	Notes
canAddProject	private : *int*	'canAddProject' attribute specifies whether a user can create a new project; 1 means user can create a project, 0 means user can not create a project
userDirectory	private : *int*	'userDirectory' attribute specifies what a user can see in his/her user directory. 1 means user can see all other users in the same company; 2 means user can see all other users in the same project; 3 means user can see only the dministrators; 4 means user can not see anyone so does not have a user directory.

Company

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 12:58:53 PM. Modified on 12/6/2004 10:26:50 PM.

'Company' class represents the company that the current logged-in user is a member of.

Company Attributes

Attribute	Type	Notes
companyName	private : *string*	
companyAddress	private : *string*	
weekManagementPolicy	private Range:1 to 3: *int*	'weekManagementPolicy' attribute holds integer values corresponding to the preference of the company on how to arrange working days of a week. That integer values have the range 1-3. The relations are 1:Monday-to-Friday, 2:Monday-to-Saturday,

		3:Monday-to-Sunday
webPageAddress	private : *string*	'webPageAddress' attribute holds the string representing the company's web page address.
emailServerAddress	private : *string*	'emailServerAddress' holds the mail-server address of the company that will be used to send e-mails using the company's server.
phoneNo	private : *string*	
companyLogoPath	private : *string*	'companyLogoPath' attribute holds the path to the image file that the company had submitted. This is used to show the company logo when it is a session of this company's users.

Company Methods

Method	Type	Notes
init (*SqlConnection*)	public: *void*	param: dbConnection [SqlConnection - in]
insert ()	public: *void*	'insert' method writes the information in this 'Company' class instance to the database, creating a new entry in the database table.
update ()	public: *void*	'update' method updates the record of this company in the database, using the current values of the attributes.
delete ()	public: *void*	'delete' method deletes the record of this company from the database.
getCompanyUserIds ()	public: *string []*	'getCompanyUserIds' returns the user ids who are members of this company.
getCompanyAdminIds ()	public: *void*	'getCompanyAdminIds' returns ids of the administrators of this company.

Filter

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:06:58 PM. Modified on 12/6/2004 9:10:25 PM.

Filter Attributes

Attribute	Type
filterId	private : *string*

selectedUserId	private : *string*
selectedProjectId	private : *string*
selectedPaymentPolicyId	private : *string*
selectedSalaryComparison	private Range:0 to 3: *int*
selectedSalaryQuantity	private : *double*
selectedAgeComparison	private : *int*
selectedAge	private : *int*
selectedGender	private : *int*
selectedGlobalProfile	private : *int*
selectedTimeEntryMode	private : *int*
selectedStartDateComparison	private : *int*
selectedStartDate	private : *date*
selectedFinishDateComp	private : *int*
selectedFinishDate	private : *date*
selectedDueDateComp	private : *int*
selectedDueDate	private : *date*
selectedPriorityId	private : *string*
selectedTypeId	private : *string*
selectedStatusId	private : *string*
selectedPercentDoneComp	private : *int*
selectedPercentDoneQuantity	private : *double*
selectedReviewerId	private : *string*

selectedGroupId	private : *string*
selectedActualHoursComparison	private : *int*
selectedActualHours	private : *int*
selectedProjectStartDateComparison	private : *int*
selectedProjectStartDate	private : *date*
selectedProjectFinishDateComparison	private : *int*
selectedProjectFinishDate	private : *date*
selectedProjectDueDateComparison	private : *int*
selectedPojectDueDate	private : *date*
selectedProjectManagerId	private : *string*

Filter Methods

Method	Type	Notes
insert ()	public: *void*	'insert' method inserts the filter information to the database creating a new entry.
update ()	public: *void*	'update' method upates the database record of this filter, using the new attribute values.
delete ()	public: *void*	'delete' method deletes the database record of this filter.

Forum

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:07:04 PM. Modified on 12/7/2004 6:30:49 PM.

Forum Attributes

Attribute	Type	Notes
forumThreadIds	private : *string []*	

Forum Methods

Method	Type	Notes
getForumThreadIds ()	public: *string []*	
getForumThreadTitles ()	public: *string []*	

ForumMessage

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/6/2004 8:35:57 PM. Modified on 12/7/2004 6:30:49 PM.

ForumMessage Attributes

Attribute	Type	Notes
senderId	private : *string*	
subject	private : *string*	
messageBody	private : *string*	

ForumMessage Methods

Method	Type	Notes
setSenderId (*string*)	public: *void*	param: sender [string - in]
getSenderId ()	public: *string*	
setSubject (*string*)	public: *void*	param: subject [string - in]
getSubject ()	public: *string*	
setMessageBody (*string*)	public: *void*	param: messageBody [string - in]

ForumThread

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model
Details: Created on 12/6/2004 8:35:47 PM. Modified on 12/8/2004 1:25:23 PM.

ForumThread Attributes

Attribute	Type	Notes
title	private : *string*	
forumMessageIds	private : *string []*	

ForumThread Methods

Method	Type	Notes
getForumThreadTitle ()	public: *string*	
getForumMessageSubjects ()	public: *string []*	

Initializer

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 8:25:31 PM. Modified on 12/6/2004 10:28:38 PM.

'Initializer' class is a helper class for logging in and initializing the session variable.

Initializer Methods

Method	Type	Notes
beforeLogin ()	public: *Session*	'beforeLogin' is called when a login screen is showed to the user but before he/she logs in.
login (*Session, string, string*)	public: *boolean*	param: session [Session - in] param: userPassword [string - in] param: userName [string - in] 'login' takes the password and loginId and checks to see if the id and password is valid and consistent. Reeturns 'true' if consistent, 'false' if inconsistent or invalid.
afterLogin (*Session*)	public: *void*	param: session [Session - in] 'afterLogin' is called after 'login' method returns as 'true' and it initializes all rquired attributes of the current session for a logged in user.

Material

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:07:22 PM. Modified on 12/7/2004 6:30:49 PM.

'Material' class represents a specific material type that has been created to be used in projects.

Material Attributes

Attribute	Type	Notes
materialId	private : *string*	
materialName	private : *string*	
materialCost	private : *double*	
materialDescription	private : *string*	
createdDate	private : *date*	
creatorUserId	private : *string*	

Material Methods

Method	Type	Notes
insert ()	public: *void*	'insert' method inserts the material information to the database creating a new entry.
update ()	public: *void*	'update' method upates the database record of this material, using the new attribute values.
delete ()	public: *void*	'delete' method deletes the database record of this material.
getPrice (*double*)	public: *void*	param: quantity [double - in] 'getPrice' method returns the price of this material for the specified quantity using the formula quantity*materialCost.

Meeting

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model

Details: Created on 12/4/2004 6:58:22 PM. Modified on 12/7/2004 6:30:49 PM.

Meeting Attributes

Attribute	Type	Notes
meetingId	private : *string*	
finalMeetingDate	private : *date*	
dateOption1	private : *date*	
dateOption2	private : *date*	
dateOption3	private : *date*	
dateOption4	private : *date*	
dateOption5	private : *date*	
creatorUserId	private : *string*	
creationDate	private : *date*	
attachement1	private : *string*	
attachement2	private : *string*	
attachement3	private : *string*	
lastReplyDate	private : *date*	

Meeting Methods

Method	Type	Notes
insert ()	public: *void*	'insert' method inserts the meeting information to the database creating a new entry.
update ()	public: *void*	'update' method upates the database record of this meeting, using the new attribute values.
delete ()	public: *void*	'delete' method deletes the database record of this meeting.
isLastReplyDatePassed ()	public: *boolean*	'isLastReplyDatePassed' returns true if the last reply/decision date for the meeting has passed; false otherwise.
isMeetingSettled ()	public: *boolean*	'isMeetingSettled' returns true if the date of this meeting had been decided and settled

		by all attendants; false otherwise.

Notification

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:06:47 PM. Modified on 12/6/2004 10:04:54 PM.

Notification Attributes

Attribute	Type	Notes
notificationId	private : *string*	
notificatedUser	private : *string*	
notificationType	private : *string*	
ownerOfAction	private : *string*	
dateOfAction	private : *date*	
attachedFile1	private : *string*	
attachedFile2	private : *int*	
attachedFile3	private : *string*	

Project

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:06:28 PM. Modified on 12/8/2004 1:56:14 PM.

Project Attributes

Attribute	Type	Notes
projectName	private : *string*	'name' is the project's name in real life.
projectId	private :	

	string	
startDate	private : *string*	'startDate'
estimatedDuration	private : *integer*	'estimatedDuration' attribute specifies the project's estimated duration.
budget	private : *double*	'budget' attribute holds the project's budget.
clientId	private : *string*	'clientId' attribute specifies this project's client.
isProjectManager	private : *boolean*	'isProjectManager' specifies if the current user is a manager of this project. 'true' means he/she is a manager, 'false' means he/she is not.
canApproveTime	private : *boolean*	'canApproveTime' specifies if the current user can approve users' timesheets in this project. 'true' means he/she can, 'false' means he/she can not.
canSeeProjectDetails	private : *boolean*	'canSeeProjectDetails' specifies whether the current user can see all tasks and meeetings in the project, or can see only the ones that he/she is assigned to. 'true' means he/she can see all, 'false' means he/she can not.
taskEditingLevel	private Range:1 to 9: *int*	'taskEditingLevel' specifies users' permissions about task editing. 1 means read-only permission, 2 means limited task editing, 3 means limited task editing and file attachement creation/deletion, 4 means partial task editing, 5 means partial task editing and deleting the tasks that he/she created, 6 means full control task editing, 7 means full control task editing and deleting the tasks that he/she created, 8 means full controol task editing and creating tasks and deleting his/her own tasks, 9 means full control task editing and creating deleting all tasks.

Project Methods

Method	Type	Notes
init (*SqlConnection*)	public: *void*	param: dbConnection [SqlConnection - in]
insert ()	public: *void*	'insert' method writes the information in this 'Project' class instance to the database, creating a new entry in the database table.
update ()	public: *void*	'update' method updates the record of this project in the database, using the current values of the attributes.

delete ()	public: *void*	'delete' method deletes the record of this project from the database.
assignToUser (*string*)	public: *void*	param: userId [string - in] 'assignToUser' method add the relation to the database so that the user specified with the userId becomes a member of this project.
hasMaterial (*double, string*)	public: *void*	param: quantity [double - in] param: materialId [string - in] 'hasMaterial' method marks the database so that this project has specified quantity of the specified material.
getProjectResources ()	public: *ProjectResource []*	'getProjectResources' method queries this projects resources from the database and returns them.
getProjectTaskIds ()	public: *string []*	'getProjectTaskIds' returns the poject's task ids.
getProjectTaskNames ()	public: *string []*	'getProjectTaskNames' returns the project's task names.
getProjectUserIds ()	public: *string []*	'getProjectUserIds' returns the users' ids who are assigned to the project.
getProjectUserNames ()	public: *string []*	'getProjectUserNames' returns the users' names assigned to this project.
addTaskType (*String*)	public: *void*	param: tasktype [String - in] 'addTaskType' method is used to add newly defined task types to this project.
addTaskPriority (*String*)	public: *void*	param: taskptiority [String - in] 'addTaskPriority' method is used to add newly defined priority types to this project.
addTaskSatatus (*String*)	public: *void*	param: taskstatus [String - in] 'addTaskStatus' method is used to add newly defined status types to this project.

ProjectResource

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:07:30 PM. Modified on 12/7/2004 6:30:49 PM.

ProjectResource Attributes

Attribute	Type	Notes
materialId	private : *string*	'materialId' specifies what type of material is included in this project resourcce.
quantity	private : *double*	'quantity' specifies the quantity of the material.
assignedProjectId	private : *string*	'assignedProjectId' specifies which project this resource belongs to.

ProjectResource Methods

Method	Type	Notes
setMaterialId (*string*)	public: *void*	param: materialId [string - in]
getMaterialId ()	public: *string*	
setQuantity (*double*)	public: *void*	param: quantity [double - in]
getQuantity ()	public: *double*	
setAssignedProjectId (*string*)	public: *void*	param: projectId [string - in]
getAssignedProjectId ()	public: *string*	

Session

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 12:15:36 PM. Modified on 12/8/2004 7:48:55 PM.

'Session' class represents a unique session started by a user. This class is always active and in association with all other classes. Session class is terminated when the user terminates the web session.

Session Attributes

Attribute	Type	Notes
date	private : *float*	'date' attribute shows the time that the session is started by the user. It is set to the server system time initially. Initial Value: <current date>;
currentCompany	private : *Company*	'currentCompany' attribute is a pointer to an instance of the Company class, which is the logged-in user's company.
currentUser	private : *User*	'currentUser' attribute is a pointer to an instance of the User class, which is a user being modified, or shown to the user.
currentProject	private :	'currentProject' attribute is a pointer to an

	Project	instance of the Project class, which is the one the user is currently working on. It is initially set to null.
currentNotifications	private : *Notification []*	'currentNotifications' attribute is an array of instances of the Notification class, which are waiting for the currenly working user. It is initially set to null. Initial Value: null;
dbConnection	private : *SqlConnection*	'dbConnection' is an instance of the class SqlConnection and is used to connect tothe database in this session.
currentMaterial	private : *Material*	'currentMaterial' attribute represents the currently active material in the session.
currentTask	private : *int*	'currentTask' attribute represents the currently active task in the session.
loggedInUser	private : *User*	'loggedInUser' attribute is a pointer to an instance of the User class, which is the logged-in user.
currentForumThread	private : *ForumThread*	
currentForumMessage	private : *ForumMessage*	

Session Methods

Method	Type	Notes
init ()	public: *void*	
setDate (*float*)	public: *void*	param: date [float - in]
getDate ()	public: *float*	
setCurrentCompany (*Company*)	public: *void*	param: currentCompany [Company - in]
getCurrentCompany ()	public: *Company*	
setCurrentUser (*User*)	public: *void*	param: currentUser [User - in]
getCurrentUser ()	public: *User*	
destroySession ()	public: *void*	'destroySession' destroys the current session if the user logs-out.
exportProject (*int, string*)	public: *void*	param: format [int - in] param: projectId [string - in] 'exportProject' exports the specified project using the specified format.
importProject (*int, string*)	public: *void*	param: fileFormat [int - in] param: fileName [string - in] 'importProject' imports a project using the

		specified file name and the specified file format.
retrieveProject (*string*)	public: *Project*	param: projectId [string - in] 'retrieveProject' returns the Project object that has the specified project id.
retrieveTask (*string*)	public: *Task*	param: taskId [string - in] 'retrieveTask' returns the Task object that has the specified task id.
retrieveMeeting (*string*)	public: *Meeting*	param: meetingId [string - in] 'retrieveMeeting' returns the Meeting object that has the specified meeting id.
retrieveNotification (*string*)	public: *Notification*	param: notificationId [string - in] 'retrieveNotification' mehod returns the Notification object that has the specified notification id.
retrieveUser (*string*)	public: *User*	param: userId [string - in] 'retrieveUser' mehod returns the User object that has the specified user id.
retrieveFilter (*string*)	public: *Filter*	param: filterId [string - in] 'retrieveFilter' mehod returns the Filter object that has the specified filter id.
retrieveForumThreadTitles ()	public: *string []*	'retrieveNotification' mehod returns the Notification object that has the specified notification id.
retrieveForumMessagesInThread (*string*)	public: *string []*	param: messageId [string - in] 'retrieveForumMessage' mehod returns the ForumMessage object that has the specified forum message id.
retrieveMaterial (*string*)	public: *Material*	param: materialId [string - in] 'retrieveMaterial' mehod returns the Material object that has the specified material id.

SqlConnection

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 8:00:28 PM. Modified on 12/6/2004 10:28:38 PM.

SqlConnection class holds the attributes needed that are to connect to the database and acts as a wrapper class for connecting to the database.

SqlConnection Attributes

Attribute	Type	Notes
dbName	private : *string*	
dbHostname	private : *int*	
dbPassword	private : *int*	
dbUserName	private : *int*	
connection	private : *Connection*	'connection' is the actual database connection that is established by the SqlConnection class. Its type depends on the programmig language that will be used. In our case it will most probably be of type java.sql.Connection.

SqlConnection Methods

Method	Type	Notes
connect ()	public: *void*	'connect' connects to the database using dbName, dbHostName, dbPassword, and dbUsername attributes of SqlConnection class and initializes the connection variable.
getConnection ()	public: *Connection*	'getConnection' returns the connection variable which was connected to the database.
closeConnection ()	public: *void*	'closeConnection' closes the database connection.

Task

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 6:58:10 PM. Modified on 12/8/2004 1:29:58 PM.

Task Attributes

Attribute	Type	Notes
taskId	private :	

	string	
taskName	private : *string*	
taskDescription	private : *string*	
startDate	private : *date*	
dueDate	private : *date*	
finishDate	private : *date*	
priorityId	private : *int*	
typeId	private : *int*	
projectId	private : *string*	
statusId	private : *int*	
percentDone	private : *int*	
reviewerId	private : *string*	
groupId	private : *int*	
attachedFile1	private : *string*	
attachedFile2	private : *string*	
attachedFile3	private : *string*	
attachedFile4	private : *string*	
actualHours	private : *double*	
lastUpdate	private : *date*	
dateCreated	private : *date*	

Task Methods

Method	Type	Notes
assignToUser (*string*)	public: *void*	param: userId [string - in] 'assignToUser' method is used to assign this task to a user specified by the user-id parameter.
assignToReviewer	public: *void*	param: reviewerId [string - in]

(*string*)		'assignToReviewer' method is used to assign this task to the reviewer specified by the reviewerid parameter.
dependOnTask (*int, string*)	public: *void*	param: dependencyType [int - in] param: taskId [string - in] 'dependOnTask' method marks the database so that this task will depend on the task specified by the taskId, by the relation specified by the dependencyType.
needsMaterial (*double, string*)	public: *void*	param: quantity [double - in] param: materialId [string - in] 'needsMaterial' assigns the specified quantity of the specified material to this task. The corresponding price will be decreased from the project budget.
userStartedWorkOn (*string*)	public: *void*	param: userId [string - in] 'userStartedWorkOn' method marks the database showing that the specified user started working on this task.
userFinishedWorkOn (*string*)	public: *void*	param: userId [string - in] 'userFinishedWorkOn' method marks the database showing that the specified user finished working on this task.
delete ()	public: *void*	
insert ()	public: *void*	
update ()	public: *void*	
addAttachment (*String*)	public: *void*	param: filepath [String - in]

User

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 12/4/2004 2:06:20 PM. Modified on 12/8/2004 7:53:59 PM.

User Attributes

Attribute	Type	Notes
userId	private : *string*	'userId' attribute holds the id that is used as a unique key to specify a user. this attribute is also used as a login-id.
password	private :	'password' attribute holds the user's

	string	password.
name	private : *string*	'name' attribute holds the user's real life name.
middleName	private : *string*	'middleName' attribute holds the user's real life middle name.
surname	private : *string*	'surname' attribute holds the user's real life surname.
birthDate	private : *string*	'birthDate' attribute holds the user's real life birth date.
speciality	private : *string*	'speciality' attribute represents what the user is specialized in as an employee.
address	private : *string*	'address' attribute holds he user's real life address.
sex	private Range:1 to 3: *int*	'sex' represents the user's sexual gender. It can only have three values; 1:male, 2:female, 3:other
emailAddress	private : *string*	'emailAddress' attribute holds the user's e-ail address.
photo	private : *string*	'photo' attribute holds te path to the image file which includes the user's photo if submitted.
paymentPolicy	private Range:1 to 3: *int*	'paymentPolicy' field holds integer values ranging from 1 to 3, representing three different payment policies. These are 1, if the user is paid monthly; 2, if the user is paid weekly; 3, if the user is paid on an hourly basis.
paymentAmount	private : *double*	'paymentAmount' attribute holds the amount that is paid to the user, for a month (if payment policy is monthly), for a week (if payment policy is weekly), for an hour (if the user is paid for hourly work).
emailNotificationForNewTaskPreference	private Range:0 to 1: *int*	If 'emailNotificationForNewTaskPreference' attribute has the value 1, the user is notified via e-mail whenever a task is assigned to him/her; if this attribute has the value 0 he/she is not notified. Initial Value: 1;
numOfTasksPerPagePreference	private Range:1 to 50: *int*	'numOfTasksPerPagePreference' attribute specifies the user's preference so that, when he/she views the tasks of a project, they are shown in groups of this quantity. Initially it is set to 10 so that in a page at most ten tasks are shown. Initial Value: 10;
numOfMonthsPerPagePrefrence	private : *int*	'numOfMonthsPerPage' attribute specifies the user's preference so that, when he/she views a monthly gantt chart, at most this much month will be shown in a page.

		Initially this will be set to 4, so that in a monthly gantt chart, 4 months at a page will be shown. Initial Value: 4;
numOfWeeksPerPagePreference	private : *int*	'numOfWeeksPerPage' attribute specifies the user's preference so that, when he/she views a gantt chart in weekly mode, at most this much week will be shown in a page. Initially this will be set to 4, so that in a weekly gantt chart, 4 weeks at a page will be shown. Initial Value: 4;
userProjects	private : *Project []*	'userProjects' array holds instances for all the projects that this user is a member of. This field is set only if this user is the current user of the session.
canAddProject	private : *boolean*	'canAddProject' attribute specifies if the user has the permission to create a new project for his/her company.
userDirectory	private Range:1 to 4: *int*	'userDirectory' attribute specifies what a user can see in his/her user directory. 1 means user can see all other users in the same company; 2 means user can see all other users in the same project; 3 means user can see only the administrators; 4 means user can not see anyone so does not have a user directory.
globalAccessRight	private : *int*	'globalAccessRight' attribute specifies this user's global permissions in the system. A value of '1' means the user is a client of a project not an employee; '2' means the user is an administrator and have all the global rights; '3' means the user is a normal user and his/her permissions are further specifiedy other attributes.

User Methods

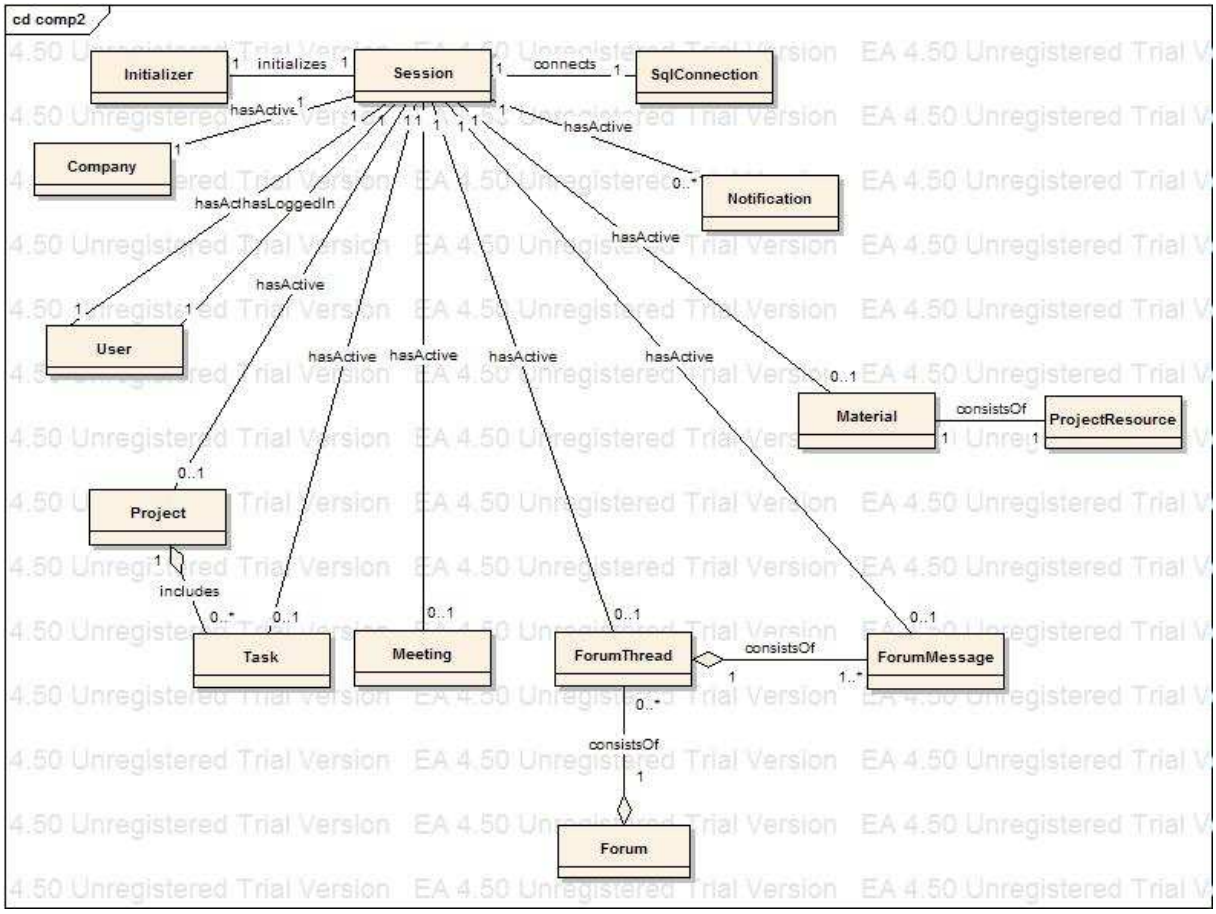
Method	Type	Notes
init (*SqlConnection*)	public: *void*	param: dbConnection [SqlConnection - in]
setUserProjects (*Project []*)	public: *void*	param: projects [Project [] - in]
getUserProjects ()	public: *Project []*	
insert ()	public: *void*	'insert' method writes the information in this 'User' class instance to the database, creating a new entry in the database table.
update ()	public: *void*	'update' method updates the record of this

		user in the database, using the current values of the attributes.
delete ()	public: *void*	'delete' method deletes the record of this user from the database.
setAccessRightsOfUser (*string, string, int, boolean, boolean*)	public: *void*	param: userId [string - in] param: projectId [string - in] param: editTaskLevel [int - in] param: canApproveTime [boolean - in] param: isProjectManager [boolean - in] 'setAccessightsOfUser' sets access rights of the user with the specified id to the specified access rights.
createNewProject (*int, string, string, string, double, date, date, date, string, string, string*)	public: *void*	param: projectTypeId [int - in] param: contactEmail [string - in] param: contactPhone [string - in] param: contactName [string - in] param: budget [double - in] param: dueDate [date - in] param: finishDate [date - in] param: startDate [date - in] param: projectDescription [string - in] param: projectName [string - in] param: projectId [string - in] 'createNewProject' method creates a new project by the spcified attributes.
createUser (*int, double, int, int, int, int, boolean, int, boolean, int, string, string, string, int, date, string, string, string, string, string, string*)	public: *void*	param: userDepartmentId [int - in] param: paymentAmount [double - in] param: paymentPolicy [int - in] param: numOfWeeksPerPage [int - in] param: NumOfMonthsPerPage [int - in] param: NumOfTasksPerPage [int - in] param: emailNotificationForNewTask [boolean - in] param: userDirectory [int - in] param: canAddProject [boolean - in] param: globalAccessRight [int - in] param: address [string - in] param: photo [string - in] param: speciality [string - in] param: gender [int - in] param: birthDate [date - in] param: email [string - in] param: phone [string - in] param: lastName [string - in] param: firstName [string - in] param: password [string - in] param: userId [string - in]

		'createUser' method creates a new user with the specified attribbutes.
ceateNewTask (*string, string, string, string, int, string, int, int, string, int, int, date, date, date, string, string, string*)	public: *void*	param: atachedFile4 [string - in] param: attachedFile3 [string - in] param: attachedFile2 [string - in] param: attachedFile1 [string - in] param: groupId [int - in] param: reviewerId [string - in] param: percentDone [int - in] param: statusId [int - in] param: projectId [string - in] param: priorityId [int - in] param: typeId [int - in] param: dueDate [date - in] param: finishDate [date - in] param: startDate [date - in] param: taskDescription [string - in] param: taskName [string - in] param: taskId [string - in]
createMaterial (*string, doule, string, string*)	public: *void*	param: materialDescription [string - in] param: materialCost [doule - in] param: materiaName [string - in] param: materialId [string - in] 'createMaterial' methood is used to create new materials with the specified attributes.
createMeeting (*date, string, string, string, date, date, date, date, date, date, string*)	public: *void*	param: lastReplyDate [date - in] param: attachement3 [string - in] param: attachement2 [string - in] param: attachement1 [string - in] param: dateOption5 [date - in] param: dateOption4 [date - in] param: dateOption3 [date - in] param: dateOption2 [date - in] param: dateOption1 [date - in] param: finalMeetingDate [date - in] param: meetingId [string - in] 'createMeeting' method is used to create a new meeting with the specified attributes.
buysMaterial (*double, double, string*)	public: *void*	param: unitPrice [double - in] param: quantity [double - in] param: materialId [string - in] 'buysMaterial' method marks the database so that the purchase information is recorded.
setUserPrefencesForMeet	public: *void*	param: meetingId [string - in] param: option5 [int - in]

ing (*string, int, int, int, int, int*)		param: option4 [int - in] param: option3 [int - in] param: option2 [int - in] param: option1 [int - in] 'setUserPreferencesForMeeting' method marks the database according to the preferences made by this user for a meeting he/she was assigned to.
approveTask ()	public: *void*	'approveTask' method is called when the reviewer approves a task of another user.
assignToProject (*int, boolean, boolean, String*)	public: *void*	param: taskeditlevel [int - in] param: istimeapprover [boolean - in] param: isprojectmanager [boolean - in] param: projectid [String - in]
createCompany (*string, string, string, int, string, string*)	public: *void*	param: companyLogoPath [string - in] param: phoneno [string - in] param: webPageAddress [string - in] param: weekManagementPolicy [int - in] param: companyAddress [string - in] param: compayName [string - in]
sendNotification (*string, string, string, string, string*)	public: *void*	param: attachedfile3 [string - in] param: attachedfile2 [string - in] param: attachedfile1 [string - in] param: notificatedUser [string - in] param: notificationId [string - in]

2.2.2 CLASS ASSOCIATIONS



The associations between the classes are shown on the diagram above. The class names are used without the attributes and operations of the classes to create a clearer diagram. The associations described here are only the static associations between classes, in terms of aggregation, inclusion and inheritance. The dynamic relations between classes are presented in the sequence diagram.

Session 'hasActive' Company

For every session that is started for a user, we will hold an instance of the Company class which represents the company of the logged-in user.

Session 'hasLoggedIn' User

For every session, we will hold an instance of the User class which represents the logged-in user.

Session 'hasActive' User

During a session, if the user wants to create a new user, or wants to modify/delete the records of an existing user, then the user whose records are being modified (or created) will be held as the 'currentUser' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Project

During a session, if the user wants to create a new project, or wants to modify/delete the records of an existing project, then the project whose records are being modified (or created) will be held as the 'currentProject' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Task

During a session, if the user wants to create a new task, or wants to modify/delete the records of an existing task, then the task whose records are being modified (or created) will be held as the 'currentTask' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Meeting

During a session, if the user wants to create a new meeting, or wants to modify/delete the records of an existing meeting, then the meeting whose records are being modified (or created) will be held as the 'currentMeeting' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' ForumThread

During a session, if the user wants to view the contents of a forum thread, then the forum thread whose records are being viewed will be held as the 'currentForumThread' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' ForumMessage

During a session, if the user wants to create a new forum message, or wants to view an existing message, then the message whose records are being created (or viewed) will be held as the 'currentForumMessage' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Material

During a session, if the user wants to create a new type of material, or wants to modify/delete the records of an existing material, then the material whose records are being modified (or created) will be held as the 'currentMaterial' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Notification

During a session, if the user wants to create a new notification, or wants to view those notifications (which may be more than one), then the notifications whose records are being

created or being viewed will be held as the 'currentNotifications' in the session. This determines the 'hasActive' relationship.

Initializer 'initializes' Session

For every session that is started for a user, an Initializer class is held to handle the initializations both before and after the login. This determines the 'initializes' relationship.

SqlConnection 'connects' Session

For every session that is started for a user, a SqlConnection class is held to handle the database connections both before and after the login. This determines the 'connects' relationship.

Project 'includes' Task

For every project there are zero or more tasks that belong to the project.

Forum 'consistsOf' ForumThread

In a Forum, there may be zero or more ForumThreads. That is; a forum consists of threads. This also determines the aggregation character of the association.

ForumThread 'consistsOf' ForumMessages

In a ForumThread, there may be one or more ForumMessages. That is; a forum thread consists of messages. This also determines the aggregation character of the association.

ProjectResource 'consistsOf' Material

Every project resource consists of some material. There may be only one material type in a project resource.

2.3 SEQUENCE DIAGRAMS

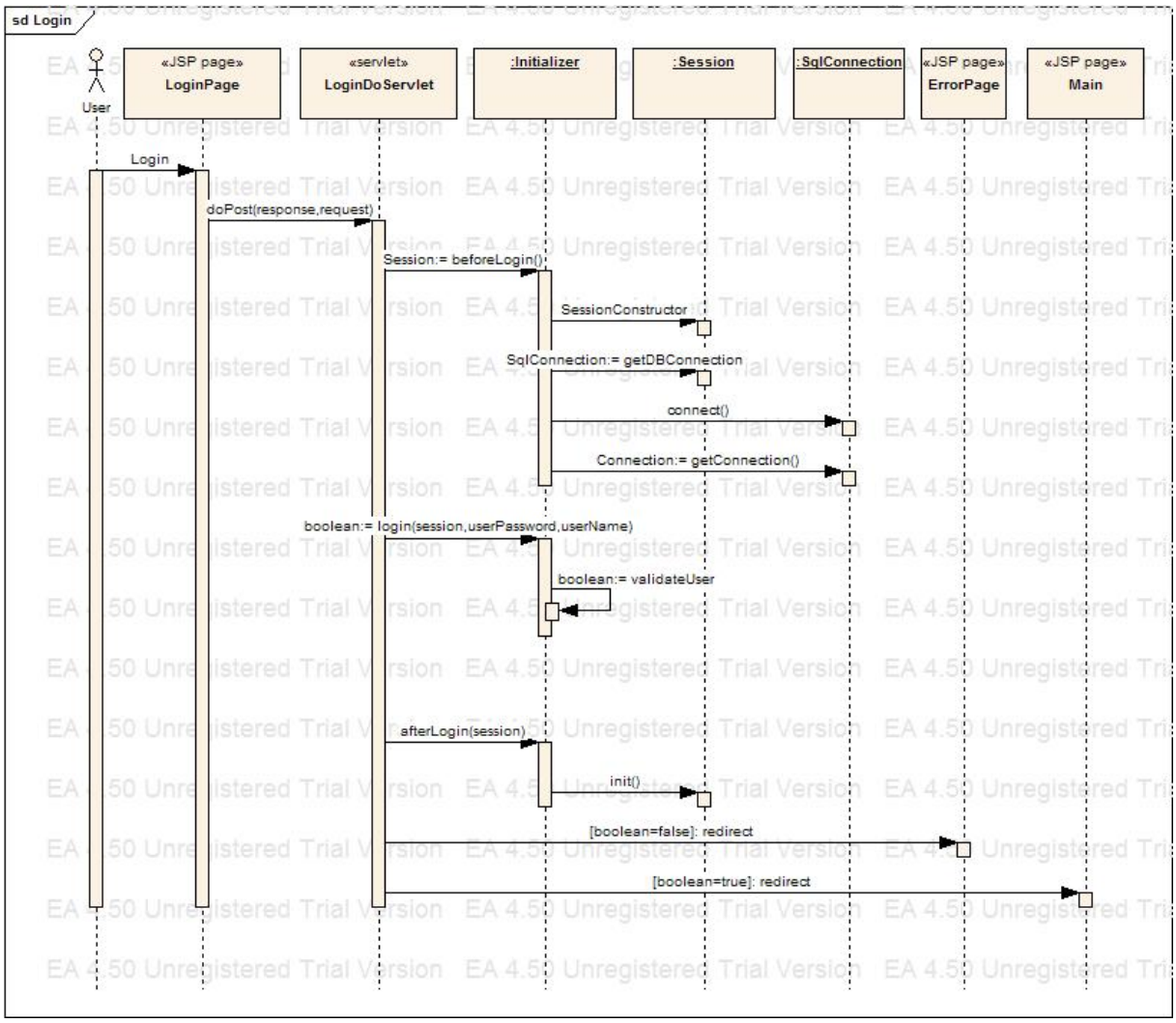


Figure 1 : Login

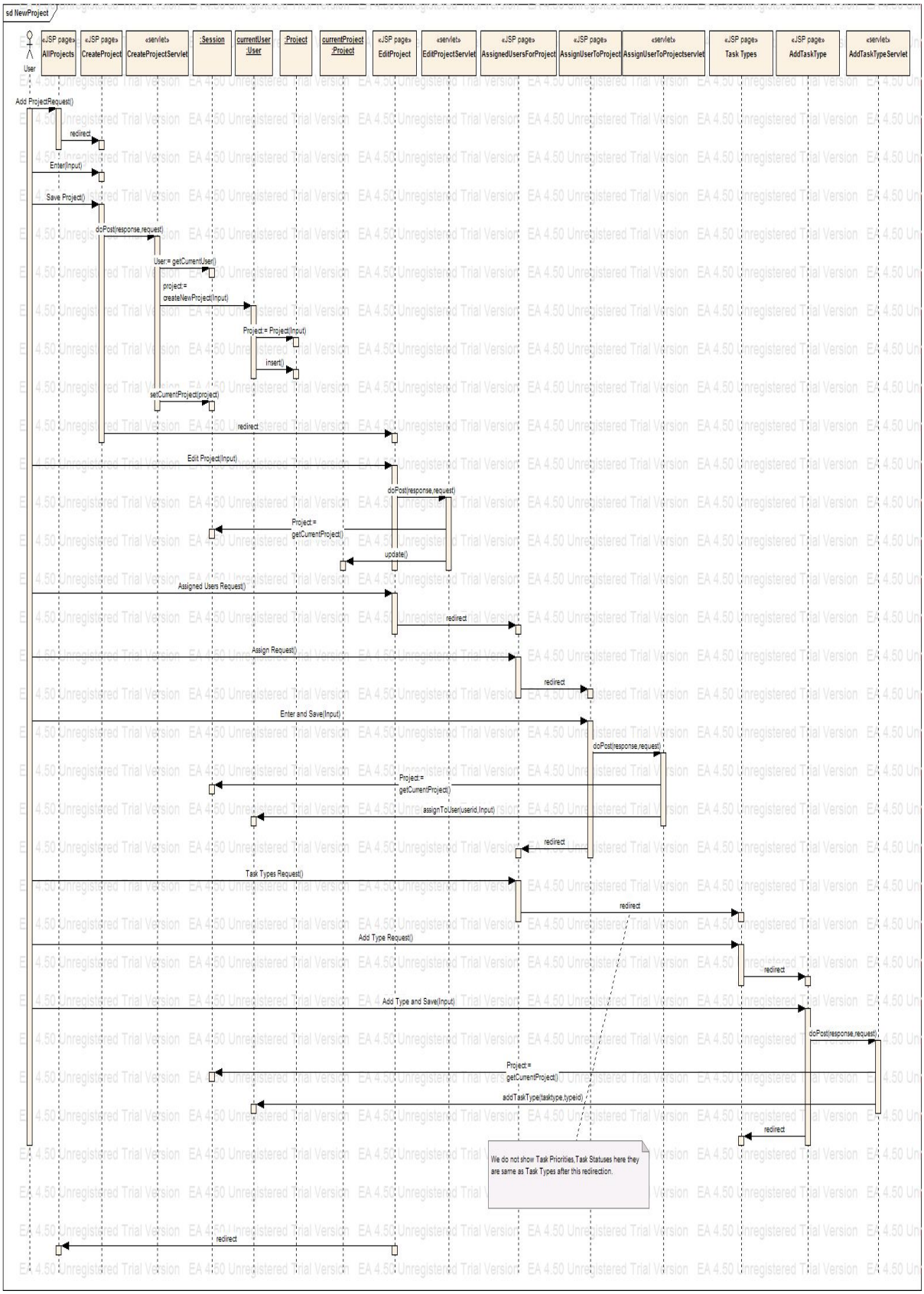


Figure 2 : NewProject

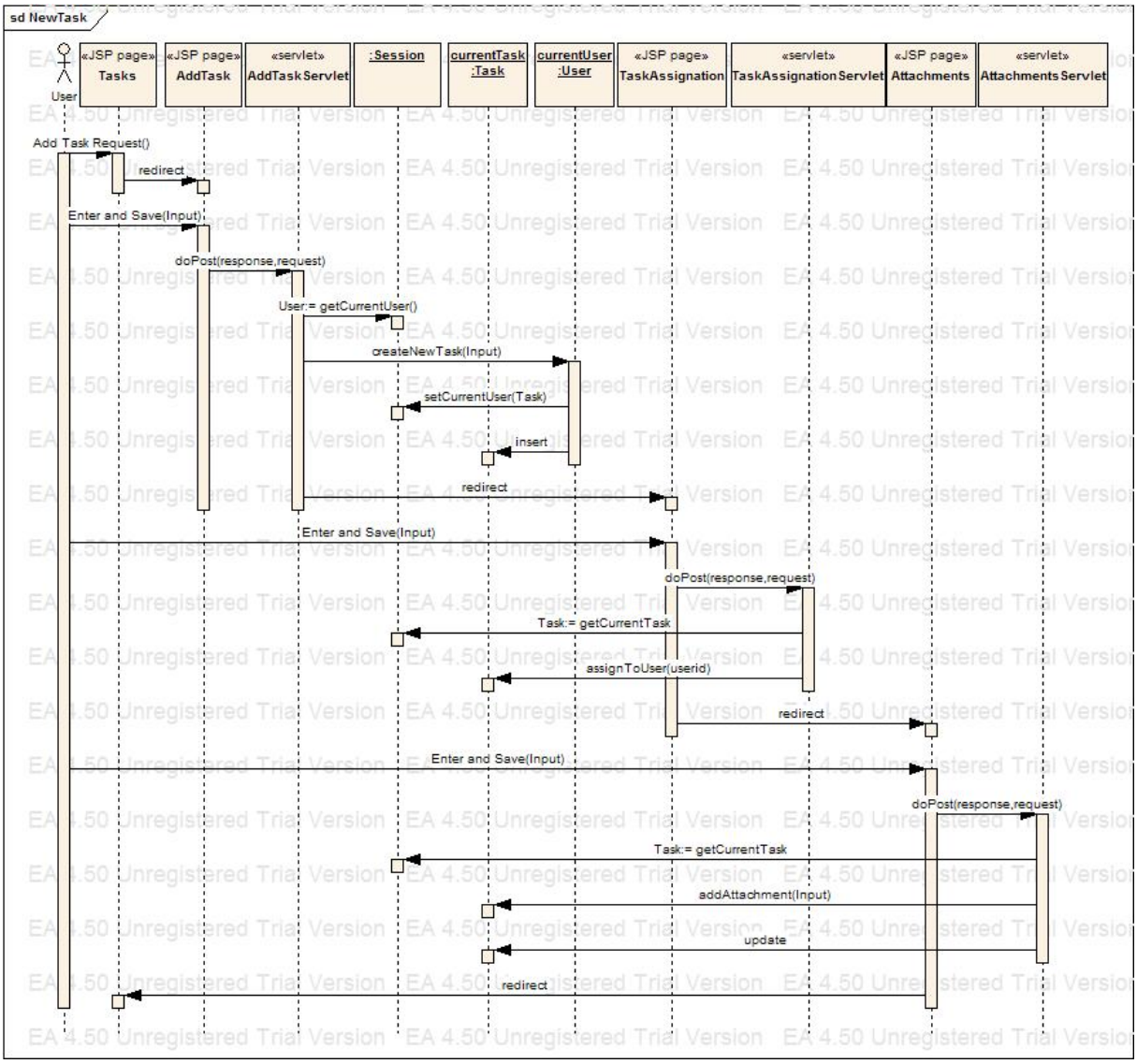


Figure 3: NewTask

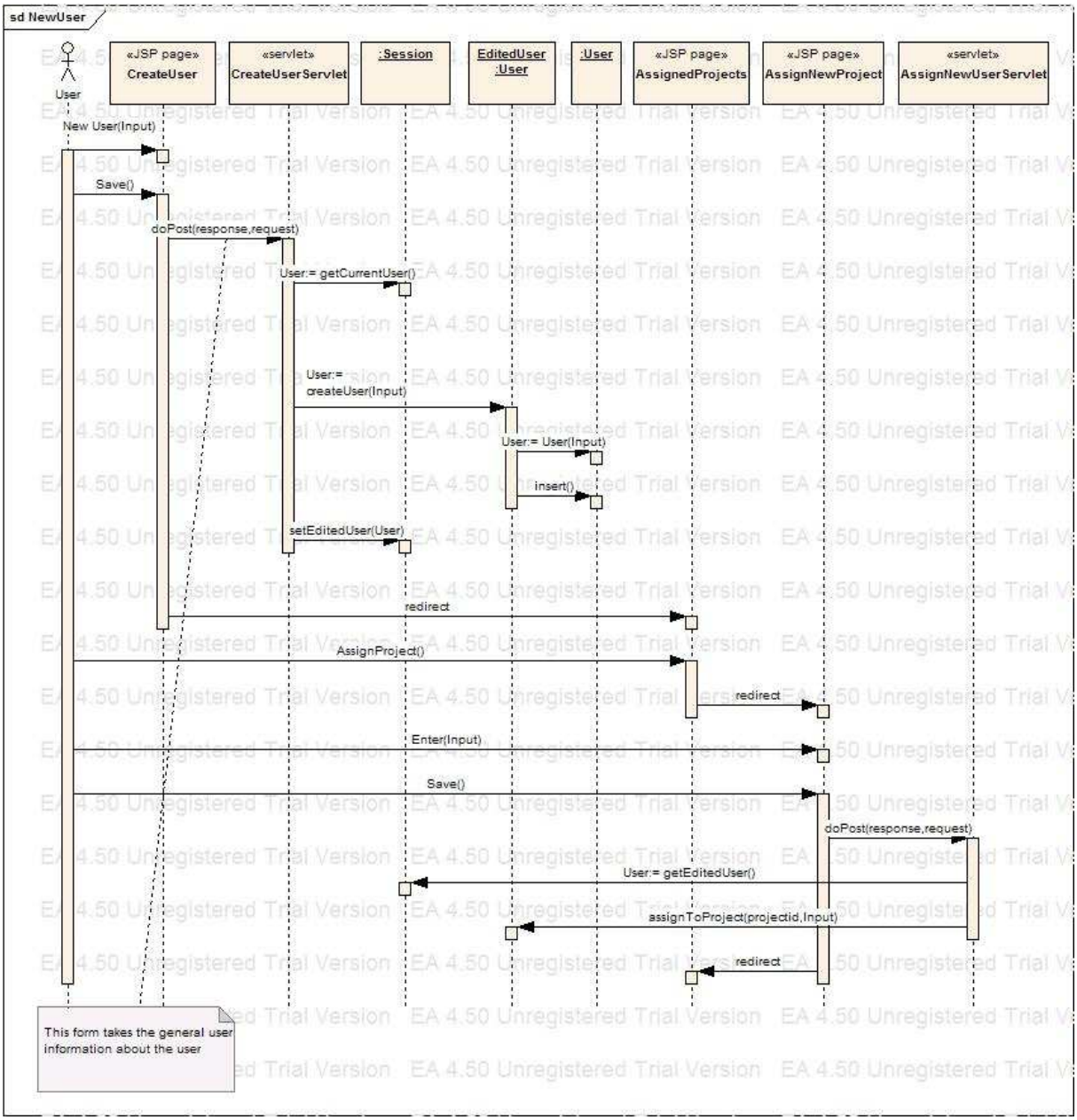


Figure 4 : NewUser

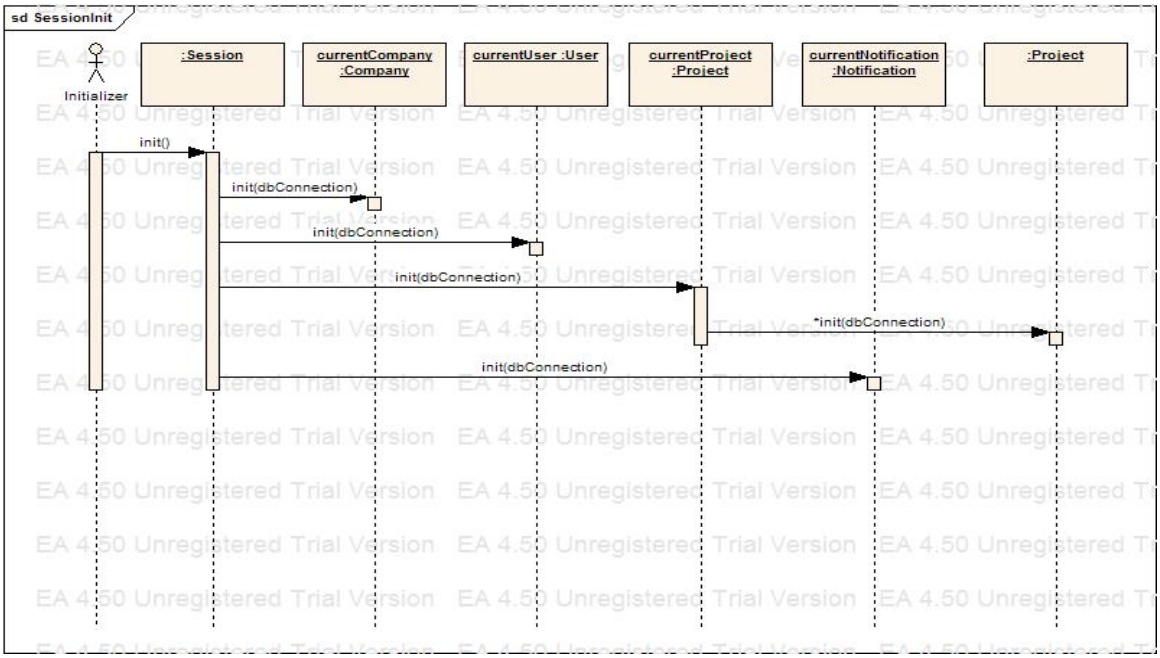


Figure 5 : SessionInit

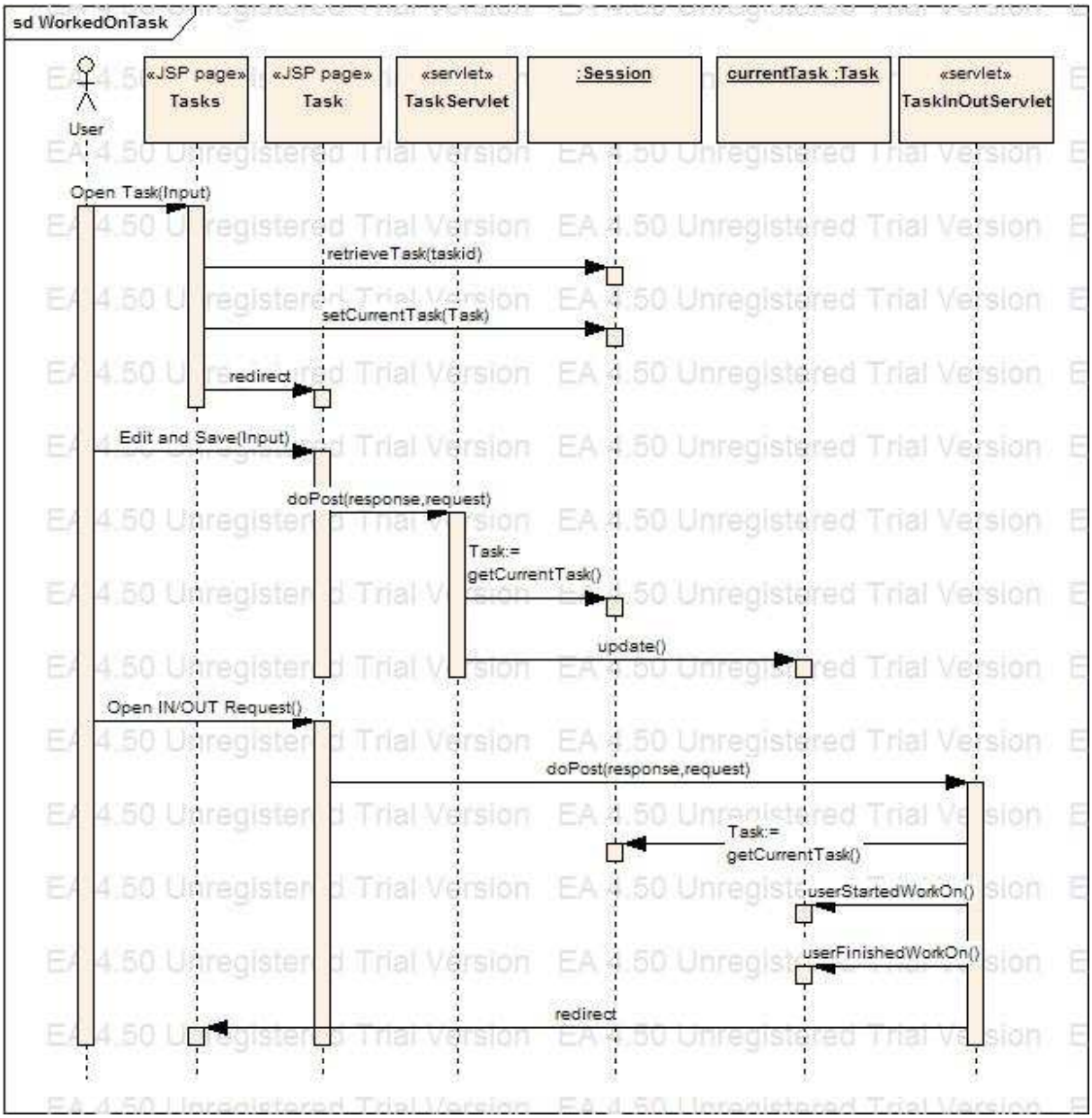


Figure 6 : WorkedOnTask

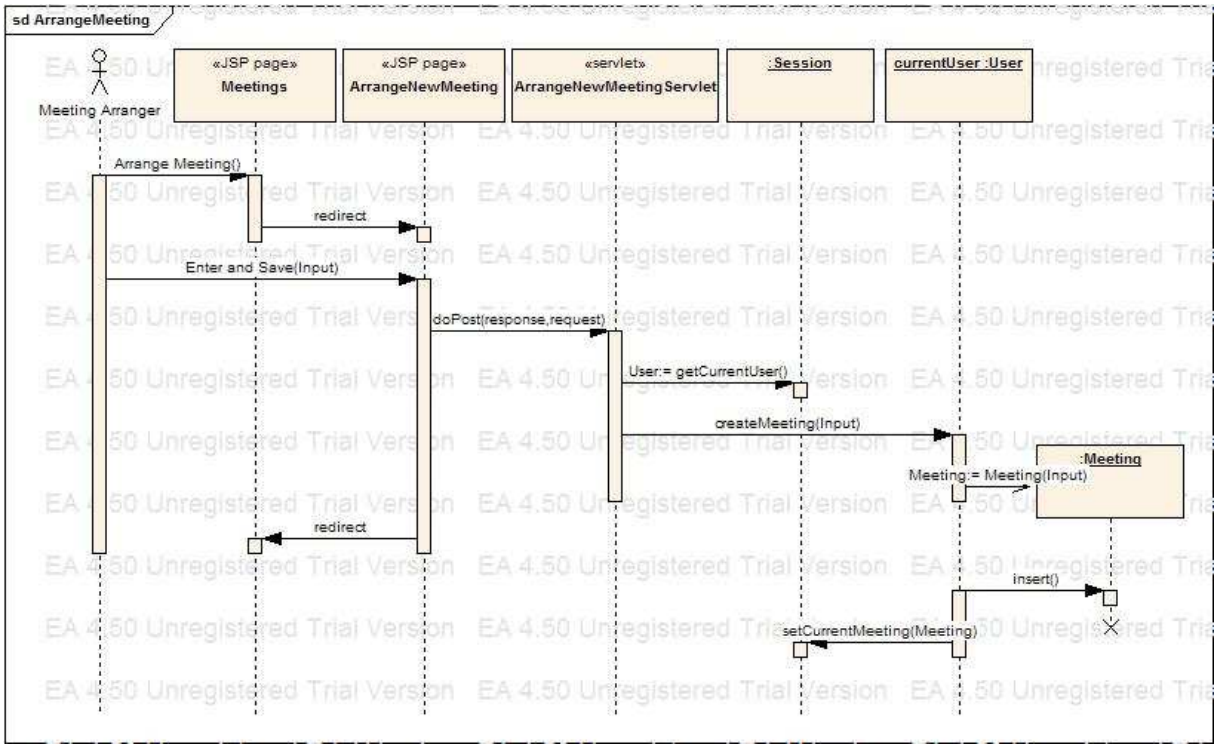


Figure 7 : ArrangeMeeting

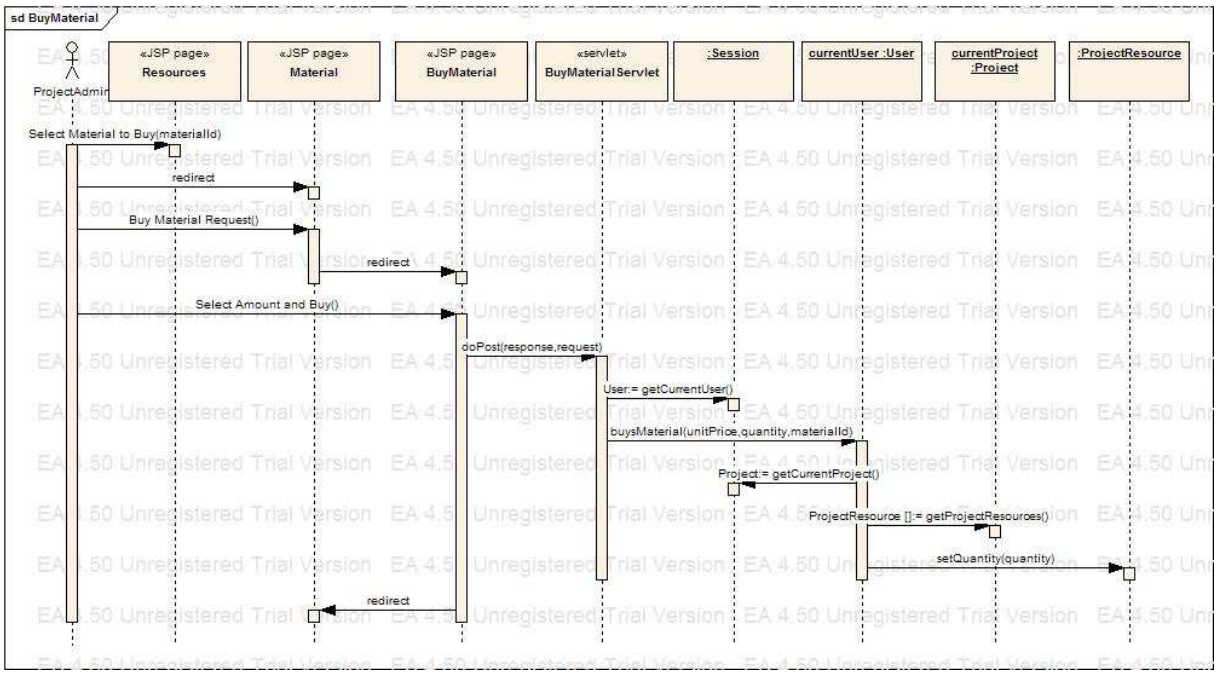


Figure 8 : BuyMaterial

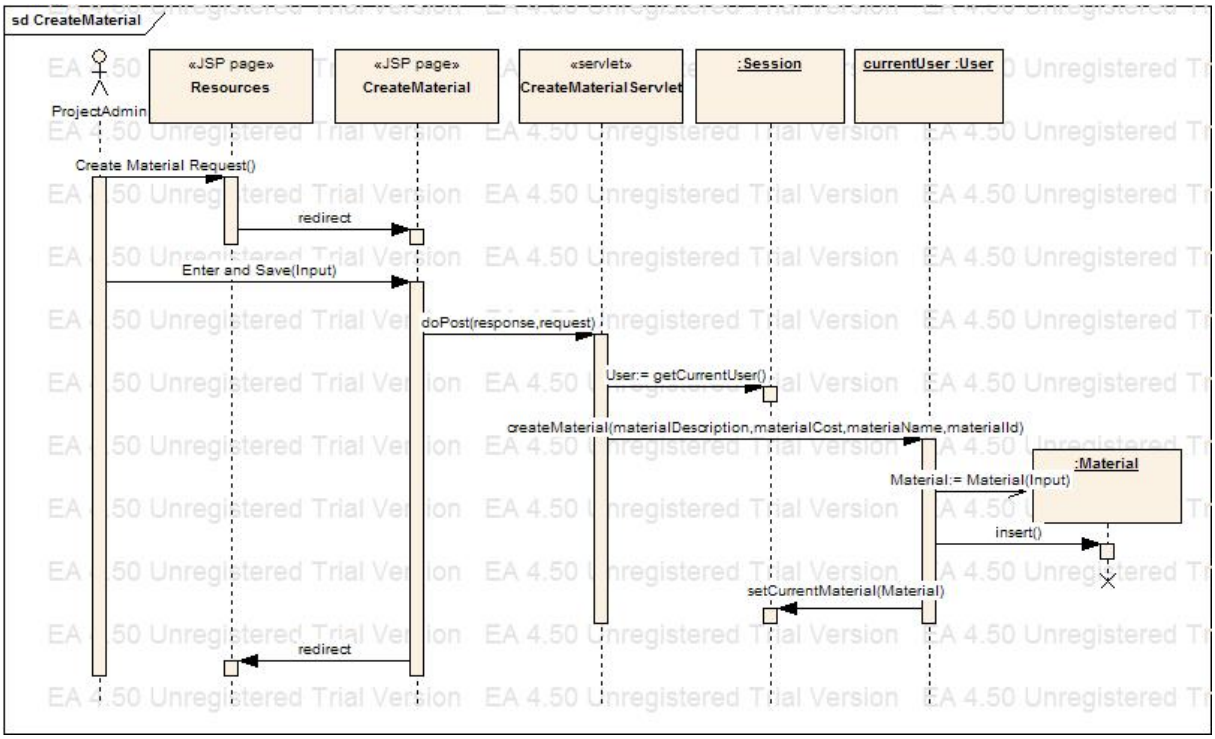


Figure 9 : CreateMaterial

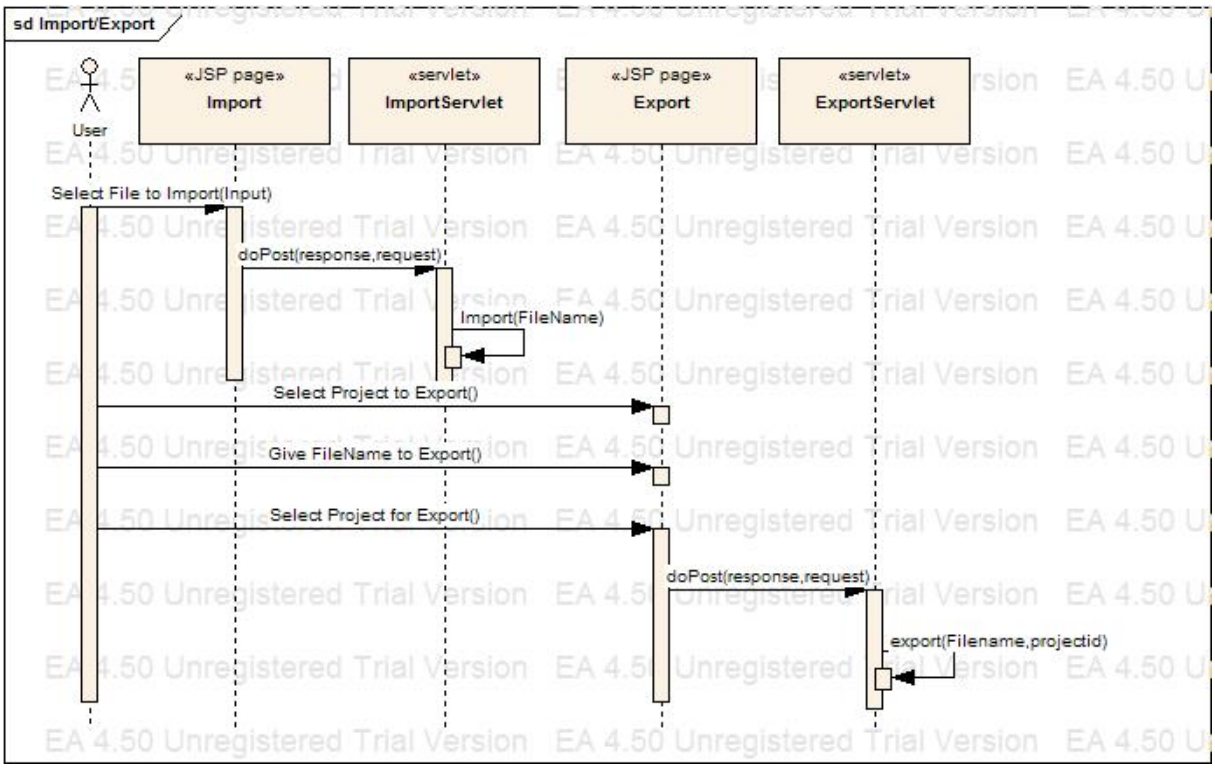


Figure 10 : Import/Export

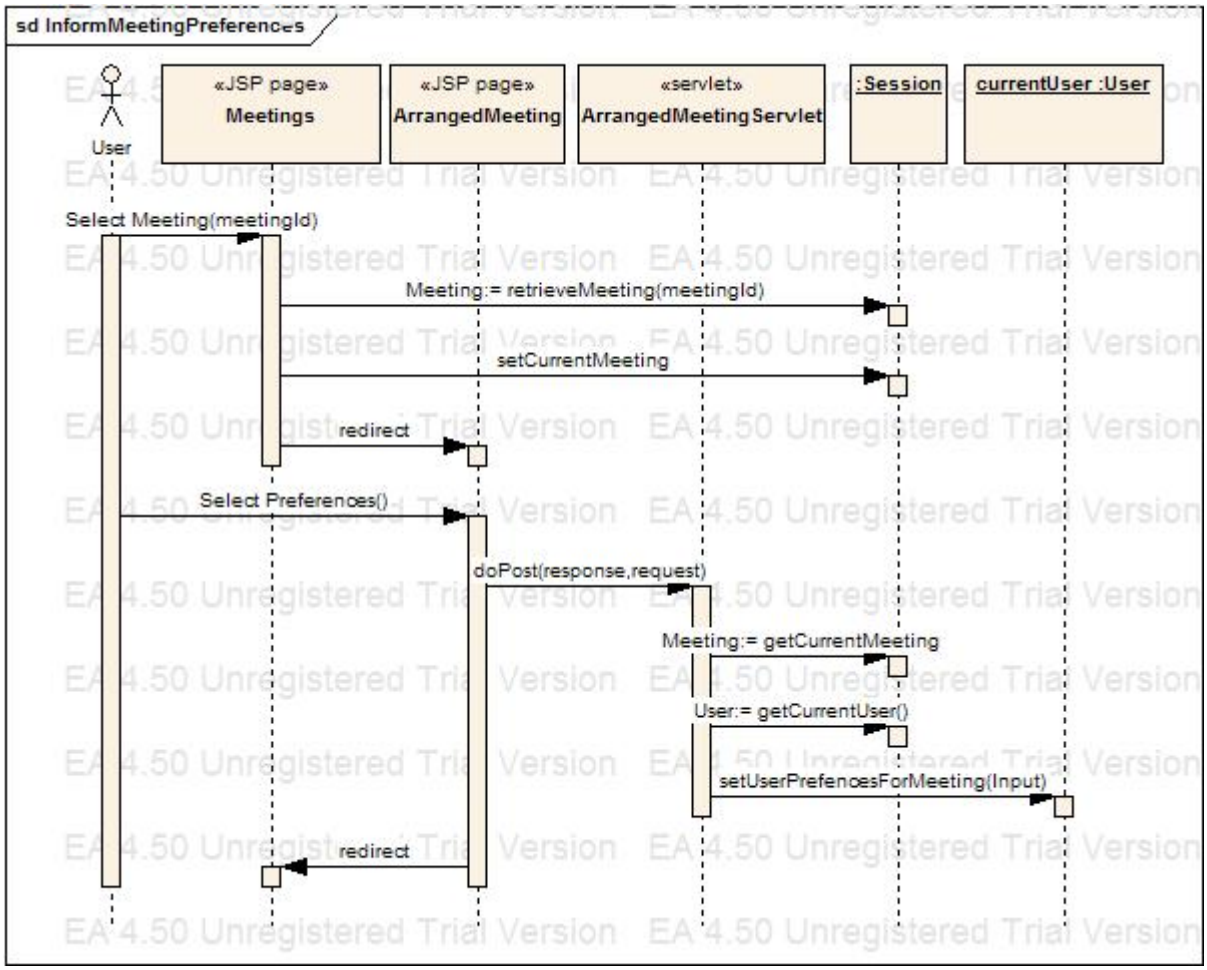


Figure 11 : InformMeetingPreferences

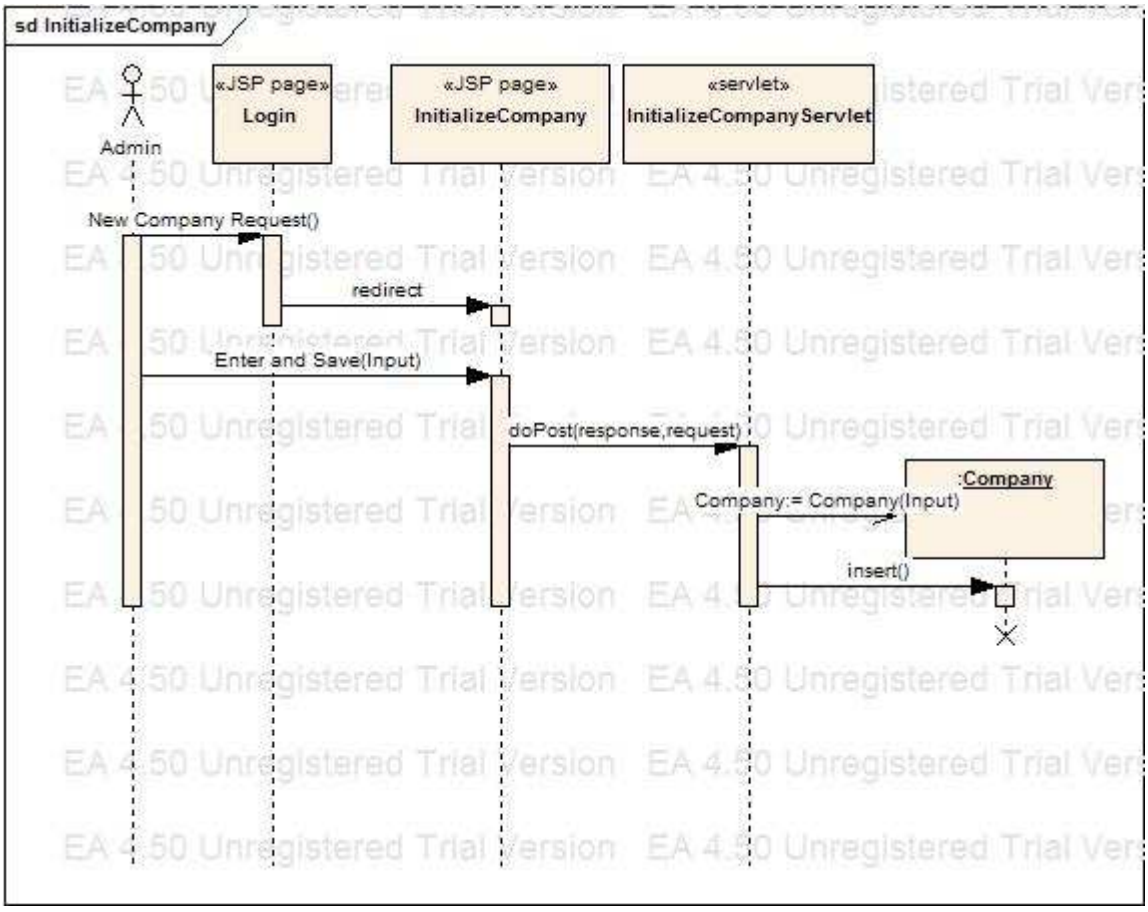


Figure 12 : InitializeCompany

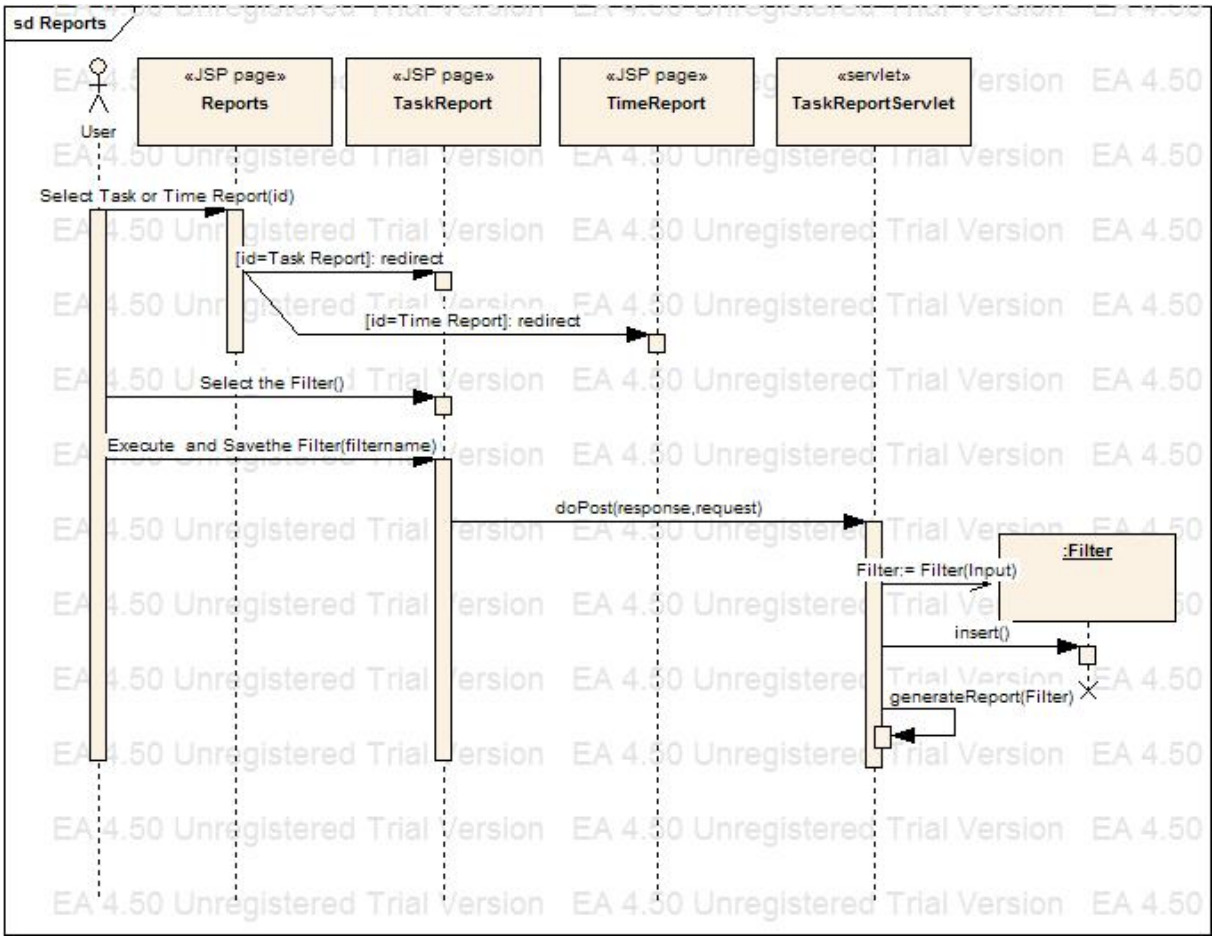


Figure 13 : Reports

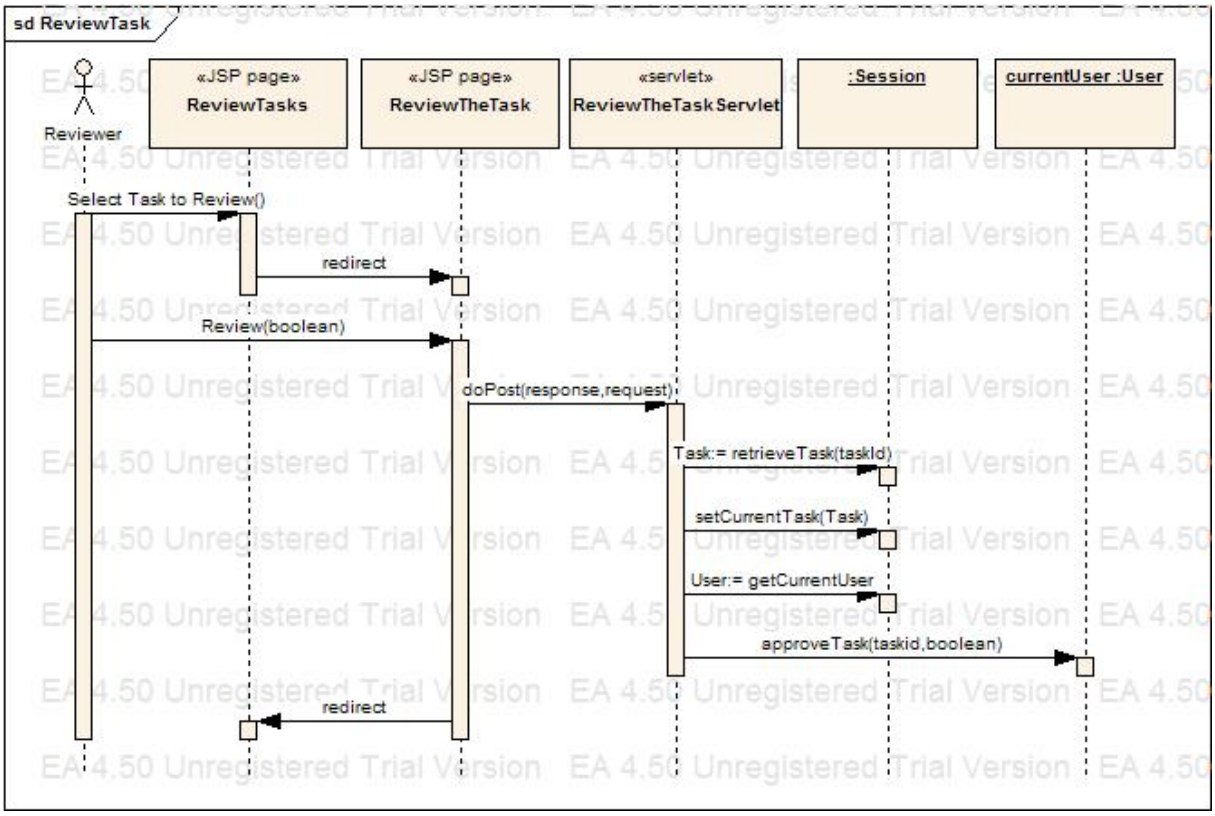


Figure 14: ReviewTask

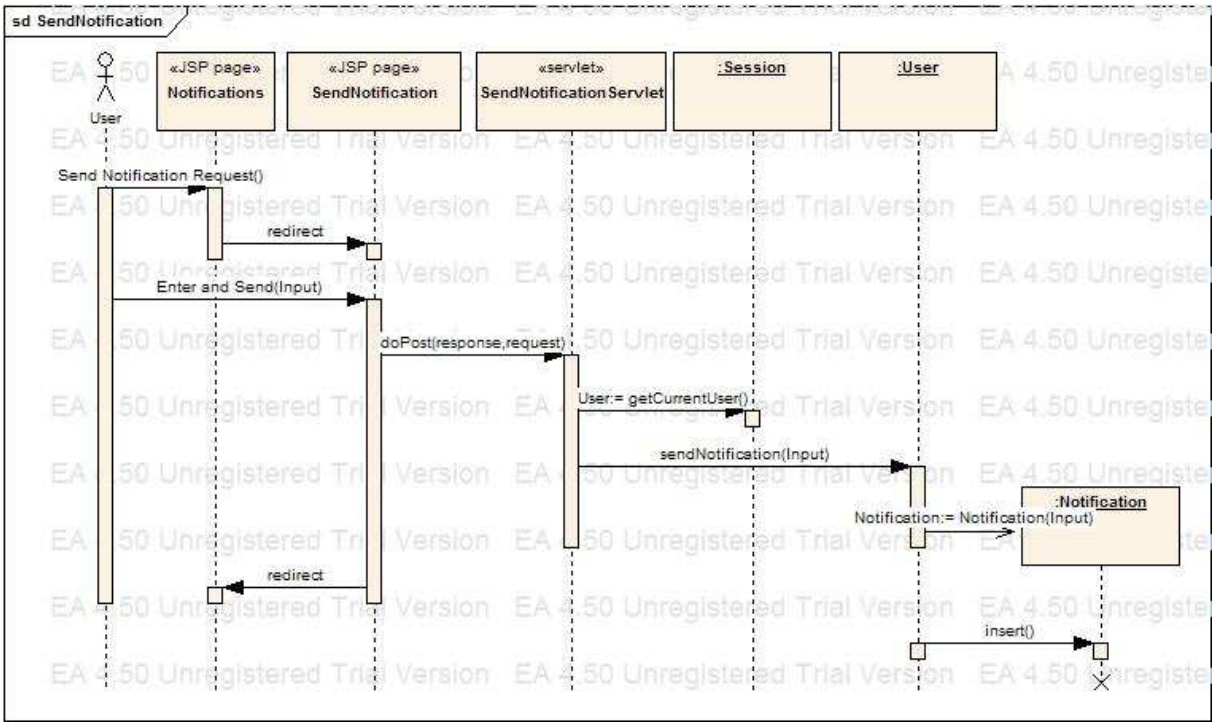


Figure 15: SendNotification

Login Messages

This diagram specifies the session initialization and login procedure in DProject for every user. User starts with the page 'Login.jsp' and after entering his 'company_name', 'user_id' and 'password', the page calls the doPost method of the 'LoginServlet'. Servlet calls the beforeLogin operation of the Initializer class. This operation constructs a session and creates connection for database by using the SQLConnection class. Then servlet calls the 'Initializer.login' operation of the 'Initializer' that checks the user login and password and returns whether user is authorized or not in which an error page is shown by the system. After that, authorization servlet calls the 'afterLogin' operation and this makes the necessary initialization for the Session variables. We show this initialization part at SessionInitialization diagram. Finally, the page is redirected to 'Main.jsp'.

NewProject Messages

This diagram shows the complete process of creating a new project. User can only create the project and leave the process but diagram shows the whole creation scneario for the user. Firstly, a new project is created using the general information about the project by the operation 'User.createNewProject()'. This operation creates a Project by calling the constructor of Project with the necessary input as argument and calls the 'insert()' operation to create the project in the database. After creating project, it sets the currentProject to this new project. Then 'EditProject.jsp' page is redirected which enables the user to edit general information in the created project. After that, user decides to assign the users to project which redirects the page to AssignedUsersForProject. This page shows the assigned users for this project which is empty since it is a new project. When 'Assign a User' request comes from user the page is redirected to 'AssignUserToProject.jsp'. After specifying the user and his/her access rights for the project, page calls doPost method of the associated servlet. This servlet takes the currentProject object from the session and calls its 'assignToUser'

operation which makes the neccesary addition to database. The third stage is the adding ‘task types’, ‘task priorities’, ‘task statuses’ to the project. In this diagram only ‘adding task type’ is shown. For these processes, associated servlets take the ‘currentProject’ from session and call the related operation of the ‘currentProject’ object.

NewTask Messages

This diagram shows the process of creating a new task for specified project. When user requests to add task to project the page redirects to the ‘AddTask.jsp’. Then user specifies the information about the task and submits to create a task. Then page call the servlets ‘doPost’ method which gets the currentUser object from session and calls the ‘createNewTask()’ method of this object. Then servlet sets the currentTask to this task and calls the ‘insert()’ method of the ‘currentTask’ object. After creating the task the assignation and adding attachment phases are done which are very similar to this phase.

NewUser Messages

Like above, this diagram shows the complete process of creating a user which also includes the assignation of the created user to some projects. The general information is used for creating the user by the operation ‘createUser’ of the ‘currentUser’ object. This operation creates the user object and calls its insert method to insert the user into database and sets the editedUser to this user. In the second phase ‘AssignNewUserServlet’ takes the ‘editeUser’ from session and calls the ‘assignToProject’ operation of this object.

SessionInit Messages

This diagram shows the intialization process of the DProject. ‘Session.init()’ calls its variables' ‘init()’ operations. The currentCompany, currentUser objects won’t change during the session. The currentProject object will show the the project that is open during the session. We also initialize the users' projects for later usage.

WorkedOnTask Messages

This diagram shows how the user can edit his work for a specified task. Editing the information and opening the In/Out processes shoud be treated as seperate processes but we show them together here one after the other. User selects one of his/her tasks to edit which in turn calls the Session.retrieveTask(taskid) operation and this operation returns the Task object. Then this page sets the currentTask to this task and redirects to ‘Task.jsp’. Edited information is passed to Servlet by doPost method and servlet takes the currentTask from the session and call ‘update()’ operation of the ‘currentTask’ object.

The other phase is opening or closing the In/Out option. When user requests this operation, the ‘TasksInOutServlet’ takes the currentTask object from session and calls userStartWork() or userFinishWork() operations. These operations make the necessary changes on database.

ArrangeMeeting Messages

In the Meetings page, if user requests to arrange a meeting, page is redirected to ‘ArrangeNewMeeting.jsp’. User specifies the necessary information and submits the meeting.

Then the page calls the doPost method of the related servlet. Servlet gets the ‘currentUser’ object from the session and calls the ‘createMeeting()’ operation of this object. This operation creates a Meeting object and calls the ‘insert()’ method of this object which builds the meeting in the database.

BuyMaterial Messages

This diagram shows the process of purchasing some quantity of specified material to project. In the Resources page user can see the project resources and defined materials. If he select a material page redirects to ‘Material.jsp’ which shows the materials properties. Then user request to purchase some quantity of this material type which redirects to page ‘BuyMaterail.jsp’. After specfiying amount of material that is bought page calls the servlets doPost method. Servlet gets the currentUser object from the session and calls the buysMaterial() method of this object. This method get the ‘ProjectResources’ of the ‘currentProject’ object and call the setQuantity() to set the new quantity to database and to object.

CreateMaterial Messages

This diagram shows the sequence of the processes for defining a new material for company. From the Resources page user can request to define new material which will redirect him to ‘CreateMaterial.jsp’. In this page user specifies the information about the material and submits to create the material which calls the servlet's doPost method. Servlet gets the currentUser object from the session and calls the createMaterial() method of this object. This method creates a Material object and calls its insert() method to create the material in database.

Import/Export Messages

This diagram shows the process of importing and exporting files for specified project. We show the processes as one after another but these processes are seperate. For the import phase, user write the file to import and submit this request. Then page calls the related servlet's doPost method which in turn calls its Import() method to import the information to database. Second phase includes exporting a project to a file. User selects a project and a filename and submits its request to export for the project. Then the servlet is called and it creates the file using the information in the database.

InformMeetingPreferences Messages

This diagram shows the process of specifying the date options for the potential meeting which are stated by the potential attendant of the meeting. User selects a meeting that he will participate and this redirects the page to ‘ArrangedMeeting.jsp’. This page calls the servlet's method after getting information from the user. Then servlet gets the currentUser object from the session and calls the setUserPreferencesForMeeting() method to save the information into database.

InitializeCompany Messages

This diagram shows the creation of new company process in Dproject. This process is used only once for the company by the admin of it. In the login page, if the user requests a Create Company operation, page redirects to the 'InitializeCompany.jsp'. User specifies the information for the company and page calls the doPost method of the servlet. Servlet creates an object of Company and calls insert() operation of this object which creates a new database (with the name of the company) in DBMS.

Reports Messages

This diagram only shows the process of creating a new task report which is very similar to time reports. User selects the filter and submits his request to generate report. Servlet's doPost method is called and this method generates the report from the information in the database. If the user also wants to save this filter by giving a name to it, servlet creates a 'Filter' object and calls its 'insert()' method to create the Filter in the database.

ReviewTask Messages

This diagram shows the process of reviewing the works of users on tasks and rejecting or accepting them. Page calls the servlet's doPost method after reviewer selects the task to review. Servlet calls the session's retrieveTask() method which returns a Task object to take the related Task from database. Then it takes the currentUser object from the session and calls its approveTask() method to save the decision of the reviewer into the database.

SendNotification Messages

This diagram shows the process of sending a notification to another user. Servlet gets the 'currentUser' object from session and calls the sendNotification() method for creating the notification in the database. This method create the notification and calls its 'insert()' method.

2.4 ACTIVITY DIAGRAM

Activity diagram can be found in the Appendix since it does not fit on an ordinary A4 page.

The explanation of the activity diagram is below:

The activities are started by displaying the login screen.

- The user will fill in with his/her id and password, after the user enters his/her id and password these information will be compared with the one that will be retrieved from the database. If the password turns out to be valid then the main screen will be displayed, if the password entered by the user turns out to be invalid then we return to the initial screen.

First of all, some information isn't shown in this activity diagram for the sake of simplicity. The omission is; after main screen is displayed, the user can select display main screen link or logout link any time s/he wants.

In the main screen the user has the following options:

- If the user selects 'Send Notification' link then the notification form will be displayed. And the user will fill in. Then the user will click send button and the notification will be sent to the specified users. (As I mentioned in the beginning, in any of the stages, the user can select the 'Main Screen' link so that the notification process will be canceled and system will return to the Main screen. Also the user can select 'Logout' in any stage so that the system will terminate the session. These possible activities won't be specified in any other option.)
- If the user selects 'Create new filter' link then the new filter creation form will be displayed. And the user will fill in. Then the user will click save button and the filter will be stored in database.
- If the user selects 'Edit Preferences' link then the 'Preferences Screen' will be displayed. And the user will edit his/her preferences. Then the user will click save button and the preferences will be stored in database.
- If the user selects 'Generate Statistics Link' link, then s/he will select the filters to be applied. After that the statistics will be generated and displayed depending on the filters selected by the user.
- If the user selects 'Help' link then the 'Help Screen' will be displayed. And the user will enter the topic that s/he wants to get information about. Then the system will display the information about the topic if there is any record about that topic in the database.
- If the user selects 'Forum' link then the 'Forum Screen' will be displayed. After that user can either read a message or write a new message. If s/he wants to read a message, s/he will simply select the thread and the message will be displayed. If the user wants to write a new message, s/he will select the thread under which s/he wants to write new message and then will write the body of the message and click the send button after that the message will be stored in database.
- If the user selects 'Reports' link then the 'Reports Screen' will be displayed. After that user has two other options :
    - If the user selects 'Import Report' link, the report will be fetched from user's computer and will be viewed.
    - If the user selects 'Generate Report' link, s/he will specify the type of the report and the filters to be applied and after that system will generate the report based on this selections and display it. In this stage user can select to export the report into his/her computer or go back to the main reports screen.
- If the user selects 'Administration' link then the access rights of the user will be checked. If the user doesn't have the necessary rights, s/he won't be able to do any administration operation and 'Main Screen' will be displayed. Else if the user has admin rights 'Administration Screen' will be displayed. And the user can either select to create a new user account to the system or create a new company account. In either case, admin will enter the necessary information, then select the save button and the records will be saved in database.
- If the user selects 'Arrange Meeting' link then the access rights of the user will be checked. If the user doesn't have the necessary rights, s/he won't be able to

arrange any meeting and ‘Main Screen’ will be displayed. Else if the user has enough rights user will specify the potential dates and the attendants of the meeting. After the potential attendants of the meeting stated their choices, user will fix the details of the meeting depending on these choices. And then user will select the save button and the records will be saved in database.

- If the user selects ‘Projects’ link then the ‘Projects Main Screen’ will be displayed. In this stage, user has following options:
    - User can view the details of a project by selecting ‘View Project’ link. After this selection the system will display the project details and now user has another two options:
        - If the user selects ‘Export Project’ link, the project will be saved in a file into the user’s computer.
        - If the user selects ‘Task creation’ link, then the access rights of the user will be checked. If the user doesn’t have the enough rights, s/he won’t be able to create any task and ‘Projects Main Screen’ will be displayed. Else if the user has enough rights, s/he will enter the necessary information to create a new task (task name, assigned users, etc) and hit the save button. After that new task will be saved in the database.
    - User can select ‘Create New Project’ link. Of course, first of all the access rights of the user are fetched from the database to see whether s/he has the necessary access rights to create a project and if not user won’t be able to create the project and ‘Projects Screen’ will be displayed. Else if the user has enough access rights, s/he will specify the creation type (from template or from scratch). To create project from template, the project file is fetched forum user’s computer and for the other case a blank project is created and the user enters the necessary information about the new project (name, tasks, assigned users, etc). After that system saves the project in database.
- If the user selects ‘Tasks’ link then the ‘Tasks Main Screen’ will be displayed. In this stage, user has following options:
        - User can select ‘View Tasks’ link. In this case, the access rights of the user will be checked. If the user doesn’t have the enough rights, s/he won’t be able to view any task and ‘Tasks Main Screen’ will be displayed. Else if the user has enough rights, task will be displayed. Now, user can either choose to send the finished tasks to the reviewer or work on a task. If the user selects to work on a task, it will be checked that whether the task is assigned to the user. If it’s not, the user won’t be able to open in/out and ‘Main Tasks Screen’ will be displayed. Else in/out will be opened, the user will work on the task and in/out will be closed.
        - User can select ‘Review Tasks’ link. In this case, the user will select from the finished tasks which are sent for reviewing and if the reviewer of the task is assigned to be the user, s/he will be able to review the task and either accept or reject the work done. In either case, a notification is sent to the user who did the task and if reviewer rejected the work done, task will be marked as undone and the assigned user will have to do it again.
- If the user selects ‘Logout’ in any stage then the system terminates the session.

2.5 STATE DIAGRAM

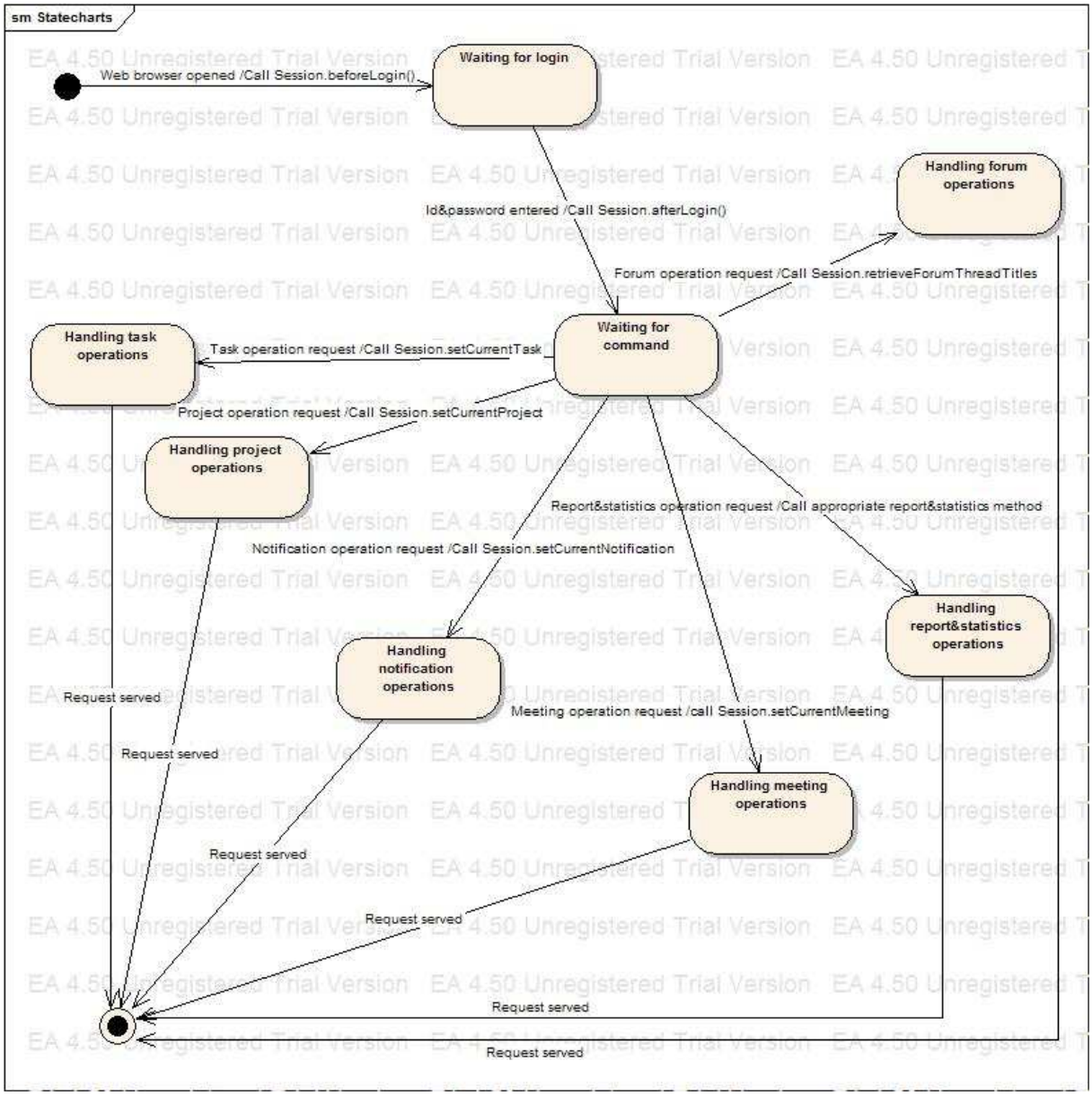


Figure 1: Session class state diagram

When the user opens the address of the project management tool in his browser, a session is created and its beginLogin method is called, triggering a translation from the initial state to the Waiting for login state. When the user enters his id and password correctly, the new state is Waiting for command state, in which requests of the user are being waited to be handled.

There are a number of possible translations from the Waiting for command state. The state changes to:

- Handling task operations state, if the user makes a task operation request
- Handling project operations state, if the user makes a project operation request
- Handling notification operations state, if the user makes a notification operation request

- Handling meeting operations state, if the user makes a meeting operation request
- Handling report&statistics operations state, if the user makes a report&statistics operation request
- Handling forum operations state, if the user makes a forum operation request

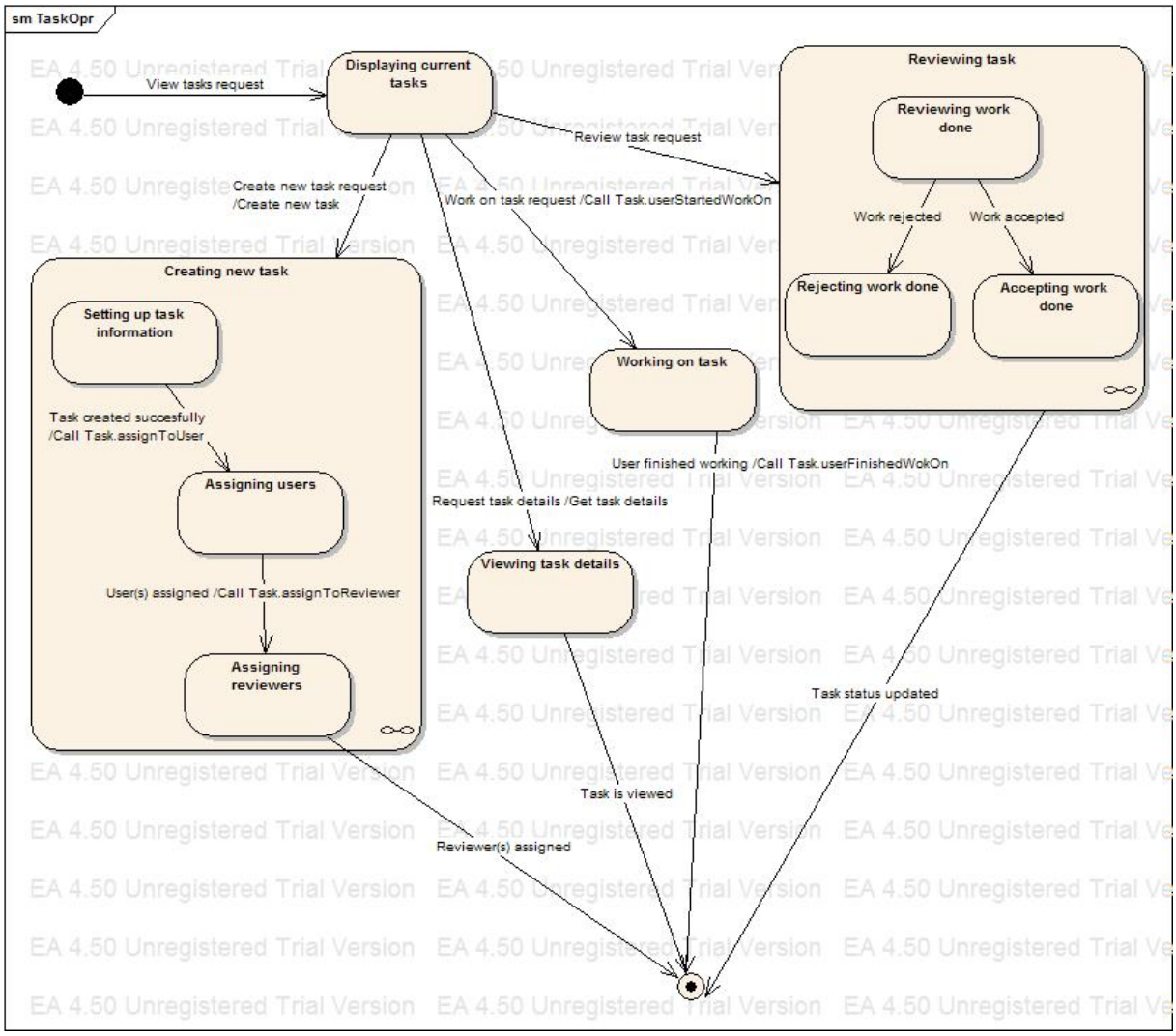


Figure 2: Task class state diagram

When the user makes a view tasks request, a transition occurs from the initial state to the Displaying current tasks state.

When the user wants to create a new task, a transition occurs from the Displaying currents tasks state to the Creating new task state. This state has three sub-states, namely the Setting up task information state, Assigning users state, and the Assigning reviewers state. In the Setting up task information sub-state, the necessary information for the creation of a task is entered. When this necessary information is provided, a translation occurs to the Assigning users state, in which the task is assigned to users. When the assignation is done properly, a transition occurs to the Assigning reviewers state, in which reviewers are assigned to the task.

When the user wants to review a task, a transition occurs from the Displaying current tasks state to the Reviewing task state. This state has three sub-states, namely the Reviewing work done state, Rejecting work done state, and the Accepting work done state. The user

reviews the task in the Reviewing work done state. Depending on the decision of the reviewer, a transition occur either to the Rejecting work done state (the work done is rejected), or to the Accepting work done state (the work done is accepted).

When the user wants work on a task, a work on task request causes a transition from the Displaying current tasks state to the Working on task state.

When the user wants to view the details of a task, requesting the details of the task causes a transition from the Displaying current tasks state to the Viewing task details state.

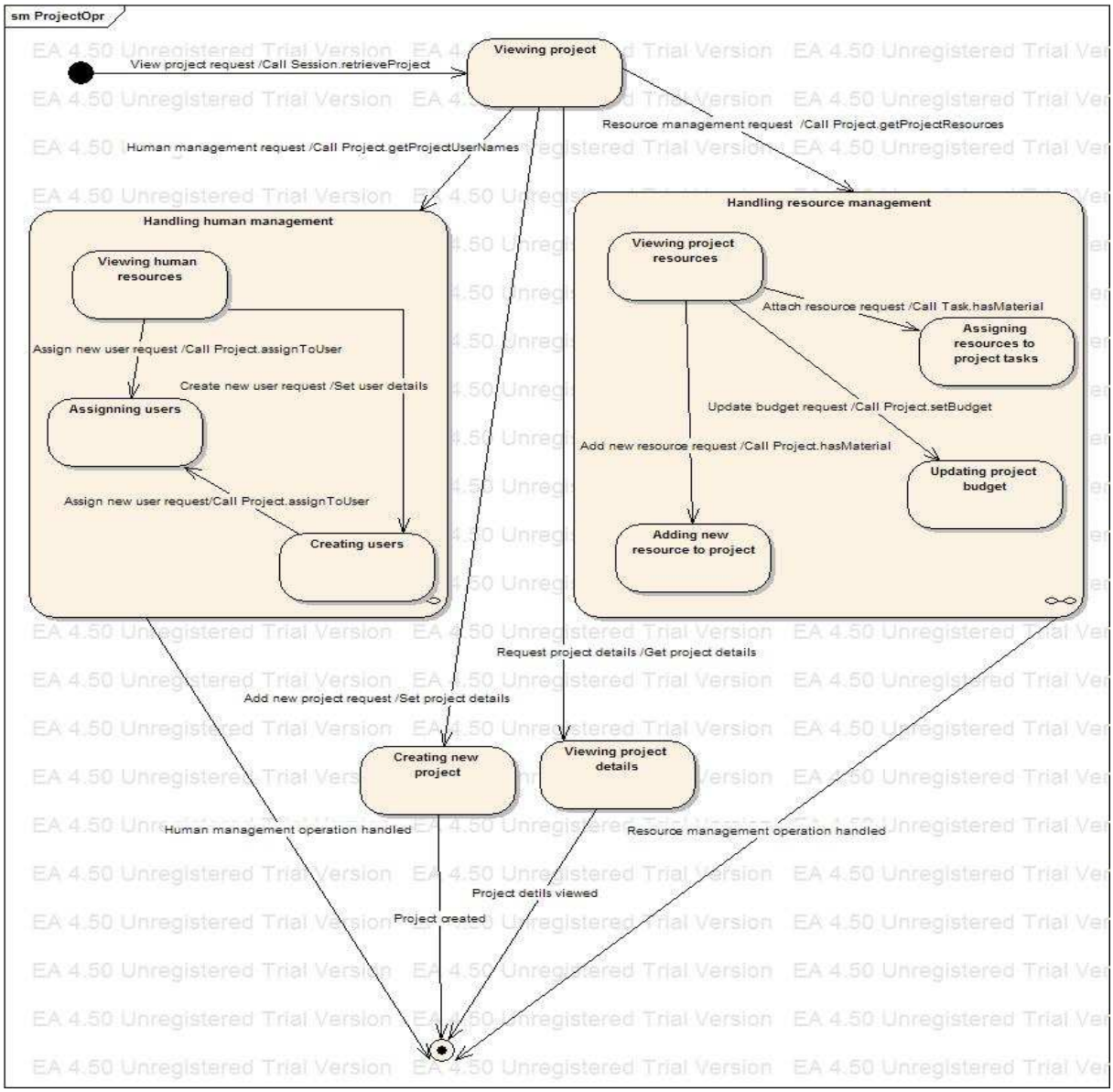


Figure 3: Project class state diagram

When the user makes a view project request, a transition occurs from the initial state to the Viewing project state.

When the user wants to perform human management operations, a transition occurs from the Viewing project state to the Handling human management state. This state has three sub-states, namely the Viewing human resources state, Assigning users state, and the Creating

users state. The user views that human resources of the project in the Viewing human resources state. If the user wants to assign a user to the project, a transition occurs to the Assigning user state. If the user wants to create a new user, a transition occurs to the Creating users state. If the user wants to assign the newly created users to the project, a transition occurs to the Assigning users state.

When the user wants to perform Resource management operations, a transition occurs from the Viewing project state to the Handling resource management state. This state has four sub-states, namely the Viewing project resources state, Assigning resources to project tasks state, Updating project budget state, and the Adding new resource to project state. The user can view the project resources in the Viewing project resources state. If the user wants to add a new resource to the project, a transition occurs to the Adding new resource to project state. If the user wants to update the budget of the project, a transition occurs to the Updating project budget state. If the user wants to assign resources to any of the project tasks, a transition occurs to the Assigning resources to project state.

When the user wants to create a new project, a transition occurs from the Viewing project state to the Creating new project state.

When the user wants to view the details of a project, a transition occurs from the Viewing project state to the Viewing project details state.

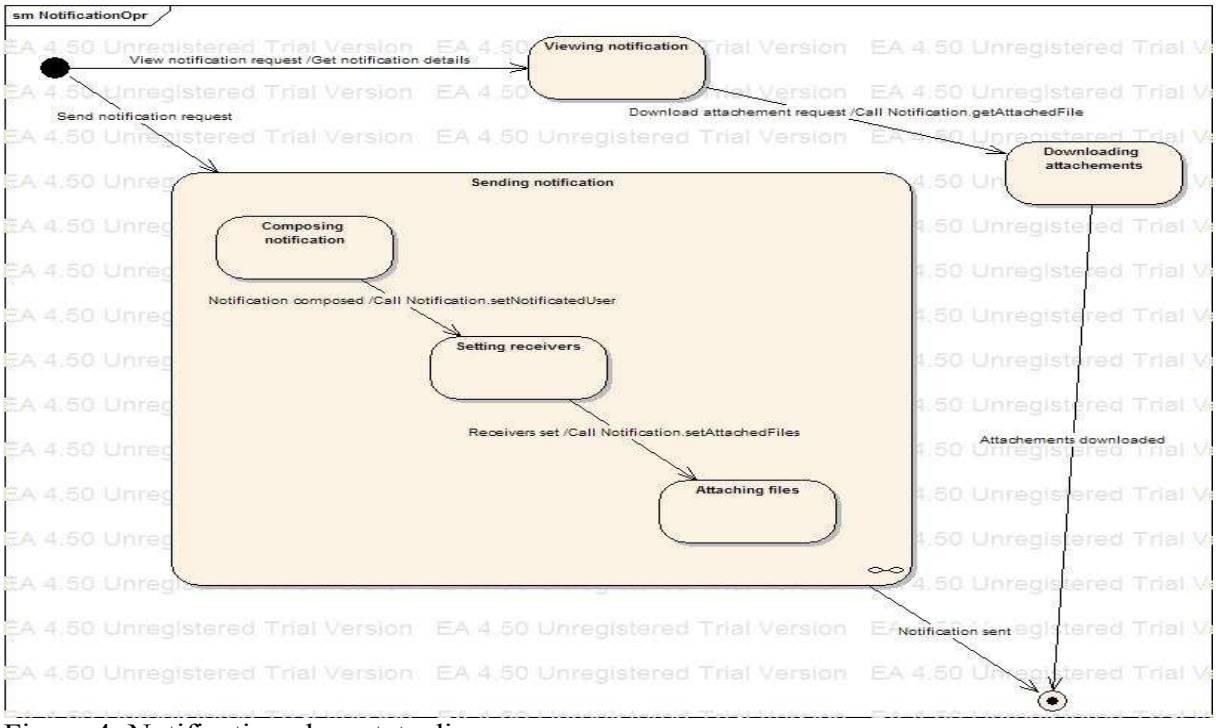


Figure 4: Notification class state diagram

When the user makes a send notification request, the transition occurs from the initial state to the Sending notification state. This state has three sub-states, namely the Composing notification state, Setting receivers state, and the Attaching files state. The user composes the notification in the Composing notification state. When the notification is composed, a transition occurs to the Setting receivers state, in which the receivers of the notification are set. When the receivers are set successfully, a transition occurs to the Attaching files state, in

which the files (if exists) of the notification are attached to it. After this state, the notification is ready to be sent.

When the user wants to view his notifications, a transition occurs from the initial state to the viewing notification state. If the user makes a download attachments request at this state, a transition occurs to the Downloading attachments state.

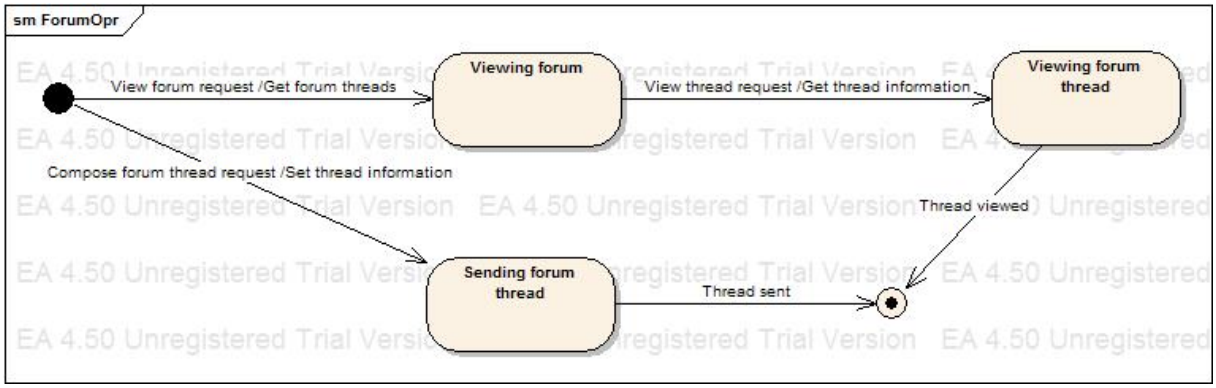


Figure 5: Forum class state diagram

When the user wants to view the forum a view forum request causes a transition from the initial state to the Viewing forum state. If the user wants to view a thread in the forum, a view thread request causes a transition to the Viewing forum thread state.

When the user wants to compose a forum thread, a compose forum thread request causes a transition from the initial state to the Sending forum thread state.

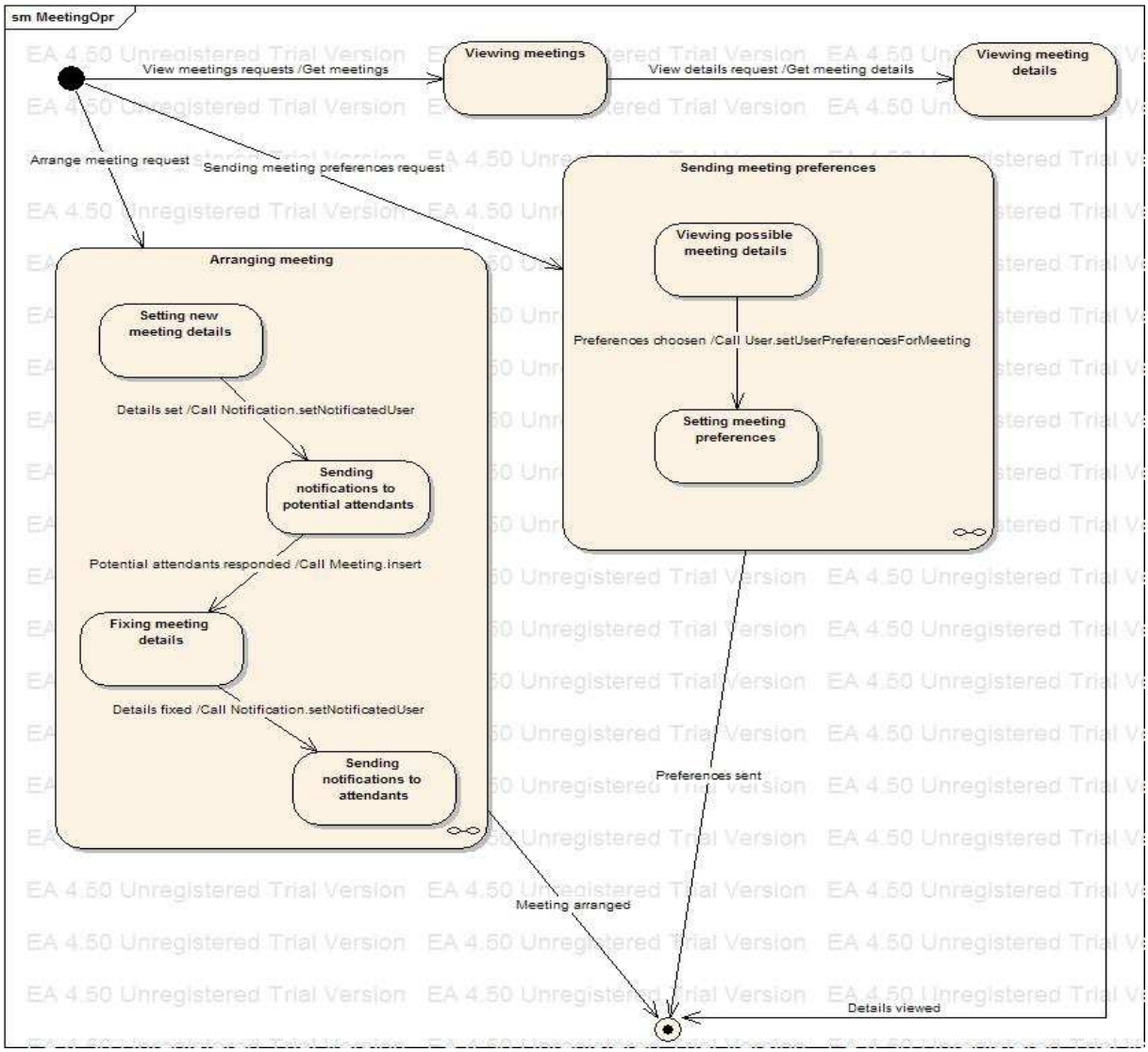


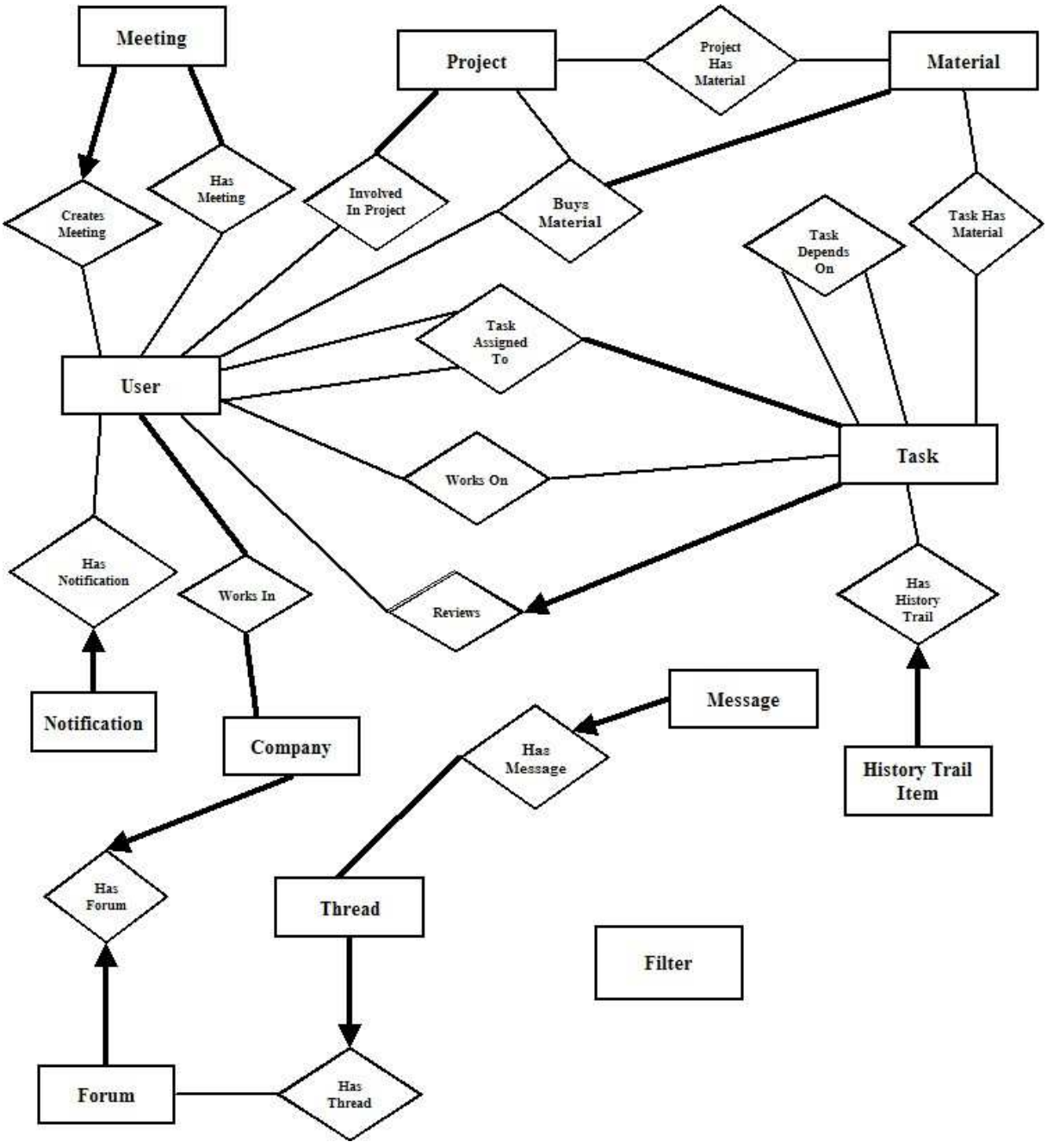
Figure 6: Meeting class state diagram

When the user wants to arrange a meeting, a transition occurs from the initial state to the Arranging meeting state. This state has four sub-states, namely the Setting new meeting details state, Sending notifications to potential attendants state, Fixing meeting details state, and Sending notifications to attendants state. In the Setting new meeting details state, the user specifies options for meeting details (e.g. meeting place, date). When these are set, a transition occurs to the Sending notifications to potential attendants state, in which the potential attendants are notified of the meeting options. When the potential attendants notify the arranger of their preferences, a transition occurs to the Fixing meeting details state, in which all the details of the meeting are fixed. Then the exact details of the meeting are sent to the attendants in the Sending notifications to attendants state.

When the user is a potential attendant of a meeting and wants notify the arranger of his choices, a transition occurs to the Setting meeting preferences state. This state has two sub-states, namely the Viewing possible meeting details state and the Setting meeting preferences state. The user views the possible meeting options in the Viewing possible meeting details state. When he wants to set his preferences for that meeting, a transition occurs to the Setting meting preferences state.

2.6 DATABASE DESIGN

2.6.1 ER DIAGRAM



2.6.2 DATABASE TABLES

USER

userid	PRIMARY KEY
password	
date_created	
last_visit_time	
first_name	

last_name
phone
email
birth_date
gender
speciality
photo
address
global_access_right
can_add_project
user_directory
email_notification_NewTask
num_of_tasks_per_page
num_of_months_per_page
num_of_weeks_per_page
payment_policy
payment_amount
user_group_id

COMPANY
company_id PRIMARY KEY
company_name
company_address
contact_info
logo
email_server
webpage
timezone_format
forum_id

TASK
task_id PRIMARY KEY
task_name
task_description
start_date
due_date
finish_date
priority_id
type_id
project_id
status_id
percent_done
reviewer_id
group_id
attached_file1
attached_file2
attached_file3
attached_file4
actual_hours
last_update

date_created
last_reviewed_percent_done int

HISTORY_TRAIL_ITEM
history_trail_id PRIMARY KEY
task_id
user_id
modification_type
old_value
new_value

MATERIAL
material_id PRIMARY KEY
material_name
material_cost
material_description
created_date
creator_user_id

PROJECT
project_id PRIMARY KEY
project_name
project_description
project_creator
create_date
start_date
finish_date
due_date
budget
contact_name
contact_phone
contact_email
project_type_id

FILTER
filter_id PRIMARY KEY
selected_user_id
selected_project_id
selected_paymentpolicy_id
selected_salary_comparison
selected_salary_quantity
selected_age_comparison
selected_age
selected_gender
selected_global_profil
selected_time_entry_mod
selected_start_date_comp
selected_finish_date_comp
selected_finish_date
selected_due_date_comp

selected_due_date
selected_priority_id
selected_type_id
selected_status_id
selected_percent_done_comp
selected_percent_done_quant
selected_reviewer_id
selected_group_id
selected_actualhours_comparison
selected_actualhours
selected_projectstartdate_comp
selected_projectstartdate
selected_projectfinishdate_comp
selected_projectfinishdate
selected_projectduedate_comp
selected_projectduedate
selected_projectmanager_id

FILTER_MEETING

filter_id	PRIMARY KEY
selected_creator_id	
selected_final_meeting_date_comp	
selected_final_meeting_date	

MEETING

meeting_id	PRIMARY KEY
final_meeting_date	
date_option1	
date_option2	
date_option3	
date_option4	
date_option5	
creator_userid	FOREIGN KEY(user)
create_date	
attachment1	
attachment2	
attachment3	
last_reply_date	

NOTIFICATION

notification_id	PRIMARY KEY
notificated_user	FOREIGN KEY(user:user_id)
notification_type	
owner_of_action	FOREIGN KEY(user:user_id)
date_of_action	
attached_file1	
attached_file2	
attached_file3	

Task_Depends_On

task_id1	PRIMARY KEY	
task_id2	PRIMARY KEY	
dependency_type		

User_Has_Meeting

user_id	PRIMARY KEY	FOREIGN KEY
meeting_id	PRIMARY KEY	FOREIGN KEY
user_selected_option1		
user_selected_option2		
user_selected_option3		
user_selected_option4		
user_selected_option5		

Task_Needs_Material

task_id	PRIMARY KEY	FOREIGN KEY
material_id	PRIMARY KEY	FOREIGN KEY
quantity		

Project_Has_Material

project_id	PRIMARY KEY	FOREIGN KEY
material_id	PRIMARY KEY	FOREIGN KEY
quantity		

User_Buys_Material

purchaser_id	PRIMARY KEY	FOREIGN KEY
material_id	PRIMARY KEY	FOREIGN KEY
project_id	PRIMARY KEY	
quantity		
unit_price		
date	PRIMARY KEY	

Task_Assigned_To

user_id	PRIMARY KEY
task_id	PRIMARY KEY
date	
assigner_id	

Works_On

user_id	PRIMARY KEY
task_id	PRIMARY KEY
start_date	PRIMARY KEY
finish_date	
is_approved	

Has_Access_Right

user_id	PRIMARY KEY
project_id	PRIMARY KEY
is_project_manager	
can_approve_time	

can_open_project
can_arrange_meeting
level

PRIORITY_TABLE
priority_id PRIMARY KEY
priority_name

TASK_TYPE_TABLE
task_type_id PRIMARY KEY
task_type

PRIORITY_TABLE
priority_id PRIMARY KEY
priority_name

STATUS_TABLE
status_id PRIMARY KEY
status_name
status_image

GROUP_TABLE
group_id PRIMARY KEY
group_name
group_logo

PROJECT_TYPE_TABLE
prpject_type_id PRIMARY KEY
project_type

FORUM
threadid PRIMARY KEY
subject
creator
date_created

MESSAGES
messageid PRIMARY KEY
threadid PRIMARY KEY
message
title
message_owner
date

3. PROJECT SCHEDULE

Project schedule is found in the Appendix since it does not fit in an ordinary A4 page.