SCHULICH
School of Engineering

UNIVERSITY OF CALGARY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

SENG 696 Agent-Based Software Engineering

Software Acquisition Request System

Assignment 2 – Detailed Design Document

Group # 6

Desiree Leal - 30063471

(Greg) Amar Gadala - 10177584

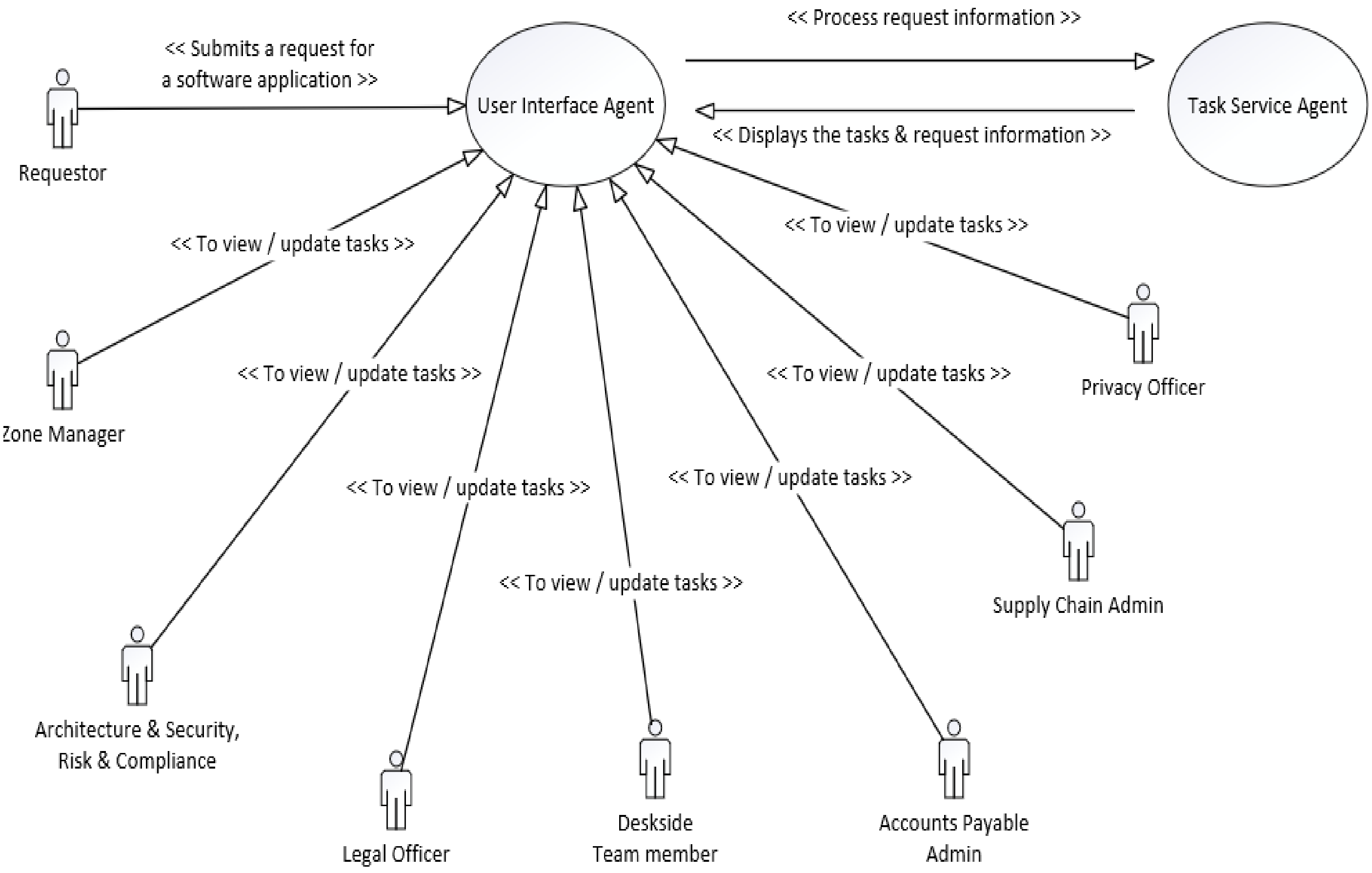
Md. Shohug Hossain - 30162605

Rohit Yeast - 30067839

Use Cases

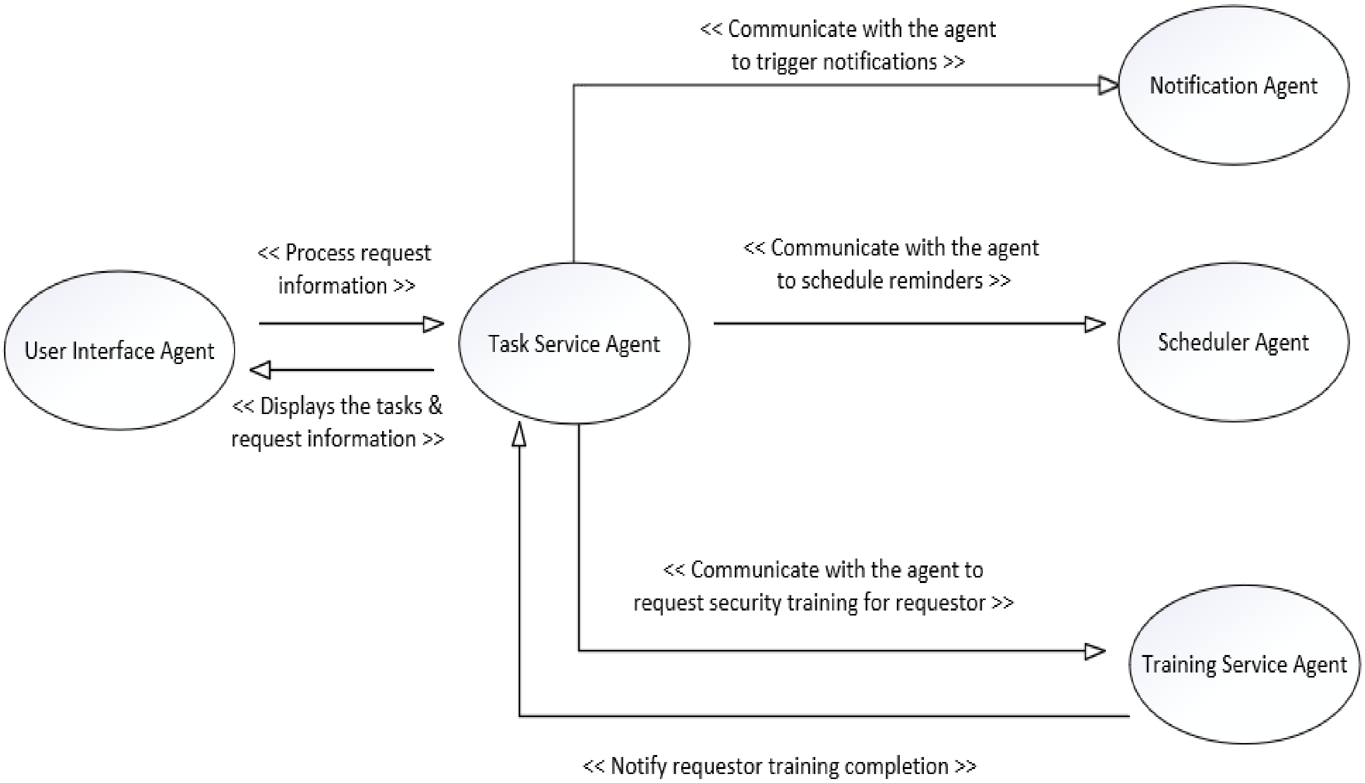
Use cases and use case definition for all the participating agents are documented below.

1. Use Case: User Interface Agent



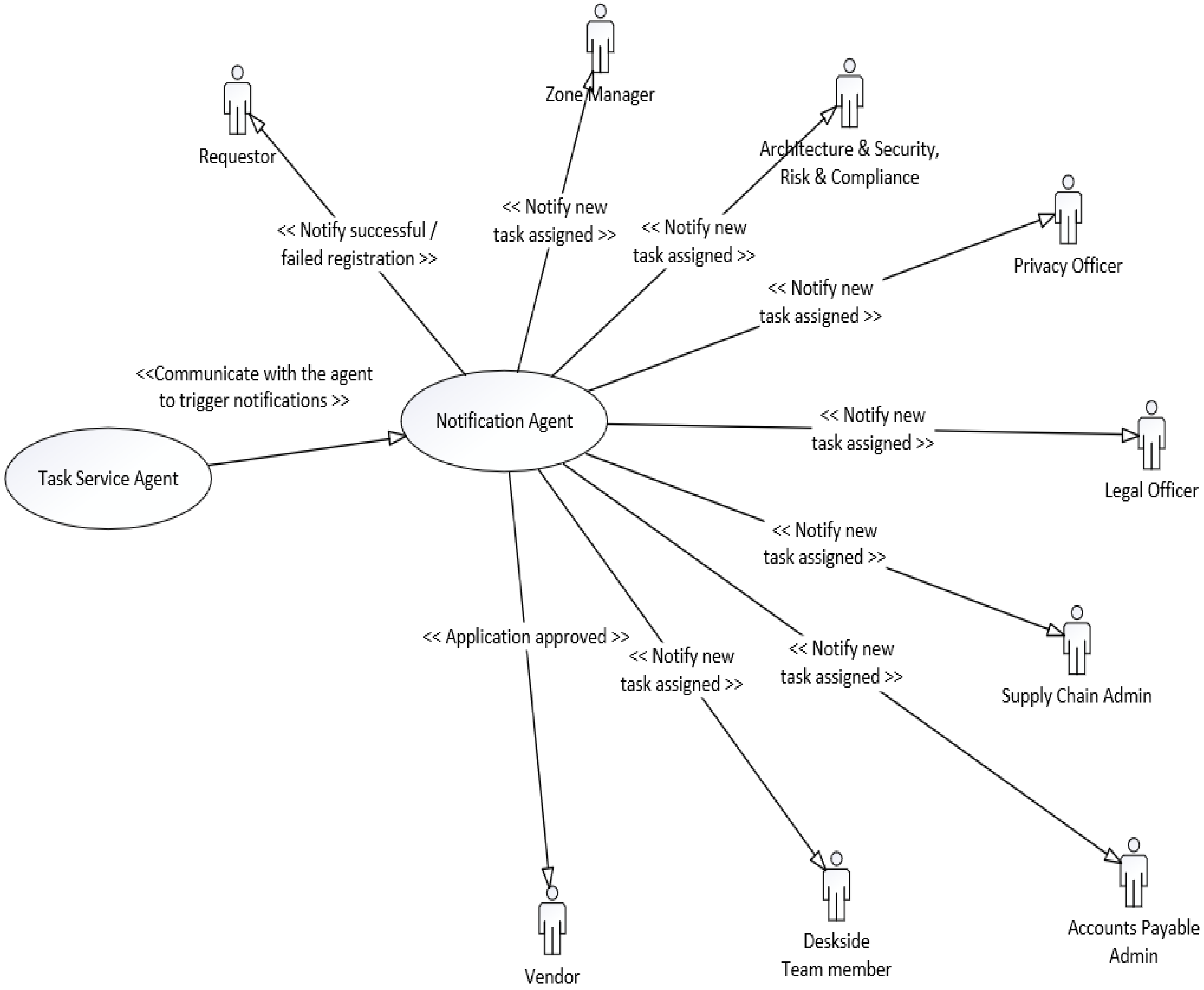
<< Process request information >>
<< Submits a request for a software application >>
User Interface Agent
Task Service Agent
<< Displays the tasks & request information >>
Requestor
<< To view / update tasks >>
<< To view / update tasks >>
Privacy Officer
Zone Manager
<< To view / update tasks >>
<< To view / update tasks >>
<< To view / update tasks >>
<< To view / update tasks >>
Supply Chain Admin
<< To view / update tasks >>
Architecture & Security, Risk & Compliance
Legal Officer
Deskside Team member
Accounts Payable Admin

2. Use Case: Task Service Agent

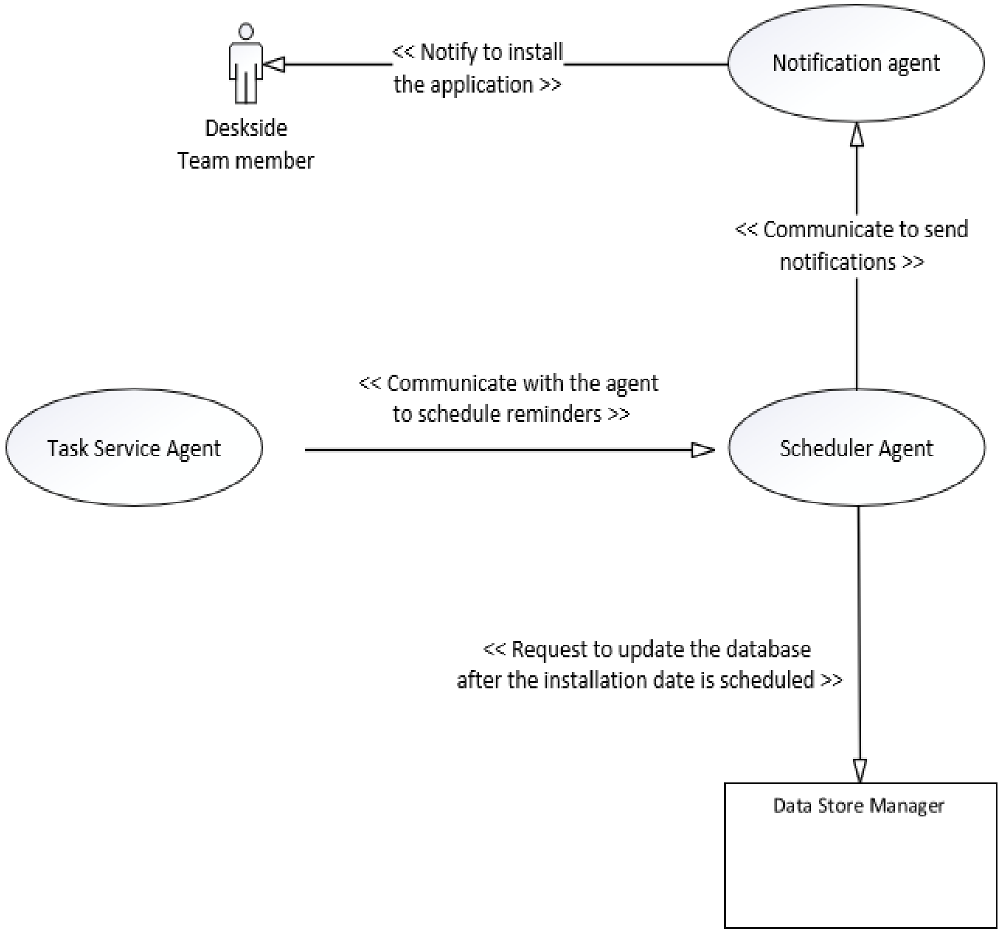


<< Communicate with the agent to trigger notifications >>
Notification Agent
<< Process request information >>
<< Communicate with the agent to schedule reminders >>
User Interface Agent
Task Service Agent
Scheduler Agent
<< Displays the tasks & request information >>
<< Communicate with the agent to request security training for requestor >>
Training Service Agent
<< Notify requestor training completion >>

3. Use Case: Notification Agent


Zone Manager
Requestor
Architecture & Security, Risk & Compliance
Privacy Officer
<< Notify new task assigned >>
<< Notify successful / failed registration >>
<< Notify new task assigned >>
<< Notify new task assigned >>
<< Notify new task assigned >>
<<Communicate with the agent to trigger notifications >>
Notification Agent
Task Service Agent
Legal Officer
<< Notify new task assigned >>
<< Application approved >>
<< Notify new task assigned >>
<< Notify new task assigned >>
Supply Chain Admin
Vendor
Deskside Team member
Accounts Payable Admin

4. Use Case: Scheduler Agent


Notification agent
<< Notify to install the application >>
Deskside Team member
<< Communicate to send notifications >>
<< Communicate with the agent to schedule reminders >>
Task Service Agent
Scheduler Agent
<< Request to update the database after the installation date is scheduled >>
Data Store Manager

5. Use Case: Training Service Agent

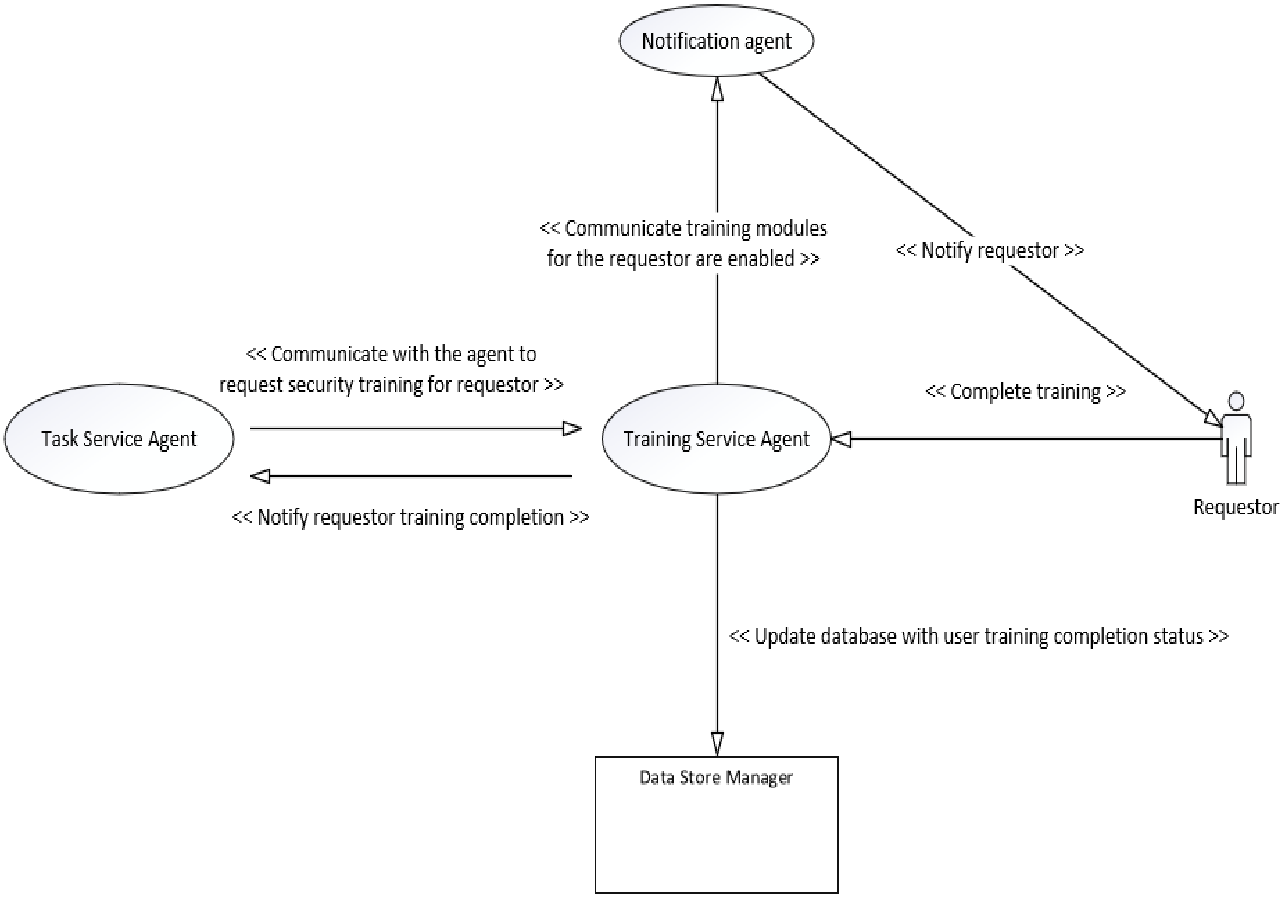
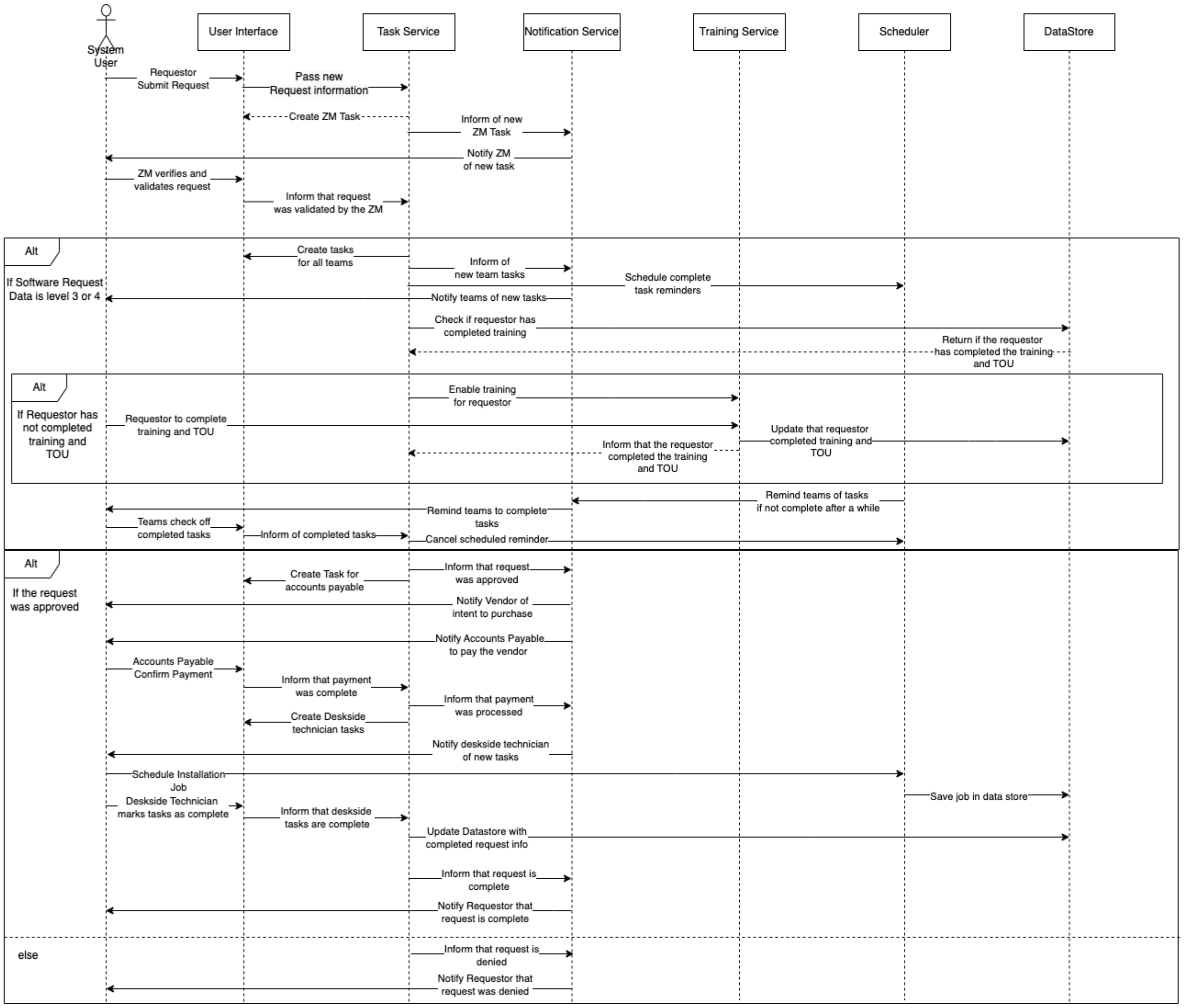

Notification agent
<< Communicate training modules for the requestor are enabled >>
<< Notify requestor >>
<< Communicate with the agent to request security training for requestor >>
Task Service Agent
Training Service Agent
<< Complete training >>
Requestor
<< Notify requestor training completion >>
<< Update database with user training completion status >>
Data Store Manager

Table 1: Use Case Definition Table

Brief Description:	The requestor uses this use case to request to acquire and install the software
Precondition(s):	• Actor profiles are created before providing any service • Software application request information is available • Requestor having access to training module to complete training
Post condition(s):	If all the requirements are successfully met, then the requestor will be able to acquire the software and get it installed
Process Steps	
1	Requestor submits a request to acquire software in User Interface agent
2	Request information is sent to Task Service agent to process the request
3	Task Service agent creates and displays all the tasks in User Interface agent
4	Task assignees views and updates respective assigned tasks
5	Task Service agent communicates with Notification agent to create respective notifications to all users
6	Notification agent notifies all the task assignees, requestor and vendor with respective notifications
7	Task Service agent communicates with Training service agent to request security training for requestor
8	Training Service agent communicates with the Notification agent to notify requestor regarding training details
9	Training Service agent notifies Task Service agent about requestor training completion
10	Training Service agent updates the database in data store manager with user training completion status
11	Task Service agent communicates with Scheduler agent to schedule respective reminders
12	Scheduler agent schedules a date and time for the software installation
13	Scheduler agent communicates with Notification agent to notify installation schedule details to Deskside Team member
14	Scheduler agent updates the database in data store manager after the installation date is scheduled
Relationships:	
Initiating	Requestor
Collaborating	Task Service agent, Notification agent, Scheduler agent, Training Service agent

Overall Message Sequence Diagram

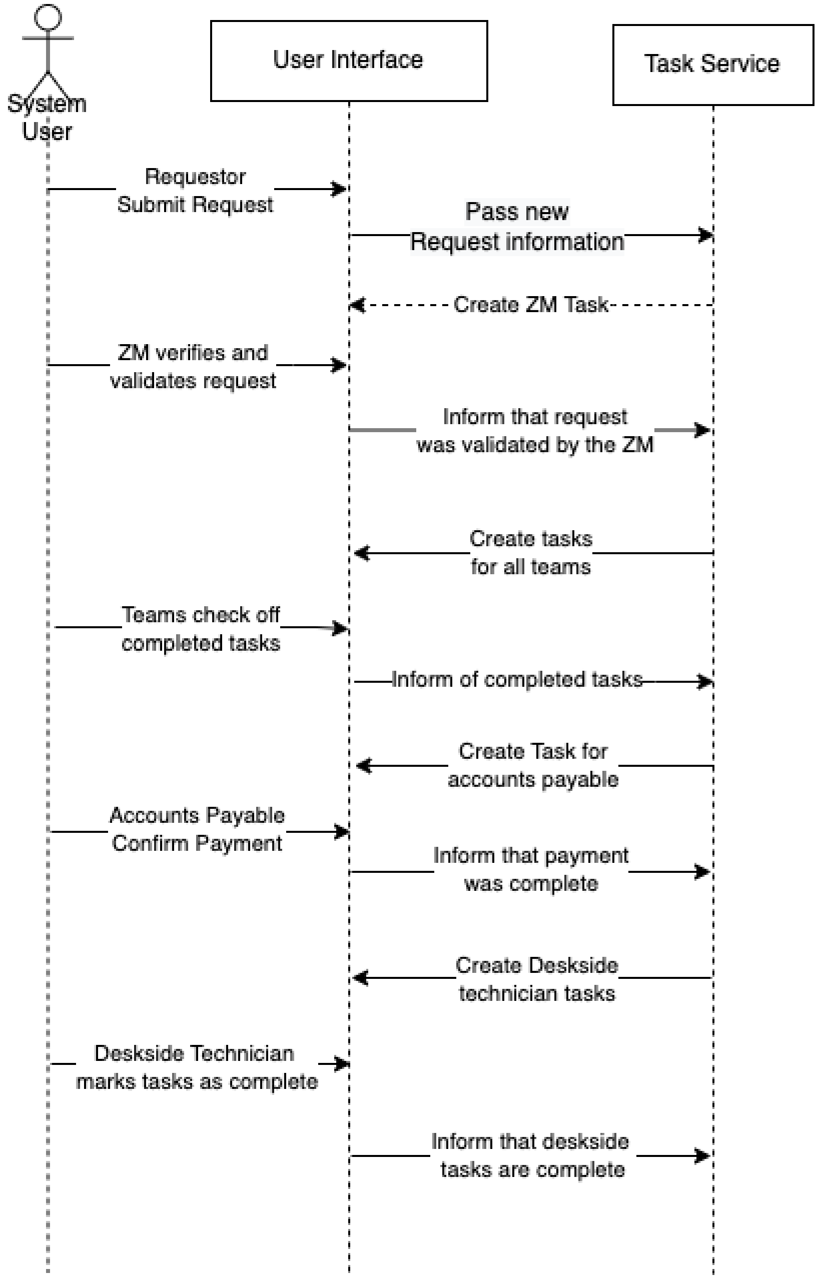
The following sequence diagram shows the sequence of messages for all agents for all different branches of the program.


System User
User Interface
Task Service
Notification Service
Training Service
Scheduler
DataStore
Requestor Submit Request
Pass new Request information
Create ZM Task
Inform of new ZM Task
Notify ZM of new task
ZM verifies and validates request
Inform that request was validated by the ZM
Alt
If Software Request Data is level 3 or 4
Create tasks for all teams
Inform of new team tasks
Schedule complete task reminders
Notify teams of new tasks
Check if requestor has completed training
Return if the requestor has completed the training and TOU
Alt
If Requestor has not completed training and TOU
Enable training for requestor
Requestor to complete training and TOU
Update that requestor completed training and TOU
Inform that the requestor completed the training and TOU
Remind teams of tasks if not complete after a while
Remind teams to complete tasks
Teams check off completed tasks
Inform of completed tasks
Cancel scheduled reminder
Alt
If the request was approved
Inform that request was approved
Create Task for accounts payable
Notify Vendor of intent to purchase
Notify Accounts Payable to pay the vendor
Accounts Payable Confirm Payment
Inform that payment was complete
Inform that payment was processed
Create Deskside technician tasks
Notify deskside technician of new tasks
Schedule Installation Job
Save job in data store
Deskside Technician marks tasks as complete
Inform that deskside tasks are complete
Update Datastore with completed request info
Inform that request is complete
Notify Requestor that request is complete
else
Inform that request is denied
Notify Requestor that request was denied

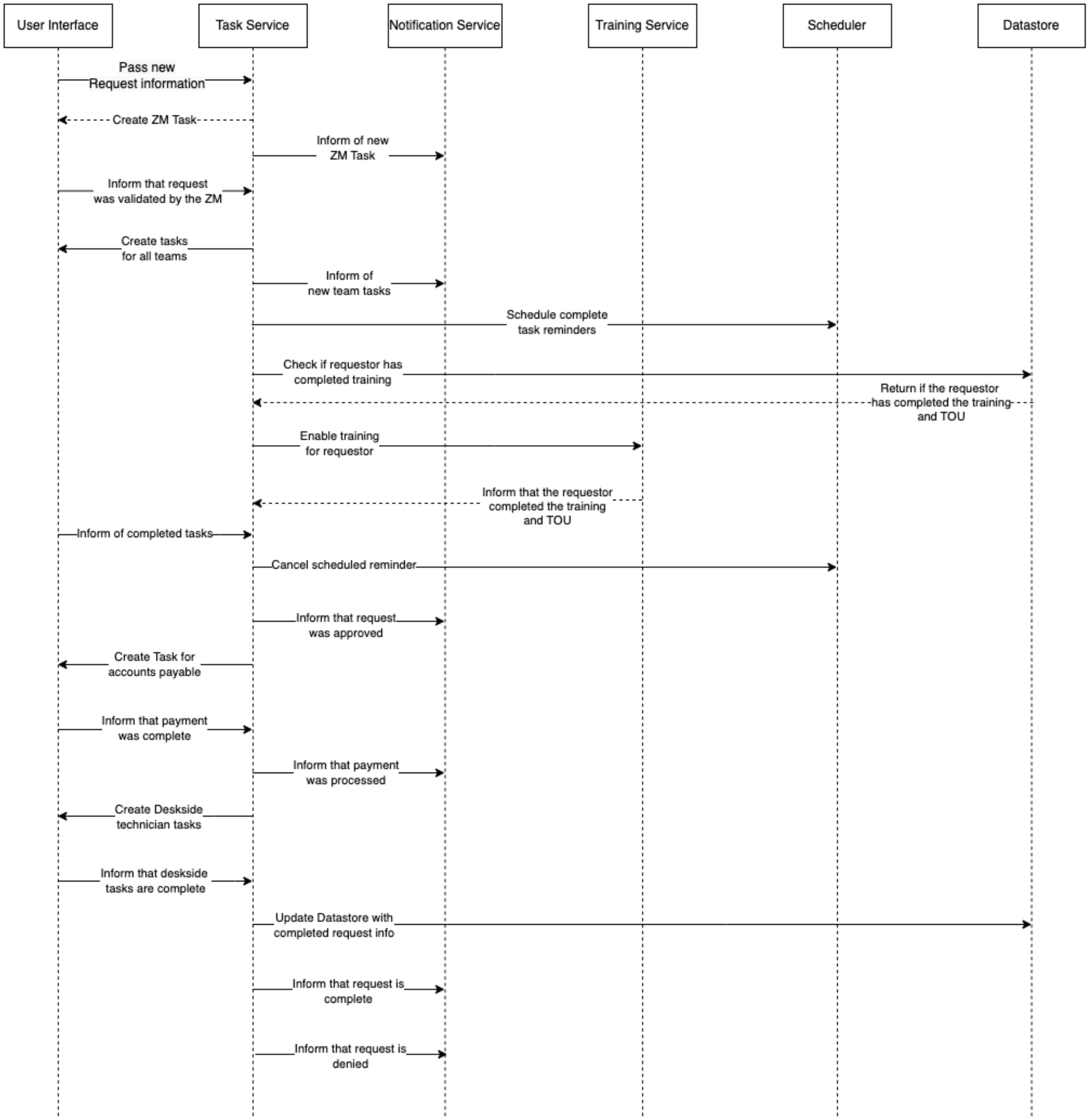
Agent Specific Message Sequence Diagrams

The following message sequence diagrams show the messaging order between agents with each diagram being focused on one agent. Because the primary use cases for the vast majority of the messages involve requests of level 3 or 4 that are also approved the diagrams are simplified to only show only these cases. The complete sequence diagram for all cases is shown above.

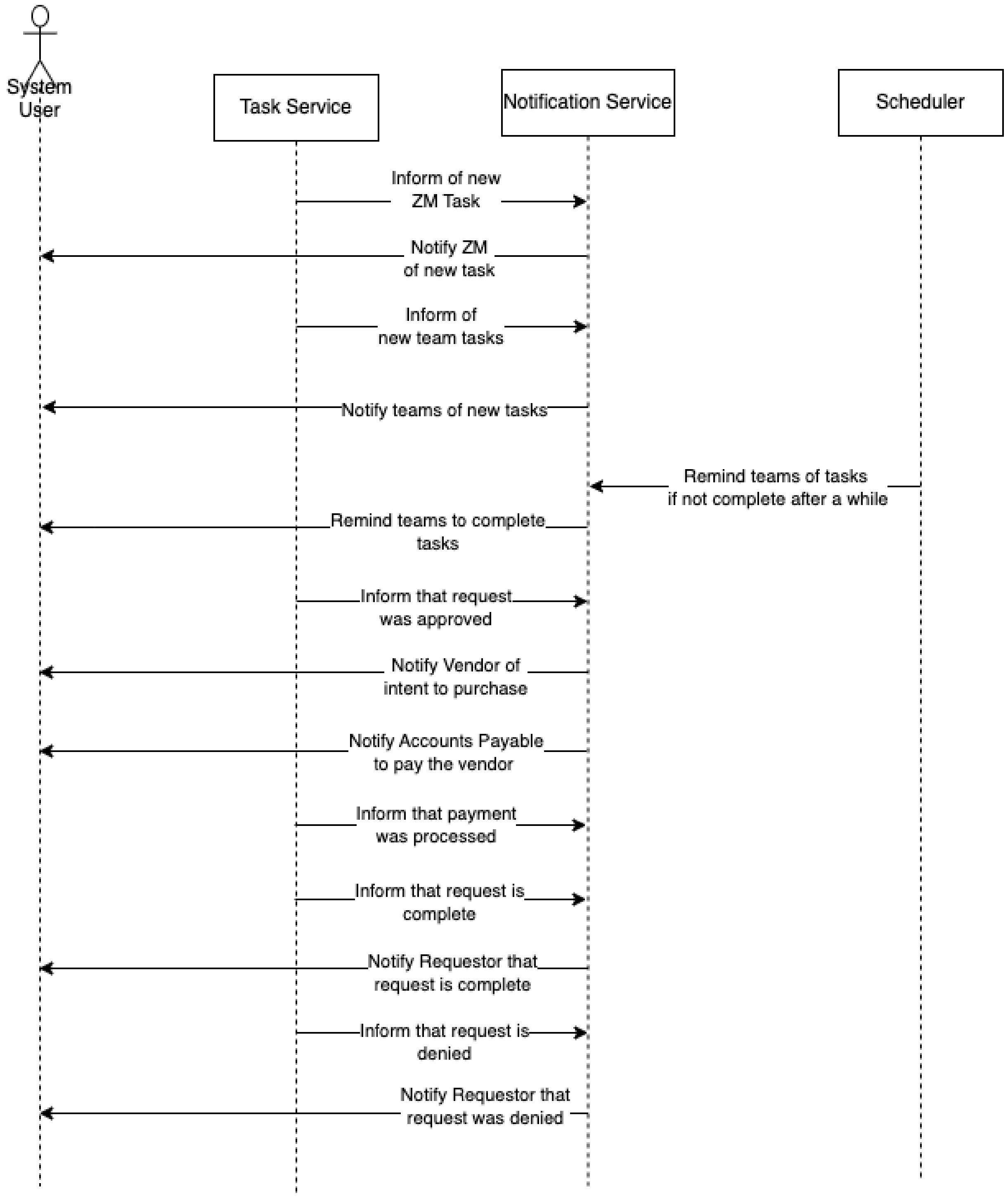
User Interface Agent


System User
User Interface
Task Service
Requestor Submit Request
Pass new Request information
Create ZM Task
ZM verifies and validates request
Inform that request was validated by the ZM
Create tasks for all teams
Teams check off completed tasks
Inform of completed tasks
Create Task for accounts payable
Accounts Payable Confirm Payment
Inform that payment was complete
Create Deskside technician tasks
Deskside Technician marks tasks as complete
Inform that deskside tasks are complete

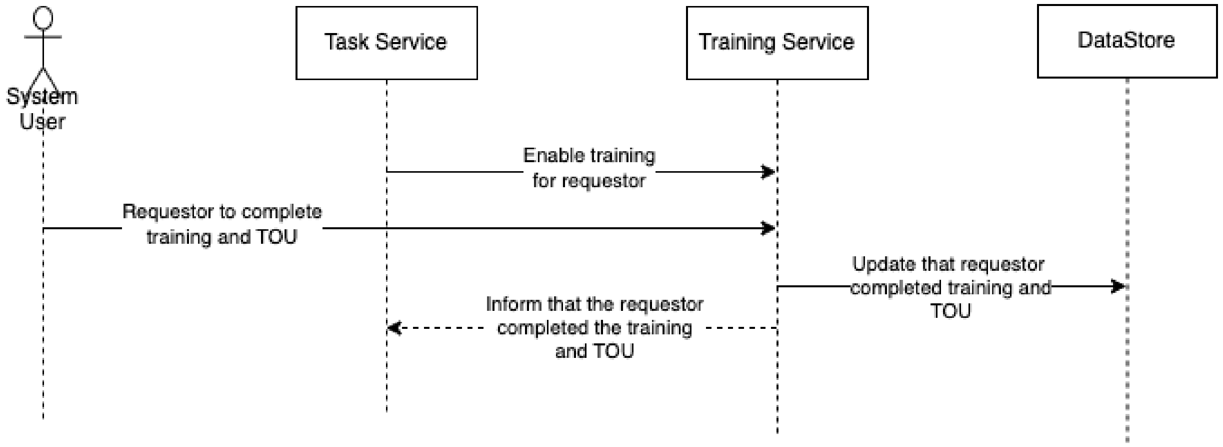
Task Service Agent


User Interface
Task Service
Notification Service
Training Service
Scheduler
Datastore
Pass new Request information
Create ZM Task
Inform of new ZM Task
Inform that request was validated by the ZM
Create tasks for all teams
Inform of new team tasks
Schedule complete task reminders
Check if requestor has completed training
Return if the requestor has completed the training and TOU
Enable training for requestor
Inform that the requestor completed the training and TOU
Inform of completed tasks
Cancel scheduled reminder
Inform that request was approved
Create Task for accounts payable
Inform that payment was complete
Inform that payment was processed
Create Deskside technician tasks
Inform that deskside tasks are complete
Update Datastore with completed request info
Inform that request is complete
Inform that request is denied

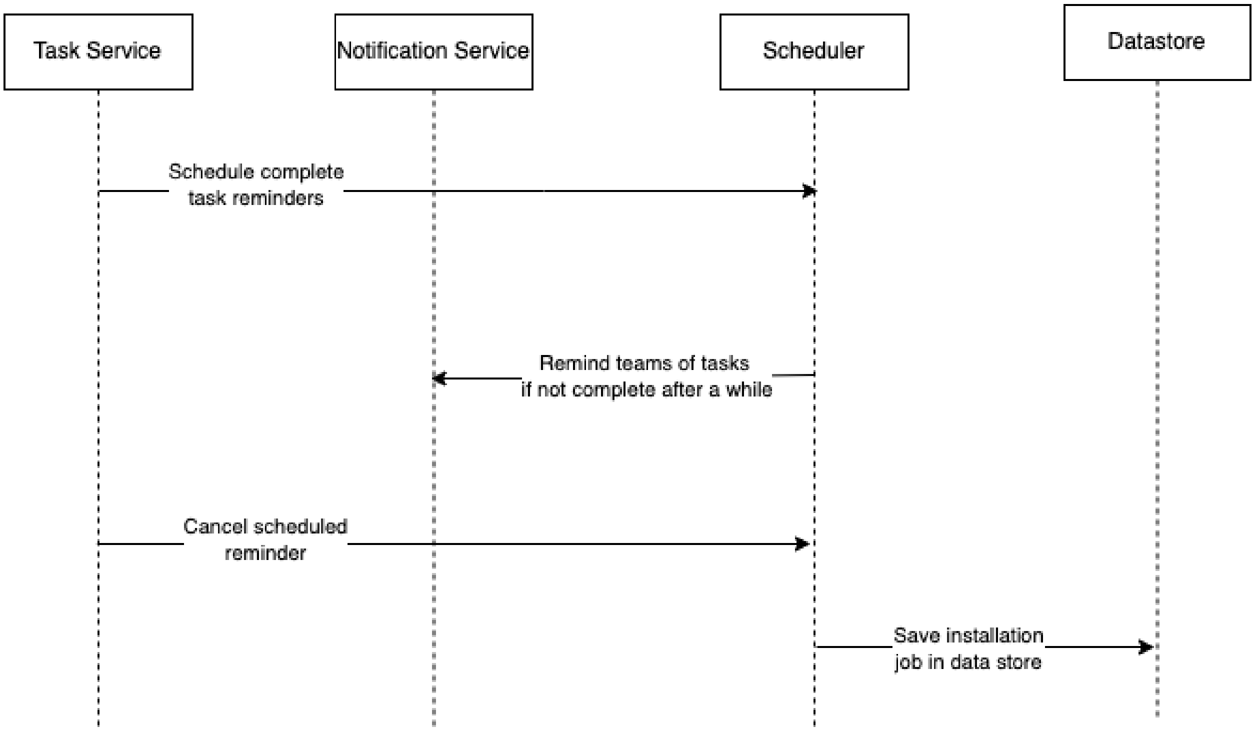
Notification Service Agent


System User
Task Service
Notification Service
Scheduler
Inform of new ZM Task
Notify ZM of new task
Inform of new team tasks
Notify teams of new tasks
Remind teams of tasks if not complete after a while
Remind teams to complete tasks
Inform that request was approved
Notify Vendor of intent to purchase
Notify Accounts Payable to pay the vendor
Inform that payment was processed
Inform that request is complete
Notify Requestor that request is complete
Inform that request is denied
Notify Requestor that request was denied

Training Agent


Task Service
Training Service
DataStore
System User
Enable training for requestor
Requestor to complete training and TOU
Update that requestor completed training and TOU
Inform that the requestor completed the training and TOU

Scheduler Agent


Task Service
Notification Service
Scheduler
Datastore
Schedule complete task reminders
Remind teams of tasks if not complete after a while
Cancel scheduled reminder
Save installation job in data store

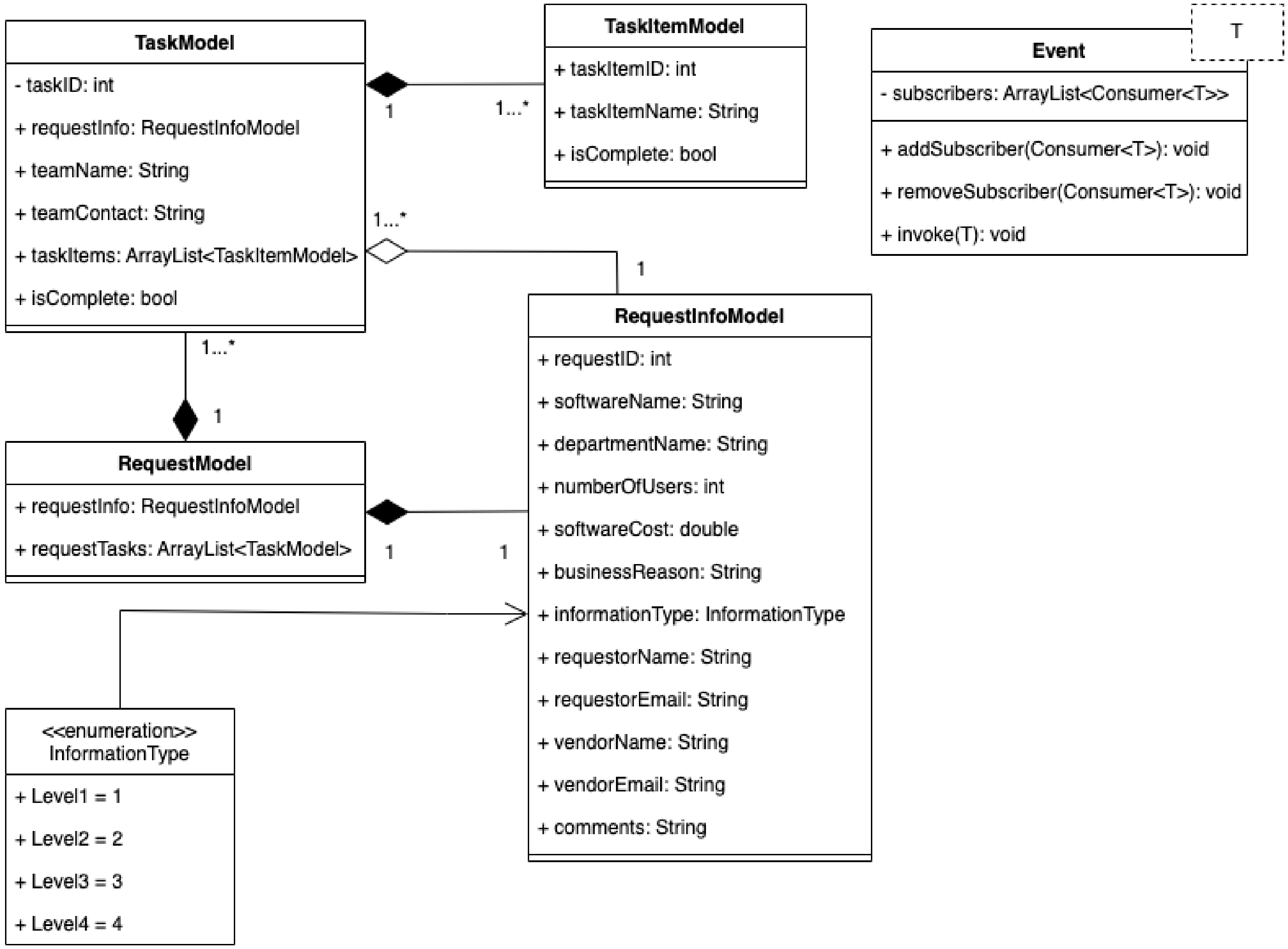
Detailed Agent Architecture

The architecture of each of the 5 agents and the shared code for them is defined in UML class diagrams. Because the agent-based system is constructed using JADE (Java Agent DEvelopment) the communication between agents is handled by JADE objects and is not shown in the class diagrams. While the class diagrams attempt to define all other functionality of the agents fully, small details and gaps in the design of the system will become apparent throughout the development process. Therefore the finished product will almost certainly not be 100% faithful to the design laid out in the following UML class diagrams.

Model and Utility Classes

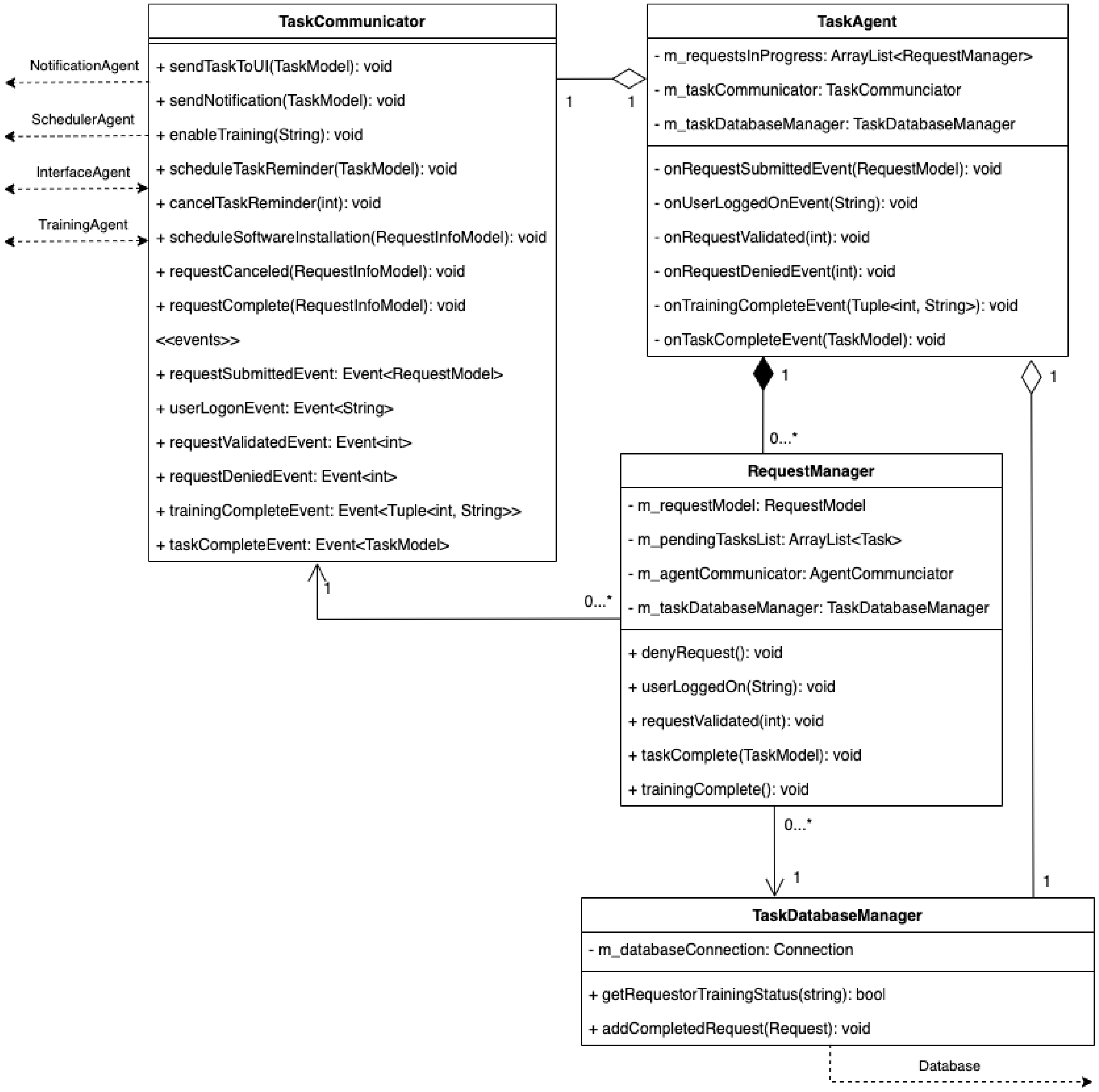
The following class diagram depicts the expected model and utility classes that will be used for the system. The model classes show the data members that would be accessible via getters and setters. The getters and setters themselves are not shown because that would become quite verbose and would not be particularly useful for understanding the system architecture. In addition to the model classes, an enumerable class is used to represent the different data levels that the requested software may require access to.

Finally, the Event class is used to help the development team reduce the amount of two-way dependencies in the program. It allows objects to subscribe to events from other objects and react to them when they are invoked.



Task Agent Architecture

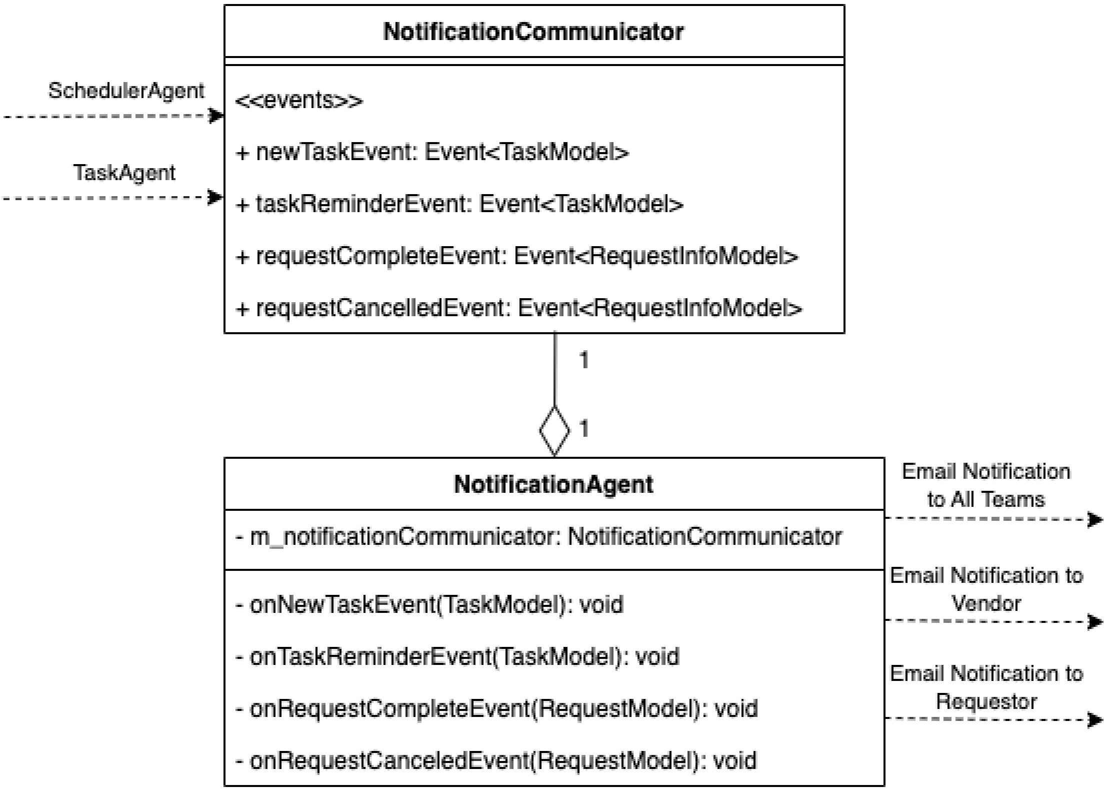
The task agent controls many of the workflows of the software acquisition request system and dispatches the other agents. It manages all the requests and tasks that the teams are assigned to work on and communicates with all other agents in the system. The task agent is also responsible for saving the request information to the database when the request is complete and finding out if the requestor completed the relevant training if the software request data is level 3 or 4.



TaskCommunicator
+ sendTaskToUI(TaskModel): void
+ sendNotification(TaskModel): void
+ enableTraining(String): void
+ scheduleTaskReminder(TaskModel): void
+ cancelTaskReminder(int): void
+ scheduleSoftwareInstallation(RequestInfoModel): void
+ requestCanceled(RequestInfoModel): void
+ requestComplete(RequestInfoModel): void
<<events>>
+ requestSubmittedEvent: Event<RequestModel>
+ userLogonEvent: Event<String>
+ requestValidatedEvent: Event<int>
+ requestDeniedEvent: Event<int>
+ trainingCompleteEvent: Event<Tuple<int, String>>
+ taskCompleteEvent: Event<TaskModel>
NotificationAgent
SchedulerAgent
InterfaceAgent
TrainingAgent
1
1
TaskAgent
- m_requestsInProgress: ArrayList<RequestManager>
- m_taskCommunicator: TaskCommunciator
- m_taskDatabaseManager: TaskDatabaseManager
- onRequestSubmittedEvent(RequestModel): void
- onUserLoggedOnEvent(String): void
- onRequestValidated(int): void
- onRequestDeniedEvent(int): void
- onTrainingCompleteEvent(Tuple<int, String>): void
- onTaskCompleteEvent(TaskModel): void
1
1
0...*
RequestManager
- m_requestModel: RequestModel
- m_pendingTasksList: ArrayList<Task>
- m_agentCommunicator: AgentCommunciator
- m_taskDatabaseManager: TaskDatabaseManager
+ denyRequest(): void
+ userLoggedOn(String): void
+ requestValidated(int): void
+ taskComplete(TaskModel): void
+ trainingComplete(): void
1
0...*
0...*
1
1
TaskDatabaseManager
- m_databaseConnection: Connection
+ getRequestorTrainingStatus(string): bool
+ addCompletedRequest(Request): void
Database

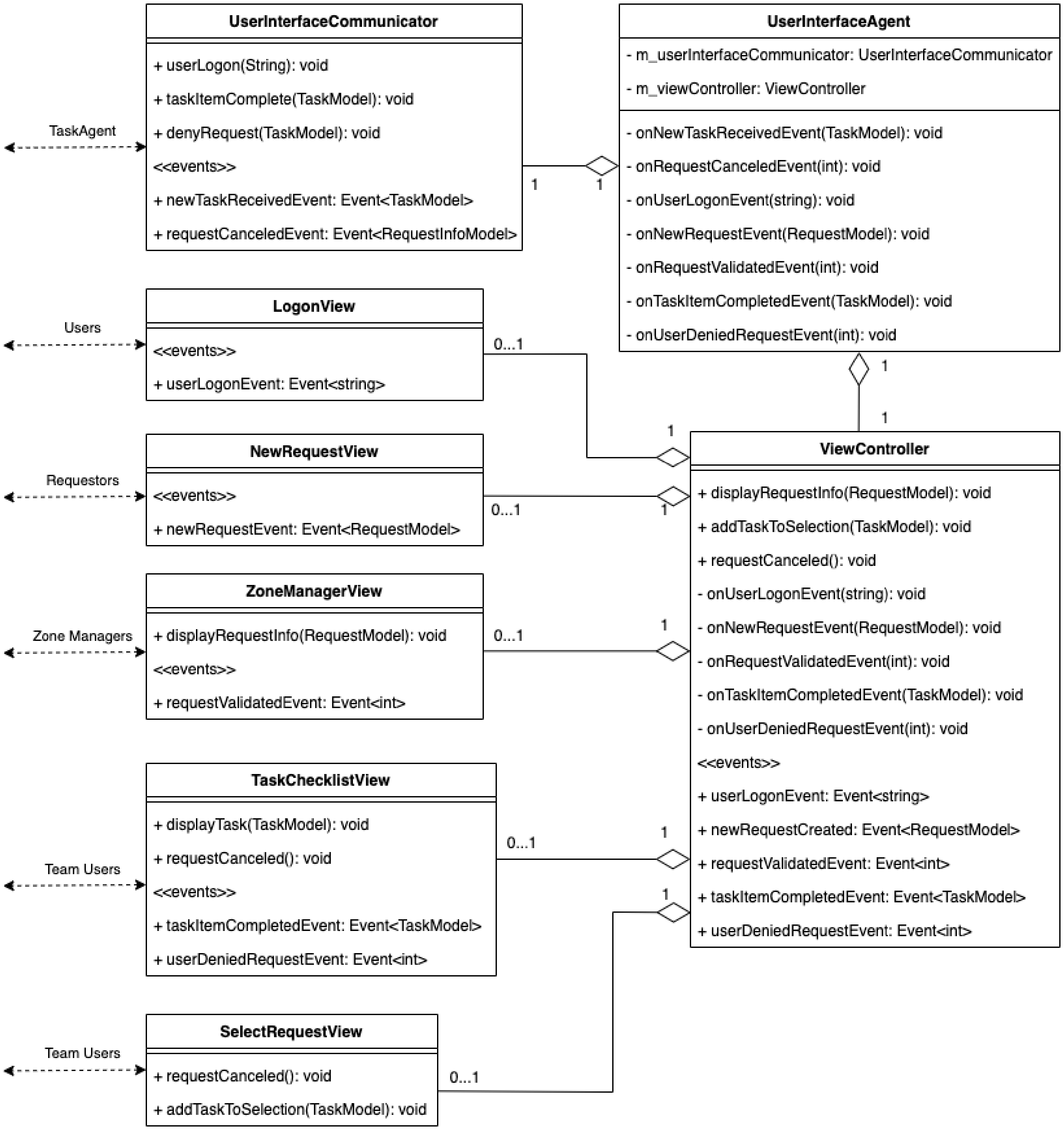
Notification Agent Architecture

The notification agent is responsible for notifying the requestor if their request was denied or completed, notifying and reminding all teams to complete their assigned tasks and notifying the vendor when of the intent to complete the purchase of the software. The notification agent receives messages from the task agent and the scheduler agent which determines what notifications to send and where to send them.


NotificationCommunicator
SchedulerAgent
TaskAgent
<<events>>
+ newTaskEvent: Event<TaskModel>
+ taskReminderEvent: Event<TaskModel>
+ requestCompleteEvent: Event<RequestInfoModel>
+ requestCancelledEvent: Event<RequestInfoModel>
1
1
NotificationAgent
- m_notificationCommunicator: NotificationCommunicator
- onNewTaskEvent(TaskModel): void
- onTaskReminderEvent(TaskModel): void
- onRequestCompleteEvent(RequestModel): void
- onRequestCanceledEvent(RequestModel): void
Email Notification to All Teams
Email Notification to Vendor
Email Notification to Requestor

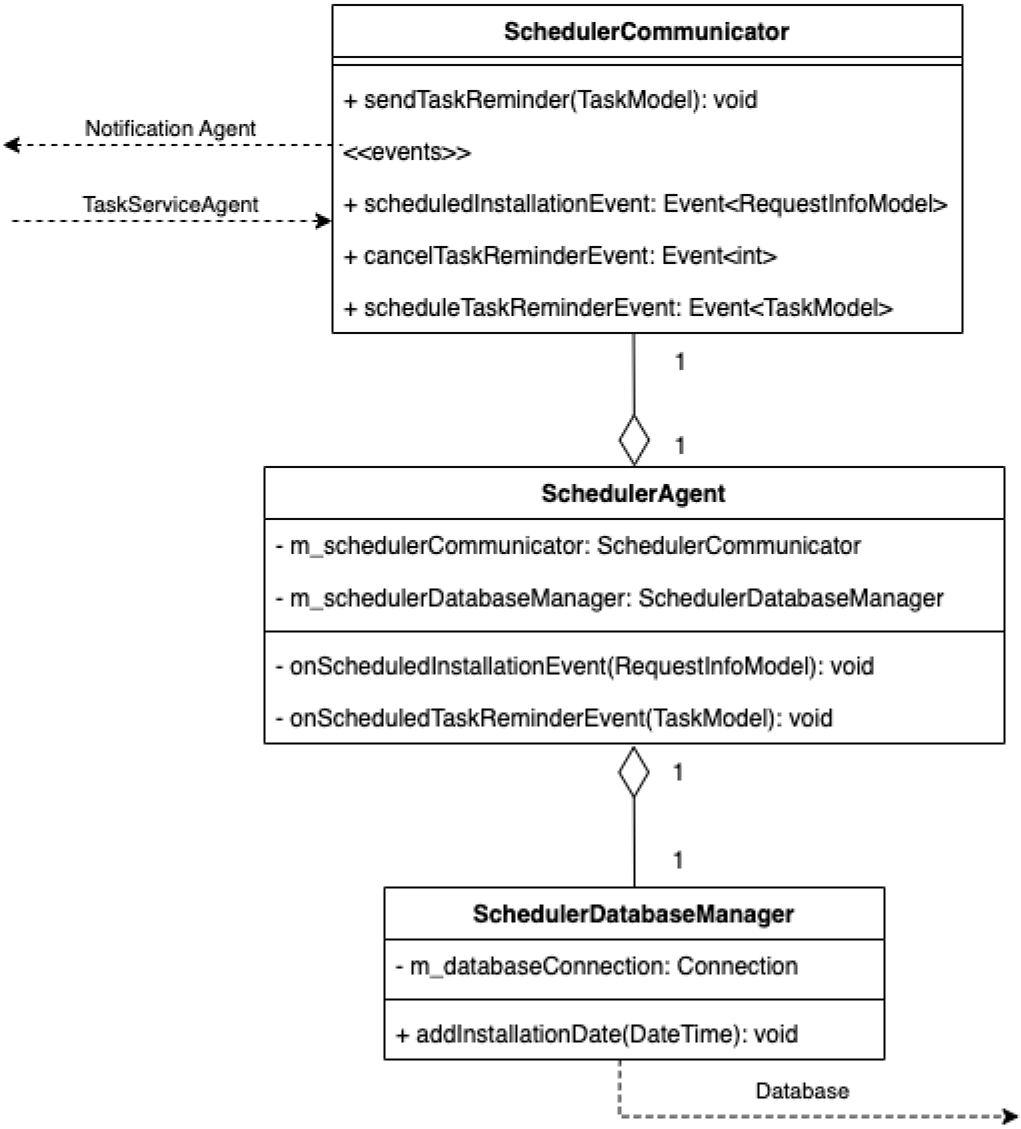
User Interface Agent Architecture

The interface agent manages the graphical user interface for the application. It allows requestors to submit their requests, the zone manager to review and validate the software acquisition request and all of the teams to mark their tasks as complete. The interface agent communicates with the task agent to get task and request information, submit requests and inform the task service when tasks are marked as completed by the teams.


UserInterfaceCommunicator
+ userLogon(String): void
+ taskItemComplete(TaskModel): void
+ denyRequest(TaskModel): void
<<events>>
+ newTaskReceivedEvent: Event<TaskModel>
+ requestCanceledEvent: Event<RequestInfoModel>
TaskAgent
1
1
UserInterfaceAgent
- m_userInterfaceCommunicator: UserInterfaceCommunicator
- m_viewController: ViewController
- onNewTaskReceivedEvent(TaskModel): void
- onRequestCanceledEvent(int): void
- onUserLogonEvent(string): void
- onNewRequestEvent(RequestModel): void
- onRequestValidatedEvent(int): void
- onTaskItemCompletedEvent(TaskModel): void
- onUserDeniedRequestEvent(int): void
1
1
Users
LogonView
<<events>>
+ userLogonEvent: Event<string>
0...1
1
Requestors
NewRequestView
<<events>>
+ newRequestEvent: Event<RequestModel>
0...1
1
ViewController
+ displayRequestInfo(RequestModel): void
+ addTaskToSelection(TaskModel): void
+ requestCanceled(): void
- onUserLogonEvent(string): void
- onNewRequestEvent(RequestModel): void
- onRequestValidatedEvent(int): void
- onTaskItemCompletedEvent(TaskModel): void
- onUserDeniedRequestEvent(int): void
<<events>>
+ userLogonEvent: Event<string>
+ newRequestCreated: Event<RequestModel>
+ requestValidatedEvent: Event<int>
+ taskItemCompletedEvent: Event<TaskModel>
+ userDeniedRequestEvent: Event<int>
Zone Managers
ZoneManagerView
+ displayRequestInfo(RequestModel): void
<<events>>
+ requestValidatedEvent: Event<int>
0...1
1
Team Users
TaskChecklistView
+ displayTask(TaskModel): void
+ requestCanceled(): void
<<events>>
+ taskItemCompletedEvent: Event<TaskModel>
+ userDeniedRequestEvent: Event<int>
0...1
1
1
Team Users
SelectRequestView
+ requestCanceled(): void
+ addTaskToSelection(TaskModel): void
0...1

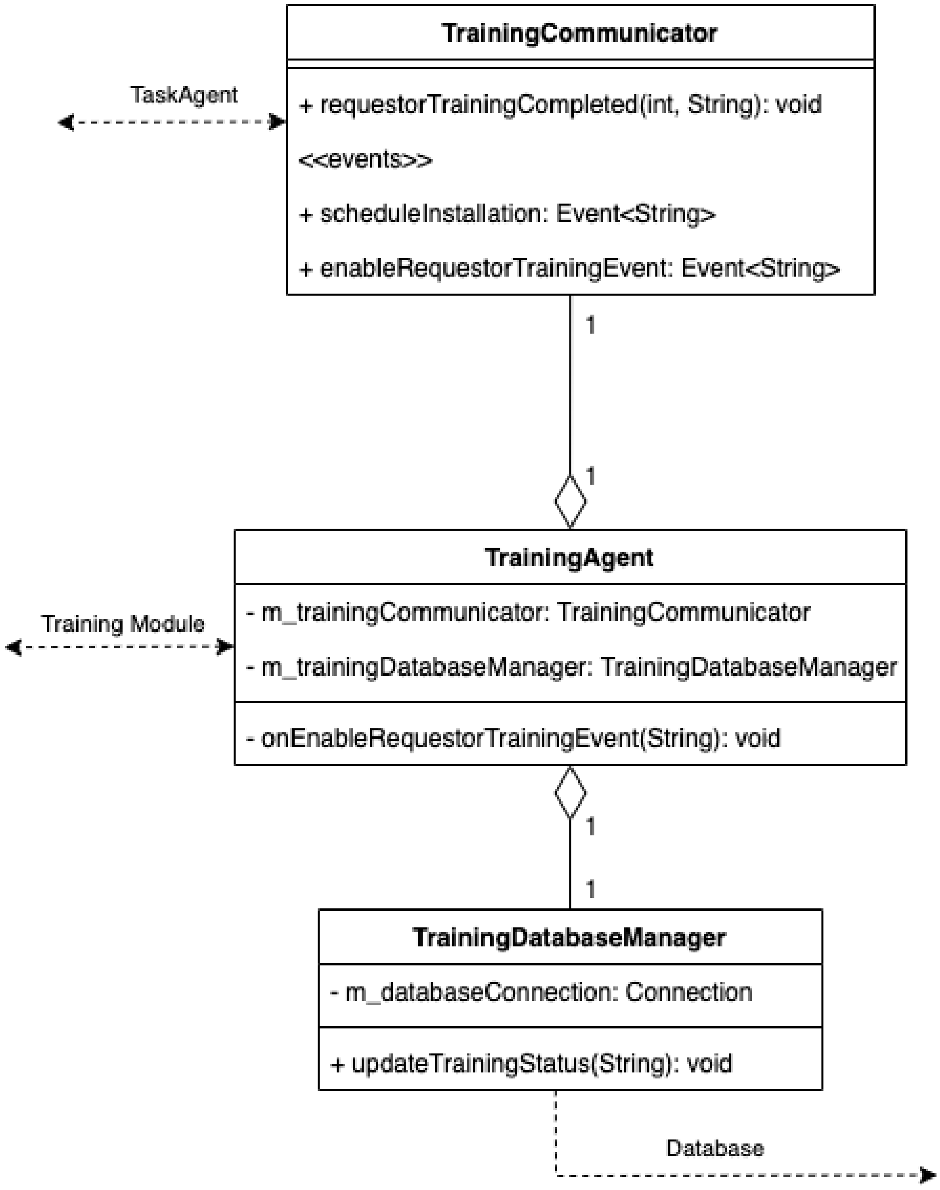
Scheduler Agent Architecture

The scheduler agent is responsible for scheduling reminders for task assignees to complete their task and for scheduling a date for a deskside agent to install the requested software. The scheduler agent receives information from the Task Service Agent when a task reminder or installation job needs to be scheduled. It then communicates with the Notification agent to request a notification to be sent with a software installation date or a task reminder. It also requests to update the database with the installation date and software installer for a specific SAR once it is scheduled.


SchedulerCommunicator
+ sendTaskReminder(TaskModel): void
<<events>>
+ scheduledInstallationEvent: Event<RequestInfoModel>
+ cancelTaskReminderEvent: Event<int>
+ scheduleTaskReminderEvent: Event<TaskModel>
Notification Agent
TaskServiceAgent
1
1
SchedulerAgent
- m_schedulerCommunicator: SchedulerCommunicator
- m_schedulerDatabaseManager: SchedulerDatabaseManager
- onScheduledInstallationEvent(RequestInfoModel): void
- onScheduledTaskReminderEvent(TaskModel): void
1
1
SchedulerDatabaseManager
- m_databaseConnection: Connection
+ addInstallationDate(DateTime): void
Database

Training Agent Architecture

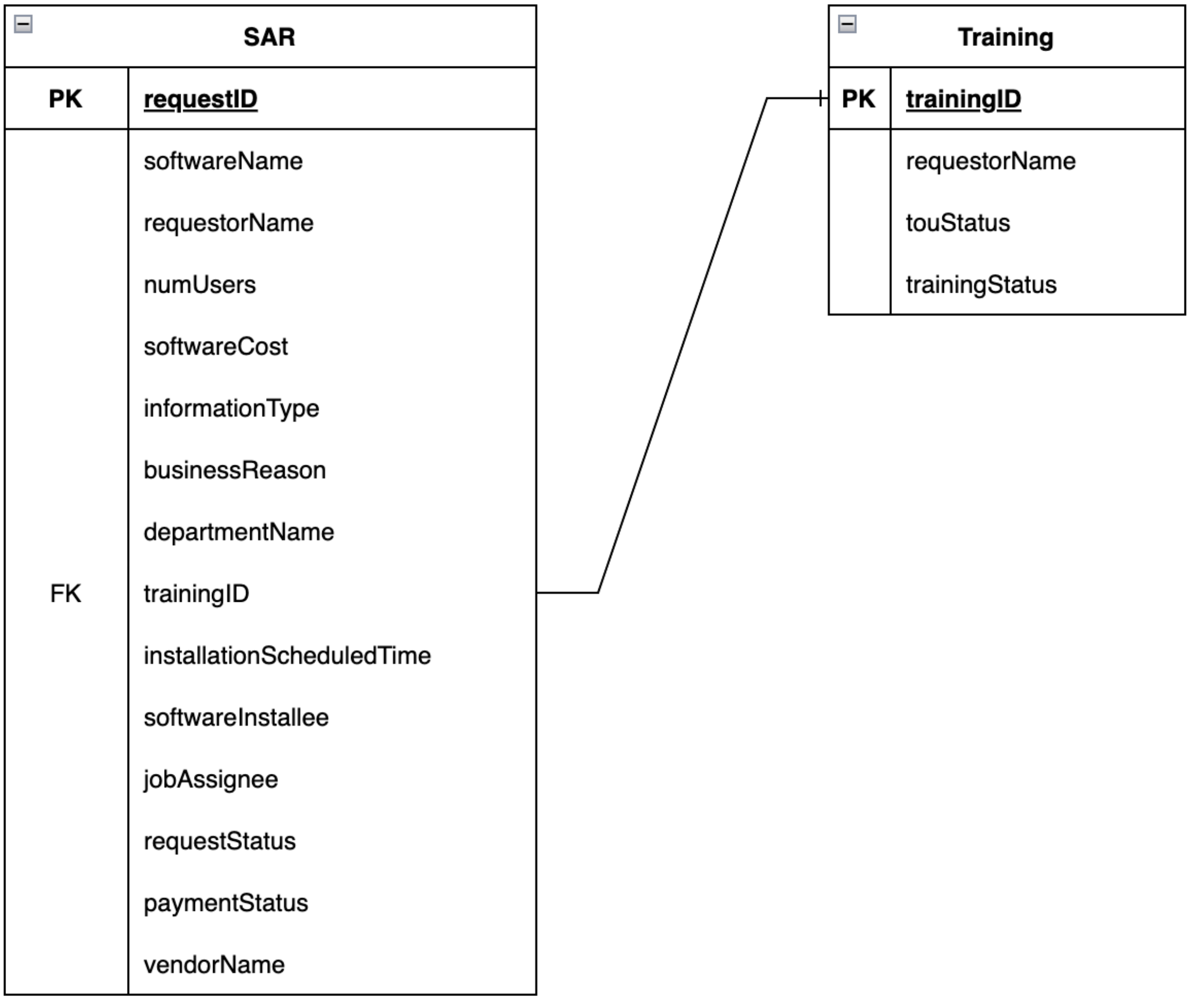
The Training Agent coordinates user security training and Terms of Use agreement between the Task Agent and the training module (external). When a user submits a SAR, the Task Agent asks the Training Agent to enable the requestor's training in the external training module. Once the requestor has completed their training, the external training module notifies the Training Agent that will then relay the completion to the Task Agent and request to update the training status for the user in the database.


TrainingCommunicator
TaskAgent
+ requestorTrainingCompleted(int, String): void
<<events>>
+ scheduleInstallation: Event<String>
+ enableRequestorTrainingEvent: Event<String>
1
1
TrainingAgent
Training Module
- m_trainingCommunicator: TrainingCommunicator
- m_trainingDatabaseManager: TrainingDatabaseManager
- onEnableRequestorTrainingEvent(String): void
1
1
TrainingDatabaseManager
- m_databaseConnection: Connection
+ updateTrainingStatus(String): void
Database

Database

The database consists of two tables: SAR (Software Acquisition Request) and Training.

- The SAR table contains the information about each software acquisition request that is approved; each task assignee has to complete their task before the database will be updated with a SAR's information.
- The Training table contains information about the user training for each SAR. Its primary key, *trainingID*, is a foreign key in the SAR table; it is used to verify that the trainingStatus and touStatus corresponding to its *trainingID* are complete.



	SAR
PK	**requestID**
	softwareName
	requestorName
	numUsers
	softwareCost
	informationType
	businessReason
	departmentName
FK	trainingID
	installationScheduledTime
	softwareInstallee
	jobAssignee
	requestStatus
	paymentStatus
	vendorName

	Training
PK	**trainingID**
	requestorName
	touStatus
	trainingStatus

All database fields are detailed in the table below with their description and data type.

Table 2: Data Definition for SAR Table

Field	**Description**	**Type**
requestNumber	Unique ID assigned to each request upon being instantiated, used to identify a request	integer
requestorName	Name of the user submitting a SAR	string
softwareName	Name of the requested software	string
businessReason	Reason why the user is submitting a SAR	string
departmentName	Department that the software will be used in	string
trainingID (foreign key)	Unique ID assigned to each user’s training for each SAR	integer
numUsers	Number of users for the requested software	integer
softwareCost	Cost of the software	float
informationType	Data information level	integer
installationScheduledTime	Date and time for the software installation	DateTime
softwareInstallee	Deskside technician who will be installing the software	string
requestStatus	Status of the request (approved, complete)	string
paymentStatus	Status of the software payment	string
vendorName	Name of the software vendor	string

Table 3: Data Definition for Training Table

Field	**Description**	**Type**
trainingID	Unique ID assigned to each user’s training for each SAR	integer
requestorName	Name of the user who submitted the SAR	string
touStatus	Requestor’s status on their Terms of Use (TOU) training; 1 for completed, 0 otherwise	bool
trainingStatus	Requestor’s status on their security training; 1 for completed, 0 otherwise	bool

Inter-Agents Messages

The following tables outline the payloads of messages that are sent between agents. The messages are grouped by sender because there may be multiple recipients. Note that the notification agent is mentioned in this section because the notification agent only receives messages, it does not send any messages. The inputs and outputs of inter-agent functions are described for each agent in XML format.

Message From the Task Agent

Table 4: XML Parameters from the Task Agent to the User Interface Agent

Parameter	Description

<sendTaskToUI>
  <task>
    <taskID>int</taskID>
    <requestInfo>
      <requestorName>String</requestorName>
      <requestorEmail>String</requestorEmail>
      <departmentName>String</departmentName>
      <softwareName>String</softwareName>
      <softwareCost>double</softwareCost>
      <numberOfUsers>int</numberOfUsers>
      <businessReason>String</businessReason>
      <informationType>int</informationType>
      <vendorName>String</vendorName>
      <vendorEmail>String</vendorEmail>
      <comments>String</comments>
    </requestInfo>
    <teamName>String</teamName>
    <teamContact>String</teamContact>
    <taskItems>
      <taskItem>
        <taskItemID>int</taskItemID>
        <taskItemName>String</taskItemName>
        <isComplete>b</isComplete>
      </taskItem>
      .
      n
      .
    </taskItems>
    <isComplete>bool</isComplete>
  </task>
</sendTaskToUI>
	Message from the task agent to the user interface agent with a task that is assigned to the logged in user.

Table 2: XML Parameters from the Task Agent to the Notifican and User Interface Agents

Parameter	Description

`<requestCanceled>`
  `<requestInfo>`
    `<requestorName>String</requestorName>`
    `<requestorEmail>String</requestorEmail>`
    `<departmentName>String</departmentName>`
    `<softwareName>String</softwareName>`
    `<softwareCost>double</softwareCost>`
    `<numberOfUsers>int</numberOfUsers>`
    `<businessReason>String</businessReason>`
    `<informationType>int</informationType>`
    `<vendorName>String</vendorName>`
    `<vendorEmail>String</vendorEmail>`
    `<comments>String</comments>`
  `</requestInfo>`
`</requestCanceled>`
	Message from the task agent to the notification agent and user interface agent to indicate that a request has been canceled.

Table 3: XML Parameters from the Task Agent to Notification Agent

Parameter	Description

`<sendNotification>`
  `<task>`
    `<taskID>int</taskID>`
    `<requestInfo>`
      `<requestorName>String</requestorName>`
      `<requestorEmail>String</requestorEmail>`
      `<departmentName>String</departmentName>`
      `<softwareName>String</softwareName>`
      `<softwareCost>double</softwareCost>`
      `<numberOfUsers>int</numberOfUsers>`
      `<businessReason>String</businessReason>`
      `<informationType>int</informationType>`
      `<vendorName>String</vendorName>`
      `<vendorEmail>String</vendorEmail>`
      `<comments>String</comments>`
    `</requestInfo>`
    `<teamName>String</teamName>`
    `<teamContact>String</teamContact>`
    `<taskItems>`
      `<taskItem>`
        `<taskItemID>int</taskItemID>`
        `<taskItemName>String</taskItemName>`
        `<isComplete>b</isComplete>`
      `</taskItem>`
	Message from the task agent to the notification agent with a task to send a notification for.

. n . </taskItems> <isComplete>bool</isComplete> </task> </sendNotification>	
<requestComplete> <requestInfo> <requestorName>String</requestorName> <requestorEmail>String</requestorEmail> <departmentName>String</departmentName> <softwareName>String</softwareName> <softwareCost>double</softwareCost> <numberOfUsers>int</numberOfUsers> <businessReason>String</businessReason> <informationType>int</informationType> <vendorName>String</vendorName> <vendorEmail>String</vendorEmail> <comments>String</comments> </requestInfo> </requestComplete>	Message from the task agent to the notification agent to indicate that a request has been completed.

Table 5: XML Parameters from the Task Agent to Training Agent

Parameter	**Description**
<enableTraining> <traineeName>String</traineeName> <enableTraining>	Message from the task agent to the training agent that contains the name of the person that needs to complete the training

Table 5: XML Parameteres from the Task Agent to Scheduler Agent

Parameter	**Description**
<scheduleTaskReminder> <task> <taskID>int</taskID> <requestInfo> <requestorName>String</requestorName> <requestorEmail>String</requestorEmail> <departmentName>String</departmentName> <softwareName>String</softwareName> <softwareCost>double</softwareCost>	Message from the task agent to the scheduler agent to schedule a task reminder for the team assigned to the task

<numberOfUsers>int</numberOfUsers>
      <businessReason>String</businessReason>
      <informationType>int</informationType>
      <vendorName>String</vendorName>
      <vendorEmail>String</vendorEmail>
      <comments>String</comments>
    </requestInfo>
    <teamName>String</teamName>
    <teamContact>String</teamContact>
    <taskItems>
      <taskItem>
        <taskItemID>int</taskItemID>
        <taskItemName>String</taskItemName>
        <isComplete>b</isComplete>
      </taskItem>
      .
      n
      .
    </taskItems>
    <isComplete>bool</isComplete>
  </task>
</scheduleTaskReminder>	

<cancelTaskReminder> <taskID>int</taskID> </scheduleTaskReminder>	Message from the task agent to the scheduler agent to cancel a task reminder.

<scheduleSoftwareInstallation>
  <requestInfo>
    <requestorName>String</requestorName>
    <requestorEmail>String</requestorEmail>
    <departmentName>String</departmentName>
    <softwareName>String</softwareName>
    <softwareCost>double</softwareCost>
    <numberOfUsers>int</numberOfUsers>
    <businessReason>String</businessReason>
    <informationType>int</informationType>
    <vendorName>String</vendorName>
    <vendorEmail>String</vendorEmail>
    <comments>String</comments>
  </requestInfo>
</scheduleSoftwareInstallation>
	Message from the task agent to the scheduler agent to schedule the software installation.

Messages From the Scheduler Agent

Table 6: XML Parameters from the Scheduler Agent to Notification Agent

Parameter	Description

<sendTaskReminder>
  <task>
    <taskID>int</taskID>
    <requestInfo>
      <requestorName>String</requestorName>
      <requestorEmail>String</requestorEmail>
      <departmentName>String</departmentName>
      <softwareName>String</softwareName>
      <softwareCost>double</softwareCost>
      <numberOfUsers>int</numberOfUsers>
      <businessReason>String</businessReason>
      <informationType>int</informationType>
      <vendorName>String</vendorName>
      <vendorEmail>String</vendorEmail>
      <comments>String</comments>
    </requestInfo>
    <teamName>String</teamName>
    <teamContact>String</teamContact>
    <taskItems>
      <taskItem>
        <taskItemID>int</taskItemID>
        <taskItemName>String</taskItemName>
        <isComplete>b</isComplete>
      </taskItem>
      .
      n
      .
    </taskItems>
    <isComplete>bool</isComplete>
  </task>
</sendTaskReminder>
	Messages from the scheduler agent to the notification agent containing the information for tasks that reminder notifications must be sent for.

Messages From the Training Agent

Table 7: XML Parameters from the Training Agent to Task Agent

Parameter	Description
<requestorTrainingCompleted> <requestorName>String</requestorName>	Message from the training agent to the task agent

<trainingID>int</trainingID> <requestorTrainingCompleted>	containing the training ID and name of the requestor who completed the training.

Messages From the User Interface Agent

Table 8: XML Parameters from the User Interface Agent to the Task Agent

Parameter	**Description**
<userLogon> <userTeam>String</userTeam> </userLogon>	Message from the user interface agent to the task agent containing the team of the user that has logged on.

<taskItemComplete>
  <task>
    <taskID>int</taskID>
    <requestInfo>
      <requestorName>String</requestorName>
      <requestorEmail>String</requestorEmail>
      <departmentName>String</departmentName>
      <softwareName>String</softwareName>
      <softwareCost>double</softwareCost>
      <numberOfUsers>int</numberOfUsers>
      <businessReason>String</businessReason>
      <informationType>int</informationType>
      <vendorName>String</vendorName>
      <vendorEmail>String</vendorEmail>
      <comments>String</comments>
    </requestInfo>
    <teamName>String</teamName>
    <teamContact>String</teamContact>
    <taskItems>
      <taskItem>
        <taskItemID>int</taskItemID>
        <taskItemName>String</taskItemName>
        <isComplete>b</isComplete>
      </taskItem>
      .
      n
      .
    </taskItems>
    <isComplete>bool</isComplete>
  </task>
</taskItemComplete>
	Message from the user interface agent to the task agent containing a task that has been marked as complete by the user.

```
<denyRequest>
  <task>
    <taskID>int</taskID>
    <requestInfo>
      <requestorName>String</requestorName>
      <requestorEmail>String</requestorEmail>
      <departmentName>String</departmentName>
      <softwareName>String</softwareName>
      <softwareCost>double</softwareCost>
      <numberOfUsers>int</numberOfUsers>
      <businessReason>String</businessReason>
      <informationType>int</informationType>
      <vendorName>String</vendorName>
      <vendorEmail>String</vendorEmail>
      <comments>String</comments>
    </requestInfo>
    <teamName>String</teamName>
    <teamContact>String</teamContact>
    <taskItems>
      <taskItem>
        <taskItemID>int</taskItemID>
        <taskItemName>String</taskItemName>
        <isComplete>b</isComplete>
      </taskItem>
      .
      n
      .
    </taskItems>
    <isComplete>bool</isComplete>
  </task>
</denyRequest>
```

Message from the user interface agent to the task agent containing a task the user has denied the request for.