Disaster Recovery Model and Resource Tracking

INSTRUCTOR

Dr. Kwok-Bun Yue

MENTORS

Mr. Dilhar De Silva

Mr. Jim Fries

PROJECT TEAM MEMBERS – TEAM 6

Anthony Smith

Eric Matlock

Lance Hoang

SEMESTER

Fall 2010

Report Date: 30 November 2010

ASBTRACT:

After a natural or manmade disaster, there is an immediate need to begin recovery operations. The scene is chaotic as recovery workers try to organize the equipment and personnel necessary to accomplish the recovery tasks. Tracking equipment and having situational awareness of progress are key elements to a successful recovery.

While there are many different software packages that can track equipment or personnel, there is not a good package that can be used for managing the entire disaster recovery process to include equipment tracking. Our goal is to create a disaster recovery management model and implement an application prototype that will utilize a GPS device integrated with Google Maps to track a fleet of equipment during a disaster recovery operation. Our project is sponsored by Atlink Communications and the Gulf Coast Procurement Group.

The GPS data should be stored in a central database, which can be accessed from users in the field or personnel in a disaster recovery center. The users in the field should be able to input updates to resources they are responsible for managing. Personnel in a disaster recovery center should be able to update data and run reports on resources being managed.

This project shall follow the Rational Unified Process with one-week sprints. Each week the team will meet with the project mentor to present the project status and receive feedback. The project will be documented using UML 2.0.

Figures

Tables

1. Introduction

1.1 Purpose

This goal of this project is to develop a disaster recovery management model and implement an application prototype that will utilize a GPS device integrated with Google Maps to track a fleet of equipment during a disaster recovery operation.

1.2 Scope

The scope of this project is to lay the foundation for a disaster recovery tool that can be built upon. The UML documentation describes a broader model than the prototype implements. It is meant to give direction for future implementation.

1.3 Overview

After a natural or manmade disaster, there is an immediate need to begin recovery operations. Recovery workers begin the initial stages of recovery. The scene is chaotic as recovery workers try to organize the equipment and personnel necessary to accomplish the recovery tasks. Equipment is spread out over the disaster area as crews begin working. Tracking equipment and having situational awareness of progress are key elements to a successful recovery.

Many of the businesses that participate in disaster recovery efforts are small to medium sized businesses. While there are many different software packages that can track equipment or personnel, there is not a good package that can be used for managing the entire disaster recovery process to include equipment tracking. Our project created a disaster recovery management model and implemented an application prototype that utilizes a GPS device integrated with Google Maps to track a fleet of equipment during a disaster recovery operation.

2. Software Development Model

With the encouragement of the mentor the team followed the Rational Unified Process (RUP) during the development of this project. "The Rational Unified Process® is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget" [1]. Figure 1 below shows the phases of the RUP. The RUP is an adaptable iterative software development process that captures many of the best practices in modern software development [1]. To deal with the fast pace and short schedule of a capstone project the team included elements of the agile software development process scrum. Scrum is a software development process for small teams that breaks the project into a series of short development phases, or sprints [2].

The team used one-week sprints where goals were established at the mentor meeting and progress was evaluated the next week. Mr. De Silva and Dr. Yue provided feedback on the teams progress and gave suggestions on problem areas the team had. The use of one week sprints helped the team adapt to changing requirements and was critical to the success of the project. The team met up to three times a week during some phases to evaluate progress and make sure the project was heading in the right direction.

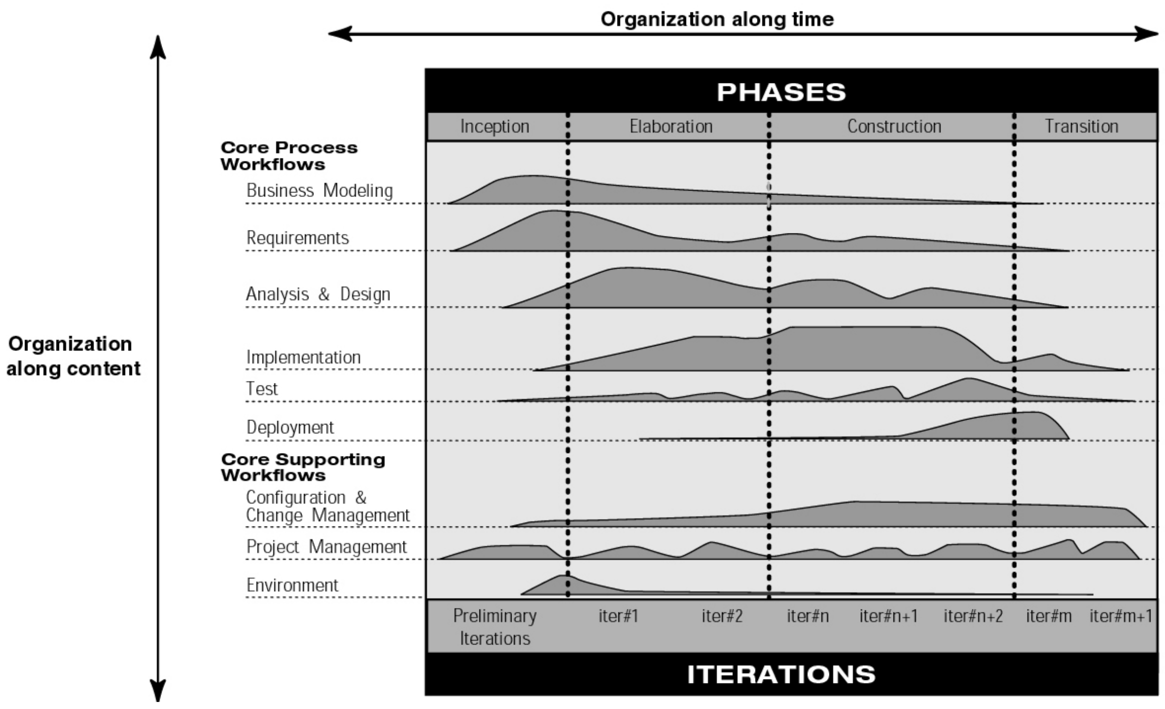


Figure 1 RUP Phases [1]

3. Design and Implementation

3.1 Inception

The team started the inception phase of the project by gathering requirements and creating use case diagrams to help understand the project. The team met with Mr. Dilhar De Silva and Mr. Jim Fries at the beginning of the project to gather requirements. Mr. Fries described what is involved in a disaster recovery process and the different people and processes that participate in the recovery. Early in the project the requirements changed. The project was originally to creating an application that would utilize a GPS device integrated with Google Maps to track a fleet of equipment during a disaster recovery operation. That requirement remained but the project focus was changed to create a generic disaster recovery management model and implement a prototype application that the GPS devices could report to. Once the team had a good understanding of the system requirements, we moved to the elaboration phase.

3.2 Elaboration

During the elaboration phase the team created the sequence diagrams and domain model. The sequence diagrams helped the team understand parts of the system that were still unclear. Mr. De Silva explained that if the sequence diagrams were too complex then that was an indication that the system design needed to be revisited. Once the team improved the design the sequence diagrams were much cleaner. The team spent more time developing a system model than on the implementation. The mentors wanted to make sure there was a good system model before the team developed the prototype.

The team chose to use the model-view-controller (MVC) architectural pattern for the project. Figure 2 shows what technology we used for each layer. The application consists of a presentation layer for the view, a service layer for the controller, and a data access layer for the model. The presentation layer was implemented in Adobe Flex; the service layer was implemented in Java; and the data access layer was implemented in Hibernate. The advantage to using this architectural pattern is that by separating the program into three layers the design is scalable and can be modified easily.

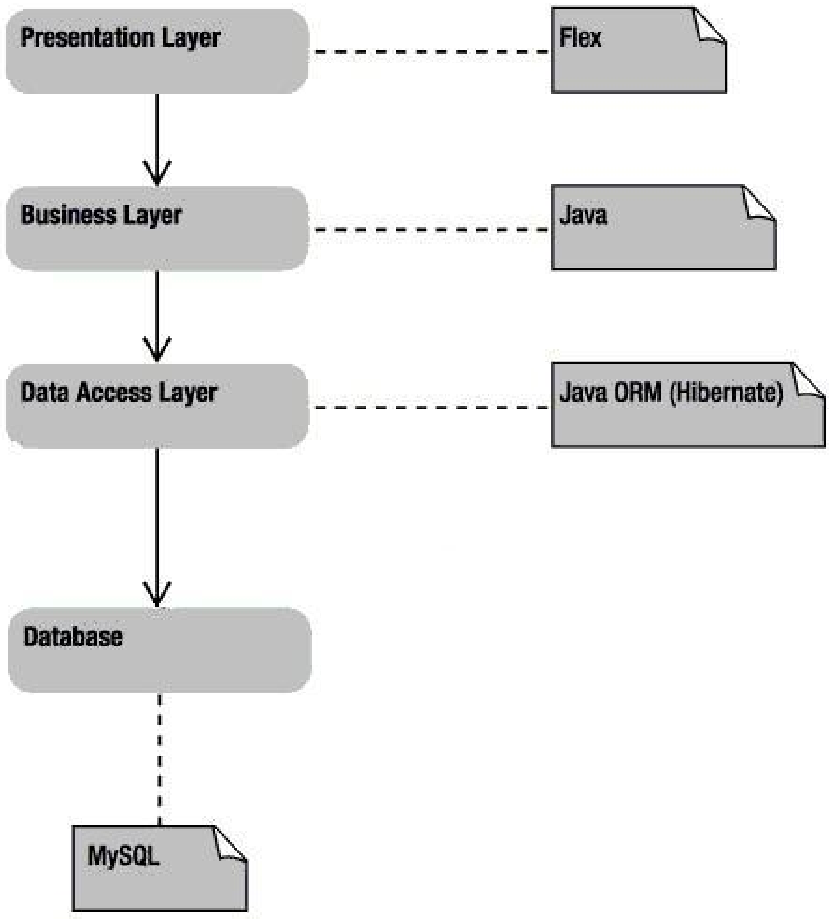


Figure 2 System Architecture

3.3 Construction

The construction phase was more difficult than the team originally expected. The team was introduced to a lot of technology that they had not used before. The team did not have any experience with Adobe Flex, Google maps, iPhone development or JBoss. To reduce the learning curve each team member was given primary responsibility for different technology so the whole team did not have to learn everything from scratch. Once a team member learned how to setup and use a technology they were responsible for they would share with the rest of the team what they needed to know. Each team member had a computer setup with everything needed to develop and run the project. The team met one weekend to help each other setup the development environment on their computer so that we all had the same development platform. We did not have a central repository for the code so we divided the work so that two people were not working on the same code at the same time. The team exchanged code through email to keep everyone updated. This is not an effective method for most projects, but it worked for this short project.

3.4 Transition

During the transition phase the team performed system level tests to verify the project me the requirements. The prototype was demonstrated to Mr. Dilhar De Silva and Mr. Jim Fries to validate that the project met the end users' expectations.

4. System Details

The system has five main subsystems that had to be developed.

1. Manage Disasters
2. Manage Resources
3. Manage Tasks
4. Reporting
5. Track GPS device location

4.1 Manage Disasters

The disaster manager component is responsible for maintaining high level information about disasters and managing all the contracts that are assigned to a disaster. A person can quickly see what contracts are still open for a disaster and which contracts have already been closed. This can be updated from any location with internet access. The contracts have expected end dates so it can be determined when a contract is supposed to be complete.

4.2 Manage Resources

The resource manager allows a contractor to manage personnel and equipment. Personnel can be added into the system and their contact information is maintained with the resource manager. Equipment can be added to the system and information such as who owns it and the lease rate are maintained by the resource manager.

4.3 Manage Tasks

The task manager is the main component of the system. The task manager is where tasks are created and assigned to disasters and contracts. Once a task is created personnel and equipment can be assigned to a task. This allows a contractor to quickly find out who is assigned to each task and what equipment is in use for each task.

4.4 Reporting

The reporting component works with the other components to provide a view of open tasks. This provides real time information of the status of open tasks that can be sorted in many different ways to give the users information they need to determine recovery progress. For example the tasks can be sorted to show which tasks are expected to be finished soon; which tasks have the highest priority; or which tasks have the most complete.

The reporting component also reports usage of equipment by displaying the tracked location of equipment with GPS devices assigned. This allows a contractor or someone in a disaster recovery center to find out the current location of equipment or to run historical reports to determine where the equipment has been. The user simply has to enter the date range they are interested in and the item they with to track. The reports are displayed on a Google maps layout which shows the locations of the equipment during the date range selected.

4.5 Track GPS device location

The GPS device used in this project was an iPhone. The team wrote an iPhone application that used the phones built-in GPS device to determine current location. The iPhone Application simulates an attached GPS Tracking Device. The application uses the CoreLocation Framework of the iOS SDK to detect whenever there is a change in the GPS location of the device. Once the phone detected the location had changed it allowed the used to transmit the coordinates to the server by updating the Button in the UI with the text "New Location is Available". The user presses this button, causing the application to send the current Longitude, Latitude, Time, TimeZone, and Date to the Server. The database stores the location of the device along with the time and date.

5. Technical Challenges and Lessons Learned

5.1 System Architecture

One of the first challenges during the design phase was determining if the system should be a two tier or three tier system. Because this is expected to be a system with many GPS devices reporting and multiple users querying the system it was decided to use a three tier architecture. The team decided to use JBoss as the application server because it is a well documented and popular open source server.

5.2 GPS Selection

Another challenge the team had was determining what type of GPS device to use for the prototype. The team researched different GPS devices to determine which would be the best. The team determined that a satellite based GPS device would be the most reliable for the application but the team did not have the equipment to use for the prototype. Most GPS devices use cell phone service to report location data to a server. Since there may not be reliable cell phone service after a disaster the tracking of equipment would not be reliable with a device that uses cell phone service. The satellite based GPS devices are the most reliable in this situation, but they are also more expensive. For this project the team decided to use the iPhone to simulate an attached GPS Tracking Device. The iPhone has a built-in GPS device which can determine current location.

5.3 iPhone development

The team discovered many challenges with developing an iPhone application. The first was that to develop an iPhone application you must use xcode on Mac OS. One team member had a Mac computer so the team was able to begin development. The next challenge was learning Objective C. All iPhone applications are developed in Objective C so the team had to learn the language. The last challenge was in deploying the application to the iPhone. A developer subscription is required to deploy application to iPhones. One team member purchased a development license so the application could be deployed to an iPhone.

5.4 JBoss and Hibernate

The team needed a scalable architecture that could be easily modified if necessary. The team decided to use JBoss as the application server and Hibernate as the object-relational mapping (ORM) tool. JBoss can easily be scaled to meet the needs of many devices reporting and multiple users querying the system. It also gives a layer of abstraction for managing database connections. The JBoss server can be setup to manage the connection pool for the database so that the application does not have to deal with it. It also contains the libraries for Hibernate. Hibernate provides another layer of abstraction to the database. This allowed the Disaster Management application to be written using Hibernate queries instead of a database specific query. The team used MySQL as the database for the project, but that could easily be changed now if the customer decided to use another database. Only a few mapping files would have to be changed to use the new database instead of changing the application.

6. Conclusions

Based on the feedback from the user demonstrations the team considers this a successful project. The prototype did implement all of the required functionality and the demonstrations showed that the system works. The team enjoyed working with the mentors and getting the opportunity to learn from this short project. There were many challenges that the team overcame while developing this project. Learning all the new technology and providing a working prototype in the short aggressive schedule was challenging. The only way it was possible to succeed was with good teamwork and timely feedback from mentors.

6.1 Future Work

There is still plenty of work to be done to take this prototype and build a robust system for customer use. This project focused on developing the model and framework for future development. The data validation needs to be improved and a security login module needs to be added. The current application assumes that a security module will allow the user to login to the application and pass the credentials to the application.

7. References

[1] IBM Rational Software. "Rational Unified Process : Best Practices for Software Development Teams." *IBM DeveloperWorks: Rational*. January 10, 2003. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[2] L. Rising, N. Janoff, "The Scrum Software Development Process for Small Teams." *Softw IEEE*, vol 17, issue 4, pp. 26-32, Jul/Aug 2000.

Appendix A : Project Management and Team Information

Since we had a small team of three people, each person contributed to almost every role. We worked together and helped team members who had to travel or became busy at work so there was not a single point of failure. We had frequent team meetings and communicated status of tasks through email. Each team member contributed to the architecture and design and implementation tasks were divided amount team members as primary and secondary roles.

Anthony Smith – Roles: Team leader, Flex and Google Maps, iPhone developer

Eric Matlock – Roles: MySQL developer, Java developer, Technical writer, iPhone developer

Lance Hoang – Roles: Webmaster, MySQL developer, Java developer, Flex and Google Maps

Table 1 Project Schedule

Phase Name	Task Name	Start date	End Date
Inception		23-Aug-10	27-Sep-10
	Gather Requirements		
	Create Use Case Diagrams		
	Scope Requirements		
	Create Risk List		
Elaboration		27-Sep-10	22-Oct-10
	Create Class Diagrams (Domain Model)		
	Create System Diagrams		
	Create Sequence Diagrams		
Construction		22-Oct-10	19-Nov-10
	Create/Simulate GPS Device		
	Finalize Database Schema		
	Create Web UI		
	Create Client Application		
	Create Reports(includes Google Map View)		
Transition		19-Nov-10	30-Nov-10
	Testing		

Appendix B: Major tasks and contributions

Table 2 Task Contributions

Task Name	Primary	Secondary
Gather Requirements	All	All
Create Use Case Diagrams	Anthony	Lance, Eric
Scope Requirements	Anthony	Lance, Eric
Create Risk List	Eric	Anthony, Lance
Create Class Diagrams (Domain Model)	Anthony	Lance
Create System Diagrams	Eric	Lance
Create Sequence Diagrams	Anthony	Lance
Stubbed out Java Class Implementation	Anthony	Lance
Designate Integration Machine	All	All
Finalize Infrastructure	Eric	Lance
Finalize Database Schema	Lance	Eric
Complete Prototype UI(Forms-based) with all interaction functional	Lance	
Fully Functional Domain Model	Lance	Anthony
Install Reference Infrastructure on Development Machines	All	All
iPhone Hello World Running	Anthony	Eric
Google Maps using location's list	Lance	
Technical Report Draft	Eric	Lance, Anthony
iPhone Location Application Implemented	Anthony	Eric
Incorporate Tour de Flex concepts into UI Workflow	Lance	Anthony
Technical Report Final	Eric	Lance, Anthony
Tools Setup	Eric	Lance
Create Reports(includes Google Map View)	Lance	Eric
Testing	All	All

Appendix C: Software Requirements

1	The system shall manage resources
1.1	Resources shall be checked out by primary contractors
1.2	Resources shall be checked in by primary contractors
1.3	Resources shall be checked out by sub-contractors
1.4	Resources shall be checked in by sub-contractors
1.5	The resource manager shall verify that a resource can be checked out
1.6	The resource manager shall verify that a resource can be checked in
1.7	The resource manager shall calculate usage on a resource when polled by user
1.7.1	The usage of a resource shall be calculated and displayed as a trip on google maps based on a time period selected by user
1.7.2	The GPS location of a resource shall be accessable to the resource manager
2	The system shall manage tasks
2.1	The system shall provide the capability to add a task
2.2	The system shall provide the capability to update a task
2.3	The system shall provide the capability to assign task to contractors
2.4	The system shall provide the capability to delete a task
3	The system shall have a scalable architecture
3.1	The system shall use MySQL as the RDBM
3.2	The system shall use Flex for the web interface
3.3	The system shall use Google Maps to display usage reports
3.4	The system shall use Jboss as the application server
4	The system shall track resources
4.1	The system shall use an iPhone to report GPS location
4.1.1	The iPhone app shall report location to the application server when user commands; The system should report automatically at a user selected time interval of 5 min to 30 days
4.2	The iPhone app should store location data for 30 days
5	User Interface
5.1	Users shall be able to update resouces
5.2	Users shall be able to update tasks
5.3	Users shall be able to run reports on resource location
5.4	Users shall be able to run reports on resource usage

Appendix D: UML Diagrams

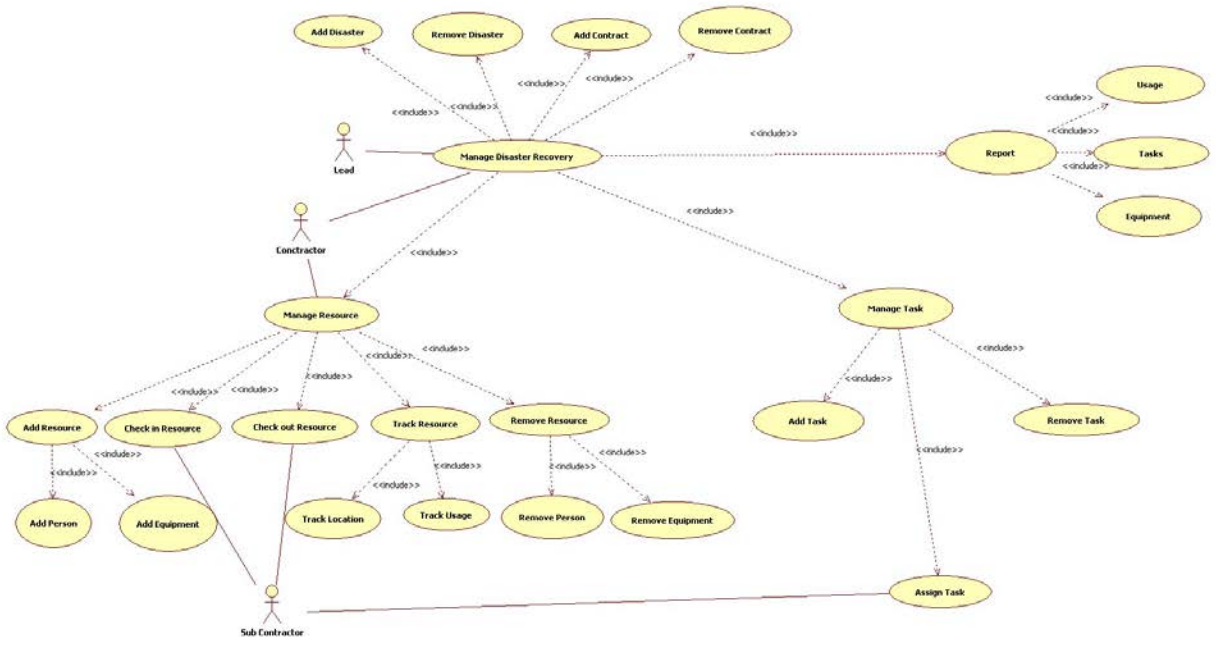


Figure 3 Use Case

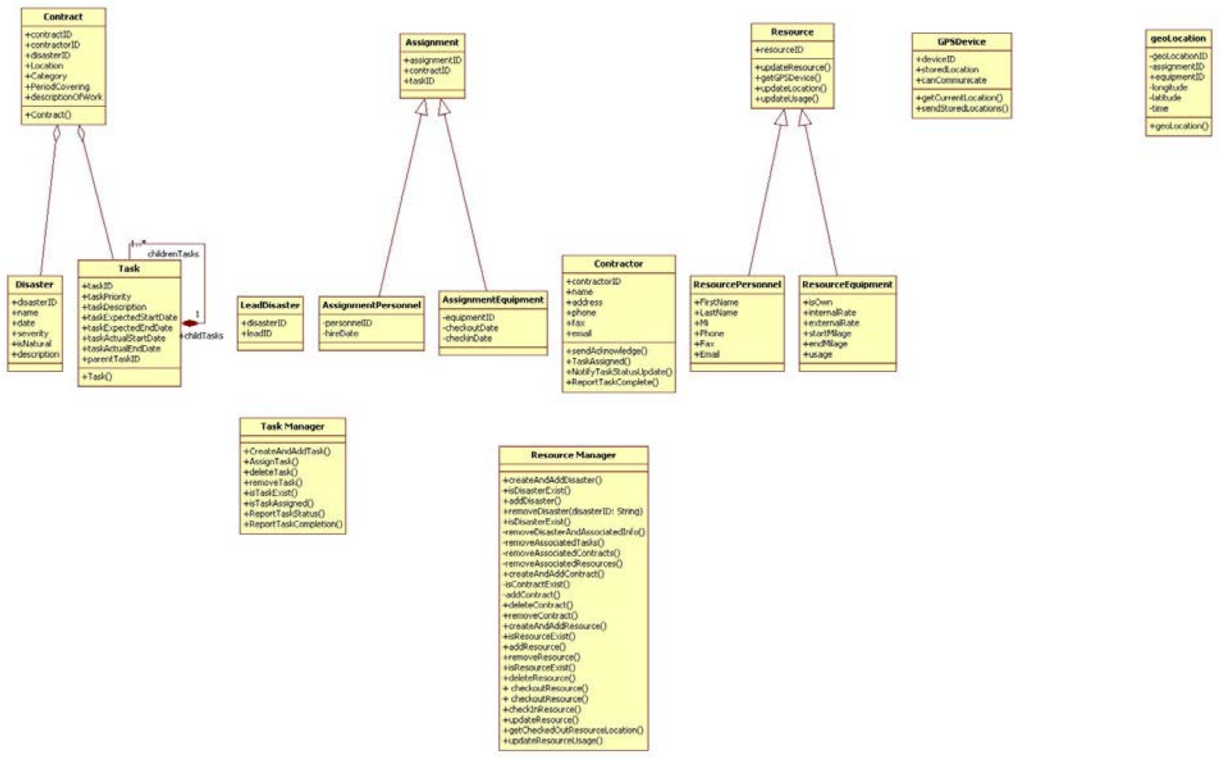


Figure 4 Class Diagram

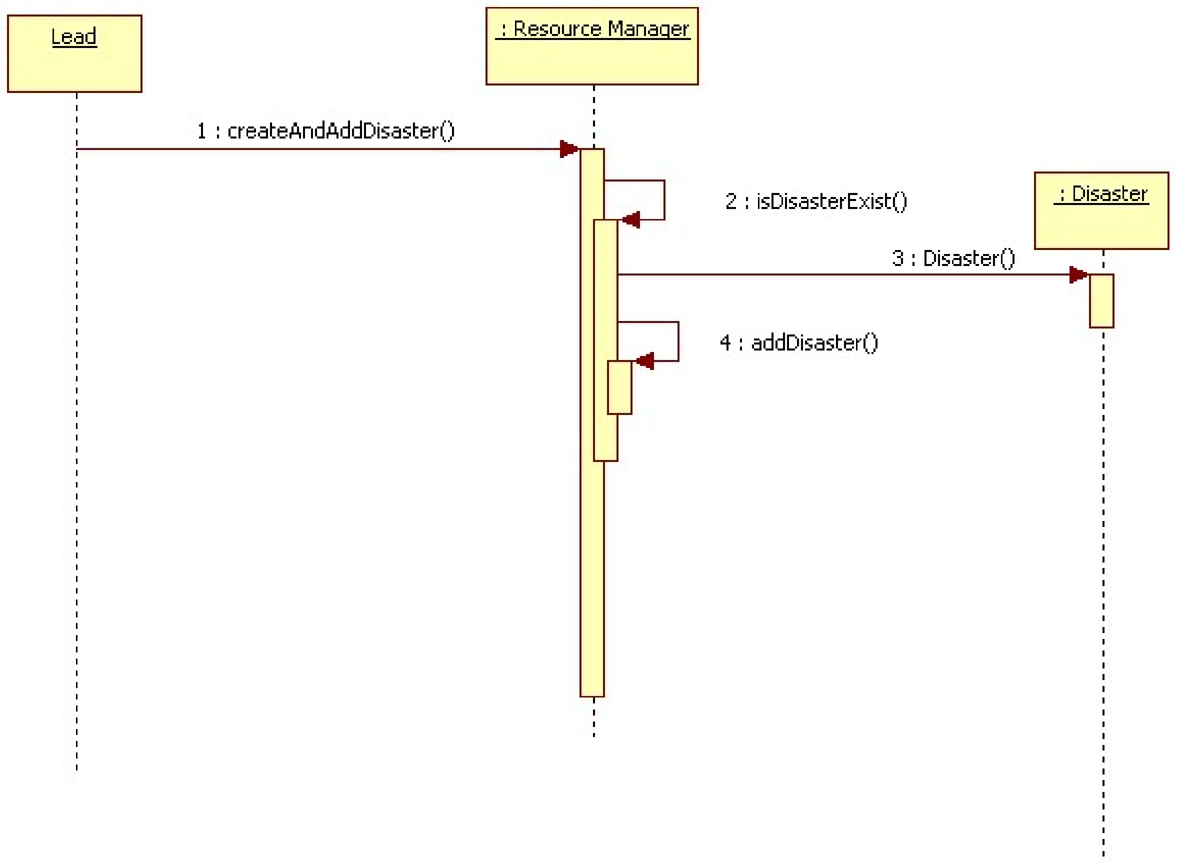


Figure 5 Add Disaster Sequence Diagram

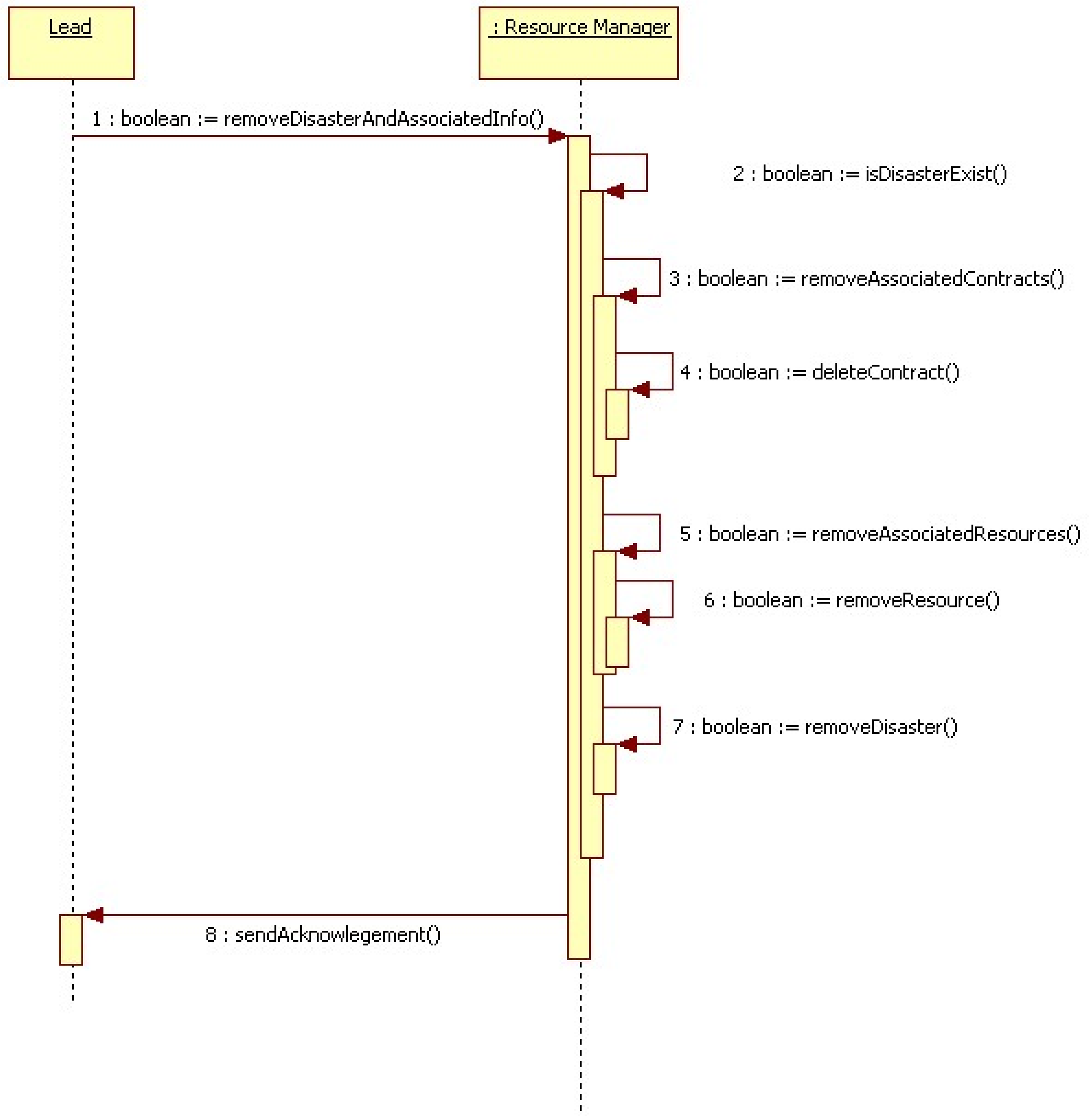

Lead
: Resource Manager
1 : boolean := removeDisasterAndAssociatedInfo()
2 : boolean := isDisasterExist()
3 : boolean := removeAssociatedContracts()
4 : boolean := deleteContract()
5 : boolean := removeAssociatedResources()
6 : boolean := removeResource()
7 : boolean := removeDisaster()
8 : sendAcknowlegement()

Figure 6 Remove Disaster Sequence Diagram

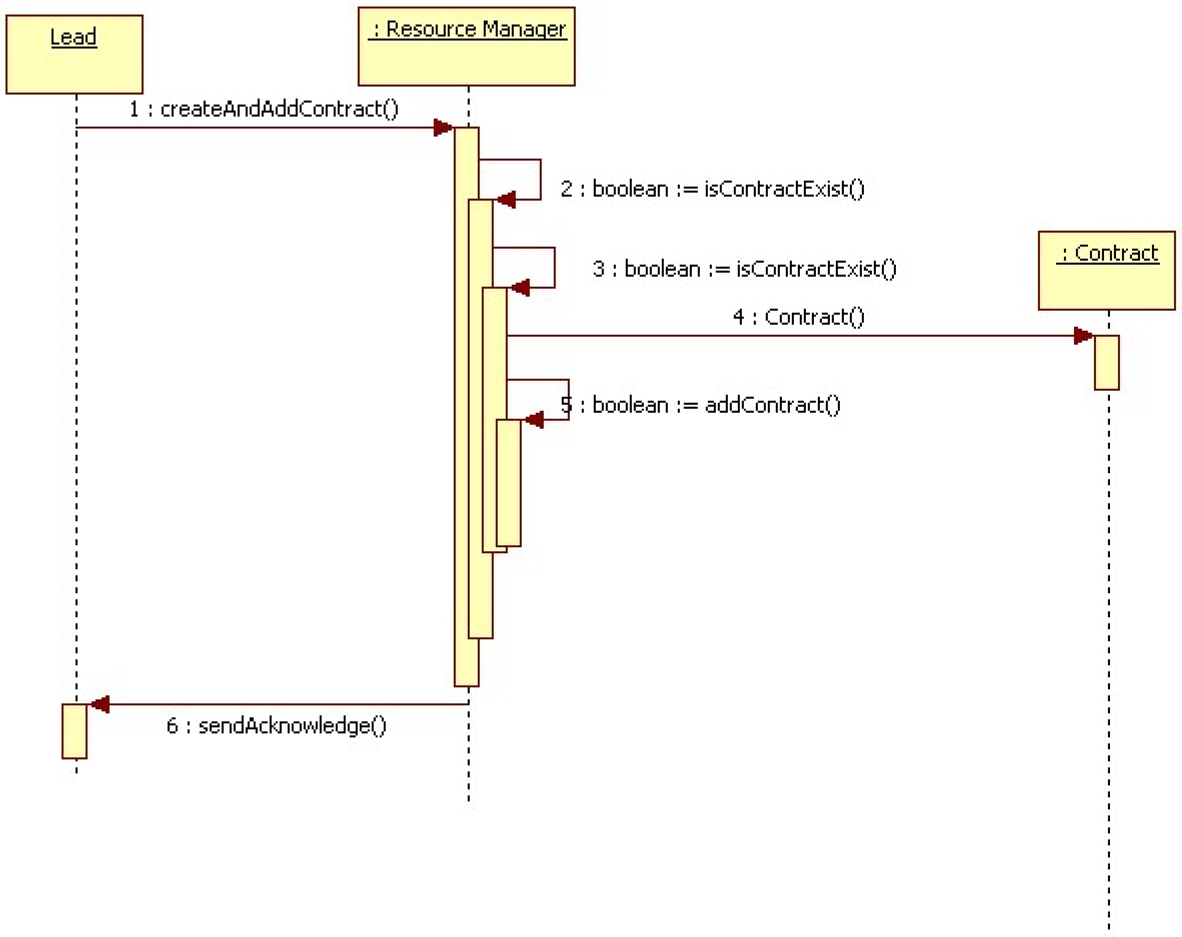

Lead
: Resource Manager
: Contract
1 : createAndAddContract()
2 : boolean := isContractExist()
3 : boolean := isContractExist()
4 : Contract()
5 : boolean := addContract()
6 : sendAcknowledge()

Figure 7 Add Contract Sequence Diagram

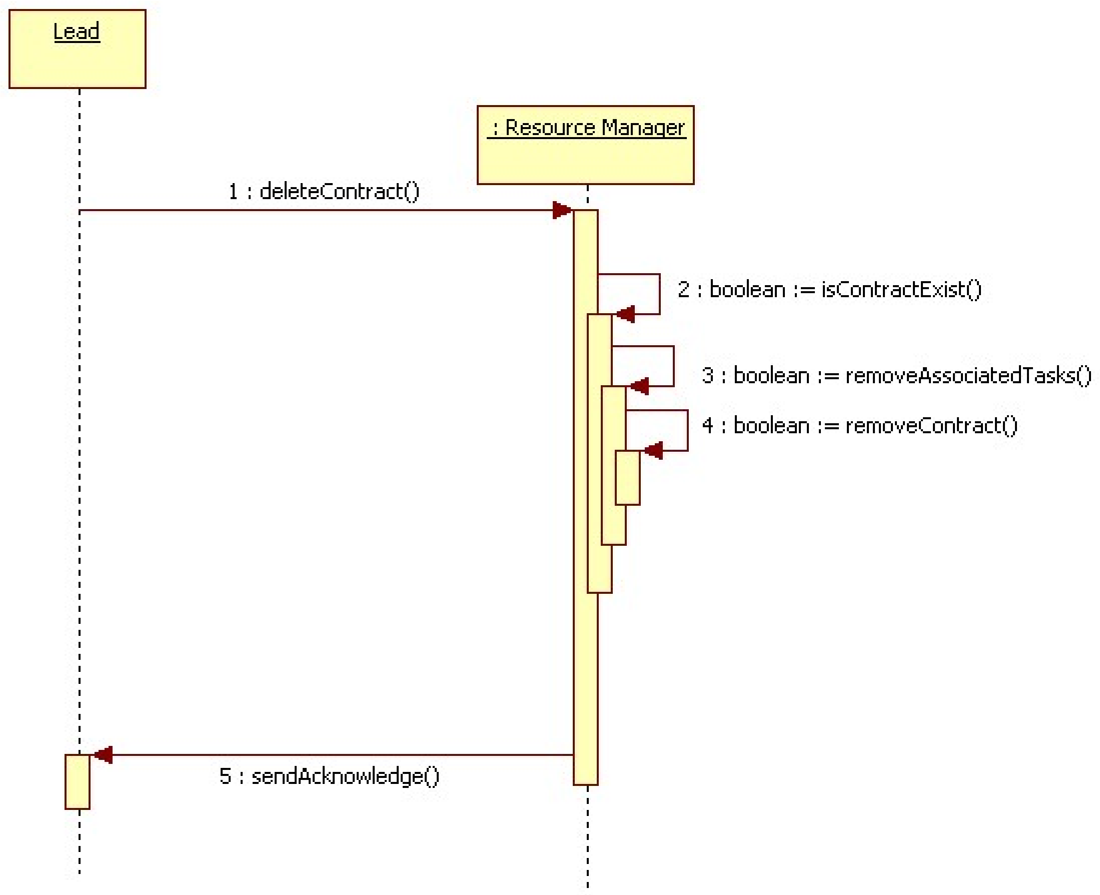
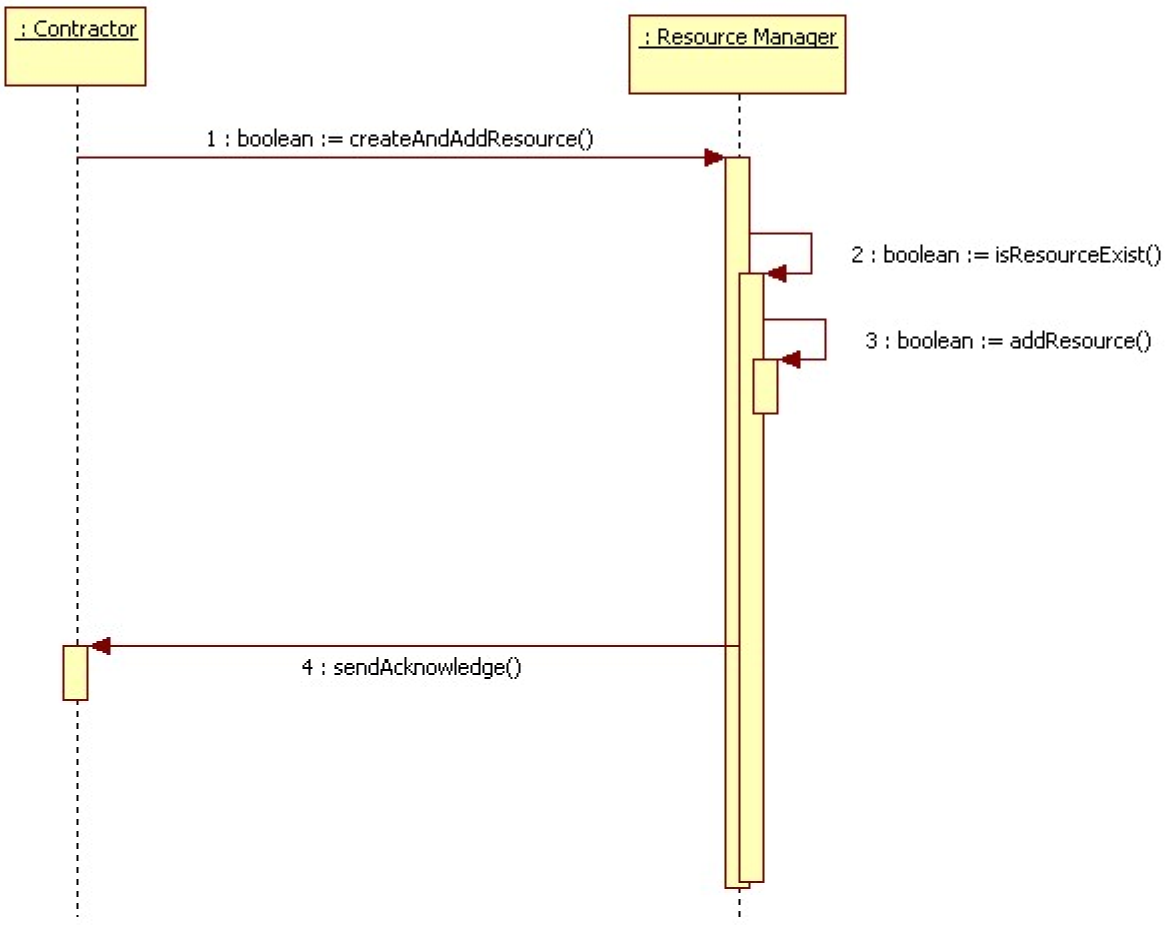

Lead
: Resource Manager
1 : deleteContract()
2 : boolean := isContractExist()
3 : boolean := removeAssociatedTasks()
4 : boolean := removeContract()
5 : sendAcknowledge()

Figure 8 Remove Contract Sequence Diagram



: Contractor
: Resource Manager
1 : boolean := createAndAddResource()
2 : boolean := isResourceExist()
3 : boolean := addResource()
4 : sendAcknowledge()

Figure 9 Add Resource Sequence Diagram

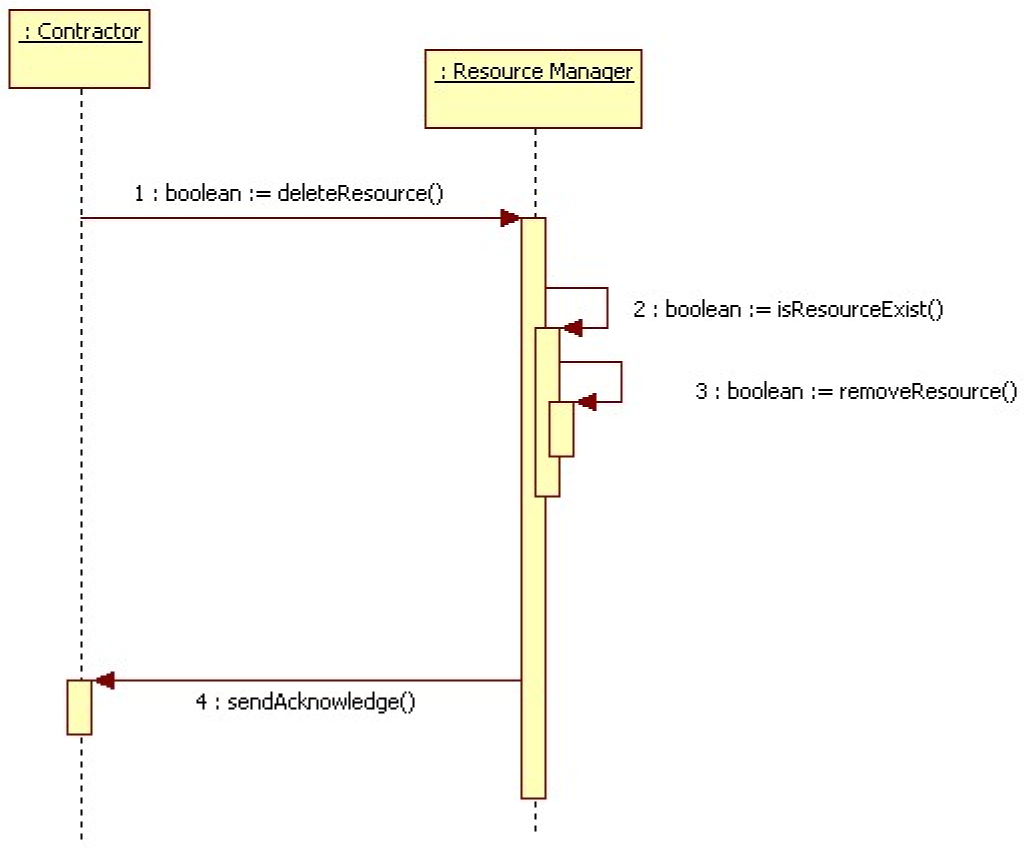

: Contractor
: Resource Manager
1 : boolean := deleteResource()
2 : boolean := isResourceExist()
3 : boolean := removeResource()
4 : sendAcknowledge()

Figure 10 Remove Resource Sequence Diagram

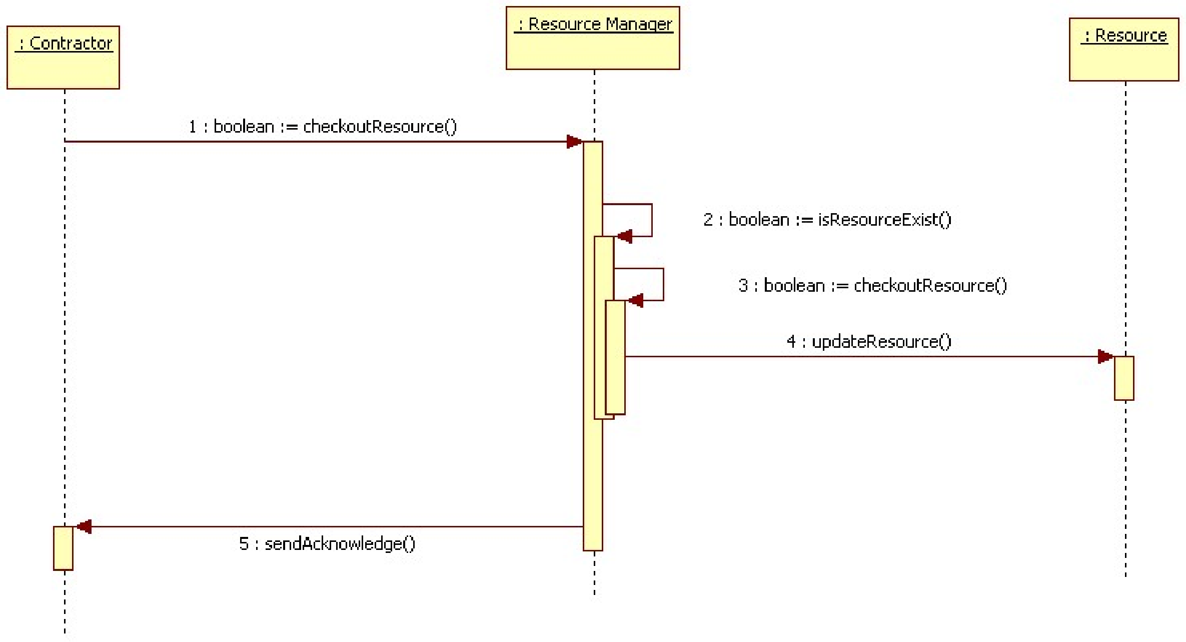

: Contractor
: Resource Manager
: Resource
1 : boolean := checkoutResource()
2 : boolean := isResourceExist()
3 : boolean := checkoutResource()
4 : updateResource()
5 : sendAcknowledge()

Figure 11 Checkout Resource Sequence Diagram

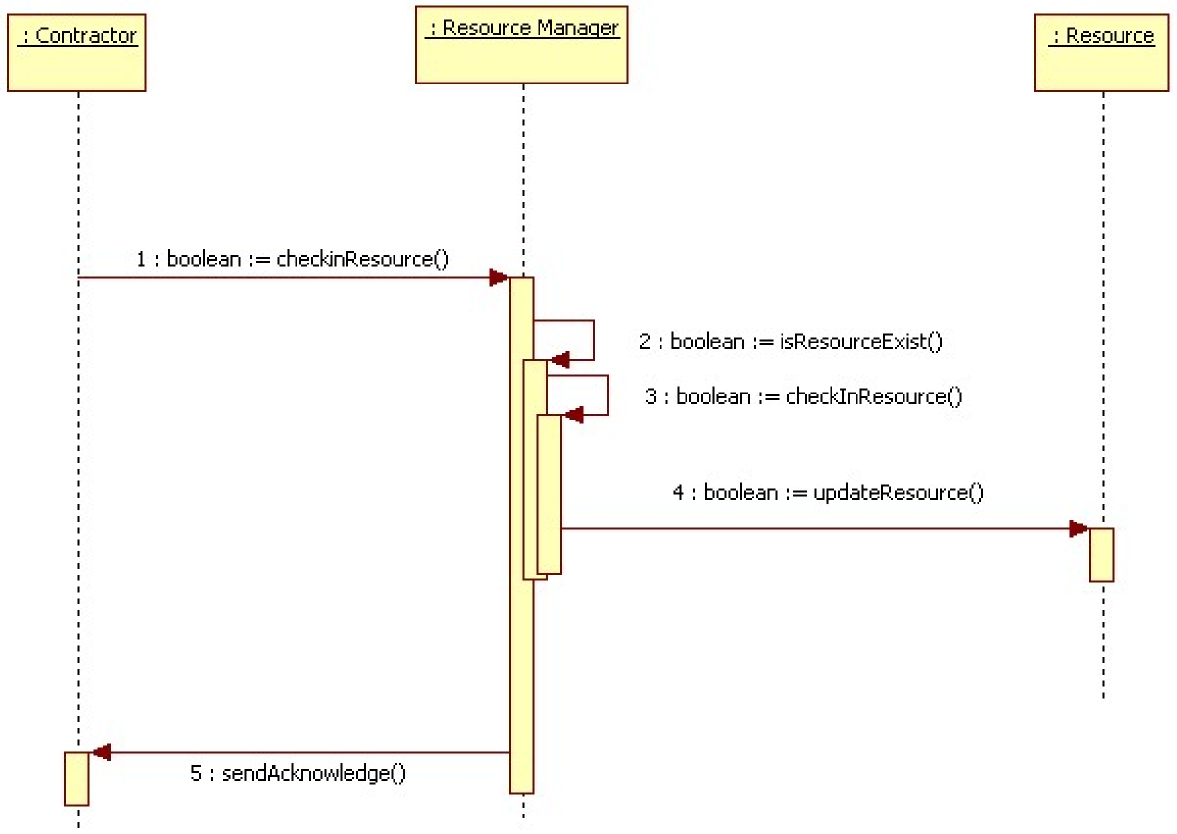

: Contractor
: Resource Manager
: Resource
1 : boolean := checkinResource()
2 : boolean := isResourceExist()
3 : boolean := checkInResource()
4 : boolean := updateResource()
5 : sendAcknowledge()

Figure 12 Check in Resource Sequence Diagram

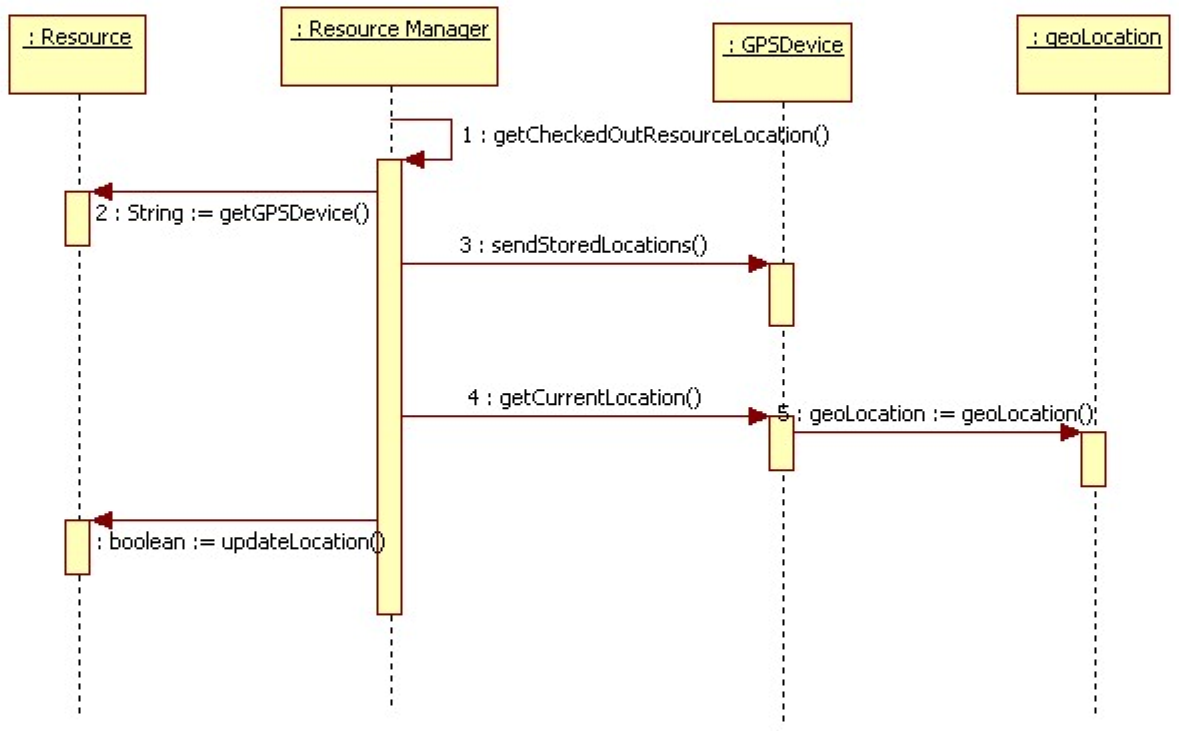

: Resource
: Resource Manager
: GPSDevice
: geoLocation
1 : getCheckedOutResourceLocation()
2 : String := getGPSDevice()
3 : sendStoredLocations()
4 : getCurrentLocation()
5 : geoLocation := geoLocation()
: boolean := updateLocation()

Figure 13 Resource Location Sequence Diagram

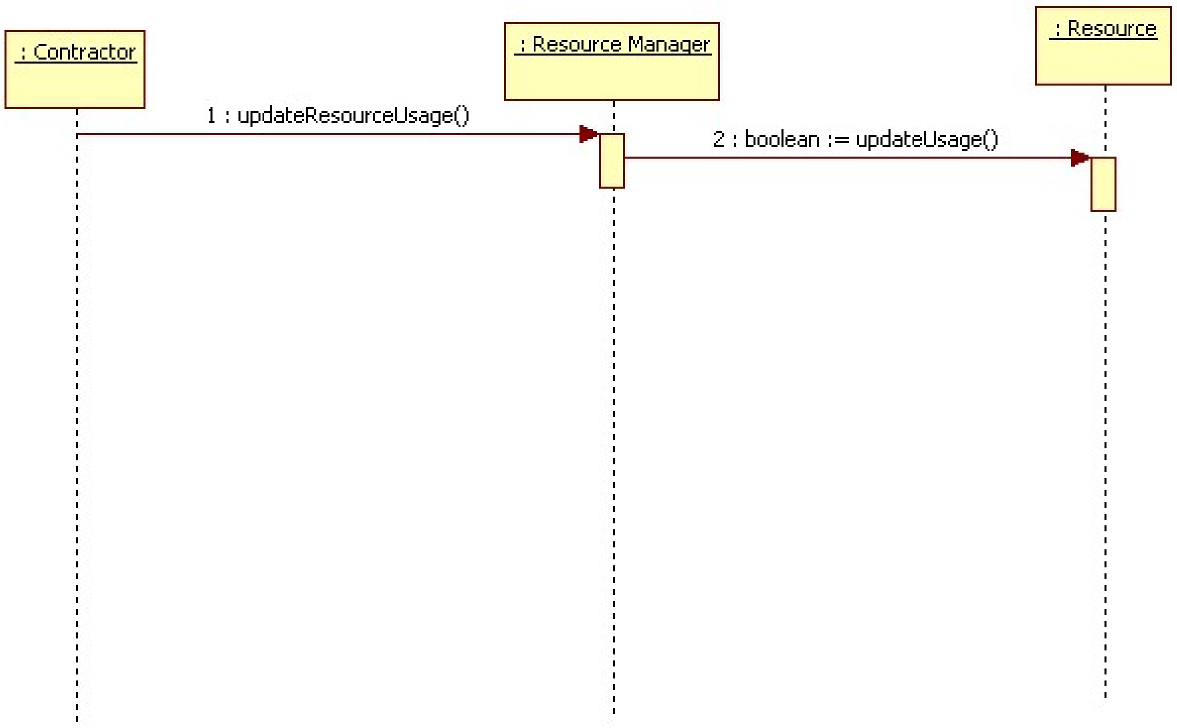


Figure 14 Resource Location Sequence Diagram

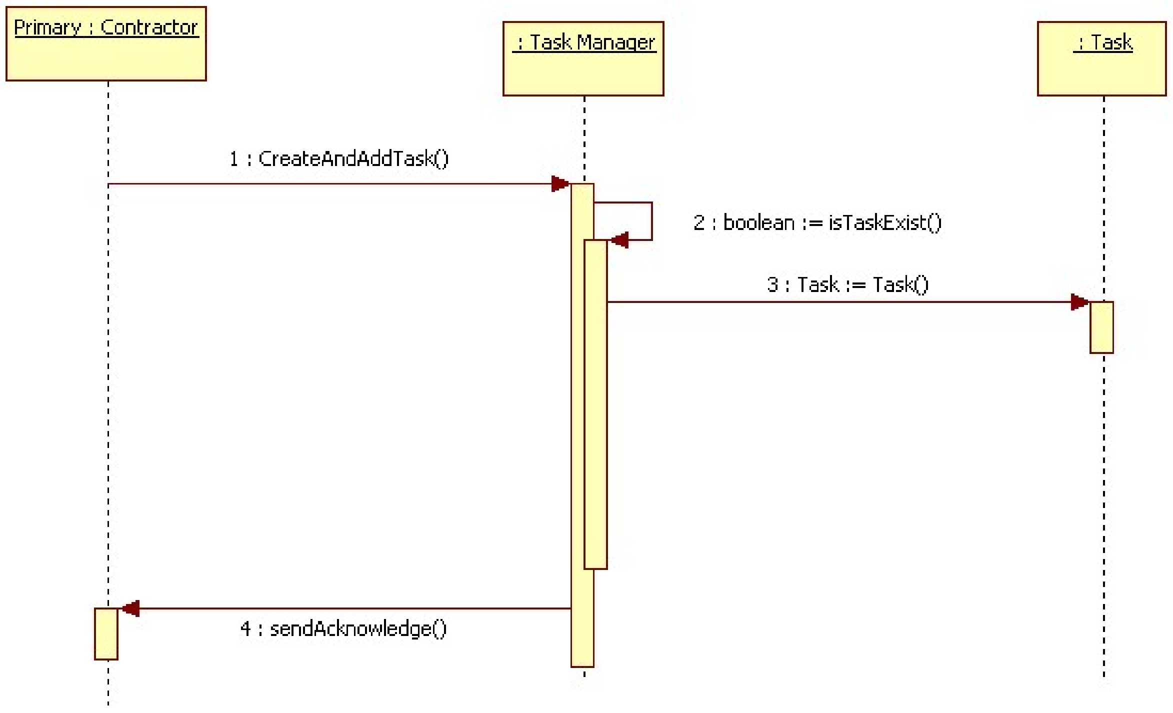


Figure 15 Create and Add Task Sequence Diagram

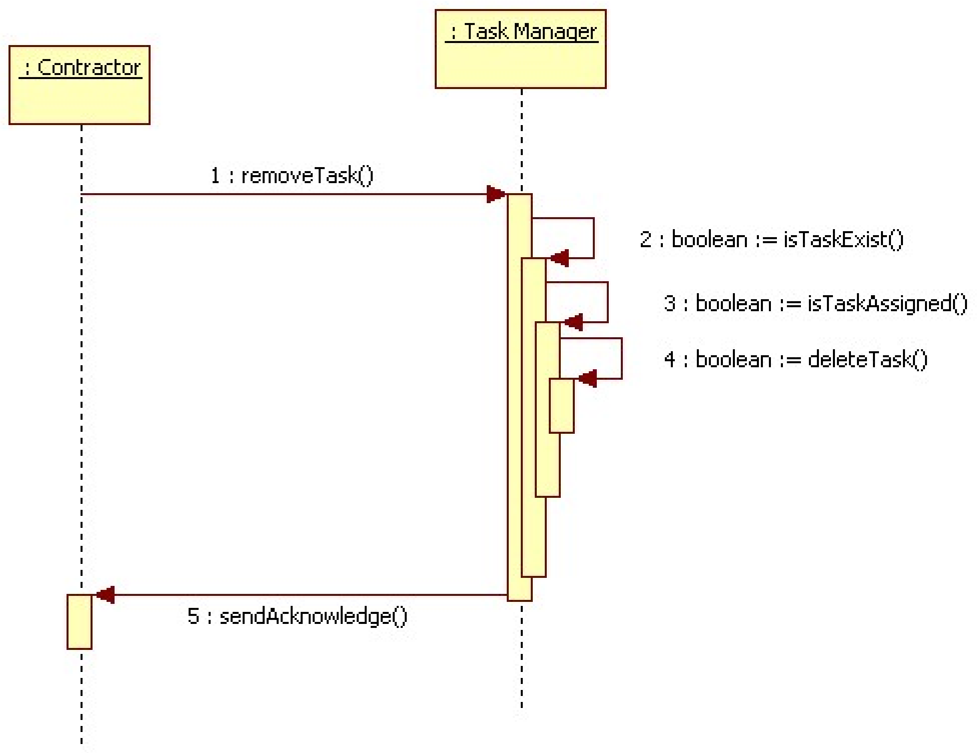

: Contractor
: Task Manager
1 : removeTask()
2 : boolean := isTaskExist()
3 : boolean := isTaskAssigned()
4 : boolean := deleteTask()
5 : sendAcknowledge()

Figure 16 Remove Task Sequence Diagram

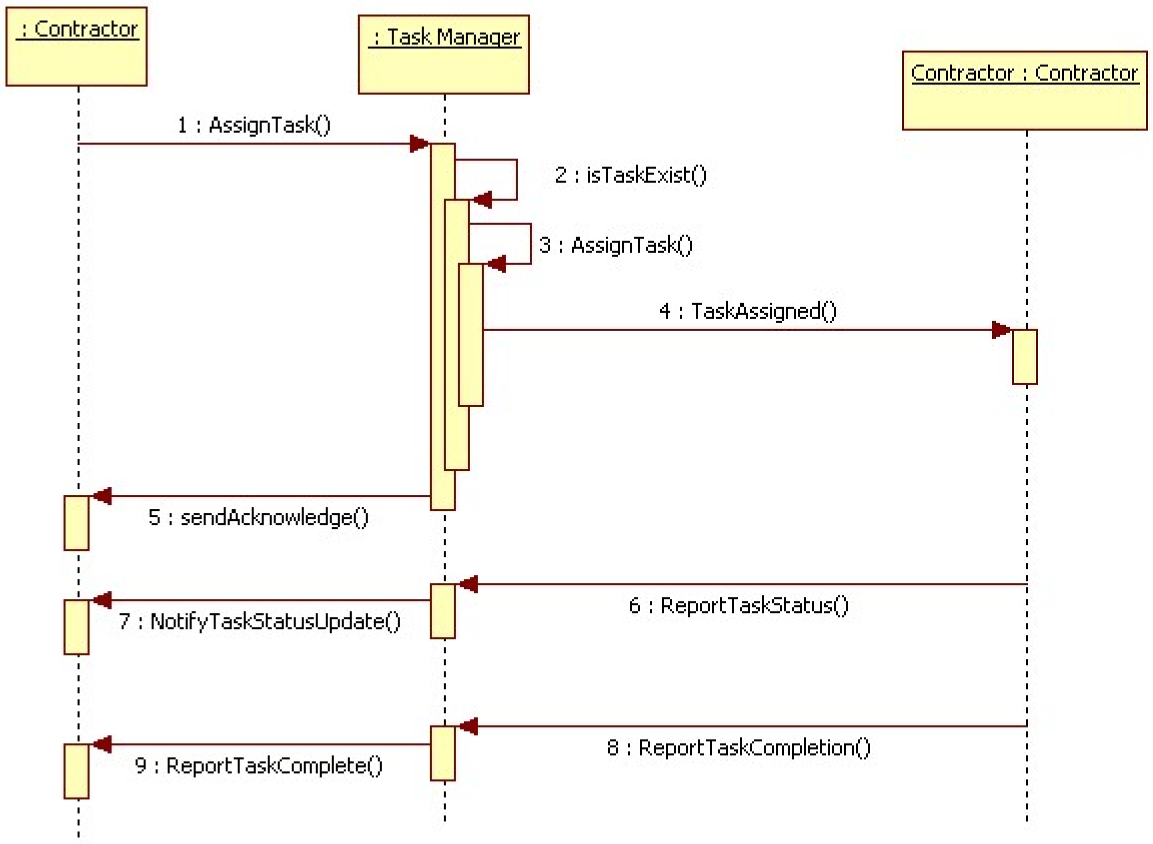


Figure 17 Assign Task Sequence Diagram