Chris Moorhead

cam0071@auburn.edu

Homework 5

1. For each of the following systems list the applicable structural and control style(s) of system architecture covered in class (e.g., repository, client-server, service-oriented architecture, two-tier / three-tier / four-tier client server, abstract machine model, centralized, event-driven). Justify your answers.

a. An electronic chess companion

A good control style for this is **event-driven control** since the companion will respond to different moves in different ways. This could work as an *interrupt-driven model* since the player moves cause the interrupts and then the move data gets passed on for processing.

An **abstract machine model** is a good structural style for the system. The companion takes player moves as input and has its own moves as output, as well as an internal set of rules maintaining legal play.

b. Floppy disk controller chip

A **centralized control model** is a good fit since there should be a control system (ie. The program on the computer accessing the floppy) that is using the functionality of the floppy. This is a *call-return model* since it follows a top down approach (root access).

Since there is an explicit data storage schema in place for floppy disks, a **repository model** can be used as its architectural style. This can be used to set up central data storage rules for efficient data access and manipulation.

c. Airplane flight simulator for a video game

A **centralized control model** such as the *manager model* would be a great choice for the system. Let the manager get direct feedback from the user controls. Then, the manager controls the system – roll, pitch, yaw, throttle, etc. This is a centralized, concurrent system.

A **client server model** is a good architectural style for the system. Specifically, a 3-tier approach where presentation (UI & graphics), logic (input, responses) and data (how input changes over time to adjust responses) are separate.

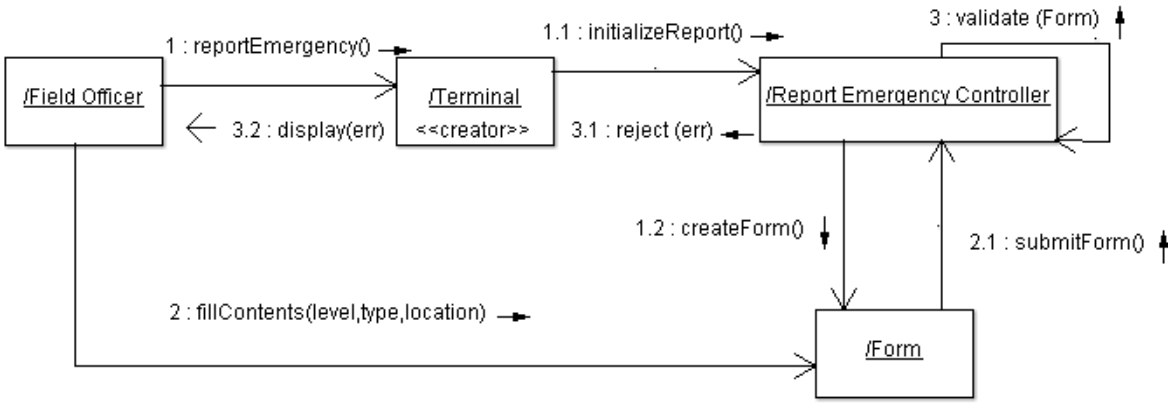
d. Sonar system

The system is a good example for the *broadcast model* of an **event-driven control** system. The system will send a signal and then listen to analyze distances and objects. Different subsystems (ie. Signal processing) will listen for specific events to begin their subroutines.

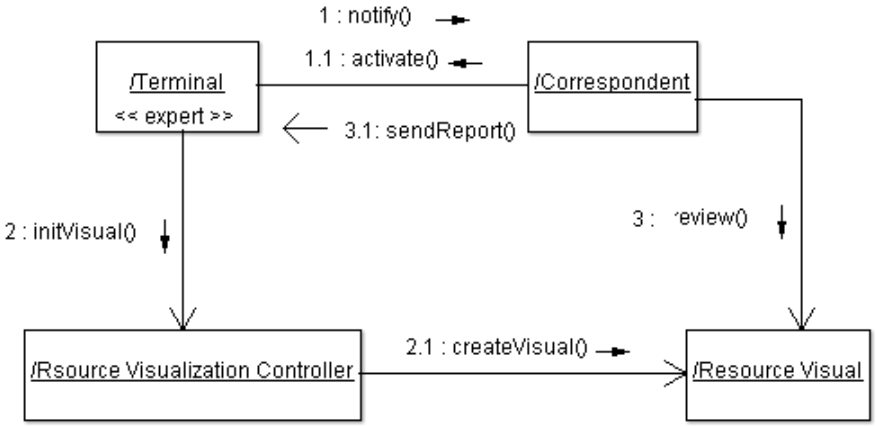
The sonar system is also a good system for a **service-oriented** architecture. It is a self-contained, black box for most of its users. Further, the subsystems of the application all interact to gather, read, and express data.

2. In a situational awareness system officers such as police officer or fire fighter, have access to a wireless computer that enables them to interact with a correspondent. The correspondent can in turn visualize the current status of all its resources, such as police cars or trucks and call a resource by issuing commands from a workstation. You should create one interaction diagram (i.e. collaboration diagram) for each distinct event in the scenarios.

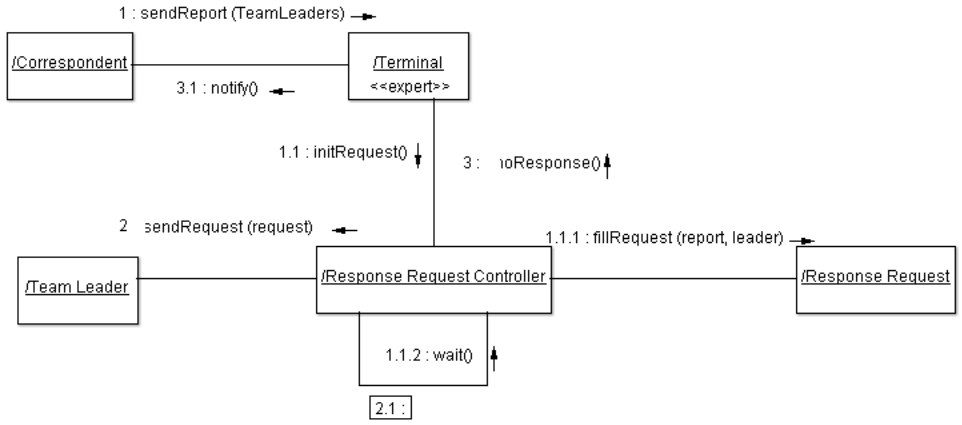
a. Scenario 1, Event: Emergency Report Submitted



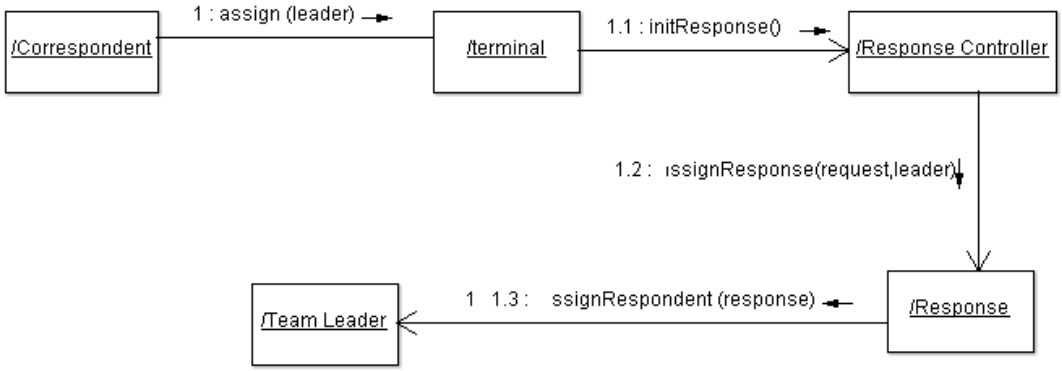
b. Scenario 2, Event: Submission Info Sent to Available Team Leaders



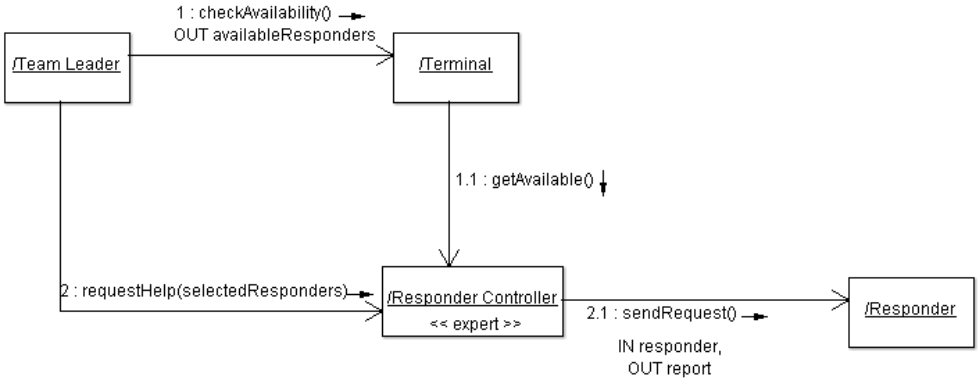
c. Scenario 2, Event: Timer expires; correspondent notified



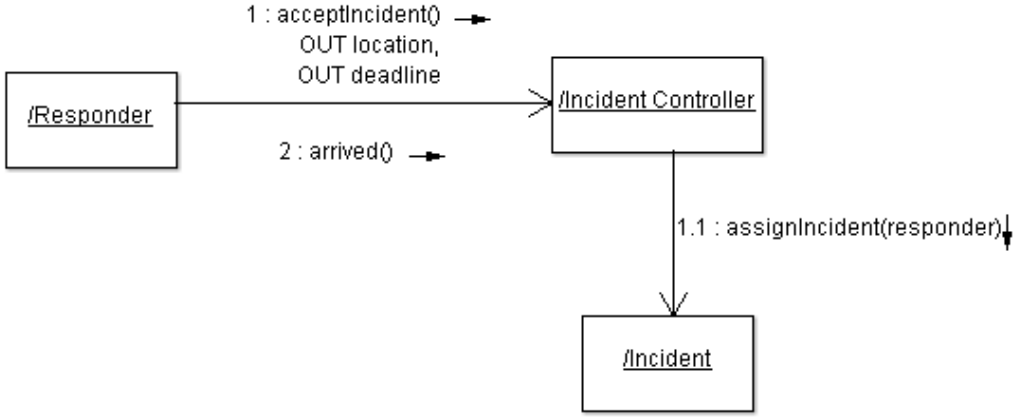
d. Scenario 2, Event: Correspondent assigns incident



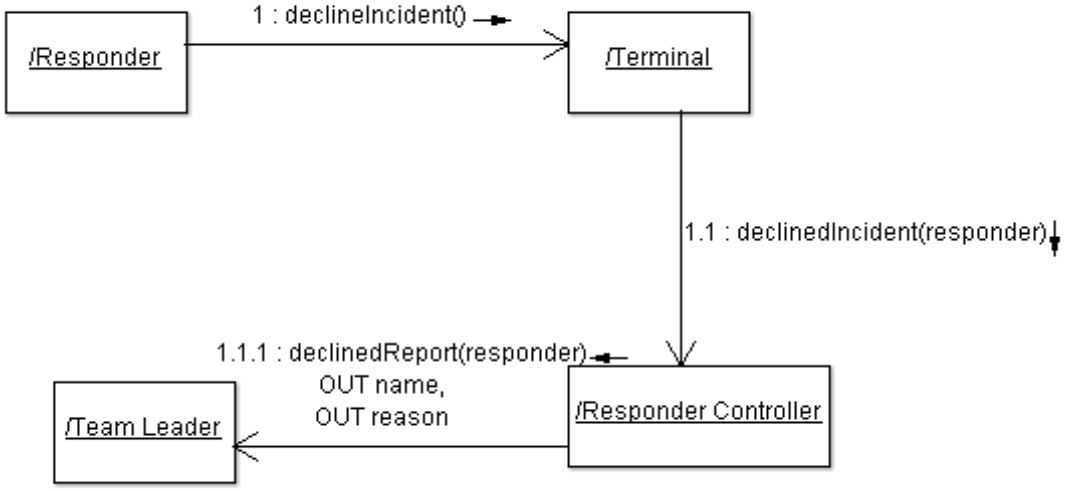
e. Scenario 3, Event: Team leader selects responders



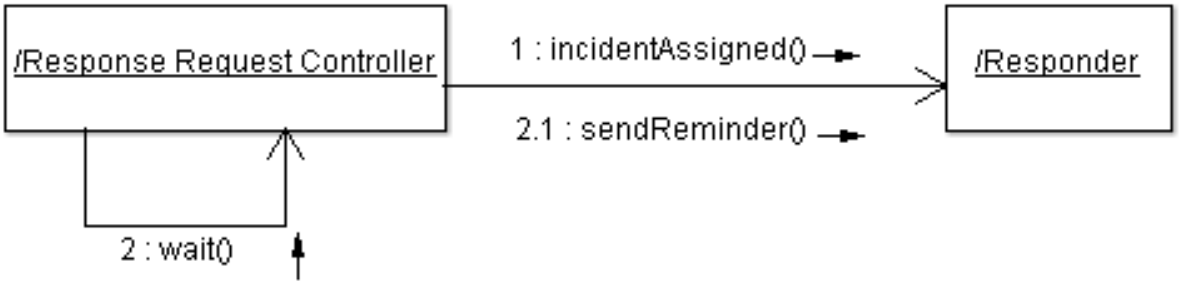
f. Scenario 3, Event: Responder accepts; incident assigned to responder with a deadline to acknowledge on arrival to the area where the incident occurred.



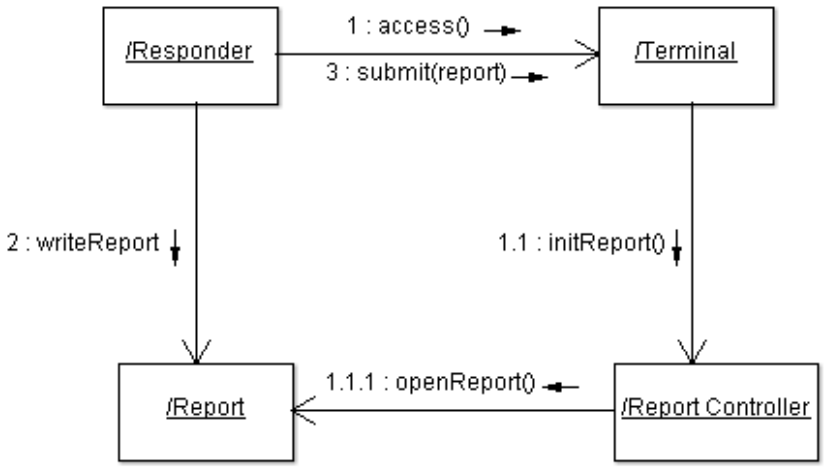
g. Scenario 3, Event: Responder declines; team leader notified



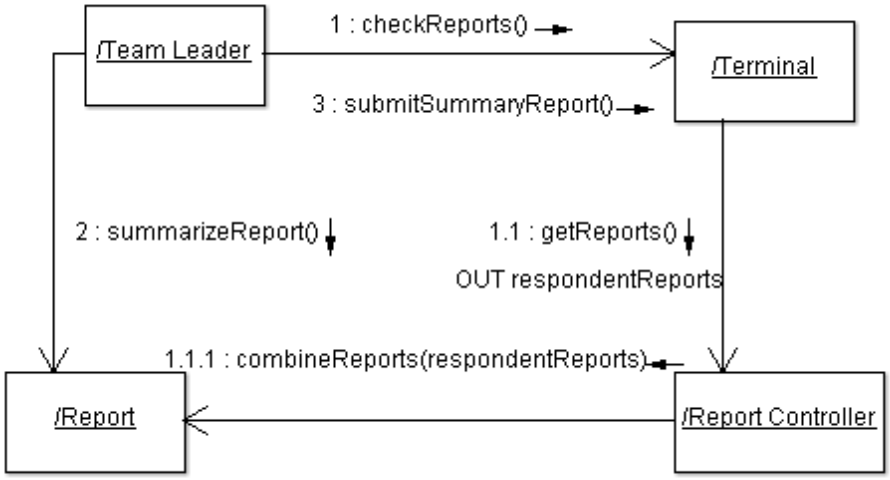
h. Scenario 3, Event: Timer expires; responder sent reminder



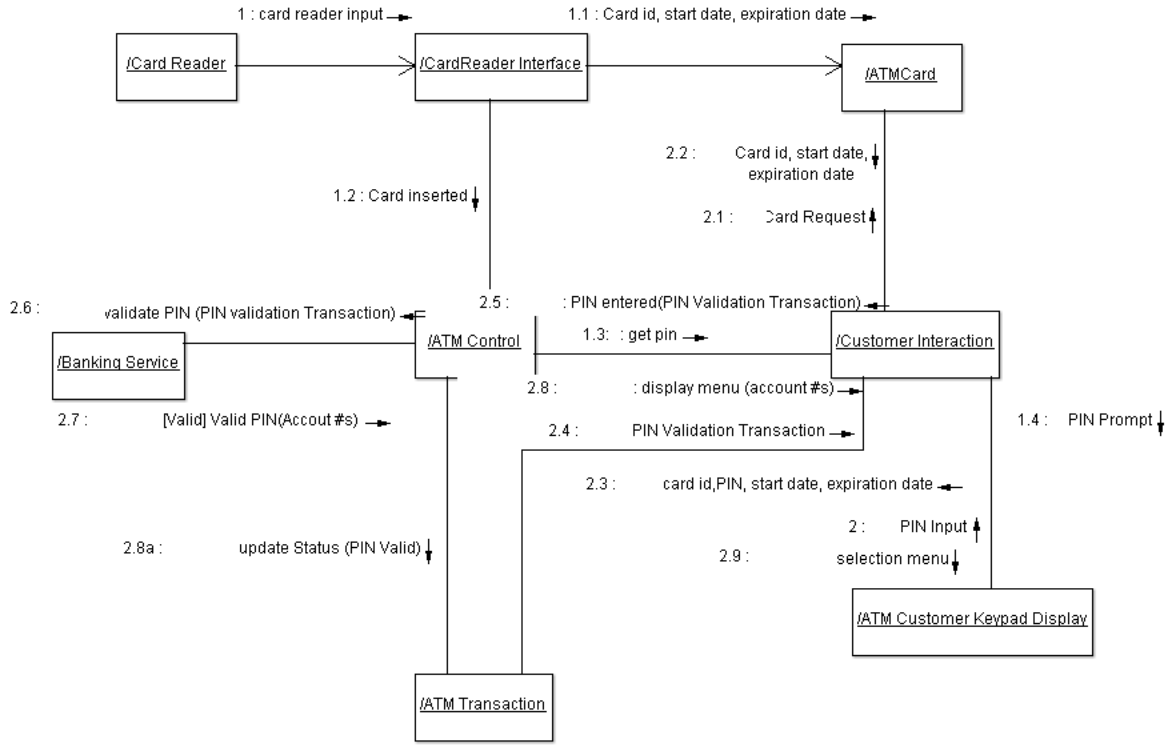
i. Scenario 3, Event: Responder submits review



j. Scenario 3, Event: Team leader submits summary review



3. Consider the given System Sequence Diagram. Design an equivalent collaboration design diagram. **(I was having a lot of trouble getting ArgoUML to do what I wanted on this one. Sorry that the formatting is a mess)**



4. Consider the collaboration design diagram.
   a. For each message listed, suggest one or more grasp guidelines suitable for justifying the allocation of responsibility.

Going by objects:

CardReader:

- cardReaderOutput – **high cohesion**

CardReaderInterface

- cardReaderInput – **don't talk to strangers**
- eject, confiscate – **indirection**

ATMCard

- write – **expert**
- read – **expert**

ATMCustomerKeypadDisplay

- display information – **low coupling**

CustomerInteraction

- customer input – **low coupling**

ATMTransaction

- updateCustomerInfo – **expert**
- updateCustomerSelection – **expert**
- updateTransaction – **expert**
- read – **creator**

ReceiptPrinterInterface

- print – **high cohesion**

ReceiptPrinter

- printerOutput – **high cohesion**

ATMControl

- cardInserted – **controller**
- cardEjected – **controller**
- cardConfiscated – **controller**
- startup – **controller**
- closedown – **controller**

ATMCash

- withdrawCash – **low coupling**
- addCash – **low coupling**

CashDispenserInterface

- dispence **- indirection**

OperatorInteraction

- operator input – **pure fabrication**

b. Draw a UML Design Class Diagram.