

**COMP 3700   
Homework 5   
Due: Monday, November 2nd, 2015 by 11:59PM (Please submit via Canvas)**

**JOHN CARROLL  
jcc0044/902521946**

This assignment is worth 100 points

**A (20)** 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, three-tier client server, abstract machine model, centralized, event-driven). Justify your answers.

**14.1. a** (An electronic chess companion)**,**

**14.1. b** (An airplane flight simulator for a video game),

**14.1. c** (A floppy disk controller chip),

**14.1. d** (A sonar system)

1. An electronic chess companion
   1. An electronic chess companion would be best designed using an event-driven model.
   2. The behavior to not allow invalid player moves is best modeled by the interrupt-driven model.
   3. Each move has to be checked by the companion and these moves are inputted through hardware switches.
   4. An invalid move would be represented by a particular interrupt type and shown to the user.
   5. Speed is also a concern because the player doesn’t want to have to wait for the computer to validate moves.
2. An airplane flight simulator for a video game
   1. A flight simulator would be best designed using a centralized control model.
   2. The game loop would control concurrent systems and other processes.
   3. The manager model is a good fit for this design. The game loop acts as a manager delegating different roles to each subsystem.
   4. Input, output, moving objects in the virtual world, keeping track of objects on screen, etc. are all handled by the central game loop through delegation of other sub-systems.
3. A floppy disk controller chip
   1. A floppy disk controller chip would be best designed using a centralized control  model.
   2. The controller chip is the “manager” of the manager model.
   3. Each of the components of the floppy disk system receives commands from the controller to complete their purpose.
   4. The manager model also suites this particular use case because of how the floppy disk system has to integrate with the computer system on the whole.
   5. A centralized command unit would make communication with other systems much simpler.
4. A sonar system
   1. A sonar system would be best designed using an event-driven model.
   2. Specifically, the interrupt-driven model.
   3. A sonar system must be capable of quickly registering external signals and reporting this information to appropriate sub-systems.
   4. The sensors that create and pick up these signals have a specific purpose and the calculations based on this data are best left to another sub- system.
   5. Picking up a signal that has bounced off an object could be seen as an interrupt that has a particular type based on the sensor that got the input.
   6. The system then handles this particular interrupt by passing the information to the relevant sub-system.

**B (80)** 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.

|  |
| --- |
| *Use case name:* Report Emergency *Participating Actors:* Initiated by the **officer** and communicates with **correspondent** *Flow of Events:*  1. The officer activates the “Report Emergency” function of her terminal  2. System responds by presenting a form to the officer  3. The officer fills the form by selecting the emergency level, type, location, and brief description of the situation. The officer also describes possible responses to the emergency situation. Once the form is completed, the field officer submits the form.  4. System receives the form and notifies the correspondent.  5. The correspondent reviews the submitted information and creates an incident in the database. The correspondent selects a response and acknowledges the report.  6. The system displays the acknowledgement and the selected response to the officer.  *Precondition:* The officer is logged into the system *Postcondition:* The officer has received an acknowledgement and the selected response from the correspondent, OR the officer has received an explanation indicating why the transaction could not be processed. |

The first step of this assignment is to ensure that you have an appropriate set of objects to arrange into a set of collaborations. The next step is to create collaborations that show how your objects work to handle various events within the following three scenarios that refine the basic use case above.

1. John (field officer) submits an emergency report to the FRIEND system via his terminal. The report omits necessary details about the incident (i.e., location, type). Therefore, the system does not accept the report and it displays an error message explaining why.

2. The correspondent is notified by the submission of an emergency report. The correspondent uses the terminal to visualize and check the current status of all the available resources, so that he can send the report and details about the incident to available responder team leaders. The correspondent chooses two team leaders. No team leader responds to the request (even though the incident falls within Matt and Dan’s area of expertise), so the system notifies the correspondent. He, in turn, assigns the incident to Sally.

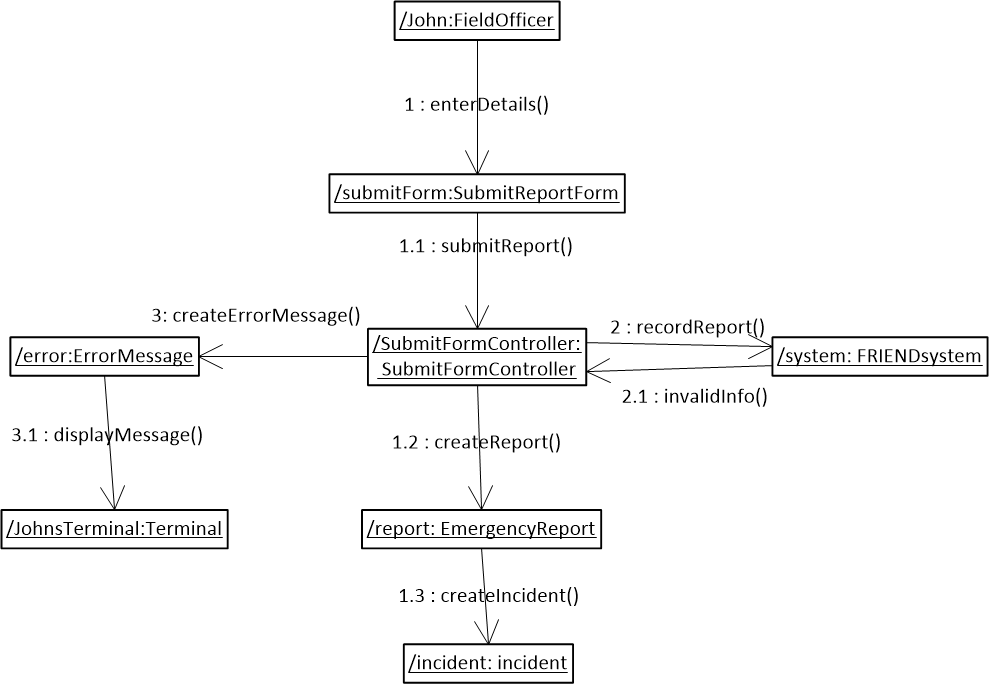
3. Sally, being a team leader, contacts Gary, Nick, and Mike asking if they are available to respond. Gary and Nick accept, but Mike declines. Sally then asks Tom if he is able to respond, and he accepts. Each responder acknowledges to the team leader when they arrive to the scene. After the incident is handled, each responder completes a report so that Sally can assemble a review report. Once all reviews are complete, Sally writes her summary report and sends using the system all four reviews to the correspondent. The correspondent sets the status of the incident to inactive and then closes the file.

You should create one interaction diagram (i.e. collaboration diagram) for each distinct event in the scenarios above. These events are:

|  |  |
| --- | --- |
| **Scenario** | **Event** |
| **1** | Emergency Report Submitted |
| **2** | Submission Info Sent to Available Team Leaders |
| **2** | Timer expires; Correspondent Notified |
| **2** | Correspondent Assigns Incident |
| **3** | Team Leader Selects Responders |
| **3** | Responder Accepts; Incident Assigned to Responder with a Deadline to Acknowledge on Arrival to the Area where the Incident Occurred |
| **3** | Responder Declines; Team Leader Notified |
| **3** | Timer Expires; Responder Sent Reminder |
| **3** | Responder Submits Review |
| **3** | Team Leader Submits Summary Review |

Your collaborations should make use of **boundary, controller, domain objects, and other new fabricated software objects** (e.g., data structure components) necessary to completely handle each event.

Interaction Diagram:



1. Emergency Report Submitted (Scenario “1”)
   * Entity
     1. EmergencyReport
   * Boundary
     1. EmergencyReportForm
   * Controller
     1. EmergencyReportFormController
2. Submission Info Sent to Available Team Leaders (2)
   * Entity
     1. Correspondent
     2. EmergencyReport
     3. Terminal
     4. TeamLeader
     5. Incident
   * Boundary
     1. ReportStatusWindow
     2. AvailableResourcesWindow
     3. TeamLeaderSelectionWindow
   * Controller
     1. ReportStatusWindowController
     2. AvailableResourcesWindowController
     3. TeamLeaderSelectionWindowController
3. Timer expires; Correspondant Notified (2)
   * Entity
     1. Correspondent
     2. Request
     3. TeamLeader
        1. Matt
        2. Dan
   * Boundary
     1. RequestResponderWindow
     2. TimeoutWindow
   * Controller
     1. RequestResponderWindowController
     2. TimeoutWindowController
4. Correspondent Assigns Incident (2)
   * Entity
     1. Correspondent
     2. TeamLeader
        1. Sally
        2. Request
   * Boundary
     1. RequestResponderWindow
   * Controller
     1. RequestResponderWindowController
5. Team Leader Selects Responders (3)
   * Entity
     1. TeamLeader
        1. Sally
     2. Responder
        1. Gary
        2. Nick
        3. Mike
        4. Tom
     3. Boundary
        1. SelectRepondersWindow
     4. Controller
        1. SelectRespondersWindowController
6. Responder Accepts; Incident Assigned to Responder with a Deadline to Acknowledge on Arrival to the Area where the Incident Occurred (3)
   * Entity
     1. Responder
        1. Gary
        2. Nick
        3. Tom
     2. Incident
   * Boundary
     1. ArrivedOnSceneWindow
   * Controller
   * ArrivedOnSceneWindowController
7. Responder Declines; Team Leader Notified (3)
   * Entity
     1. Responder
        1. Mike
     2. TeamLeader
        1. Sally
   * Boundary
     1. DeclineRequestForm
   * Controller
     1. DeclineRequestFormController
8. Timer Expires; Responder Sent Reminder (3)
   * Entity
     1. Responder
     2. Request
   * Boundary
     1. LateToSceneMessage
   * Controller
     1. LateToSceneMessageController
9. Responder Submits Review (3)
   * Entity
     1. Responder
        1. Gary
        2. Nick
        3. Tom
     2. ResponseReport
   * Boundary
     1. ResponseReportForm
   * Controller
     1. ResponseReportFormController
10. Team Leader Submits Summary Review (3)
    * Entity
      1. TeamLeader
         1. Sally
      2. ReviewReport
      3. Correspondent
      4. EmergencyReport
    * Boundary
      1. ResponseReportSummaryWindow
      2. ReviewReportForm
      3. CloseEmergencyReportForm
    * Controller
      1. ResponseReportSummaryWindowController
      2. ReviewReportFormController
      3. CloseEmergencyReportFormController