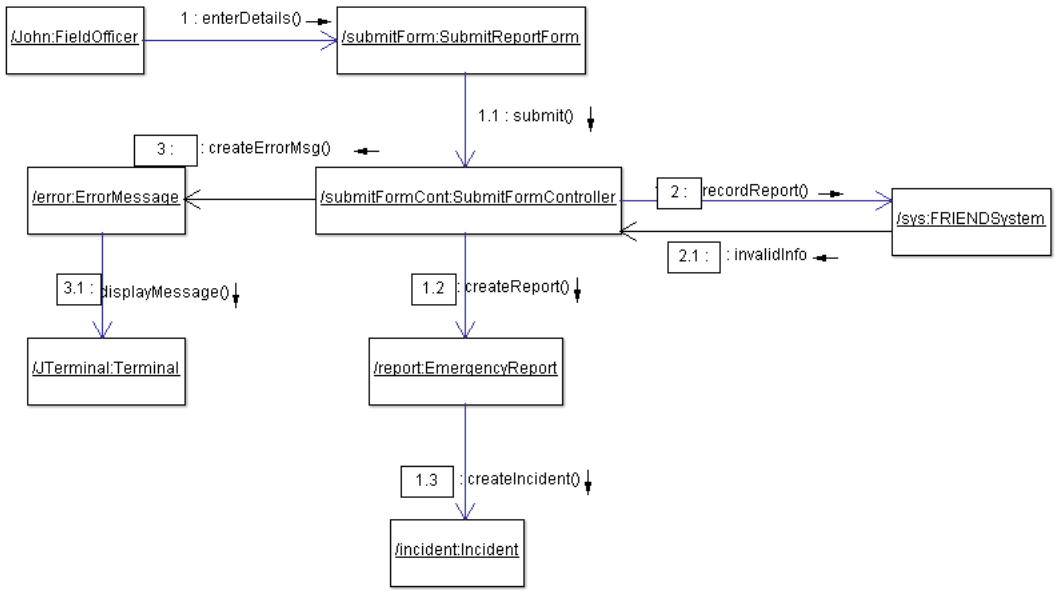
Homework 5

Stephen Ward

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, three-tier client server, abstract machine model, centralized, event-driven). Justify your answers.
    1. An electronic chess companion
        - An electronic chess companion would be best designed using an event-driven model. The behavior to not allow invalid player moves is best modeled by the interrupt-driven model. Each move has to be checked by the companion and these moves are inputted through hardware switches. An invalid move would be represented by a particular interrupt type and shown to the user. 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
        - A flight simulator would be best designed using a centralized control model. The game loop would control concurrent systems and other processes. The manager model is a good fit for this design. The game loop acts as a manager delegating different roles to each subsystem. 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
        - A floppy disk controller chip would be best designed using a centralized control model. The controller chip is the "manager" of the manager model. Each of the components of the floppy disk system receives commands from the controller to complete their purpose. 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. A centralized command unit would make communication with other systems much simpler.
    4. A sonar system
        - A sonar system would be best designed using an event-driven model. Specifically, the interrupt-driven model. A sonar system must be capable of quickly registering external signals and reporting this information to appropriate sub-systems. 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. 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. The system then handles this particular interrupt by passing the information to the relevant sub-system.

2. You should create one interaction diagram (i.e. collaboration diagram) for each distinct event in the scenarios above.
    1. Emergency Report Submitted (1)



    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; Correspondent Notified (2)
        - Entity
            1. Correspondent
            2. Request
            3. TeamLeader

        a. Matt
        b. Dan
    - Boundary
        1. RequestResponderWindow
        2. TimeoutWindow
    - Controller
        1. RequestResponderWindowController
        2. TimeoutWindowController
4. Correspondent Assigns Incident (2)
    - Entity
        1. Correspondent
        2. TeamLeader
            a. Sally
        3. Request
    - Boundary
        1. RequestResponderWindow
    - Controller
        1. RequestResponderWindowController
5. Team Leader Selects Responders (3)
    - Entity
        1. TeamLeader
            a. Sally
        2. Responder
            a. Gary
            b. Nick
            c. Mike
            d. Tom
    - Boundary
        1. SelectRepondersWindow
    - 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
            a. Gary
            b. Nick
            c. Tom
        2. Incident
    - Boundary
        1. ArrivedOnSceneWindow

- Controller
    1. ArrivedOnSceneWindowController

7. Responder Declines; Team Leader Notified (3)
    - Entity
        1. Responder
            a. Mike
        2. TeamLeader
            a. 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
            a. Gary
            b. Nick
            c. Tom
        2. ResponseReport
    - Boundary
        1. ResponseReportForm
    - Controller
        1. ResponseReportFormController
10. Team Leader Submits Summary Review (3)
    - Entity
        1. TeamLeader
            a. Sally
        2. ReviewReport
        3. Correspondent
        4. EmergencyReport
    - Boundary
        1. ResponseReportSummaryWindow
        2. ReviewReportForm

    3. CloseEmergencyReportForm
- Controller
    1. ResponseReportSummaryWindowController
    2. ReviewReportFormController
    3. CloseEmergencyReportFormController