SWE 727 QoS for Software Architectures Spring 2011

Lab2: Event Styles

Sousa **Discuss:** Feb 15, **Due:** Mar 08

Problem Description

The objective of this assignment is to experiment with event-oriented architectural styles, to discover their inherent strengths and weaknesses. An event manager and API access class for interfacing with the event manager is provided for this assignment. A sample system implementation and support classes has also been provided that illustrates how to use the event manager to create systems. Part of this assignment will be implementation-oriented, allowing you to experiment with a particular event mechanism to gain a clearer understanding of the issues associated with carrying an architectural design through to code. Note, however, as with previous assignments that this is not a programming class, and so the emphasis of this assignment is on the architecture issues.

The assignment consists of two parts: for the first part of the assignment, you will be provided with an operational event manager that allows you to send and receive events from processes or threads. The application domain for this assignment will be an environmental control system as described below in the business context. Your task in part one is to extend the existing system as specified in the requirements below, using the existing framework and support classes.

The second part of the assignment consists of analyzing the architecture of the systems you have designed and built. You will reflect upon your work and answer questions related to the design decisions that your team made in part one.

While you may discuss this assignment with your colleagues, it must be done individually.

Business Context and Key Architectural Approaches

The principal stakeholder for this system is an organization that builds environmental control and security systems. These systems are used to control heating, ventilation, and air conditioning (HVAC) equipment as well as security systems to detect intrusion, motion, smoke, and fire. These are typically highly distributed systems that use remote sensors and HVAC controllers connected to monitoring consoles. A key requirement for this organization is to have a highly extensible system where sensors, equipment controllers, and consoles can be easily added to the system at runtime. It is also important that the system reliably delivers events from sensors to consoles and from consoles to controllers in a timely fashion. The role of sensors, controllers, and consoles is described below:

Sensor – Sensors read environmental data. Sensors are configured to sample the environment and post events to the event manager at a periodic rate. Example sensors include temperature, smoke, motion, intrusion, humidity, and so forth.

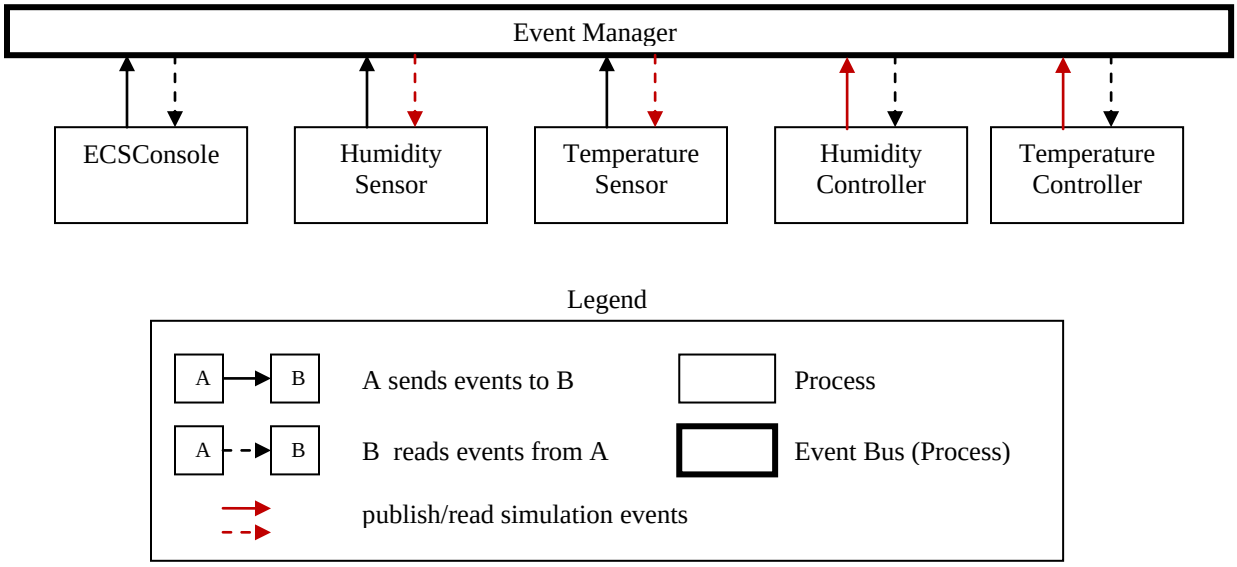
Controller – A controller has 1 or more devices directly connected to it and is able to control the devices (turn them on/off). Controllers receive information from the event manager and control the connected devices. A controller acts as an interface between actual devices and the event manager. The type of equipment connected to a controller might control include: heaters, chillers, humidifiers, dehumidifiers, ventilation, door locks, shades, sprinklers, and so forth. Control-

lers read data from the event manager (sensors and monitors) and effect control over the equipment they are connected to.

Console – A console is a display for human operators responsible for observing and responding to alerts and messages from the system. Consoles also allow users to set temperatures, humidity, enable/disable security and so forth. The operators then monitor and control the overall behavior of the system. Consoles use the event manager to send and receive data to and from the system elements.

The Sample Application:

A sample application has been provided. This system is a humidity and temperature control system for a museum's exhibit of famous oil paintings. Oil paintings must be kept at a temperature of 70°F-75°F and at a humidity of 45%-55% relative humidity. The current system consists of 4 elements: a monitor, two controllers, and a temperature sensor. A C&C view of the system is shown below:



Each of the system elements shown above are explained in more detail below:

ECSConsole – This monitor process consists of two threads. One thread (the main) serves as a console that allows the user to set humidity and temperature set values. The other thread monitors and controls the ambient temperature and humidity by turning on and off the chiller, heater, humidifier, and dehumidifier to maintain the set temperature and humidity. The console displays the status of the temperature and humidity.

Temperature Sensor (Simulator) – This is a process that simulates the action of a temperature sensor. Normally a sensor would only write to the event manager, but to simulate the action of a real device, the temperature sensor simulator reads events from the temperature and humidity controllers and reacts to these devices being turned on or off by trending the temperature values of higher or lower in response. The temperature sensor simulator posts temperature events once every two seconds.

Humidity Sensor (Simulator) – This is a process that simulates the action of a humidity sensor. Normally a sensor would only write to the event manager, but to simulate the action of a real device, the humidity sensor simulator reads events from the humidity controller and reacts to these devices being turned on or off by trending the humidity values of higher or lower. The temperature sensor simulator posts temperature events once every two seconds.

Humidity Controller – The humidity controller is a process that interfaces with, and controls two devices: a humidifier to add moisture to the air, and a dehumidifier to remove moisture from the air. This element only reads from the event manager.

Temperature Controller – The temperature controller is a process that interfaces with, and controls, two devices: a heater to raise the ambient temperature, and a chiller to lower the ambient temperature of the room. This element only reads from the event manager.

These processes communicate via an event manager. Events are objects that include an integer event number and a string text field that can be used to pass along a message. Event objects have no semantic meaning to the event manager. To enable communication for the museum environmental control system, an event semantic has been established where the event number is used to signal system elements and how they should respond to them. The string text field of the event is used to convey a specific action or message to the element. The events are described for each element in the tables below:

Element: ECSConsole	Event IDs this element responds to: 1, 2
Action Codes:	Action:
"xxx"	If event ID = 1, then xxx is a string representation of the floating point ambient room temperature in degrees Fahrenheit.
"yyy"	If event ID = 2, then yyy is a string representation of the floating point ambient room humidity in percentage relative humidity.
Application Notes: None.	

Element: Temperature-Sensor	Event IDs this element responds to: -5 (simulation event)
Action Codes: None	Action: None
"H1"	Confirmation turn on the heater
"H0"	Confirmation turn off the heater
"C1"	Confirmation turn on the chiller
"C0"	Confirmation turn off the chiller
Application Notes: Note that this element is a simulator. In an actual application this element will only write the ambient room temperature to the monitor. However, this element has to simulate this action. To do this, the TemperatureSensor element listens for confirmation events sent by the TemperatureController (event -5) and will trend the temperature readings up or down whether the chiller or heater has been turned on/off.	

Element: HumiditySensor	Event IDs this element responds to: -4 (simulation event)
Action Codes: None	Action: None
“H1”	Confirmation turn on the humidifier
“H0”	Confirmation turn off the humidifier
“D1”	Confirmation turn on the dehumidifier
“D0”	Confirmation turn off the dehumidifier
Application Notes: Note that this element is a simulator. In an actual application this element will only write the ambient room humidity to the monitor. However, this element has to simulate this action. To do this, the HumiditySensor element listens for confirmation events sent by the HumidityController (event -4) and will trend the humidity readings up or down whether the humidifier or dehumidifier has been turned on/off.	

Element: Humidity Controller	Event IDs this element responds to: 4
Action Codes:	Action:
“H1”	Turn on the humidifier
“H0”	Turn off the humidifier
“D1”	Turn on the dehumidifier
“D0”	Turn off the dehumidifier
Application Notes: This element will read the pending events from the event manager and will only respond to event numbers of 4. The strings in the text field are of the form XY where X indicates the device: H = humidifier and D = dehumidifier; Y indicates on (1 or one) and off (0 or zero).	

Element: Temperature Controller	Event IDs this element responds to: 5
Action Codes (YY):	Action:
“H1”	Turn on the heater
“H0”	Turn off the heater
“C1”	Turn on the chiller
“C0”	Turn off the chiller
Application Notes: This element will read the pending events from the event manager and will only respond to event numbers of 5. The strings in the text field are of the form XY where X indicates the device: H = heater and C = chiller; Y indicates on (1 or one) and off (0 or zero).	

Installing the Event Manager and Sample Application

Unzip the *A3 Source.zip* file into a working directory. The design details, application notes, and installation instructions for the event manager and the museum environmental control system are included in a separate document: *EM Application Framework*. For this assignment you will *not* have to modify the event manager and you will not have access to its source code. You *will,* however, have access to a set of classes that will enable you to quickly create applications that will use the event manager. The sample application illustrates the use of these classes. Instructions for installing and running the sample application are also in the accompanying *EM Application Framework* document.

Part 1: Design and Construction

Your team’s task is to use the existing framework as a basis for creating 3 new systems. Each new system has one or more requirements. For each system, please adhere to the event-oriented architectural pattern as closely as possible. Make sure that you use good programming practices, including comments that indicate how you changed the base system. If you do not follow reasonably good programming practices, your team grade will be penalized.

System A

Add intrusion detection and monitoring to the existing museum system. The system should include 3 different alarms: window break, door break, and motion detection. Assume that these alarms are on a single controller. Create a separate security monitor that allows a guard to arm and disarm the system (you can use the ECSConsole as a model). When the system is disarmed, security events are not reported; however when the system is armed, security intrusions should be reported to the user via the security monitor. The type of alarm (window break, door break, motion) should be reported. To simplify your application and test your system you should simulate security alerts by providing a simple text interface (use the temperature and humidity sensors as a model) that allows a user to create a security event.

System B

Add fire detection and monitoring to the existing museum system. The system should include one fire alarm controller and one sprinkler controller. When the controller indicates that a fire has been detected, your system should show the alarm in the same console as the security console. Allow the user to confirm or cancel sprinkler action. If there is no input from the user in

15 seconds, the sprinkler will turn on automatically. Once on, the fire detection monitor should allow the user to turn off the sprinkler.

System C

Add to the existing system a service maintenance console with a function that shows all the equipment currently installed in the museum. The maintenance console should list those elements that are connected to the system - each device should have a device name and short description. The maintenance console should detect when a device is no longer responding and notify the user via the service maintenance console.

Packaging and submitting Part 1

- Systems A, B, and C should be clearly separated, that is there should be three distinct systems. Place each implementation in a different folder for each of the systems.
- Include very clear description of how to install and run your systems. If we cannot install and run your systems, then you will not receive credit for the assignment.
- Part 1 must be emailed in a compressed folder named as LAB2-*yourLastName*.

Part 2: System Analysis

Please answer the following questions. Each question has several parts. Make sure that you answer each question completely in your write-up:

1. For the overall system:
   a. According to the business context, what are the key architectural drivers of the system and what is their relative priority?
   b. From a dynamic perspective, is the event manager a connector or a component? Explain your answer.
   c. How well do the architect's initial structural choices (the use of an event-oriented architectural pattern and the provided framework) support the business goals as described in the business context?
2. For each system A, B, and C:
   a. Describe the architecture of your system. Be sure to include appropriate views of the system. You are free to use whatever notations you prefer, but of course you must follow good standards of architectural documentation.
   b. Discuss how well the design choices that you made support the business goals of the system in terms of the relevant quality attributes. What were the key design decisions, tradeoffs, and what motivated your choices?
   c. Discuss how each set of modifications for systems A, B, and C effect system startup, shutdown of the respective systems?
3. Given your design and implementation of these systems:
   a. Describe which of these modifications were relatively easy and which of these modifications were relatively difficult? What role did the event-oriented pattern play in making these modifications easy or difficult?
   b. Describe how reuse was promoted and/or inhibited by your design decision and/or the event style (hint: be precise when describing "reuse").

4. Extension 1 (you are not required to implement this extension; however, describe the system in sufficient detail so that the architecture can be understood):
    a. Suppose that there were two instances of event managers running on two different machines. Describe how you might have the event managers share events with all the elements connected to them. Assume that you can't modify the event manager and only one event manager can be running on a computer at a time (hint: please list any other assumptions you make).
5. Extension 2 (you are not required to implement this extension; however, describe the system in sufficient detail so that the architecture can be understood):
    a. Modify the system so that In case of a fire event in the museum, assume that there are fireproof glass protective coverings for the paintings that must be lowered and the sprinklers activated. However, it must be ensured that the water is not turned on before the protections are completely lowered.

Grading Criteria

Your solutions and commentary will be graded based upon:

- The quality and contents of your write-up: this includes describing the design, discussions of trade-offs, and your discussion of the implications of changes to the system's architecture and the inherent systemic properties. Be clear, concise, and complete.
- The consistency between the design representations and the implementation.
- The degree to which your solutions respect the event oriented architectural pattern.
- The extent to which deviations from the event architectural pattern (as well as their effect) are clearly explained.
- Professionalism, which includes the quality of the report, timely submission, and well-structured and documented source code.
- The correct operation of the solutions - we will test your solutions with our test data.

Each question will be weighted as follows (100 points maximum):

Part 1: Implementation and Write-up

- Java implementation of System A: 10 points
- Java implementation of System B: 15 points
- Java Implementation of System C: 25 points

Part 2: Write-up

- Question 1: 10 points
- Question 2: 10 points
- Question 3: 10 points
- Question 4: 10 points
- Question 5: 10 points