SOEN 342 - Sections H and II:
Software Requirements and Specifications

Iteration 2: Sprint 2 Part 1/3 OCL specifications

Antoine Cantin
40211205

Tuan Anh Pham
40213926

Mustafa Sameem
40190889

Sunday 12th November, 2023

1 OCL formal specifications

```
context TempMonitor::DeploySensor(sensor,location,temperature)
    pre:
    self.deployedSensors->excludes(sensor)
    not self.slTable->exists(entry | entry.key = sensor)
    post:
    self.deployedSensors = self.deployedSensors@pre->including(sensor)
    self.slTable =
       self.slTable@pre->including(Tuple{key=sensor,value=location})
    self.stTable =
       self.stTable@pre->including(Tuple{key=sensor,value=temperature})

context TempMonitor::ReadTemperature(location)
    pre:
    self.slTable -> exists(entry | entry.value = location)
    post:
    temperature = self.stTable->at(self.slTable->select(entry |
       entry.value = location)->first().key)

context TempMonitor::ReplaceSensor(old_sensor,new_sensor)
    pre:
    self.deployedSensors->includes(old_sensor)
    self.sensorRegistry->includes(new_sensor) and
       self.deployedSensors->excludes(old_sensor)
    not new_sensor.isOutOfService()

    post:
    self.deployedSensors = self.deployedSensors@pre - Set{old_sensor}

    self.slTable = self.slTable@pre->union(
    Tuple{key=new_sensor,value=self.slTable@pre->at(old_sensor)})
    self.stTable = self.stTable@pre->union(
    Tuple{key=new_sensor,value=self.stTable@pre->at(old_sensor)})

```

```
    self.slTable = self.slTable->excluding(
    Tuple{key=old_sensor,value=self.slTable@pre->at(old_sensor)})
    self.stTable = self.stTable->excluding(
    Tuple{key=old_sensor,value=self.stTable@pre->at(old_sensor)})

context TempMonitor::RemoveSensor(sensor)
    pre:
        if sensor.isDeployed() then
            self.deployedSensors->includes(sensor)
        else
            self.sensorRegistry->includes(sensor)
        endif
    post:
        if sensor.isDeployed() then
            self.deployedSensors = self.deployedSensors@pre - Set{sensor}
            self.slTable = self.slTable->excluding(
            Tuple{key=sensor,value=self.slTable@pre->at(sensor)})
            self.stTable = self.stTable->excluding(
            Tuple{key=sensor,value=self.stTable@pre->at(sensor)})
        else
            self.sensorRegistry = self.sensorRegistry@pre - Set{sensor}
            sensor.setOutOfService(true)
        endif
```

Listing 1: Core operations

2 OCL requirements

```
context TempMonitor
    inv: self.sensorsRegistry->select(isDeployed = true) =
       self.deployedSensors

-- Each sensor deployed by the system must have a unique id:
context Sensor
    inv: Sensor.allInstances()->forAll(s1, s2 | s1 <> s2 implies s1.id <>
       s2.id)
--All sensors deployed in the system are sensors that are maintained by
   the system:
context TempMonitor
inv: SensorRegistry.sensors->includesAll(self.deployedSensors)

--Every deployed sensor must be associated with a location that exists in
   the corresponding location registry:
context Sensor
inv: self.deployed implies
   LocationRegistry.locations->includes(self.location)

--Every location maintained by the system is associated with a unique
   sensor:
context TempMonitor
inv: LocationRegistry->forAll(l1, l2 | l1 <> l2 implies
   l1.associatedSensor <> l2.associatedSensor)
```

Listing 2: Requirements pt1

```
Context SensorRegistry
inv: self.sensors->forAll(s1, s2 : Sensor | s1 <> s2 implies s1.id <>
    s2.id)
inv: self.sensors->select(deployed = true)->forAll(s : Sensor |
    self.sensors->includes(s))
inv: self.sensors->select(deployed = true)->forAll(s : Sensor |
    s.deployed implies not s.outOfService)

Context LocationRegistry
inv: self.locations->forAll(l1, l2 | l1 <> l2 implies l1.id <> l2.id)

Context SensorLocationTable
inv: self.allInstances->forAll(sl : SensorLocationPair |
    sl.sensor.deployed)
inv: self.allInstances->forAll(sl : SensorLocationPair |
    sl.sensor.deployed implies not sl.sensor.outOfService)
inv: self.allInstances->forAll(sl : SensorLocationPair |
    SensorRegistry.sensors->includes(sl.sensor))
inv: self.allInstances->forAll(sl : SensorLocationPair |
    LocationRegistry.locations->includes(sl.location))
inv: self.allInstances->forAll(sl : SensorLocationPair | sl.location ->
    exists(l : Location | l = sl.location))
inv: self.allInstances->forAll(sl : SensorLocationPair | sl.sensor ->
    exists(s : Sensor | s = sl.sensor))
inv: self.allInstances->forAll(sl1, sl2 : SensorLocationPair | sl1 <> sl2
    implies sl1.sensor.id <> sl2.sensor.id)
inv: self.allInstances->forAll(sl1, sl2 : SensorLocationPair | sl1 <> sl2
    implies sl1.location.id <> sl2.location.id)

Context SensorTemperatureTable
inv: self.allInstances->forAll(st : SensorTemperaturePair |
    st.sensor.deployed)
inv: self.allInstances->forAll(st : SensorTemperaturePair |
    st.sensor.deployed implies not st.sensor.outOfService)
inv: self.allInstances->forAll(st : SensorTemperaturePair |
    SensorRegistry.sensors->includes(st.sensor))
inv: self.allInstances->forAll(st : SensorTemperaturePair |
    st.temperature -> exists(t : Temperature | t = st.temperature))
inv: self.allInstances->forAll(st : SensorTemperaturePair | st.sensor ->
    exists(s : Sensor | s = st.sensor))
inv: self.allInstances->forAll(st1, st2 : SensorTemperaturePair | st1 <>
    st2 implies st1.sensor.id <> st2.sensor.id)
```

Listing 3: Requirements pt2