Yinong Dai

CS201

Factory v.0 Conveyor Family 2 Design

3/28/13

Content:

I. Interaction Diagrams

II. Agent Designs

1. Design: Conveyor Family

2. Design: Conveyor Agent

3. Design: Pop-up Agent

4. Design: In-line Processing Agent

Design of Agents

Design: Conveyor Family

Data

int index;

String function;

ConveyorAgent conveyor;

PopupAgent popup;

InlineProcessingAgent inline;

Messages

```
void msgIAmFree() {
        popup.msgIAmFree();
}
void msgHereIsGlass(glass) {
        conveyor.msgHereIsGlass(glass);
}
void msgHereIsFinishedGlass(glass) {
        popup.msgHereIsFinishedGlass(glass);
}
```

Scheduler

N/A

Actions

N/A

Design: Conveyor Agent

Data

```
int conveyorIndex;
String function;
Mode mode;
enum Mode {Offline, Inline};
Transducer transducr;
List<Glass> glasses;
PopupAgent popup;
PopupState popupState;
enum PopupState {BUSY, FREE};
InlineAgent inline;
boolean inlineFree;
ConveyorFamily previouscf;
boolean conveyorRunning;
SensorState sensor1State;
SensorState sensor2State;
enum SensorState {PRESSED, RELEASED, NOTHING};
```

Messages / Eventfires

```
void msgHereIsGlass(glass) {
        glasses.add(new MyGlass(glass, INITIAL);
        stateChanged();
}
void msgPopupBusy() {popupState = BUSY;}
void msgPopupFree() {popupState = FREE;}
void eventFired() { //Registered to SENSOR channel
        if( (event == SENSOR_GUI_PRESSED) &&
                (args[0]/2 == conveyorIndex) ) {
                if(args[0]%2 == 0) sensor1State = PRESSED;
                if(args[0]%2 == 1) sensor2State = PRESSED;
        }
```

```
        if( (event == SENSOR_GUI_RELEASED) &&
                (args[0]/2 == conveyorIndex) ) {
                if(args[0]%2 == 0) sensor1State = RELEASED;
                if(args[0]%2 == 1) sensor2State = RELEASED;
        }
}
```

Scheduler

```
if(mode == OFFLINE) {
        if( (popupState == BUSY) && (sensor2State == PRESSED) && (conveyorRunning) )
        {stopConveyorAndNotify(); return true;}
        if( (popupState == FREE) && (!conveyorRunning) ) {startConveyor(); return true;}
        if( sensor2State == RELEASED ) {giveGlassToPopup(); return true;}
}
if(mode == INLINE) {
        if( (!inlineFree ) && (sensor2State == PRESSED) && (conveyorRunning )
        {stopConveyor(); return true;}
        if( (inlineFree) && (!conveyorRunning)  ) {startConveyor(); return true;}
        if( sensor2State == RELEASED ) {giveGlassToInline(); return true;}
}
if( sensor1State == RELEASED ) {notifyPrevious(); return true;}
return false;
```

Actions

```
void stopConveyor() {
        transducer.fireEvent(CONVEYOR, CONVEYOR_DO_STOP);
        conveyorRunning = false;
}
void stopConveyorAndNotify() {
        stopConveyor();
        popup.msgIHaveGlass(glasses.get(0));
}
```

```
void startConveyor() {
        transducer.fireEvent(CONVEYOR, CONVEYOR_DO_START);
        conveyorRunning = true;
}
void giveGlassToPopup() {
        popup.msgHereIsGlass(glasses.remove(0));
        sensor2State == NOTHING;
}
void giveGlassToInline() {
        inline.msgHereIsGlass(glasses.remove(0));
        sensor2State = NOTHING;
}
void notifyPrevious() {
        previous.msgIAmFree();
        sensor1State == NOTHING;
}
```

Design: Pop-up Agent

Data

int popupIndex;

String function;

Transducer transducr;

Operator[2] operators;

boolean[2] operatorFree;

ConveyorFamily next;

boolean nextFree;

ConveyorAgent conveyor;

List<MyGlass> glasses;

class MyGlass {Glass glass; GlassState state; int conveyorIndex;}

enum GlassState {ON_ENTRY, PENDING, NEED_PROCESSING, DONE, PASS, NOTHING};

PopupState popupState;

PopupLevel popupLevel;

enum PopupState {LOADED,EMPTY};

enum LevelState {UP, DOWN};

Semaphore loadSemaphore;

Semaphore elevSemaphore;

Messages / Eventfires

```
void msgIHaveGlass(glass) {
        for(MyGlass g: glasses) {
                if(g.glass == glass) {
                        g.state = ONLINE;
                        return;
                }
        }

        glasses.add(new MyGlass(glass, ONLINE);
        stateChanged();
```

```
}
void msgHereIsGlass(glass) {
        for(MyGlass g: glasses) {
                if(g.glass == glass) {
                        loadSemaphore.acquire();
                        popupState = LOADED;
                        g.state = PENDING;
        }
        stateChanged();
}
void msgIAmFree() {
        nextFree = true;
}
void msgIHaveGlassFinished() {
        for(MyGlass g: glasses) {
                if(g.glass == glass)
                        g.state = DONE;
        }
        stateChanged();
}
void msgHereIsFinishedGlass(glass) {
        for(MyGlass g: glasses) {
                if(g.glass == glass) {
                        loadSemaphore.acquire();
                        popupState = LOADED;
                        g.state = PASS;
                }
        }
        stateChanged();
}
void eventFired(…) {
```

```
        if( (event == POP_UP_LOAD_FINISHED) && (args[0] == popupIndex) ) {
                loadSemaphore.release();
        }
        if( (event == POP_UP_RELEASED_FINISHED) && (args[0] == popupIndex) ) {
                loadSemaphore.release();
        }
        if( (event == POP_UP_MOVED_UP) && (args[0] == popupIndex) ) {
                elevatorSemaphore.release();
        }
        if( (event == POP_UP_MOVED_DOWN) && (args[0] == popupIndex) ) {
                elevatorSemaphore.release();


        }
```

Scheduler

```
if there exists MyGlass g such that g.state == DONE then {acceptFinishedGlass(g);return true;}
if there exists MyGlass g such that g.state == PENDING then identifyGlass();return true;}
if there exists MyGlass g such that g.state == PASS && nextFree then {pushGlass();return true;}
if there exists MyGlass g such that g.state == NEED_PROCESSING && (operatorFree[0] ||
operatorFree[1]) then {giveOperatorGlass();return true;}
if there exists MyGlass g such that g.state == ONLINE then {acceptGlass();return true;}
return false;
```

Actions

```
void acceptFinishedGlass(g) {
        raisePopup();
        operator[g.operatorIndex].msgIAmFree();
        loadSemaphore.acquire();
        g.state = PASS;
        operatorFree[g.operatorIndex] = true;
}
void giveOperatorGlass(g) {
        if(operatorFree[0]) {
                raisePopup();
```

```
                operator[0].msgHereIsGlass(g.glass);
                //transducer.fireEvent(WORK_STATION,
WORKSTATION_DO_LOAD_GLASS,popupIndex*2);
                loadSemaphore.acquire();
                operatorFree[0] = false;
                //Here, Operator Agent will make the animation load the glass. When loading is
finished, semaphore should be released
                g.state = NOTHING;
        } else if(operatorFree[1]) {
                raisePopup();
                operator[1].msgHereIsGlass(g.glass);
                //transducer.fireEvent(WORK_STATION,
WORKSTATION_DO_LOAD_GLASS,popupIndex*2+1);
                loadSemaphore.acquire();
                operatorFree[1] = false;
                //Here, Operator Agent will make the animation load the glass. When loading is
finished, semaphore should be released
                g.state = NOTHING;
        }
}

void raisePopup() {
        transducer.fireEvent(POP_UP, POP_UP_DO_MOVE_UP);
        conveyor.msgPopupBusy();
        elevatorSempahore.acquire();
        popupLevel = UP;
}
void lowerPopup() {
        transducer.fireEvent(POP_UP, POP_UP_DO_MOVE_DOWN);
        elevatorSempahore.acquire();
        popupLevel = DOWN;
}
```

Design: Inline Agent

Data

```
String function;
Transducer transducr;
TChannel channel;
Glass glassOnSpot;
ConveyorFamily next;
boolean nextFree;
ConveyorAgent conveyor;
Semaphore machineSemaphore;
```

Messages

```
void msgHereIsGlass(glass) {
        glassOnSpot = glass;
        stateChanged();
}
void msgIAmFree() {
        nextFree = true;
        stateChanged();
}
void eventFired(,,,) {
if(Event == WORKSTATION_GUI_ACTION_FINISHED)
        machineSemaphore.release();
if(Event == WORKSTATION_RELEASE _FINISHED)
        machineSemaphore.release();
}
```

Scheduler

```
if glassOnSpot != null then {processGlass(glassOnSpot); return true;}
return false;
```

Action

```
void processGlass(glass) {
```

```
        transducer.fireEvent(channel, WORKSTATION_DO_ACTION);
        machineSemaphore.acquire();
        while(!nextFree) {}
        next.msgHereIsGlass(glassOnSpot);
        transducer.fireEvent(channel, WORKSTATION_RELEASE_PART);
        machineSemaphore.acquire();
        glassOnSpot = null;
        conveyor.msgIAmFree();
}
```