Stand Agent

Kitting Cell v0

Data

```
Public class Kit{
KitStatus KS
…
}

enum KitStatus{
AwaitingPickup, PickedUp,
PlacedOnStand, Assembled,
MarkedForInspection,
AwaitingInspection, Inspected,
Shipped
}

Map<Kit k, Int destination>
ReceivedKits
Int numKitsToMake

//Shared data with KitRobot
List<Kit> KitsOnStand

//Prevent collisions (not for v0)
//Shared with PartsRobot
Semaphore AccessKit

KitRobot kitrobot
FCS fcs
```

Messages

```
//From KitRobot
ShippedKit (){
Kits.get(0).KS = Shipped;
}

//From PartsRobot
KitIsAssembled(Kit k){
k.KS = Assembled;
numKitsToMake--;
}

//From FCS
MakeKits(int numKits){
numKitsToMake = numKits;
}

//From KitRobot
HereIsKit(Kit k, int dest){
ReceivedKits.put(k, dest);
}
```

Scheduler

```
// Need to synchronize
If ∃ k in KitsReceived
call PlaceKit(k)

∃ k in KitsOnStand ϵ
k.KS = Shipped
//Always the 0th kit
KitsOnStand.remove(k)

if(KitsOnStand[1] = null ||
KitsOnStand[2] = null)
        (KitsOnStand[1] = null) ?
        RequestKit(1) :
        RequestKit(2)

If ∃ k in KitsOnStand ϵ
k.KS = Assembled
call RequestInspection(k)

if (numKitsToMake = 0)
call FinalizeOrder()
```

Actions

```
RequestKit(int index){
Position pos = new
Position(index);
kitrobot.NeedKit(pos)
}

PlaceKit(Kit k){
Int spot = ReceivedKits.get(k)
ReceivedKits.remove(k);
KitsOnStand.set(spot, k);
k.KS = PlacedOnStand;
}

RequestInspection(Kit k){
kitrobot.MoveKitToInspectionAre
a(k);
}

FinalizeOrder(){
fcs.OrderFinished();
// Don't call statechanged() after
// this
}

//This agent has no associated
//DoXXX animations
```