# Pràctica simulador a mida

# Aeroport de Barcelona-El Prat

**Andreu Orensanz Bargalló**
QT 2022/23
Grup Lab-11
Simulació

# Índex

# 1. Objecte

L'objecte que m'ha tocat representar en aquesta simulació són les unitats d'*air cargo*. Aquestes es tracten d'unes plataformes mecàniques de càrrega per a carregar i descarregar equipatges, mercaderies i correu dins d'unes estructures grans en forma de contenidor a avions de fuselatge ample i a alguns avions de fuselatge estret.


fig. 1: Operari descarregant contenidors LD3 d'un Boeing 747 a l'Aeroport de Zagreb

# 2. Hipòtesis

Per a la realització de l'estudi he considerat un seguit d'hipòtesis que estableixen clarament els processos i dades que utilitzaré en la nostra simulació, aquestes no afectaran en cap cas a les conclusions obtingudes.

## 2.1. Hipòtesis sistèmiques

### 2.1.1. Estructurals

En aquest apartat es recullen totes aquelles consideracions que estan associades a la relació existent entre els diferents conceptes del nostre model conceptual, que alhora del procés d'experimentació es tindran en compte.

- Hi ha senyalitzacions per a guiar a l'*air cargo* i evitar accidents

### 2.1.2. Dades

En aquest apartat es recullen totes aquelles consideracions al voltant de les dades, que alhora del procés d'experimentació es tindran en compte.

- Temps maniobra d'arribada a l'avió (distribució triangular 1 a 5 min). Event start de l'objecte predecessor
- Temps maniobra càrrega (distribució triangular 1 a 5 min)
- Temps maniobra descàrrega (distribució triangular 1 a 5 min)

## 2.2. Hipòtesis simplificadores

En aquest apartat es recullen totes aquelles consideracions que permetran simplificar el model sense perdre el rigor de l'estudi que a l'hora del procés d'experimentació es tindran en compte.

- No es considera la càrrega o descàrrega d'elements que no siguin contenidors amb equipatge o mercaderies
- No es consideren avaries de cap dels components de l'objecte
- No es contempla la càrrega de més contenidors respecte a la capacitat màxima
- Es considera que a l'aeroport es carrega i a l'avió es descarrega quan l'avió s'ha d'enlairar i viceversa quan ha acabat d'aterrar.

# 3. Especificació

En aquest apartat es mostren els diagrames de components, d'estats i processos de l'unitat d'*air cargo*.

## 3.1. Diagrama de components

**Diagrama complet**:

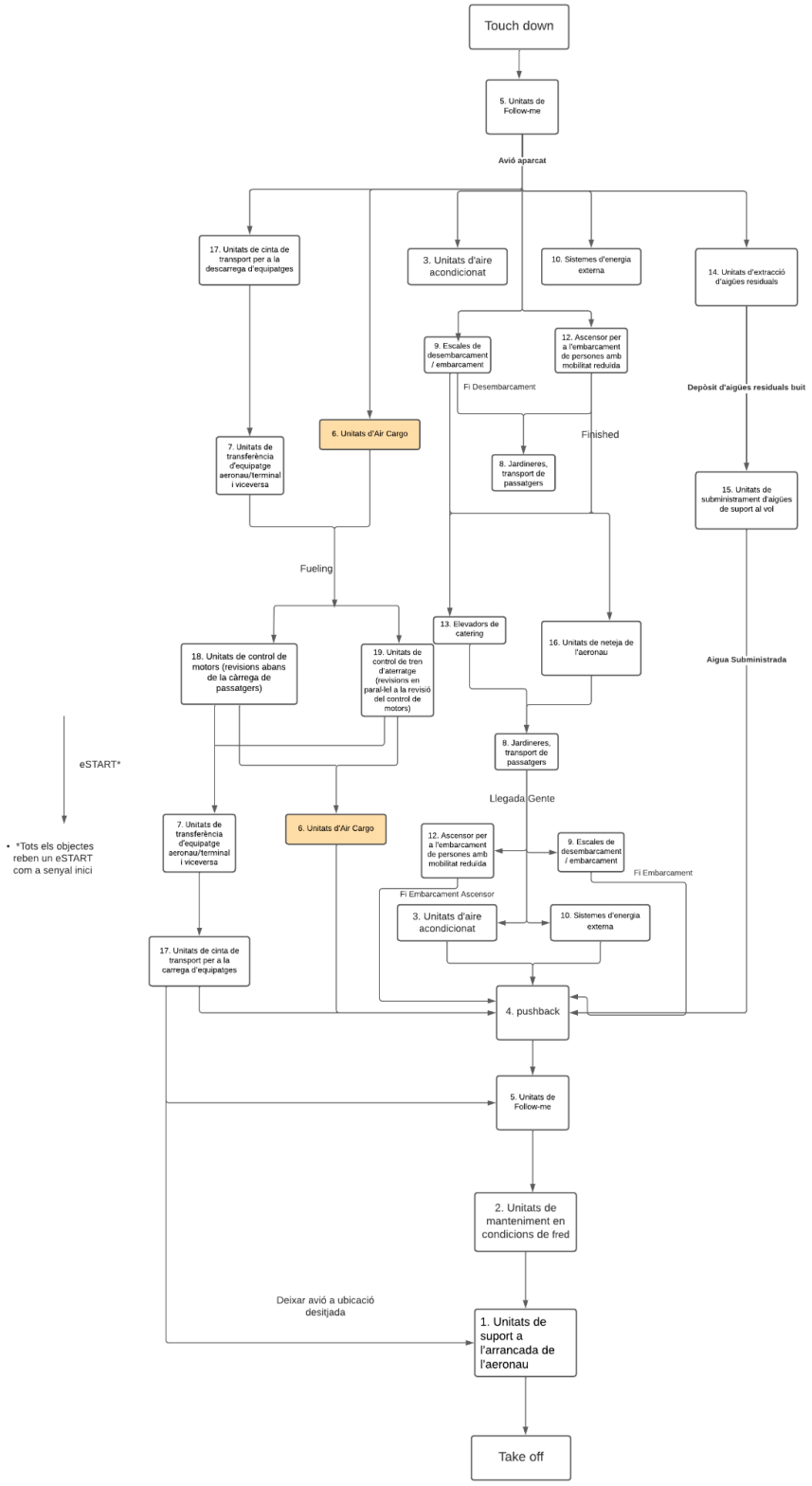
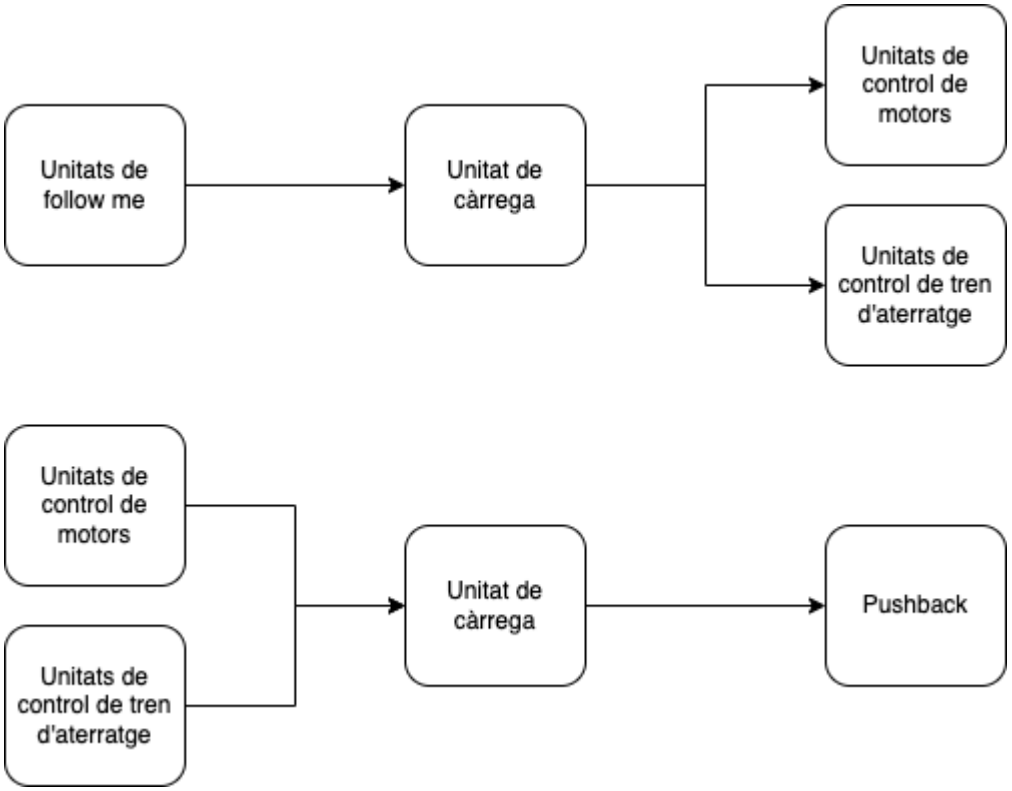
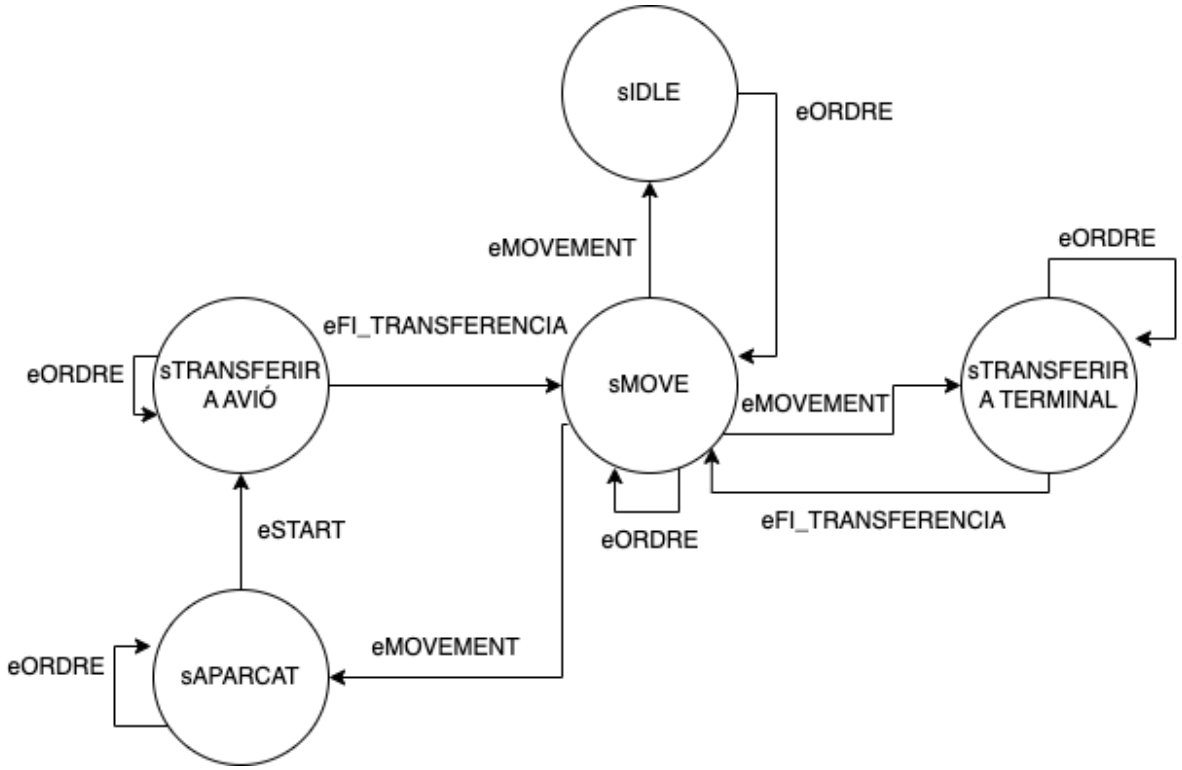


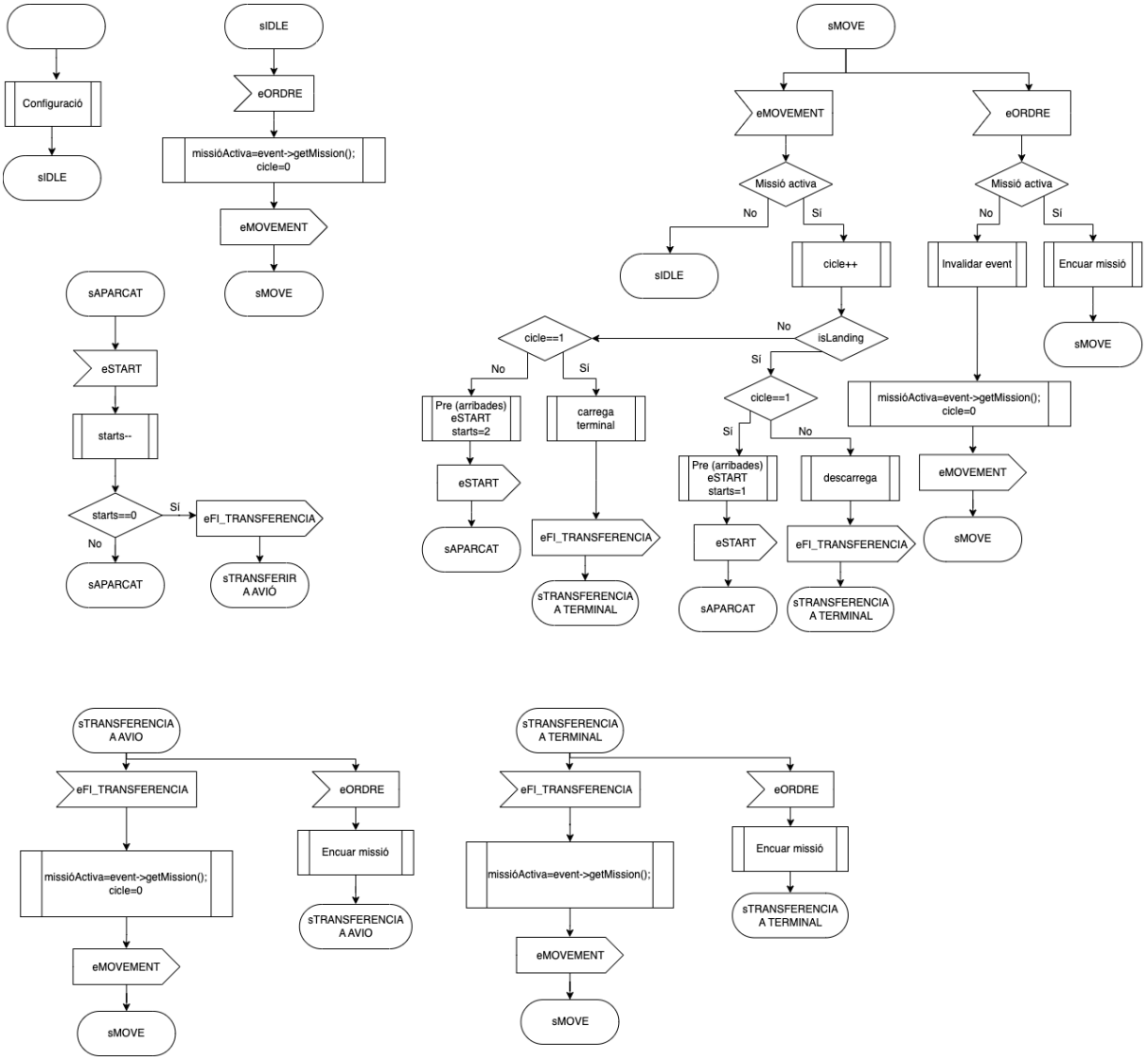
**Diagrama de l'objecte**:




## 3.2. Diagrama d'estats



## 3.3. Diagrama de processos



# 4. Codi

A continuació es mostra el codi del mètode *ProcessEvent* de l'objecte creat classificat pels estats definits al diagrama d'estats anterior:

**sIDLE:**

```
case sIDLE:
    cout << "++++++++++++++++++++++sIDLE++++++++++++++++++++++" << endl;
    switch(event->getEventType()) {
        case eORDRE: {
            cout << "en estat IDLE i rebo una missió" << endl;
            float dist = triangular_random(1, 5, 2.5);
            m_missioActiva=event->getMission();
            m_cicle=0;
            CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,event->getMission(),eMOVEMENT);
            m_Simulator->scheduleEvent(e);
            setState(sMOVE);
            break;
        }
        default:
            break;
    }
    break;
```

**sMOVE**

```
case sMOVE:
    cout << "++++++++++++++++++++++sMOVE++++++++++++++++++++++" << endl;
    switch(event->getEventType()) {
        case eORDRE: {
            if (this->m_missioActiva != NULL) { // Sí
                cout << "Encuar missió" << endl;
                m_missions.push_back(m_missioActiva);
            } else { // No
                cout << "Invalidar element" << endl;
                float dist = triangular_random(1, 5, 2);
                m_cicle=0;
                CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,event->getMission(),eMOVEMENT);
                m_Simulator->scheduleEvent(e);
            }
            break;
        }
```

```
case eMOVEMENT: {
    if (this->m_missioActiva != NULL) { // Sí
        cout << "en estat MOVE i missió activa" << endl;
        m_cicle++;
        if (m_missioActiva->isLanding()) { // Sí
            cout << "avio isLanding()" << endl;
            float dist = triangular_random(1, 5, 2);
            if (m_cicle == 1) { // Sí
                cout << "primer cicle" << endl;
                m_starts=1;
                m_predecessor1->sendMeEvent(new CSimulationEvent(m_Simulator->getCurrentTime()+dist,m_predecessor1,this,event->getMission(),eSTART));
                setState(sAPARCAT);
            } else { // No
                cout << "descarregar" << endl;
                CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,event->getMission(),eFI_TRANSFERENCIA);
                m_Simulator->scheduleEvent(e);
                setState(sTRANSFERIR_A_TERMINAL);
            }
        } else { // No
            cout << "avio !isLanding()" << endl;
            float dist = triangular_random(1, 5, 2);
            if (m_cicle == 1) { // Sí
                cout << "#num cicle " << m_cicle << endl;
                cout << "Carrega terminal" << endl;
                CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,event->getMission(),eFI_TRANSFERENCIA);
                m_Simulator->scheduleEvent(e);
                setState(sTRANSFERIR_A_TERMINAL);
            } else { // No
                cout << "*num cicle " << m_cicle << endl;
                cout << "primer cicle" << endl;
                m_starts=2;
                cout << "m'envien START predecessor2 i predecessor3" << endl;
                m_predecessor2->sendMeEvent(new CSimulationEvent(m_Simulator->getCurrentTime()+dist,m_predecessor2,this,event->getMission(),eSTART));
                m_predecessor3->sendMeEvent(new CSimulationEvent(m_Simulator->getCurrentTime()+dist,m_predecessor3,this,event->getMission(),eSTART));
                setState(sAPARCAT);
            }
        }
    } else { // No
        setState(sIDLE);
    }
    break;
```

## sAPARCAT

```
case sAPARCAT:
    cout << "++++++++++++++++++++++sAPARCAT++++++++++++++++++++++" << endl;
    switch(event->getEventType()) {
        case eSTART: {
            cout << "en estat APARCAT i rebo START del predecessor" << endl;
            m_starts--;
            float dist = triangular_random(1, 5, 2);
            float temps = m_Simulator->getCurrentTime() - m_Simulator->getCurrentTime()+dist;
            if (m_starts == 0) { // Comença a fer missió
                if (m_missioActiva->isLanding()) {
                    cout << "APARCAT Avio a terra" << endl;
                    CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,m_successor1,event->getMission(),eFI_TRANSFERENCIA);
                    m_Simulator->scheduleEvent(e);
                } else {
                    cout << "APARCAT Avio a l'aire" << endl;
                    float dist = triangular_random(1, 5, 2);
                    CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,m_successor2,event->getMission(),eFI_TRANSFERENCIA);
                    m_Simulator->scheduleEvent(e);
                    CSimulationEvent* e1 = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,m_successor3,event->getMission(),eFI_TRANSFERENCIA);
                    m_Simulator->scheduleEvent(e1);
                }
                m_tempsTransferencia.insert(pair<int, float>(m_numMissions, temps));
                m_numMissions++;
                cout << "Inserta transf. Temps transferencia: " << dist << endl;
                setState(sTRANSFERIR_A_AVIO);
            } else {
                cout << "descarregar" << endl;
                setState(sAPARCAT);
            }
            break;
        }
        default:
            break;
    }
    break;
```

## sTRANSFERIR_A_AVIO

```
case sTRANSFERIR_A_AVIO:
    cout << "++++++++++++++++++++++sTRANSFERIR_A_AVIO++++++++++++++++++++++" << endl;
    switch(event->getEventType()) {
        case eFI_TRANSFERENCIA: {
            float dist = triangular_random(1, 5, 2);
            m_cicle = 0;
            m_missioActiva=event->getMission();
            CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,m_missioActiva,eMOVEMENT);
            m_Simulator->scheduleEvent(e);
            setState(sMOVE);
            break;
        }
        case eORDRE: {
            cout << "sTRANSFERIR_A_AVIO Encuar missió. Num missions: " << m_missions.size() << endl;
            m_missions.push_back(m_missioActiva);
            setState(sTRANSFERIR_A_AVIO);
            break;
        }
        default:
            break;
    }
    break;
```

## sTRANSFERIR_A_TERMINAL

```
case sTRANSFERIR_A_TERMINAL:
    cout << "++++++++++++++++++++++++sTRANSFERIR_A_TERMINAL++++++++++++++++++++++++" << endl;
    switch(event->getEventType()) {
        case eFI_TRANSFERENCIA: {
            cout << "Transferencia a terminal " << endl << endl;
            float dist = triangular_random(1, 5, 2);
            m_missioActiva=event->getMission();
            CSimulationEvent* e = new CSimulationEvent(m_Simulator->getCurrentTime()+dist,this,this,event->getMission(),eMOVEMENT);
            m_Simulator->scheduleEvent(e);
            setState(sMOVE);
            break;
        }
        case eORDRE: {
            cout << "sTRANSFERIR_A_TERMINAL Encuar missió" << endl;
            m_missions.push_back(m_missioActiva);
            cout << m_missions.size() << endl;
            setState(sTRANSFERIR_A_TERMINAL);
            break;
        }
        default:
            break;
    }
    break;
```