

# **FLUXVATOR control Simulator**

**OSP stage 2040 ver. 3**

May. 27<sup>th</sup> 2014

Team 3

Team Organization

200913215 기세파

201013275 강태호

201013760 이인구

## Index

1. Activity 2041. Design Real Use-case
  - 1.1 탑승요청(Request Aboard)
  - 1.2 층 선택(Select Level)
  - 1.3 문 제어 요청(Request Door)
  - 1.4 요청 취소(Cancel Request)
  - 1.5 상태 시뮬레이트(Config State)
  - 1.6 최대 하중 조정(Config Maxload)
  - 1.7 문 열기(Open Door)
  - 1.8 문 닫기(Close Door)
  - 1.9 움직임 제어(Movement Control)
  - 1.10 상황판별(Determine State)
  - 1.11 하중판별(Determine Load)
  - 1.12 Flux계산(Flux Calculation)
  - 1.13 큐 할당(Enqueue Order)
  - 1.14 큐 삭제(Dequeue Order)
  - 1.15 화재 컨트롤(Fire Control)
  - 1.16 정전 컨트롤(Blackout Control)
  - 1.17 문 제어 계산(Door Control Simulation)
  - 1.18 큐 읽기(Read Queue)
2. Activity 2042 Define Reports, UI and Storyboard
  - 2.1 Entire GUI
  - 2.2 요청 메뉴
  - 2.3 시뮬레이트 메뉴
3. Activity 2043 Refine System Architecture
4. Activity 2044 Define Interaction Diagrams
  - 4.1 탑승 요청(Request Aboard)
  - 4.2 층 선택(Select Level)
  - 4.3 문 제어 요청(Request Door)
  - 4.4 요청 취소(Cancel Request)
  - 4.5 상태 시뮬레이트(Config State)
  - 4.6 최대 하중 조정(Config Maxload)
5. Activity 2045 Define Design Class Diagram

# 1. Activity 2041 Design Real Use-case

## 1.1 탑승 요청(Request Aboard)

|                           |   |
|---------------------------|---|
| Use Case                  | 1.1.1 탑승요청  |
| Actor                     | System  |
| Purpose                   | 층을 선택하여 상승 혹은 하강 요청   |
| Overview                  | 층을 선택하여 상승 혹은 하강 요청을 한뒤, 현재 시행되고 있는 큐와 비교하여, 우선적인 것을 실행하도록 함.   |
| Type                      | Primary and essential   |
| Cross Reference           | 3.1.1 3.2.1   |
| Pre-Requisites            | 현재 엘리베이터들의 상태가 정상임  |
| Typical Courses of Events | <p>(A) : Actor, (S) : System</p> <ol style="list-style-type: none"> <li>(A) : 유저는 UI를 통해 requestAboard(int level, int direction, int person)을 선택해서, Simulator Controller에 보낸다.</li> <li>(S) : Simulator Controller에서 int getState()로 Elevator의 상태를 확인하여, request(int destination, int requestID, int person), 아니면 메시지를 유저한테 보내준다.</li> <li>(S) : 유저가 보낸 정보를 토대로 Enqueue를 수행한다.</li> <li>(S) : 엘리베이터의 현재 방향과, 목적 층을 비교하여 가장 가까운 큐노드를 찾는다. 이 큐노드를 현재 실행중인 큐노드와 비교하여 더 가까운 큐노드를 elevator1(2)DestinationNode에 저장하여 주고, 목적지를 엘리베이터의 목적지로 설정해준다.</li> <li>(S) : 엘리베이터는 목적지와 엘리베이터의 위치를 이용하여 checkDirection을 수행한다. checkDirection값은 지속적으로 directionDelta에 저장된다. checkDirection의 결과가 0이 되면 멈춘 것으로 간주하여 다음 과정인 arrivalCalibration을 수행한다.</li> <li>(S) : 엘리베이터는 멈추고 addLoad()를 수행하여 해당 큐의 하중을 추가한다. 그 후 큐알고리즘에 해당큐의 dequeue를 요청하여 해당큐를 dequeue를 수행하게 한다.</li> <li>(S) : 엘리베이터는 문이 열린채로 지정한 시간 동안 기다린뒤, departureCalibration을 수행한다. 즉 다음 수행될 노드를 4번과정과 같은 방식으로 확인 후, 수행하게 된다.</li> </ol> |

|                               |   |
|-------------------------------|---|
| Alternative Courses of Events | <p>4. 엘리베이터의 방향과 찾게 된 AboardQueueNode의 방향이 같을 경우 AboardQueueNode를 elev1(2)DestinationNode에 저장하여 주고, SelectLevelQueueNode는 elev1(2)DestinationNode에 저장하여 준다.</p> <p>6. directionDelta와 elev1(2)DestinationNode에 저장된 노드의 방향이 같으면 Dequeue를 요청할 elev1(2)DestinationNode에 저장된 노드도 Dequeue를 요청한다.</p> |
| Exceptional Courses of Events | <p>목적 층과 방향이 동일한 탑승요청이 이미 큐에 존재할 경우 중복되는 해당 탑승요청은 무시하도록 한다 : 똑같은 탑승요청을 중복해서 할 수 있다는건 조금 말이 안되보이므로 허용하지 않기로 결정하였다.</p> <p>엘리베이터 층수의 경우 정수값으로 설정되어 있다. 건물에 따라 층수가 다르다는 점을 고려하여 현재로선 최대/최저 층을 설정해 두진 않았다. 또한 0층도 존재할 수 있다고 정해두었다.</p>  |

## 1.2 층 선택(Select Level)

|                           |  |
|---------------------------|--|
| Use Case                  | 1.1.2 층 선택   |
| Actor                     | System   |
| Purpose                   | 층 선택을 하여 하차 요청.  |
| Overview                  | 층 선택을 하여 하차 요청을 한 뒤, 현재 시행되고 있는 큐와 비교하여, 우선적인 것을 실행하도록 함.  |
| Type                      | Primary and essential  |
| Cross Reference           | 3.1.1 3.2.1  |
| Pre-Requisites            | 현재 엘리베이터들의 상태가 정상임   |
| Typical Courses of Events | <p>(A) : Actor, (S) : System</p> <ol style="list-style-type: none"> <li>(A) : 유저는 UI를 통해 selectLevel(int level, int elevatorID, int person)을 선택해서, Simulator Controller에 보낸다.</li> <li>(S) : Simulator Controller에서 int getState()로 Elevator의 상태를 확인하여, request(int destination, int requestID, int person), 아니면 메시지를 유저한테 보내준다.</li> <li>(S) : 유저가 보낸 정보를 토대로 Enqueue를 수행한다.</li> <li>(S) : 엘리베이터의 현재 방향과, 목적 층을 비교하여 가장 가까운 큐노드를 찾는다. 이 큐노드를 현재 실행중인 큐노드와 비교하여 더 가까운 큐노드를 elev1(2)DestinationNode에 저장하여 주고, 목적지를 엘리베이터의 목적지로 설정해준다.</li> <li>(S) : 엘리베이터는 목적지와 엘리베이터의 위치를 이용하여 checkDirection을 수행한다. checkDirection이 0가되면 멈춘것으로 간주하여 다음 과정인 arrivalCalibration을 수행한다.</li> <li>(S) : 엘리베이터는 멈추고 addLoad()를 수행하여 해당 큐의 하중을 제</li> </ol> |

|                               |  |
|-------------------------------|--|
|                               | <p>거한다. 그 후 큐알고리즘에 해당큐의 dequeue를 요청하여 해당큐를 dequeue를 수행하게 한다.</p> <p>7. (S) : 엘리베이터는 문이 열린채로 지정한 시간 동안 기다린뒤, departureCalibration을 수행한다. 즉 다음 수행될 노드를 4번과정과 같은 방식으로 확인 후, 수행하게 된다.</p>  |
| Alternative Courses of Events | <p>4. 엘리베이터의 방향과 찾게 된 AboardQueueNode의 방향이 같을 경우 AboardQueuenode를 elev1(2)NDestinationNode에 저장하여 주고, SelectLevelQueueNode는 elev1(2)DestinationNode에 저장하여 준다.</p> <p>6. directionDelta와 elev1(2)DestinationNode에 저장된 노드의 방향이 같으면 Dequeue를 요청할 elev1(2)DestinationNode에 저장된 노드도 Dequeue를 요청한다.</p> |
| Exceptional Courses of Events |  |

### 1.3 문 제어 요청(Request Door)

|                  |  |
|------------------|--|
| Use Case Name    | 3. Door Request  |
| Actor            | 사용자  |
| Purpose          | 이전 유스케이스(Essential, Business)참조  |
| Overview         | 이전 유스케이스(Essential, Business)참조  |
| Type             | Primary & Essential  |
| Prerequisite     | 엘리베이터의 int stoppedOrMoving이 stopped값과 동일함.   |
| References       | 1.1.3, 2.1.1. 2.1.2, 2.1.3, 3.4  |
| Normal Flow      | <p>1. (A) : 유저가 UI의 문 열기/닫기 버튼을 누른다.</p> <p>2. (S) :컨트롤러에서 doorControlRequest()가 요청되며, 문제어 요청이 문 열기 요청인지 문 닫기 요청인지를 requestID로 전달한다.</p> <p>3. (S) : 엘리베이터로부터 현재 엘리베이터가 이동중인지 아닌지를 전달받는다.</p> <p>4. (S) 엘리베이터가 정지되어 있을 경우 엘리베이터 내의 DoorControl 기능 중 handleDoorRequest()가 호출된다.</p> <p>5. (S) : 이 다음부터의 변화는 18.door control calculation의 handleDoorReqeust() 패턴을 따른다.</p> |
| Alternative Flow |  |
| Exception Flow   | 이동중일 경우 할수 없다는 메시지 표시  |

### 1.4 요청 취소(Cancel Request)

|          |                       |
|----------|-----------------------|
| Use Case | 1.1.4 요청 취소           |
| Actor    | System                |
| Purpose  | 유저가 이전의 요청을 취소하게 해준다. |

|                               |  |
|-------------------------------|--|
| Overview                      | 유저가 이전의 요청에 대한 큐를 요청값에 따라, 큐에서 찾아서 제거해준다.  |
| Type                          | Primary and essential  |
| Cross Reference               | 3.1.1 3.2.1  |
| Pre-Requisites                | 엘리베이터가 정상상태임   |
| Typical Courses of Events     | <p>(A) : Actor, (S) : System</p> <ol style="list-style-type: none"> <li>(A) : 유저는 UI를 통해 cancelRequest(int level, int requestID, int elevatorID)을 선택해서, Simulator Controller에 보낸다.</li> <li>(S) : Simulator Controller에서 int getState()로 Elevator의 상태를 확인하여, 정상이면 진행하고, 아니면 메시지를 유저한테 보내준다.</li> <li>(S) : Queue Algorithm에서 요청값에 따라 Queue내에 있는 Queuenode를 찾고 dequeue해준다.</li> <li>(S) : 이하는 requestAboard와 selectLevel의 4번 이후와 동일하다.</li> </ol> |
| Alternative Courses of Events |  |
| Exceptional Courses of Events | 해당 요청이 없을 경우 요청이 없음을 알림<br>requestID에 load를 쓴 것으로 추정될 경우(3 이상의 값들)<br>requestID로 쓰지 않음을 알림   |

### 1.5 상태 시뮬레이트(Config State)

|               |   |
|---------------|---|
| Use Case Name | 5. Config State   |
| Actor         | 사용자   |
| Purpose       | 이전 유스케이스(Essential, Business)참조   |
| Overview      | 이전 유스케이스(Essential, Business)참조   |
| Type          | Primary & Essential   |
| Prerequisite  |   |
| References    | 3.1.1, 3.1.2, 3.3.1, 3.3.2, 3.4   |
| Normal Flow   | <p>(A) : Actor (S) : System</p> <ol style="list-style-type: none"> <li>(A) :사용자는 ElevatorController를 통해 엘리베이터의 현재 상태를 1.정상 2.화재 3. 정전 4.수리 중 어떤 것으로 바꿀 것인지 알린다. 정상,정전,수리 = setStateRequest(stateID), 화재, = setStateRequest(stateID, stateArg)를 통해 1개 혹은 2개의 값을 넘기며, stateID는 4가지 상태중 어떤 상태인지, stateArg는 화재일 경우 몇 층에서 불이 난 것으로 정의할 것인지 정의하기 위한 정수값을 전달한다.</li> <li>(S) :사용자의 요청은 상태 판별 기능이 존재하는 Elevator로 전달된다.</li> </ol> |

|  |  |
|--|--|
|  | <p>사용자의 요청이 어떤 요청인가에 따라 다음과 같이 나뉜다:</p> <p>A. 요청한 상태가 정상일 경우</p> <p>A1. (S) : <code>getCurrentState()</code>를 통해 현재 상태가 요청한 상태와 같은지 확인한다.</p> <p>A2-1. (S) : 요청한 상태와 현재 상태가 다를 경우 아무것도 하지 않으며 이미 상태가 정상이다 라는 메시지를 전달한다.</p> <p>A2-2. (S) : 요청한 상태가 정상이나 현재 상태는 화재/정전/수리일 경우 elevator 내 <code>setState(stateID)</code>를 통해 현재 상태를 정상으로 변경한다.</p> <p>A3. 요청한 상태가 정상이 되어서 평소처럼 탑승요청, 층선택, 문열기/닫기 등 사용자의요청을 받아들인다.</p> <p>B. 요청한 상태가 화재일 경우</p> <p>B1. (S) : <code>getCurrentState()</code>를 통해 요청한 상태와 현재 상태를 비교한다.</p> <p>B2-1. (S) : 동일할 경우 추가로 아무것도 하지 않으며 이미 그 상태가 선택된 상태임을 알린다.</p> <p>B2-2. (S) : 이전 상태가 동일하진 않으나 정전 혹은 수리일 경우 이미 비정상 상태여서 상태를 변환할 수 없음을 알린다.</p> <p>B2-3. (S) : 이전 상태가 정상이었을 경우 각 엘리베이터의 <code>setCurrentState(stateID)</code>를 통해 상태를 화재로 변환한다.</p> <p>B3. (S) : <code>QueueAlgorithm</code> 내의 <code>handleFireControl()</code>을 호출하여 화재 프로토콜을 진행한다.</p> <p>B4. 화재 상태에서 정상 상태를 새로 누르면 다시 정상 상태로 되돌린다.</p> <p>C. 요청한 상태가 정전일 경우</p> <p>C1. (S) : <code>getCurrentState ()</code>를 통해 요청한 상태와 현재 상태를 비교한다.</p> <p>C2-1. (S) : 이전상태가 동일하거나 화재 혹은 수리일 경우 이미 비정상 상태여서 상태를 변환할 수 없음을 알린다.</p> <p>C2-2. (S) : 이전 상태가 정상이었을 경우 각 엘리베이터의 <code>setCurrentState(stateID)</code>를 통해 상태를 정전으로 변환한다.</p> <p>C3. (S) : <code>QueueAlgorithm</code>내의 <code>handleBlackoutControl()</code>을 호출하여 정전 프로토콜을 진행한다.</p> <p>C4. 정전 상태에서 정상 상태를 새로 설정하면 다시 원래대로 움직인다.</p> <p>D. 요청한 상태가 수리일 경우</p> <p>D1. (S) : <code>getCurrentState()</code>를 통해 요청한 상태와 현재 상태를 비교한다.</p> <p>D2-1. (S) : 이전상태가 동일하거나 화재 혹은 정전일 경우 이미 비정상 상태여서 상태를 변환할 수 없음을 알린다.</p> <p>D2-2. (S) : 이전 상태가 정상이었을 경우 <code>setCurrentState(stateID)</code>를 통해 해당 엘리베이터의 상태를 수리로전환하고, 해당 엘리베이터의 <code>Queue</code>가 더 이상 요청을 받지 않도록 하며 아직 큐에 남아있는 요청들을 진행한다.</p> <p>D2-3. (S) 큐에 마저 남은 요청들은 진행한 뒤에는 상태가 다시 정상으로</p> |
|--|--|

|                  |   |
|------------------|---|
|                  | <p>전환되기 전까지 모든 추가적인 요청을 무시한다.</p> <p>D3. 정전 상태에서 다시 정상 상태로 변경되면 새로 요청을 받기 시작한다.</p> |
| Alternative Flow |   |
| Exception Flow   | Normal flow 참조  |

### 1.6 최대 하중 조정(Config Maxload)

|                  |  |
|------------------|--|
| Use Case Name    | 6. Config MaxLoad  |
| Actor            | 사용자  |
| Purpose          | 이전 유스케이스(Essential, Business)참조  |
| Overview         | 이전 유스케이스(Essential, Business)참조  |
| Type             | Primary & Essential  |
| Prerequisite     |  |
| References       | 1.2.2  |
| Normal Flow      | <p>(A) : Actor (S) : System</p> <ol style="list-style-type: none"> <li>1. (A) : 사용자가 UI를 통해 setMaxLoadRequest(setLoad)를 호출한다. setLoad로 원하는 최대 하중을 넘겨준다.</li> <li>2. (S) : 각 Elevator 내의 setMaxLoad()를 호출하게 되고, 이는 다음을 진행한다.</li> <li>3. (S) : 현재 하중과 새로 정의하는 최대 하중을 비교하게 되고, 결과는 다음 처럼 나뉜다. <ol style="list-style-type: none"> <li>3-1. (S) : 현재 하중이 새로 정의하는 최대 하중보다 클 경우 이에 해당하는 엘리베이터에 대해서는 새로운 최대 하중을 적용하지 않게 되고, 사용자에게 현재 하중이 새로 정의할 최대 하중보다 크므로 최대 하중 변경을 할 수 없음을 알린다.</li> <li>3-2. (S) : 현재 하중이 새로 정의하는 최대 하중보다 작을 경우 각 엘리베이터의 최대 하중이 변경되게 된다.</li> </ol> </li> </ol> |
| Alternative Flow |  |
| Exception Flow   | <p>음수값 등 비정상적 값을 적을 경우 적용 안됨을 알림</p> <p>최대하중 설정값 0 의 경우 비실용적이나 현실적으로 가능하긴 하다고 판단하여 냅두기로 설정</p>   |

### 1.7 문 열기(Open Door)

|          |                             |
|----------|-----------------------------|
| Use Case | 2.1.1 문 열기(Open Door)       |
| Actor    | System                      |
| Purpose  | 엘리베이터의 문 열기 동작              |
| Overview | 명령이 들어올 경우 엘리베이터 문을 열도록 한다. |



|                               |   |
|-------------------------------|---|
| Type                          | Primary and Essential   |
| Cross Reference               | 3.4, 2.1.1  |
| Pre-Requisites                | Elevator의 int stoppedOrMoving 값이 stopped의 값과 동일   |
| Typical Courses of Events     | (A) : Actor (S) : System<br>1. (A)or(S) : 문 제어 요청 Use-case나 문제어계산 Use-case 의 arrivalCalibration() 으로 문열기 Use-case가 호출됨.<br>2. (S) : int doorState의 값을 open의 값과 동일하게 바꾸어 준다. |
| Alternative Courses of Events |   |
| Exceptional Courses of Events |   |

### 1.8 문 닫기(Close Door)

|                               |  |
|-------------------------------|--|
| Use Case                      | 2.1.2 문 닫기(Close Door)   |
| Actor                         | System   |
| Purpose                       | 엘리베이터의 문 닫기 동작   |
| Overview                      | 명령이 들어올 경우 엘리베이터 문을 닫도록 한다.  |
| Type                          | Primary and Essential  |
| Cross Reference               | 3.4, 2.1.1   |
| Pre-Requisites                | Elevator의 int stoppedOrMoving 값이 stopped의 값과 동일  |
| Typical Courses of Events     | (A) : Actor (S) : System<br>1. (A)or(S) : 문 제어 요청 Use-case나 문제어계산 Use-case 의 departureCalibration() 으로 문닫기 Use-case가 호출됨.<br>2. (S) : int doorState의 값을 close의 값과 동일하게 바꾸어 준다. |
| Alternative Courses of Events |  |
| Exceptional Courses of Events |  |

### 1.9 움직임 제어 (Movement Control)

|               |                                 |
|---------------|---------------------------------|
| Use Case Name | 9. MovementControl              |
| Actor         | System                          |
| Purpose       | 이전 유스케이스(Essential, Business)참조 |
| Overview      | 이전 유스케이스(Essential, Business)참조 |

|                  |   |
|------------------|---|
| Type             | Primary & Essential   |
| Prerequisite     |   |
| References       | 2.1.3, 3.2.2, 3.2.3, 3.4, 1.1.2   |
| Normal Flow      | <ol style="list-style-type: none"> <li>1. movementControl()은 엘리베이터의 stoppedOrMoving 변수(움직임/정지를 나타내주는 변수)가 1일 경우 항상 호출된다.</li> <li>2. checkDirection을 호출하여 현재 엘리베이터의 방향을 확인한다. 이때 방향은 엘리베이터의 목적층 - 현재위치의 값으로 결정된다.</li> <li>3. 방향이 상승/하강일 경우 그에 맞춰 한층을 오르/내린다.</li> <li>4. 또한 방향이 상승/하강일 경우 현재의 방향을 directionDelta로 저장한다.</li> <li>5. 방향이 0일 경우 arrivalCalibration을 호출한다.</li> <li>6. Arrivalcalibration은 엘리베이터를 정지시키고 그에따른 추가적인 진행을 시킨다.</li> </ol> |
| Alternative Flow |   |
| Exception Flow   |   |

### 1.10 상황판별(Determine State)

|                               |  |
|-------------------------------|--|
| Use Case                      | 3.1.1 상황판별(Determine State)  |
| Actor                         | System   |
| Purpose                       | 엘리베이터의 현재 상태(화재/정전/정상/수리)를 파악한다.   |
| Overview                      | Simulator controller가 엘리베이터의 현재 상태(화재/정전/정상/수리)를 파악한다.   |
| Type                          | Primary and essential  |
| Cross Reference               | 3.1.1, 1.2.1   |
| Pre-Requisites                | 시뮬레이터가 작동 중  |
| Typical Courses of Events     | <p>(A) : Actor, (S) : System</p> <ol style="list-style-type: none"> <li>1. (A) : 유저가 탑승요청, 층선택, 요청 취소등의 요청을 보낸다.</li> <li>2. (S) : Simulator Controller가 엘리베이터의 상태를 int currentStated 통하여 파악한다.</li> <li>3. (S) : 정상일경우 Normal Flow를 진행시키게 하고, 아닐경우 Exceptional Flow를 진행시키게 한다.</li> </ol> |
| Alternative Courses of Events |  |
| Exceptional Courses of Events |  |

### 1.11 하중판별(Determine Load)

|                               |  |
|-------------------------------|--|
| Use Case                      | 3.1.1 상황판별(Determine State)  |
| Actor                         | System   |
| Purpose                       | 이전 유스케이스(Essential, Business)참조  |
| Overview                      | 이전 유스케이스(Essential, Business)참조  |
| Type                          | Primary and essential  |
| Cross Reference               | 1.2.2, 3.2.1   |
| Pre-Requisites                | 시뮬레이터가 작동 중  |
| Typical Courses of Events     | (A) : Actor, (S) : System<br>1. (S) : 큐 알고리즘이 큐 노드를 수행하도록 엘리베이터에 요청한다..<br>2. (S) : 엘리베이터가 큐노드를 수행하고 addLoad()를 수행한다.<br>3. (S) : 엘리베이터의 int maxLoad값과 엘리베이터의 int currentLoad + addLoad()값을 비교하여 후자가 작을경우 addLoad()를 수행한다. |
| Alternative Courses of Events | 3. 후자가 더 클 경우 addLoad()내부의 parameter인 int person값을 줄여가면서 수행하고, 수행하지 못한 int person값과 기존의 큐노드들의 나머지 정보들로 enqueue를 요청한다.  |
| Exceptional Courses of Events |  |

### 1.12 Flux계산(Flux Calculation)

|                           |  |
|---------------------------|--|
| Use Case                  | 3.2.1. Flux계산(Flux Calculation)  |
| Actor                     | System   |
| Purpose                   | 이전 유스케이스(Essential, Business)참조  |
| Overview                  | 이전 유스케이스(Essential, Business)참조  |
| Type                      | Primary and essential  |
| Cross Reference           | 3.2.2, 1.1.2, 1.1.4, 2.1.3, 3.2.1, 3.3.1, 3.3.2  |
| Pre-Requisites            | 시뮬레이션 작동 중   |
| Typical Courses of Events | (A): Actor (S): System<br>1. (A) or (S) : 탑승요청, 층 선택, 요청 취소의 요청이나 departureCalibration()의 다음명령 수행 요청.<br>2. (S) : 큐 읽기의 일반상황과, 대체상황을 수행하여 두 개의 Queuenode를 비교하여 현재 엘리베이터의 int elevatorPostion값에 더 가까운 Queuenode를 elev1(2)DestinationNode에 할당한다.<br>3. (S) : 엘리베이터가 elev1(2)DestinationNode의 큐노드를 실행 |

|                               |   |
|-------------------------------|---|
| Alternative Courses of Events | 2. 두개의 큐노드가 같은 int destination값을 지닐 경우, selectLevel의 Queuenode를 우선적으로 elev1(2)DestinationNode에 할당하고, AboardQueueNode를 elev1(2)NDestinationNode에 할당한다. |
| Exceptional Courses of Events | 목적 층과 방향이 동일한 탑승요청이 이미 큐에 존재할 경우 중복되는 해당 탑승요청은 무시하도록 한다 : 똑같은 탑승요청을 중복해서 할 수 있다는건 조금 말이 안되보임.   |

### 1.13 큐 할당(Enqueue Order)

|                               |   |
|-------------------------------|---|
| Use Case                      | 3.2.2. 큐 할당 (Enqueue Order)   |
| Actor                         | System  |
| Purpose                       | 엘리베이터 운행 큐에 새로운 명령을 할당 한다.  |
| Overview                      | 엘리베이터 운행 큐에 새로운 명령을 할당 한다.  |
| Type                          | Primary and essential   |
| Cross Reference               | 3.2.2, 1.1.2, 1.1.4, 2.1.3, 3.2.1, 3.3.1, 3.3.2   |
| Pre-Requisites                | 시뮬레이션 작동 중  |
| Typical Courses of Events     | (A): Actor (S): System<br>1. (A) or (S) : 탑승요청, 층 선택의 요청이나, 화재 컨트롤, 정전컨트롤 등이 큐 할당의 요청을 함.<br>2. (S) : 해당 요청들이 전달한 parameter에 따라 QueueNode를 만들어준다.<br>3. (S) : 알맞은 큐에 할당시켜줌. |
| Alternative Courses of Events |   |
| Exceptional Courses of Events | 목적 층과 방향이 동일한 탑승요청이 이미 큐에 존재할 경우 중복되는 해당 탑승요청은 무시하도록 한다 : 똑같은 탑승요청을 중복해서 할 수 있다는건 조금 말이 안되보임.   |

### 1.14 큐 삭제(Dequeue Order)

|                 |   |
|-----------------|---|
| Use Case        | 3.2.3. 큐 삭제 (Dequeue Order)                     |
| Actor           | System  |
| Purpose         | 엘리베이터 운행 큐에서 큐노드를 삭제한다.                         |
| Overview        | 엘리베이터 운행 큐를 요청에 따라 적합한 큐노드를 삭제해준다.              |
| Type            | Primary and essential                           |
| Cross Reference | 3.2.3, 1.1.1, 1.1.4, 2.1.3, 3.2.1, 3.3.1, 3.3.2 |
| Pre-Requisites  | 시뮬레이션 작동 중                                      |

|                               |  |
|-------------------------------|--|
| Typical Courses of Events     | (A): Actor (S): System<br>1. (A)or(S) : 유저의 요청취소 요청이나 엘리베이터의 큐노드의 수행에 따른 큐노드에 대한 큐 삭제 요청.<br>2. (S) : 해당 요청에서 정보를 읽어 같은 큐노드를 찾음.<br>3. (S) : 해당 큐노드의 prev큐노드의 next값을 해당 큐노드의 next값으로 할당해주고, 해당 큐노드의 next큐노드의 prev값을 해당 큐노드의 prev값으로 할당해준다.<br>4. 해당 큐노드를 삭제한다. |
| Alternative Courses of Events | 1. (A) : 유저가 비상상황을 상태 시뮬레이트를 통해 일으킨다.<br>2. (S) : 현재 실행되는 큐노드를 포함한 모든 큐노드를 큐에서 삭제  |
| Exceptional Courses of Events |  |

### 1.15 화재 컨트롤(Fire Control)

|                             |  |
|-----------------------------|--|
| Use case                    | 3.3.1 화재컨트롤  |
| Actor                       | System   |
| Purpose                     | 화재상황 발생 시에 주어진 행동을 실행한다.   |
| Overview                    | 화재상황 발생 시에 이미 존재하는 모든 큐를 삭제하고 상황에 따라 그에 대응하는 큐를 넣어준다.  |
| Type                        | Secondary  |
| Cross Reference             | 1.2.1, 3.2.2, 3.2.3 1.1.1, 1.1.2, 1.1.3, 1.1.4   |
| Pre-Requisite               | 상태시뮬레이트로 화재를 발생시킴  |
| Typical course of event     | (A) : Actor, (S) : System<br>1. (A) : 유저가 상태 시뮬레이트를 이용하여 int stateArg값을 이용하여 화재 층수 값을 넘겨주면서 화재를 일으킨다.<br>2. (S) : 현재 존재하는 모든 큐 삭제 Use-case를 이용하여 삭제한다,<br>3. (S) : 엘리베이터로부터 int currentPosition을 통하여 현재 층을 읽고 화재가 일어난 층을 제외한 가장 가까운 층에 모든 하중을 하차시키는 큐를 enqueue시킨다.<br>4. (S) : 화재층수와 엘리베이터 현재 층수를 비교하여 현재 층수가 더 낮을 경우, 추락을 방지하기 위해 1층으로 엘리베이터를 이동시키는 큐노드를 enqueue시킨다. 그렇지 않으면 아무것도 하지 않는다.<br>5. QueueNode를 수행하게 한다. |
| Alternative Course of event | 화재가 1층에서 났을 경우<br>1. (S) : Line 4를 수행하지 않음   |

|                             |  |
|-----------------------------|--|
| Exceptional Course of event |  |
|-----------------------------|--|

### 1.16 정전 컨트롤(Blackout Control)

|                             |  |
|-----------------------------|--|
| Use case                    | 3.3.2 정전 컨트롤   |
| Actor                       | System   |
| Purpose                     | 정전상황 발생 시에 주어진 행동을 실행한다.   |
| Overview                    | 정전상황 발생 시에 이미 존재하는 모든 큐를 삭제하고 상황에 따라 그에 대응하는 큐를 넣어준다.  |
| Type                        | Secondary  |
| Cross Reference             | 1.2.1, 3.2.2, 3.2.3 1.1.1, 1.1.2, 1.1.3, 1.1.4   |
| Pre-Requisite               | 상태시뮬레이트로 정전을 발생시킴  |
| Typical course of event     | (A) : Actor, (S) : System<br>1. (A) : 유저가 상태 시뮬레이트를 이용하여 정전을 일으킨다.<br>2. (S) : 엘리베이터의 currentState값을 정전값과 동일하게 바꾸어줌.<br>3. (S) : 현재 존재하는 모든 큐 삭제 Use-case를 이용하여 삭제한다,<br>4. (S) : 현재 엘리베이터이 층을 int elevatorPosition을 통하여 읽고 가장 가까운 곳에 모든 인원을 하차 시키는 큐노드를 enqueue시킨다.<br>5. (S) : QueueNode를 수행시킨다. |
| Alternative Course of event |  |
| Exceptional Course of event |  |

### 1.17 문 제어 계산(Door Control Calibration)

|               |   |
|---------------|---|
| Use Case Name | 18. Door Control Calibration                        |
| Actor         | System  |
| Purpose       | 이전 유스케이스(Essential, Business)참조                     |
| Overview      | 이전 유스케이스(Essential, Business)참조                     |
| Type          | Primary & Essential                                 |
| Prerequisite  |   |
| References    | 2.1.2, 2.1.1, 2.1.3, 3.1.2, 3.1.1                   |
| Normal Flow   | 문 제어 요청시 요청의 타당성을 검토하며 엘리베이터 출발 및 도착시 자동 문제어를 정의한다. |

|                  |   |
|------------------|---|
|                  | <p>A.문제어 요청의 경우handleDoorRequest()<br/> A1. handleDoorRequest의 input으로 requestType을 받는다. (문열기/닫기)<br/> A2. 엘리베이터의 현재 움직임 상태를 확인하여 움직이는 중이면 요청을 무시하고 메시지를 되돌려 보낸다.<br/> A3. 멈춰있는 중이면 다음과 같이 처리한다<br/> A4-1. 문열기인 경우 먼저 문이 열려있는지 닫혀있는지 확인한다.<br/> A4-2. 문이 닫혀있을 경우 openDoor()를 호출하여 문 상태를 open으로 바꿔준다.<br/> A4-3. standby()를 호출한다.<br/> A5-1 문닫기인 경우 문이 열려있는지 닫혀있는지 확인한다.<br/> A5-2. 문이 닫혀있을 경우 아무것도 하지 않으며 메시지를 되돌린다.<br/> A5-3. 문이 열려있을 경우 closeDoor()를 호출하여 문 상태를 닫힘으로 바꿔준다.<br/> A5-4 departurecalibration()을 호출하여 출발준비한다.</p> <p>B.엘리베이터 도착의 경우arrivalCalibration()<br/> B1. 엘리베이터에서 stop()이 수행된다.<br/> B2. openDoor()를 호출한다.<br/> B3. 큐알고리즘에 저장된 목적지 노드에서 탑승자 정보를 불러온다.<br/> B4. addLoad()로 탑승자를 태우거나 내린다.<br/> B5. 해당요청을 deQueue하여 삭제한다.</p> <p>C. 엘리베이터 문을 닫은 경우(출발 대기의 경우) departureCalibration()<br/> C1. Wait()로 n초간 대기한다.<br/> C2. 큐알고리즘의 setNextDestinationByComparison()을 호출하여 다음 목적지를 정한다.<br/> C3. move()를 호출하여 엘리베이터를 이동상태로 전환한다.<br/> C4. 자동으로 movemencontrol이 호출될 것이다.</p> <p>D. 문이 열려있을 때 자동대기 Standby()<br/> D1. Wait() n초간 대기한다.<br/> D2. 대기중 추가 문제어 요청이 올경우 그를 따른다.<br/> D3. 대기시간이 지나면 closeDoor를 수행한다.<br/> D4. departureCalibration을 호출한다.</p> |
| Alternative Flow | <p>B에서 엘리베이터 방향과 목적층수가 같아 겹치는 층선택 요청과 탑승요청이 공존할경우<br/> BB1. 이 상황일경우에 앞서 미리 큐알고리즘에 저장된 두번째 목적지 노드의 탑승자 정보를 읽어온다.<br/> BB2. addLoad()<br/> BB3. deQueue()<br/> BB4. Standby()</p>   |
| Exception Flow   |   |

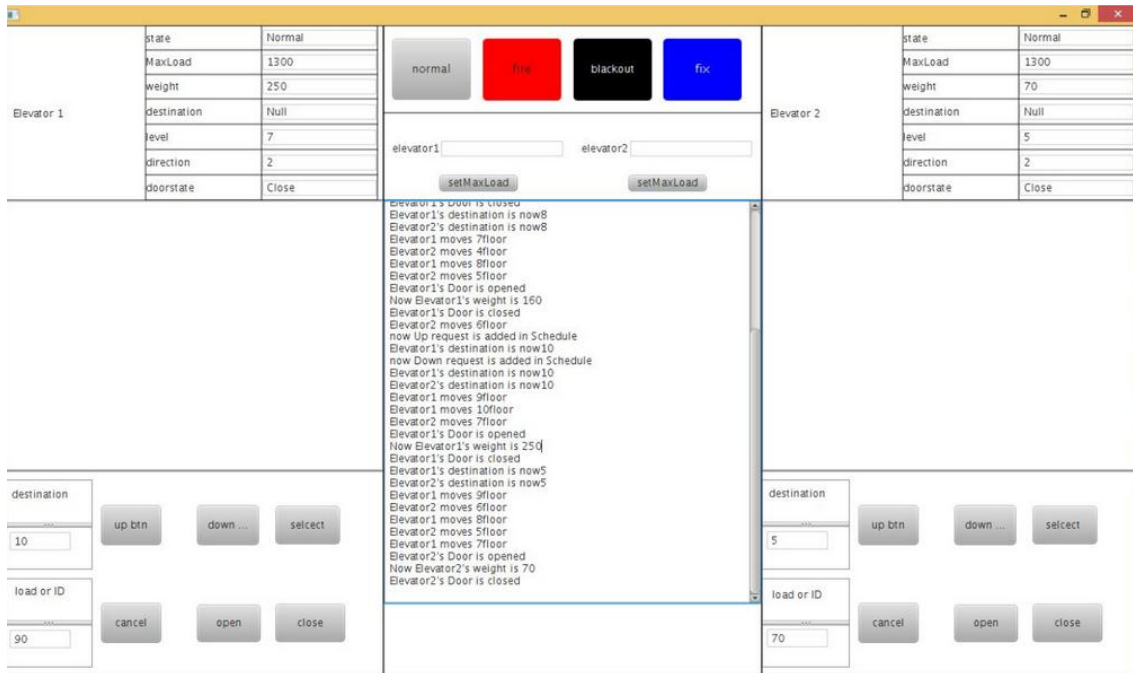
### 1.18 큐 읽기(Read Queue)

|                             |   |
|-----------------------------|---|
| Use case                    | 3.2.4 큐 읽기(Read Queue)  |
| Actor                       | System  |
| Purpose                     | 이전 유스케이스(Essential, Business)참조   |
| Overview                    | 이전 유스케이스(Essential, Business)참조   |
| Type                        | Secondary   |
| Cross Reference             | 3.2.4 3.2.1 2.1.3   |
| Pre-Requisite               |   |
| Typical course of event     | (A) : Actor, (S) : System<br>1. (S) : Flux 계산의 가장 가까운 AboardQueueNode 요청<br>2. (S) : 현재 엘리베이터의 int currentPostion과 int directionDelta을 참조하여 가장 가까운 AboardQueueNode를 찾아서 반환한다. |
| Alternative Course of event | 1. (S) : Flux 계산의 수행큐 요청<br>2. (S) : Queue의 Queuenode first를 반환한다.  |
| Exceptional Course of event |   |

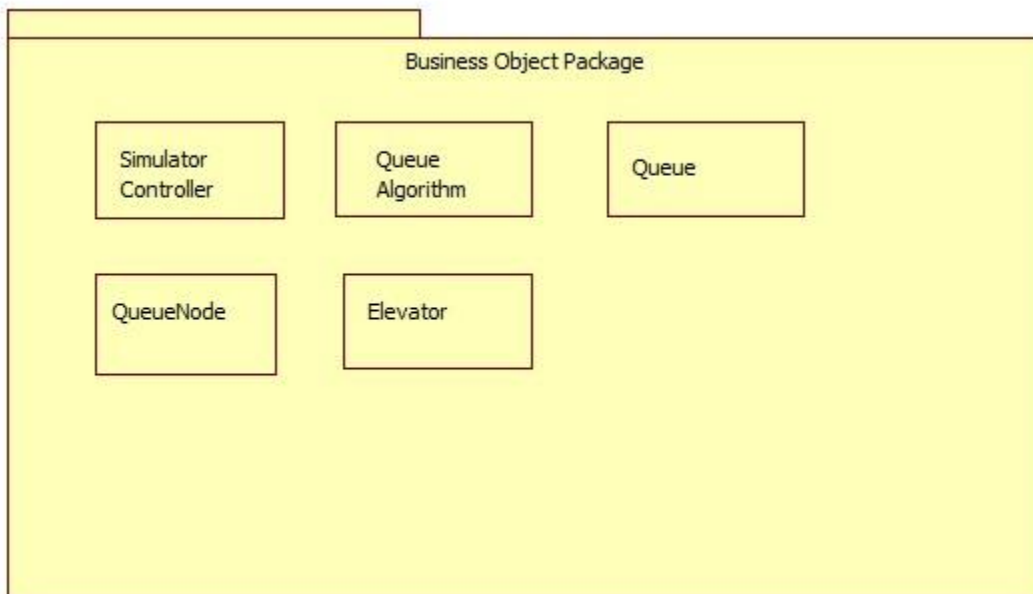
## 2. Activity 2042 Define Reports, UI, and Storyboards

### 2.1 Entire GUI



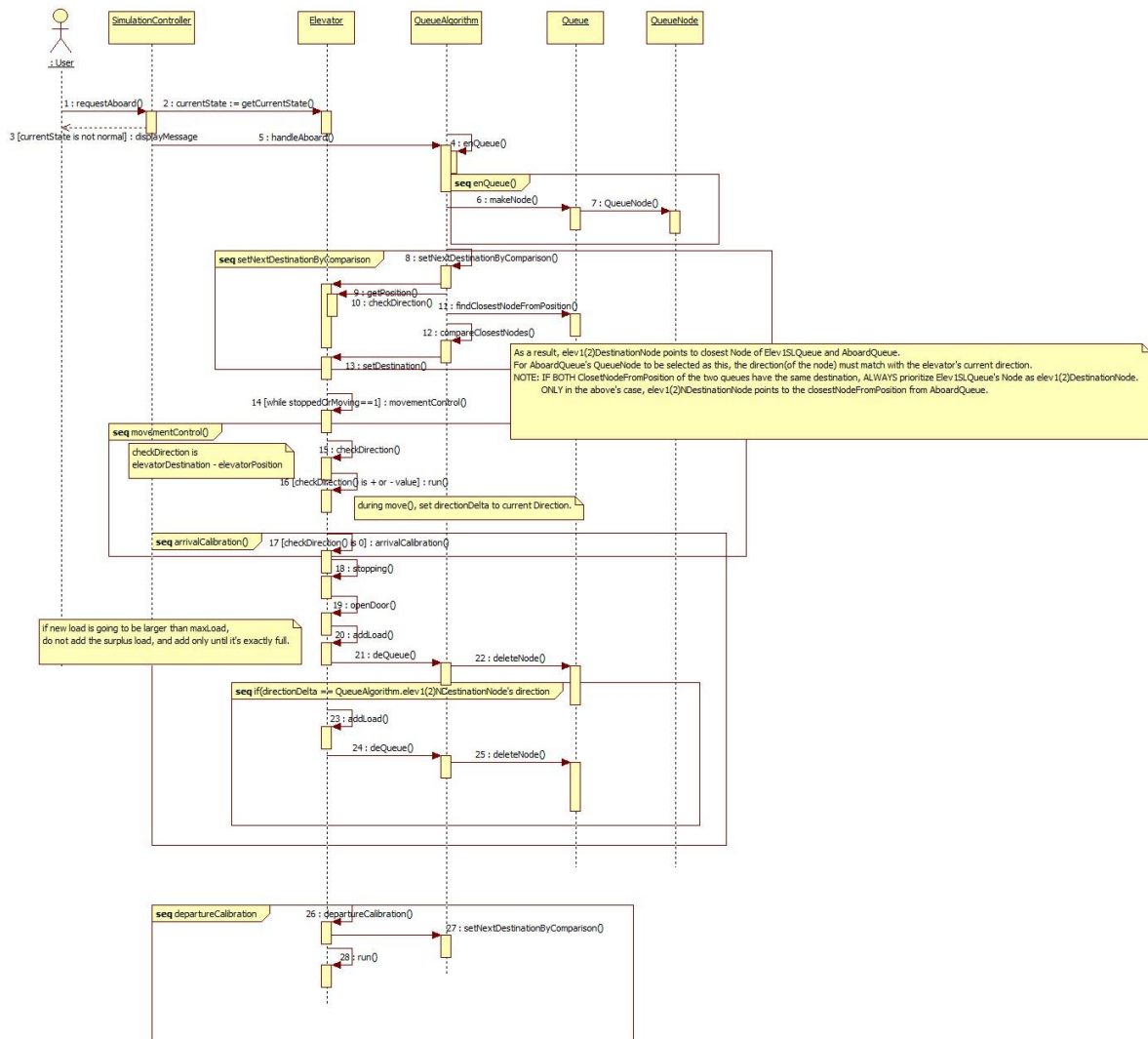


### 3 Activity 2043 Refine System Architecture

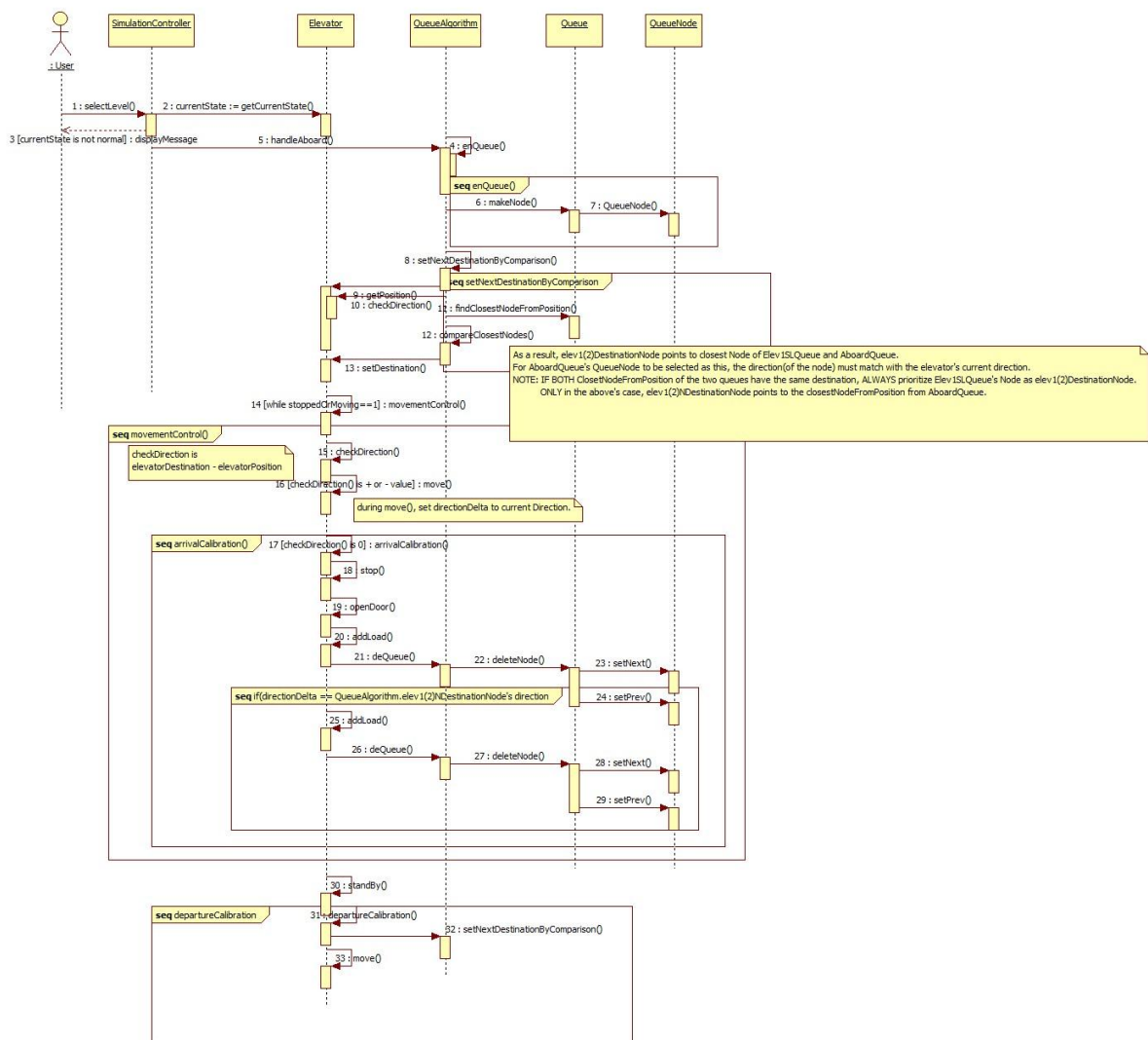


### 4 Activity 2044 Define Interaction Diagram

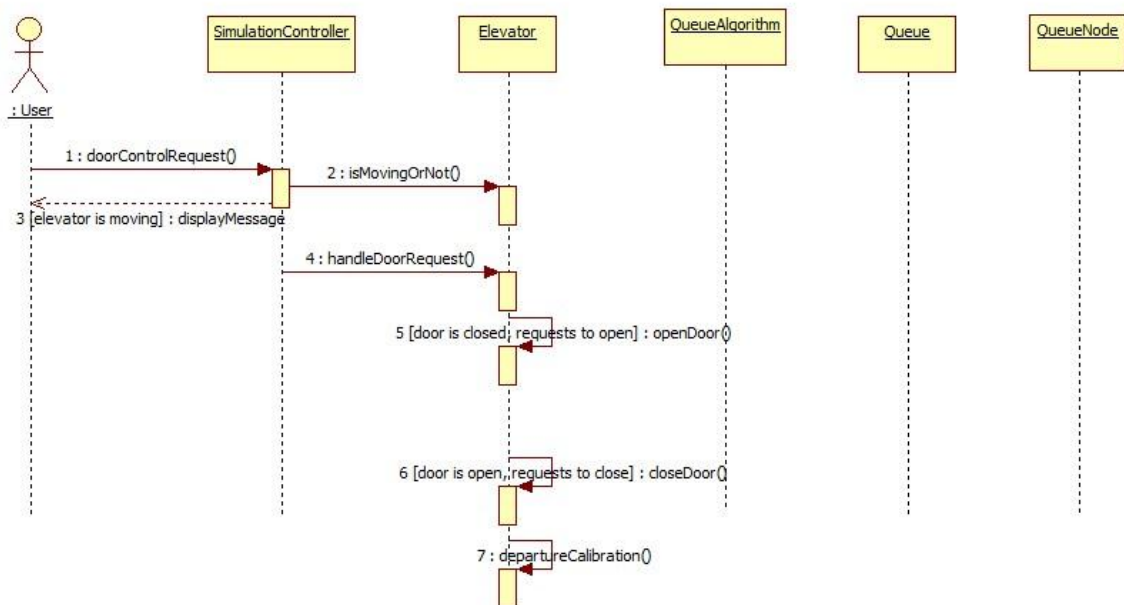
#### 4.1 탑승 요청(Request Aboard)



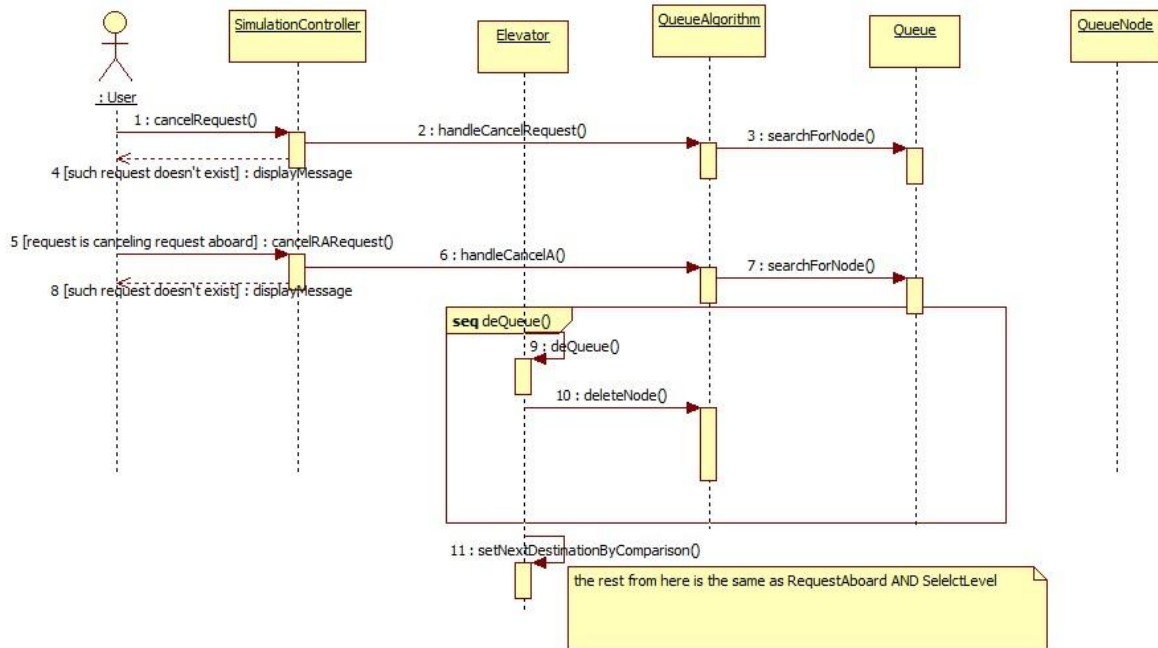
## 4.2 층 선택(Select Level)



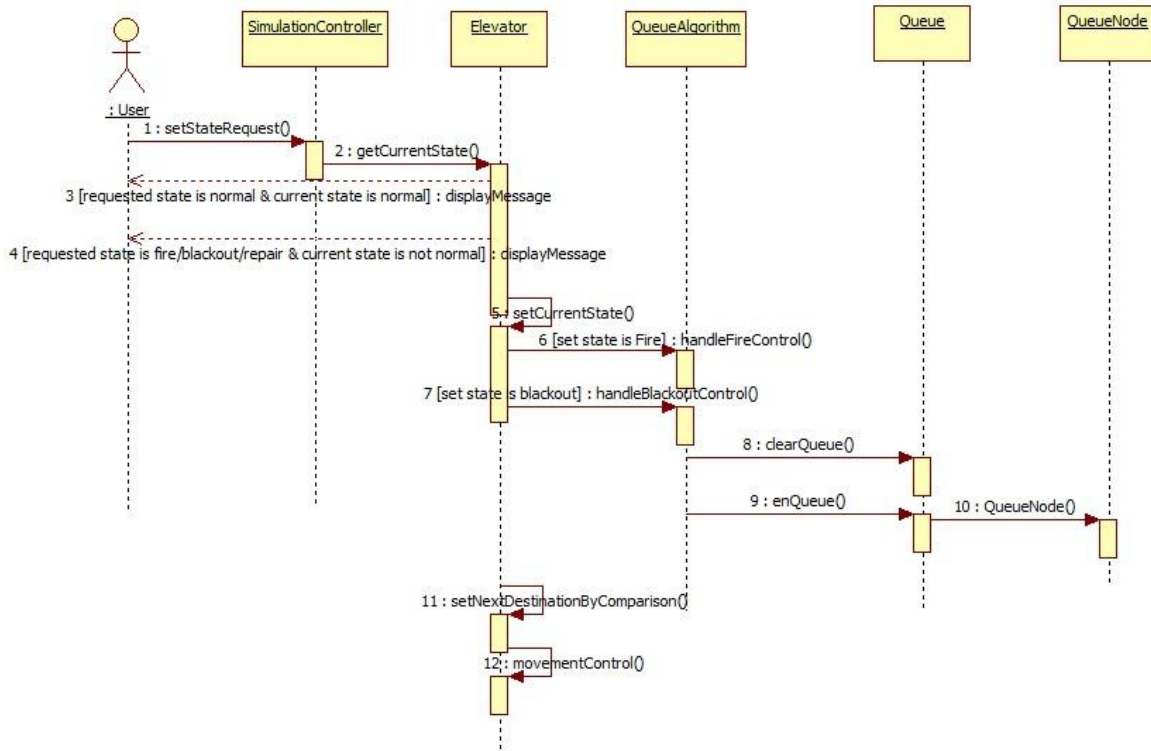
### 4.3 문 제어 요청(Request Door)



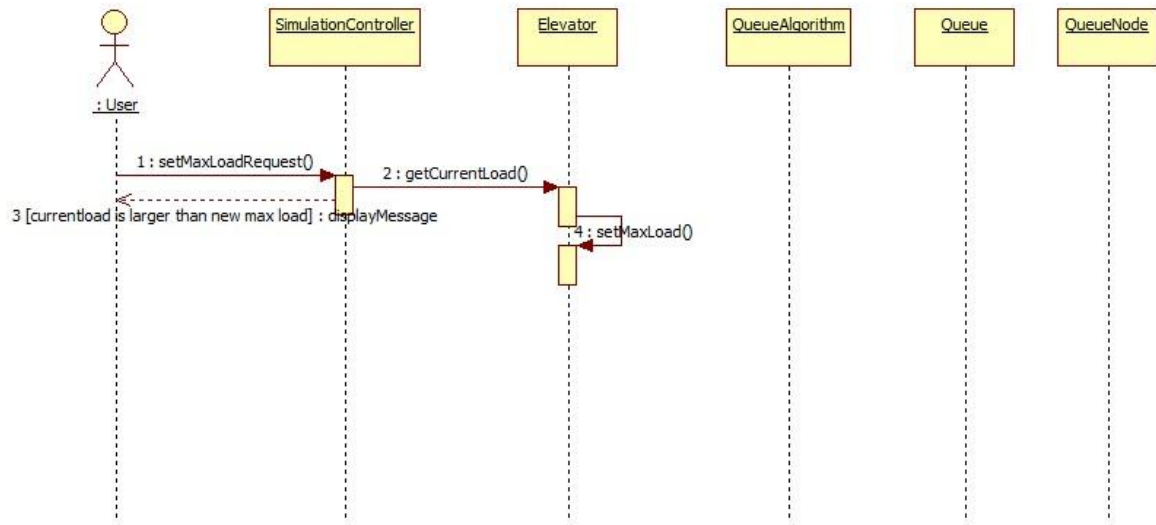
#### 4.4 요청 취소(Cancel Request)



#### 4.5 상태 시뮬레이트(Config State)



#### 4.6 최대 하중 조정(Config Maxload)



# 5. Activity 2045 Define Design Class Diagram

참조 <http://i.imgur.com/jns7wlk.png>

