**소프트웨어 공학**

**최종보고서**

분반 : 01

2015920015 김환석

2015920017 박민근

목차

1. **개요**
2. **Use Case Diagram**
3. **Sequence Diagram**
4. **Architecture Diagram**
5. **Class Diagram**
6. **Activity Diagram**
7. **객체 State Diagram**
8. **Component Package 구성**

**분석 산출물**

|  |  |
| --- | --- |
| 문서버전 | 1.0 |
| 문서 ID | H&M-2019-001 |
| 최종변경일 | 2019- 12- 05 |
| 문서상태 | 릴리즈 |

**요약**

Mobile Robot Controller 시스템의 분석 산출물을 기술.

주요 산출물

* 유즈케이스도와 설명
* 엑티비티도

표1 문서 변경 기록

|  |  |  |  |
| --- | --- | --- | --- |
| * **문서 이름** | * **<H&M>소프트웨어공학** | | |
| * 문서 ID | * H&M\_2019\_001 | | |
| * 버전 | | * 변경일 | * 설명 |
| * 1 | * 1 | * 2019 - 10 - 08 | * 1.1 요구사항 정리 |
| * 2 | * 2019 - 10 - 15 | * Use case Diagram 작성 |
| * 3 | * 2019 - 10 - 18 | * Use case Diagram 수정 |
| * 4 | * 2019 - 10 - 29 | * Use case Diagram 작성 * Sequence Diagram 작성 |
| * 6 | * 2019 - 11 - 3 | * Class Diagram 수정 * Sequence Diagram 수정 |
| * 7 | * 2019 - 11 - 4 | * 중간 보고서 작성 |
| * 9 | * 2019- 11 -12 | * 구현 시작 |
| * 11 | * 2019- 11- 18 | * Class Diagram 수정 * Sequence Diagram 수정 |
| * 13 | * 2019- 12- 02 | * Activity Diagram 작성 |
| * 15 | * 2019- 12- 04 | * 최종보고서 작성 |
|  |  |  |

개요

1. 목적

재난지역 정보 파악을 위한 mobile Robot인 SIM을 control 하고 지도를 갱신하는 ADD-ON System의 개발

1. 참고문헌

-없음

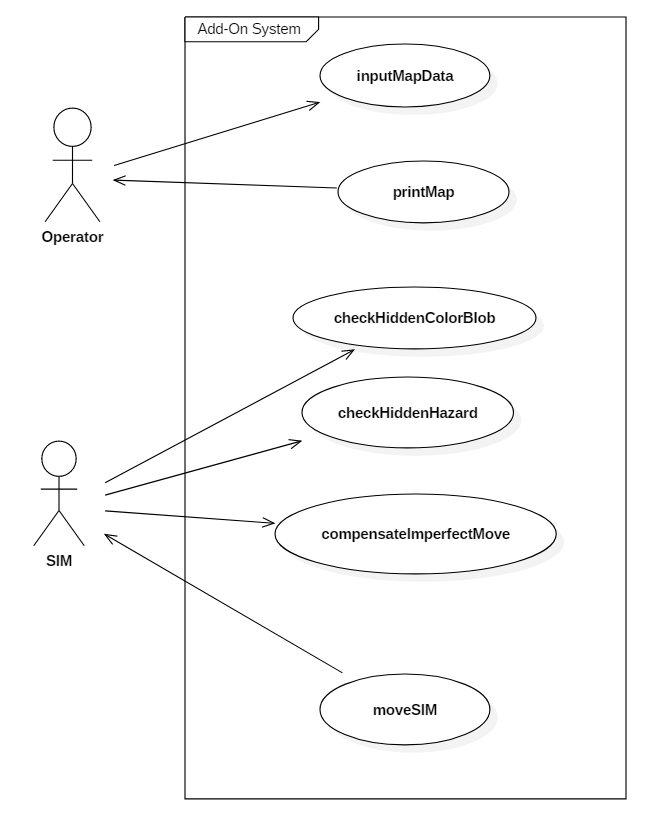
3. 요구사항

1. 로봇 SIM은 Operator 에게서 지도를 입력 받는다.
2. 지도를 이용하여서 예상 이동 경로를 구축한다. 여기에는 위험지역을 피해가는 방향으로 경로를 결정 짓는다.
3. Hidden hazard sensor를 이용해서 전방 한칸에 미발견 위험지역을 탐사한다. 미발견 위험지역 발견시 지도를 수정한다.
4. Hidden colorblob sensor를 이용해서 상하좌우 한칸에 있을 미발견 중요지역을 탐사한다. 미발견 중요지역 발견시 지도를 수정한다.
5. 만일 지도가 수정되었다면 경로를 재탐사 한다.
6. 수정된 경로를 가지고 다음 동작을 지시한다. SIM의 robot movement interface를 이용. robot movement interface는 robot이 전방 앞으로 1칸 이동과 90도 회전하는 기능을 제공
7. SIM의 현재 위치의 값을 받아서 그 값과 이동명령을 비교해서 잘못된 이동에 관하여 보상을 해준다.

Use Case Diagram

1. Actor

* Operator : 초기의 재난지역 모델에 데이터를 넣어주는 엑터
* SIM: ADD-ON 시스템에 따라 움직이는 Mobile Robot



1. Use Case 설명서와 시나리오

본 장에서는 각 유즈케이스에 대한 설명을 기술한다. 표로 구성되어 있는 부분이 설명서 이고 각 표 아래에 시나리오(sequence diagram) 를 기술한다.

**Use Case Document**

**inputMapData**

<개략명세>

이 use case는 operator가 초기화 하며, operator가 ADD-ON기능에 필요한 지도를 생성 초기화 하기 위한 기능을 제공한다.

<사건의 흐름>

이 use case는 operator가 지도의 크기, 로봇의 시작 위치, 초기 탐색지역 위치, 초기 위험지역 위치를 입력하면서 시작한다. 그리고 난 다음에 아무 특성도 포함하지 않은 지역에 한해서 랜덤으로 미발견 위험지역과 미발견 탐색지역이 일정 확률로 정해진다.

Operator가 해당 모든 입력값들을 입력하면 지도는 저장된다.

<Alternative flow>

Operator가 만약 잘못된 입력값(지도의 크기를 벗어나는 입력값 또는 초기 지도의 크기를 자연수로 설정하지 않은 것 등)을 입력시에 재입력 받기를 요구한다.

<지도의 크기 입력>

지도의 크기를 정해주기 위해 입력값을 받는다. 이에 따라서 지도의 크기가 결정된다.

>> 지도의 행, 열 이 되는 숫자(자연수) 값을 차례대로 입력 받는다.

>> 지도의 각 좌표에 좌표 특성을 저장하기 적절한 형태로 파싱되어서 저장된다.

<로봇의 시작위치>

로봇의 시작위치가 결정된다. 이는 지도의 안에 있는 좌표가 되어야 한다.

<초기 탐색지역 위치>

초기 탐색 위치를 입력 받는다. 이는 한 개 이상이 되어야 한다. 그리고 로봇의 시작위치와 겹치지 않아야 한다.

<초기 위험지역 위치>

초기 위험지역 위치를 입력 받는다. 이는 0개 이상이 되어야 한다. 그리고 이는 로봇의 시작위치와 초기 위험지역 위치와 겹치지 않아야한다.

<미발견 탐색지역, 미발견 위험지역 결정>

미발견 탐색지역과 미발견 위험지역이 일정 확률로 결정된다. 이는 각각 0개 이상이 되어야한다. 그리고 오직 각각 colorblob sensor와 hazard sensor로만 탐사 할 수 있다. 지도만으로는 판별 할 수 없다.

**printMap**

<계략명세>

이 use case는 operator가 요구 할 수 있는 것이다. Operator가 현재 지도 상태에 대한 출력을 요구한다.

<사건의 흐름>

이 use case는 operator가 현재 지도 상태를 출력을 요구하면 이에 따라서 지도 출력

<Alternative flow>

없음

<지도출력>

최근 지도에 관한 정보가 저장되있는 곳에 접근하여서 지도정보를 가시성 높게 파싱해서 출력해준다.

**checkHiddenColorBlob**

<계략명세>

이 use case는 SIM이 사용한다. SIM은 ADD-ON 시스템을 통해서 colorblob sensor를 사용하는데 현재 위치에서 상하좌우 1칸에 해당하는 좌표에 미발견 탐색지역이 있는지를 발견 가능하다.

<사건의 흐름>

이 use case는 SIM이 자신의 현재 위치를 확인 하면서 시작한다. 그 후에 아래와 같이 순차대로 수행된다

>> 로봇의 현재 위치 확인

>> 미발견 탐색지역 확인

>> 지도 수정

지도가 수정되거나 미발견 탐색지역이 발견되지 않으면, 이 사용 사례는 완료 된다.

<Alternative flow>

없음

<로봇의 현재 위치 확인>

Positioning sensor을 이용하여서 로봇의 현재위치를 알아낸다.

<미발견 탐색지역 확인>

Colorblob sensor를 사용해서 지도에서 상하좌우에 1칸에 해당하는 지역의 지도 정보를 확인해서 미발견 탐색지역을 확인한다.

<지도 수정>

확인한 미발견 탐색지역을 이용하여서 지도를 수정하게 된다.

**checkHiddenHazard**

<계략명세>

이 use case는 SIM이 초기화 한다. 미발견 위험지역을 발견하고 지도를 수정하는 기능을 제공한다.

<사건의 흐름>

이 use case는 SIM이 자신의 현재위치를 확인 하면서 시작한다. 현재 위치를 확인하게 되면 아래의 기능들을 순차 대로 수행한다

>> 미확인 위험지역 겸사

>> 지도 수정

<Alternative flow>

없음

<로봇의 현재 위치 확인>

Positioning sensor을 이용하여서 로봇의 현재위치를 알아낸다.

<미발견 위험지역 확인>

Hazard sensor를 사용해서 로봇의 방향의 전방 1칸에 해당하는 지역의 지도 정보를 확인해서 미발견 위험지역을 확인한다.

<지도 수정>

만일 미발견 위험지역의 존재가 확인 되면 지도를 수정하여 준다.

**compensateImperfectMove**

<계략 명세>

이 use case는 올바르지 않은 로봇의 이동을 보상하거나 경로를 새롭게 설정하기 위한 기능을 제공한다.

<사건의 흐름>

이 use case는 SIM이 초기화한다. 먼저 SIM의 현재 위치를 알아내는 것부터 시작한다. 그리고 나서 아래 기능이 수행된다.

>> 위치와 이동명령 비교

<위치와 이동명령 비교>

만일 이동명령과 위치정보를 비교하여서 이동명령에 맞게 수행되지 않았다면 이를 보상하거나 경로를 다시 설정하여줘서 다음이동에 문제 없게 만든다.

**moveSIM**

<계략 명세>

이 use case는 SIM이 초기화 한다. 로봇의 이동명령을 수행하기 위한 기능을 제공한다.

<사건의 흐름>

먼저 SIM이 이동 명령을 요청한다. 그럼 경로를 설정한 다음에 robot movement interface에 의

해서 아래와 같이 수행 가능하다.

>> 90도 회전

>> 앞으로 한칸 이동

<Alternative flow>

이동 명령을 입력하면 현재위치에서부터 중요 지점까지의 경로를 만들어 주는데 만약 경로가 존재 하지 않으면 이동 명령 수행이 불가능 하므로 위의 use case가 종료 된다.

<90도 회전>

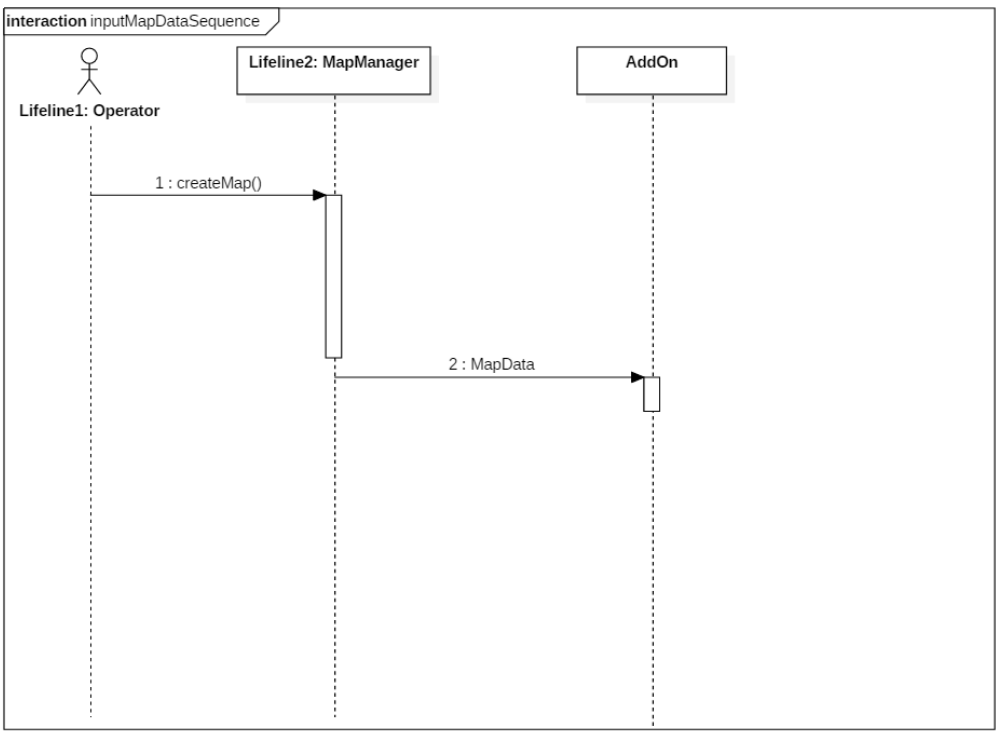
처음에 경로를 만들어 주는데 로봇의 direction이 경로를 향하게 로봇을 회전 시켜주는데 90도 씩 회전 시켜서 해당 방향으로 향하게 한다.

<앞으로 한칸 이동>

경로를 향하여서 90도 회전을 마친 상태 이거나 90도 회전을 할 필요가 없는 경우에 경로를 따라서 앞으로 한칸 이동한다(follow given path). 이때 주의 할점은 이동되는 지점에 미발견 위험지역이 절대 있어서는 안된다.

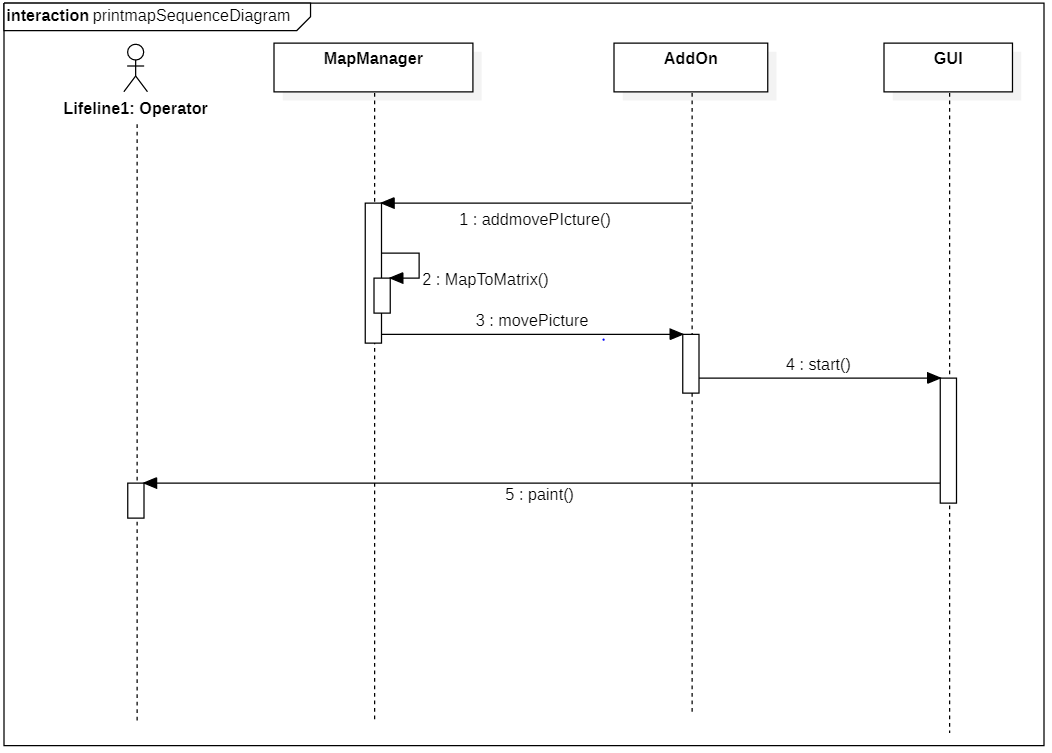
* 1. inputMapData

|  |  |  |
| --- | --- | --- |
| 유즈케이스 | 지도 데이터 입력(inputMapData) | |
| 유즈케이스 ID | OP-UC01 | |
| 엑터들 | Operator | |
| 요약 | 초기에 지도 데이터를 입력하기 위한 use case이다. Operator는 초기 지도 구성에 필요한 값을 입력한다. | |
| 유형 |  | |
| 참고 유즈케이스 | 없음 | |
| 사전 조건 | 입력 받는 지도 정보에는 지도 크기, 시작 위치, 탐색 위치 가 포함되어야한다. 위험지역은 랜덤으로 설정 | |
| 사후 조건 | ADD-ON에 지도가 행렬로 파싱되어 저장된다. | |
| 주 흐름 | operator | AddOn |
| 1)지도의 초기 생성을 요청하면 MapManager가 createMap()메소드를 통해서 초기 입력값들(지도의 크기, 중요지점, 탐색지점, 위험지점, 시작위치)을 operator에게서부터 입력받는다 | 2)해당 입력값들을 기준으로 하여 초기 지도를 생성한다. |
| 부 흐름 | 없음 | |
| 제약 사항 | 없음 | |



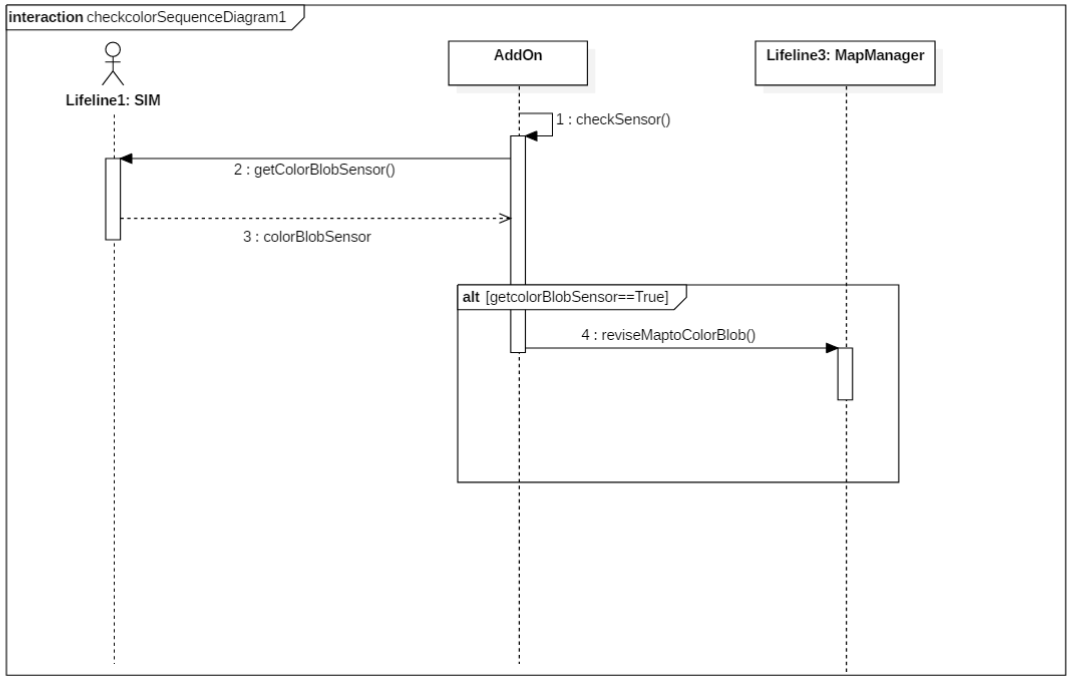
* 1. printMap

|  |  |  |
| --- | --- | --- |
| 유즈케이스 | 지도 출력(printMap) | |
| 유즈케이스 ID | OP-UC02 | |
| 엑터들 | operator | |
| 요약 | 지도의 내용을 불러와서 출력한다. | |
| 유형 |  | |
| 참고 유즈케이스 | 없음 | |
| 사전 조건 | 현재 지도를 표현한 행렬이 ADD-ON에 저장되어 있어야 한다. | |
| 사후 조건 | 지도를 출력하기 전과 후의 지도의 내용에 변함이 없다. | |
| 주 흐름 | MapManager | ADD-ON |
| 2) 지도의 내용을 보고 이를 배열로 변환한다.  3) 현재까지의 저장된 배열형태로 저장된 지도 상태 정보들을 AddOn에 전달 | 1)현재 지도 이미지를 저장하도록 MapManger에 명령한다.  4)배열의 정보들을 이미지로 추출하는 명령을 내린다. |
| GUI |
| 5)GUI객체는 사용자에게 지도를 시각적으로 보여준다. |
| 부 흐름 | 없음 | |
| 제약 사항 | 없음 | |



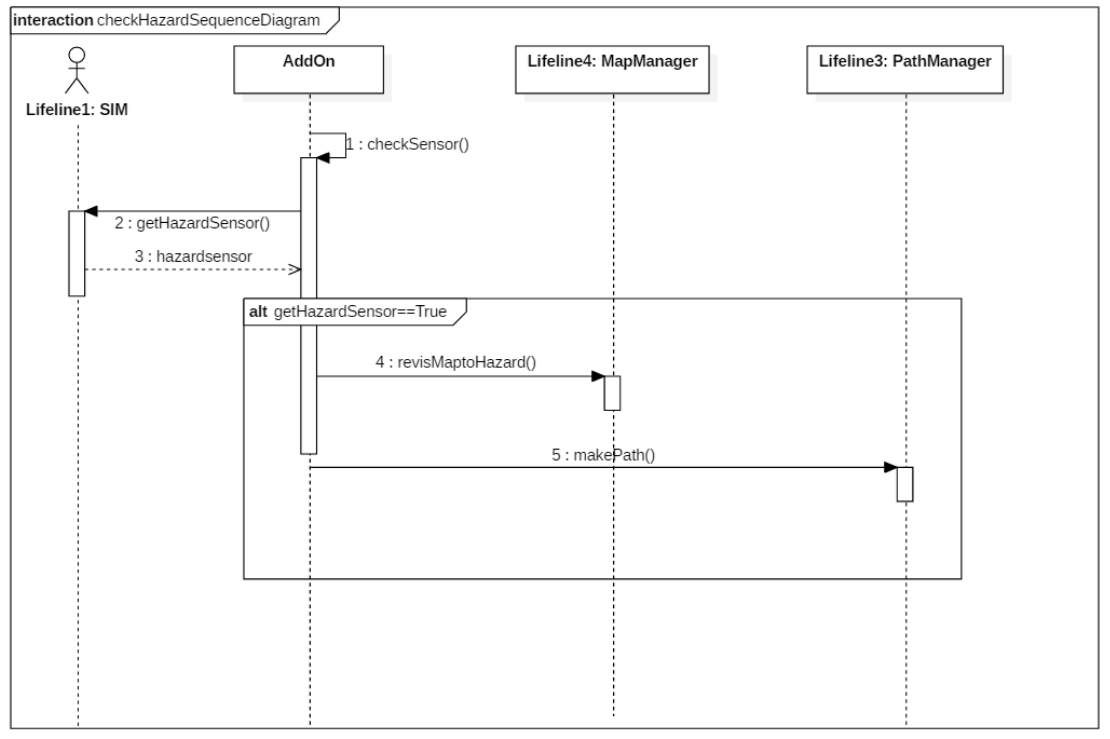
* 1. checkHiddenColorBlob

|  |  |  |  |
| --- | --- | --- | --- |
| 유즈케이스 | 미발견 탐색지역 탐사(checkHiddenColorBlob) | | |
| 유즈케이스 ID | SIM-UC01 | | |
| 엑터들 | SIM | | |
| 요약 | 미발견 탐색지역을 탐사하고 발견시 지도를 수정한다. | | |
| 유형 |  | | |
| 참고 유즈케이스 | 없음 | | |
| 사전 조건 | 1. 로봇의 현재위치를 알아낸다 2. 지도의 정보가 저장된 객체를 받는다. | | |
| 사후 조건 | 미발견 탐색지역 발견시 지도를 수정해줘야 한다. | | |
| 주 흐름 | SIM | | AddOn |
| 2)Add-On 의 명령을 받아서colorblob sensor 를 검사하는 sensor check를 수행한다. | 1)SIM에게 sensor check할 것을 명령한다.  4)받은 센서결과값과 지도를 비교해서 미발견 탐사지역임이 밝혀지면 지도를 수정한다. | |
| 부 흐름 | 없음 | | |
| 제약 사항 | 없음 | | |



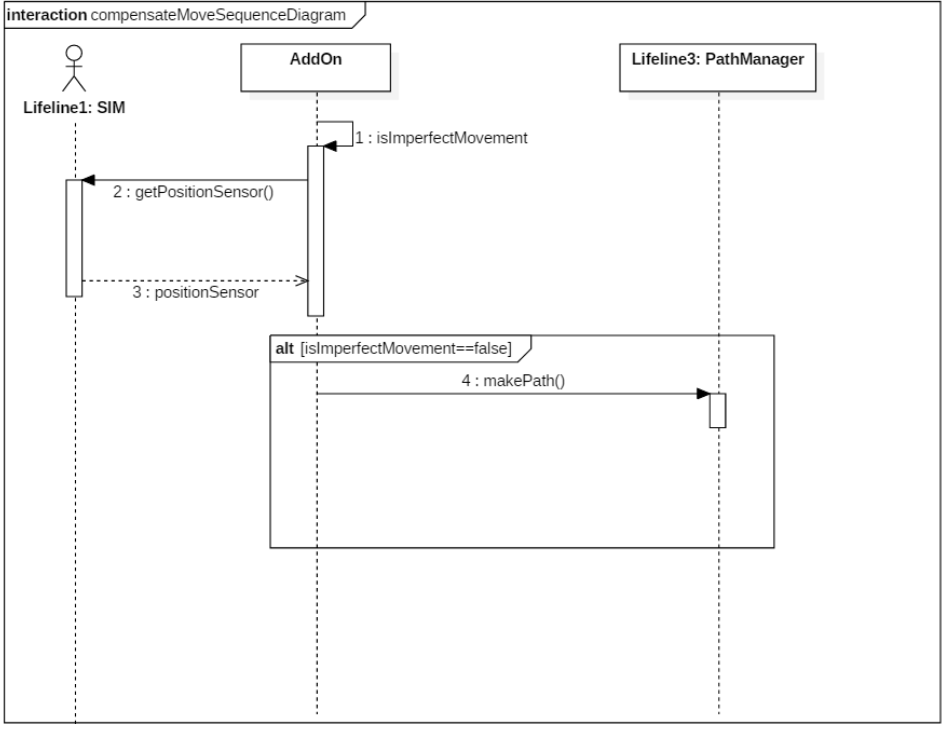
* 1. checkHiddenHazard

|  |  |  |
| --- | --- | --- |
| 유즈케이스 | 미발견 위험지역 탐사(checkHiddenHazard) | |
| 유즈케이스 ID | SIM-UC02 | |
| 엑터들 | SIM | |
| 요약 | Robot의 전방 1칸에 위치한 좌표이 미발견 위험지역인지를 판별한다. | |
| 유형 |  | |
| 참고 유즈케이스 |  | |
| 사전 조건 | 1. positioning sensor로부터 로봇의 현재 위치 , 방향을 알아낸다 2. 최근 지도 정보를 불러온다 | |
| 사후 조건 | 미발견 위험지역이 발견되면 지도를 수정한다 | |
| 주 흐름 | SIM | ADD-ON |
| 3)hazard sensor를 통해 탐사한 값을 AddOn에 보낸다 | 1) checksensor 를 시작한다  2)SIM의 hazard sensor값을 알아온다  4)SIM에서 받아온 hazard sensor값과 지도정보를 비교해서 지도를 수정한다  5) pathManager에게 경로를 재설정명령을 보낸다 |
| 부 흐름 | 없음 | |
| 제약 사항 | 없음 | |



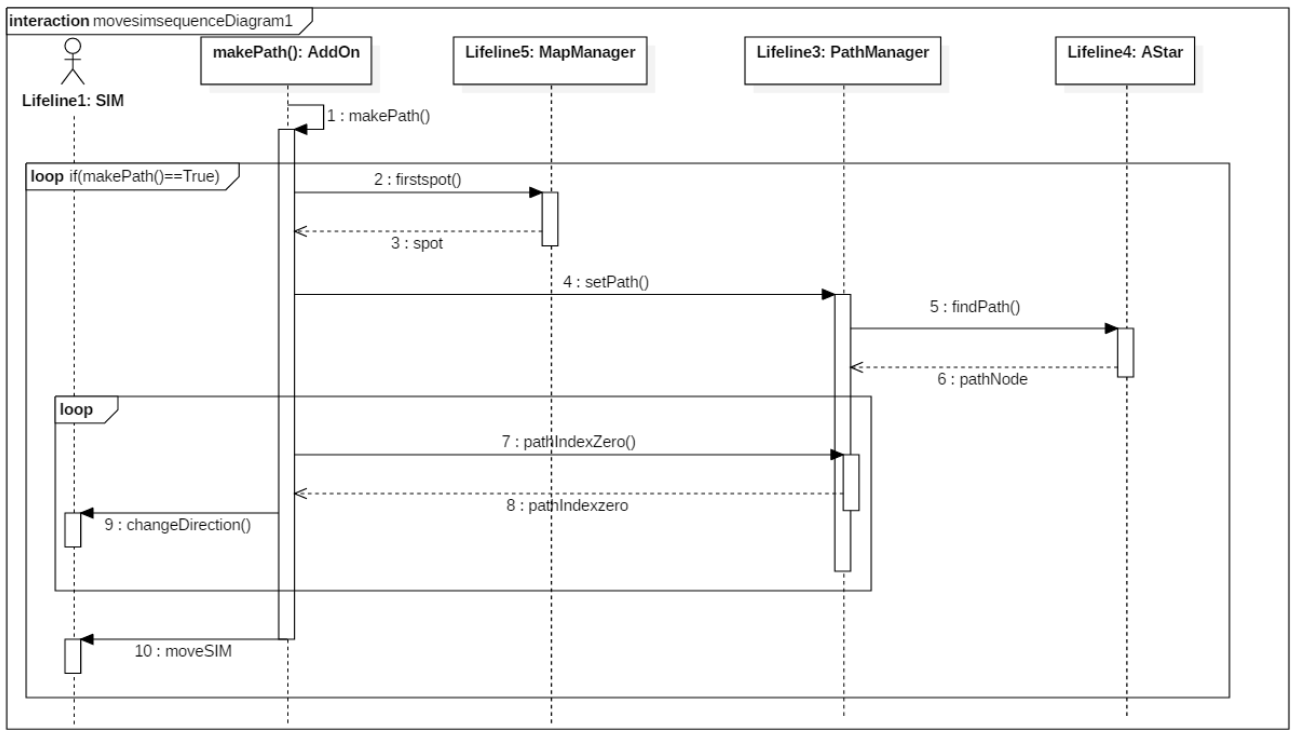
* 1. compensateImperfectMove

|  |  |  |
| --- | --- | --- |
| 유즈케이스 | 불완전 이동 보상(compensateImperfectMove) | |
| 유즈케이스 ID | SIM-UC03 | |
| 엑터들 | SIM | |
| 요약 | positioning sensor값과 robot movement interface를 이용한 이동명령을 이용해서 잘못된 이동을 알아내고 이를 보상한다. | |
| 유형 |  | |
| 참고 유즈케이스 | 없음 | |
| 사전 조건 | 1. 이동한 곳이 위험지역이 아니여야 한다 2. 이동 직후에 해당 usecase를 수행한다 | |
| 사후 조건 | 만약 잘못된 이동임이 판별될 시에 경로를 재설정 하던지 다음 이동에 영향이 없도록 하여야 한다. | |
| 주 흐름 | SIM | ADD-ON |
| 3) position sensor를 이용해서 현재위치를 전달한다 | 1) 잘못된 이동이 일어났는지 검사한다.  2)position sensor를 이용해서 robot의 현재위치를 알아낸다  4)현재위치 값을 분석해서 잘못된 이동시 새로 경로를 설정하여준다. |
| 부 흐름 | 없음 | |
| 제약 사항 | 없음 | |

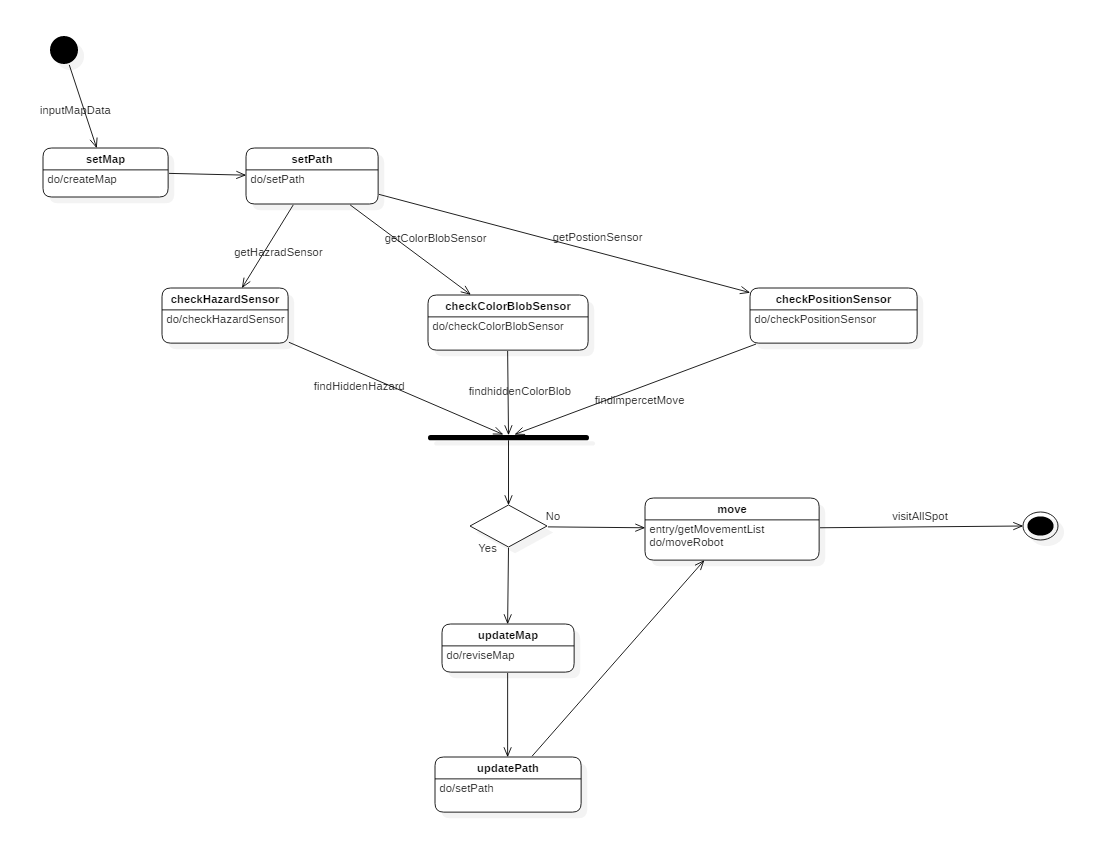


* 1. moveSIM

|  |  |  |
| --- | --- | --- |
| 유즈케이스 | SIM에 대한 이동(moveSIM) | |
| 유즈케이스 ID | SIM-UC04 | |
| 엑터들 | SIM | |
| 요약 | 경로를 따라서 ADD-ON은 이동명령을 수행한다. | |
| 유형 |  | |
| 참고 유즈케이스 | 없음 | |
| 사전 조건 | 1. 현재 좌표에서 hazard sensor와 colorblob sensor를 사용하여서 알아낼수 있는 미발견 위험지역, 미발견 탐색지역은 이미 다 찾은 상태여야 한다. 2. 이에 따른 경로가 계산된 후여야 한다. | |
| 사후 조건 | 이동한 후에 불완전한 이동에 대한 보상이 실행 되어야한다. | |
| 주 흐름 | SIM | AddOn |
| 1)탐색지역을 향한 이동을 요청한다 | 1)경로를 만드는 명령을 내린다.  2)여러 탐색지역중 먼저 방문할 탐색지역을 선별한다.  4)경로를 만드는 명령을 PathManager에게 전달한다.  7)경로중에서 첫번째로 가야될 좌표를 요구한다.  9)첫번째로 가야될 좌표를 보고 로봇의 방향을 회전하는 명령을 SIM에게 내린다  10)Robot을 1칸 전진시키는 명령을 SIM에게 내린다. |
| MapManager | PathManager |
| 3)먼저 방문할 탐색지역을 AddOn에게 반환한다. | 5)경로를 만드는 명령을 Astar에게 전달한다.  8)경로중에서 첫번째로 가야될 좌표를 AddOn에게 반환한다. |
| Astar |
| 6)Astar방식으로 경로를 만들어 반환한다. |
| 부 흐름 | 없음 | |
| 제약 사항 | 없음 | |



Activity Diagram



setMap

▶ operator가 inputMapData를 통해서 지도를 생성하게 되면 setMap의 상태가 된다.

▶ 잘못된 지도의 입력값이 주어지면 요청을 거부하고 트랜잭션을 만들지 않고 재입력을 요구한다.

setPath

▶ADD-ON시스템이 경로를 찾기를 요청하면 setPath상태가 된다. 현재 지점부터 중요지점까지 탐색지역을 거처서 갈 수 있는 지역을 탐사한다.

checkHazardSensor

▶먼저 선 조건은 경로를 이미 찾은 상태이다. 이 상태에서 getHazardSensor의 요청을 보내면 전방 1칸의 위험지역을 탐사하고 난 후에 checkHazardSensor 상태로 도입하게 된다. 이 상태에서는 지도와 앞에서 탐사한 내용을 서로 비교하는 기능을 수행한다.

checkColorBlobSensor

▶먼저 선 조건은 경로를 이미 찾은 상태이다. 이 상태에서 getColorBlobSensor의 요청을 보내면 상하좌우 1칸의 탐색지역을 탐사하고 난후에 checkColorBlobSensor상태로 도입하게 된다. 이 상태에서는 앞에서 탐사한 내용과 지도를 서로 비교하는 기능을 수행한다.

checkPositionSensor

▶먼저 선 조건은 이동명령을 수행한 직후이다. 그리고 나서 바로 getPositionSensor를 사용하면 CheckPositionSensor상태로 도입하게 된다. 이 상태에서는 이동명령과 현재 위치를 비교하는 기능을 수행한다.

updateMap

▶checkHazardSensor, checkColorBlobSensor, checkPositionSensor상태에서 각각 비교한 값들이 false를 반환하게 될경우에는 각각 미발견 위험지역, 미발견 탐색지역, 잘못된 이동 에 대해서 YES를 뜻하게 된다. 이럴 경우에는 지도의 로봇의 현재위치 , 위험지역의 좌표, 탐색지역의 좌표, 현재 위치의 좌표 등을 수정하는 기능을 수행하게 된다.

Move

▶ checkHazardSensor, checkColorBlobSensor, checkPositionSensor상태에서 각각 비교한 값들이 True를 반환하게 될경우에는 각각 미발견 위험지역, 미발견 탐색지역, 잘못된 이동 에 대해서 NO를 뜻하게 된다. 이럴 경우에는 다음번 이동 명령에 대한 기능을 수행 할 수 있다.

updatePath

▶updateMap을 수행한 상태라면 다음으로 경로를 새로이 수정해주는 상태로 가기 위해서 updatePath상태로 도입하게 된다.

**설계 산출물**

|  |  |
| --- | --- |
| 문서버전 | 1.0 |
| 문서 ID | H&M-2019-002 |
| 최종변경일 | 2019- 12- 05 |
| 문서상태 | 릴리즈 |

**요약**

Mobile Robot Controller 시스템의 설계 산출물을 기술.

서브 시스템의 구성과 각 서브 시스템의 구조를 기술

주요 산출물

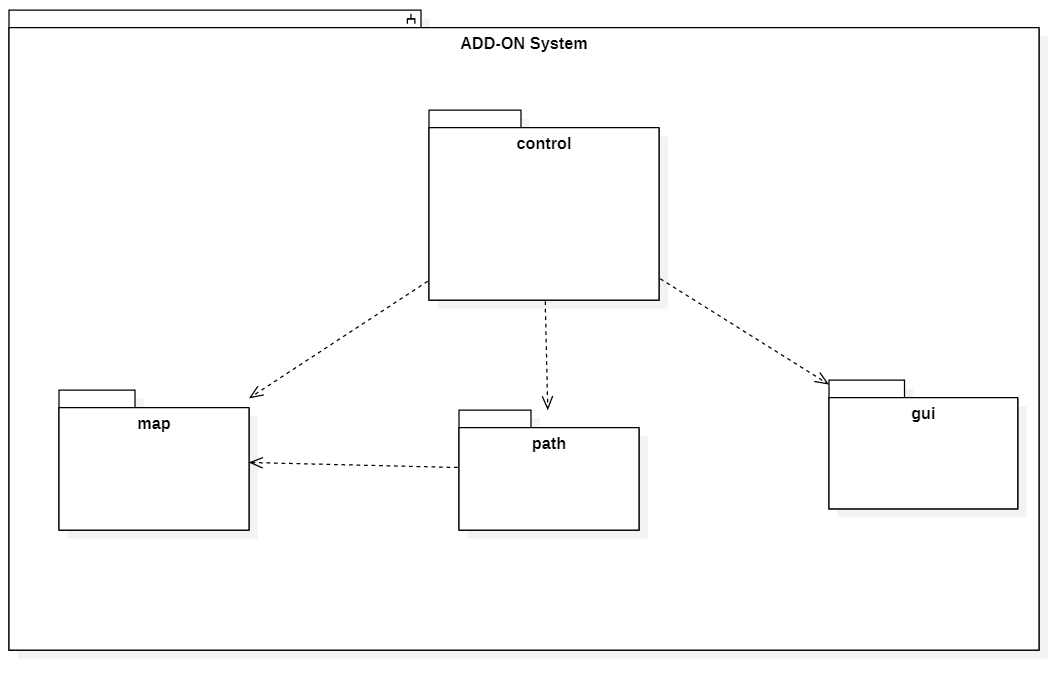
* 아키텍처도
* 클래스도
* 객체 상태도
* 컴포넌트 패키지 구성

표1 문서 변경 기록

|  |  |  |  |
| --- | --- | --- | --- |
| * **문서 이름** | * **<H&M>소프트웨어공학** | | |
| * 문서 ID | * H&M\_2019\_002 | | |
| * 버전 | | * 변경일 | * 설명 |
| * 1 | * 5 | * 2019 - 11 - 2 | * State Diagram 작성 |
| * 6 | * 2019 - 11 - 3 | * Class Diagram 수정 * Sequence Diagram 수정 * Architecture Diagram 수정 |
| * 7 | * 2019 - 11 - 4 | * 중간 보고서 작성 |
| * 8 | * 2019- 11 -9 | * 객체 State Diagram 작성 |
| * 9 | * 2019- 11 -12 | * 구현 시작 |
| * 10 | * 2019 – 11 -15 | * Component Diagram 작성 |
| * 12 | * 2019- 11- 27 | * Architecture Diagram 수정 |
| * 14 | * 2019- 12- 03 | * State Diagram 수정 |
| * 15 | * 2019- 12- 04 | * 최종보고서 작성 |
|  |  |  |

Architecture Diagram

Package Diagram



전체 시스템(ADD-ON)은 아래의 서브 시스템으로 구성된다.

* map 시스템
* path 시스템
* control 시스템
* gui 시스템

map 시스템

▶ADD-ON시스템에 필요한 지도 정보를담고 있으며 Path시스템에 지도내용을 전달하는 역할을 수행한다.

▶JAVA언어를 사용하여 작성한다.

path시스템

▶ADD-ON시스템에 필요한 경로에 관한 내용을 담고 있으며 Move시스템에 경로에 관한 내용이 필요한 명령을 수행할 때 이를 Move시스템으로 전달하는 역할을 수행한다.

▶JAVA언어를 사용하여 작성한다.

control

▶이 시스템은 SIM의 움직임을 Path 시스템으로부터 온 값을 이용하여서 수행하는 역할을 한다.

▶로봇의 움직임을 옳바르게 제공하는 기능을 제공한다.

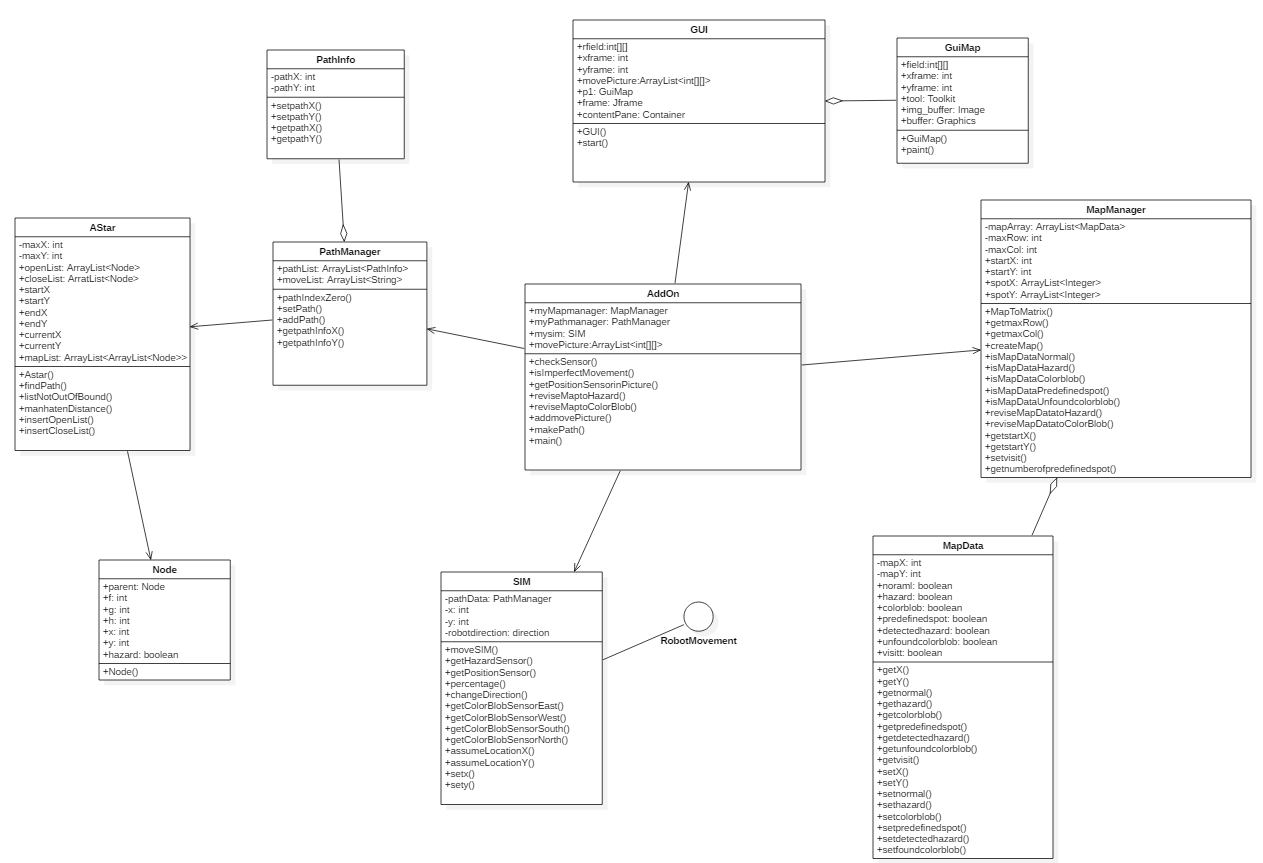
▶ JAVA언어를 사용하여 작성한다.

gui

▶이 시스템은 지도의 모습과 로봇의 움직임을 시각적으로 나타내기 위한 역할을 수행한다.

▶ JAVA 언어를 사용하여 작성한다.

Class Diagram



**<Class>**

-AddOn

▶ ADD-ON 시스템은 AddOn 클래스를 통해서 경로 설정, 미발견 위험지역, 미발견 탐색지역 조사, 움직임 제어, 지도 수정 등의 기능을 제공한다.

▶지도를 담당하는 MapManager, 경로를 담당하는 PathManager, 센서와 지도의 비교를 담당하는 SensorCheck의 객체를 만들어서 저장한다.

▶로봇의 잘못된 이동을 하였는지를 분석하여서 만약 잘못된 이동일시 이를 보상해주는 기능을 제공한다.

-SIM

▶ 이 클래스는 SIM의 현재 위치와 방향 그리고 경로등을 갖는다.

▶ AddOn에서 계산된 움직임을 실제로 수행가능한 메소드를 가지고 있다. 이는 RobotMovement interface로부터 받는다.

▶ 각 센서를 실행할수 있는 메소드를 포함하고 있다.

-PathManager:

▶이 클래스는 경로를 담당한다.

▶포함 관계로 pathInfo 클래스를 갖는다. 이를 ArrayList로 객체를 갖는다.

▶Astar class를 이용하여서 Astar알고리즘으로 경로를 계산한 값이 여기에 저장된다.

-PathInfo

▶좌표의 row, col을 저장할 수 있는 클래스 이다.

-AStar

▶ 경로를 Astar 알고리즘으로 계산하기 위한 클래스

-Node

▶Astar알고리즘에 사용되는 Node 클래스 이다.

-MapManager

▶이 클래스는 지도를 담당한다

▶포함관계로 MapData클래스를 갖는다. 이를 ArrayList로 객체를 갖는다.

▶create()메소드로 초기의 지도를 설정해줄수 있도록 한다.

▶맵의 크기를 attribute로 maxRowm, maxCol을 갖는다.

▶지도를 업데이트 할 수 있는 updateMap()메소드를 갖는다.

-MapData

▶ 지도의 좌표값과 그 좌표의 상태가 저장 될 수 있다. 여기서 상태는 위험지역, 탐색지역, 중요지역, 미발견 위험지역, 미발견 탐색지역, 일반지역 등이 될 수 있다.

-SensorManager

▶각 sensor로부터 나온 값과 지도상의 MapData에서 상태를 비교해 준다. 그래서 이것이 미발견 위험지역 또는 미발견 탐색지역인지를 확인할 수 있는 기능을 제공한다.

-GUI

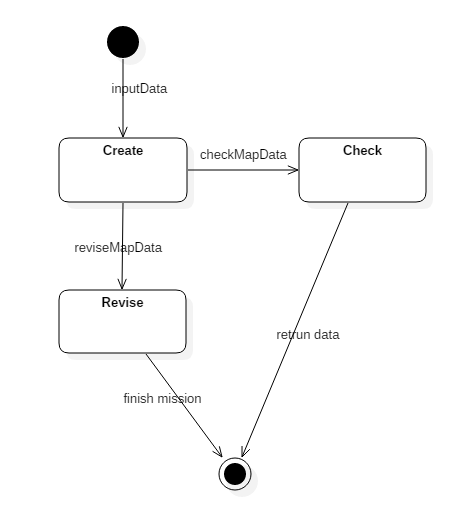
▶Add On으로부터 가져온 이미지를 프레임 안에 그려 넣는다.

-GuiMap

▶movePicture ArrayList의 값 하나씩 가져와 이차원 배열의 요소를 이미지로 만든다.

객체 State Diagram

MapManager객체 상태도



MapManager 는 트랜잭션의 상태를 나타내는 클래스이다. 트랜잭션은 다음 세 개의 강태 중 하나를 가진다.

-Create

■먼저 AddOn시스템이 MapManager의 객체를 생성하면 MapManager는 새로운 트랙젝션을 만들고 inputData명령을 통해서 Create상태로 초기화한다.

■ 잘못된 입력이 들어오면 요청을 받아들이지 않고 트렌젝션을 만들지 않는다.

-Check

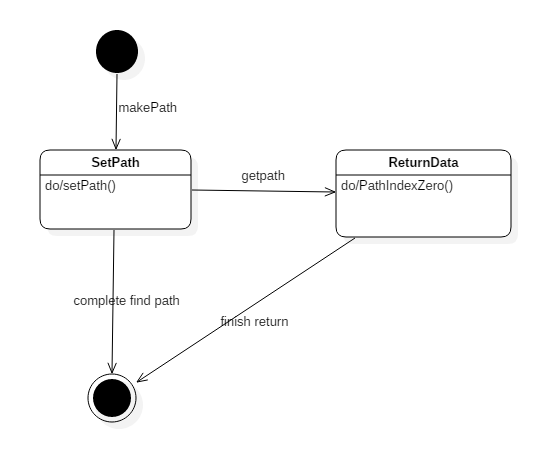
■AddOn시스템이 sensor 를 check하는명령을 요청하면 checkMapData을 통해서 트랜젝션의 상태를 Check로 만든다.

-Revise

■ AddOn시스템이 MapManager의 지도를 수정 하는명령을 요청하면 reviseMapData를 통해서 트랜젝션의 상태를 Revise로 만든다.

-지도 수정을 완료 했거나 AddOn 시스템이 sensor check값을 토대로 맵을 수정할 필요가 없다는 것을 판단하면 terminated상태로 바뀔 수 있다.

PathManager 객체 상태도



PathManager 는 트랜잭션의 상태를 나타내는 클래스이다. 트랜잭션은 다음 두 개의 강태 중 하나를 가진다.

-SetPath

■ 먼저 AddOn시스템이 PathManager의 객체를 생성하면 PathManager는 새로운 트랙젝션을 만들고 makePath명령을 통해서 SetPath상태로 초기화한다.

■만일 경로가 없을시에는 요청을 받아들이지 않고 트렌젝션을 만들지 않는다.

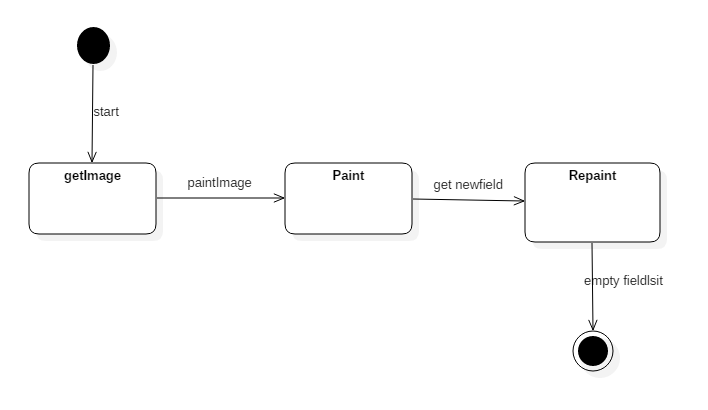
-ReturnData

■ AddOn이 경로중에 첫번째 요소에 해당하는 좌표를 요구하면 getPath명령을 통해서 ReturnData상태가 된다

■ PathIndexZero를 수행하여 위의 동작을 행한다.

-만약 경로 찾기를 완료하거나 경로가 없을 경우에 complete find path, finish return 명령을 통해서 terminated 상태가 될 수 있다.

Gui 객체 상태도



Gui 는 트랜잭션의 상태를 나타내는 클래스이다. 트랜잭션은 다음 세 개의 강태 중 하나를 가진다.

-getImage

■ 먼저 AddOn시스템이 Gui의 객체를 생성하면 Gui는 새로운 트랙젝션을 만들고 start명령을 통해서 getImage상태로 초기화한다

-Paint

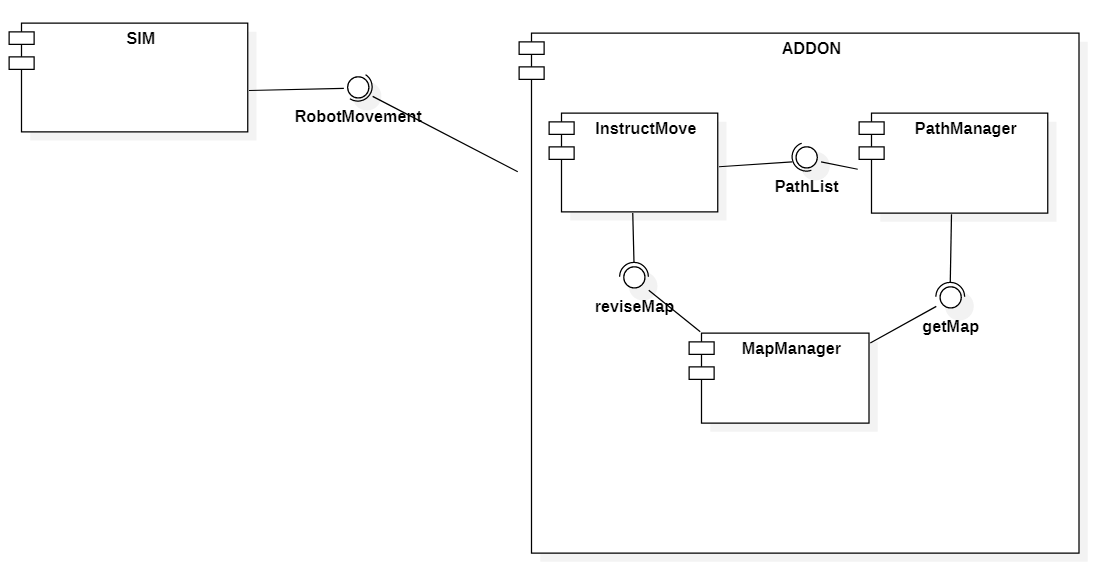
■ AddOn이 지도를 그리는 명령을 요구하면 paintImage 명령을 통해서 Paint 상태로 도입한다.

-Repaint

■ AddOn이 업데이트된 지도를 새로 그려주는 명령을 요구하면 get newfield 명령을 통해서 Repaint상태로 도입한다.

-지도정보가 들어있는 field가 비워져 있으면 empty fieldlist명령을 통해서 teminated상태가 될 수 있다.

Component Package 구성



-SIM component

▶로봇의 현재 상태 즉, 로봇의 방향, 로봇의 현재위치 좌표 정보를 포함하고, RobotMovement 인터페이스를 제공한다.

-ADDON component

▶IntrrucMove, PathManager, MapManager component들로 구성되며 전체적인 기능을 제어하는 역할을 담당한다. RobotMovement 인터페이를 필요로 한다.

InstrucMove component

▶ PathManager, MapManager component에서의 명령을 집합시켜서 수행시키는 역할을 담당한다.

PathList, reviseMap 인터페이스를 필요로 한다.

PathManager component

▶ 경로를 설정하고 이 설정된 경로를 저장하는 기능을 제공하고, PathList인터페이스를 제공한다. getMap 인터페이스를 필요로 한다.

MapManager component

▶ 지도를 생성하고 지도의 변화를 수정하며 지도를 저장하는 기능을 제공한다. reviseMap, getMap 인터페이스를 제공한다.