



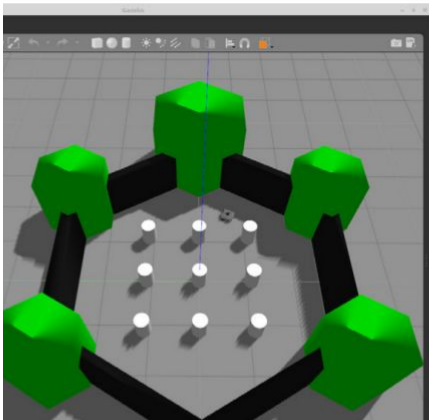
# Navigation2 Behavior Tree @ 오로카 (Navigation2 State Machine)

2020.12.15  
Minwoo Kim

1. Navigation 이란
2. Behavior Tree 란
3. Nav2 가면서 StateMachine으로 behavior tree 추가
4. Nav2 에서 Behavior tree Sequence 설정파일 위치
5. Groot 를 이용한 Nav2 behavior tree를 Visual 하게 보기
  - 5-1 . Groot 를 이용하여 제작한 Behavior tree Visual 하게 보기
  - 5-2 . Groot 를 이용한 Nav2 State 실시간 Monitoring
6. Nav2에서 제공하는 Behavior tree xml 파일 Sequence 분석

# 1. Navigation 이란

## Gazebo Simulation

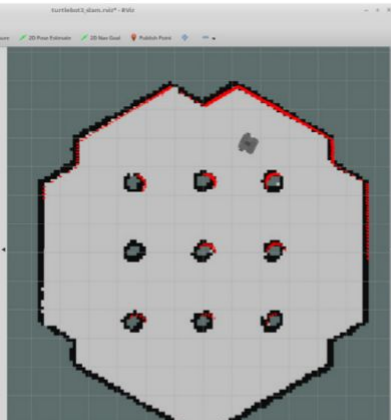


다양한 ros slam 패키지 있음. 선택가능 Turtlebot3 서  
기본 예제로 제공하는 2D lidar slam 종류만도 4개  
- gmapping, karto, cartographer, hector

그외 3D lidar Slam or Visual Slam Ros packages  
- Loam, Lego Loam, HDL, RTAB, VINS, ORB

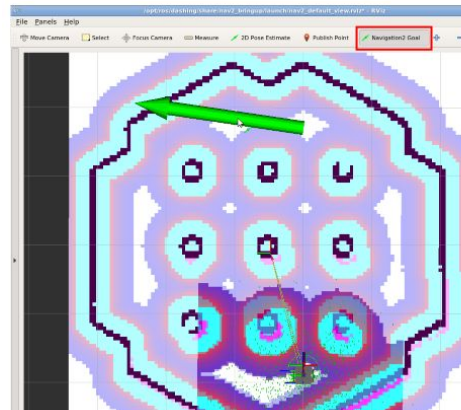
## SLAM

자신의 위치를 추정(Localization)하면  
지도를 만드는 것



## navigation

만들어진 지도를 가지고 장애물을 피해 지정한  
위치로 경로를 생성해 이동 하는 것

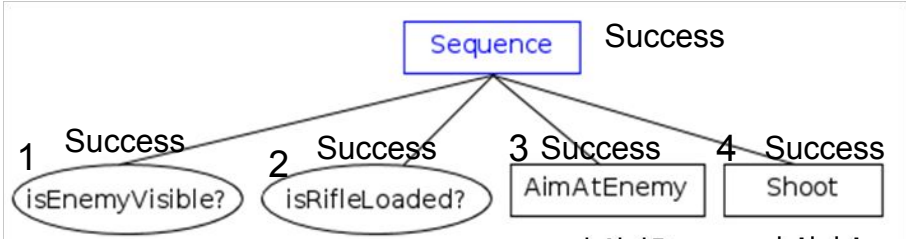


navigation은 여러개의 패키지가 합쳐진 meta 패키지이지만 그 안의  
plugin을 교체 하거나 amcl, map server 를 다른 패키지로 replace 가능  
costmap - static, Obstacle, Inflation, Socal, Range, Voxel etc.. plugin 사용  
local planner - base, dwa, dwb, ted etc..plugin 사용  
global planner - NavfnPlanner, Smac, A\* Path, Dijkstra's etc .. plugin 사용  
amcl - localization 패키지, slam package나 robot\_localization로 대체 가능  
map\_server - 만들어진 map 을 불러옴, slam package 로 대체 가능  
move\_base

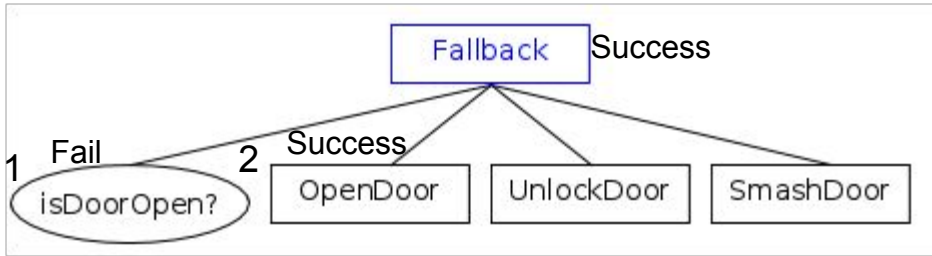
# 2.Behavior Tree 관

게임 업계에서 unity게임 캐릭터 만들때 자주 사용 되는 State Machine 입니다.  
ros2 navigation2 패키지의 경우 behavior tree 를 state machin으로 채택해서 사용하고 있습니다.

간단히 Control node 와 Action Node 로만 구현된 behavior tree 모델



child node 실행 순서 →



용어 설명

Control node - 하위 child node들의 flow 를 control 하는 노드  
Action node - leaf( 종단) node 로 실제 어떠한 행동을 하는 노드  
child node - 어떠한 node 하위에 있는 node 들  
tick - node 실행 trigger

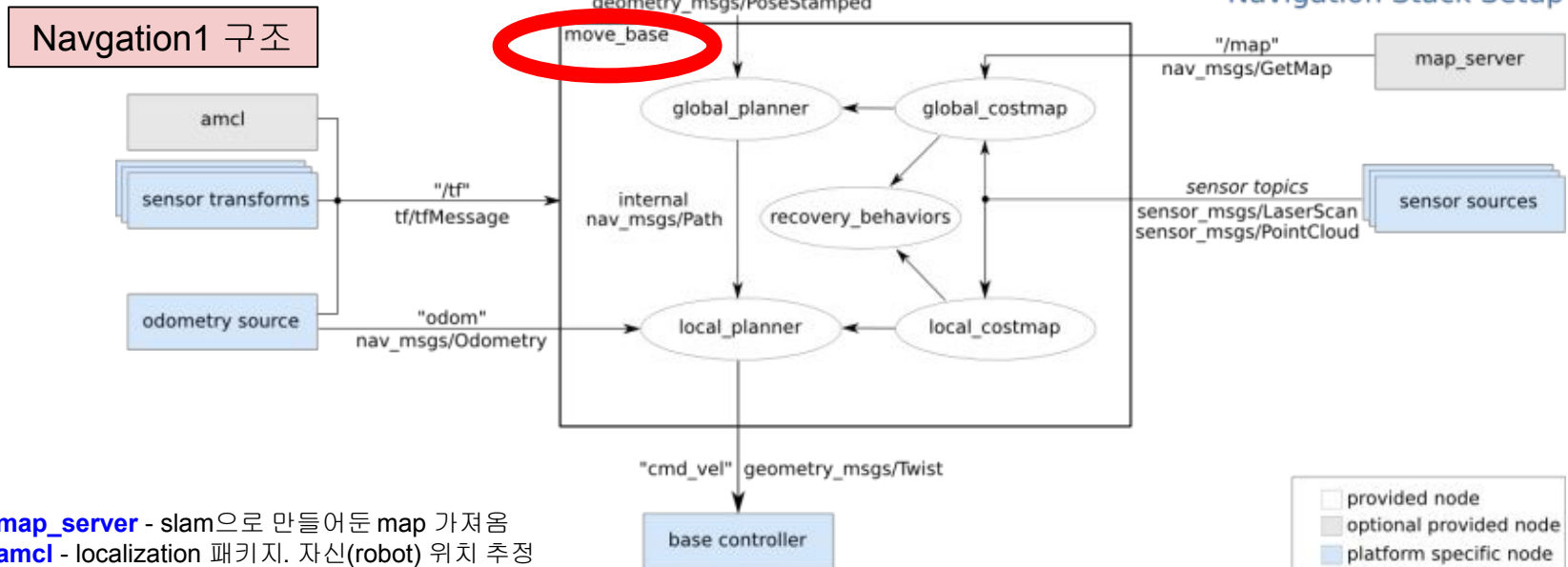
## 1. Sequence Control Node

어떠한 child node가 Success를 반환한다면 다음 child를 tick 한다.  
success 반환 - 모든 child node 가 Success가 되는 경우  
fail 반환 - 왼쪽부터 순차적으로 실행 중 하나의 child node라도 fail 반환

## 2. Fallback Control Node

어떠한 child node가 Failure을 반환해도 다음 child를 tick 한다.  
success 반환 - 왼쪽부터 순차적으로 실행 할때 중 하나의 child node라도 success 반환하는 경우  
fail 반환 - 모든 child node 가 fail 되는 경우

### 3. Nav2 가면서 StateMachine으로 behavior tree 추가



**map\_server** - slam으로 만들어진 map 가져옴

**amcl** - localization 패키지. 자신(robot) 위치 추정

**costmap** - 장애물이 있음을 표시함. **static**, **Obstacle**, **Inflation**, **Socal**, **Range**, **Voxel** etc.. plugin

**global planner** - 위 패키지들로 받는 data 기반으로 목적지로 가는 path를 생성함. **NavfnPlanner**, **Smac**, **A\* Path**, **Dijkstra's** etc .. plugin

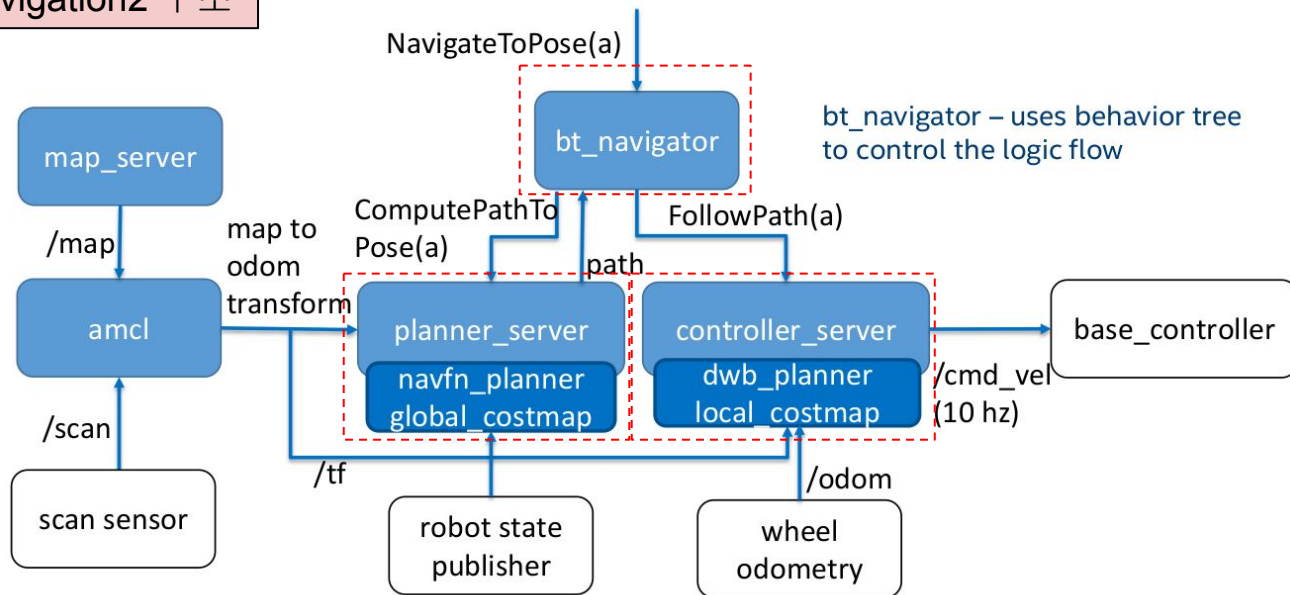
**local planner** - 위 패키지들로 받는 data 기반으로 실제 로봇을 움직이는 controller. **base**, **dwa**, **dwb**, **ted** etc..plugin 사용

**recovery** - navigation 중 문제 발생시 해당 action 취함.

**move base** - 위의 패키지들을 control 하는 역할을 함. 이 패키지가 ros2 navigation2로 가면서 behavior tree를 state machin으로 이용한 bt navigator 노드로 변경이 됨. move base 패키지는 다른 node들을 컨트롤 하는 부분이 하드 코딩 되어 있고 별도의 state machine 을 사용하지 않음. 따라서 내부 flow를 사용자가 custom 하여 수정 할수 없음. 이를 개선하기 위해 ros1 에서도 movebase\_flex 라는 smach와 bt를 사용하여 내부 시퀀스를 변경할수 있는 패키지가 있기는 함.

### 3. Nav2 가면서 StateMachine으로 behavior tree 추가

#### Navigation2 구조



KEY:

nav2 node

plugin

external  
node

/topic

Action(a)

ros1 navigation1	move_base	global planner	global costmap	local planner	local costmap
ros2 navigation2	bt_navigator	planner_server		controller_server	

수정

### 3. Nav2 가면서 StateMachine으로 behavior tree 추가

---

**behavior tree 로 state machine 을 customize 할수 있으면 무엇을 할수 있을까요 ?**

- 지정한 목적지로 도착하는게 아닌 특정 물체를 특정 거리를 두고 따라가게 stateMachine 수정 ( 실제 follow\_point.xml 파일로 예제 제공해주고 있음 )
- 출발하기 전에 자신의 위치가 navigation 이 불가능한 상황인지 판단해 이부분에 대한 action 추가
- 목적지가 navigation 이 불가능한 위치인지 판단해 이부분에 대한 action 추가
- recovery action sequence 수정

**Navigation2 에서 Behavior tree 를 꼭알아야 Navigation 할수 있나요 ?**

- 기본 navigation 을 위한 behavior tree sequence 인 navigate\_w\_replanning\_and\_recovery.xml 를 기본 제공 하기 때문에 굳이 navigation sequence 를 변경할 필요가 없다면 기본 시퀀스로 네비게이션 동작을 하시면 됩니다.  
nav2 도 nav1 처럼 simple goal 이나 action , Rviz를 통해 똑같이 navigation goal 을 할수 있습니다.

## 4. Nav2 에서 Behavior tree Sequence 설정파일 위치

### 1. Behavior tree sequence 파일 설정하는 위치

[https://github.com/ros-planning/navigation2/blob/main/nav2\\_bringup/bringup/launch/bringup\\_launch.py](https://github.com/ros-planning/navigation2/blob/main/nav2_bringup/bringup/launch/bringup_launch.py)

```
75     declare_bt_xml_cmd = DeclareLaunchArgument(  
76         'default_bt_xml_filename',  
77         default_value=os.path.join(  
78             get_package_share_directory('nav2_bt_navigator'),  
79             'behavior_trees', 'navigate_w_replanning_and_recovery.xml'),  
80         description='Full path to the behavior tree xml file to use')
```

### 2. Behavior tree sequence 파일들 위치

[https://github.com/ros-planning/navigation2/tree/main/nav2\\_bt\\_navigator/behavior\\_trees](https://github.com/ros-planning/navigation2/tree/main/nav2_bt_navigator/behavior_trees)

follow\_point.xml

navigate\_w\_replanning\_and\_recovery.xml

navigate\_w\_replanning\_and\_round\_robin\_recovery.xml

navigate\_w\_replanning\_distance.xml

navigate\_w\_replanning\_speed.xml

navigate\_w\_replanning\_time.xml

nav2 동작시 기본 호출 되는  
behavior tree sequence xml 파일

원하는 behavior tree sequence xml 파일을  
만들어 위에서 호출해 주면됨



## 4. Nav2 에서 Behavior tree Sequence 설정파일 위치

navigate\_w\_replanning\_and\_recovery.xml 파일 내용

```
<root main_tree_to_execute="MainTree">
  <BehaviorTree ID="MainTree">
    <RecoveryNode number_of_retries="6" name="NavigateRecovery">
      <PipelineSequence name="NavigateWithReplanning">
        <RateController hz="1.0">
          <RecoveryNode number_of_retries="1" name="ComputePathToPose">
            <ComputePathToPose goal="{goal}" path="{path}" planner_id="GridBased"/>
            <ClearEntireCostmap name="ClearGlobalCostmap-Context" service_name="global_costmap/clear_entirely_global_costmap"/>
          </RecoveryNode>
        </RateController>
        <RecoveryNode number_of_retries="1" name="FollowPath">
          <FollowPath path="{path}" controller_id="FollowPath"/>
          <ClearEntireCostmap name="ClearLocalCostmap-Context" service_name="local_costmap/clear_entirely_local_costmap"/>
        </RecoveryNode>
      </PipelineSequence>
      <ReactiveFallback name="RecoveryFallback">
        <GoalUpdated/>
        <SequenceStar name="RecoveryActions">
          <ClearEntireCostmap name="ClearLocalCostmap-Subtree" service_name="local_costmap/clear_entirely_local_costmap"/>
          <ClearEntireCostmap name="ClearGlobalCostmap-Subtree" service_name="global_costmap/clear_entirely_global_costmap"/>
          <Spin spin_dist="1.57"/>
          <Wait wait_duration="5"/>
        </SequenceStar>
      </ReactiveFallback>
    </RecoveryNode>
  </BehaviorTree>
</root>
```

**branch 1**  
global path  
생성하고  
Controller  
(local planner)  
로 해당 경로  
따라가는  
sequence  
branch

**branch2**  
recovery 상황  
발생시 recovery  
action을 어떠한  
것을 어떤  
순서로 취할  
것인지에 대한  
sequence  
branch

## 5-1 . Groot 를 이용하여 제작한 Behavior tree Visual 하게 보기

### 1.Groot 설치 ( 로스 전용 프로그램 아님 )

navigation2 에 Groot 관련 readme 가 있음. 해당 링크서 설치법 설명 된 링크를 링크 하고 있음.

[https://github.com/ros-planning/navigation2/blob/main/nav2\\_behavior\\_tree/groot\\_instructions.md](https://github.com/ros-planning/navigation2/blob/main/nav2_behavior_tree/groot_instructions.md)

### 2.Groot 실행

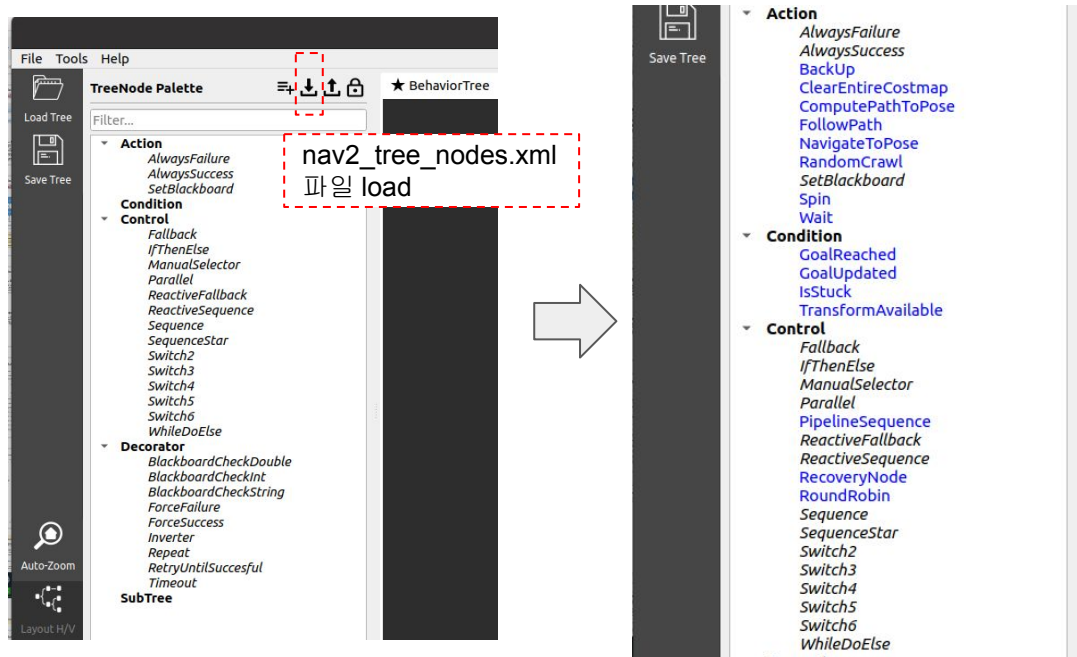
minwoo@minwoo:~/Groot/build\$ ./Groot

### 3.Groot editor mode 선택하고 Start 누름



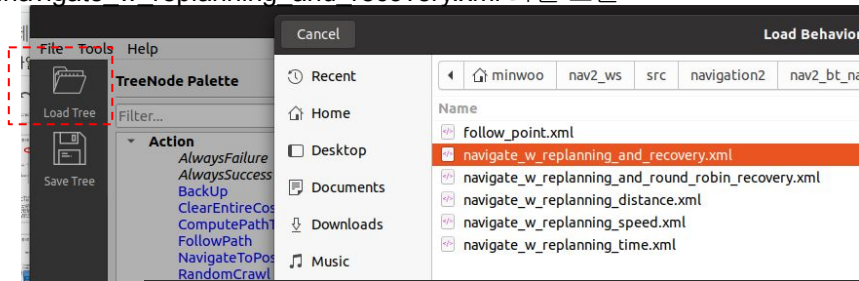
4. navigate\_w\_replanning\_and\_recovery.xml 의 경우 BehaviorTree.CPP 에서 기본 제공하는 node 외에 Nav2 전용 Custom node 를 만들어 사용 하는게 많기 때문에 해당 nav2 custom node 를 정의 해둔 xml 파일을 따로 load 해주어야함. 해당 파일은 아래 위치에 있음

[https://github.com/ros-planning/navigation2/blob/main/nav2\\_behavior\\_tree/nav2\\_tree\\_nodes.xml](https://github.com/ros-planning/navigation2/blob/main/nav2_behavior_tree/nav2_tree_nodes.xml)

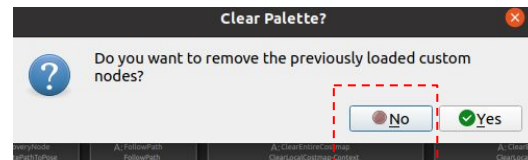


## 5-1 . Groot 를 이용하여 제작한 Behavior tree Visual 하게 보기

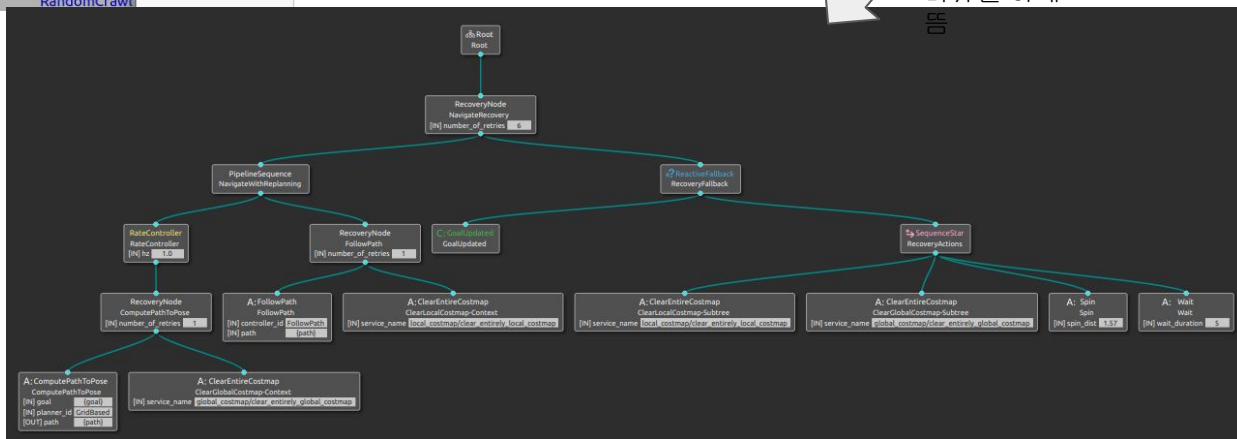
## 5.navigate w replanning and recovery.xml 파일 호출



이전에 열어 놓은게 있으면 아래 팝업창이 뜬



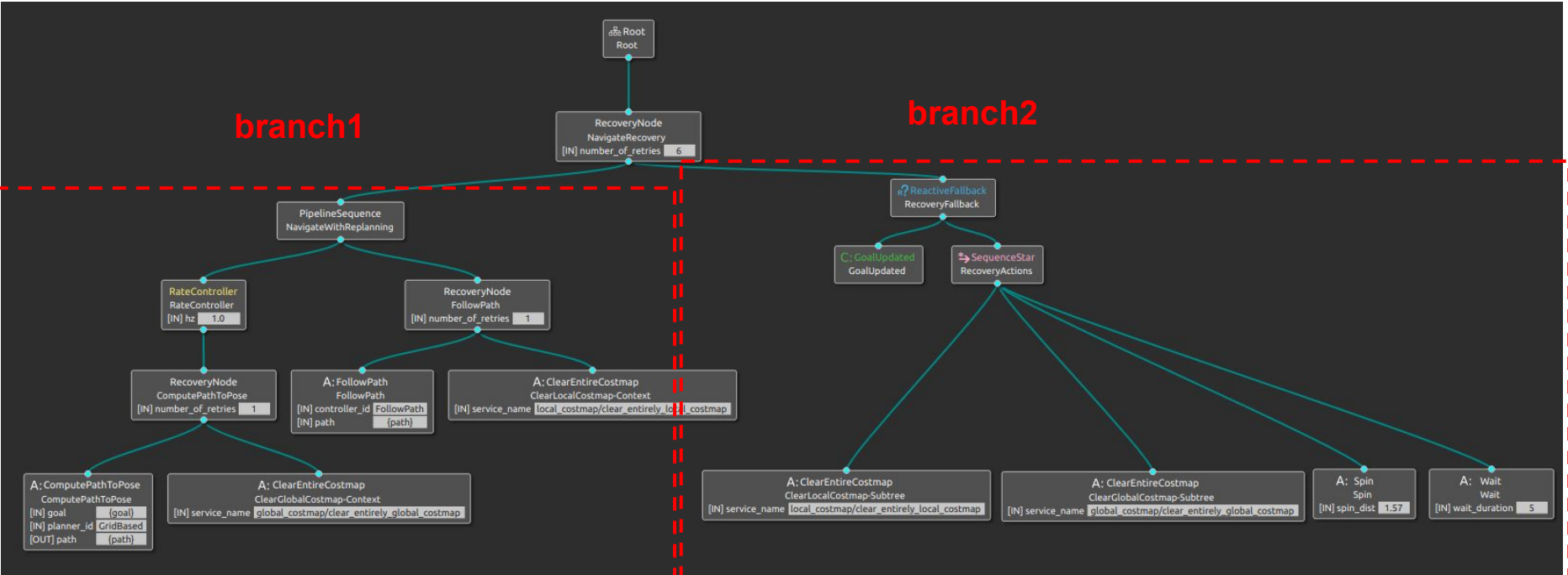
비주얼 하게



navigate\_w\_replanning\_and\_recovery.xml 에 대한 설명은 아래 링크에도 설명이 되어있음.

[https://github.com/ros-planning/navigation2/tree/main/nav2\\_bt\\_navigator#navigate-with-replanning-and-simple-recovery-actions](https://github.com/ros-planning/navigation2/tree/main/nav2_bt_navigator#navigate-with-replanning-and-simple-recovery-actions)

# 5-1 . Groot 를 이용하여 제작한 Behavior tree Visual 하게 보기



global path 생성하고 controller( local planner ) 로 해당 경로 따라가는 sequence

recovery 상황 발생시 recovery action을 어떤 순서로 어떻게 취할 것인지에 대한 sequence

## 5-1 . Groot 를 이용한 Nav2 State 실시간 Monitoring

Monitor mode PR 은 12월 9일에 main branch 에만 PR 반영 되었음. 따라서 지금은 쓰려면 nav2 main branch 빌드해서 써야지만 사용 가능.

<https://github.com/ros-planning/navigation2/pull/1958> 참고로 nav2 main branch 는 ros2 특정 버전을 위한 브랜치가 아닌 범용적 브랜치라 foxy서 쓰려면 조금 수정 해야 함.

1.nav2 param yaml 파일에 monitor 파라미터 True 설정 ,

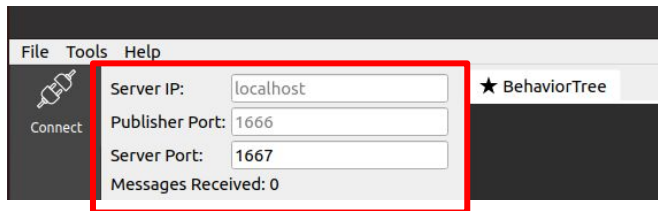
[https://github.com/ros-planning/navigation2/blob/main/nav2\\_bringup/bringup/params/nav2\\_params.yaml](https://github.com/ros-planning/navigation2/blob/main/nav2_bringup/bringup/params/nav2_params.yaml)

```
50 bt_navigator:  
51   ros__parameters:  
52     use_sim_time: True  
53     global_frame: map  
54     robot_base_frame: base_link  
55     odom_topic: /odom  
56     enable_groot_monitoring: True  
57     groot_zmq_publisher_port: 1666  
58     groot_zmq_server_port: 1667
```

2.Groot monitor mode 선택하고 Start 누름



3. nav2가 pose estimation 까지 되어 costmap도 전부 활성화 된 상태에서 Connect 를 누르면 Monitoring 이 시작 됨.



특별히 설정할 것 없음. 동작 영상 <https://youtu.be/eTQTjQAoanY>

## 6 . Nav2에서 제공하는 Behavior tree xml 파일 Sequence 분석

[https://github.com/ros-planning/navigation2/tree/main/nav2\\_bt\\_navigator/behavior\\_trees](https://github.com/ros-planning/navigation2/tree/main/nav2_bt_navigator/behavior_trees)

구조설명하기 좋은 간단한 Sequence Tree. 하지만 recovery action sequence 가 빠져있어 해당 부분 추가 없이 실제 쓰기는 어려운듯 함.

 `navigate_w_replanning_distance.xml`

로봇이 일정 거리를 움직일때 마다 global path 생성하고 Controller가 로봇을

 `navigate_w_replanning_speed.xml`

움직임

로봇이 일정 속도로 움직일때 마다 global path 생성하고 Controller가 로봇을

 `navigate_w_replanning_time.xml`


움직임

특정 시간 주기로 global path 생성하고 Controller가 로봇을 움직임

recovery action sequence 가 구현되어있어 실제 사용 가능한 sequence tree

 `navigate_w_replanning_and_recovery.xml`

Navigation Default로 동작하는 Sequence Tree

 `navigate_w_replanning_and_round_robin_recovery.xml`

Default로 동작하는 Sequence Tree에서 recovery action sequence 를 변경함.

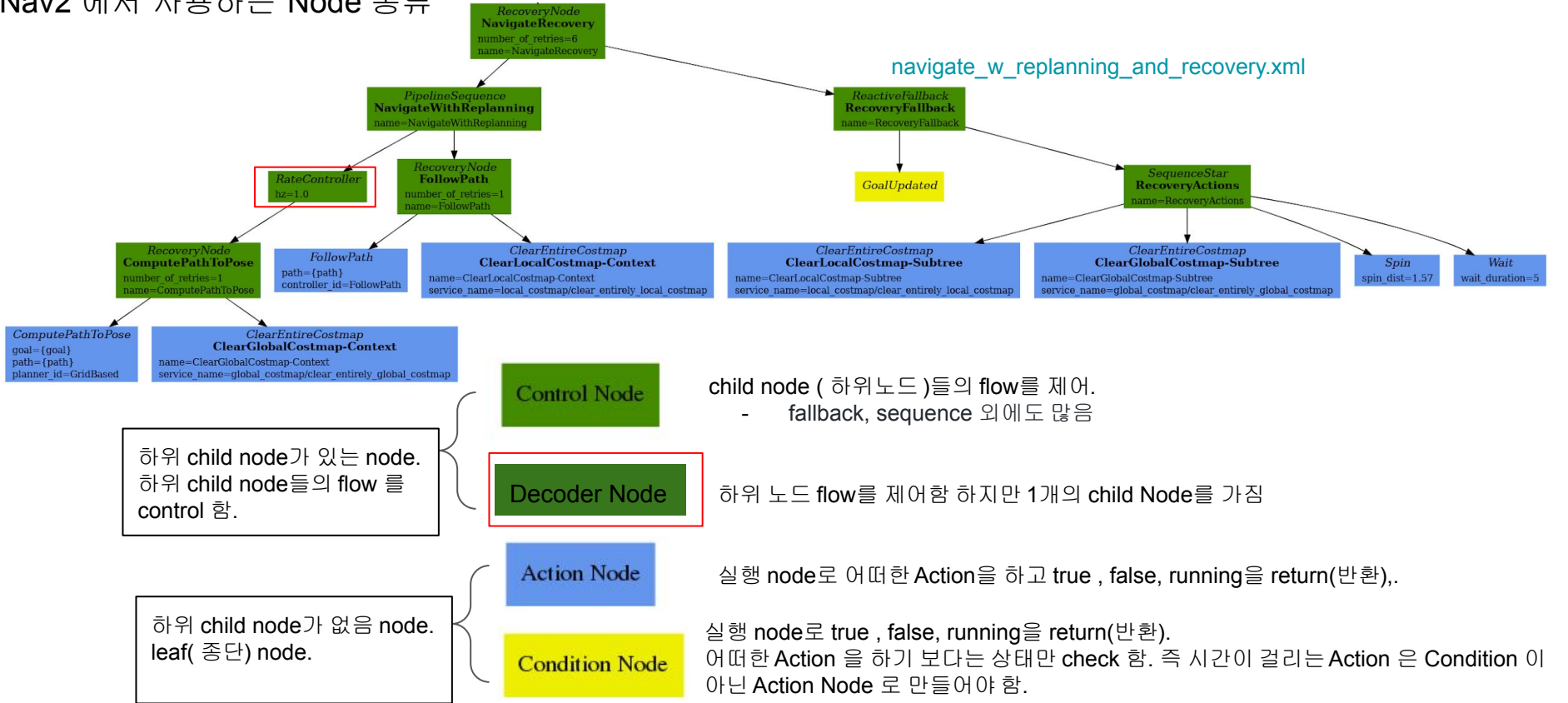
Navigation 기능이 아닌 특정 물체를 특정 거리를 유지하면 계속 따라다니도록 변경된 sequence tree.  
하지만 recovery action sequence 가 빠져있어 해당 부분 추가 없이 실제 쓰기는 어려운듯 함.

 `follow_point.xml`



# 6-1. Nav2 기본 Behavior tree Sequence 분석하기 위해 필요한 내용

## Nav2 에서 사용하는 Node 종류



# 6-1. Nav2 기본 Behavior tree Sequence 분석하기 위해 필요한 내용

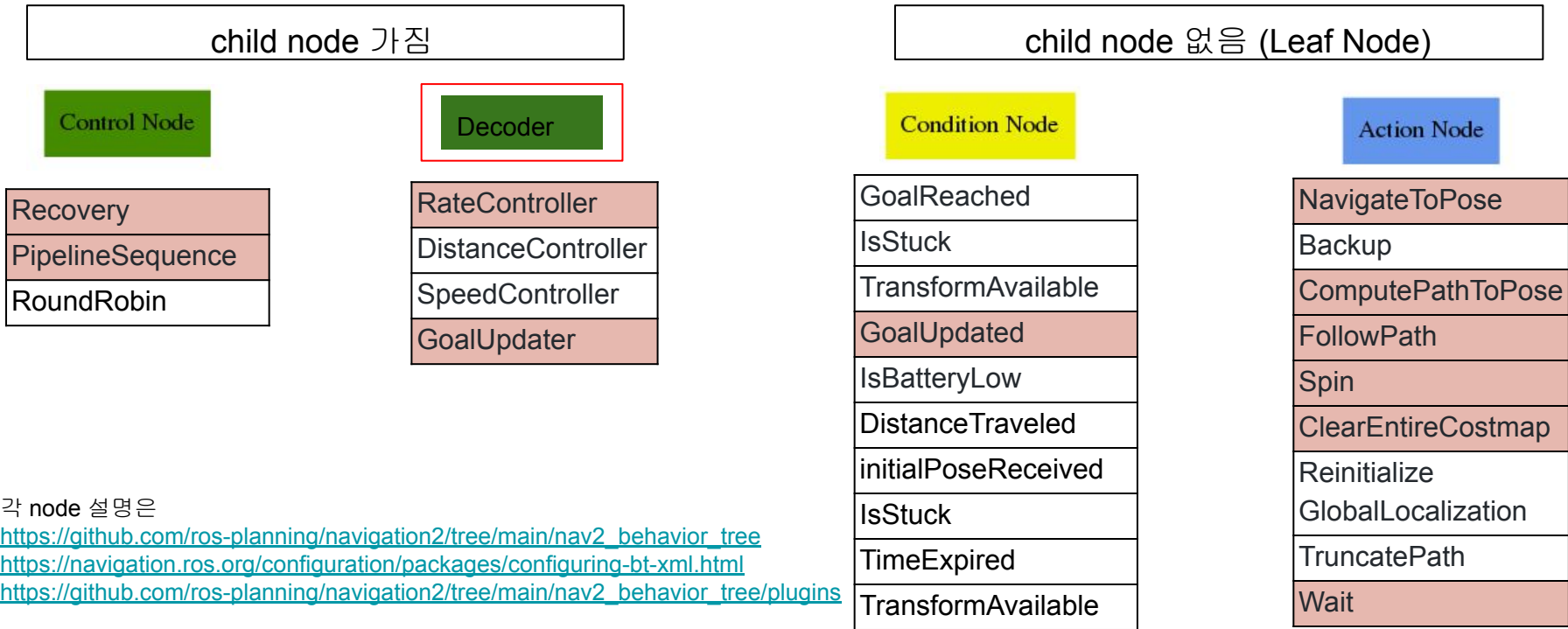
Behavior Tree.CPP 라이 블러리에서 제공하는 Control Node . 종류가 많지만 일단 대표 Control Node 만 설명  
navigate\_w\_replanning\_and\_recovery.xml에서 사용하는 Node

Node Type	Control Node 종류	Description	사용 사례
Sequence type (BT 기본 library)	Sequence	child가 Fail return 시 다음 tick 에서 전체 Sequence Restart 하여 첫번째 child Tick , child가 Running return 시 다음 tick 에서 해당 child 다시 Tick	child 1,2,3 가 전부 성공해야 성공을 보내고 싶을 때 쓰는 구조
	ReactiveSequence	child가 Fail return 시 다음 tick 에서 전체 Sequence Restart 하여 첫번째 child Tick , child가 Running return시 다음 tick 에서 전체 Sequence Restart 하여 첫번째 child Tick	child2 가 runing 중 Tick이 들어오면 child1 에서부터 다시 시작 되므로 child2 running중에도 child1을 계속 실행시켜 Codition 을 확인 할때 사용되는 구조
	SequenceStar	child가 Fail return 시 다음 tick 에서 해당 child 다시 Tick child가 Running return 시 다음 tick 에서 해당 child 다시 Tick	정찰 로봇이 Locaion A,B,C 를 각가 한번씩만 방문해야 할때 사용되는 구조
Fallback type의 (BT 기본 library)	Fallback	child가 Running return 발생시 다음 Tick 실행시 Fallback Restart 하여 첫번째 child Tick ,	child 1,2,3 가 하나라도 성공하면 성공을 보내고 싶을 때 쓰는 구조
	ReactiveFallback	FallBack과 차이를 잘 모르겠음	캐릭터가 지칠 경우(child1)나 8시간 활동할 경우(child2) Success 를 보낼 경우 child1 은 이미 Fail을 return 해서 child2 가 running을 8시간 동안 하면서 running 중에서 child1을 통해 캐릭터가 지쳤는지 child1 의 상태 변화를 확인하기 위해 필요한 시퀀스
	FallbackStar	child가 Running return 시 다음 tick 에서 해당 child 다시 Tick	-에시 없음



# 6-1. Nav2 기본 Behavior tree Sequence 분석하기 위해 필요한 내용

Nav2 Behavior Tree 패키지서 구현된 Custom Node List ( BehaviorTree.CPP 라이브러리서 기본 제공 않는 Node )  
navigation2 / nav2\_behavior\_tree / plugins / navigate\_w\_replanning\_and\_recovery.xml에서 사용하는 Node



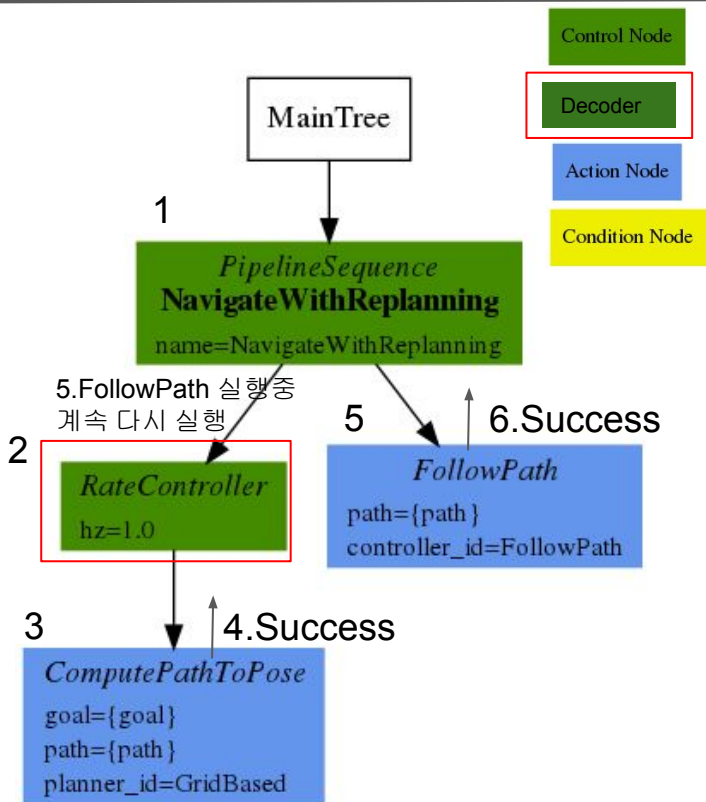
# 6-1. Nav2 기본 Behavior tree Sequence 분석하기 위해 필요한 내용

## Nav2 Behavior Tree 패키지서 구현된 Custom Control Node 설명

	Description	사용 사례
Pipeline Sequence Control Node	<p>이것은 Sequence Node 의 한종류 이다. 우선 첫번째 child가 성공할때 까지 tick 한다. 첫번째 child가 성공하면 그다음에는 첫번째 child 와 두번째 child 를 두번째 child가 성공할때까지 동시에 tick을 한다.</p> <p>두번째 child가 성공하면 다음으로는 첫번째 child, 3번째 child를 tick한다. 역시 3번째가 성공할때 까지 동시에 tick 한다. 더이상 tick할 child 가 없어질때까지 반복한다. 그리고 해당 시퀀스 노드는 마지막 child 가 success 를 반환할때 까지 running을 반환 한다. 그리고 마지막 child node 가 Success 를 반환하면 해당 sequence 도 success 를 반환한다. 만약 어떤 child 가 failure 를 반환하면 모든 노드가 정지하고 해당 sequence node는 failure 를 반환한다.</p>	<p>Nav2에서 ComputePathTo pose 와 FollowPath 연결하는 Seaqueunce node로 사용됨.</p> <p>ComputePathTo 가 성공해서 FollowPath로 넘어가도 FollowPath 가 success 를 반환하기 전까지 ComputePathTo 가 동시에 계속 같이 Tick 이 됨.</p>
Recovery Control Node	<p>이 Control node 는 두개의 child node 를 가진다.</p> <p>오로지 첫번째 child가 성공을 반환해야지만 Success 를 retrun 한다. 두번째 child는 오로지 첫번째 child가 failure 되었을 때만 실행이 된다.</p> <p>따라서 두번째 child 는 recovery action 을 책임 지고 있다.</p> <p>만약 두번째 child에 의해서 recovery 가 성공한다면 첫번째 child 가 다시 실행이 된다. 그리고 사용자는 해당 node 가 failure 를 return 하기전에 몇번의 recovery action 을 시도할지 정할 수 있다.</p>	<p>기본 xml 에서 Navigate Recovery Node 는 child1 로 패스를 생성하고 따라가는 Navigation With Replanning 노드를 가진다.</p> <p>이 child1이 Fail 을 return 하면 child2인 Recovery Fallback node 가 실행됨.</p> <p>Child2에서 리커버리 액션들을 실행하고 리커버리가 성공하면 다시 child1이 실행이 된다.</p>
Round Robin Control Node	<p>첫번째 child가 Success 나 Failure 를 return 할때 까지 첫번째 child를 실행한다. 만약 child 가 Success 나 Failure 를 return 한다면 해당 Sequence 는 다음 child 를 Tick한다.</p> <p>Sequence node와 Round Robin 의 가장 큰 차이는 child 가 Failure 를 return 했을 때 RoundRobin 은 다음 child 를 Tick 하고 Sequence node 는 Failure 을 반환한다.</p>	<p>RoundRobin 이 child 1 로 clear Global map , child2 로 wait 를 두게 되면 child1 이 Success, Fail 그 무엇을 반납하든 wait 가 무조건 동작 한다</p>

각 node 설명은 [https://github.com/ros-planning/navigation2/tree/main/nav2\\_behavior\\_tree](https://github.com/ros-planning/navigation2/tree/main/nav2_behavior_tree) 여기에도 되어있고 설명이 안된 node 는 [https://github.com/ros-planning/navigation2/tree/main/nav2\\_behavior\\_tree/plugins](https://github.com/ros-planning/navigation2/tree/main/nav2_behavior_tree/plugins) 여기에 각 nav2 custom node code가 있으니 보면 되기는함.

## 6-2. navigate\_w\_replanning\_time.xml 설명



### Pipeline Sequence Control Node

child node 전부 성공해야 성공 반환하는 이것은 Sequence Node 의 한종류 이다. 첫번째 child가 성공하고 두번째 child를 실행할때 두번째 child가 실행중에도 첫번째 child 도 동시에 실행을 합니다.

### RateController Decorator Node

하위 child Node 실행 주기를 Control하는 Node 여기서는 1hz로 설정함.

### ComputePathToPose Action Node

최종 목적지에 대한 Goal 과 어떠한 Global planner Plugin In을 사용할지에 대한 `planner_id` 을 input으로 받음.

여기서는 `planner_id` 로 `GridBased` 가 설정이 되어있고 `nav2_params.yaml` 파일을 보면 `GridBased` 는 `nav2_navfn_planner/NavfnPlanner`로 설정이 되어있음. 이렇게 생성된 path 를 output으로 던져줌.

```
planner_server:
  ros_parameters:
    expected_planner_frequency: 20.0
    use_sim_time: True
    planner_plugins: ["GridBased"]
  GridBased:
    plugin: "nav2_navfn_planner/NavfnPlanner"
```

### FollowPath Action Node

input으로 `ComputePathToPose` Action Node에서 생성한 global path 와 어떠한 controller\_id( 어떠한 Local planner Plug In 을 쓸건지 ) 를 받음. 여기서는 `FollowPath`로 설정이 되어있고 `nav2_params.yaml` 파일을 보면 `FollowPath`는 `DWBLocalPlanner`로 설정이 되어있음.

`nav2_bringup/bringup/params/nav2_params.yaml`

```
controller_server:
  ros_parameters:
    use_sim_time: True
    controller_frequency: 20.0
    min_x_velocity_threshold: 0.001
    min_y_velocity_threshold: 0.5
    min_theta_velocity_threshold: 0.001
    progress_checker_plugin: "progress_checker"
    goal_checker_plugin: "goal_checker"
    controller_plugins: ["FollowPath"]

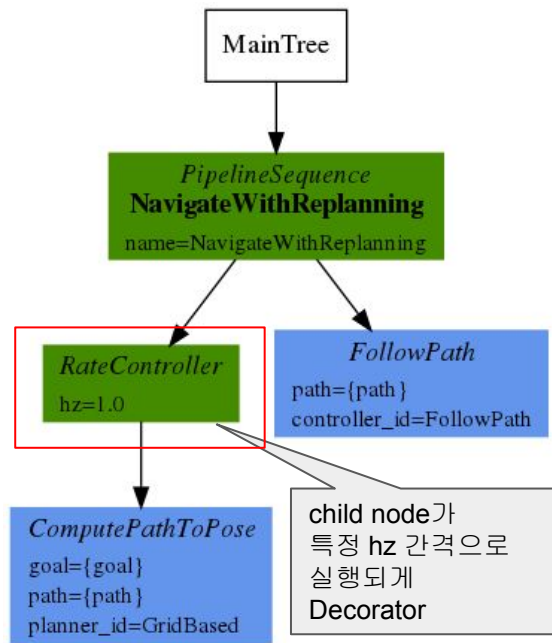
# DWB parameters
FollowPath:
  plugin: "dwb_core::DWBLocalPlanner"
  debug_trajectory_details: True
```

recovery Action 관련된 부분이 빠져 있음 실제 사용시 이 부분 추가 필요.

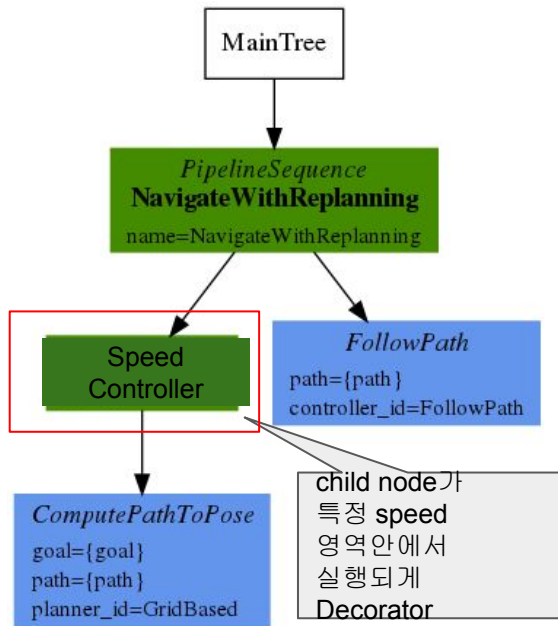
## 6-3. navigate\_w\_replanning\_speed.xml and navigate\_w\_distance.xml 설명

아래 3개의 behavior sequence xml 파일은 Compute PathToPose Action Node 의 주기를 Control 하는 Decorator 의 종류만 틀림.

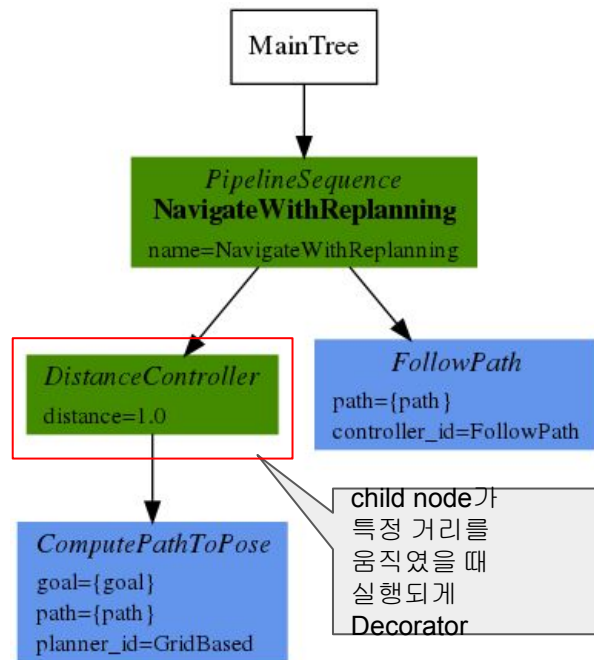
navigate\_w\_replanning\_time.xml



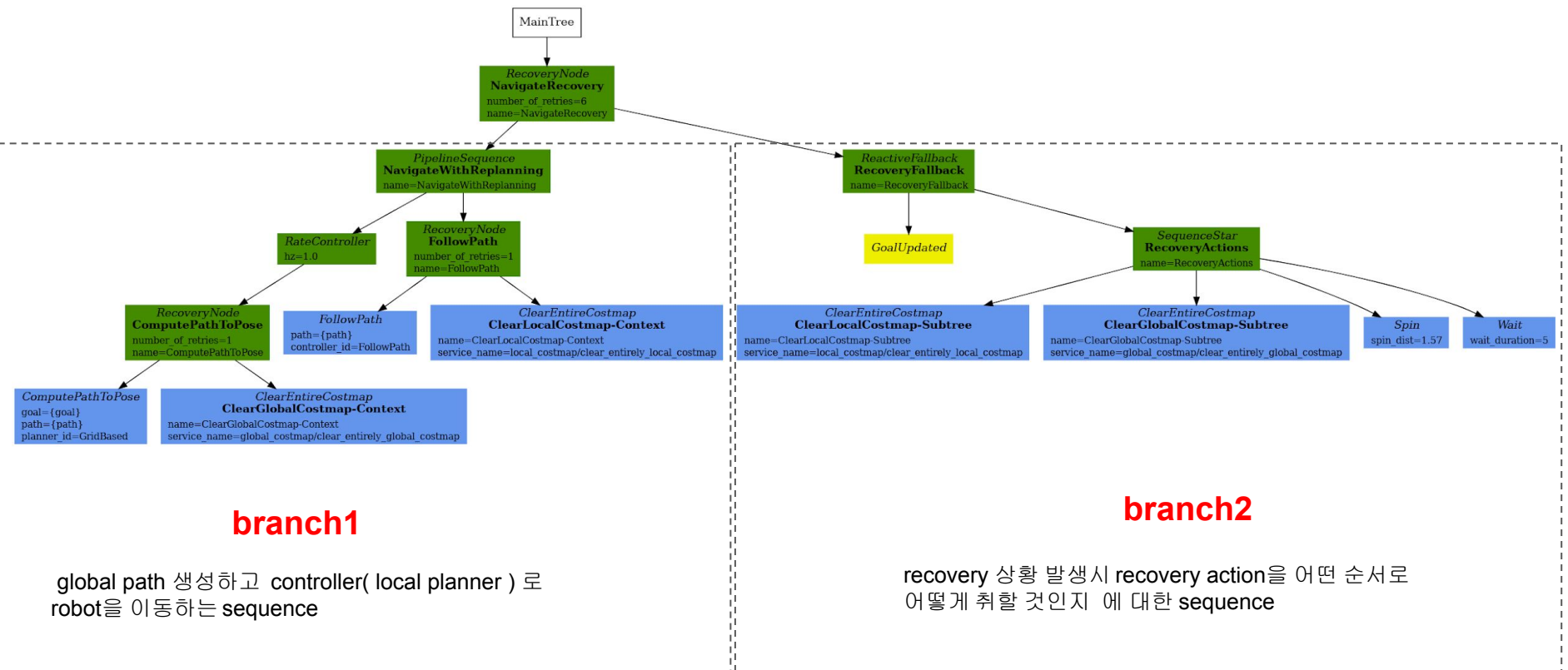
navigate\_w\_replanning\_speed.xml



navigate\_w\_replanning\_distance.xml



# 6-4. navigate\_w\_replanning\_and\_recovery.xml 설명



# 6-4. navigate\_w\_replanning\_and\_recovery.xml branch1 설명

실제 global path 를 생성하고 생성된 global path 를 따라가는 Controller(local plan) 을 실행하는 부분임.

## Recovery Control Node

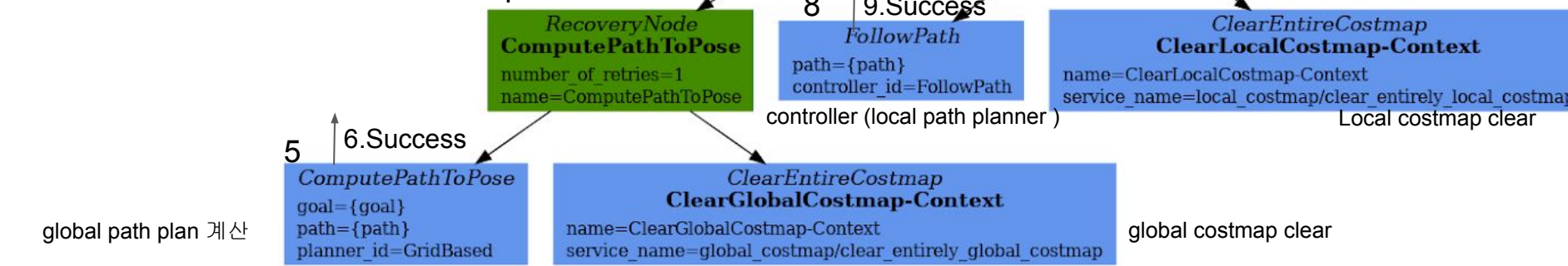
이 Control node 는 두개의 child node 를 가진다.  
오로지 첫번째 child가 성공을 반환해야지만 Success 를 retrun 한다. 두번째 child는 오로지 첫번째 child가 failure 되었을 때만 실행이 된다. 따라서 두번째 child 는 recovery action 을 책임 지고 있다. 만약 두번째 child에 의해서 recovery 가 성공한다면 첫번째 child 가 다시 실행이 된다

## Pipeline Sequence Control Node

child node 전부 성공해야 성공 반환하는 이것은 Sequence Node 의 한종류 이다.  
첫번째 child가 성공하고 두번째 child를 실행할때 두번째 child가 실행중에도 첫번째 child 도 동시에 실행을 합니다.

## RateController Decorator Node

하위 child Node 실행 주기를 Control하는 Node  
여기서는 time 기반으로 1hz로 설정함.





# 6-4. navigate\_w\_replanning\_and\_recovery.xml branch2 설명

이쪽 브런치는 recovery action 을 수행하는 브런치임.  
여기서는 behavior tree.CPP 기본 Library서 제공해주는 ReactiveFallback과 SequenceStar Control Node도 사용함.

Control Node

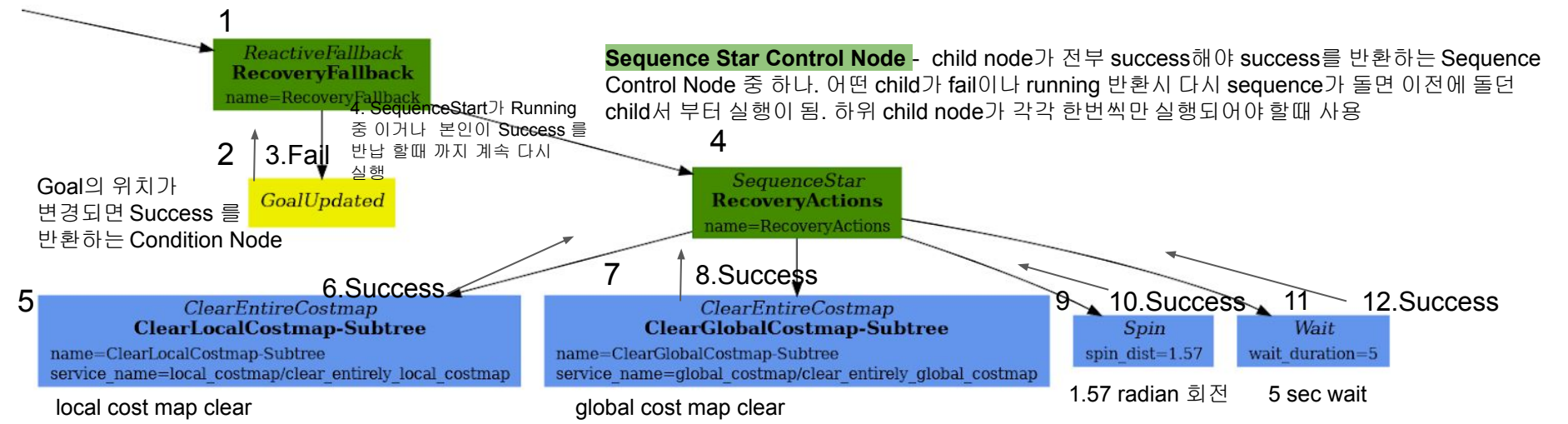
Decoder

Action Node

Condition Node

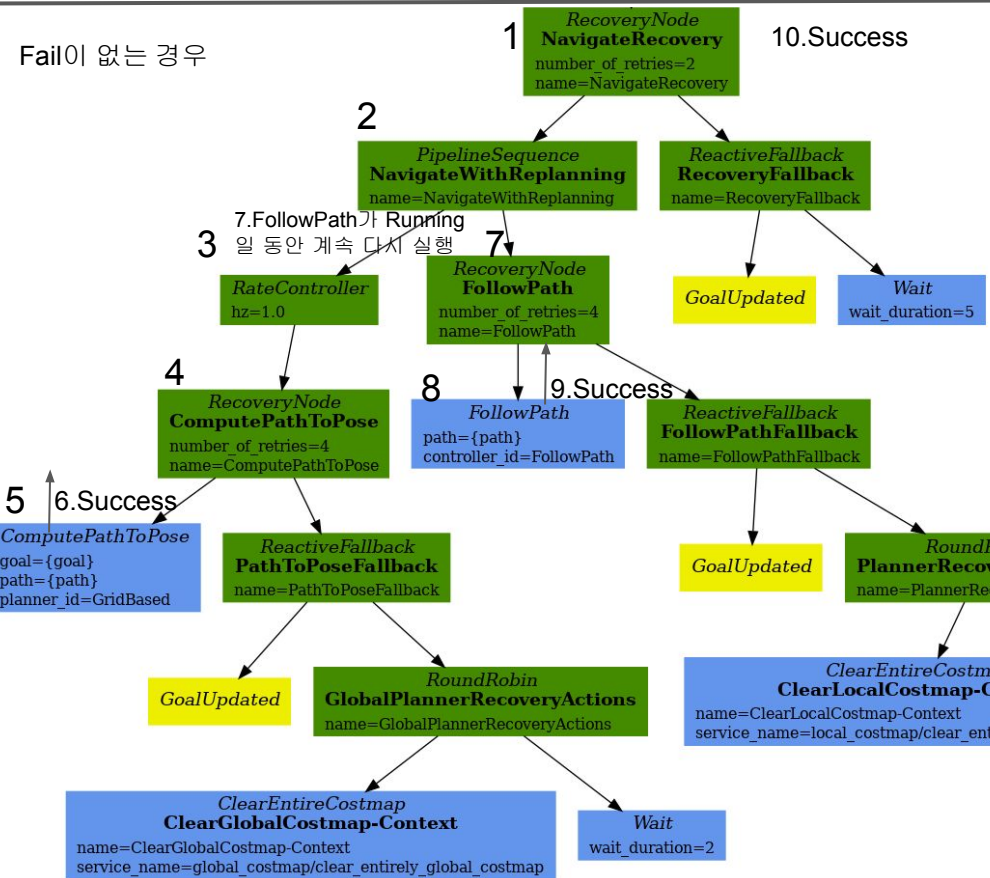
**Reactive Fallback Control Node** - child node 중 하나만 success해도 success를 반환하는 Fallback Control Node 중 하나. 첫번째 child node가 Fail 을 반환하고 두번째 child node가 실행 중이라 running을 반환하는 상황서 sequence가 다시 돌아도 첫번째 child node 부터 시작함. 즉 두번째 child node 실행중에도 첫번째 child node 를 계속 확인해야 하는 경우 사용

**Sequence Star Control Node** - child node가 전부 success해야 success를 반환하는 Sequence Control Node 중 하나. 어떤 child가 fail이나 running 반환시 다시 sequence가 돌면 이전에 돌던 child서 부터 실행이 됨. 하위 child node가 각각 한번씩만 실행되어야 할때 사용



# 6-5 . navigate\_w\_replanning\_and\_round\_robin\_recovery.xml 설명

Fail이 없는 경우



## Recovery Control Node

이 Control node 는 두개의 child node 를 가진다.  
오로지 첫번째 child가 성공을 반환해야지만 Success 를 return 한다. 두번째 child는 오로지 첫번째 child가 failure 되었을 때만 실행이 된다.

## Pipeline Sequence Control Node

child node 전부 성공해야 성공 반환하는 이것은 Sequence Node 의 한종류 이다.  
첫번째 child가 성공하고 두번째 child를 실행할때 두번째 child가 실행중에도 첫번째 child도 동시에 실행을 합니다.

## Reactive Fallback Control Node

child node 중 하나만 success해도 success를 반환하는 Fallback Control Node 중 하나.  
첫번째 child node ( GoalUpdated Condition Node) 가 Fail 을 반환해서 두번째 child node( Sequence Star Control Node ) 실행 중이더라도 두번째 child node가 동작 시간이 오래 걸려 중간에 running을 반환하여 Sequence가 다시 실행 되더라도 첫번째 child node ( GoalUpdated Condition Node)를 실행하여 상태를 체크하게 함.

## Round Robin Control Node

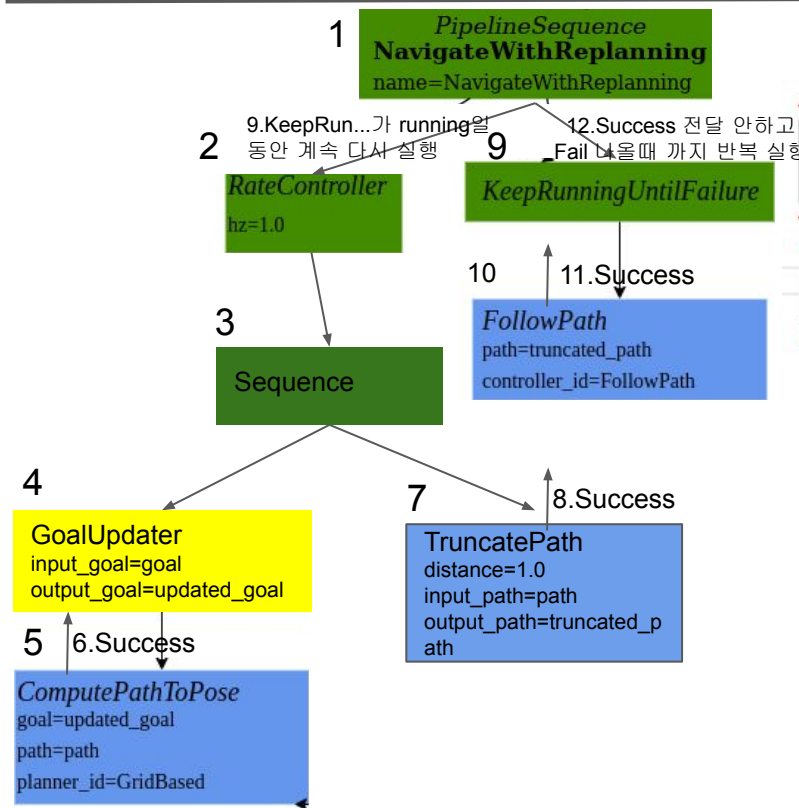
첫번째 child가 Success 나 Failure 를 return 할때 까지 첫번째 child를 실행한다. 만약 child 가 Success 나 Failure 를 return 한다면 해당 Sequence 는 다음 child 를 Tick한다.

node와 Round Robin 의 가장 큰 차이는 child 가 Failure 를 return 했을 때 RoundRobin 은 다음 child 를 Tick 하고 Sequence node 는 Failure 을 반환한다.

RoundRobin 이 child 1 로 clear Global map , child2 로 wait 를 두게 되면 child1 의 Success, Fail 그 무엇을 반납하든 wait 가 무조건 동작 한다.



## 6-6. follow\_point.xml 설명



nav2\_tree\_nodes.xml 파일에 TruncatePath action node 와 GoalUpdater Decorator node 정의를 아래 처럼 추가해 주어야 follow\_point.xml 파일은 Groot 서 open 가능

```

<Action ID="TruncatePath">
  <input_port name="distance">1.0</input_port>
  <input_port name="input_path">path</input_port>
  <output_port name="output_path">truncated_path</output_port>
</Action>

<Decorator ID="GoalUpdater">
  <input_port name="input_goal">goal</input_port>
  <output_port name="output_goal">updated_goal</output_port>
</Decorator>
  
```

### Pipeline Sequence Control Node

child node 전부 성공해야 성공 반환하는 이것은 Sequence Node 의 한종류 이다.

첫번째 child가 성공하고 두번째 child를 실행할때 두번째 child가 실행중에도 첫번째 child 도 동시에 실행을 합니다.

### GoalUpdater Decorator Node

Nav 시작 시점에서 받은 Goal이 아닌 Subscribe를 통해 Goal 을 update 한다. 그리고 이걸로 child branch 가 ComputePathToPose 를 하게 한다. 실시간 어떠한 물체를 따라가는 경우에 이러한 Decorator Node 구현이 필요

### TruncatePath Action Node

distance 파라미터에 정의 한 만큼 ComputePathToPose Action Node 생성한 global path 를 잘라준다. 즉 목적지까지 global path 를 생성하는게 아닌 distance 파라미터에 정의 한 만큼 여기서는 1M 만큼 짜른다. 여기서는 이 잘린 global path 를 FollowPath Action Node가 받아서 기체가 움직인다.

### KeepRunningUntilFailure Decorator Node

하위 child 노드가 Success를 반환해도 전달 하지 않는 Node. 하위노드가 Failure나 Running 을 전달하면 그대로 위에 전달함. 따라가기 기능에서 목적지에 한번 도착했다고 Navigation이 종료되지 않으려면 recovery Action 관련된 부분이 빠져 있음 실제 사용시 이부분 추가 필요해

즉 위에는 PipelineSequence이므로 KeepRunningUntilFailure가 Fail을 보내야지만 해당 전체 Sequence가 종료가 됨.

# Reference

---

1. roscon 2019 navigation2 overview
  - [https://roscon.ros.org/2019/talks/roscon2019\\_navigation2\\_overview\\_final.pdf](https://roscon.ros.org/2019/talks/roscon2019_navigation2_overview_final.pdf)
  - <https://vimeo.com/378682188>
2. navigation2 tutorial
  - <https://navigation.ros.org/>
3. navigation2 git
  - <https://github.com/ros-planning/navigation2>
4. behavior tree.cpp 설명서  
<https://www.behaviortree.dev/>
5. Groot git  
<https://github.com/BehaviorTree/Groot>
6. behavior tree.cpp git  
<https://github.com/BehaviorTree/BehaviorTree.CPP>