How custom tasks are defined, assigned, and executed in OpenRMF
Operational challenges with multiple fleets

Modelling and Assignment of Composable Tasks
The need for robotic interoperability is on the rise
Managing daily operations of multiple robot fleets is challenging.
Functional requirements for a task management framework

**Requirement**: A framework for constructing task definitions at runtime, assigning the task to the most available fleet/robot and managing the task’s execution.
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

(1) Task Initialization

Task Requests → Task Status

Task Dispatcher Node

(2) MultiFleet Task Assignment

Task Bidding

Fleet Adapter

(3) Task Allocation
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

(1) Task Initialization

Task Requests → Task Status

GUI

Operators

(2) MultiFleet Task Assignment

Task Bidding

Fleet Adapter

(3) Task Allocation

Fleet Adapter

Fleet Adapter
Overview

Fleet Adapters Bid for Tasks
Going once, going twice, sold!

1. Task Initialization
2. MultiFleet Task Assignment
3. Task Allocation
Fleet Adapters Bid for Tasks
Going once, going twice, sold!

(1) Task Initialization

(2) MultiFleet Task Assignment

(3) Task Allocation

Task Dispatcher Node

Task Bidding

Fleet Adapter

Fleet Adapter

Fleet Adapter

Task Requests

Task Status

GUI

Task Dispatcher Node

Task Bidding

Fleet Adapters

Task Dispatcher Node

Task Bidding

Fleet Adapters

Queue / Execute task

Publish a call for proposals

Configurable criteria to award task to the “best” fleet

Each FA will return its proposal for accommodating the task

/rmf_task/bid_proposal

/rmf_task/dispatch_request
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

1. Task Initialization
2. MultiFleet Task Assignment
3. Task Allocation

Overview
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

(1) Task Initialization

(2) MultiFleet Task Assignment

(3) Task Allocation
Fleet Adapters Bid for Tasks
Going once, going twice, sold!

(1) Task Initialization

(2) MultiFleet Task Assignment

(3) Task Allocation

Bid cost

Winner!
inputs: (initial state prediction, robot description)  
output: predicted state after task completion

Provided to a multi-agent task planner to search for a "minimum-cost" assignment of tasks to robots

⚠️ The current implementation assumes each task is assigned to one mobile robot and that individual tasks do not depend on each other.

Future versions of RMF will support multi-agent tasks and constraints between tasks.

Human operators or external systems can request that a phase is skipped or repeated. This is helpful if a phase did not go as intended.
Simple, premade

```
{
"category": "delivery",
"description": {
  "pickup": {
    "place": "L2_pharmacy",
    "payload": [{"sku": "48052", "quantity": 2}, {"sku": "37981", "quantity": 1}]
  },
  "dropoff": {
    "place": "L3_ward32_bed4"
  }
}
```

Common tasks can be given simple premade description schemas with a minimal set of parameters to fill in

Each category is associated with its own description schema that can be interpreted by task planners and executors.
Predictive Models for Composed Tasks

Each leaf-node activities need to be either:

- an activity primitive with built-in support implemented in RMF
- a custom activity that the system integrator has plugged in an interpreter for

A predictive model for the whole task is assembled by chaining together the predictive models of the leaf-node activities
Task Descriptions

Task Acceptance Criteria

Not all robots can perform all tasks...

Each different robot platform is integrated with its own RMF Adapter

- The adapter knows the **description** schema of each **category** that the platform can support
- The adapter knows robot-specific parameters, e.g. battery, speed, navigation graph, payload capacity, and other capabilities like cleaning, scanning, greeting

If none of an adapter's robots can perform a task because of incompatibility, the task is rejected.
**Allocation of tasks**

A* based search algorithm to determine the right sequence in which tasks should be executed within the fleet to minimize overall time.

```
rmf_task::TaskPlanner
```

https://github.com/open-rmf/rmf_task
Execution is broken down into a hierarchy of "activities"

- **delivery**: medicine from pharmacy to ward31
- **pickup**: medicine from pharmacy
- **go_to_place**: pharmacy
- **move_to**: atrium door-entry wait point
- **pass_through_door**: atrium door
  - **open_door**: atrium door
  - **move_to**: atrium door-exit wait point
  - **close_door**: atrium door
- **move_to**: pharmacy door-entry wait point
- ...
Each hierarchy is contained within a "Phase"

**delivery task:** medicine from pharmacy to ward31

**pickup:** medicine from pharmacy
- **go_to_place:** pharmacy
  - **move_to:** atrium door-entry wait point
  - **pass_through_door:** atrium door
    - **open_door:** atrium door
    - **move_to:** atrium door-exit wait point
    - **close_door:** atrium door
    - **move_to:** pharmacy door-entry wait point
  - ...

**dropoff:** medicine to ward31
- **go_to_place:** ward31
  - **move_to:** pharmacy door-exit wait point
  - **pass_through_door:** pharmacy door
    - **open_door:** pharmacy door
    - **move_to:** pharmacy door-entry wait point
    - **close_door:** pharmacy door
    - **move_to:** service lift Lobby A2
  - ...

**Phases are always sequential**

**dropoff:** medicine to pharmacy