How custom tasks are defined, assigned, and executed in Open-RMF
Overview of this talk

Operational challenges with multiple fleets

Multi-fleet task allocation framework

Modelling and Assignment of Composable Tasks

Execution of Tasks
The need for robotic interoperability is on the rise
Managing daily operations of multiple robot fleets is challenging.

- **Deliver**
- **Patrol**
- **Greet**
- **Clean**

Capabilities of fleets may be specialized or shared.

Dashboard for each fleet:
- I need supplies to be delivered
- I can deliver but I’m busy now
- I cannot fulfill deliveries
- I can deliver but I have low battery

Operators in the facility:
**Functional requirements for a task management framework**

- Flexible
- Predictive modelling
- Platform agnostic

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

- Can I coordinate robots from two different fleets to perform a task?
- Can I reuse an existing robot to execute a more complex workflow?
- Do I need to decide which robot performs which task?
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!

Overview

Pickup X items at A and drop Y & Z items at B and C resp.
Fleet Adapters Bid for Tasks
Going once, going twice, sold!

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

Task Dispatcher Node

Task Requests → Task Status

GUI

Task Request
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

1. Task Initialization

2. MultiFleet Task Assignment

3. Task Allocation

Overview
Overview

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

(1) Task Initialization
Task Requests → Task Status

(2) MultiFleet Task Assignment
Task Bidding

(3) Task Allocation
Fleet Adapter

Task Dispatcher Node

Fleet Adapters

GUI

/rmf_task/bid_notice
/rmf_task/bid_proposal
/rmf_task/dispatch_request
Queue / Execute task

Publish a call for proposals
Each FA will return its proposal for accommodating the task
Configurable criteria to award task to the "best" fleet

Task to the "best" fleet
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

(1) Task Initialization

(2) MultiFleet Task Assignment

(3) Task Allocation

Each fleet adapter is capable of:

- Validating whether its fleet can perform the task
- Solving a Multi-robot Task Allocation problem
Fleet Adapters Bid for Tasks

Going once, going twice, sold!

(1) Task Initialization

(2) MultiFleet Task Assignment

(3) Task Allocation

Bid cost

Winner!
What is a "Task"?

**Task Description**
serializable data structure that can be interpreted into...

**Predictive Model**
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

**Runtime**
generates a sequence of task "phases"

**Task Phase**
monitors state of robot and infrastructure to issue commands (e.g. navigate to location, open door, summon elevator) to fulfill an objective of the task

⚠ Current Scope ⚠
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.

https://github.com/open-rmf/rmf_task
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.

Simple, premade

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

Custom, composed

```json
{
  "category": "compose",
  "description": {
    "detail": "Drop off medication and then greet the patient",
    "phases": [
      {
        "activity": {
          "category": "pickup",
          "description": {
            "place": "L2_pharmacy",
            "items": [{"sku": "48052", "quantity": 2}]
          }
        },
        "activity": {
          "category": "dropoff",
          "description": {
            "place": "L3_ward10_bed4",
            "items": [{"sku": "48052", "quantity": 2}]
          }
        },
        "on_cancel": {
          "category": "dropoff",
          "description": {
            "place": "L2_pharmacy"
          }
        }
      },
      {
        "activity": {
          "category": "greet",
          "description": {
            "place": "L3_ward10_bed4",
            "language": "Hokkien"
          }
        }
      }
    ]
  }
}
```

More detailed instructions:
https://osrf.github.io/ros2multirobotbook/task_new.html
Task Descriptions

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

https://github.com/open-rmf/rmf_task
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

Given $M$ tasks of varying start times and descriptions, and $N_i$ robots in $F$ fleets,

- Distribute $M$ tasks across $F$ fleets such that
  - Robots are only assigned tasks they are capable of performing
  - Robots have sufficient resources (e.g. battery) to perform all assigned tasks
  - Overall optimality of assignments

**Assumptions**

- Each task is executed by only one robot (no collaboration)
- A robot will perform a task only after fully completing the previous task
- Each robot is assigned a charger

[https://github.com/open-rmf/rmf_task](https://github.com/open-rmf/rmf_task)
Allocation of tasks

Given M tasks of varying start times and descriptions, and N_i robots in F fleets,

- Distribute M tasks across F fleets such that
  - Robots are only assigned tasks they are capable of performing
  - Robots have sufficient resources (e.g. battery) to perform all assigned tasks
  - Overall optimality of assignments
  - Assumptions
    - Each task is executed by only one robot (no collaboration)
    - A robot will perform a task only after fully completing the previous task
    - Each robot is assigned a charger

rmf_task::TaskPlanner

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

RMF Task Allocation Planner- Other features

● Priority Assignment
  ○ Add a Priority field to task request
  ○ During node expansion, check if new node assignments are valid
    ■ Valid = high priority tasks are assigned prior to low priority ones
  ○ If invalid, \( f(n) = g(n) + h(n) \times \text{penalty} \)

● Finishing Task
  ○ Automatically include a task that the robot has to perform at the end of its assignments
  ○ Park, ChargeBattery, etc

● Fleet adapters automatically replan task assignments when a task is cancelled

● Battery charging tasks are automatically inserted when needed
Execution

Activity Hierarchy

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 activity is publishing requests (e.g. navigation requests, open/close door requests) and subscribing to state updates to manage the progress of the task.

⚠ Currently activities at the same hierarchy level are treated as sequential, but future versions of RMF will support parallel activities, conditional execution, branching, and activity loops

## Execution

### Phase Management

Each hierarchy is contained within a "Phase"

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

<table>
<thead>
<tr>
<th>pickup: medicine from pharmacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>go_to_place: pharmacy</td>
</tr>
<tr>
<td>move_to: atrium door-entry wait point</td>
</tr>
<tr>
<td>pass_through_door: atrium door</td>
</tr>
<tr>
<td>open_door: atrium door</td>
</tr>
<tr>
<td>move_to: atrium door-exit wait point</td>
</tr>
<tr>
<td>close_door: atrium door</td>
</tr>
<tr>
<td>move_to: pharmacy door-entry wait point</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

**on success**

<table>
<thead>
<tr>
<th>dropoff: medicine to ward31</th>
</tr>
</thead>
<tbody>
<tr>
<td>go_to_place: ward31</td>
</tr>
<tr>
<td>move_to: pharmacy door-exit wait point</td>
</tr>
<tr>
<td>pass_through_door: pharmacy door</td>
</tr>
<tr>
<td>open_door: pharmacy door</td>
</tr>
<tr>
<td>move_to: pharmacy door-entry wait point</td>
</tr>
<tr>
<td>close_door: pharmacy door</td>
</tr>
<tr>
<td>move_to: service lift Lobby A2</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

**Phases are always sequential**

*this limitation will be eased in future versions*

A web-based API can be used to skip, restart, or retry phases

Each phase can be assigned a cancellation sequence that it will follow if the task is cancelled while the phase is active

<table>
<thead>
<tr>
<th>on cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropoff: medicine to pharmacy</td>
</tr>
</tbody>
</table>

[GitHub Repository](https://github.com/open-rmf/rmf_ros2/tree/main/rmf_fleet_adapter/schemas)
Future work

- Generalized task compositions
- GUI for designing workflows
- Backend for scheduling recurring tasks
- Endpoints for modifying the schedule

[https://github.com/open-rmf/rmf/discussions/169](https://github.com/open-rmf/rmf/discussions/169)