Executing Robot Task Models in Dynamic Environments

Kai Adam, Arvid Butting, Oliver Kautz, Bernhard Rumpe, Andreas Wortmann
Software Engineering
RWTH Aachen

http://www.se-rwth.de/
Motivation

- Research project iserveU \(^1\)
- Support hospital caregiver tasks:
  - delivering items
  - following persons
  - guiding persons
- Easily integrate new task types
- Enable technically unaware staff to operate robot
- Support arbitrary robots

\(^1\) http://www.se-rwth.de/materials/iserveu/
Research Statement

- Service robotics is a challenging domain:
  - involves many different roles
  - dynamic environment
  - heterogeneous robot tasks

- Use models:
  - reduce complexity
  - increase reuse
  - enable platform-independence
    - reduce costs

- Execution of tasks

How to model this? How do the models interplay?
<table>
<thead>
<tr>
<th></th>
<th>Introduction &amp; Research Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>iserveU DSLs &amp; Software Architecture</td>
</tr>
<tr>
<td>3.</td>
<td>Executing Task Models</td>
</tr>
<tr>
<td>4.</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Overview

Robot

- send parametrized tasks

Editor

- uses

Staff

- Editor

Domain Expert

- Goals

- Tasks

- comprises

- references

Framework Developer

- Architecture

- Robot

- Robot GPL Interface

Robotic Expert

- Robot

- uses

- Robot GPL Interface

ROS
Example: iserveU Models

Define capabilities as actions and properties

- «entity»
  - TransportRobot

- «class diagram»
  - LogisticsDomain

- «entity»
  - RoomsWorld

- «goal»
  - LoadedAt

- «task»
  - DeliverItem

- «goal»
  - UnloadedAt

Properties of the domain model

- uses
- has
- requires

Boolean conditions over entity properties

Sequences of goals
Example: Domain

- **WayPoint**
  - adjacent

- **Room**
  - String name

- **Item**
  - String name

- **LogisticsDomain**
  - uses
  - requires

- **RoomsWorld**
  - has

- **DeliverItem**
  - has

- **LoadedAt**
  - has

- **UnloadedAt**
  - has

- **TransportRobot**
  - uses
  - requires

- **LogisticsDomain**
Example: Entities

```java
01  domain LogisticsDomain;
02  world RoomsWorld rw;

04  robot TransportRobot {
05     property Waypoint robotLoc();
06     property Boolean hasLoaded(Item item);
07
08     action pickUp(Item item, Room room) {
09         pre:  robotLoc() == room && rw.itemLoc(item, room);
10         post: hasLoaded(item) && !rw.itemLoc(item, room);
11     }
12
13     action move(Waypoint from, Waypoint to) { /* .. */ }
14 }
```

This refers to other entity and domain.
Example: Goals

```
01 robot TransportRobot rob;
02 goal LoadedAt(Room room, Item item) {
03    (rob.robotLoc() == room) && (rob.hasLoaded(item))
04 }
```

```
Goal

Boolean property
references entity properties
```
Example: Tasks

```
01 domain LogisticsDomain;
02 task DeliverItem(Room src, Item i, Room dst) {
03     LoadedAt(src, i);
04     UnloadedAt(dst, i);
05 }
```

How are tasks executed?

ordered list of goals
Outline

1. Introduction & Research Statement

2. iserveU DSLs & Software Architecture

3. Executing Task Models

4. Conclusion
Task Execution

- Dynamic domain: deterministic, fixed plans will fail
- Use PDDL\(^1\) planner to execute tasks:
  - reach goals in order (i.e., fulfill condition)
  - fulfill goal through action execution
- Action:
  - can be executed if precondition is fulfilled
  - after fulfilling: effect manipulates environment
- Planning data: CD types, entity model, goal models
- Derive a plan through model transformations at run-time

---

Process

RTA – Model Realization Base Class instances

Entity models

CD models

Goal models

Entity parser

Entity ASTs

Intra-language transformations

Entity ASTs

CD parser

CD ASTs

Inter-language transformations

Partial PDDL models

Goal parser

Goal ASTs

Planner

PDDL models

PDDL planner

PDDL plan

Plans

provided by architecture at run time e.g., Metric-FF

Arthur Butting
Software Engineering
RWTH Aachen

Slide 13
Effects on Architecture

- Implementation of ActionExecutor generated
  - action types depend on entity models
- Implementation of StateProvider generated
  - properties depend on entity models & domain types
Conclusion

- Controller orchestrates robot task execution
- Planner calculates how to achieve goals
- **Action executor** delegates to middleware

- **Evaluated** in a German hospital

- Operate in dynamic environment

- Separation of concerns

- Good extensibility & reusability
Thank you for your attention