Dynamic Allocation of Service Robot Resources to an Order Picking Task Considering Functional and Non-Functional Properties

Timo Blender, Christian Schlegel
The Fundamental Problem: Flexible Machines and Varying Tasks

How can service robots decide based on its available software capabilities and variability whether and how to fulfill varying task requirements of end users in dynamic environments?
The Context: Mastering the Software Engineering Challenge in Robotics

➢ Business ecosystem for robotics based on separation of roles and on composition

Step-change towards a European ecosystem for open and sustainable industry-grade software development for robotics

➢ https://robmosys.eu/

Reference: https://robmosys.eu/application/

Reference: https://robmosys.eu/approach/
The Ideal: Synthesis in a Robotics Software Ecosystem (RSE)

➢ Automatic composition and configuration of available (software) building blocks from a marketplace to a customized, flexible robotic application based on a formal requirement specification of an user

- Enables deployment of requirement-related robotic applications
- Significant effort reduction of the associated development and configuration process

The figure illustrates the process of designing and running robotic applications in an ecosystem. It includes design-time activities such as defining requirements, selecting from a repository, managing variability, and composing models/variability. Run-time activities involve continuous variability management and exploiting variability at runtime for adequate task achievements.
The Challenge: Managing Variability

Example:

Possible nowadays
- What to do to deliver a cup of coffee?
- How to deliver a cup of coffee as hot as possible?
- How to deliver a cup of coffee hot enough?
- How to reduce the risk of spilling a hot cup of coffee?

Research topics addressed in this work
- How to save resources as good as possible?

Step Change

Service robot systems composed out of different building blocks with different configuration variants and resulting different properties, influenced additionally by dynamic environments should configure themselves according to variable user requirements.

Generalization:

Problems/Domains/Requirements
- Several variability entities may exist in each dimension

Environments

Linking different dimensions of variability

Variability Management

Solutions/Building Blocks/Properties

Modeling and resolving dependency relationships between variability entities and their variability spaces of different building blocks.
The Approach: Dependency Variability Graphs (DVGs)

1) The DVG method enables to model a VPS

2) The DVG method enables to derive a VSS

Deriving the VSS can be either a Constraint Satisfaction Problem or a Constraint Satisfaction Optimization Problem.

The VPS may be higher dimensional!
DVGs: Basics

- DVGs are a novel combination of different methods and structures:
  - Dependency graphs
  - Value trees
  - Decision trees
  - Constraint graphs
  - Function composition

  This novel combination results in a generalized, more powerful expressiveness required to represent appropriate problem spaces in the context of a RSE

- DVGs are tailored to the boundary conditions of a RSE (composition, separation of roles)
DVGs: Modeling Patterns and Systems

Based on the basics different modeling patterns were derived that serve as templates to compose DVG systems.
Implementation: Variability Management in a RSE by MDSD

➢ Realization of Variability Management in a RSE by
  - building block modeling
  - DVG system modeling
  - Generating solver code for DVG system resolution
The Use Case of the Paper: Overview

➢ Order Picking task with variable requirements
  - Minimizing time or
  - Staying below a maximum time limit

➢ Different robot resources available with different capabilities and properties
  - Can grasp objects or not, maximum velocity, current position, ...

➢ Different environment conditions
  - Layout of the warehouse

Which combination of available robots in which individual configuration is suitable for the current environment to fulfill the specified task requirements?

We apply our general method for variability management to model and solve this problem!
The Use Case of the Paper: Building Block and DVG System Model

Utilizes composition and separation of roles!
Experiment of the Use Case: Minimizing Time (3 Robot Candidates)

Table 1: Properties of our service robot candidates.

<table>
<thead>
<tr>
<th>Robot</th>
<th>DockToStation</th>
<th>PickAndPlace</th>
<th>Max Vel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robotino</td>
<td>true</td>
<td>false</td>
<td>1200.0</td>
</tr>
<tr>
<td>Larry</td>
<td>false</td>
<td>true</td>
<td>1000.0</td>
</tr>
<tr>
<td>Macy</td>
<td>true</td>
<td>true</td>
<td>1000.0</td>
</tr>
</tbody>
</table>

Table 2: The possible task allocations in this example based on the available service robots (table 1).

<table>
<thead>
<tr>
<th>Allocation</th>
<th>$R^S$</th>
<th>$R^M$</th>
<th>$R^N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Macy</td>
</tr>
<tr>
<td>2</td>
<td>Robotino</td>
<td>Larry</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Robotino</td>
<td>Macy</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Macy</td>
<td>Larry</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Constant values in the example

<table>
<thead>
<tr>
<th>Names</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberObjects</td>
<td>FetchStation</td>
</tr>
<tr>
<td>PickingPlace</td>
<td>DeliverStation</td>
</tr>
<tr>
<td>DockToStation</td>
<td>LoadFromStation</td>
</tr>
<tr>
<td>UnloadToStation</td>
<td>UndockFromStation</td>
</tr>
<tr>
<td>DetectObjects</td>
<td>DockToRobot</td>
</tr>
<tr>
<td>Pick</td>
<td>Place</td>
</tr>
</tbody>
</table>

Table 4: Varying start positions for the service robots in the different experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Robotino</th>
<th>Larry</th>
<th>Macy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(-11.0, -14.0)</td>
<td>(3.0, 12.0)</td>
<td>(1.0, 1.0)</td>
</tr>
<tr>
<td>2</td>
<td>(1.0, 1.0)</td>
<td>(-2.0, -8.0)</td>
<td>(-11.0, -14.0)</td>
</tr>
<tr>
<td>3</td>
<td>(-11.0, -14.0)</td>
<td>(-2.0, -8.0)</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

DVG solver determines the corresponding allocation for the current situation fulfilling the specified requirements.