Towards a Robot Perception Specification Language

Nico Hochgeschwender,¹,² Sven Schneider,¹ Holger Voos,² and Gerhard K. Kraetzschmar,¹

¹Bonn-Rhein-Sieg University, Sankt Augustin, Germany
nico.hochgeschwender@h-brs.de
²University of Luxembourg, Luxembourg

November 8, 2013
General Purpose Service Robots

In order to successfully achieve its task every robot need to perceive its environment.
Robot Perception Architectures (RPAs)

Sensors
- Laser Range Finder
- Webcam
- Kinect 3d Camera
- ...

Robot Perception Architecture

Result
- Person Id
- Object Pose
- Robot Pose
- ...
- ...

Motivation

Problem Statement

RPSL: Robot Perception Specification Language

Conclusion and Future Work
Robot Perception Architectures (RPAs)

- **Sensors:** several different modalities.

![Diagram showing Robot Perception Architecture with sensors connected to various results including Person Id, Object Pose, Robot Pose, and additional unspecified results.](image)
Robot Perception Architectures (RPAs)

- **Sensors:** several different modalities.
- **Results:** dependent on the task. Very high diversity.
Robot Perception Architectures (RPAs)

- **Sensors**: several different modalities.
- **Results**: are dependent on the task. Very high diversity.
- **RPAs**: composition of components providing different functionalities found in libraries such as OpenCV and PCL.
Example: RPA for People Detection

Design Time Challenges

Implicit Design Decisions by Domain Experts

Concerning the robot, the task, and the environment. Some exemplary design decisions include:

1. The **sensor configuration** of a particular robot.
2. The general **composition** of an RPA (e.g., selection, configuration and organization of components).
3. The **configuration** of a specific **composition** of an RPA for solving a task-relevant perception problem.
Run Time Challenges

Design Time vs. Run Time

A **systematic** reaction to changing conditions concerning robot capabilities, task requirements, and environment features is very challenging. It is not possible

1. To adapt parameters of computational components (e.g., the \( \sigma \) value of a Gaussian filter) during run time,
2. To execute complete processing chains in a demand-driven manner, and
3. To modify and reconfigure robot perception processing chains during run time.
Objective and Approach

Enabling the design and development of task-oriented RPAs...

Approach: Explicit Representations

We will adopt model-based techniques to develop representations (models) for

1. Sensors and their properties (√)
2. Components and their integrated algorithms (√)
3. Task-relevant information

Approach: (Re-) Configuration Methods

We need to develop schemes working on the representations (models) to enable dynamic and flexible control and data flow within RPAs.
Objective and Approach

Enabling the design and development of task-oriented RPAs...

Approach: Explicit Representations

We will adopt model-based techniques to develop representations (models) for

1. Sensors and their properties (√)
2. Components and their integrated algorithms (√)
3. Task-relevant information

Approach: (Re-) Configuration Methods

We need to develop schemes working on the representations (models) to enable dynamic and flexible control and data flow within RPAs.
Objective and Approach

Enabling the design and development of task-oriented RPAs...

Approach: Explicit Representations

We will adopt model-based techniques to develop representations (models) for

1. Sensors and their properties (✓)
2. Components and their integrated algorithms (✓)
3. Task-relevant information

Approach: (Re-) Configuration Methods

We need to develop schemes working on the representations (models) to enable dynamic and flexible control and data flow within RPAs.
Objective and Approach

Enabling the design and development of task-oriented RPAs...

Approach: Explicit Representations
We will adopt model-based techniques to develop representations (models) for
1. Sensors and their properties (✓)
2. Components and their integrated algorithms (✓)
3. Task-relevant information

Approach: (Re-) Configuration Methods
We need to develop schemes working on the representations (models) to enable dynamic and flexible control and data flow within RPAs.
Objective and Approach

Enabling the design and development of task-oriented RPAs...

Approach: Explicit Representations

We will adopt model-based techniques to develop representations (models) for

1. Sensors and their properties (√)
2. Components and their integrated algorithms (√)
3. Task-relevant information

Approach: (Re-)Configuration Methods

We need to develop schemes working on the representations (models) to enable dynamic and flexible control and data flow within RPAs.
Robot Perception Specification Language (RPSL)

Domain Analysis (Assessment of Existing Applications)

- Levels of abstraction (from raw sensor data to symbols).
- Views on the output and “backtracking” requirements.

...
Robot Perception Specification Language (RPSL)

Requirements

RPSL is aimed to be a specification language and should be independent of

- the type of sensor data processed by the RPA, and
- the type of functional components which are assembled to make up an RPA.
Robot Perception Specification Language (RPSL)

**Domains**

RPSL is composed of the following sub-domains each represented in ECore (plus constraints).

1. Object domain
2. Spatial domain
3. Timing domain
4. Dependency domain
5. And more?
6. Composition domain

**Implementation**

- Current version of RPSL realized as an external DSL with Eclipse/Xtext and Eclipse OCL.
**RPSL: Object Domain**

**Conceptual Spaces (CS)**

CS [9] as a uniform data model in RPSL.

- A **CS** consists of several quality dimensions.
- A **Concept** in a CS is a convex region in that space.
- A **Point** in a CS represents concrete instances of a concept.

**Example: Concept Definition**

```plaintext
myConcepts: Namespace {
    myBox: Concept {
        use_domain Size
        use_domain RGB
        p: Polytope {
            // ...
            Point(RGB.Red, 0)
            Point(RGB.Green, 0)
            Point(RGB.Blue, 100)
            Point(RGB.Blue, 140)
        }
    }
}
```
Example: Prototype Definition

```plaintext
use Namespace myConcepts
darkBlueBox: Prototype {
    use_concept myConcepts.myBox
    v: Values {
        // ...
        Point(myBox.RGB.Blue, 139)
    }
}
```
“Only” object information is not sufficient.
Spatial surrounding is important as well (e.g., for manipulation)
Examples are (assuming an egocentric view) statement such as “right of” and “next to”.
We investigate currently which spatial model we will include in RPSL such as RCC-5, RCC-8 and CYS$_t$. 
RPSL: Timing Domain

- Notion of a deadline to encode a particular point in time by which the specified information needs to be available.
- Future Work: Policies to cope with missed deadlines etc.

Example: Timing Specification

```plaintext
spec: Specification {
  d: Data {
    get Quantity from myBox
    where myBox.Size.Width >= 20mm
    and myBox.Size.Length > 100mm
    ensure Deadline(100ms)
  }
}
```
We need to model dependencies among specifications (some information is required before/after some other).

We model these dependencies with a Directed Acyclic Graph (DAG).
RPSL: Dependency Domain

Example: Dependency

```plaintext
a: Specification {
    d: Data {
        get Quantity ...
    }
}

b: Specification {
    d: Data {
        get Pose ...
    }
}

c: Specification {
    dg: DependencyGraph {
        a before b
        ...
    }
}
```
RPSL: Composition Domain

- Composition of the previous domains (whereas some are optional).

Example: Composition with Similarity Measure

```plaintext
class Specification {
    class Data {
        get Pose from darkBlueBox
        where Similarity(EuclideanDistance) == 0
    }
}
```
RPSL is not (yet) finished, but it supported already the assessment of RPAs.

In its current version RPSL is limited to “simple” scenes (e.g., no active perception) due to missing concepts.

We are currently applying RPSL to model existing RPAs.
Thanks and Acknowledgement

Thank You! Questions?

Nico Hochgeschwender received a PhD scholarship from the Graduate Institute of the Bonn-Rhein-Sieg University of Applied Sciences.