Exploring the Design Space of Robot Perception Systems*

*Or, how to create synergies between robotics and software engineering?

Nico Hochgeschwender and Loïc Gammaitoni
Interdisciplinary Center for Security, Reliability, and Trust (SnT)
University of Luxembourg

1st International Workshop on Robotics Software Engineering (RoSE'18)
International Conference on Software Engineering (ICSE 2018)
Gothenburg, Sweden

28 May 2018
Robot Perception Systems (RPS)
Starting Point: Robot Perception Systems (RPS)

To perform purposeful tasks, robots need to extract **knowledge** about the world.

Where is the cavity?  
How big is the cavity?  
Is the cavity big enough to insert the object?
Robot perception systems need to be equipped with a broad set of sophisticated **perception capabilities** interpreting the **sensory data**.
Challenges in Developing Robot Perception Systems

Huge **variability** of robot perception systems caused by time-varying **task**, **environmental** and **platform requirements**.

**Problem Space** (aka. Application Engineering)

- **Task Variations**
  - Pick and Place
  - Peg-In-Hole
- **Environmental Variations**
  - Objects
  - Varying lighting
- **Platform Variations**
  - Sensors
  - Varying computational resources

**Solution Space** (aka. Domain Engineering)
Challenges in Developing Robot Perception Systems

Huge **variability** of robot perception systems caused by time-varying **task**, **environmental** and **platform requirements**.

**Problem Space**
(aka. Application Engineering)

- **Task Variations**
  - Pick and Place
  - Peg-In-Hole
- **Environmental Variations**
  - Objects
  - Varying lighting
- **Platform Variations**
  - Sensors
  - Varying computational resources

**Solution Space**
(aka. Domain Engineering)
Specifying Robot Perception Systems
Specifying Robot Perception Systems with Domain-specific Languages (DSLs)

To enable domain experts to specify robot perception systems in the presence of functional, architectural and deployment variability we proposed two DSLs [1][2][3], namely the **Robot Perception Specification Language (RPSL)** and the **Deployment Specification Language (DepSL)***

*Both RPSL and DepSL are implemented as Ruby-based, internal and textual DSLs.

---


Case Study: RPS for a Pick-and-Place Application

How to specify the constituents of robot perception systems?

Specifying the **functional variability** with RPSL:

Perception capabilities expressed as high-level features*, organized in a tree-like structure with integrity constraints (e.g. dependencies) and aggregation and specialization relations.

```rpsl
rpsl.feature_model do
  name "PickAndPlace"
  add_feature "Application", :is_root
  add_feature "ServiceArea", :is_mandatory, :child_of => "Application"
  add_feature "ObjDetection", :child_of => "Application",
    :requires => "ServiceArea"
  ...
end
```

* not features in the sense of image features like edges, points or corners
Case Study: RPS for a Pick-and-Place Application

How to specify the constituents of robot perception systems?

Specifying the **architectural variability** with RPSL:

Components as building blocks of robot perception systems, namely a) sensor components, b) processing components and c) perception graphs (composition of components).

```ruby
rpsl.sensor_component do
  name "Kinect"
  add_port :out, "outCloud", "xyzRGB"
  ...
end

rpsl.processing_component do
  name "RansacPlaneDetect"
  ...
end
```
Exploring Robot Perception Systems
Although dimensions are specified in an individual manner at some point we need to combine them. Combining the functional, architectural and deployment dimension results in a design space. The design space constrains the set of all possible implementations (design alternatives) with respect to each modeled dimension [4].

Combining the functional, architectural and deployment dimension results in a design space. The design space constrains the set of all possible implementations (design alternatives) with respect to each modeled dimension [4].

Combining the functional, architectural and deployment dimension results in a design space. The design space constrains the set of all possible implementations (design alternatives) with respect to each modeled dimension [4].

---

Combining the functional, architectural and deployment dimension results in a **design space**. The design space constrains the set of all possible implementations (design alternatives) with respect to each modeled dimension [4].

**Functional Variability**
- (ServiceArea)
- (ObjDetection, ServiceArea)
- (CavityRecognition)
- ...

**Architectural Variability**

**Deployment Variability**

---

Enabling Design Space Exploration of Robot Perception Systems with a Tool
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

Eclipse-based tool based on the language workbench Lightning* which enables to produce and visualize design alternatives

Nico, you can formalize the design space exploration problem in Alloy to benefit from automatic generation of design alternatives.

Alloy is a formal language used to express concepts, inter-concepts relations and constraints used in language definitions. It uses the associated Alloy Analyzer, a SAT-solver based tool, to exhaustively generate instances conforming to the given specifications.

* https://lightning.gforge.uni.lu/
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

Eclipse-based tool based on the language workbench Lightning which enables to produce and visualize design alternatives.

Actually, if you use Lightning, you can even define how those instances can be visualized.

Lightning [5] is an Eclipse-plugin allowing the definition and instantiation, using Alloy, of domain specific modeling languages. It relies on the F-Alloy transformation language to express the transformations between two Alloy models.

Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

RPSL Model

```rpsl
sensor_component do
  name "force_sensor"
end
```

M2M transformation

Fed in

Lightning-based Framework

Design alternative

```
M2M transformation
```

Produces
Generating Design Alternatives with the Tool

Based on the provided RPSL specifications the tool generates design alternatives conforming to our design alternative Alloy meta-model.

Example: Alloy constraint

The components of the resulting perception graph should realize the selected features and should have at least one input/output connection.
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

```
rpsl.sensor_component do
  name "force_sensor"
end
```

Fed in

Lightning-based Framework

Performs exploration

写的

```
             | Fed in
rpsl.sensor_component do
  name "force_sensor"
end
```

Produces

```
             | Produces
rpsl.sensor_component do
  name "force_sensor"
end
```

Performs exploration
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

```
rpsl.sensor_component do
  name "force_sensor"
end
```

Fed in

muştur

Guide exploration

Lightning-based Framework

Produces

Performs exploration

Writes

uni.lu

UNIVERSITÉ DU LUXEMBOURG
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

```rpsl
sensor_component do
  name "force_sensor"
end
```

Lightning-based Framework

Fed in

Performs exploration

ObjRecognition in Configuration.selectedFeatures and some (SensorComponent & xyzRGB.-type.-output ) & SuperGraph.components

Writes
Eclipse-based Tool for Exploring the Design Space of Robot Perception Systems

```rpsl
sensor_component do
  name "force_sensor"
end
```

Fed in

Lightning-based Framework

Design alternative should realize the ObjRecognition feature and should include a sensor providing xyzRGB data.
Video
Conclusion

The tool supports exploring rather complex design spaces in an agile manner:

1. Improved DSE review process through visualization definition
2. Improved efficiency by enabling to guide the DSE
3. Improved effectiveness by relying on formal models

Lessons learned and future work:

1. Right now, DSE is limited to a design space of functional and architectural variability and we plan to include also the platform variability.
2. Using Alloy helped to identify missing constraints also for the Robot Perception Specification Language (RPSL)