Addressing Deployment-time and Run-time Variability in Robotics Software Systems

Luca Gherardi

Bergamo - October 20th, 2014
Approach overview

- **Problem**
  - Huge amount of variability makes hard the configuration of robotics software architectures
  - Run-time variability resolution (aka run-time adaptation) is often hard-coded in the functional components

- **Goal**
  - Model variability and context information
  - To configure and adapt software architecture
  - Based on deployment-time requirements and state of the context
A possible scenario

Static

Static

Dynamic
Configure architecture based on the user's requirements
Feature Model

- Task
  - Detection
  - Manipul.

- Environment
  - Static
  - Dynamic

- Object Material
  - Fragile
  - Firm

- Communic. Channel
  - Channel A
  - Channel B

- Rack Size
  - Small
  - Large

- Task: Detection, Manipul.
- Environment: Static, Dynamic
- Object Material: Fragile, Firm
- Communic. Channel: Channel A, Channel B
- Rack Size: Small, Large
Template System Model

- 3 types of configurations
  - Change component’s implementation
  - Change connections
  - Set properties
Deployment-time Resolution Model

Legend

Run-time (RT) Activity
Deployment-time (DT) Activity
Information flow
Dependencies

Feature Model
DT Resolution Model
Template System Model
RT Resolution Model
RT Configured System Model
RT Feature Selection and Attribute Values
RT Feature Selector
DT Configure System Model
Adaptation Engine
Context Monitor
Context Dependent Meas.
Adaptation Model
Deployment-time Feature Selection

Legend

- **Run-time (RT)** Activity
- **Model**
- **Deployment-time (DT)** Activity
- Information flow
- Dependencies

- **Feature Model**
- **Feature Selection**
- **DT Feature Selection**
- **DT Resolution Model**
- **Template System Model**
- **DT Configure System Model**
- **RT Resolution Engine**
- **RT Configured System Model**
- **RT Feature Selection and Attribute Values**
- **Adaptation Engine**
- **Context Monitor**
- **Context Dependent Meas. Model**
- **Adaptation Model**

Luca Gherardi - Bergamo, October 20th, 2014
Deployment-time Feature Selection

- The feature selection is sent to the Deployment-time Resolution Engine
Deployment-time configured system model
Deployment-time configured system model

Diagram:
- Radio Switch
- Channel B Radio Driver
- Map Based Navigation
- Task Manager
- Laser Scanner
- Local Navigation
- Object Detector
- Arm Planner
- Robot Driver
- Arm Controller

Connections:
- Rack ID
- Path
- Twist
- Odom
- Detected Objects
- Target Object Pose
- Des. Joint Values
- Meas. Joint Values
- Odom Meas. Joint Values
- Odom

Description:
The diagram illustrates the flow of information and components in a system model configured at deployment time. Each component is interconnected with arrows indicating the direction of data flow. The system includes
- Radio Switch for communication
- Channel B Radio Driver
- Map Based Navigation
- Task Manager
- Laser Scanner
- Local Navigation
- Object Detector
- Arm Planner
- Robot Driver
- Arm Controller

The flow of information includes
- Rack ID
- Path
- Twist
- Odom
- Detected Objects
- Target Object Pose
- Odom Meas. Joint Values
- Meas. Joint Values
Run-time Variability Resolution
Luca Gherardi and Nico Hochgeschwender

Adapt run-time architecture based on the state of the context
Deployment-time Feature Selection

- Some of the features can be better chosen at run-time
  - More information about the context is available
- We have to model this information and reason about it
Approach overview

- The run-time variability resolution is a feed-back loop continuously executed
- It is based on the architecture defined in the Deployment-time Configured System Model
Deployment-time Feature Selection

<table>
<thead>
<tr>
<th>Feature</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel A</td>
<td>Latency</td>
<td>Low</td>
</tr>
<tr>
<td>Channel A</td>
<td>Power Consumption</td>
<td>High</td>
</tr>
<tr>
<td>Channel B</td>
<td>Latency</td>
<td>High</td>
</tr>
<tr>
<td>Channel B</td>
<td>Power Consumption</td>
<td>Low</td>
</tr>
</tbody>
</table>
Context Dependent Measurement Model

Legend
- Run-time (RT) Activity
- Deployment-time (DT) Activity
- Information flow
- Dependencies

Feature Model
- Feature Selector
- DT Feature Selection
- DT Resolution Engine
- DT Configure System Model

Template System Model
- RT Feature Selection and Attribute Values

RT Resolution Engine
- RT Configured System Model

Adaptation Engine
- Context Monitor
- Context Dependent Measure Model

RT Resolution Model
- RT Configured System Model

Context Dependent Meas.

Adaptation Model

Luca Gherardi - Bergamo, October 20th, 2014
## Context Dependent Measurement Model

<table>
<thead>
<tr>
<th>CDM</th>
<th>Component</th>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BattLev</td>
<td>Robot Driver</td>
<td>Battery Level</td>
<td>—</td>
</tr>
<tr>
<td>SemLoc</td>
<td>Map Based Nav.</td>
<td>Semantic Location</td>
<td>—</td>
</tr>
<tr>
<td>ObstDist</td>
<td>Laser Scanner</td>
<td>Laser Scan</td>
<td>getClosestObst</td>
</tr>
<tr>
<td>Material</td>
<td>Radio Switch</td>
<td>Object Material</td>
<td>—</td>
</tr>
</tbody>
</table>

In the case study an attribute called `Material` is defined: (a) an input interface associated to the topic that stores the data, and the data type (e.g. by running an estimator). For instance, given the output of a component, but in the information that can be retrieved from the last output or from a series of outputs because in most cases one is not directly interested in the produced by the interface. Additionally, the model describes the limit more precisely, for example according to the battery level and the distance to the closest obstacle. We address this problem by enriching the feature with a constraint that must be respected when the robot is moving in the corridor.

For each attribute, one might need to apply a function to retrieve the distance to the closest obstacle. (e.g. max indicates that the attribute value is the maximum allowed for the non-functional requirement represented by the attribute).

### Table I

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Non-Functional Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laser Scan</td>
<td>VelocityLimit</td>
</tr>
<tr>
<td></td>
<td>Continuous Variability</td>
</tr>
</tbody>
</table>

### Ladder Diagram for the Robotic Platform

- **Laser Scanner**
  - **Sensor Data**
    - **Distance to Obstacle**
      - **Range** (0-100m)
    - **Velocity**
      - **Max Vel**
    - **Material**
      - **Object Material**
- **Robot Driver**
  - **Control Data**
    - **Motion Behavior**
      - **Max Vel**
    - **Material**
      - **Object Material**
- **Battery Level**
  - **Charge Status**
    - **Battery Level**
- **Map Based Nav.**
  - **Location Data**
    - **Semantic Location**
  - **Navigation Status**
    - **Max Vel**
- **Material**
  - **Object Material**

This ladder diagram illustrates the components and their interactions, highlighting the non-functional requirements and their relevance in the robotic platform's context.
- A ros-cpp node is automatically generated based on the Context Dependent Measurements Model
  - Listens for components outputs
  - Produces as output Context Dependent Measurements
Adaptation Model

```java
import caseStudy.featureModel
import caseStudy.contextDepMeasModel as cdm

rule RackSize:
  if( cdm.SemLoc == "RoomA")
    activate_feature(Small)
  else if( cdm.SemLoc == "RoomB")
    activate_feature(Large)
  else deactivate_feature((Large AND Small)

rule Radio:
  if( cdm.batteryLevel > "50" )
    select_feature_from_variants_of(Communic. Channel)
    where_attribute MIN(Latency)
  else select_feature_from_variants_of(Communic. Channel)
    where_attribute MIN(Power Consump)
```
A ros-java node is automatically configured based on the Adaptation Model
- Listens for Context Dependent Measurements
- Produces as output a selection of features
Run-time Resolution Engine

- 3 types of transformations
  - Start/stop components
  - Change connections
  - Set properties
A simulation example

M. W. Mueller and R. D’Andrea. Stability and control of a quadro-copter despite the complete loss of one, two or three propellers.

A simulation example
A simulation example
A simulation example
A simulation example
A simulation example
A simulation example

- Desired Collective Thrust
- Meas. Motor Current
- Est. Altitude

![Graph showing time [sec] vs. desired collective thrust, measured motor current, and estimated altitude over a time range from 366 to 368 seconds.]
Thank you!
Any Questions?