ETH zürich

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

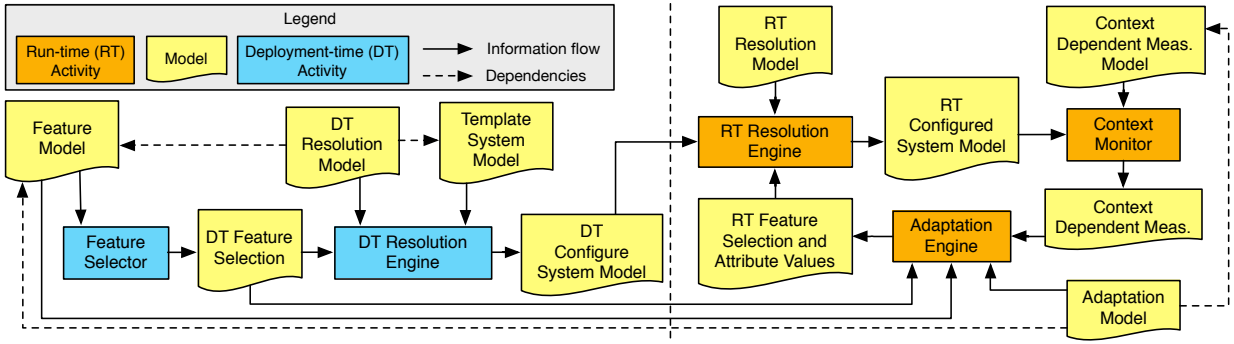
Luca Gherardi

Bergamo - October 20th, 2014

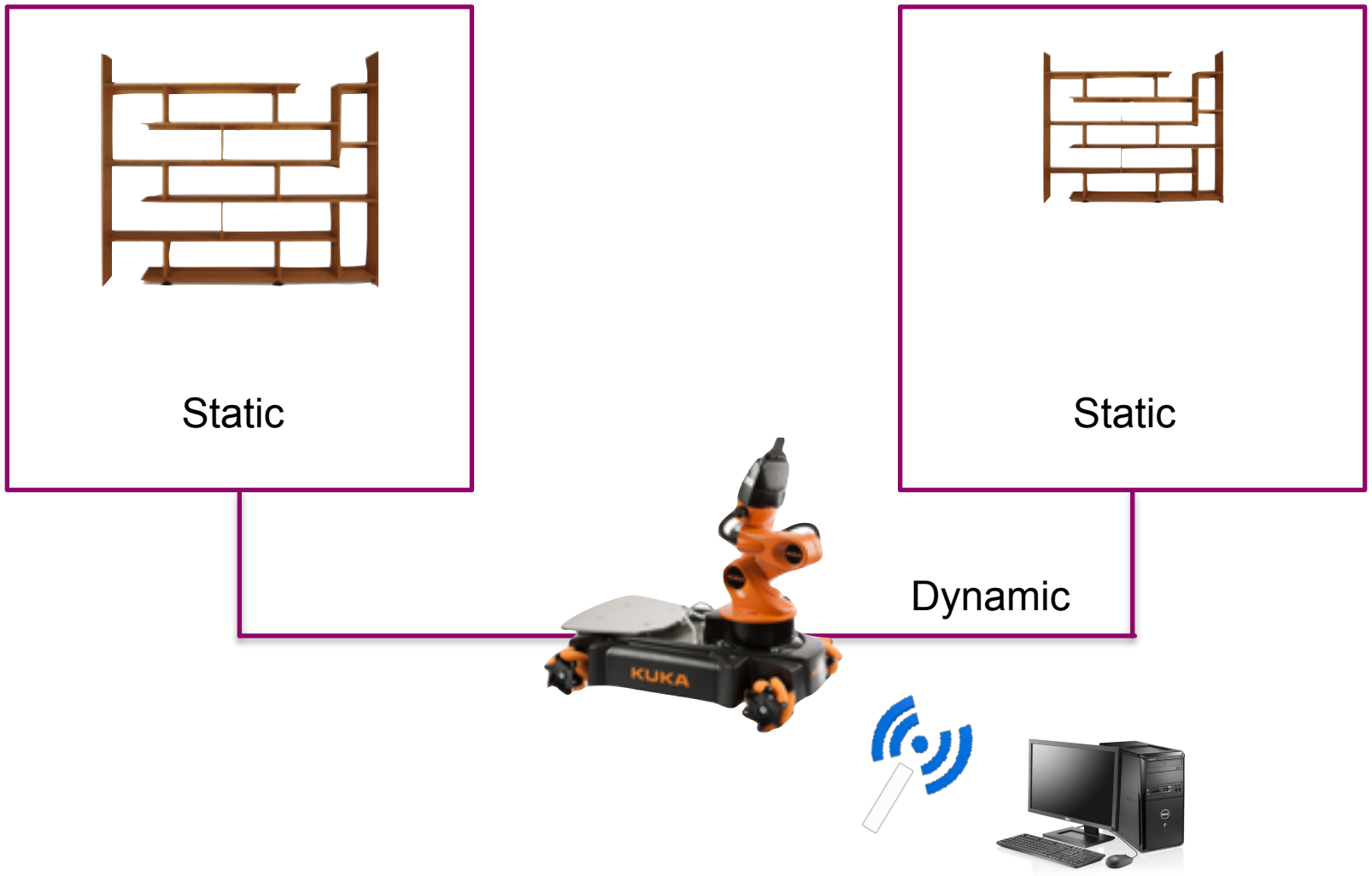
IDSC

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



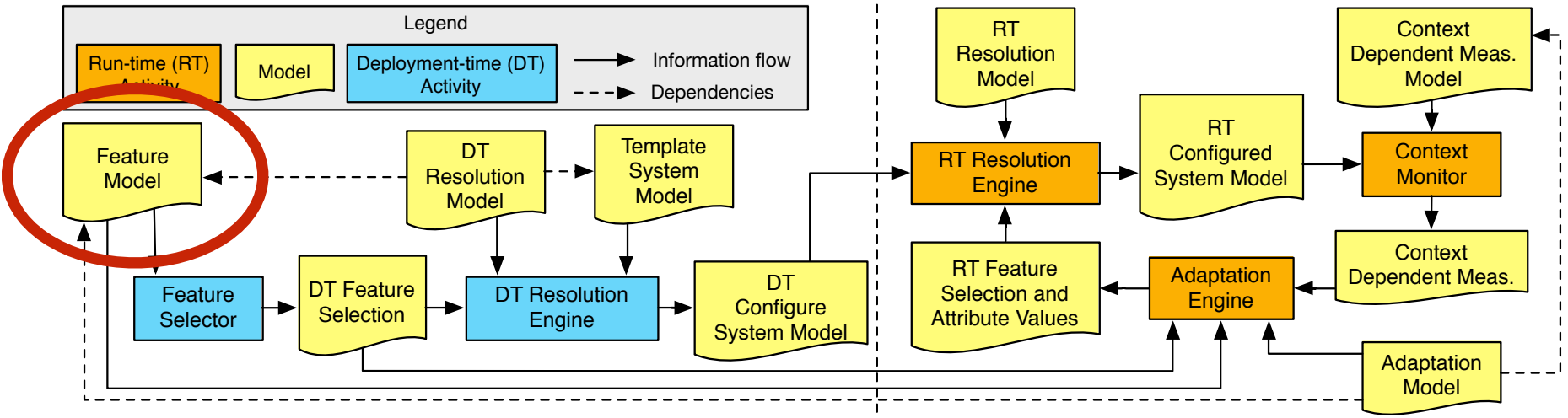
ETH zürich

Deployment-time Variability Resolution

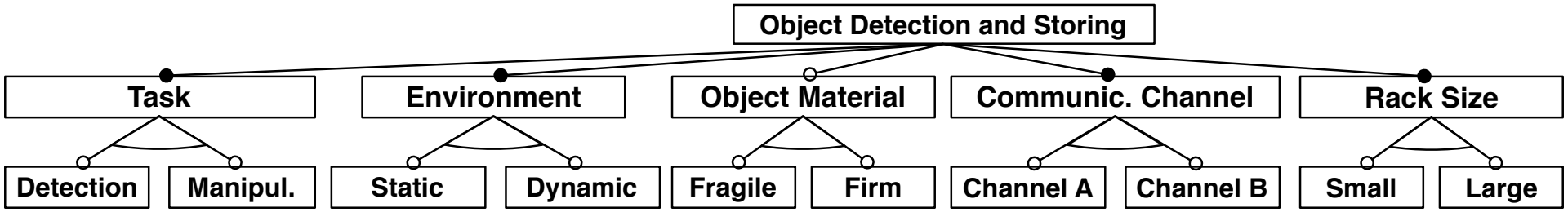
Luca Gherardi and Davide Brugali

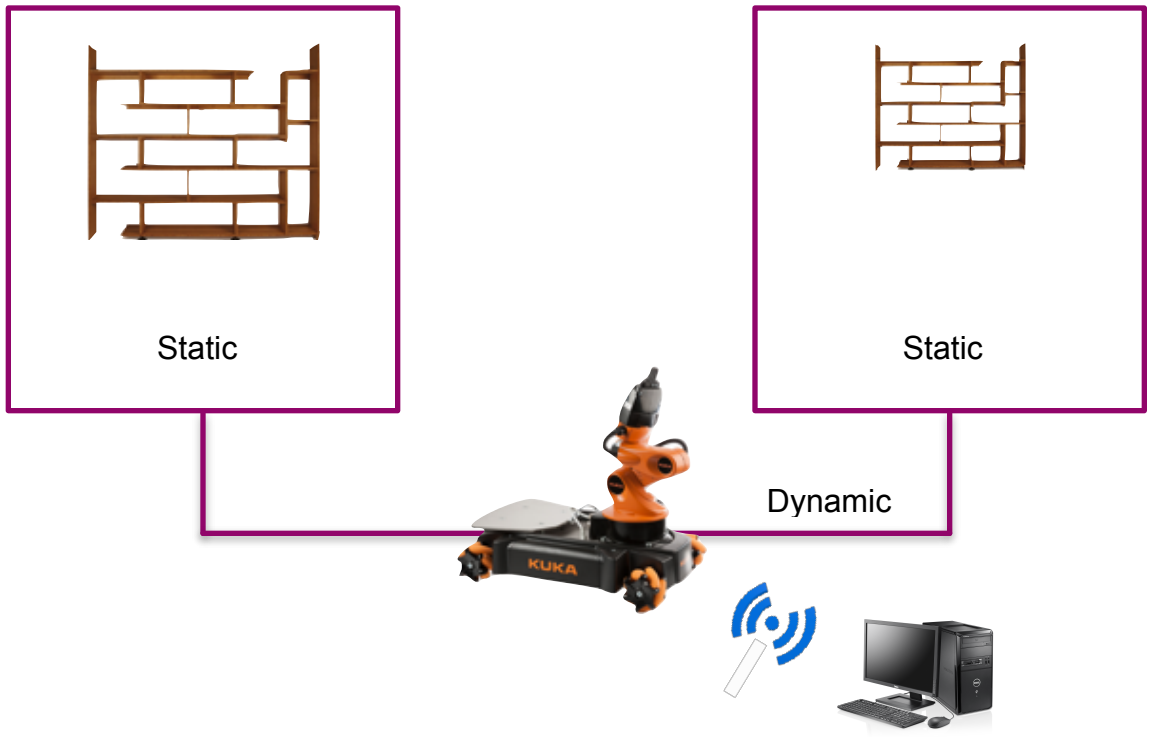
Configure architecture based on the user's requirements

Feature Model

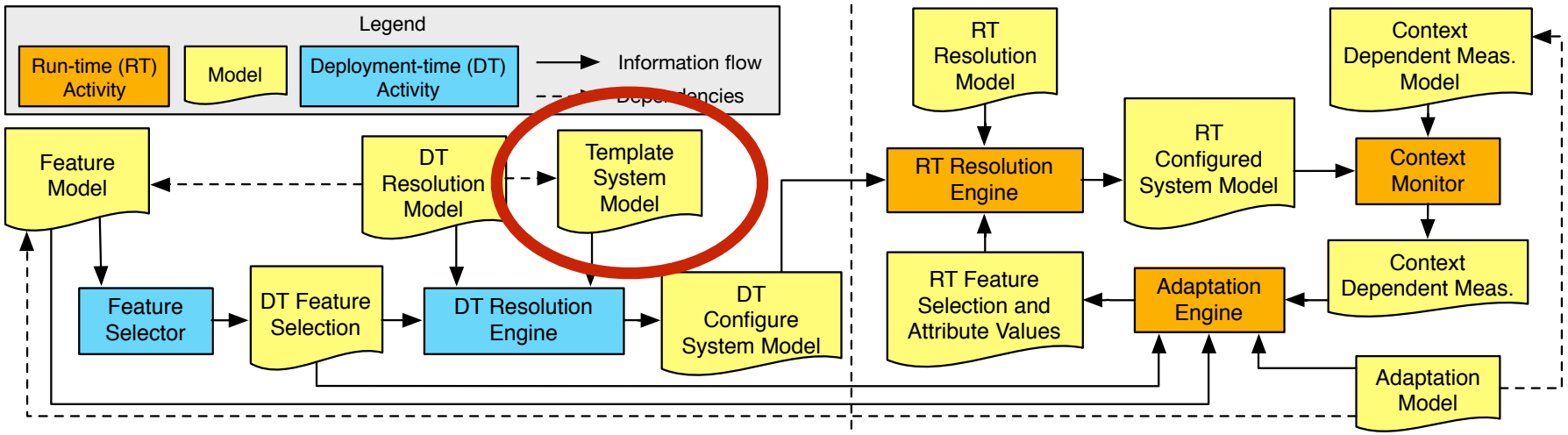


Feature Model

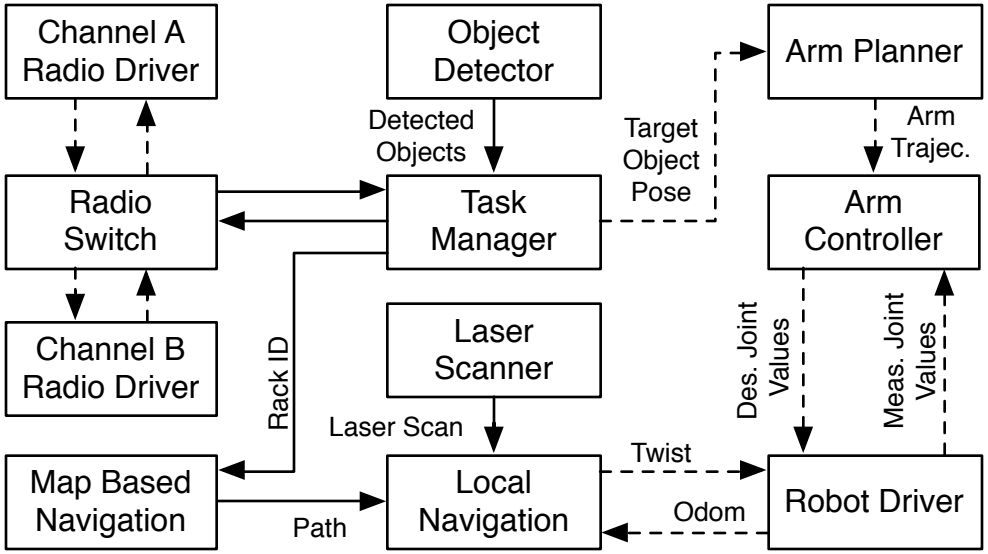




Template System Model

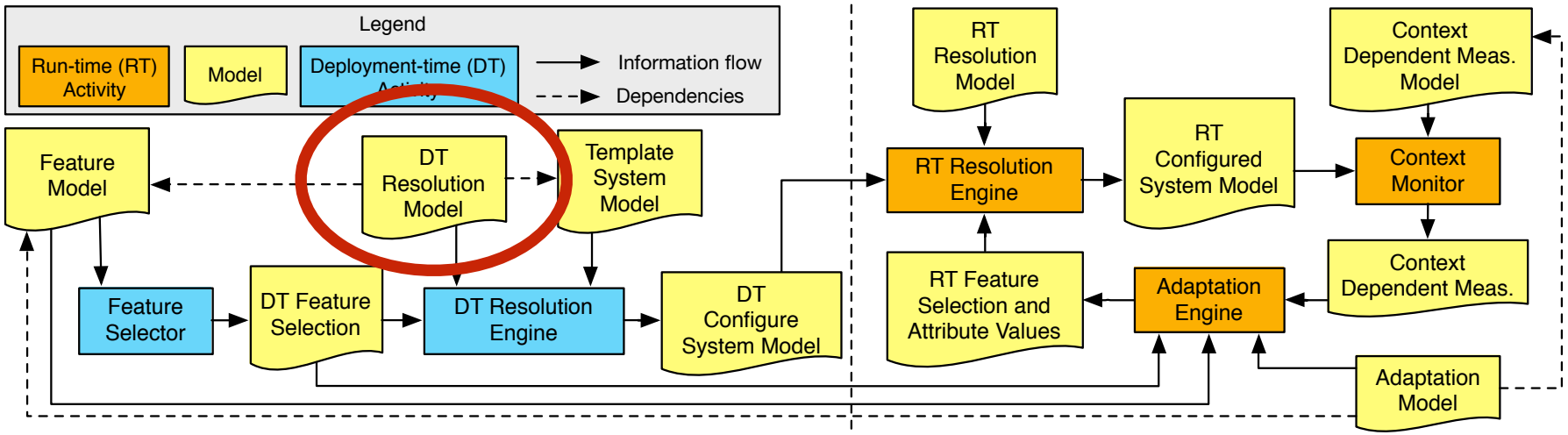


Template System Model

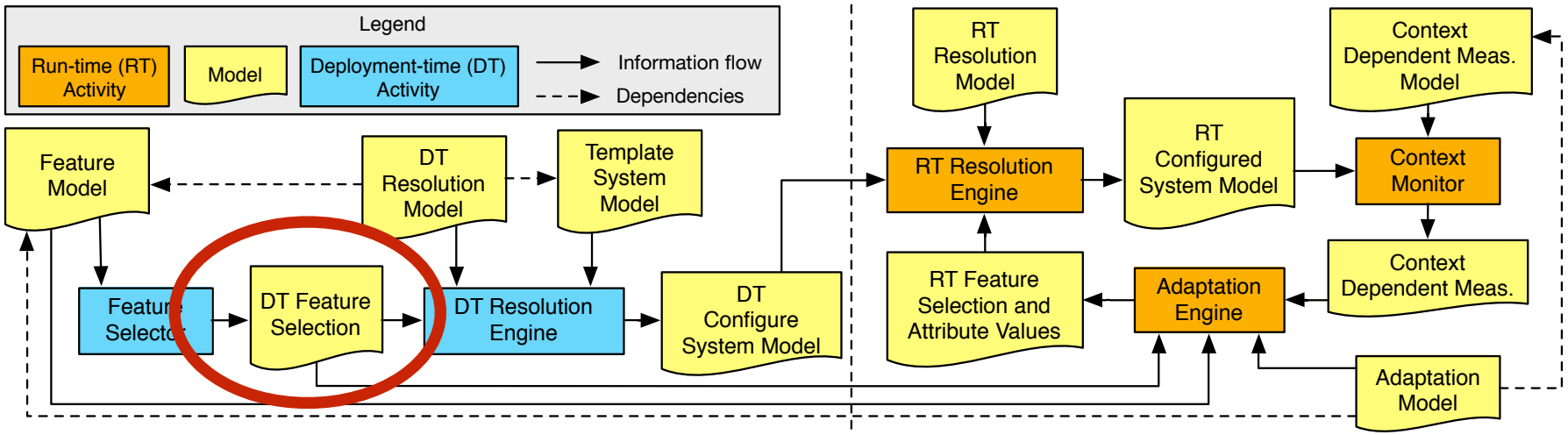


- 3 types of configurations
  - Change component's implementation
  - Change connections
  - Set properties

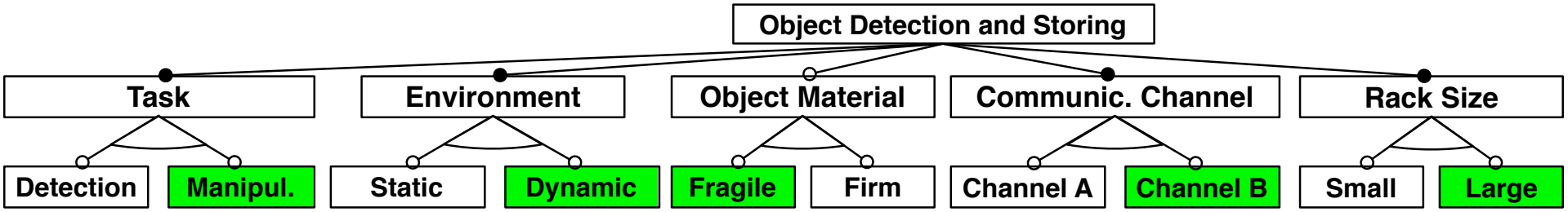
Deployment-time Resolution Model



Deployment-time Feature Selection

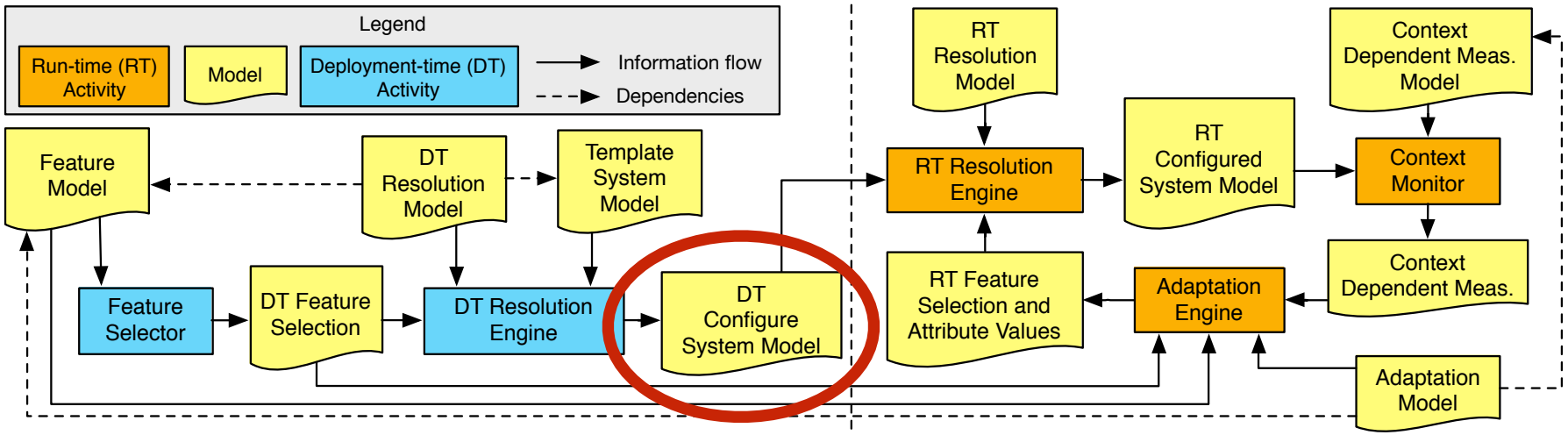


Deployment-time Feature Selection

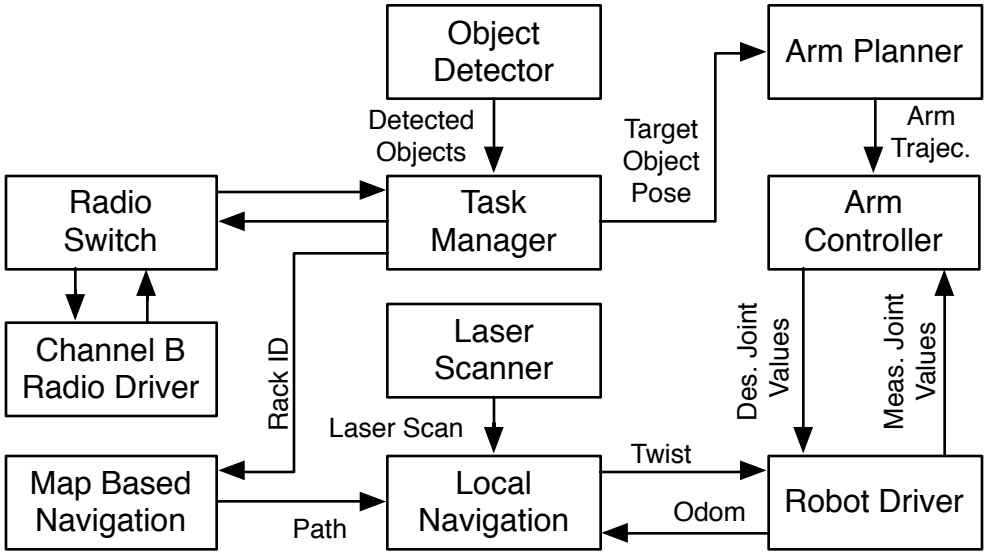


- The feature selection is sent to the Deployment-time Resolution Engine

Deployment-time configured system model



Deployment-time configured system model



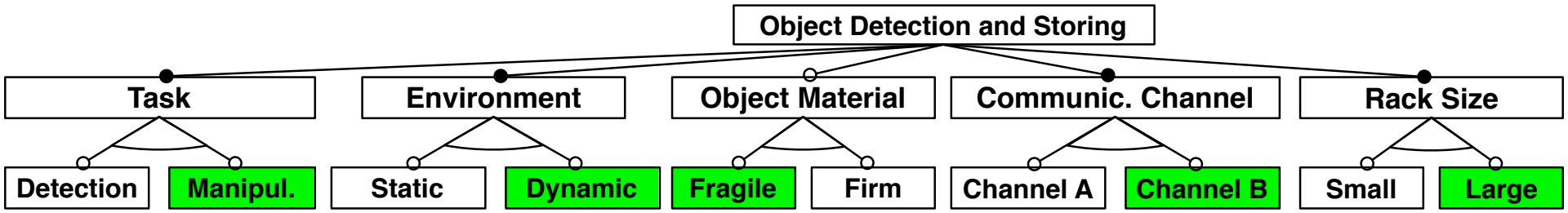
ETH zürich

Run-time Variability Resolution

Luca Gherardi and Nico Hochgeschwender

Adapt run-time architecture based on the state of the context

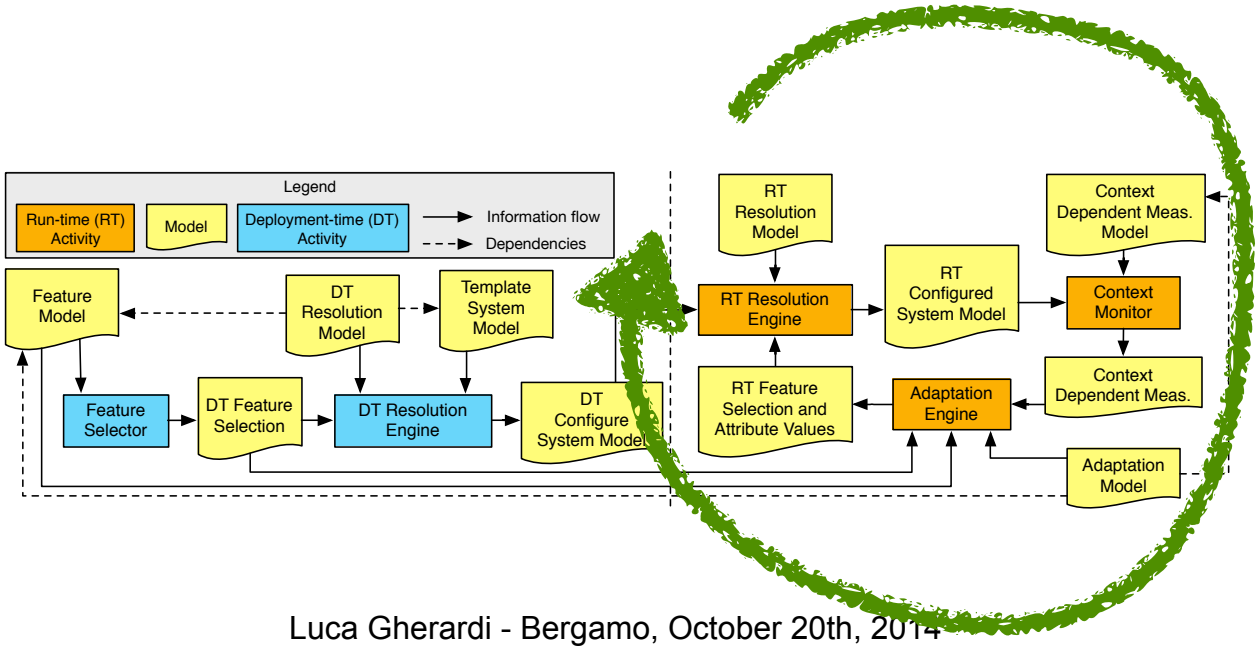
Deployment-time Feature Selection



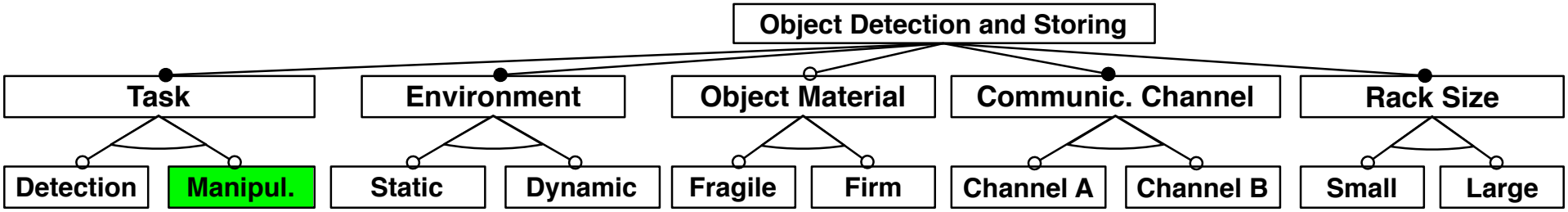
- Some of the features can be better chosen at run-time
  - More information about the context is available
- We have to model this information and reason about it

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

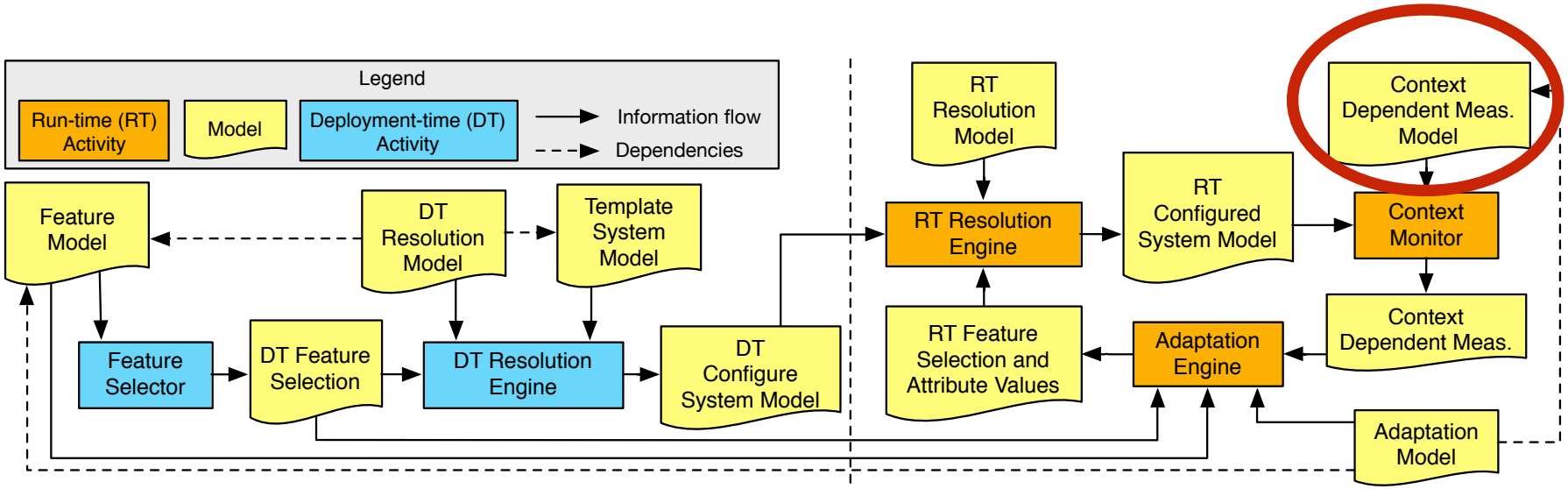


Deployment-time Feature Selection



Feature	Attribute	Value
Channel A	Latency	Low
Channel A	Power Consumption	High
Channel B	Latency	High
Channel B	Power Consumption	Low

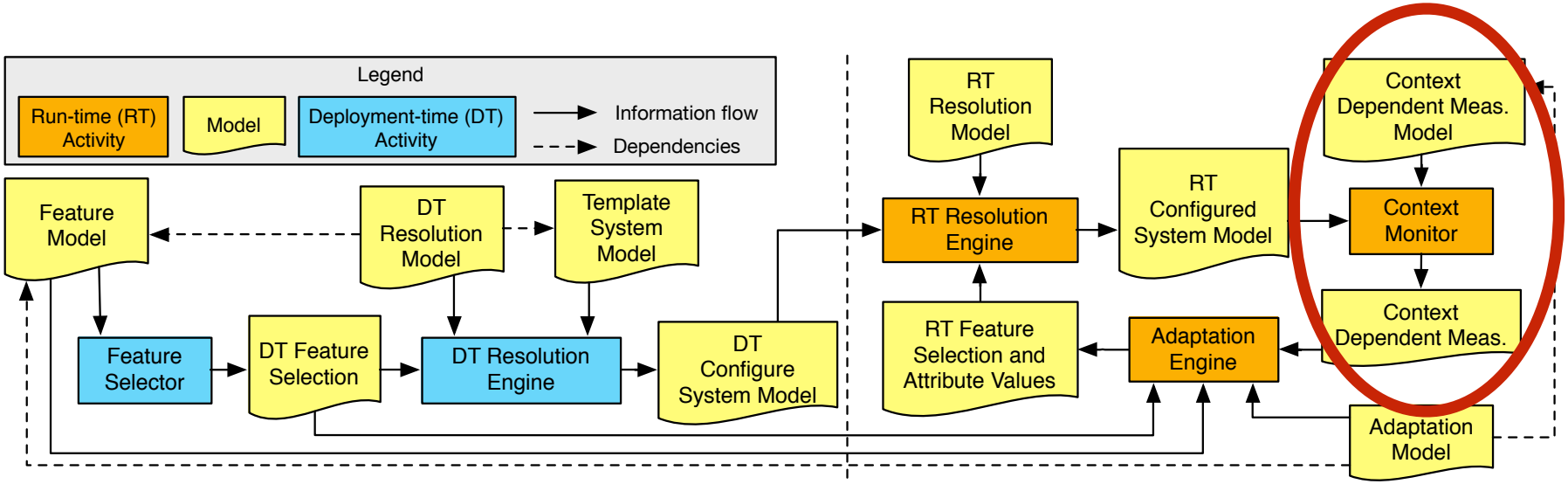
Context Dependent Measurement Model



Context Dependent Measurement Model

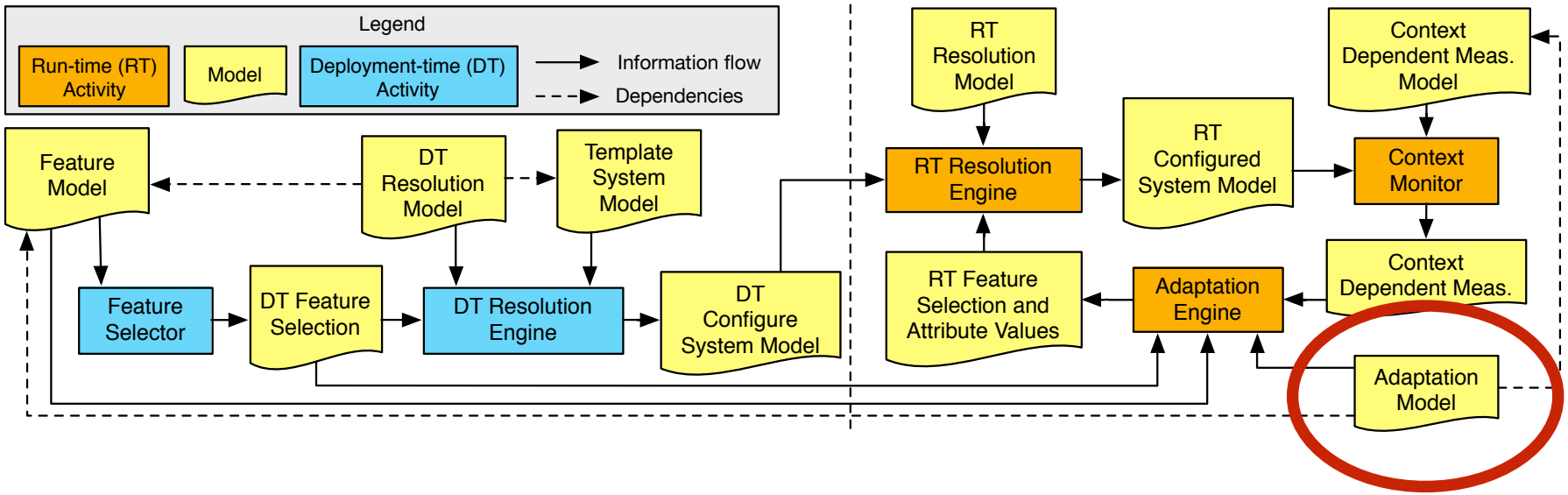
CDM	Component	Interace	Function
BattLev	Robot Driver	Battery Level	—
SemLoc	Map Based Nav.	Semantic Location	—
ObstDist	Laser Scanner	Laser Scan	getClosestObst
Material	Radio Switch	Object Material	—

Context Monitor



- 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

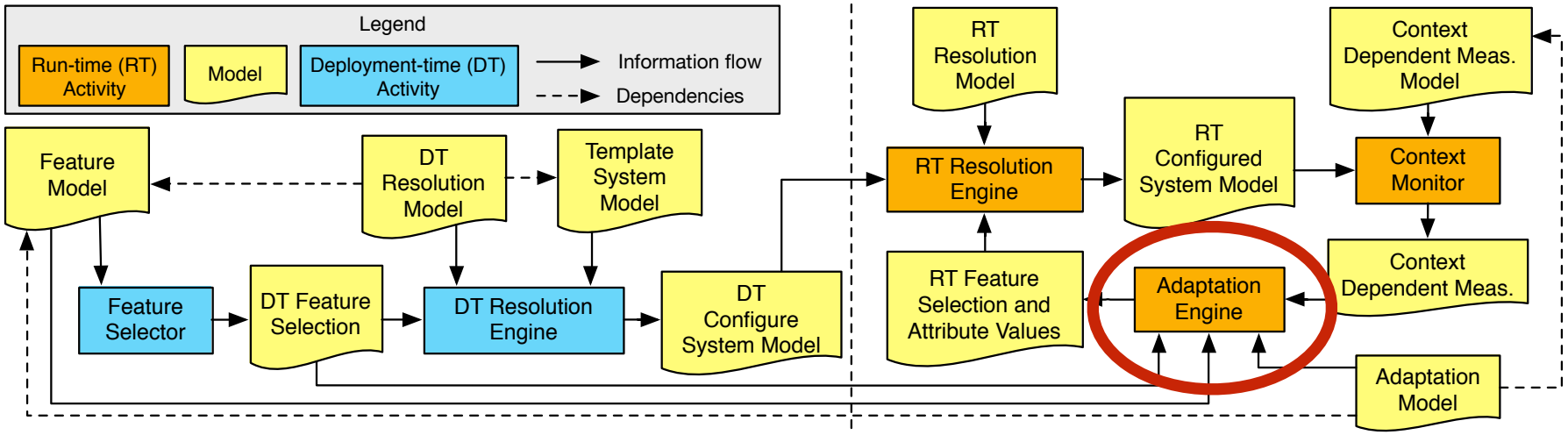


Adaptation Model

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

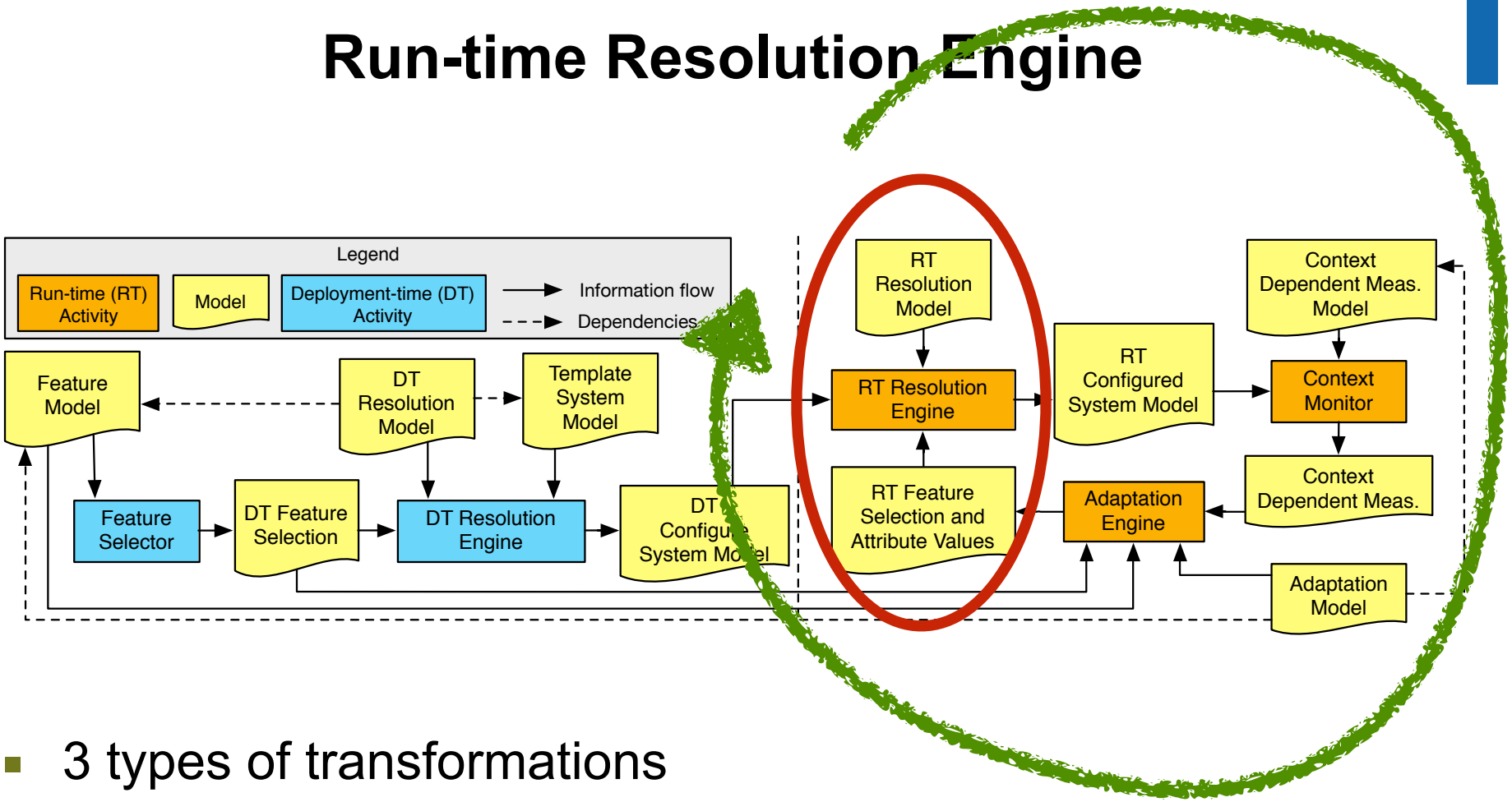
```
rule RackSize:
 if( cdm.SemLoc == "RoomA")
  activate_feature(Small)
 else if( cdm.SemLoc == "RoomB")
  activate_feature(Large)
 else deactivate_feature((Large AND Small)

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(PowerConsump)
```

Adaptation Engine



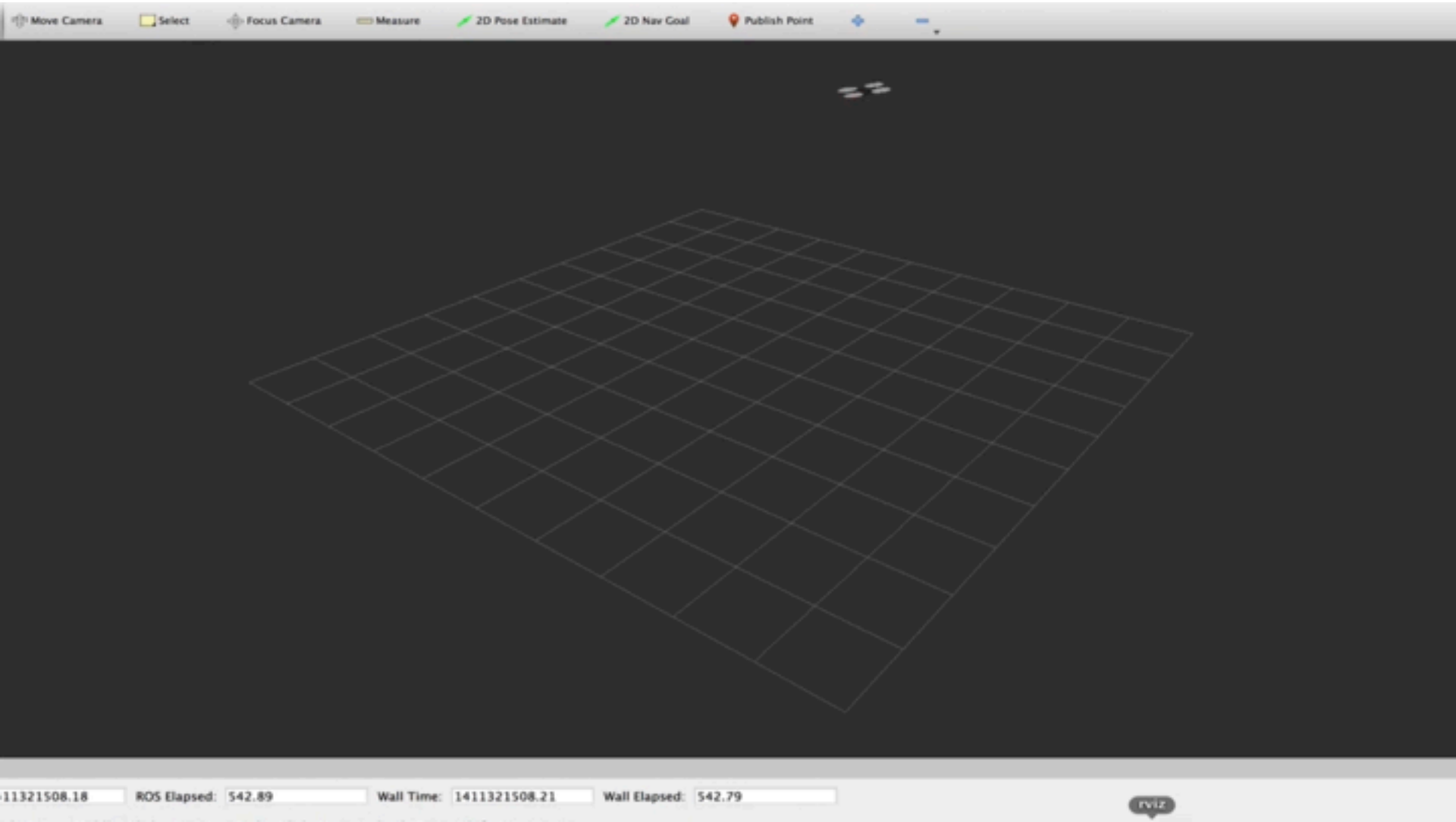
- A ros-java node is automatically configured based on the Adaptation Model
  - Listens for Context Dependent Measurements
  - Produces as output a selection of features

Run-time Resolution Engine



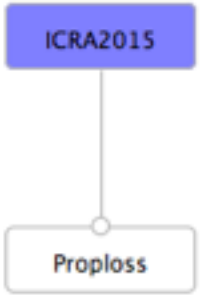
- 3 types of transformations
  - Start/stop components
  - Change connections
  - Set properties

A simulation example

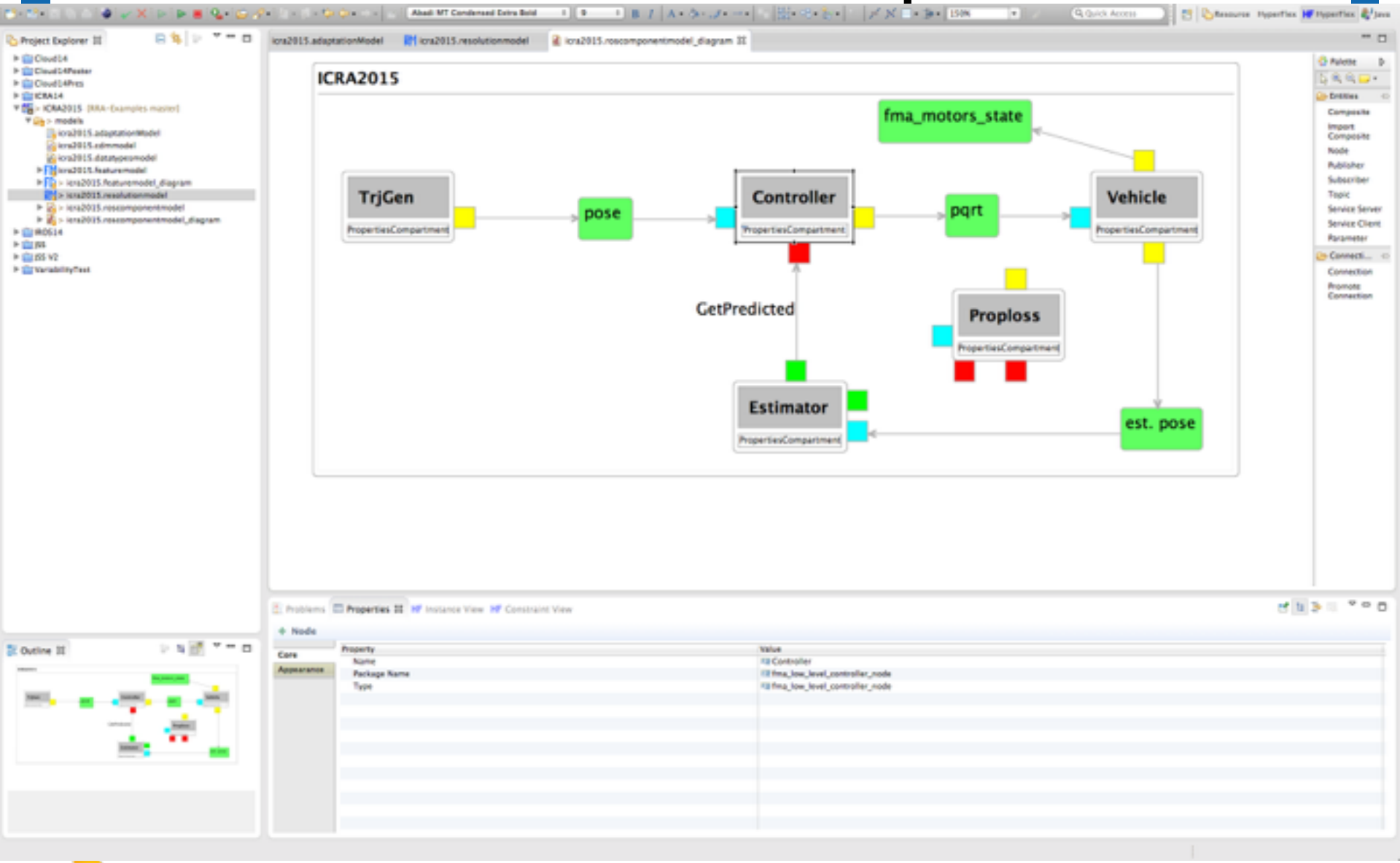


M. W. Mueller and R. D'Andrea. Stability and control of a quadro-copter despite the complete loss of one, two or three propellers.

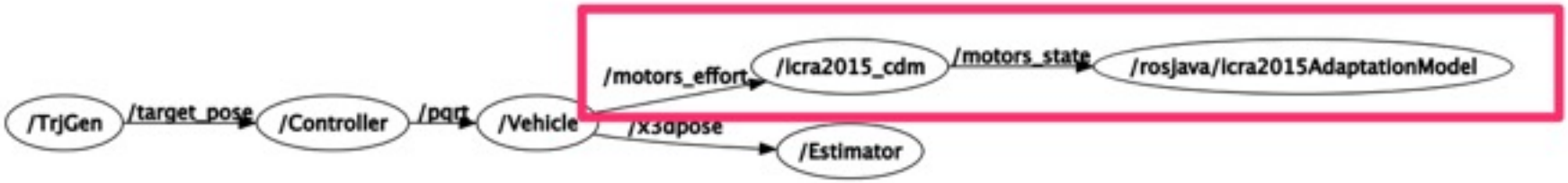
In *International Conference on Robotics and Automation (ICRA)*, 2014.


ICRA2015
Proploss

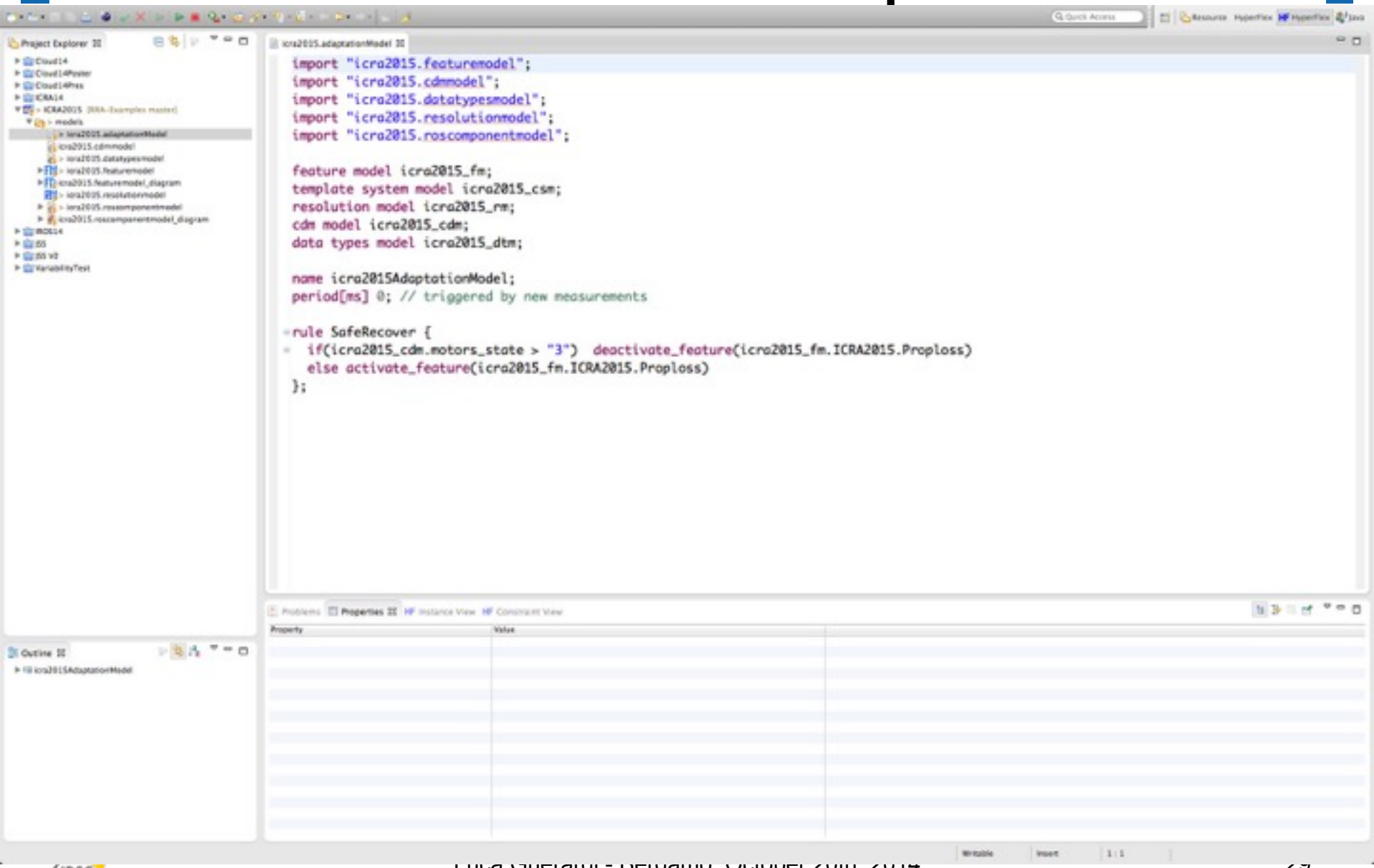
A simulation example



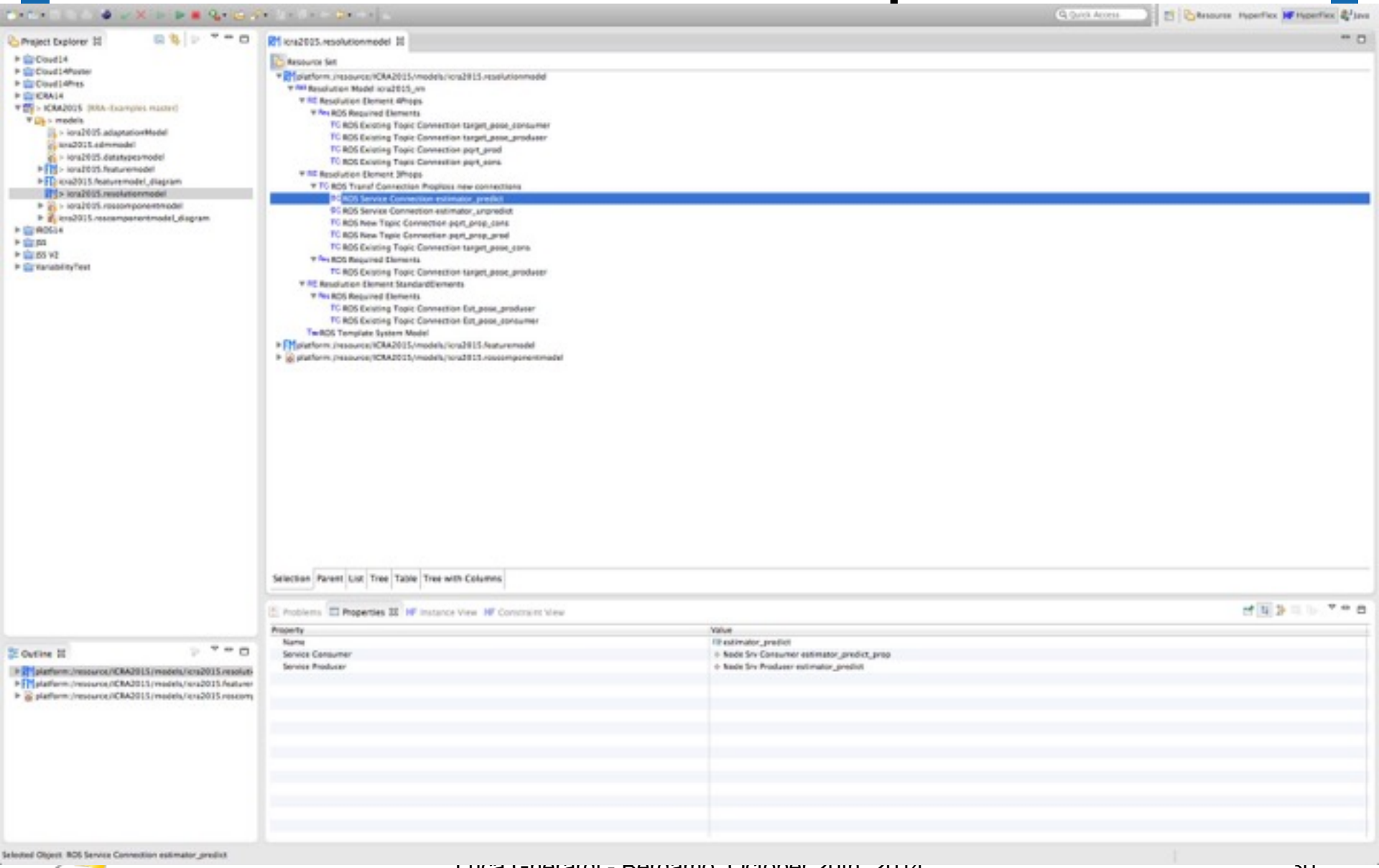
A simulation example



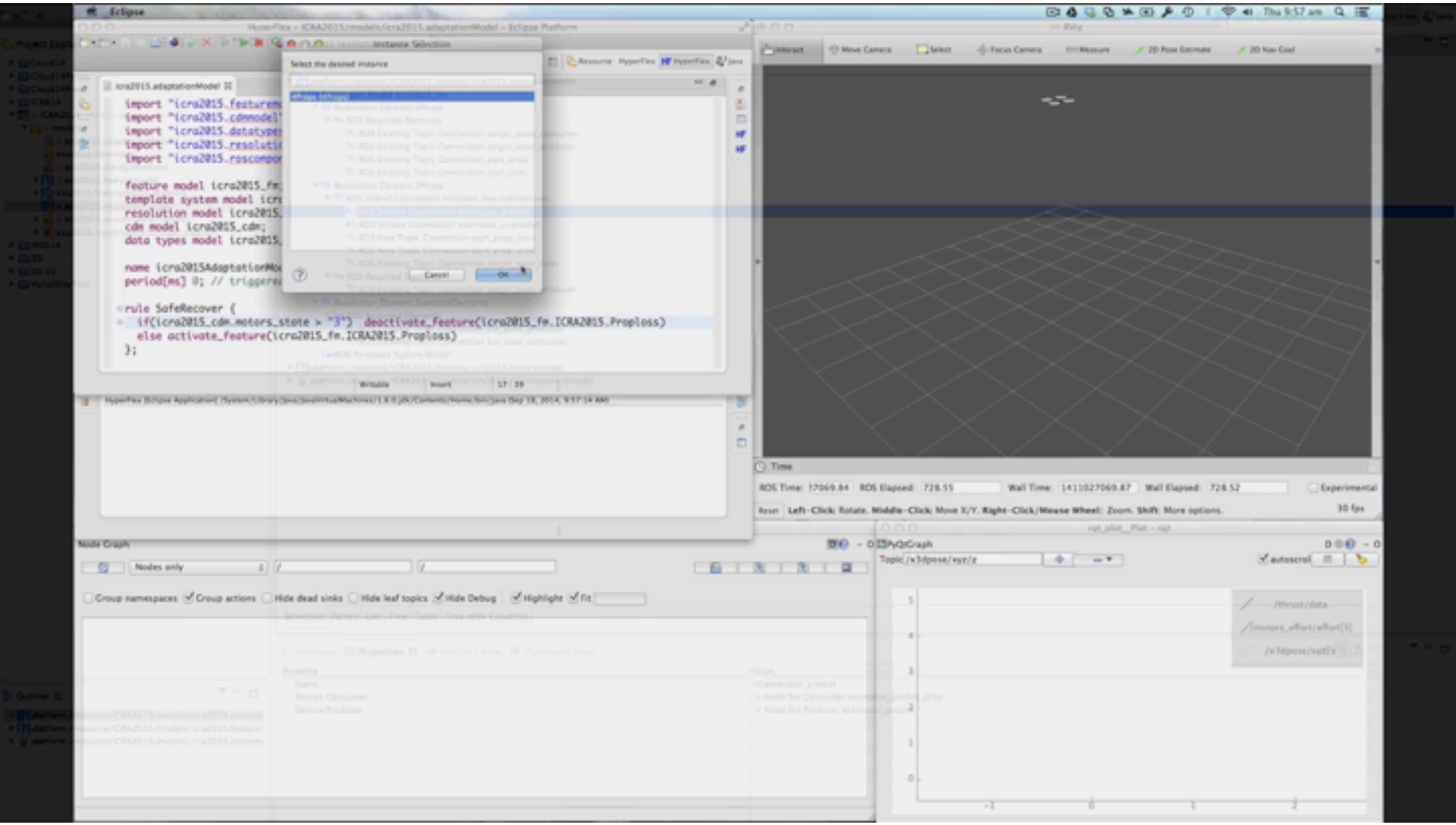
A simulation example



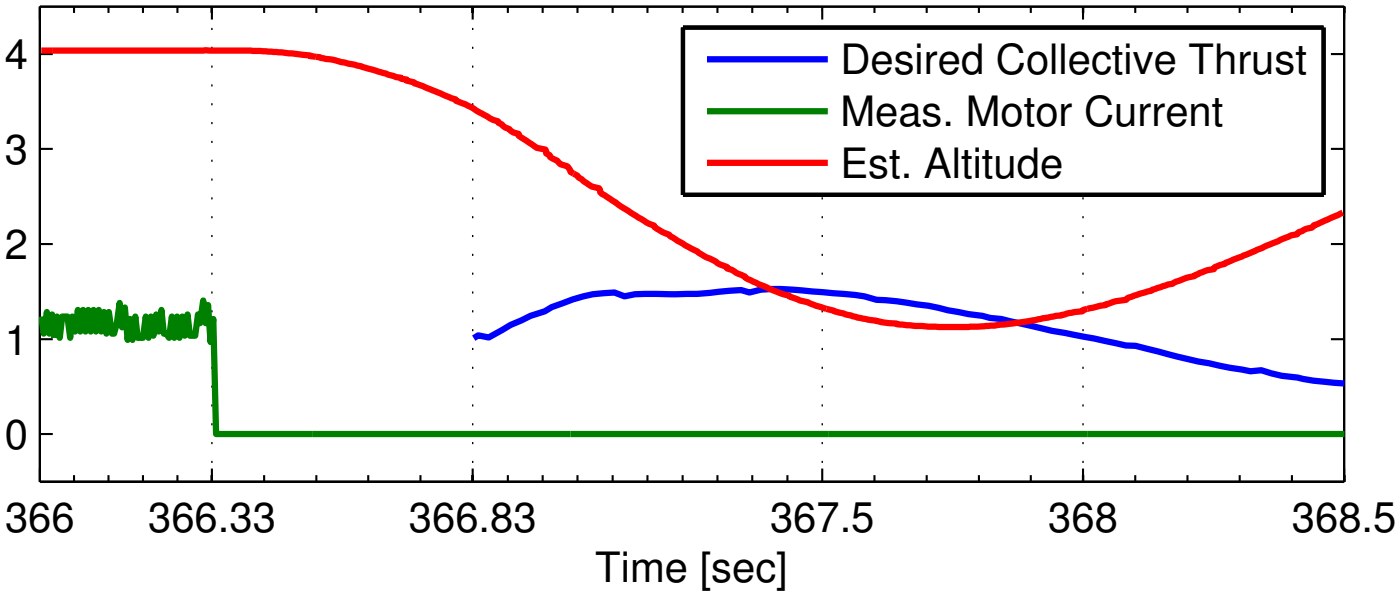
A simulation example



A simulation example



A simulation example



ETH zürich

Thank you!
Any Questions?