Multi-Target Code Generation

13.03.2018

Dipl.-Ing. Evgeny Kusmenko
Software Engineering
RWTH Aachen

http://www.se-rwth.de/

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Simulatoren

TORCS / OpenDaVINCI

Gazebo / ROS

Simulink

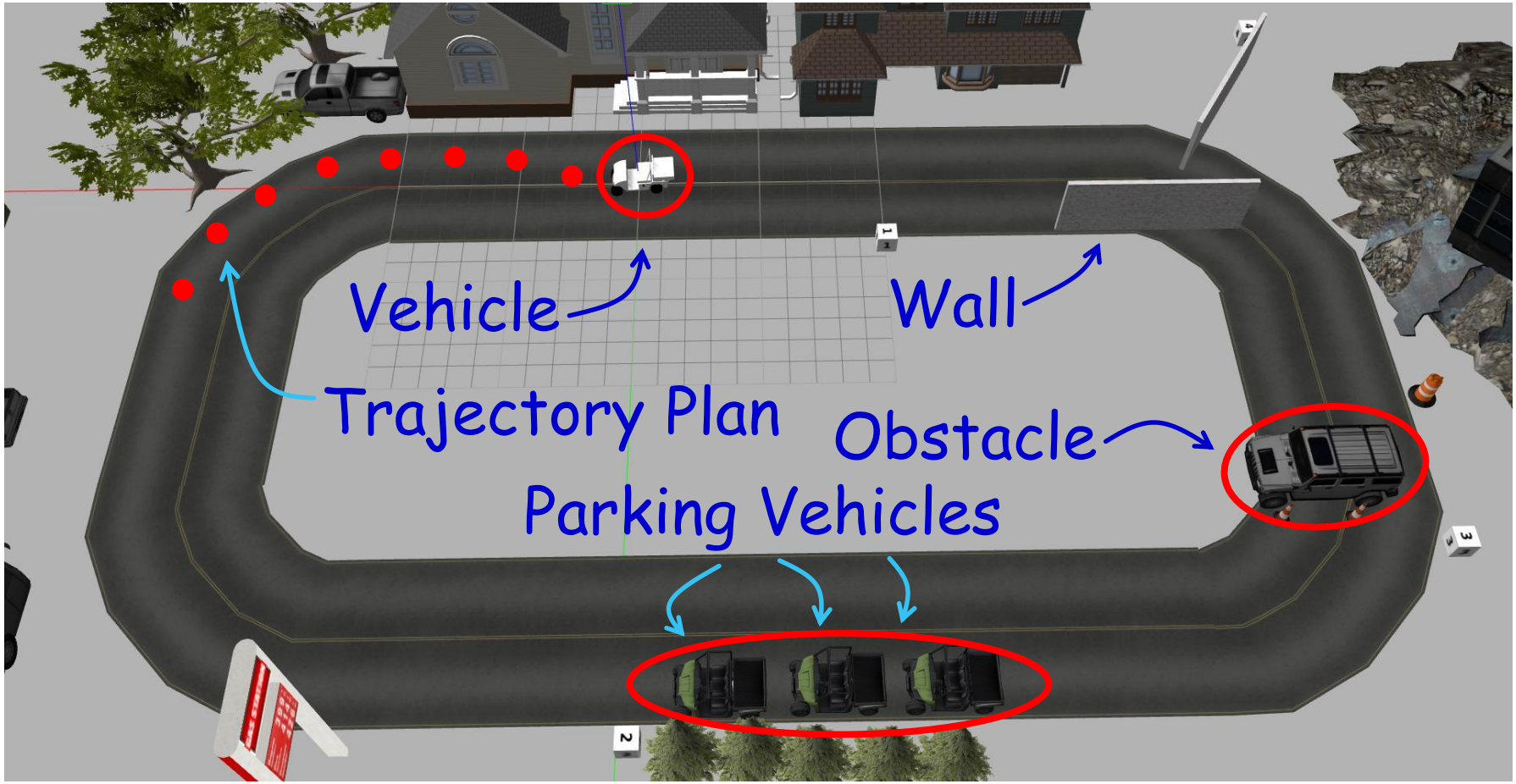
How to get models into a simulator?

VDrift / OpenDaVINCI

MontiSim

Veins

Gazebo Self-Driving Vehicle Example



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

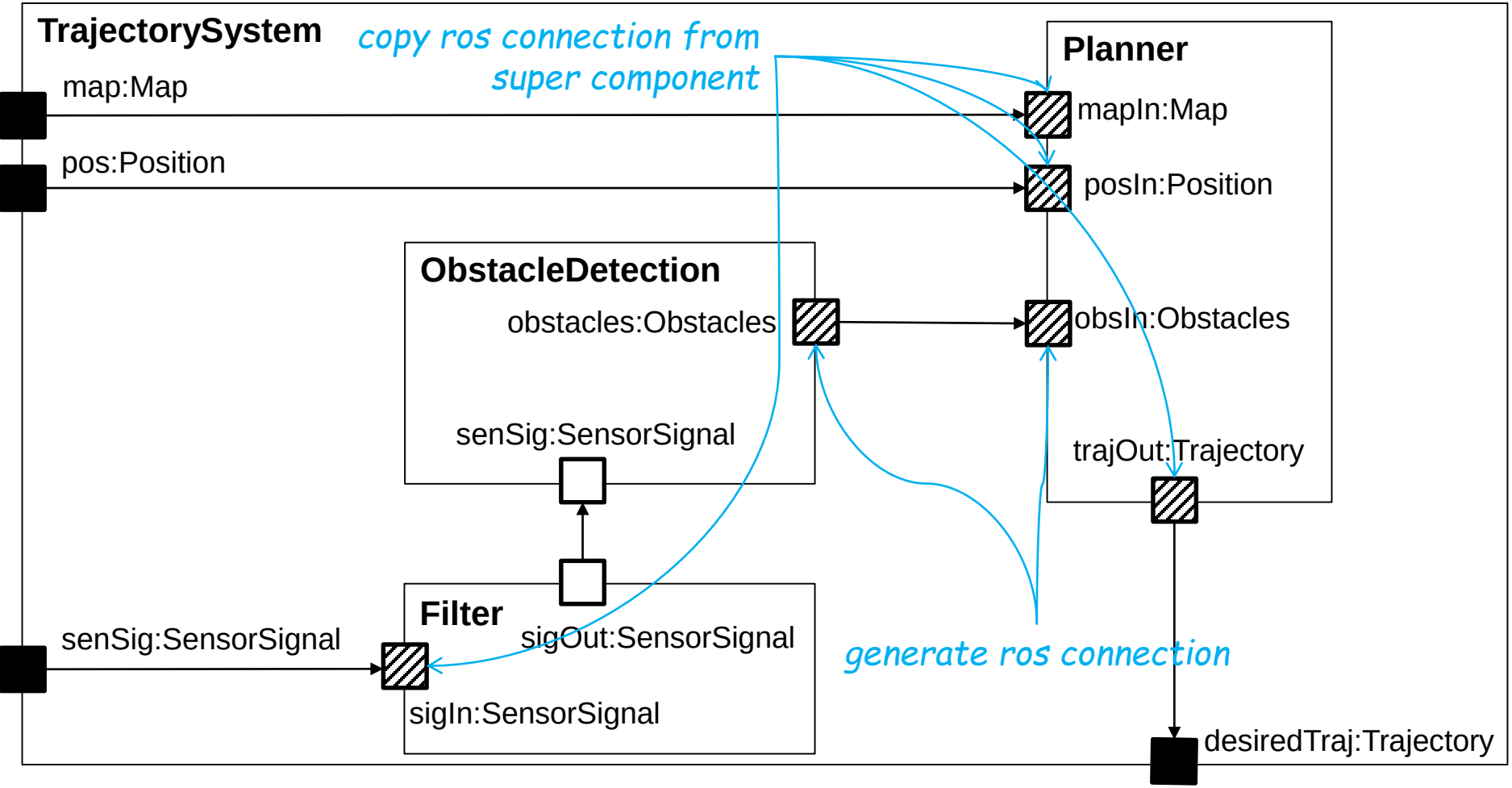
EmbeddedMontiArc Model

```
component TrajectorySystem{
        direction   type      name
  ports in          Map       map,
        in Position pos,
        in SensorSignal senSig,
        out Trajectory desiredTraj;

    instances of subcomponents defined in other artifacts
  instance ObstacleDetection obsDetection;
  instance Filter filter;
  instance Planner planner;
        source port       target port
  connect map  ->  planner.mapIn;
  connect pos  ->  planner.posIn;
  connect planner.trajOut -> desiredTraj;
  connect obsDetection.obstacles -> planner.obsIn;
  /* other connections */ }
```

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

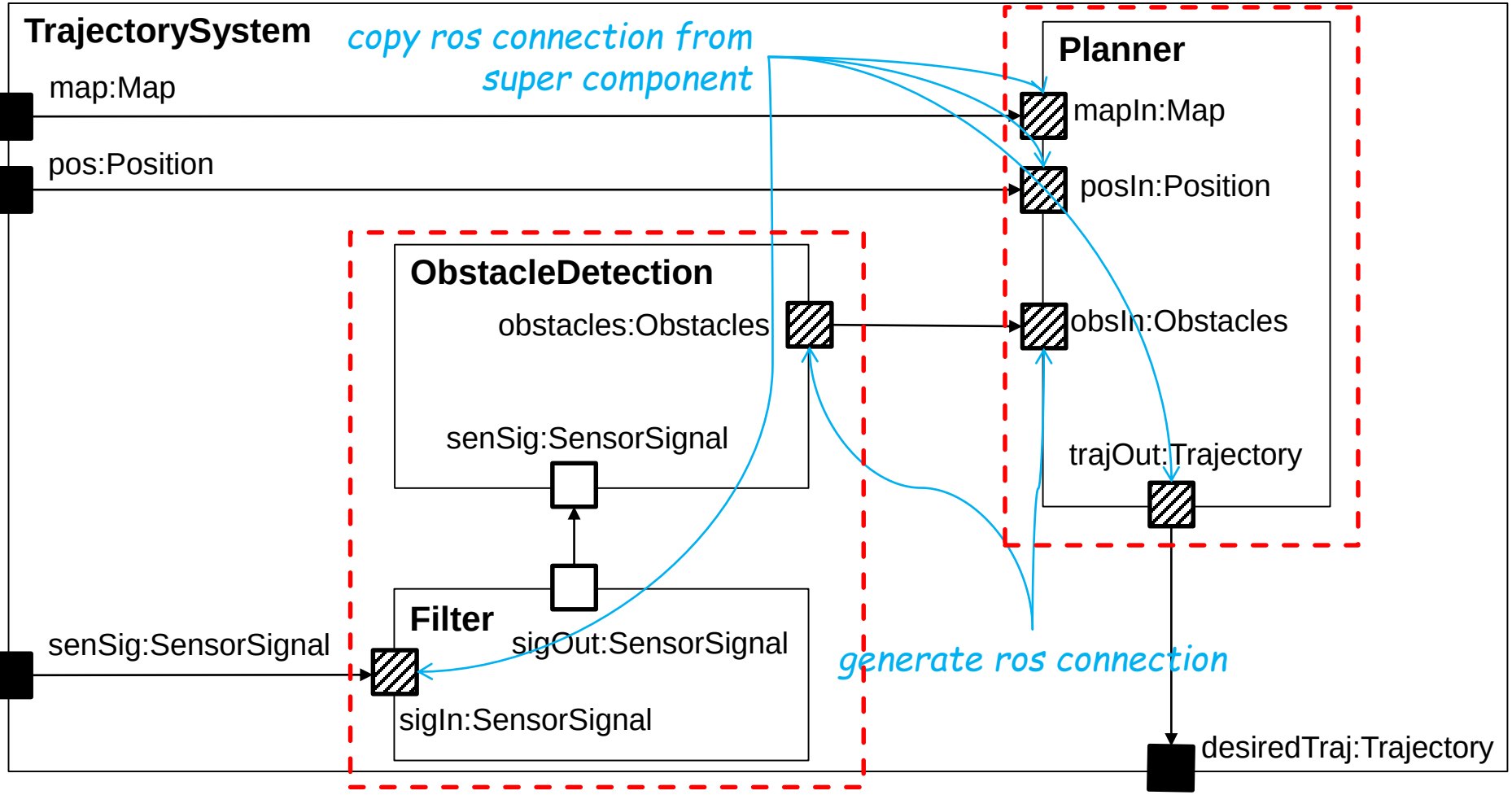
EMAM Diagram



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

EMAM Diagram + Port Tagging

- normal EMAM-Port
- EMAM-Port with incomplete ros info(no topic name or type)
- EMAM-Port with complete ros info



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Tagging Model

tagged symbol

tag type

tag value

```
tags RosTags{
tag trajectorySystem.map with RosConnection =
  {topic= (name=/map, type=struct_msgs/Map)};
```

RosConnection with complete information(■)

```
tag trajectorySystem.desiredTraj with RosConnection =
  {topic= (name=/desiredMotion, type=struct_msgs/Trajectory)};
```

RosConnection with incomplete information(▨)

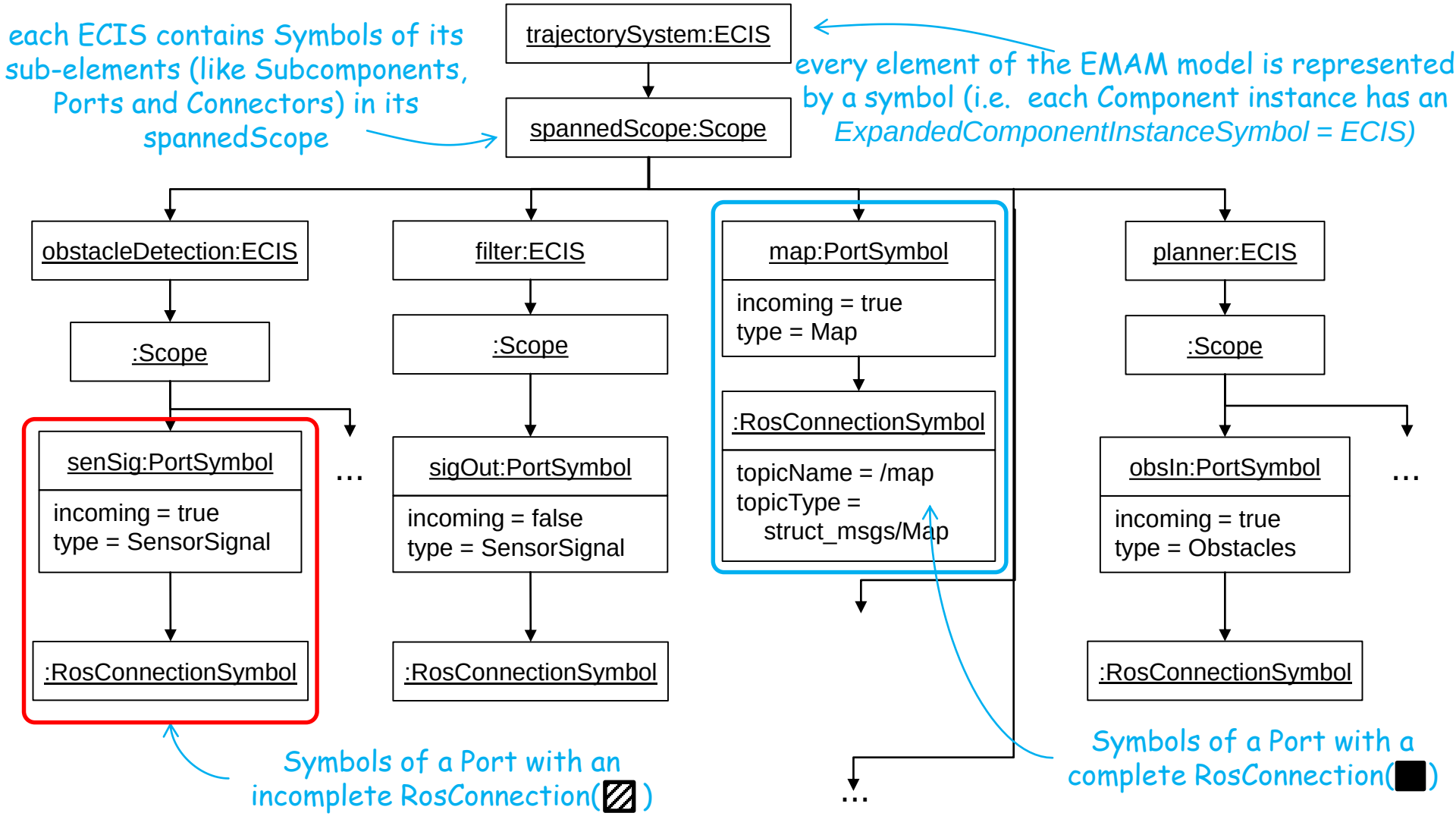
```
tag trajectorySystem.planner.mapIn with RosConnection;
tag trajectorySystem.planner.obsIn with RosConnection;
[...]}
```

type min value step size max value name

```
struct Position{
 Q      ( -90° : 0.000001° :  90°  ) longitude;
 Q      ( -180° : 0.000001° : 180°) latitude;
 Q      ( -10km : 10cm : 10km) altitude;}
```

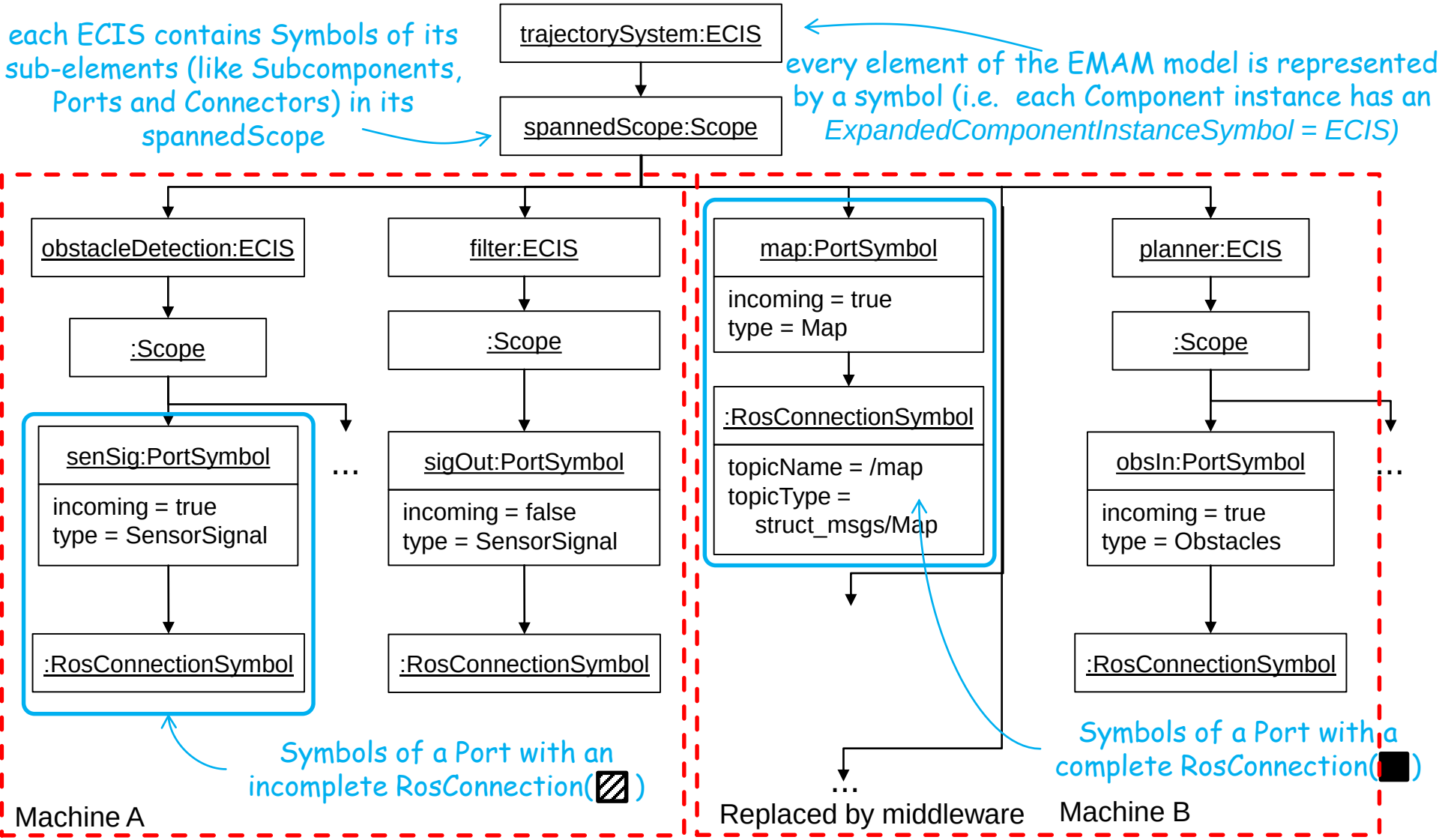
Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Symbol Table

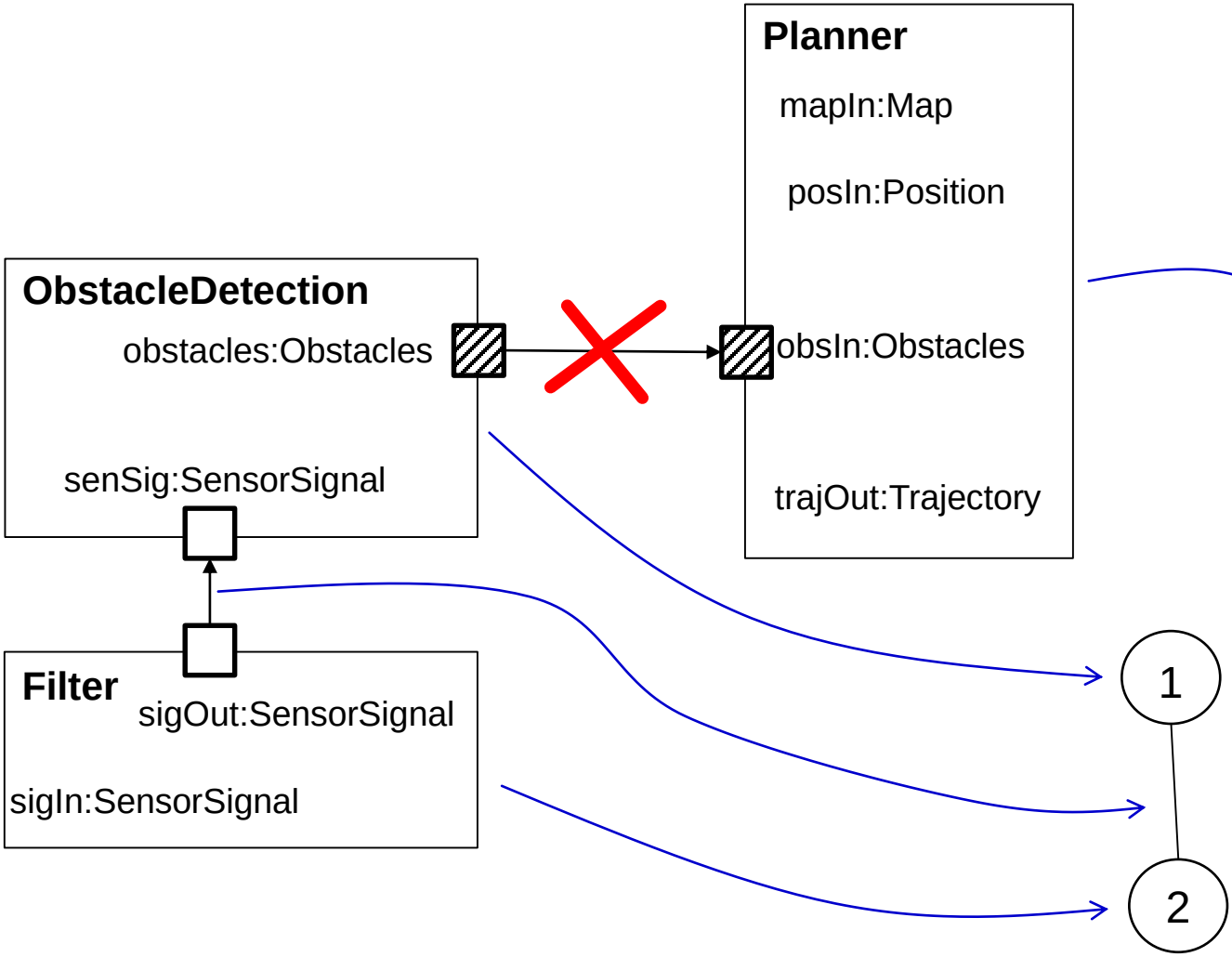


Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Clustered Symbol Table



Exkurs Machine Learning in SE

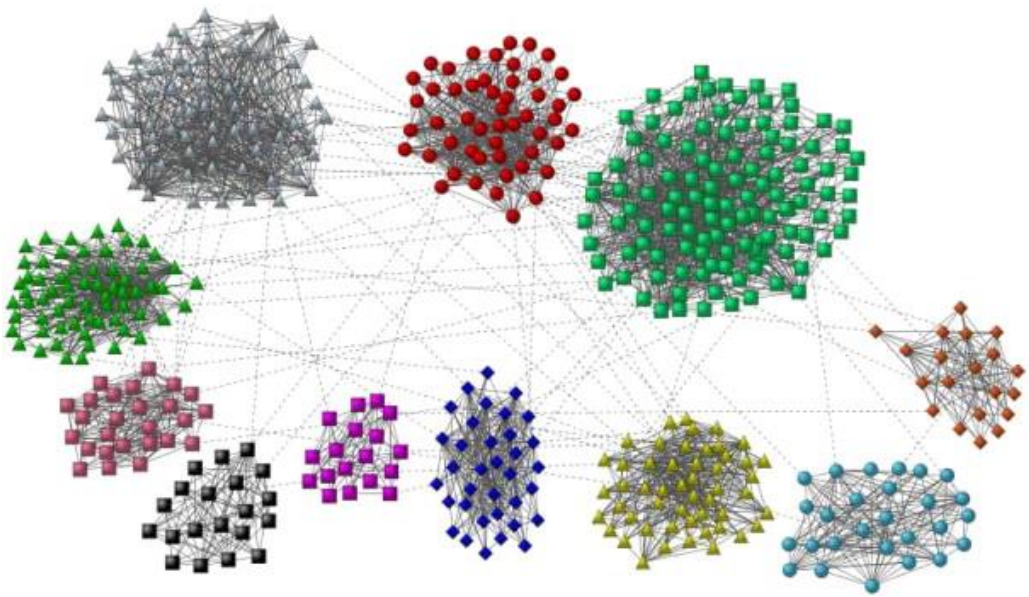


$$G = (V, E)$$
$$E \subseteq V \times V$$

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spectral Analysis

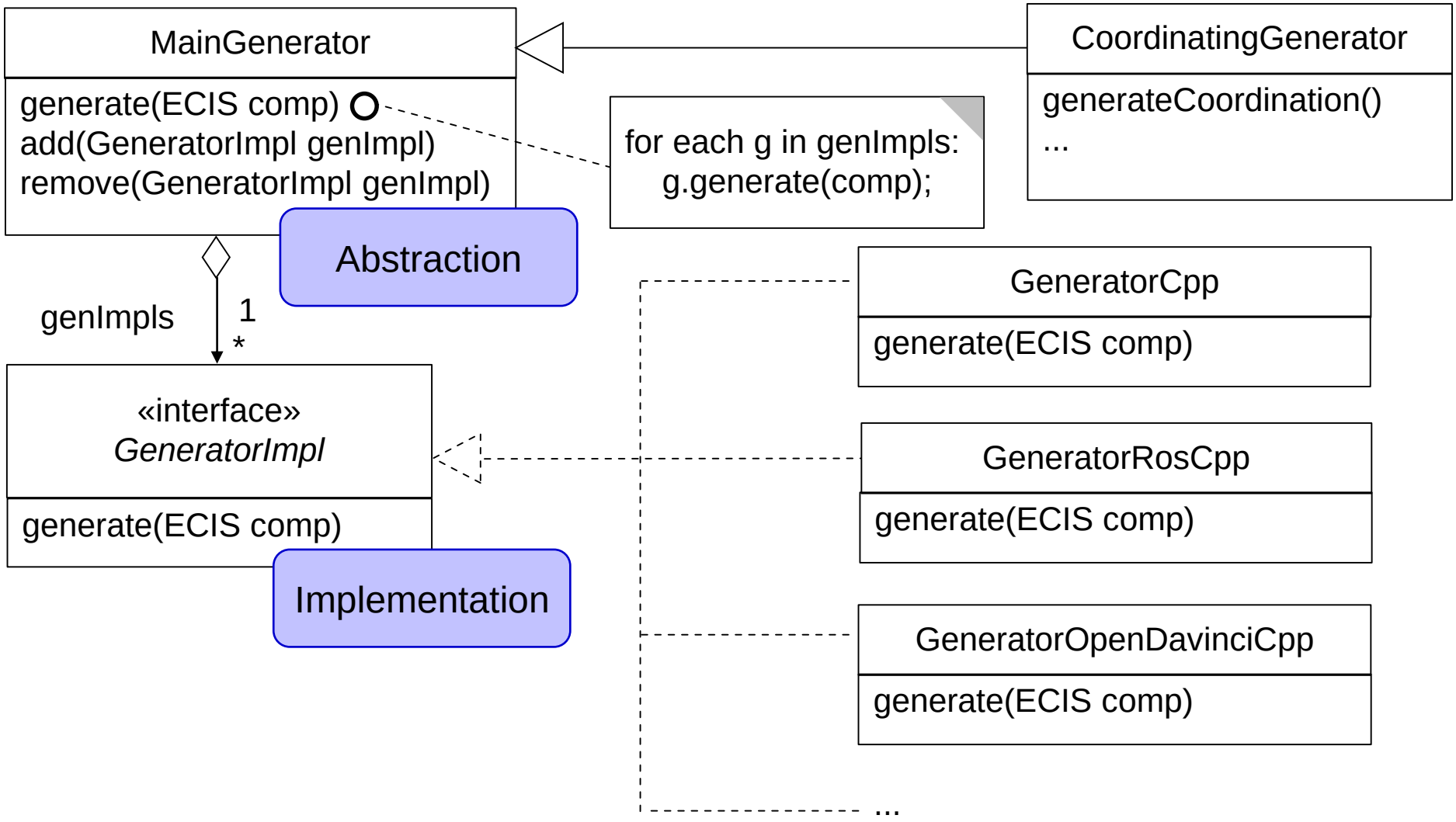
- Solve the eigenproblem
  - $Av = ev$
- For our example:
  - $v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$
  - $v_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$
  - $v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$



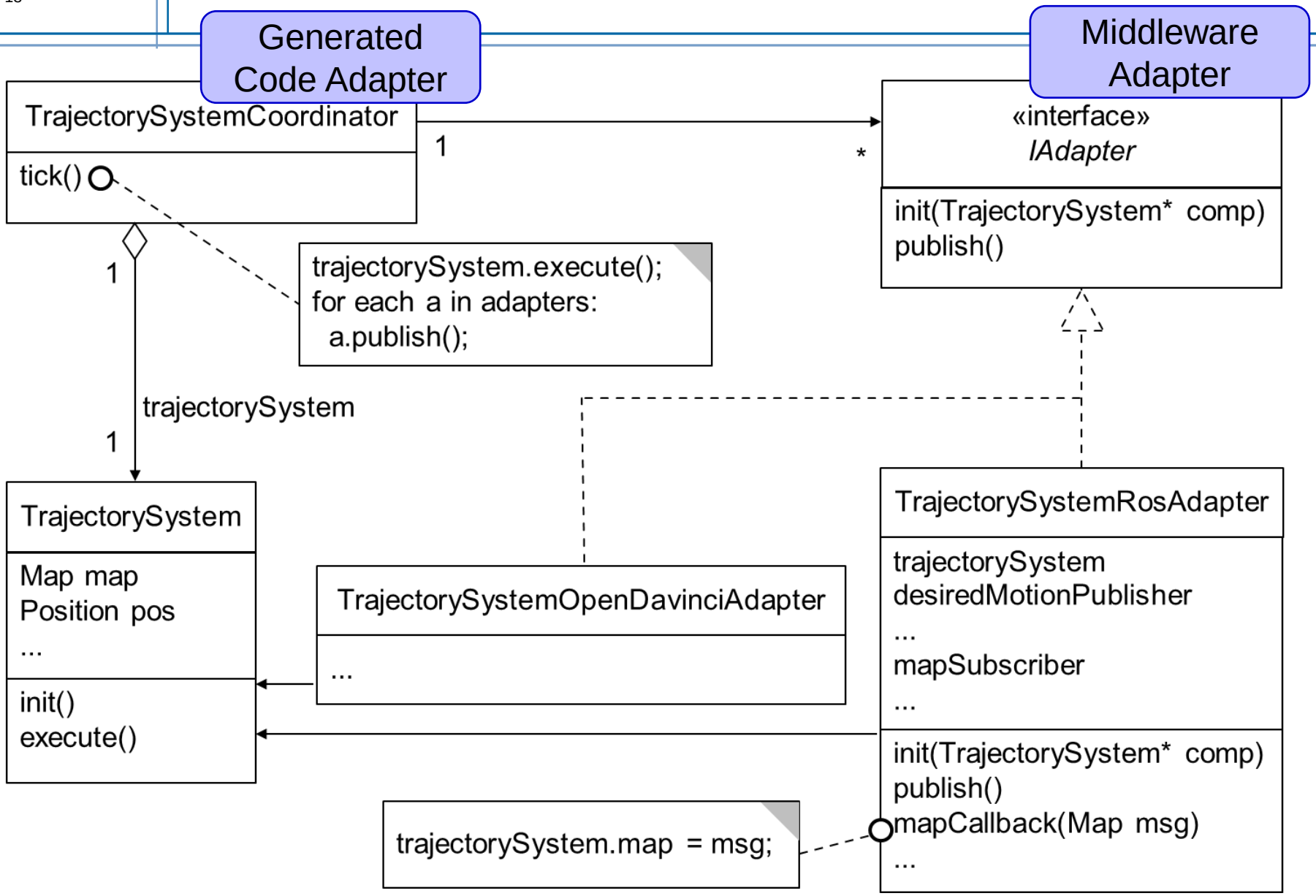
- Non-zero entries appearing in the same ev designate clusters which need to e dedployed on the same machine (no middleware communication)
- Potential: Spectral Methods can help us distribute code to a predefined number of machines
- We use the same technique for error pattern detection for BMW

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

*-Bridge Generator Composition



Generated System incl. Adapter



Generated Artifacts

Generator	Generated files
CoordinatingGenerator	```├── CMakeLists.txt ├── coordinator │ ├── CMakeLists.txt │ ├── TrajectorySystemCoordinator.cpp │ └── IAdapter.h```
GeneratorCpp	```├── cpp │ ├── CMakeLists.txt │ └── TrajectorySystem.h```
GeneratorOpenDavinciCpp	```├── opendavinci │ ├── CMakeLists.txt │ └── TrajectorySystem │ OpenDavinciAdapter.h```

GeneratorRosCpp	```└── roscpp
	  ├── CMakeLists.txt
	  └── TrajectorySystemAdapter.h```

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Generated Adapter

```
#include "TrajectorySystem.h"
[...] /* other includes*/
class TrajectorySystemRosAdapter : public IAdapter{
  TrajectorySystem* trajectorySystem;
  ros::Subscriber mapSubscriber;
  ros::Publisher desiredMotionPublisher;

  public:
  void init(TrajectorySystem* comp){
    trajectorySystem = comp;
    /*init publishers, subscribers and start ROS thread*/
    [...]
  }
  void publish(){
    struct_msgs::Map tmpMsg =
            msgFromStructMap(trajectorySystem->desiredTraj);
    desiredMotionPublisher.publish(tmpMsg);
  }

  void mapCallback(struct_msgs::Map& msg){
    trajectorySystem->map = structFromMsgMap(msg);
  } [...] };
```

instance shared between the coordinator and other adapters

called once by TrajectorySystemCoordinator

called in a defined interval by TrajectorySystemCoordinator

called by ROS every time a message is published on the topic **/map**