SE SOFTWARE ENGINEERING

RWTH AACHEN UNIVERSITY

Multi-Target Code Generation

12.06.2018

Alexander Hellwig
Evgeny Kusmenko

Software Engineering
RWTH Aachen

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

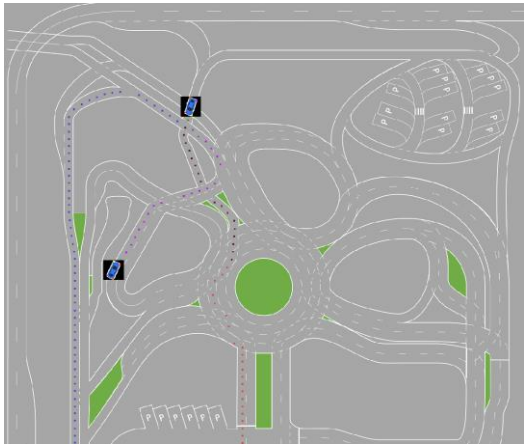
Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Simulatoren

TORCS / OpenDaVINCI

Gazebo / ROS

Simulink



How to get models into a simulator?

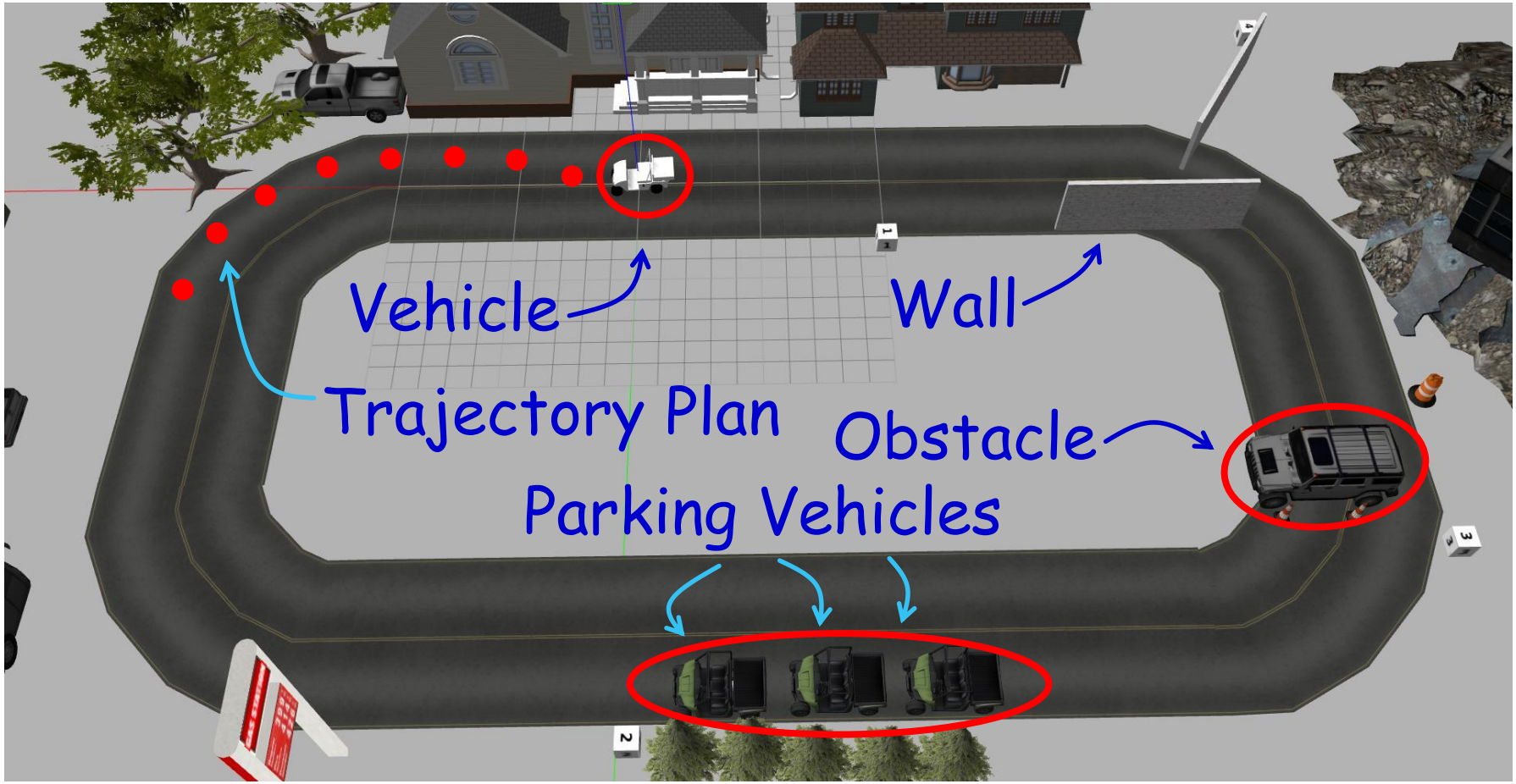
VDrift / OpenDaVINCI

MontiSim

Veins

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Gazebo Self-Driving Vehicle Example



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

EmbeddedMontiArc Model

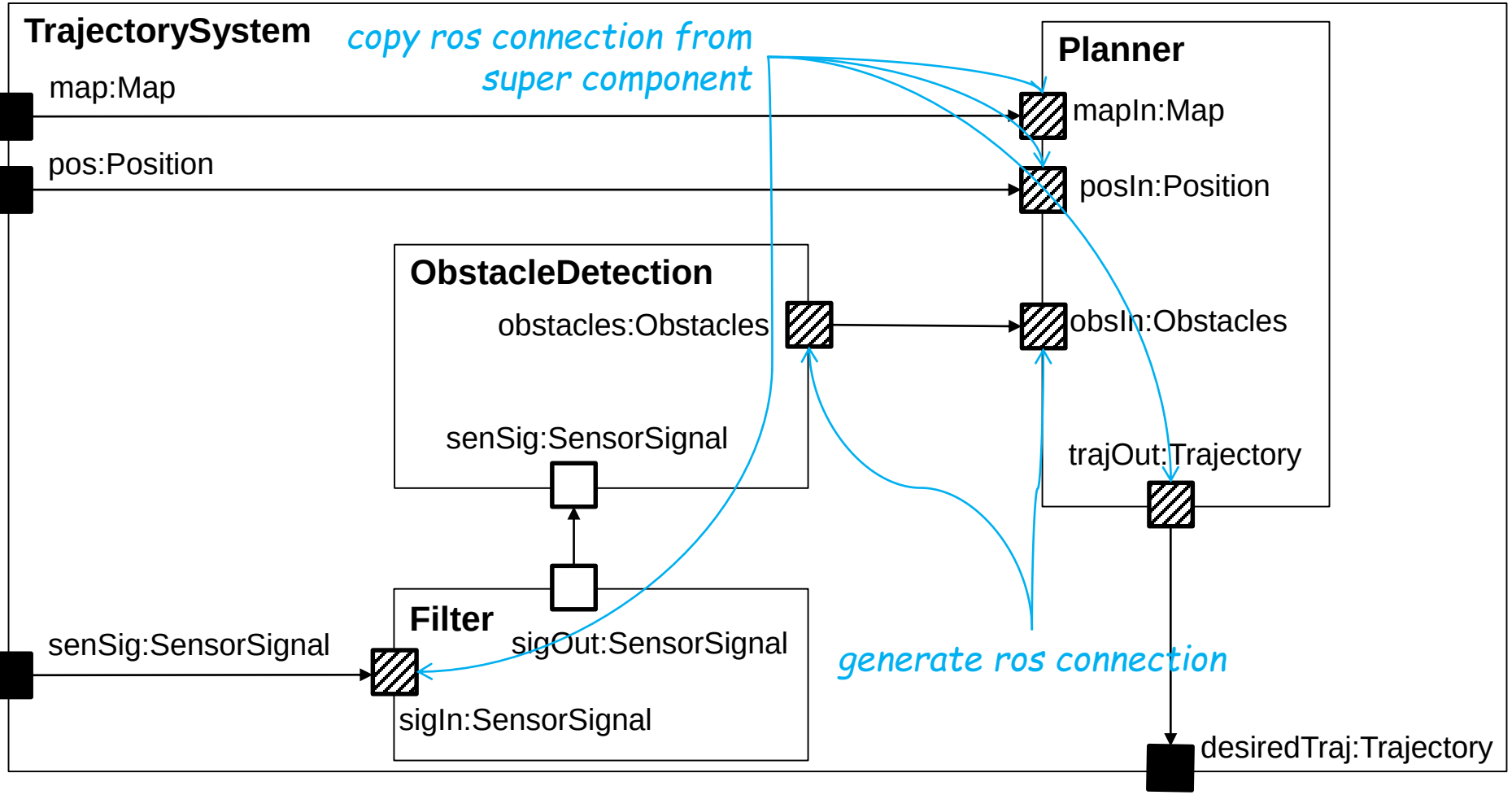
```
1  component TrajectorySystem{
       direction  type     name
2    ports in     Map      map,
3          in Position pos,
4          in SensorSignal senSig,
5          out Trajectory desiredTraj;

     instances of subcomponents defined in other artifacts
6    instance ObstacleDetection obsDetection;
7    instance Filter filter;
8    instance Planner planner;

         source port      target port
9    connect map   ->  planner.mapIn;
10   connect pos   ->  planner.posIn;
11   connect planner.trajOut -> desiredTraj;
12   connect obsDetection.obstacles -> planner.obsIn;
13   /* other connections */ }
```

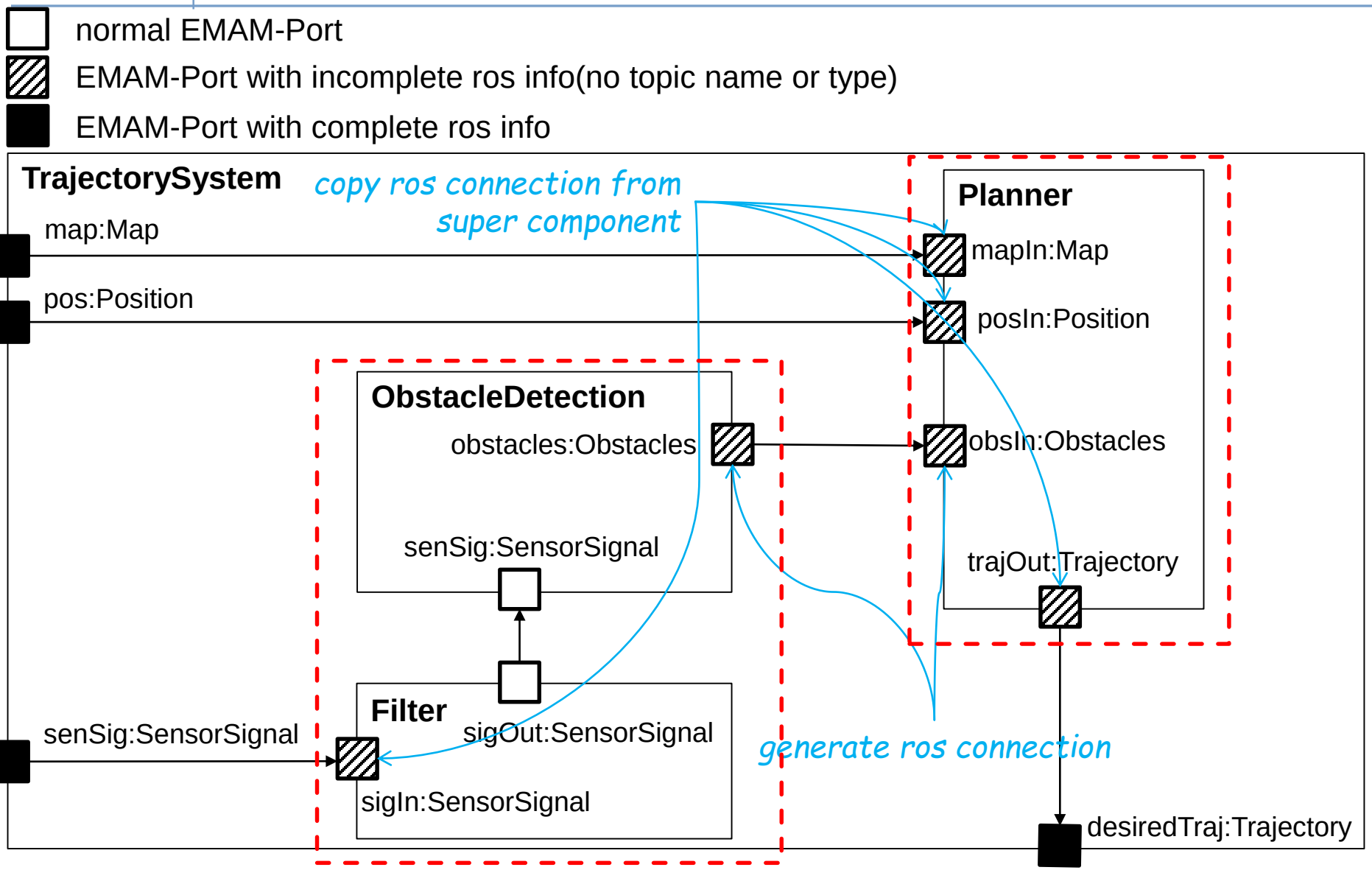
Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

EMAM Diagram



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

EMAM Diagram + Port Tagging



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Tagging Model

```
1 tags RosTags{
2 tag trajectorySystem.map with RosConnection =
    {topic= (name=/map, type=struct_msgs/Map)};
3 tag trajectorySystem.desiredTraj with RosConnection =
    {topic= (name=/desiredMotion, type=struct_msgs/Trajectory)};
4 tag trajectorySystem.planner.mapIn with RosConnection;
5 tag trajectorySystem.planner.obsIn with RosConnection;
6 [...]}
```

tagged symbol

tag type

tag value

RosConnection with complete information( )

RosConnection with incomplete information( )

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

type

min value

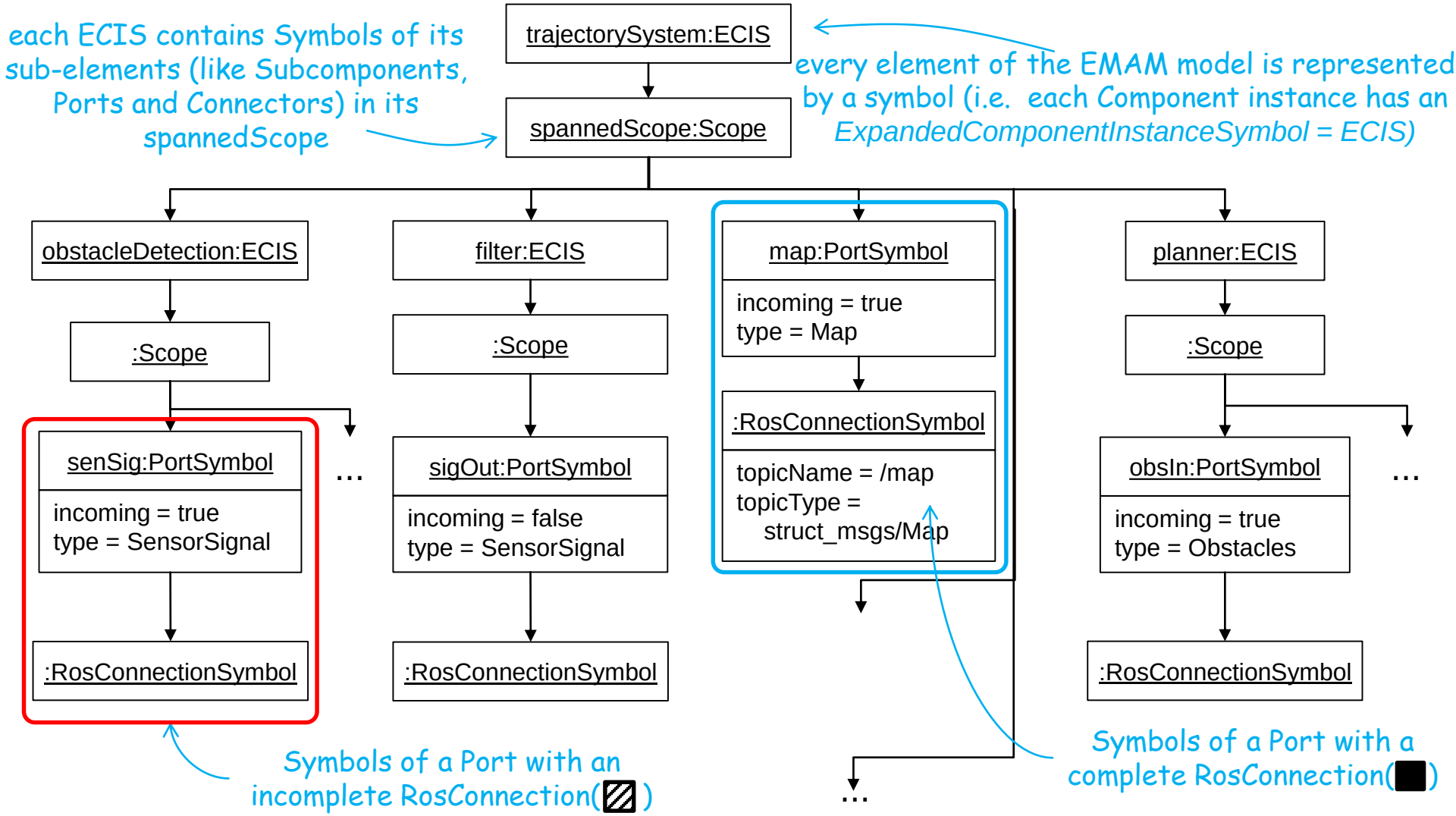
step size

max value

name

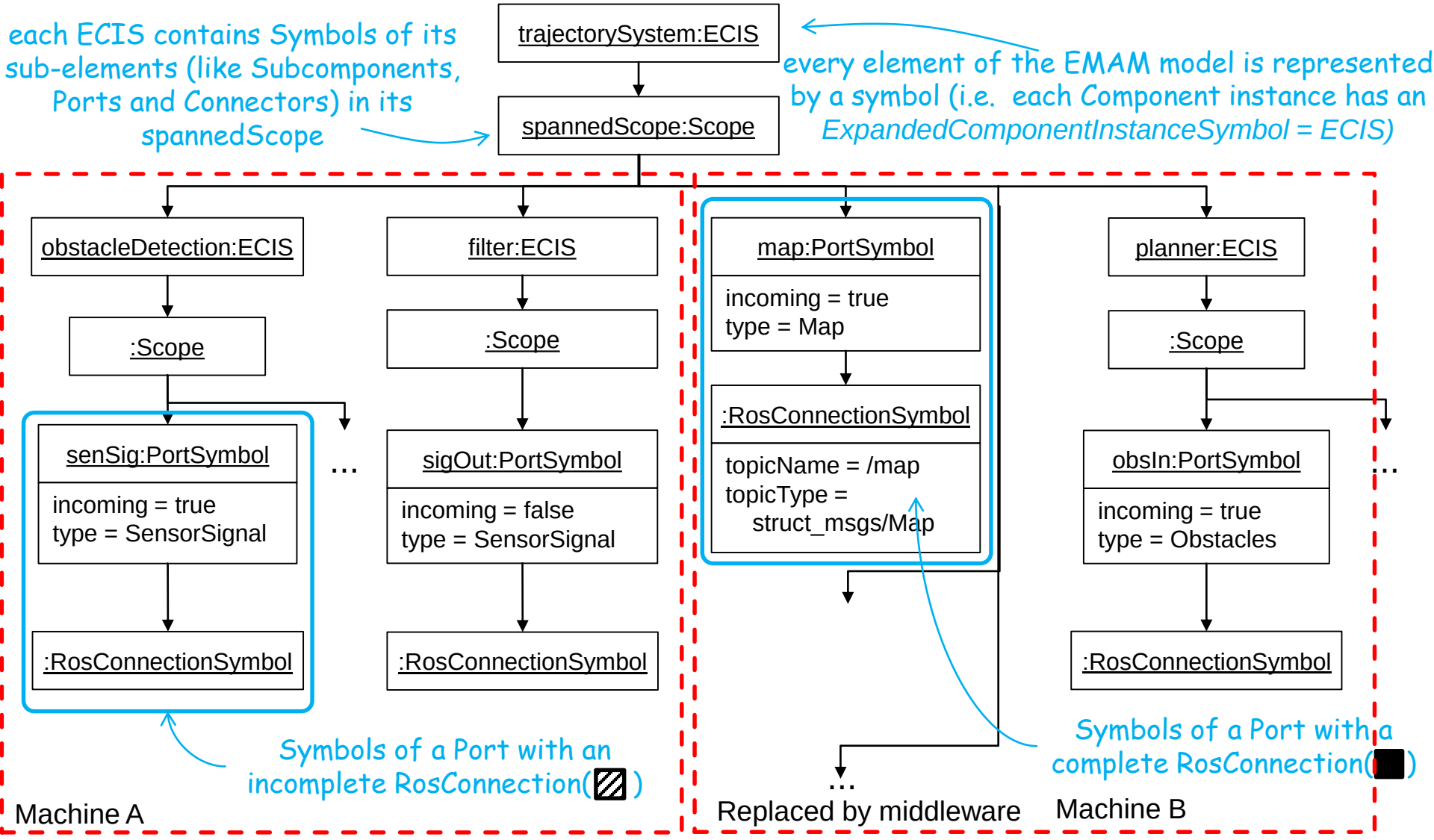
Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Symbol Table

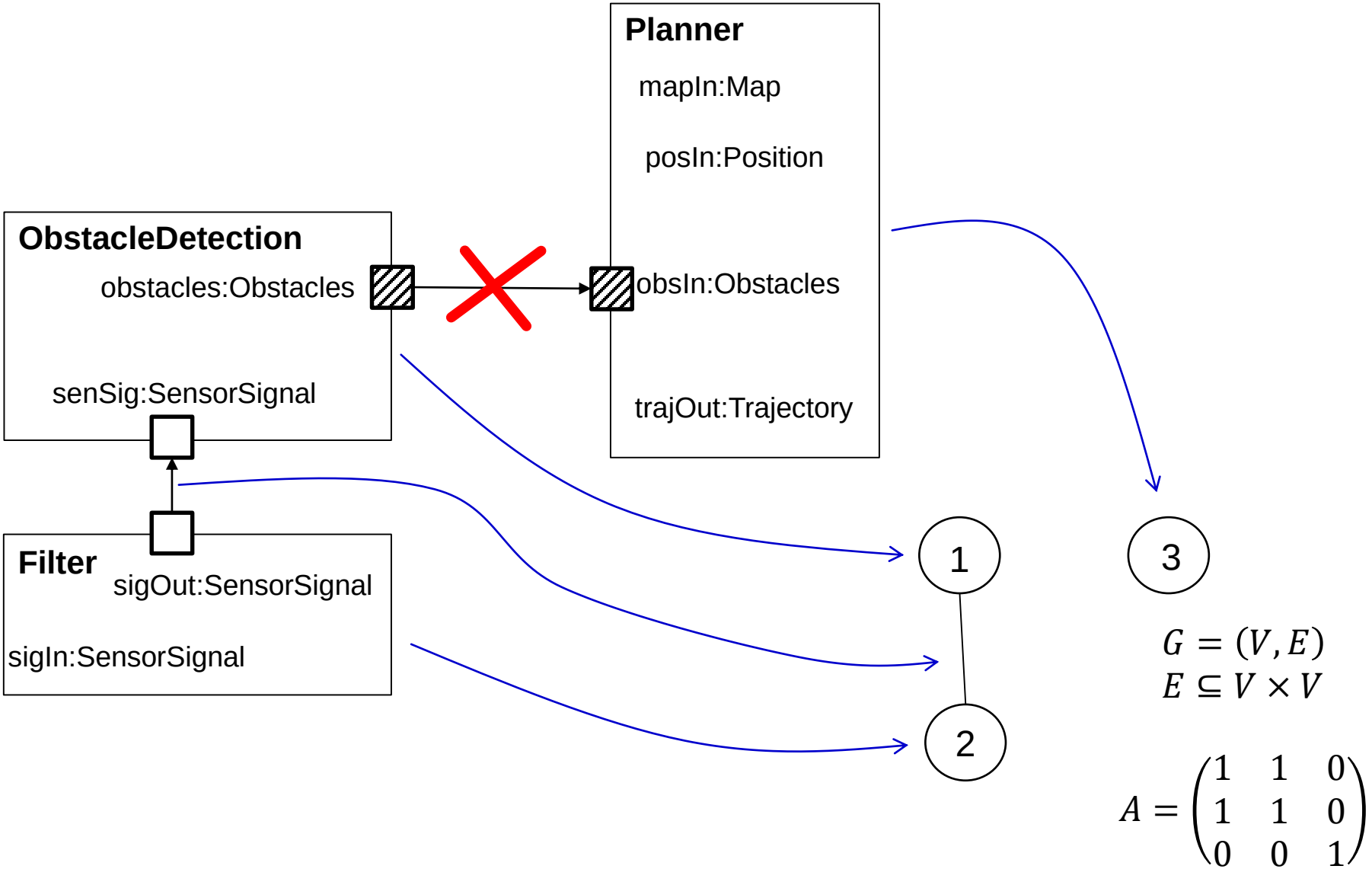


Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Clustered Symbol Table



Exkurs Machine Learning in SE



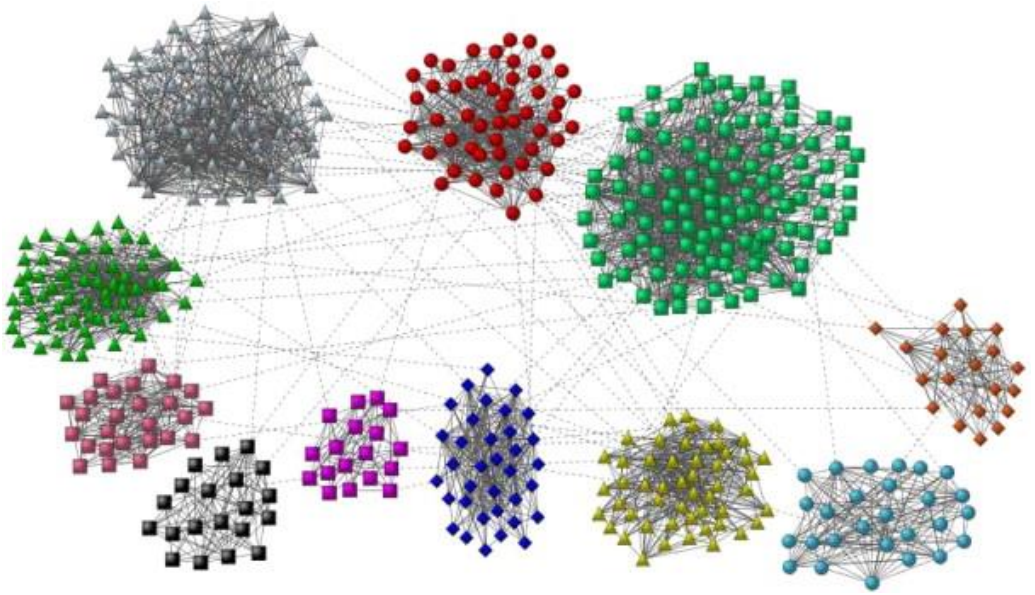
$$G = (V, E)$$
$$E \subseteq V \times V$$

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

Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

Spectral Analysis

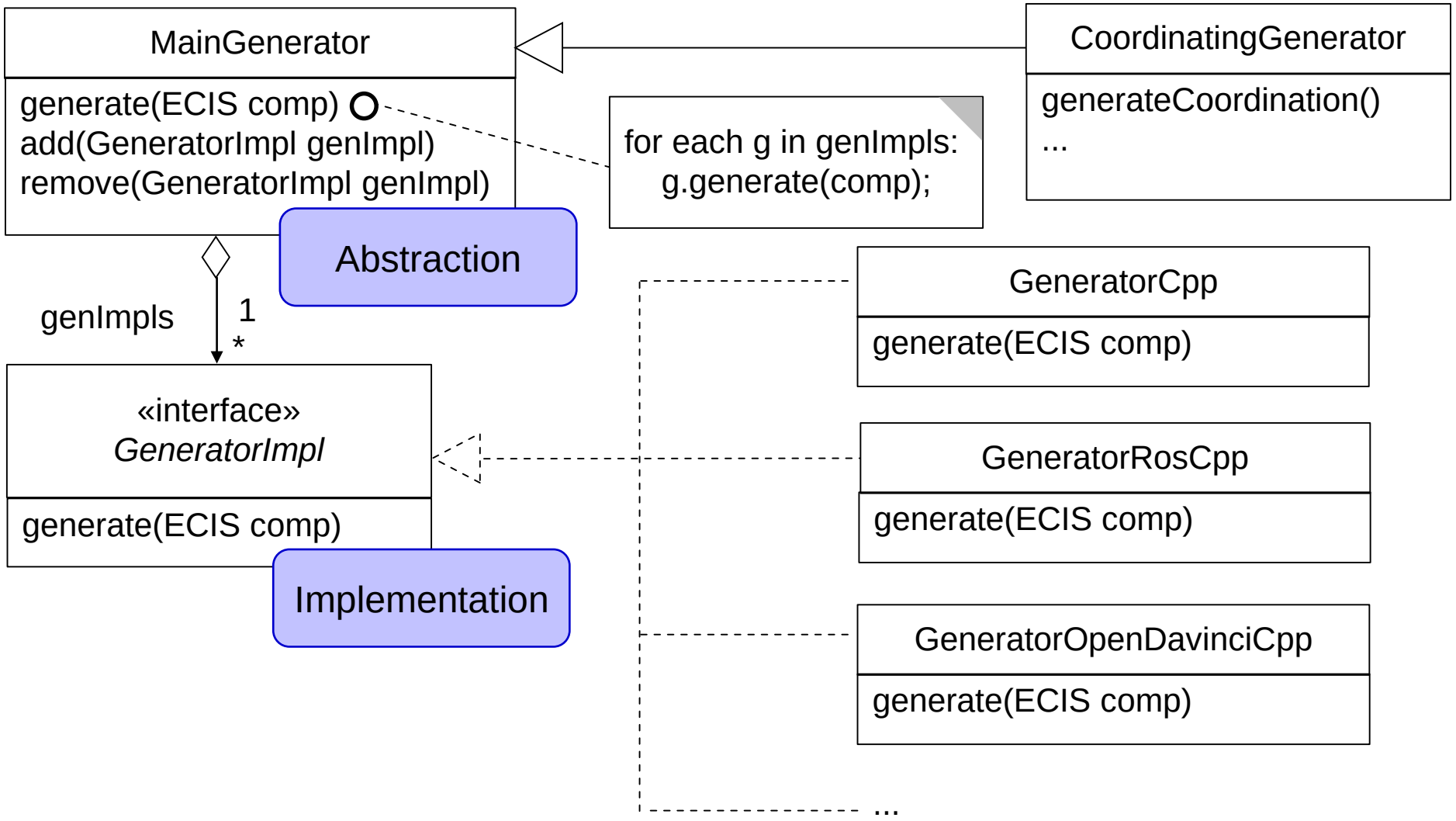
- Solve the eigenproblem
  - $Av = ev$
- For our example:
  - $v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$
  - $v_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$
  - $v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$



- 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

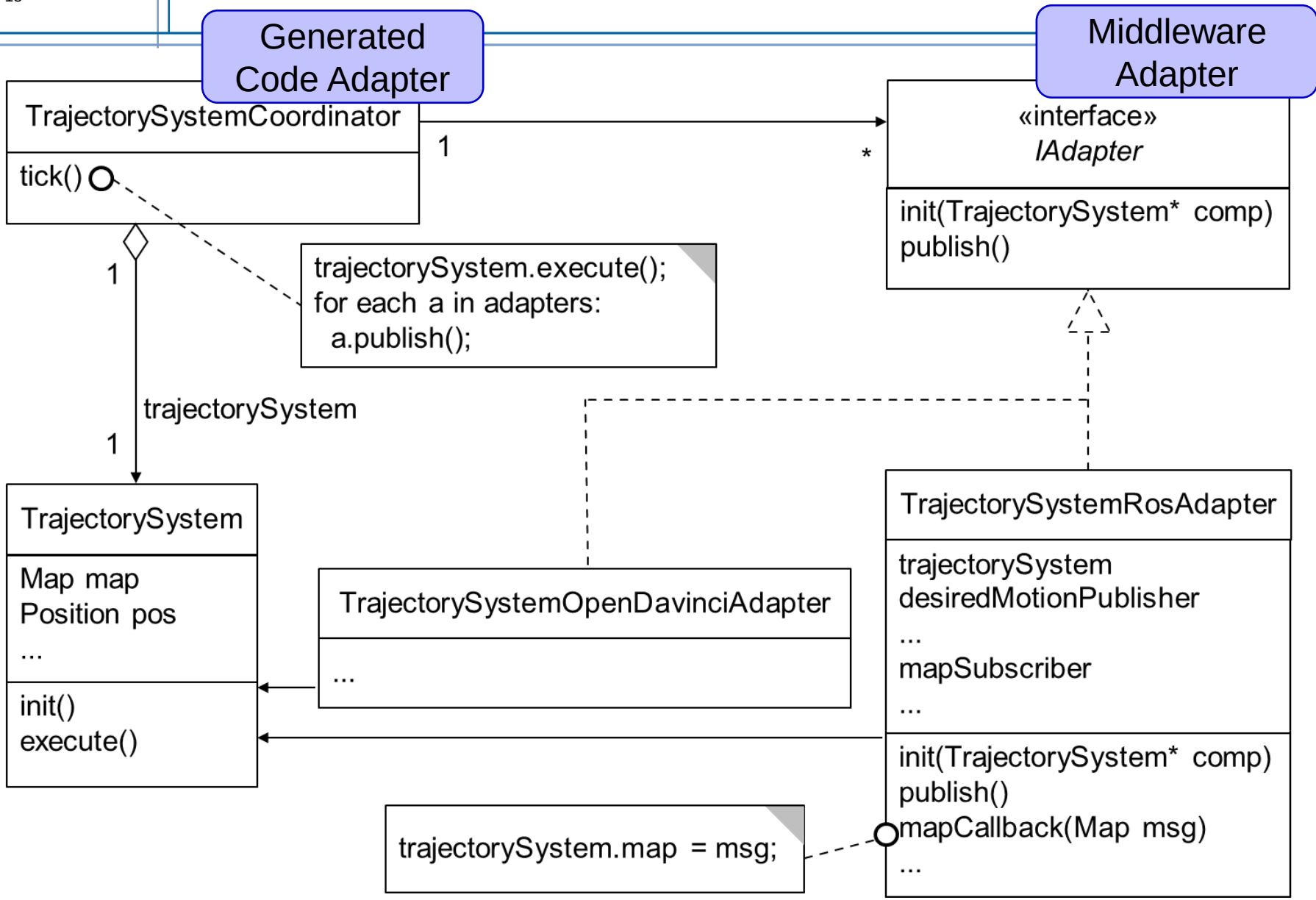
Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

*-Bridge Generator Composition



Evgeny Kusmenko
Chair of Software Engineering
RWTH Aachen

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

```
1  #include "TrajectorSystem.h"
2  [...] /* other includes*/
3  class TrajectorySystemRosAdapter : public IAdapter{
4    TrajectorySystem* trajectorySystem;
5    ros::Subscriber mapSubscriber;
6    ros::Publisher desiredMotionPublisher;
7
8    public:
9    void init(TrajectorySystem* comp){
10     trajectorySystem = comp;
11     /*init publishers, subscribers and start ROS thread*/
12     [...]
13   }
14   void publish(){
15     struct_msgs::Map tmpMsg =
                 msgFromStructMap(trajectorySystem->desiredTraj);
16     desiredMotionPublisher.publish(tmpMsg);
17   }
18
19   void mapCallback(struct_msgs::Map& msg){
20     trajectorySystem->map = structFromMsgMap(msg);
21   } [...] };
```

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**