

Multi-Target Code Generation

12.06.2018

Alexander Hellwig
Evgeny Kusmenko

Software Engineering
RWTH Aachen

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



Simulatoren

TORCS / OpenDaVINCI



Gazebo / ROS



Simulink



How to get models into a simulator?

VDrift / OpenDaVINCI



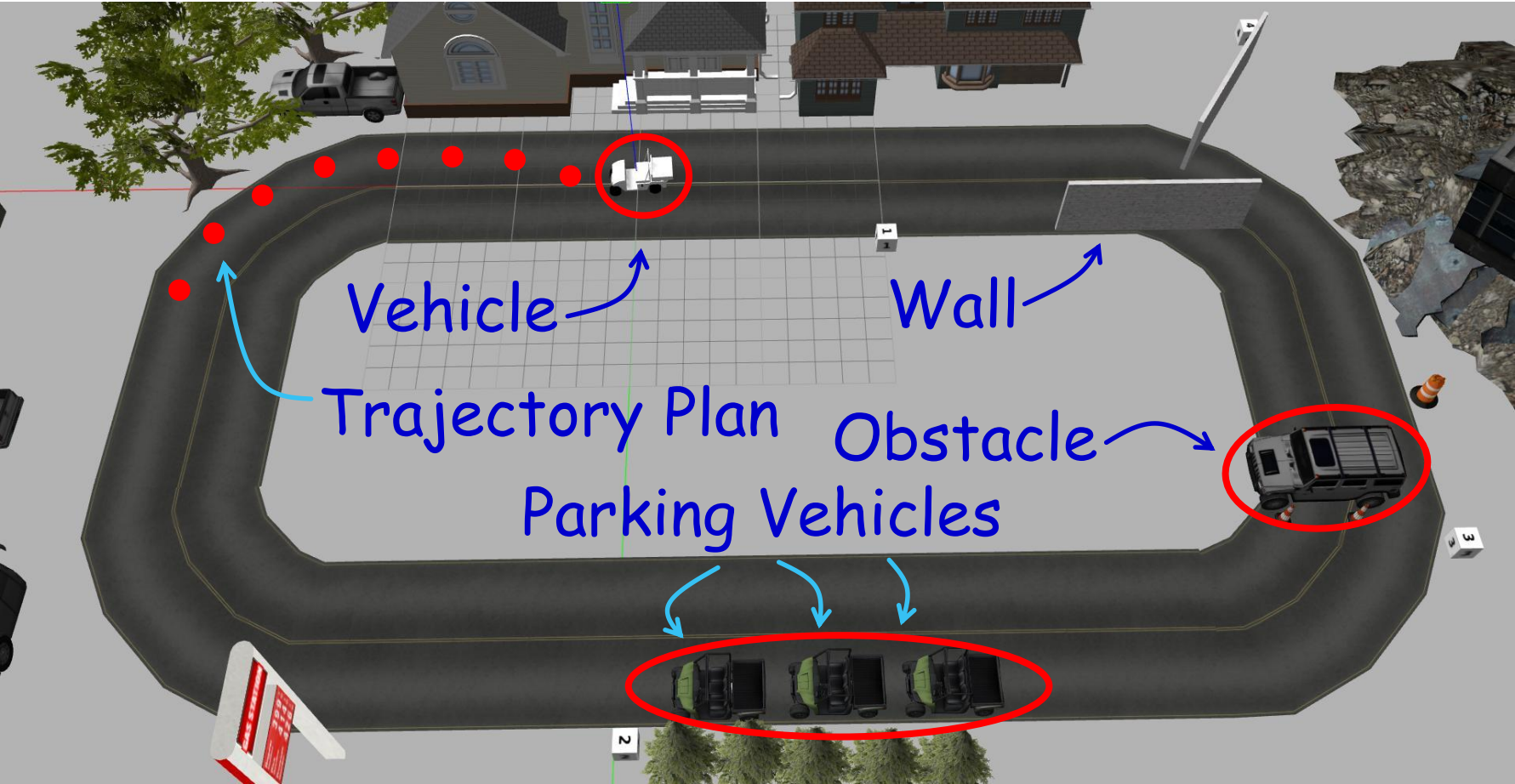
MontiSim



Veins



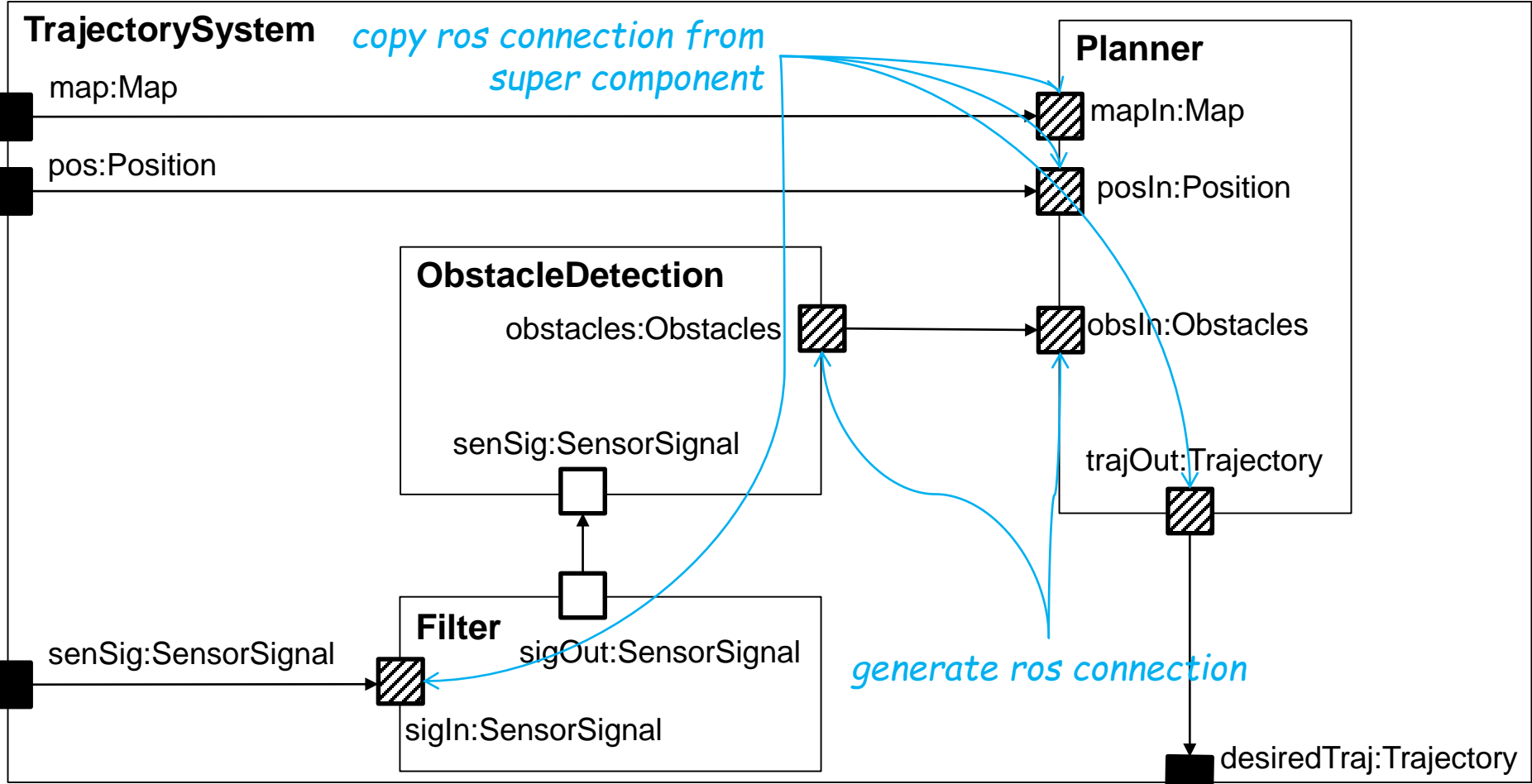
Gazebo Self-Driving Vehicle Example






EmbeddedMontiArc Model

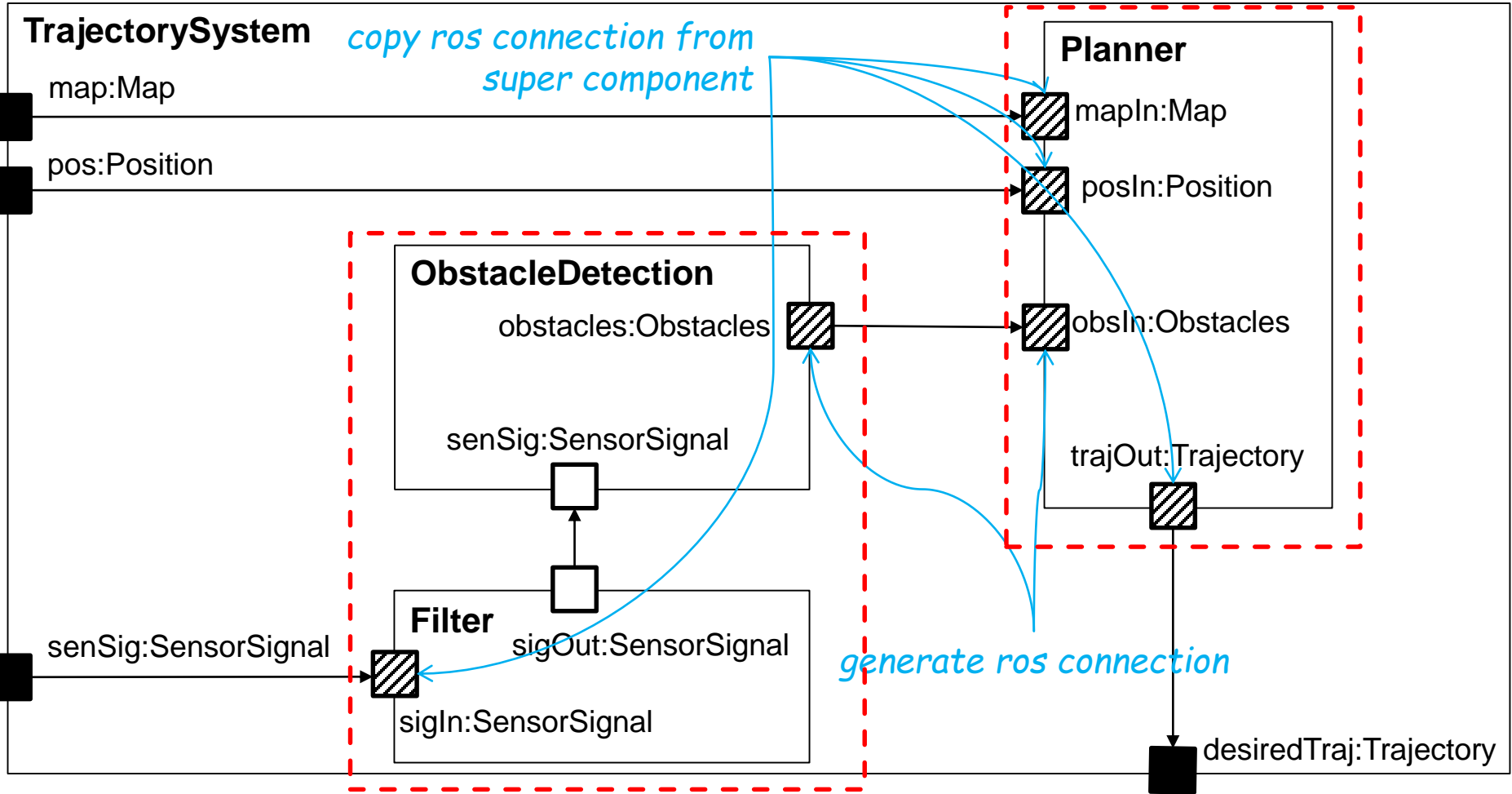
```
1 component TrajectorySystem{
2     direction      type      name
3     ports in      Map      map,
4         in Position pos,
5         in SensorSignal senSig,
6         out Trajectory desiredTraj;
7
8     instances of subcomponents defined in other artifacts
9
10    instance ObstacleDetection obsDetection;
11    instance Filter filter;
12    instance Planner planner;
13
14    source port      target port
15    connect map      -> planner.mapIn;
16    connect pos      -> planner.posIn;
17    connect planner.trajOut -> desiredTraj;
18    connect obsDetection.obstacles -> planner.obsIn;
19    /* other connections */}
```

EMAM Diagram



EMAM Diagram + Port Tagging

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



Tagging Model

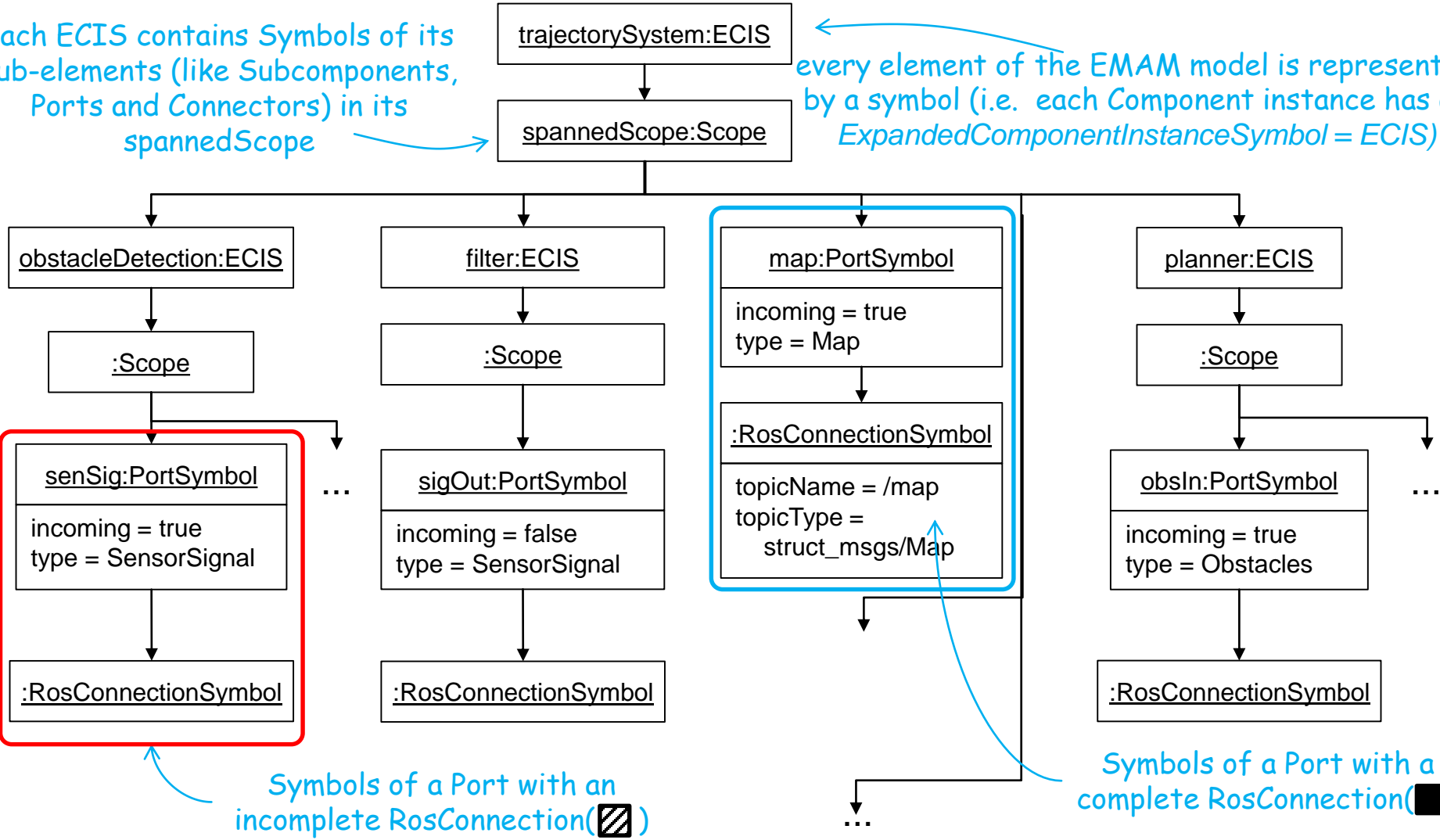
```
1 tags RosTags{ tagged symbol                tag type
2 tag trajectorySystem.map with RosConnection = tag value
    {topic= (name=/map, type=struct_msgs/Map) };
    RosConnection with complete information(■)
3 tag trajectorySystem.desiredTraj with RosConnection =
    {topic= (name=/desiredMotion,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

1 struct Position{
   type      min value  step size  max value  name
2  Q        ( -90° : 0.000001° : 90° ) longitude;
3  Q        ( -180° : 0.000001° : 180° ) latitude;
4  Q        ( -10km : 10cm : 10km ) altitude;}
```


Symbol Table

each ECIS contains Symbols of its sub-elements (like Subcomponents, Ports and Connectors) in its `spannedScope`

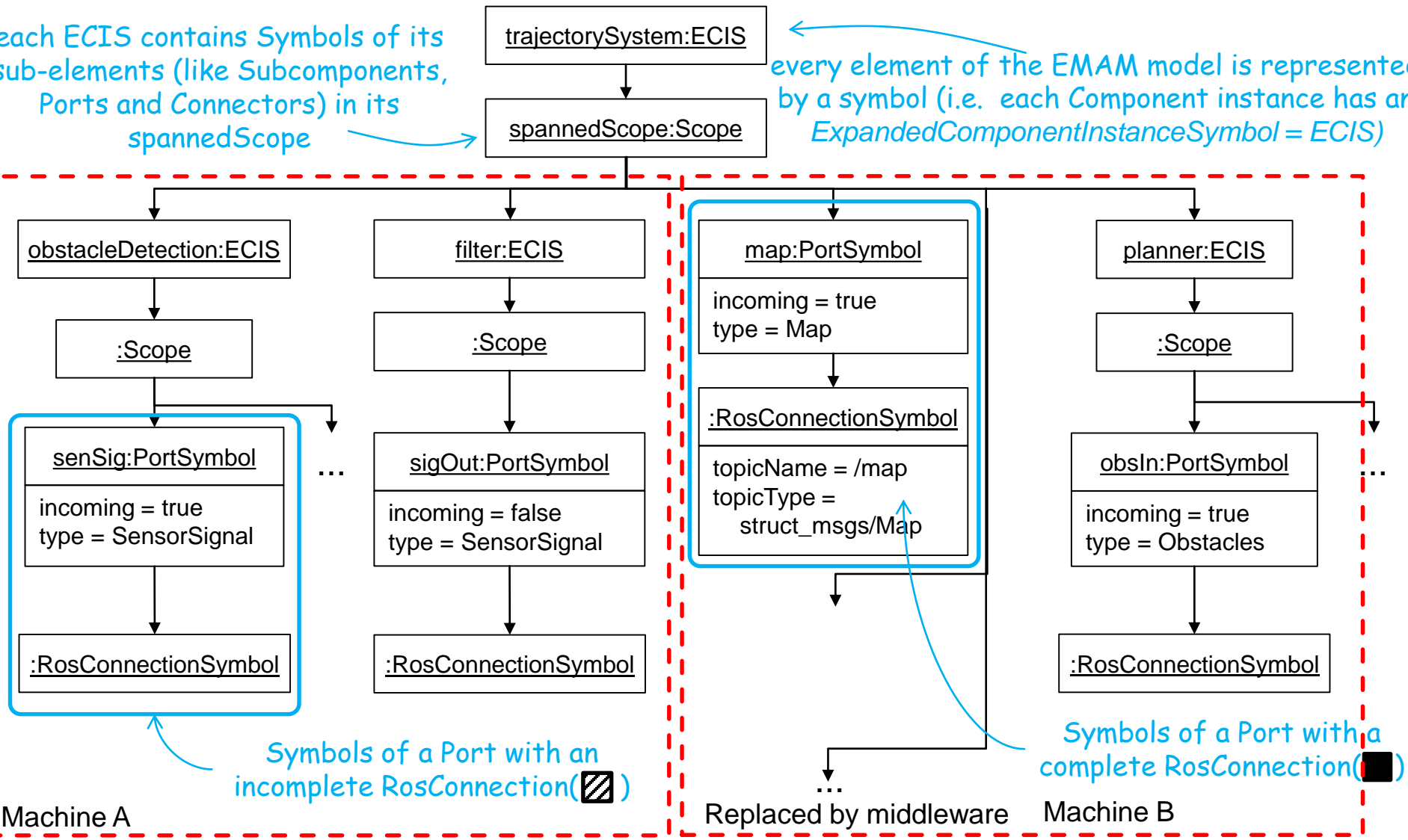
every element of the EMAM model is represented by a symbol (i.e. each Component instance has an `ExpandedComponentInstanceSymbol = ECIS`)



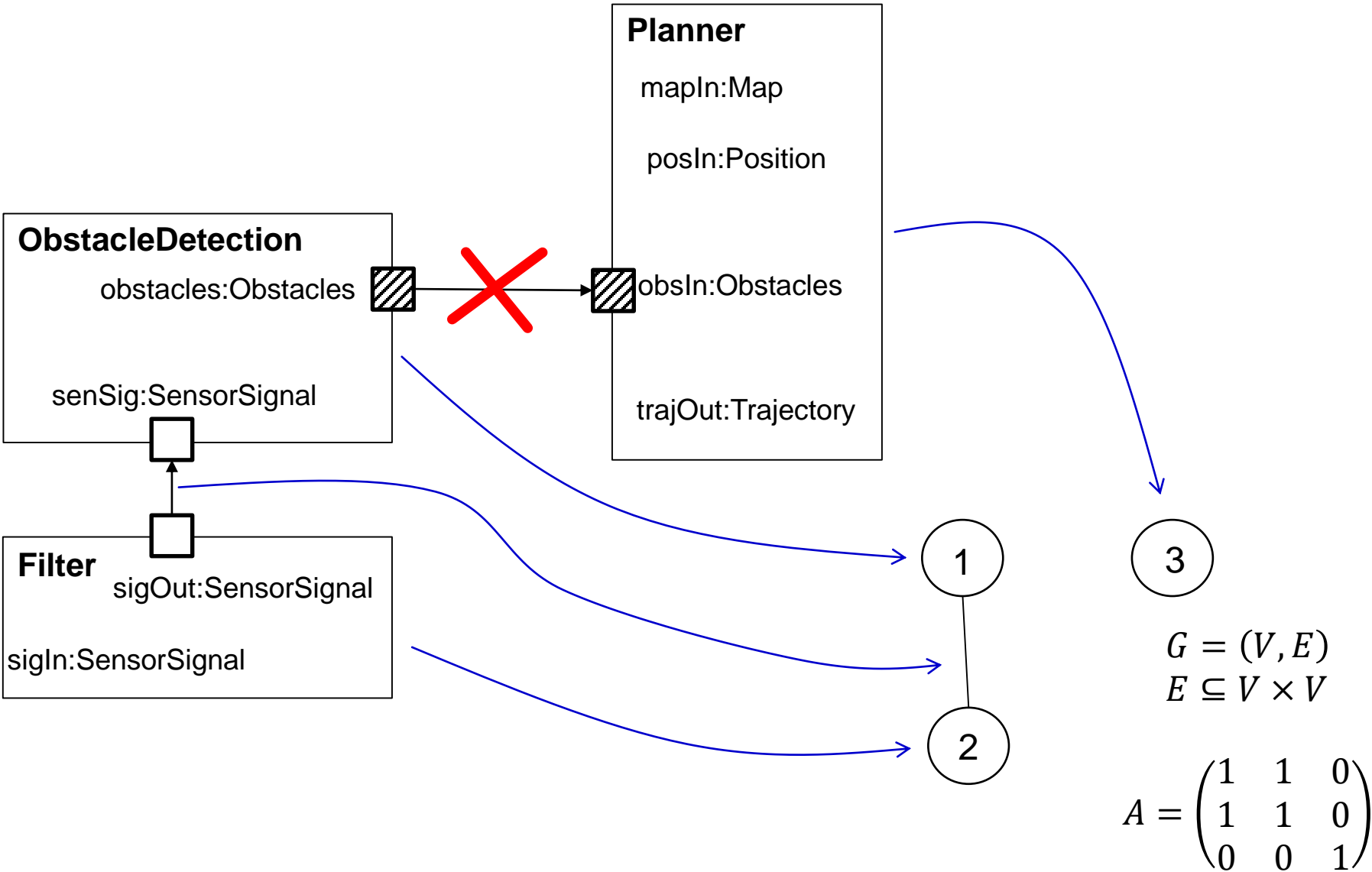
Clustered Symbol Table

each ECIS contains Symbols of its sub-elements (like Subcomponents, Ports and Connectors) in its spannedScope

every element of the EMAM model is represented by a symbol (i.e. each Component instance has an *ExpandedComponentInstanceSymbol = ECIS*)



Exkurs Machine Learning in SE



Spectral Analysis

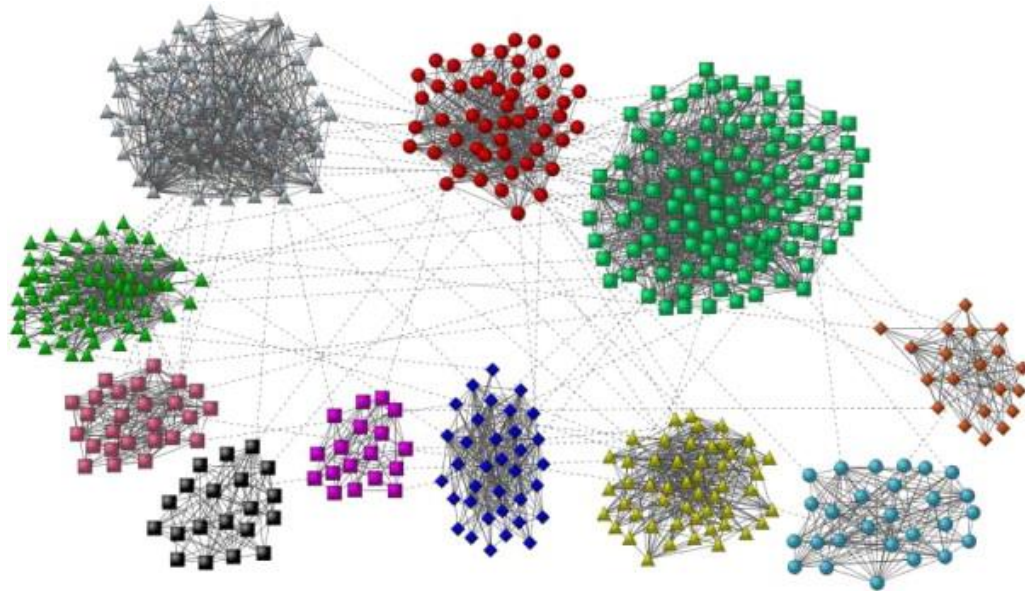
- Solve the eigenproblem
 - $Av = ev$

- For our example:

- $v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

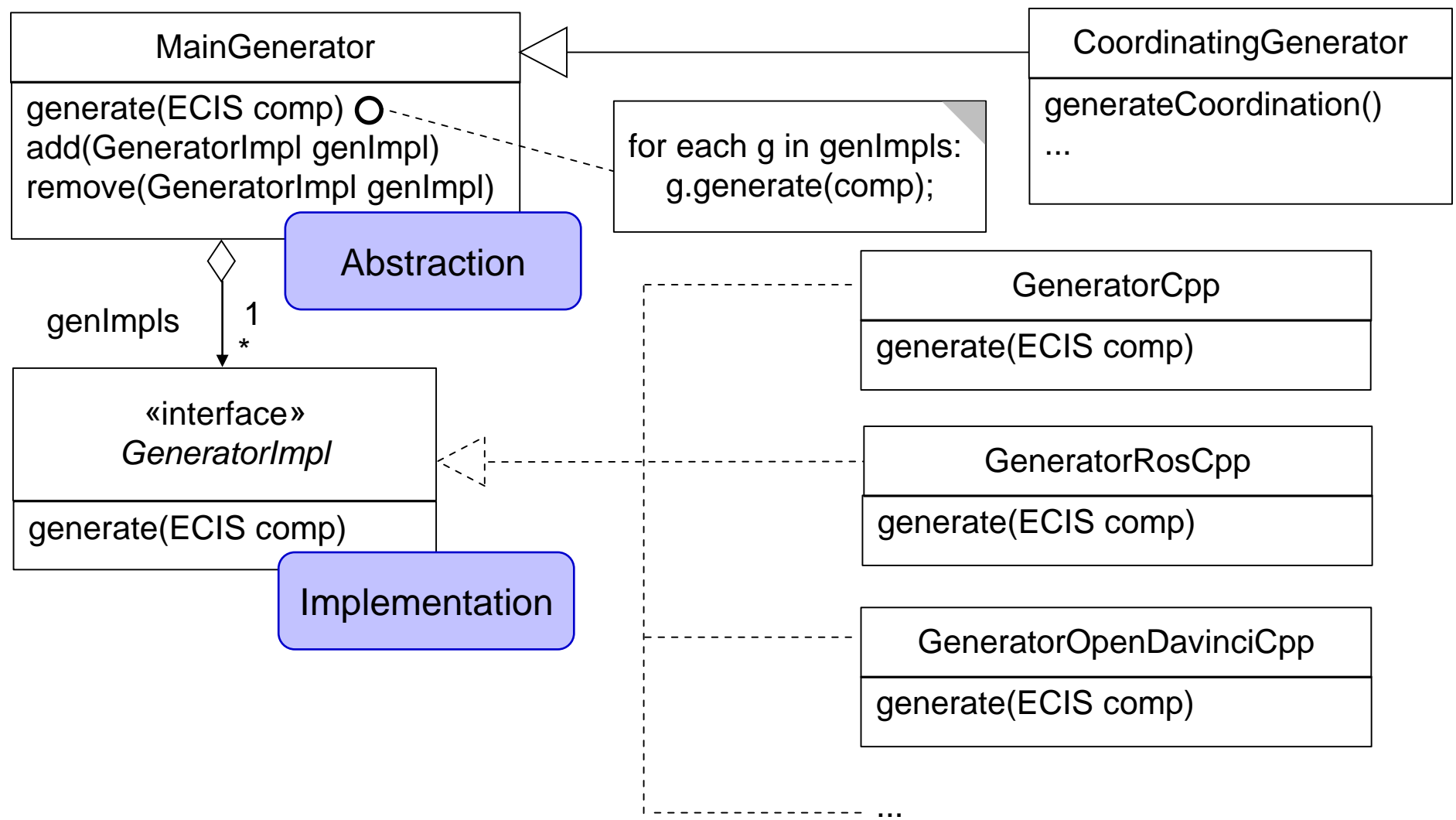
- $v_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$

- $v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

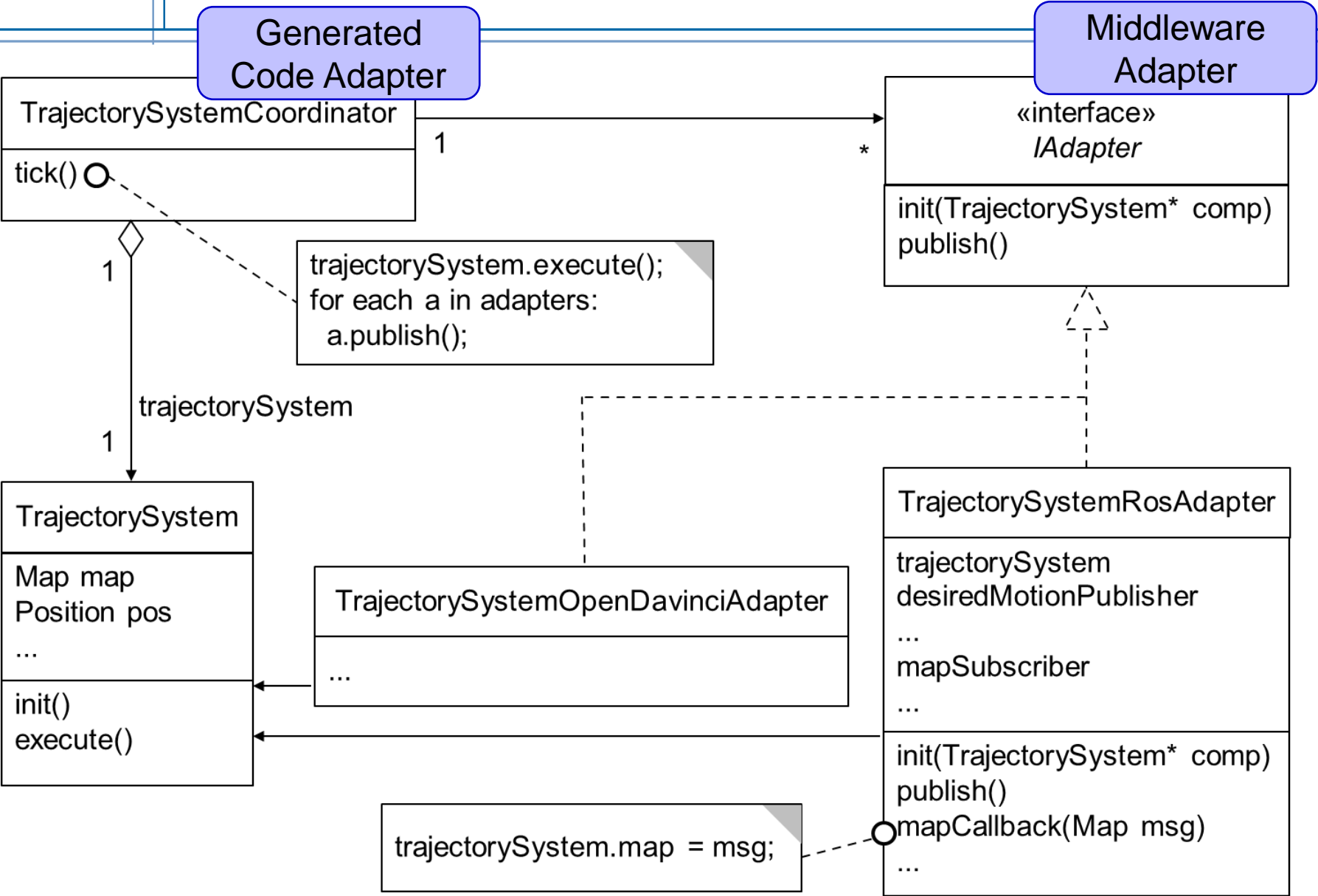


- Non-zero entries appearing in the same ev designate clusters which need to be deployed on the same machine (no middleware communication)
- Potential: Spectral Methods can help us distribute code to a predefined number of machines
- We use the same technique for error pattern detection for BMW

*-Bridge Generator Composition



Generated System incl. Adapter



Generated Artifacts

Generator	Generated files
CoordinatingGenerator	<ul style="list-style-type: none">└─ CMakeLists.txt└─ coordinator<ul style="list-style-type: none">└─ CMakeLists.txt└─ TrajectorySystemCoordinator.cpp└─ IAdapter.h
GeneratorCpp	<ul style="list-style-type: none">└─ cpp<ul style="list-style-type: none">└─ CMakeLists.txt└─ TrajectorySystem.h
GeneratorOpenDavinciCpp	<ul style="list-style-type: none">└─ opendavinci<ul style="list-style-type: none">└─ CMakeLists.txt└─ TrajectorySystem<ul style="list-style-type: none">└─ OpenDavinciAdapter.h
GeneratorRosCpp	<ul style="list-style-type: none">└─ roscpp<ul style="list-style-type: none">└─ CMakeLists.txt└─ TrajectorySystemAdapter.h

Generated Adapter

```
1 #include "TrajectorySystem.h"
2 [...] /* other includes */
3 class TrajectorySystemRosAdapter : public IAdapter{
4     TrajectorySystem* trajectorySystem; ← instance shared between
5     ros::Subscriber mapSubscriber;      the coordinator and other
6     ros::Publisher desiredMotionPublisher; adapters
7
8     public:
9     void init(TrajectorySystem* comp) { ← called once by
10         trajectorySystem = comp;        TrajectorySystemCoordinator
11         /*init publishers, subscribers and start ROS thread*/
12         [...]
13     }
14     void publish() { ← called in a defined interval by
15         struct_msgs::Map tmpMsg =        TrajectorySystemCoordinator
16             msgFromStructMap(trajectorySystem->desiredTraj);
17         desiredMotionPublisher.publish(tmpMsg);
18     }
19     void mapCallback(struct_msgs::Map& msg) { ← called by ROS every time a message is
20         trajectorySystem->map = structFromMsgMap(msg);    published on the topic /map
21     } [...] };
```