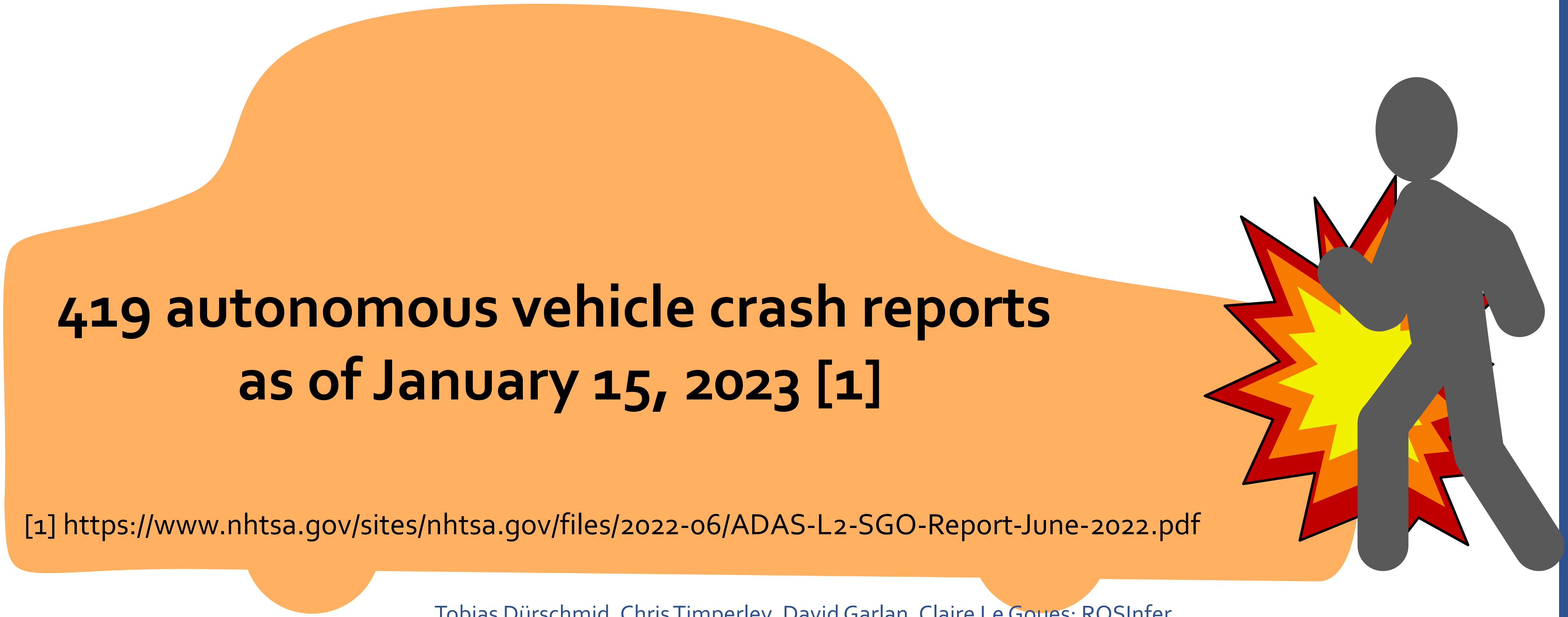
ROSINFER: STATICALLY INFERRING BEHAVIORAL COMPONENT MODELS FOR ROS-BASED ROBOTICS SYSTEM

Tobias Dürschmid, Christopher S. Timperley, David Garlan, Claire Le Goues

Carnegie Mellon University

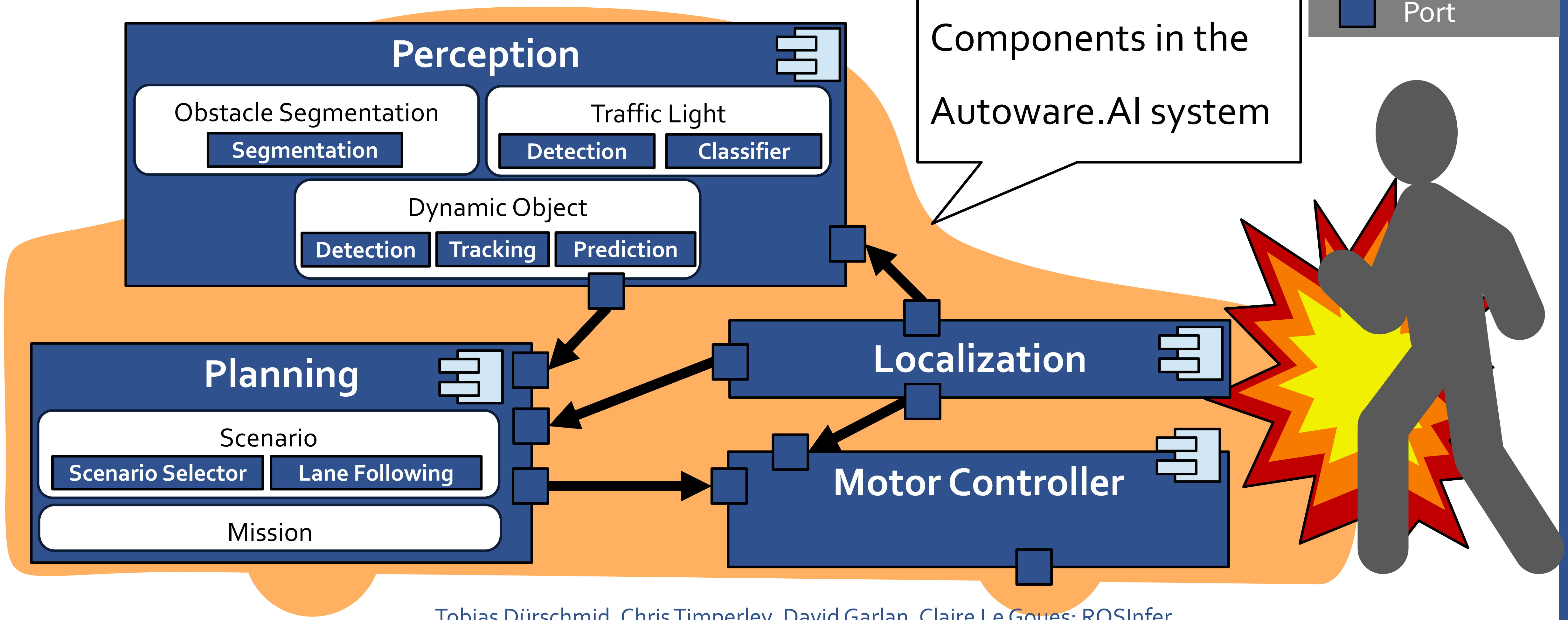
Robotics Systems are Safety-Critical



419 autonomous vehicle crash reports as of January 15, 2023 [1]

[1] https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf

Robotics Systems are Complex Component-based Systems



In Total: **230**

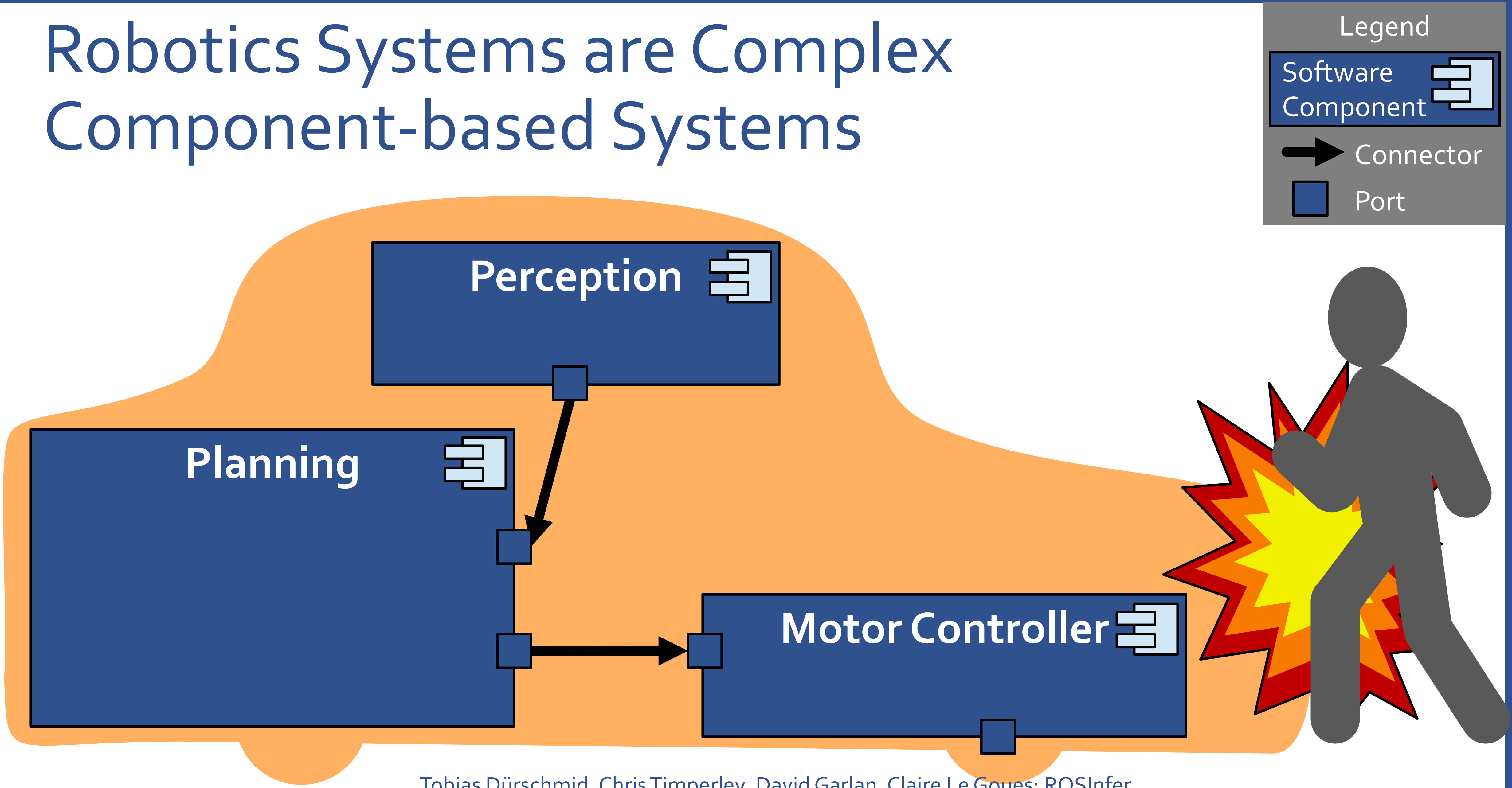
Components in the

Autoware.AI system

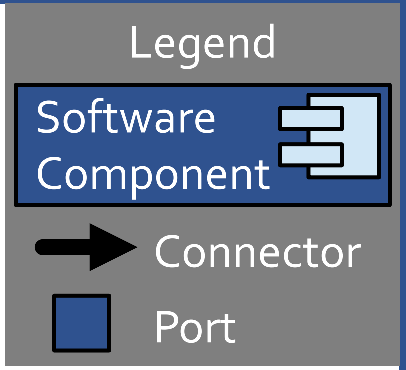
Legend

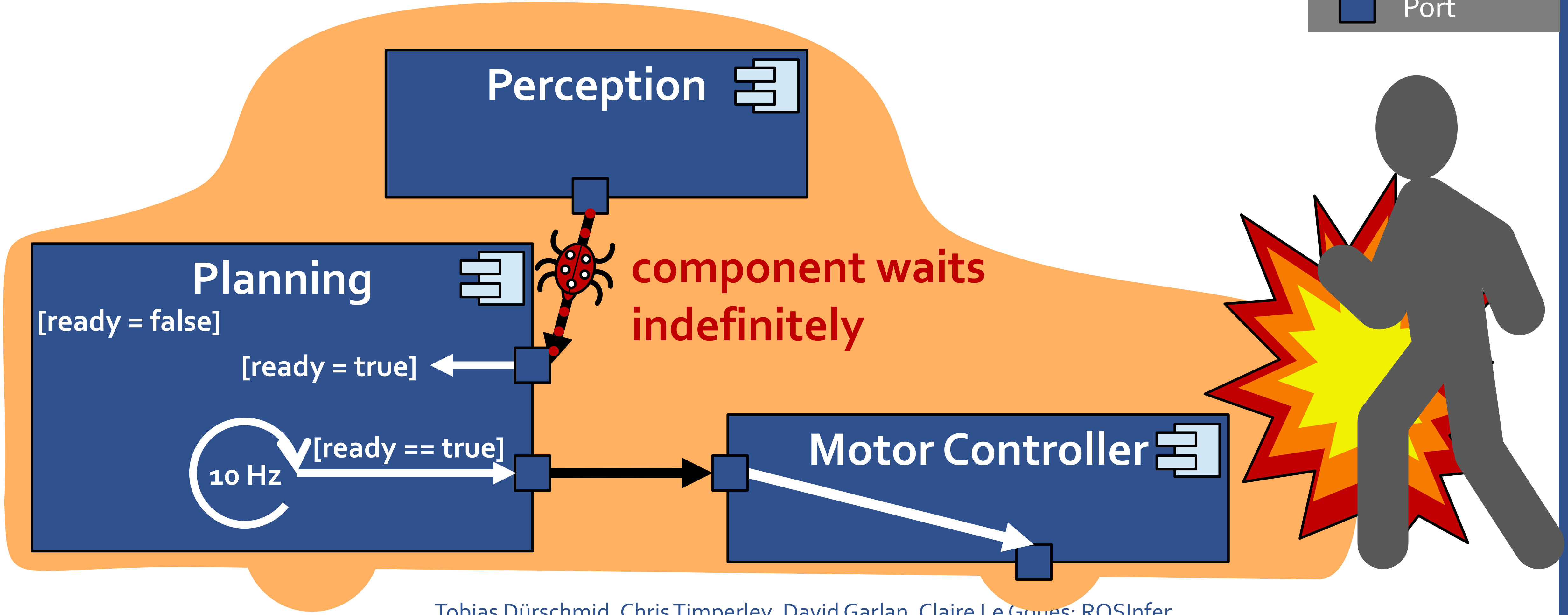
- Software Component
- Connector
- Port

Robotics Systems are Complex Component-based Systems



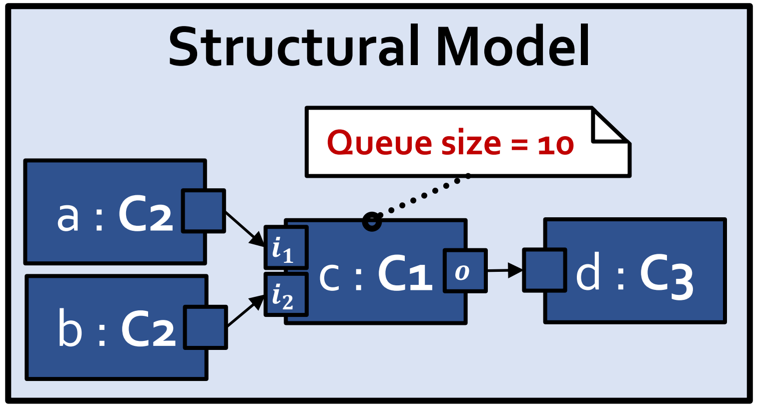
A Class of Very-hard-to-find Bugs Results from Incorrect Component Composition



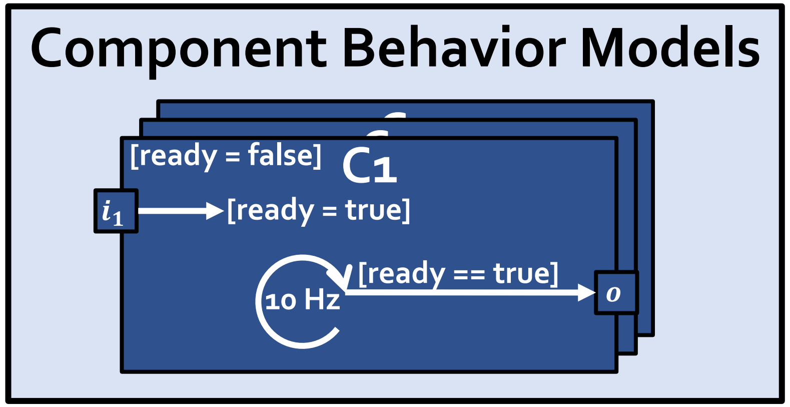


component waits indefinitely

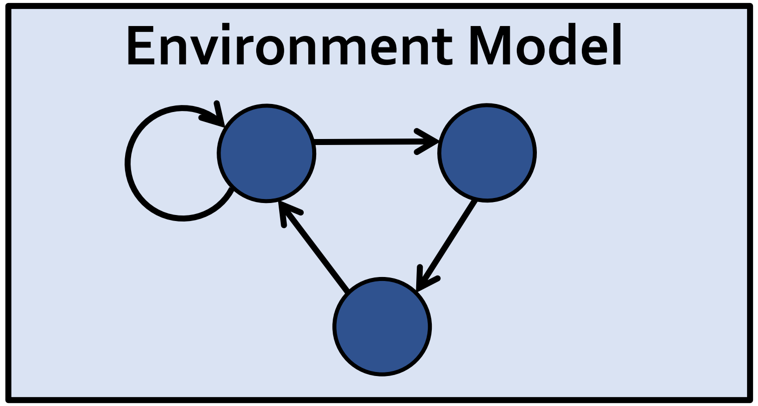
Good News: Model-Based Analysis can Find Bugs


Structural Model

+


Component Behavior Models

+


Environment Model

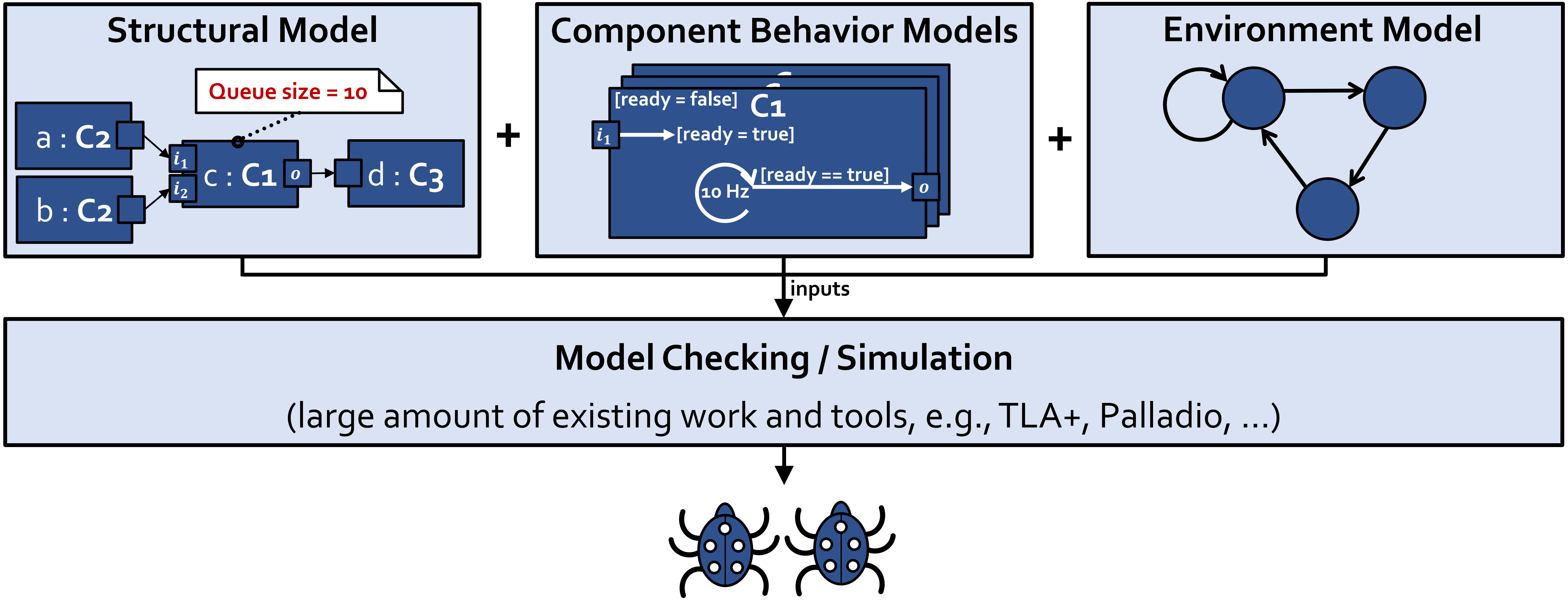
Models the **interface** of **components** (i.e., port) and **connectors**

Models the **states and state transitions** of components, their **input-output relationship**

Models **types of inputs** from the environment and how the environment **reacts** to actions of the system

This is what we mean with "behavior" throughout the talk

Good News: Model-Based Analysis can Find Bugs



Structural Model

+

Component Behavior Models

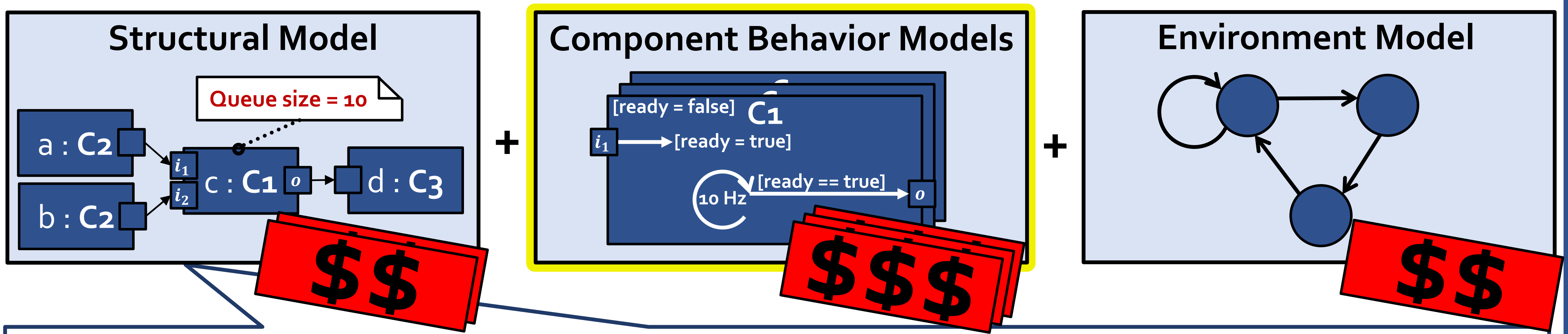
+

Environment Model

Model Checking / Simulation

(large amount of existing work and tools, e.g., TLA+, Palladio, ...)

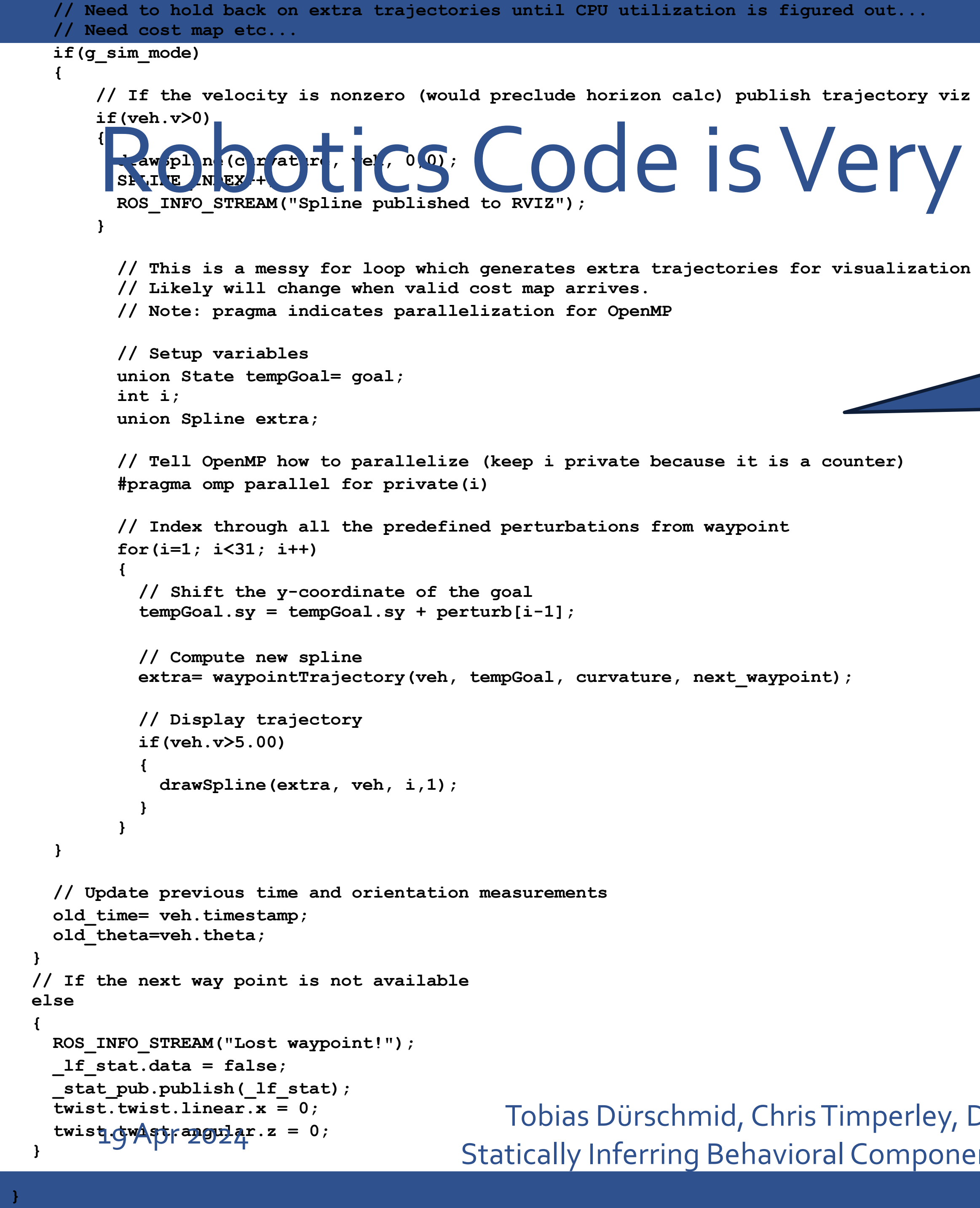
Bad News: Models are Expensive To Create



Our Previous Work **ROSDiscover**[1] infers these models, but cannot find **behavioral architecture composition bugs** (e.g., components waiting indefinitely for a message, deadlocks due to incompatibles component state, message loss, ignored inputs, …)

[1] C. S. Timperley, T. Dürschmid, B. Schmerl, D. Garlan and C. Le Goues, "ROSDiscover: Statically Detecting Run-Time Architecture Misconfigurations in Robotics Systems," *(ICSA 2022)*

Robotics Code is Very Complex

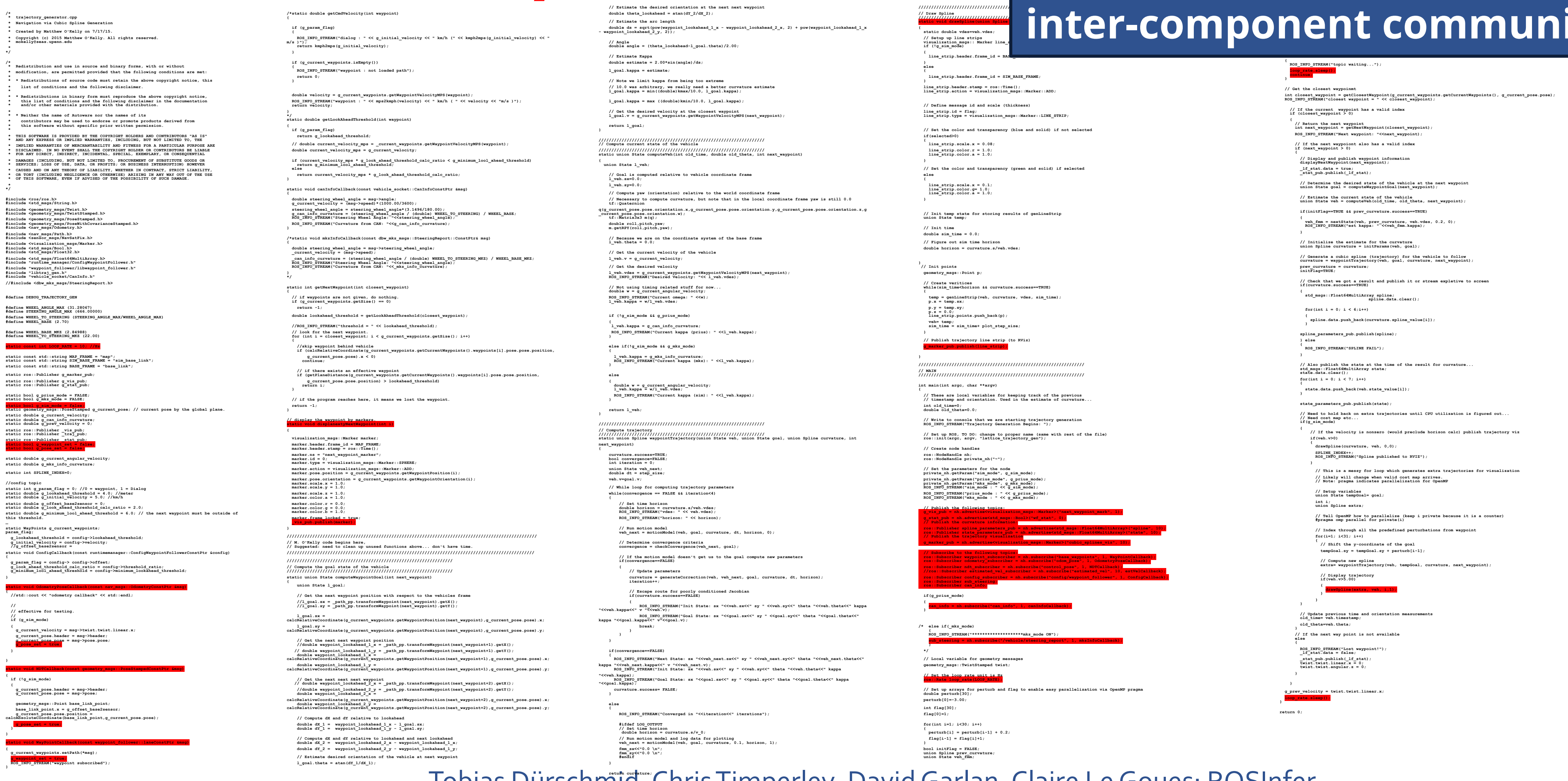


```
  // Need to hold back on extra trajectories until CPU utilization is figured out...
  // Need cost map etc...
  if(g_sim_mode)
  {
    // If the velocity is nonzero (would preclude horizon calc) publish trajectory viz
    if(veh.v>0)
    {
     drawSpline(curvature, veh, 0,0);
     SPLINE_INDEX++;
     ROS_INFO_STREAM("Spline published to RVIZ");
    }

     // This is a messy for loop which generates extra trajectories for visualization
     // Likely will change when valid cost map arrives.
     // Note: pragma indicates parallelization for OpenMP

     // Setup variables
     union State tempGoal= goal;
     int i;
     union Spline extra;

     // Tell OpenMP how to parallelize (keep i private because it is a counter)
     #pragma omp parallel for private(i)

     // Index through all the predefined perturbations from waypoint
     for(i=1; i<31; i++)
     {
      // Shift the y-coordinate of the goal
      tempGoal.sy = tempGoal.sy + perturb[i-1];

      // Compute new spline
      extra= waypointTrajectory(veh, tempGoal, curvature, next_waypoint);

      // Display trajectory
      if(veh.v>5.00)
      {
       drawSpline(extra, veh, i,1);
      }
     }
  }

  // Update previous time and orientation measurements
  old_time= veh.timestamp;
  old_theta=veh.theta;
 }
 // If the next way point is not available
 else
 {
  ROS_INFO_STREAM("Lost waypoint!");
  _lf_stat.data = false;
  _stat_pub.publish(_lf_stat);
  twist.twist.linear.x = 0;
  twist.twist.angular.z = 0;
 }
}
```

Source code of the `lattice_trajectory_gen` component of the system **`Autoware.AI`**

Only A **Small Portion** of Code is **Architecturally-Relevant**

defines structure and behavior for inter-component communication



Problem Statement

We have strong reason to believe that the approach generalizes to other component frameworks

How to automatically infer behavioral models of components **written for the Robot Operating System (ROS)?**

Popular framework for component-based robot software

Used by Amazon, Bosch, Siemens, and many other companies

Why is static architecture recovery hard?

Architecture-defining code is **scattered across the entire system**.

In theory, **any code statement** could impact the architecture

Observation: ROS Systems Implement Architectural Behavior using APIs & Idioms

```
int main(int argc, char** argv)
{
  ros::Subscriber sub = nh.subscribe("t_sub", receive_initial);
  ros::Publisher  pub = nh.advertise("t_pub");
  const int local_LOOP_RATE = 10;
  ros::Rate loop_rate(local_LOOP_RATE);
  while (ros::ok())
  {
    if (!ready)
    {
      loop_rate.sleep();
      continue;
    }
    pub.publish(msg);
    loop_rate.sleep();
  }
  return 0;
}
```

Subscriber Port

Publisher Port

Periodic Rate

Periodic Loop

State Condition

Periodic Sleep

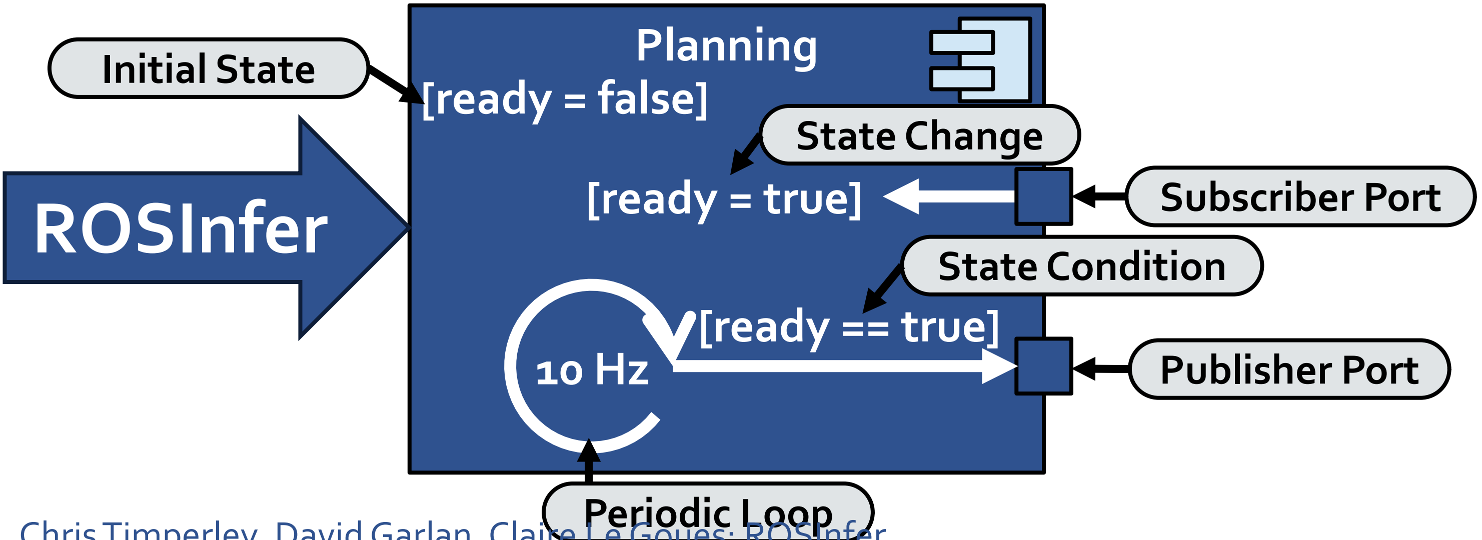
Message Output

```
bool ready = false;

void receive_initial(const Message msg)
{
  ready = true;
}
```

Initial State

Subscriber Callback

State Change



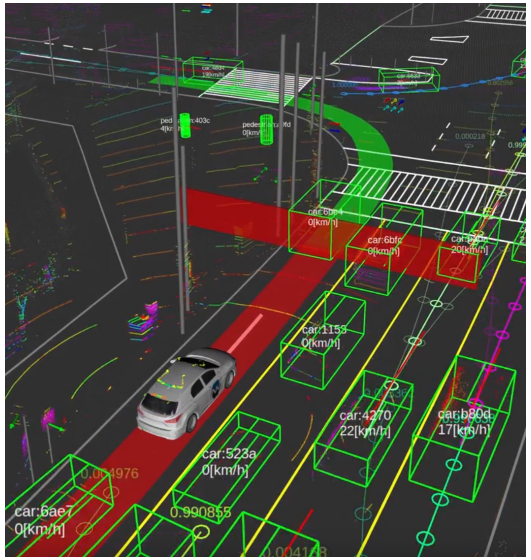
We Evaluated ROSInfer on 4 Real-World Open Source Systems

Turtlebot

(108 Components)

Fetch

(94 Components)



Autoware

(230 Components)

AutoRally

(26 Components)

We Built a Novel Data Set of Component Behaviors In ROS Component

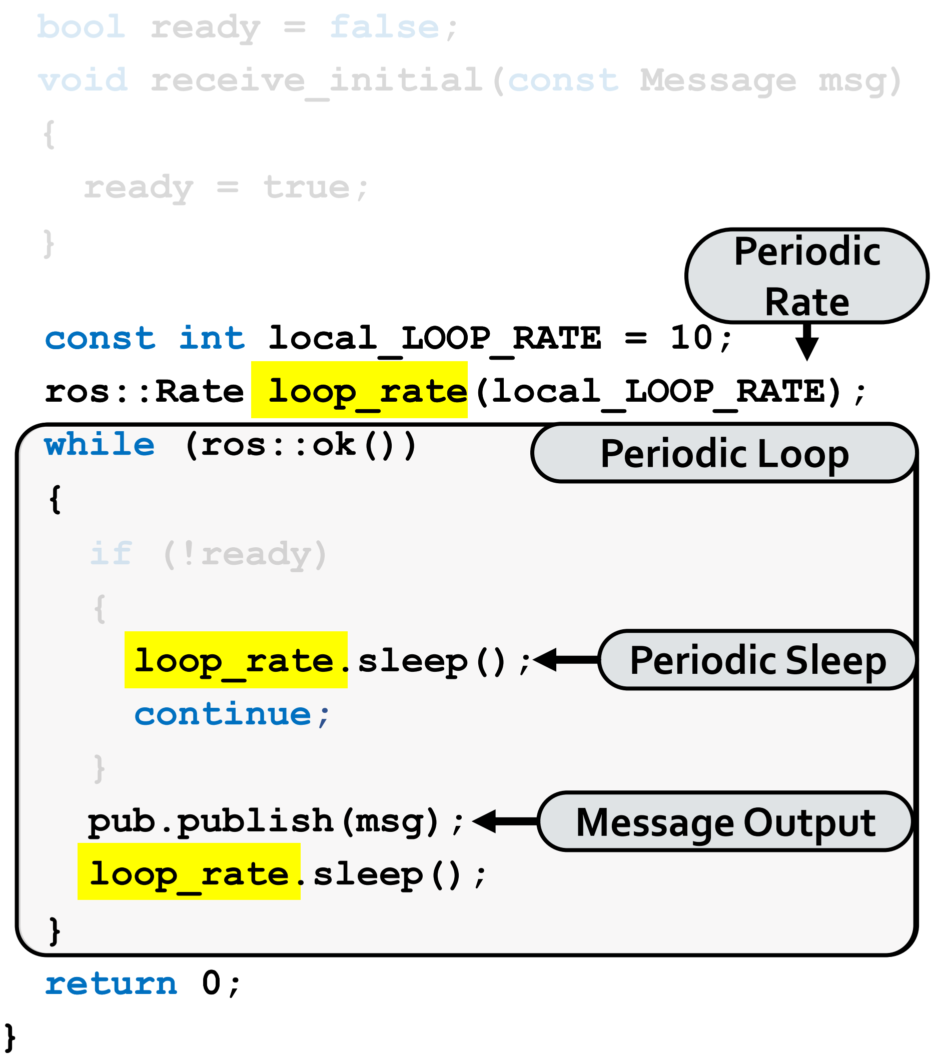
- There are **no available ground truth models** for comparison
- Two authors **manually inferred** ground truth models from the code base of 155 ROS components
- Data Set is available on GitHub!

API-Call-Guided Static Recovery has a High Precision and Recall

Method: Compare to 149 manually inferred models

Results for Periodic Behavior:

- Precision: **100.0 %**
- Recall: **92.9 %**



```
 bool ready = false;
 void receive_initial(const Message msg)
 {
   ready = true;
 }

 const int local_LOOP_RATE = 10;
 ros::Rate loop_rate(local_LOOP_RATE);
 while (ros::ok())
 {
   if (!ready)
   {
     loop_rate.sleep();
     continue;
   }
   pub.publish(msg);
   loop_rate.sleep();
 }
 return 0;
}
```

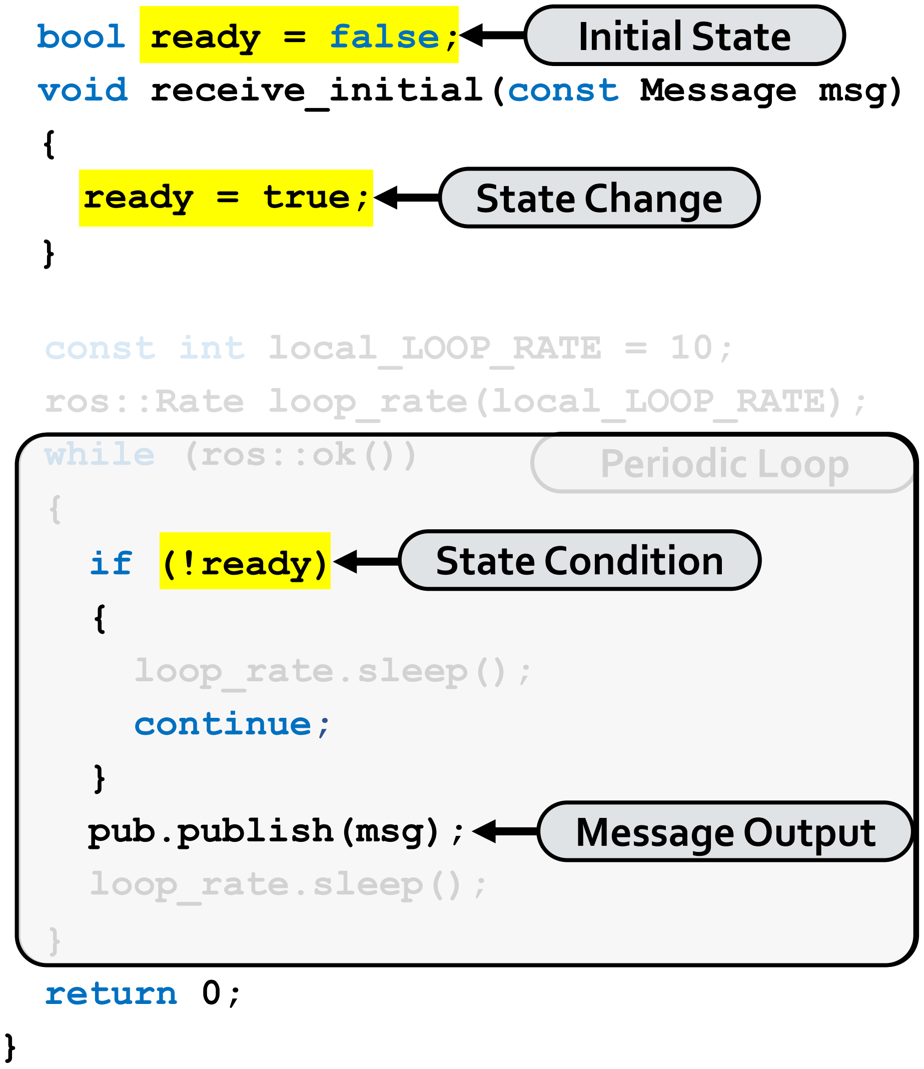
API-Call-Guided Static Recovery has a High Precision and Recall

Method: Compare to 149

manually inferred models

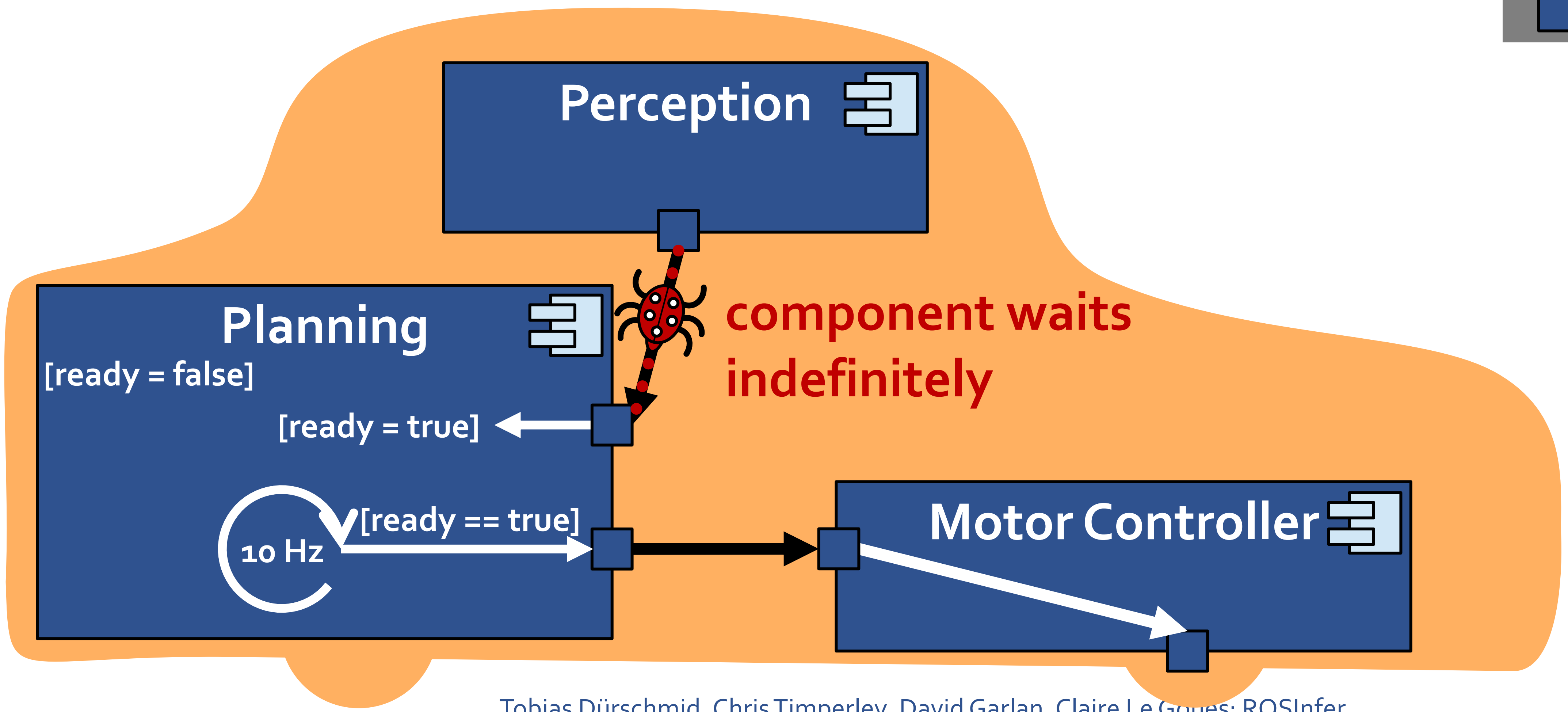
Results for State-based Behavior:

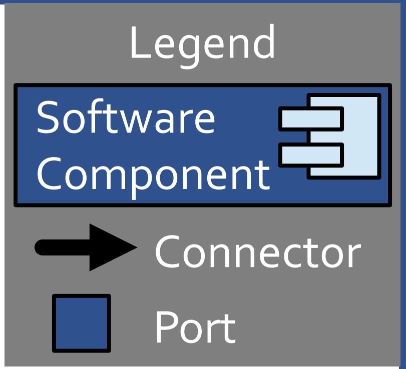
- Recall (State Vars): **70.8 %**
- Precision (State Vars): **75.6 %**
- Recall (State Changes): **62.2%**
- Precision (State Changes): **88.2 %**



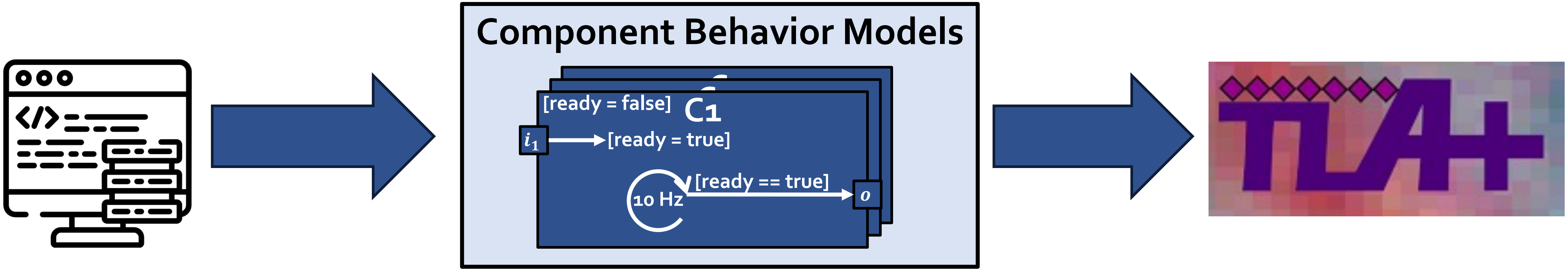
```
bool ready = false;
void receive_initial(const Message msg)
{
  ready = true;
}

 const int local_LOOP_RATE = 10;
 ros::Rate loop_rate(local_LOOP_RATE);
 while (ros::ok())
 {
   if (!ready)
   {
     loop_rate.sleep();
     continue;
   }
   pub.publish(msg);
   loop_rate.sleep();
 }
 return 0;
}
```

ROSInfer Finds Real-World Bugs, such as this Bug from Autoware.AI





ROSInfer Finds Real-World Bugs Via TLA+/PlusCal Generation



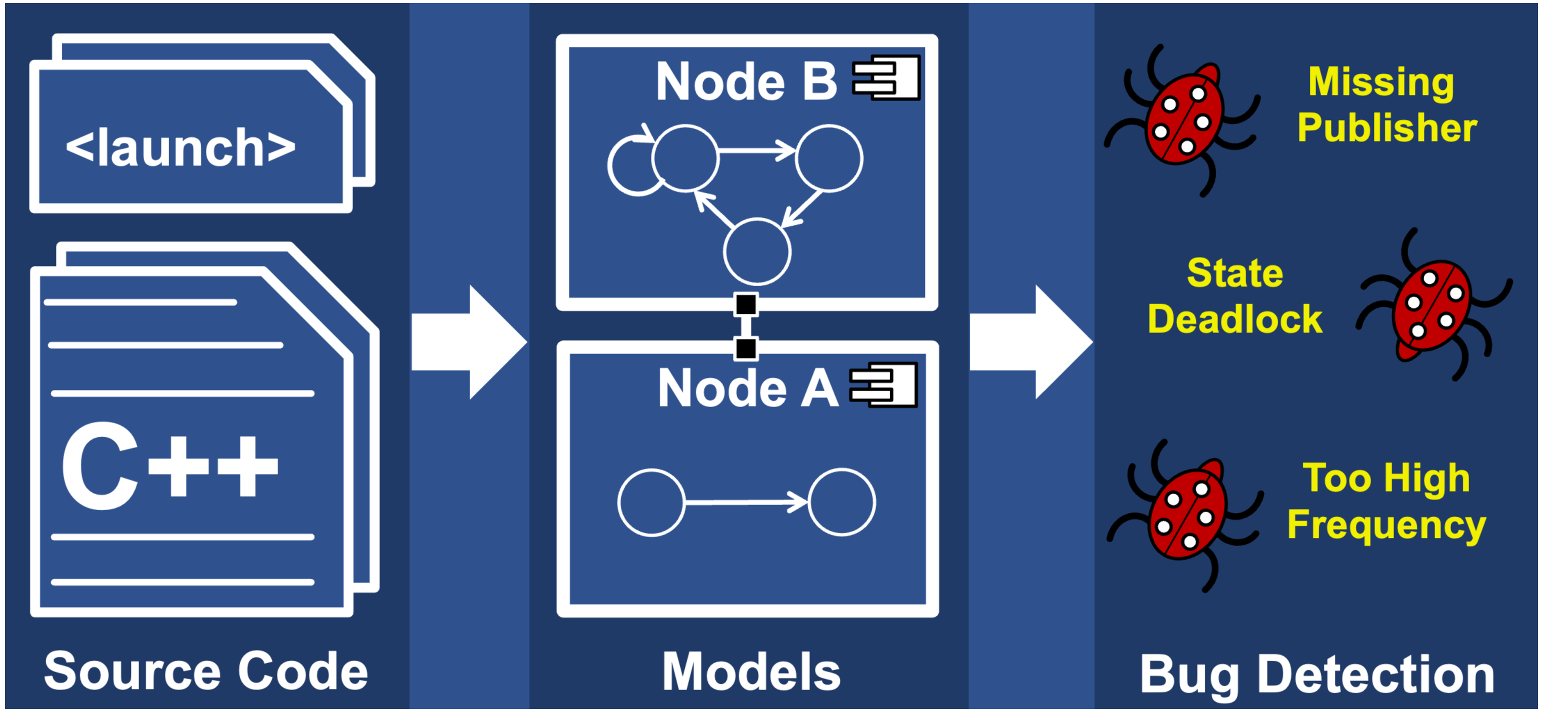
- **No** extensive **bug data set** of behavioral architecture composition bugs is **available**
- We ran ROSInfer on **three bugs from the ROSDiscover data set** that ROSDiscover could not find
- ROSInfer **found all three bugs**

Summary

Data Set &
Tool Available

- **Problem**: automatically infer behavioral models of ROS components

- **Solution**: ROSInfer – API-Guided Static Recovery of State Machines



I am on the SE **Teaching Job Market!**

I am happy to talk about the two courses that

I have designed and instructed at CMU:

Designing Large-scale Software Systems

Design Patterns & API Design