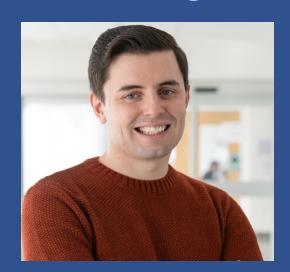
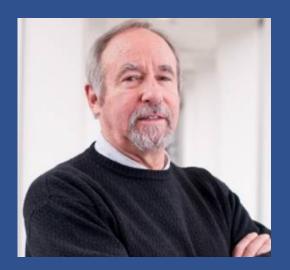
ROSINFER: STATICALLY INFERRING BEHAVIORAL COMPONENT MODELS FOR ROS-BASED ROBOTICS SYSTEM

<u>Tobias Dürschmid</u>, Christopher S. Timperley, David Garlan, Claire Le Goues

Carnegie Mellon University







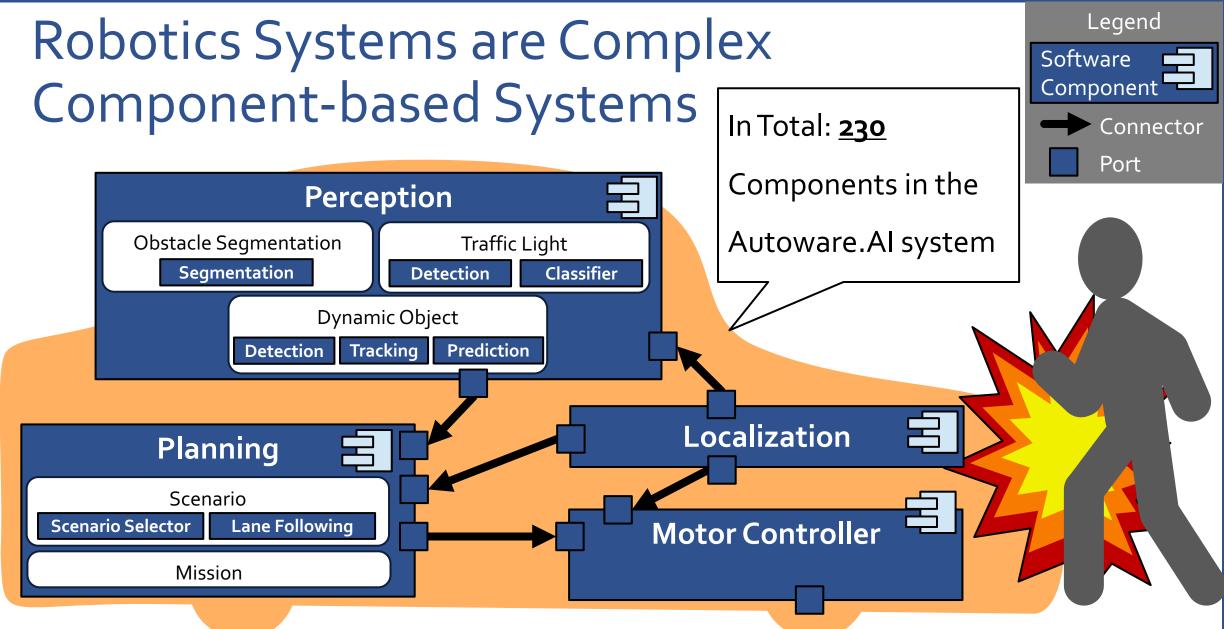


Robotics Systems are Safety-Critical

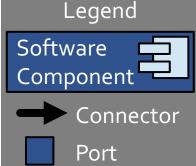
419 autonomous vehicle crash reports as of January 15, 2023 [1]

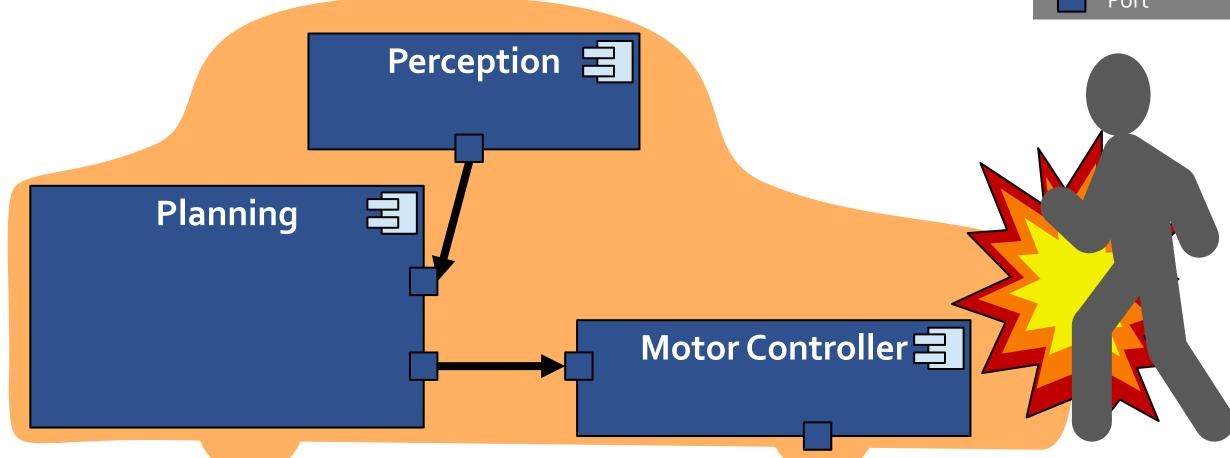
[1] https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf





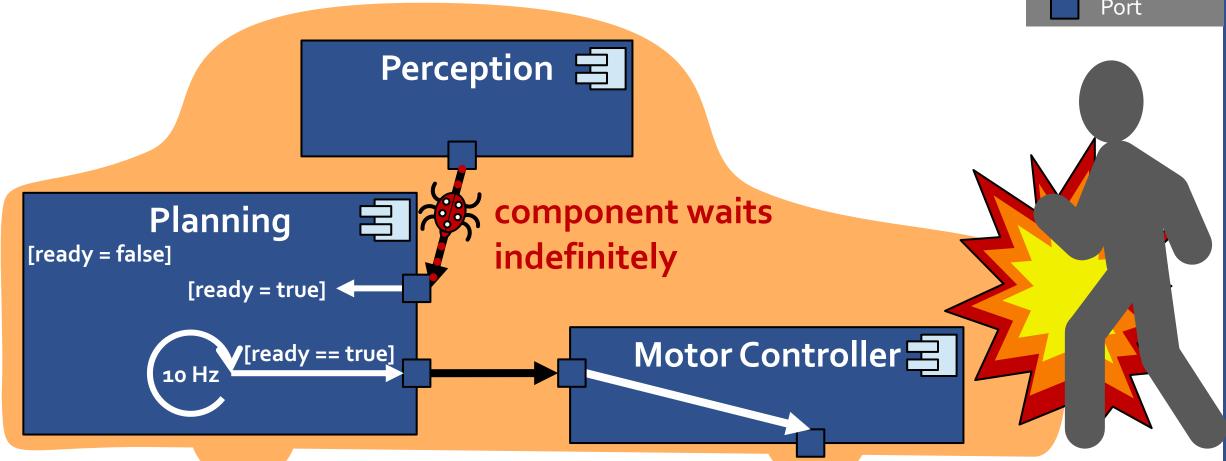
Robotics Systems are Complex Component-based Systems



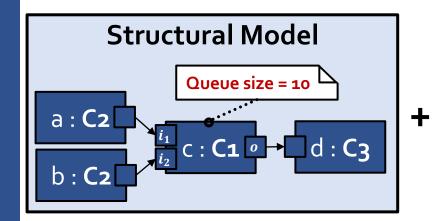


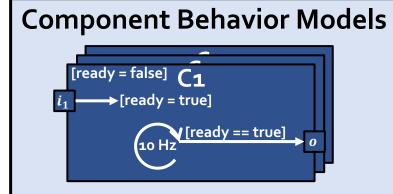
A Class of Very-hard-to-find Bugs Results from Incorrect Component Composition

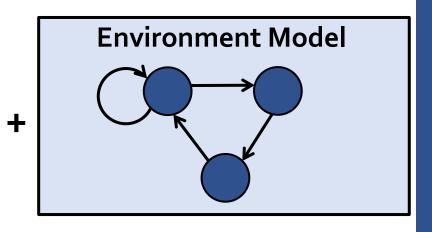
Software Component Connector Port



Good News: Model-Based Analysis can Find Bugs







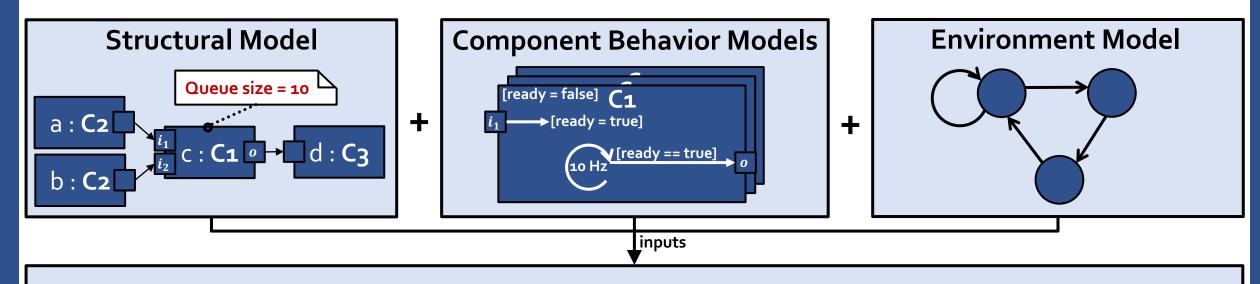
Models the **interface** of **components** (i.e., port) and **connectors**

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

Models the states and state transitions of components, their inputoutput relationship

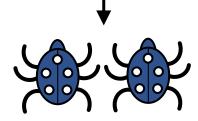
Models **types of inputs** from the environment and how the environment **reacts** to actions of the system

Good News: Model-Based Analysis can Find Bugs

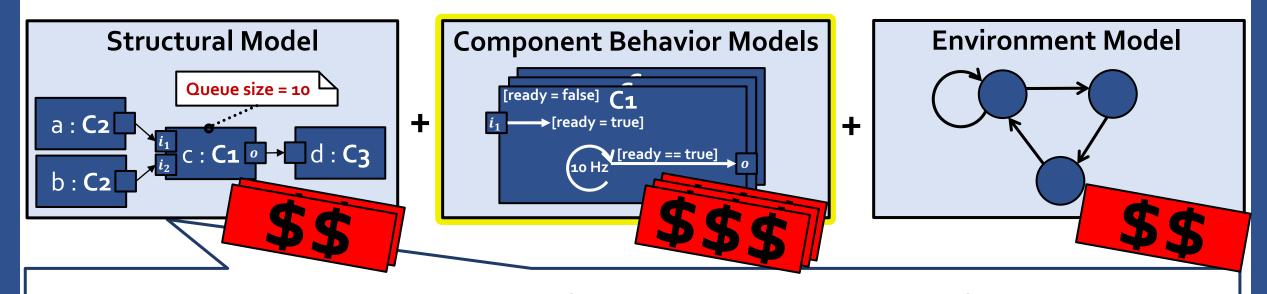


Model Checking / Simulation

(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)

```
// Need to hold back on extra trajectories until CPU utilization is figured out..
 if (g sim mode)
     // If the velocity is nonzero (would preclude horizon calc) publish trajectory viz
                            Code is Very Complex
       // 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.sv = tempGoal.sv + 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 twastrangular.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

```
Exipotory_question.opp
Mavigation via Cubic Spline Osneration
Created by Matthew O'Raily on 7/17/15.
Coppright (c) 2015 Matthew O'Raily, All rights reserved.
mokally@east upcas.edu
         Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions
           * Redistributions of source code must retain the above ospyright motice, this
list of conditions and the following disclaimer.
           IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND
DISCLARMED. IN SU EVERY SHALL THE COPYRIGHT SOLDER OR CONTRIBUTIONS BE LIABLE
FOR ANY DIRECT, INCIDENTAL, SPECIAL, EXCHIPANÇOR CORRESPONDED.
         DANAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS ON
         CATHED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, ON TORY (INCLUDING HEILIGENCK ON OTREWWIRE) ARLEING IN ANY WAY OUT OF THE GRE
  finclude openetry mags/PoseEtamped.b>
finclude openetry mags/PoseEtamped.b>
finclude openetry mags/DoseEtamped.b>
#include Cnaw mags/Fath.h>
#include Cnawlor mags/MavFatFix.h>
#include Cnawlor mags/Marker.h>
#include Cntd mags/MoolTh>
 #include <atd mage/Float32.h>
#include <atd mage/Float32.h>
#include <atd mage/Float6MultiArray.h>
#include *rusTime manager/ConfigNaypointFollower.h
 #include "waypoint follower/libwaypo
#include "libtraj jen.h"
#include "vehicle scoket/Caninfo.h"
 #define MEEKL_ANGLE_MAX (31.20167)
 #Mediae MERKL TO STERRING (STRERING ANGLE MAX/MHERL ANGLE MAX
#Mediae MERKL TANK (2.70)
 #define MEEKL BASK MKI (2.84998)
#define MEEKL TO STEKKING MKI (22.00)
    statio ros::Publisher vis pub;
statio ros::Publisher traj pub;
      g lookahead threshold = config->lookahead threshold
g laitial velocity = config->welcoity)
```

```
NOR IMPO STREAM("dialog : " << q_initial_velocity << " km/h (" << km/hlups(q_initial_velocity) <
  (a )");
return kmph2mps(g_initial_velocity);
                                                                                                                                                                                                                                                                                           return 1_qual;
                                                                                                                                                                                                                                                                                            /// Compute oursent state of the vehicle
        (current velocity mps * q look ahead threshold calc ratio < q minimum lool ahead threshold return q minimum lool ahead threshold;
                                                                                                                                                                                                                                                                                                  union State 1_veh;
 double steering wheel angle = msg->angle;
g current velocity = (msg->apsed)*(1000.00/1600)
                                                                                                                                                                                                                                                                                             steering wheel angle = steering wheel angle*(2.1496/180.00);
g can info curvature = (steering wheel angle / (double) MREEL TO STEERING) / MREEL BASE,
NOT INFO TOTALM("Steering Wheel Angle" (<a href="">Control of the Angle ("Accessing wheel angle")</a>)
                                                                                                                                                                                                                                                                                                    double roll,pitch,yaw;
m.getEFF(roll,pitch,yaw)
                                                                                                                                                                                                                                                                                                    // Because we are on the coordinate system of the base frame
 consists overstors = (see-represent)
can info conventure = (steering wheel angle / (double) NEEEL TO STEERING NEEL | MEEEL BASE NEEL
NOO INFO ITERAN("Steering Wheel Angle: "Cotteering wheel angle);
NOO INFO ITERAN("Curvature from Car", "co whi info Survature);
                                                                                                                                                                                                                                                                                                    // Det the desired velocity
1 veb.vdes = g current waypoints.getMaypointVelocityMFE(next waypoint)
RDE INFO STEEME (Desired Velocity: "<< 1 veb.vdes);</pre>
 double lookshead threshold = getLookAheadThreshold(closest waypoint
                                                                                                                                                                                                                                                                                                    if (tg sim mode as g price mode)
                                                                                                                                                                                                                                                                                                itatio union Egline wagpointTrajectory(union State veh, union State geal, union Egline ourvature, int
ant wagpoint
                                                                                                                                                                                                                                                                                                         // Net time horizon
double horizon = corvature.s/veh.vdes;
ROS_INFO_STREAM("vdes: " << veh.vdes);
ROS_INFO_STREAM("horizon: " << horizon;
                                                                                                                                                                                                                                                                                                              // Num motion model veh next = motionModel(veh, qual, curvature, dt, horizon, 0);
                                                                                                                                                                                                                                                                                                              // If the motion model doesn't get us to the goal compute new parameters
                                                                                                                                                                                                                                                                                                                        // update parameters
curvature = questateCorrection(veh, veh_mest, qual, ourvature, dt, horizon);
iteration**;
                                                                                                                                                                                                                                                                                                                          // Escape route for poorly conditioned Jacobian if (conventure, successions)
                                                                                                                                                                                                                                                                                             (BIL THE STREET THAT I BLASS AS "COURL MADE" BY "COURL MADE" LAST COURL MADE COURT COURT MADE COURT COURT COURT COURT COURT COURT COURT COURT CO
 // Out the sext next suppoint position
// foolube regionic loakhaefi | x = yeak pp. transformMuppoint(next_suppoint(n).pstX())
// double regionic loakhaefi | y = yeak pp. transformMuppoint(next_suppoint(n).pstX()).
// double regionic loakhaefi | y = yeak pp. transformMuppoint(next_suppoint(n).pstX()).
// double regionic loakhaefi | y = yeak pp. transformMuppoint(n).pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX().pstX(
                                                                                                                                                                                                                                                                                                           and then properly many many or record over every or a count over these record over these
                                                                                                                                                                                                                                                                                             ROW_INFO_ETHERM("Init Elate: as "coveh.sect" sy " coveh.spct" theta "coveh.thetact" kappa
                                                                                                                                                                                                                                                                                                    Cush.kappa);
MDE_IMPO_BTMEMM("Deal State: aw "Cogoal.awcc" ay " cogoal.aycc" theta "cogoal.thetacc" kappa
opoal.kappa);
calculative/condinate (g current waypoints.getMaypointFoilion(sext waypoint*2), g current yees.pose).x/
oblie waypoint lookshead 2 y =
calculative/condinate (g current waypoints.getMaypointFoilion(sext waypoint*2), g current yees.pose).y/
```

defines structure and behavior for inter-component communication

```
line strip.header.stamp = ros::Time();
line strip.action = visualization_mags::Marker::ACD
                                                                                                                                                                                                                         int closest waypoint = qetClosestMaypoint(g current waypoints.qetCurrentMaypoints(), g current pose.pose)
NOW INTO NTREMM("closest waypoint = " << climent waypoint);
                                                                                                                                                                                                                         // If the oursent waypoint has a valid index
if (closest_waypoint > 0)
                                                                                                                                                                                                                              // Neturn the next waypoint
int next waypoint = getNextwaypoint(closest waypoint)
                                                                                                                                                                                                                                 // Display and publish waypoint information displayMeatMaypoint(seat waypoint);
  // Det the color and transparency (green and solid) if selected
                                                                                                                                                                                                                                 _if stat.data = true;
_stat pub.ucblish( if stat)
                                                                                                                                                                                                                                // Determine the desired state of the vehicle at the rest waypoin union State coal = computeMarcointDoal(next warpoint);
    line_strip.color.g= 1.0;
line_strip.color.g= 1.0;
line_strip.color.a = 1.0;
                                                                                                                                                                                                                                // Estimate the current state of the vehicle usion State web = computeTeh(old_time, old_theta, rest_waypoint)
 // Init temp state for storing results of gesLineStrip
                                                                                                                                                                                                                                    veh fmm = nextItate(veh, prev_curvature, veh.vdes, 0.2, 0);
NOS INFO HTMANK("est kappa: "-<veh fmm.kappa);
  // Figure out sim time horizon
double horizon = curvature.s/veh.vdes
                                                                                                                                                                                                                                // Initialize the estimate for the curvature union Toline curvature = initParams(veh. coal)
                                                                                                                                                                                                                                // Generate a cubic spline (trajectory) for the vehicle to follow curvature = wavecistTratectory(veh. coal. curvature. seat wavecist
                                                                                                                                                                                                                                // Check that we got a result and publish it or stream expletive to screen if (ourvature.scoceas==TRUE)
                                                                                                                                                                                                                                  std_megs::Float60MultiArray spline;
spline.data.clear()
   temp = qualimentrip(vah, curvature, vdes, sim_time);
p.w = temp.ax;
p.y = temp.ay;
p.w = 0.0;
line_stuip.pointe.push_back(p);
                                                                                                                                                                                                                                    for(int i = 0; i < 6;i++)
                                                                                                                                                                                                                                        spline.data.push_back(curvature.spline_value[i])
                                                                                                                                                                                                                                  spline_parameters_pub.publish(spline)
                                                                                                                                                                                                                                // Also publish the state at the time of the result for curvature.
std mage: rfm.ofbultikray state;
std%.dstc.olber();
int main(int argo, char **argy)
                                                                                                                                                                                                                                    state.data.push back(veh.state value(i))
 // Ret up NOS, TO DO: change to proper name (same with rest of the file)
                                                                                                                                                                                                                                           SPLINE INDEX++;
NOS INFO_STREAM("Epline published to EVIZ");
                                                                                                                                                                                                                                         // Likely will change when valid cost map arrives.
// Note: pragma indicates parallelization for OpenMF
                                                                                                                                                                                                                                        // Setup variables
union State tempCoal= goal;
                                                                                                                                                                                                                                         // Tell OpenMT how to parallelize (keep i private because it is a counter; 
#pragma comp parallel for private(i)
                                                                                                                                                                                                                                           // Index through all the predefined perturbations from wavpoint
                                                                                                                                                                                                                                              // Shift the y-coordinate of the goal
temptoal.sy = temptoal.sy + perturb(i-
                                                                                                                                                                                                                                             // Compute new spline extraw waypointTrajectory(weh, tempGoal, curvature, next_waypoint
                                                                                                                                                                                                                                             // Display trajectory
if(veb. vob.00)
                                                                                                                                                                                                                                 // Update previous time and orientation measurements old_time= veh.time=tamp/ \,
                                                                                                                                                                                                                                NOS IMPO STREAM("Lost waypoint!"
                                                                                                                                                                                                                                 stat pub.publish( lf stat)
Twist.twist.linear.z = 0;
twist.twist.angular.z = 0;
```

Tobias Dürschmid, Chris Timperley, David Garlan, Claire Le Goues: ROSInfer Statically Inferring Behavioral Component Models for ROS-based Robotics System

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)
                                                                   bool ready = false; ←
                                                                                            Initial State
                                      Subscriber Port
                                                                                          Subscriber Callback
 ros::Subscriber sub = nh.subscribe("t sub", receive initial);
 void receive initial(const Message msg)
 const int local LOOP RATE = 10;
                                                                                        State Change
                                           Periodic Rate
 ros::Rate loop rate(local LOOP RATE);
  while (ros::ok())
                             Periodic Loop
   if (!ready) ← State Condition
                                                                       Planning
                                               Initial State
                                                             [ready = false]
     loop rate.sleep(); ← (Periodic Sleep
                                                                              State Change
                                                                     [ready = true] ◀—
                                                                                                Subscriber Port
                                            ROSInfer
                                                                                     State Condition
   pub.publish (msg); ← ( Message Output
                                                                         [ready == true]
    loop rate.sleep();
                                                                                               Publisher Port
 return 0;
                              Tobias Dürschmid, Chris Timperley, David Garlan, Claire Le Courses ROSInf
       19 Apr 2024
                                                                                                   12
                           Statically Inferring Behavioral Component Models for ROS-based Robotics System
```

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;
                                 Periodic
                                  Rate
const int local LOOP RATE = 10;
ros::Rate loop rate (local_LOOP_RATE);
while (ros::ok())
                          Periodic Loop
  if (!ready)
                           Periodic Sleep
    loop rate.sleep();
    continue;
  pub.publish(msg); ←
                         Message Output
  loop rate.sleep();
return 0;
```

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

Method: Compare to 149

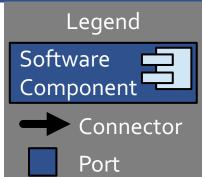
manually inferred models

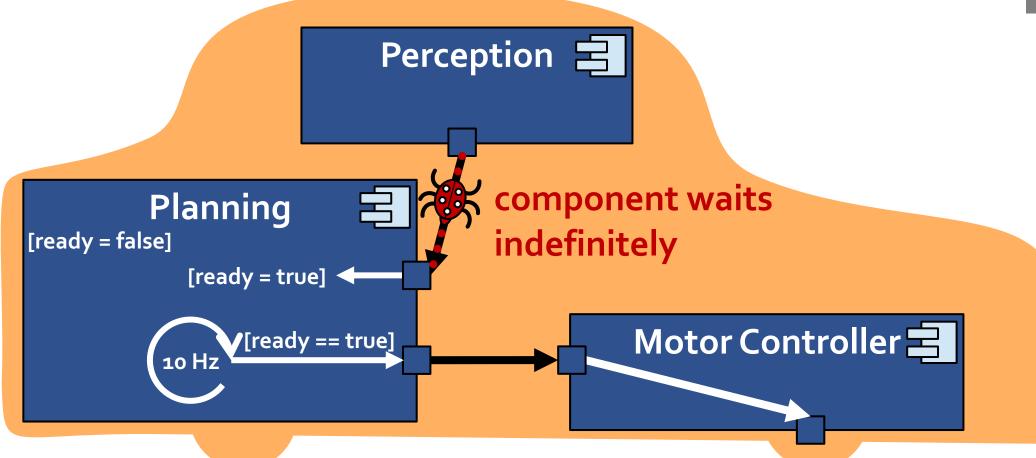
Results for State-based Behavior:

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

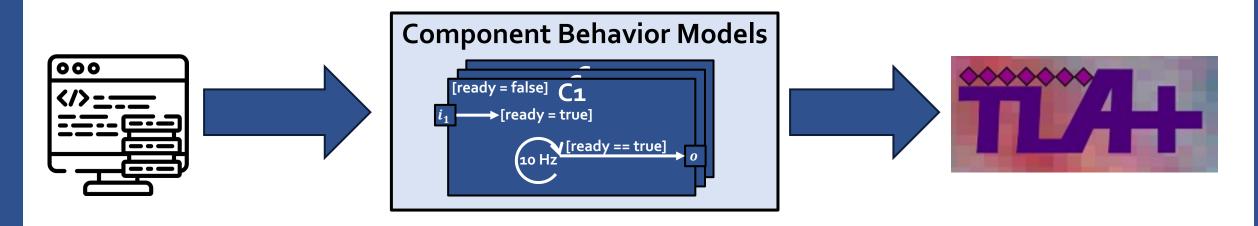
```
bool ready = false; ←
                         Initial State
void receive initial(const Message msg)
 ready = true; ←
                    State Change
const int local LOOP RATE = 10;
ros::Rate loop rate(local LOOP RATE);
   le (ros::ok())
                          Periodic Loop
                State Condition
     (!ready) ←
    loop rate.sleep();
    continue;
  pub.publish (msg); ←
                         Message Output
  loop rate.sleep();
return 0;
```

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





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

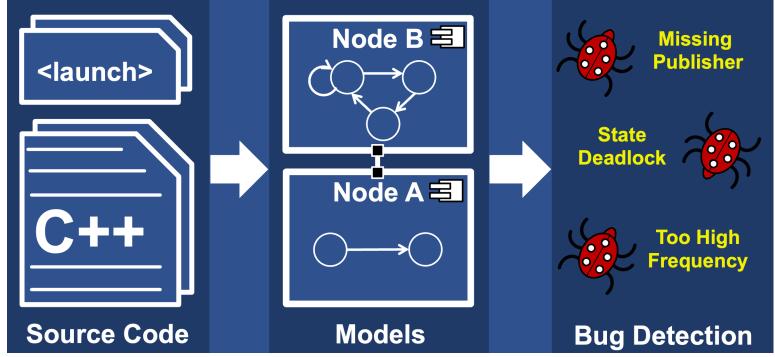


- No extensive bug data set of behavioral architecture composition bugs is available
- We ran ROSInfer on three bugs from the ROSDiscover data set that ROSDiscover could not find
- ROSInfer found all three bugs

Summary



- **<u>Problem</u>**: 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