

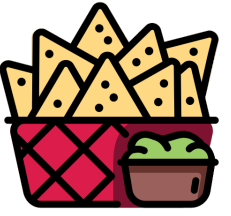
Modelling and model-checking a ROS2 multi-robots system using Timed Rebeca

Presenter: **Hiep Hong Trinh**

PhD student, hiep.hong.trinh@mdu.se

Date: *January 18th, 2024*





Appetizer: Rebeca model checker

Rebeca IDE

File Edit Navigate Project Window Help

Project Explorer

- dva436_test_rebecca
- MobileRobots
- ros2rebecca
 - backup
 - out
 - script
 - src
 - ros2rebecca.property
 - ros2rebecca.rebeca
 - ros2rebecca.stateSPACE
 - ros2rebecca-5.property
 - ros2rebecca-5.rebeca
 - ros2rebecca-5.stateSPACE
 - ros2rebecca-5b.property
 - ros2rebecca-5b.rebeca
 - ros2rebecca-5b.stateSPACE
 - ss
 - LICENSE
 - README.md
 - test-timed
 - Train-Controller-Two-Trains (1) (in Tr

```
@priority(2) Node r4(theMap):(4,5,35,48,15,2,0.8,1500,false);
@priority(6) Node r5(theMap):(5,5,45,48,5,2,1.2,1500,false);
@priority(1) MapServer theMap(r1,r2,r3,r4,r5):(5);
*/
//bad config
@priority(2) Node r1(theMap):(1,15,5,48,45,2,1.6,1500,false);
@priority(3) Node r2(theMap):(2,25,5,48,35,2,1.6,1500,false);
@priority(4) Node r3(theMap):(3,5,25,48,25,2,1.6,1500,false);
@priority(5) Node r4(theMap):(4,5,35,48,15,2,1.6,1500,false);
@priority(6) Node r5(theMap):(5,5,45,48,5,2,1.6,1500,false);
@priority(1) MapServer theMap(r1,r2,r3,r4,r5):(5);
}
```

```
<state id="1614_0" atomicpropositions="" >
  <rebec name="r1">
    <statevariables>
      <variable name="Node.rindex" type="byte">1</variable>
      <variable name="Node.targetX" type="int">48</variable>
      <variable name="Node.targetY" type="int">45</variable>
      <variable name="Node.rx" type="int">37</variable>
      <variable name="Node.ry" type="int">28</variable>
      <variable name="Node.rdir" type="int">1</variable>
      <variable name="Node.velocity" type="int">7</variable>
      <variable name="Node.distance2target" type="int">22</variable>
      <variable name="Node.target_tolerance" type="int">2</variable>
      <variable name="Node.obstacles" type="int[101]">[8, 10241038, 10251038, 10251039, 10261039, 10281044, 104510
```

Counter Example

r4.UPDATEMOVINGSTATUS from r4 @(4960)
647_0
theMap.UPDATEROBOTLOCATION from r4 @(4960)
648_0
r5.UPDATEMOVINGSTATUS from r5 @(4960)
649_0
theMap.UPDATEROBOTLOCATION from r5 @(4960)
650_0
Time progress by 38 units @(4998)
651_0
r3.UPDATEMOVINGSTATUS from r3 @(4998)
652_0
theMap.UPDATEROBOTLOCATION from r3 @(4998)
assertion failed

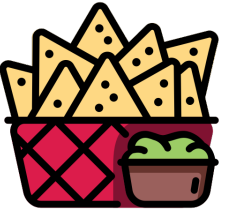
Attribute	Value
Queue Content	
Now	4998
r3	
State Variables	
Node.rindex	3
Node.targetX	48
Node.targetY	25
Node.rx	37
Node.ry	28
Node.rdir	0
Node.velocity	7
Node.distance2target	13
Node.target_toleran	2
Node.obstacles	[13, 10111034, 10111035, 1...
Node.targets	[1, 10251048, 0, 0, 0, 0, 0, ...
Node.moves	[25, 10251006, 10251008, 1...
Node.moveidx	18
Node.isWaiting	false

Problems Analysis Result Console

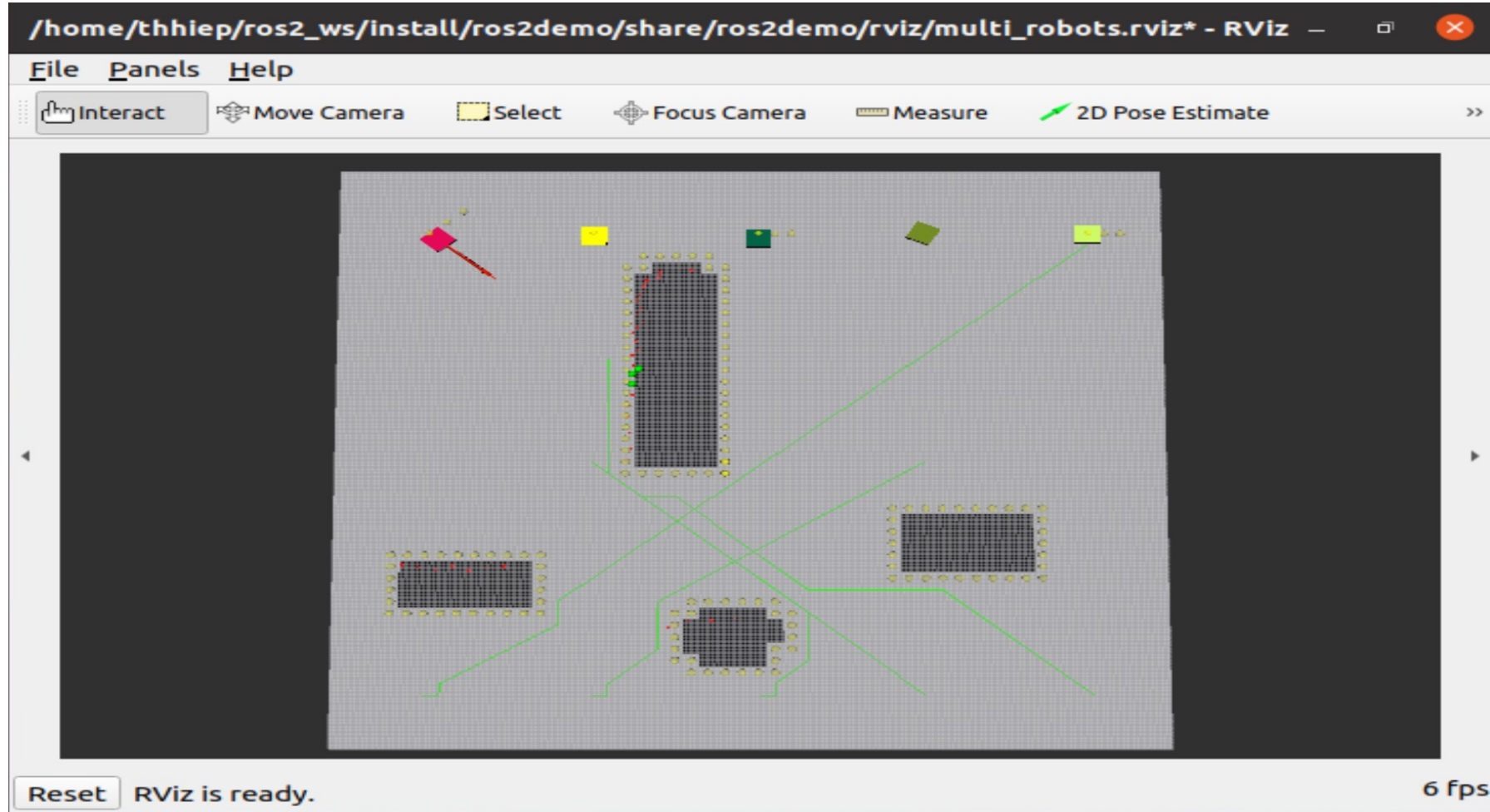
Attribute	Value
SystemInfo	
Total Spent Time	116
Number of Reached States	1614
Number of Reached Transitions	3427
Consumed Memory	38736
CheckedProperty	
Property Name	Deadlock-Freedom and No Deadline Missed
Property Type	Reachability
Analysis Result	assertion failed
Message	Collision with another robot!

Writeable Insert 1314 : 1 : 87378

9:04 AM 22-May-23

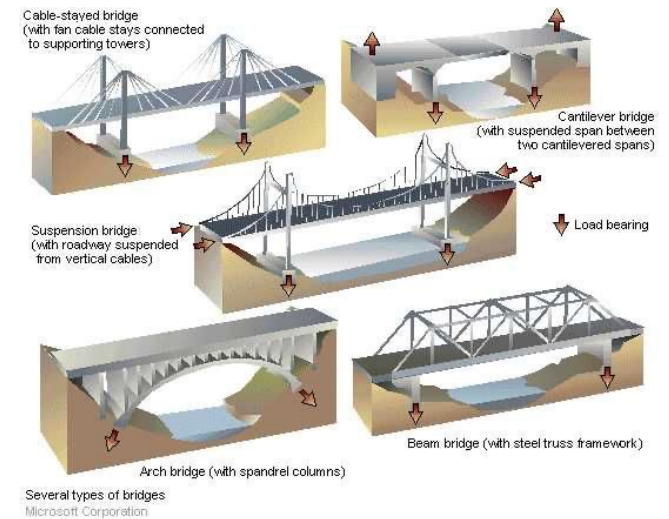


Appetizer: ROS2 simulation with 5 robots



Robotics domain

- Robotic applications:
 - Complex in structure, complicated in behaviors
 - Mathematical models
 - shape transformation, motion, dynamics
 - Sophisticated algorithms
 - Optimization, searching, recognition ...
 - Interactions with environment
 - map, static and mobile obstacles, sensors, actuators
 - Autonomy:
 - human-like in sensing, thinking, making decisions, learning
- Challenges to modelling and model-checking:
 - Complexity in data structures, communications and algorithms
 - Heavy computation amount
 - Lack of domain knowledge → toy problems

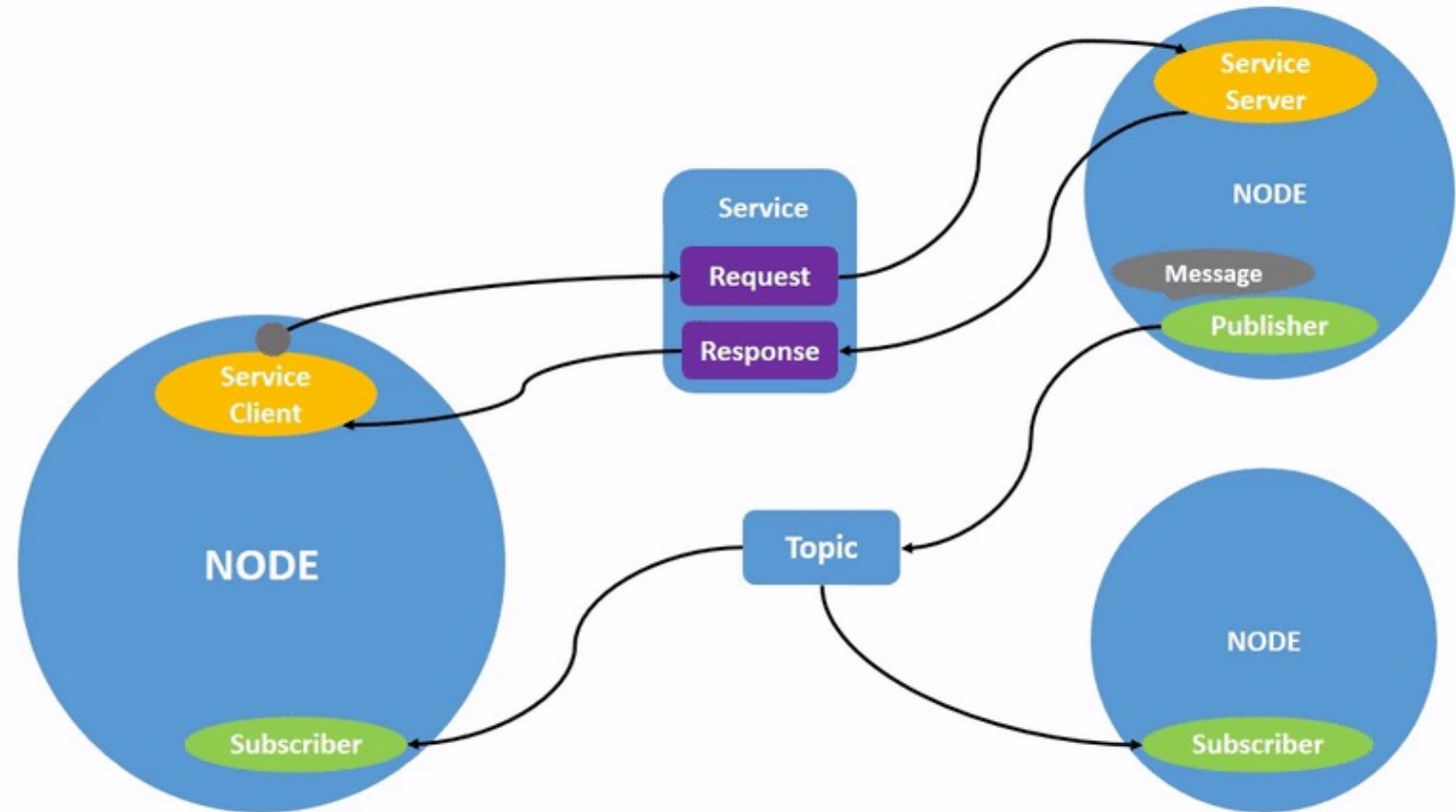


Component-based modelling & dev.

- [David G.]: “CPS ecosystems. There are several ecosystems of reusable building blocks in CPS. For example, the robotics operating system (ROS) is widely used in robotics applications, and it provides extensive libraries of components for assembling systems. This presents a challenge and an opportunity for projects such as SACSys. It is a challenge, because real systems of the future will not be built from scratch – but largely created through component composition ... This means that it should be possible to gain huge leverage by specifying just those core components, and providing guidance on how to use them correctly.”
- → Reusable components and templates from this work, later explained.
- → Ground-breaking work, first step

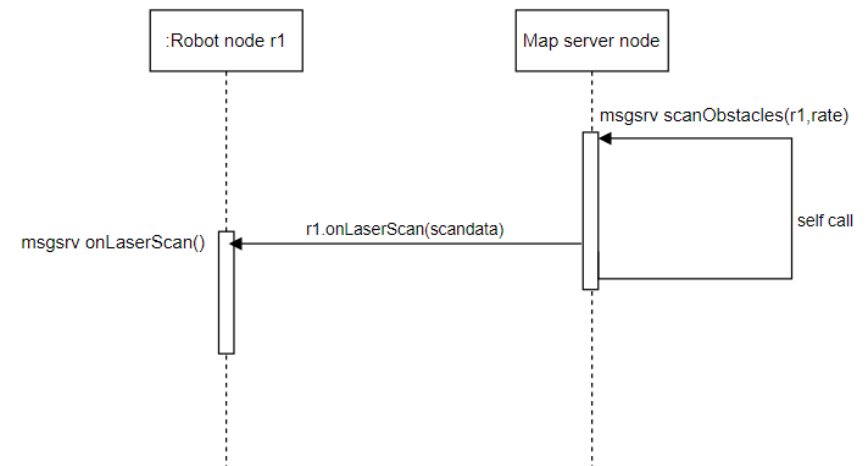
ROS2 architecture – node topography

- Nodes: parallel processing units
- Keep running, wait for incoming events, respond & send outputs
- Asynchronous interactions: topics, services, actions



Timed Rebeca

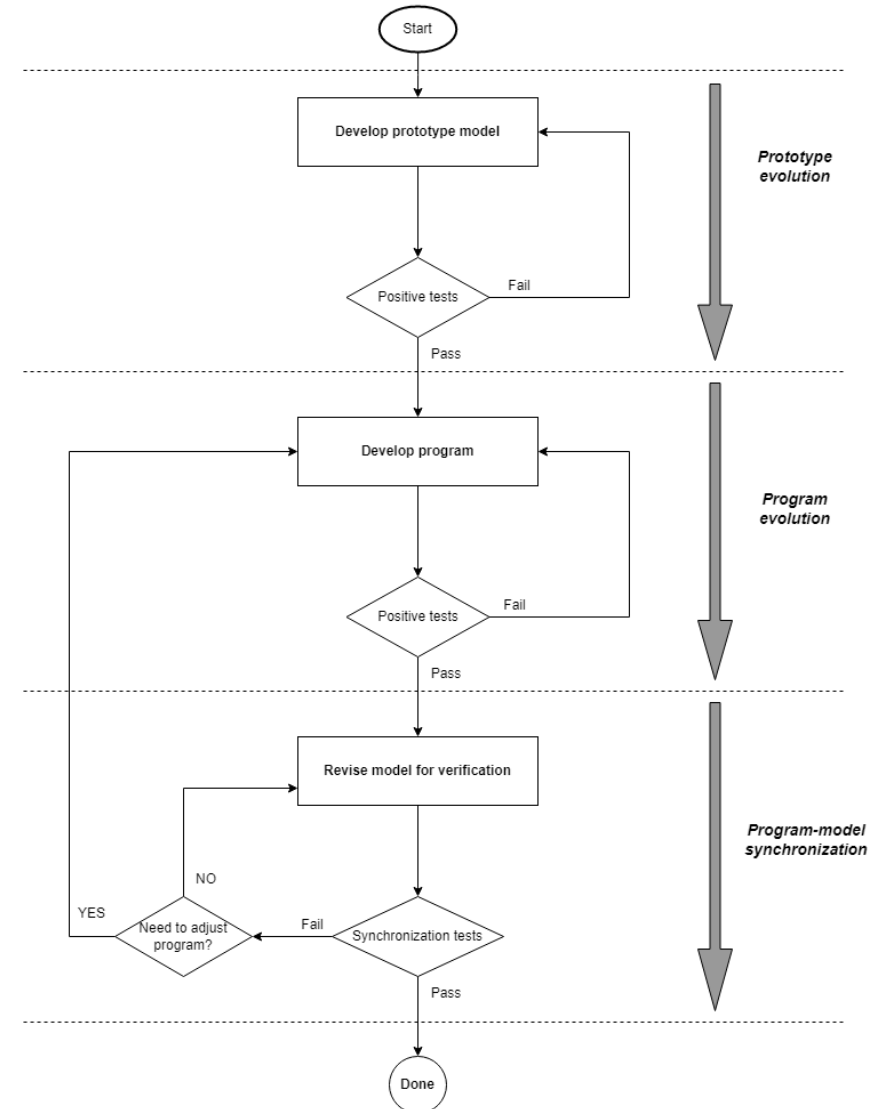
- Actor-based model
 - architectural modelling (entities and links)
- Concurrent, reactive systems (*rebecs*)
- Message-based async. Interactions (*msgsvr*)
- Timing semantics → timed loops, exec. time, time limits
 - *after*(time_taken or period),
deadline(max_age)
- Plus: developer-friendly syntax & flow, IDE



The flow

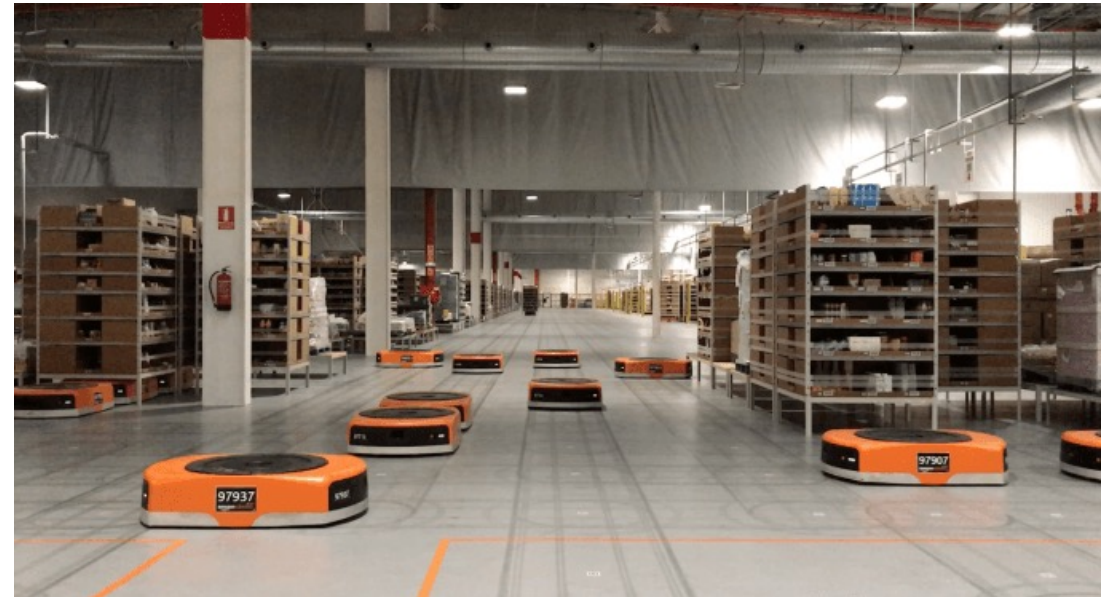
- Select a robotic problem
- Design architecture in ROS2 node topography
- Model in Rebeca, make it pass basic positive tests
- Develop corresponding ROS2 code
- Modify code to smooth robotic behaviours
- Revise model to match with code evolution
- Check program-model synchronization (match test)

Model-based development & evolution flow

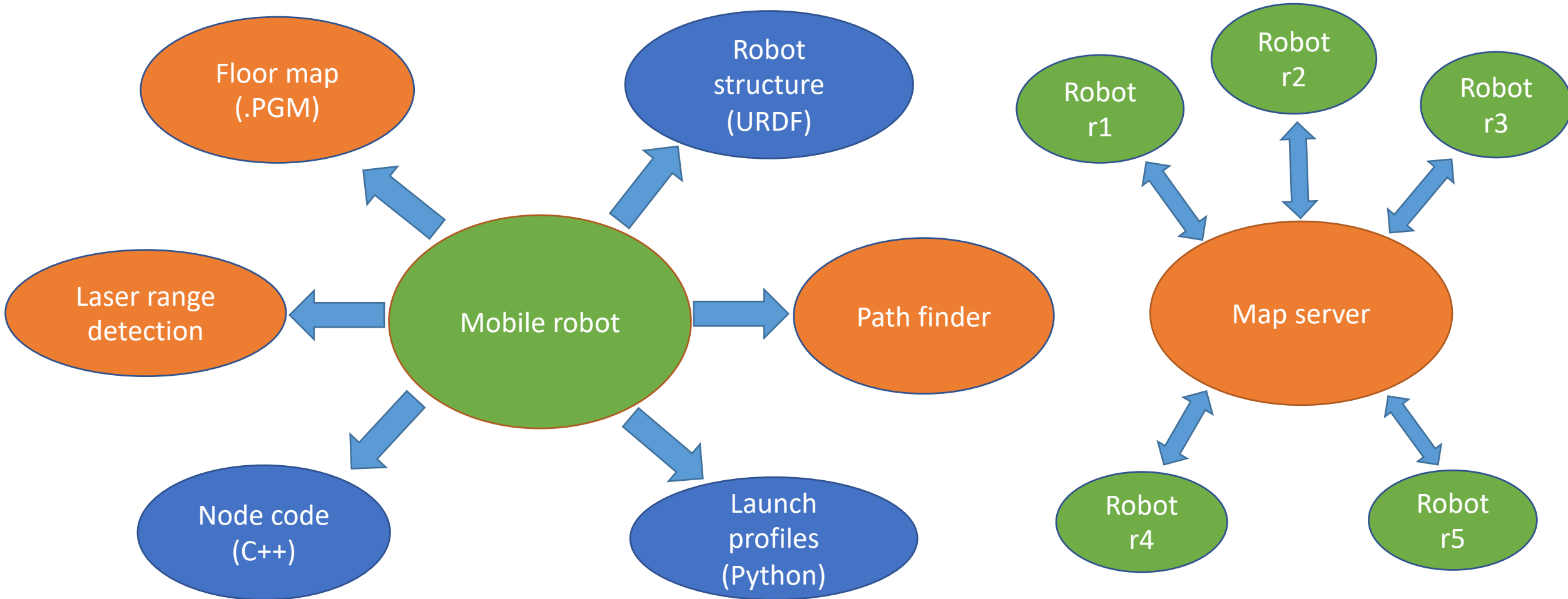


Industrial robotic problem: multiple AMRs

- Can project onto a 2D Cartesian → dimension reduction
- A mobile robot is equal to a mobile obstacle
- Multi robots → more complexity, not less
- Typical problems:
 - Detecting obstacles (static & mobile)
 - Avoiding collisions
 - Resolving congestions
 - Planning & replanning paths
- Properties to verify:
 - Deadlock freedom
 - Collision freedom
 - Target reachability



Basic blocks of AMRs problem



Results: working & non-working

Parameters	Case C1a		Case C1b		Case C1c	
Scan rate	100		100		100	
Speed limit	1.275		1.275		1.275	
Stop zone	0.3		0.3		0.3	
<i>Robots</i>	<i>Speed</i>	<i>Wait</i>	<i>Speed</i>	<i>Wait</i>	<i>Speed</i>	<i>Wait</i>
R1	0.5	1500	0.3	1500	0.4	1500
R2	0.7	2000	0.3	2000	0.5	2000
R3	0.8	2500	0.3	2500	0.7	2500
R4	0.5	3000	0.3	3000	0.8	3000
R5	0.3	1500	0.7	1500	0.9	1500
Analysis result	Satisfied		Satisfied		Satisfied	
States	19264		20554		15747	
Transitions	43814		46868		35642	
Simulations	5		5		5	
Simulation results	5/5 passed 0/5 failed		5/5 passed 0/5 failed		5/5 passed 0/5 failed	

Parameters	Case C2a		Case C2b		Case C2c	
Scan rate	140		100		140	
Speed limit	0.91		1.275		0.91	
Stop zone	0.3		0.3		0.3	
<i>Robot</i>	<i>Speed</i>	<i>Wait</i>	<i>Speed</i>	<i>Wait</i>	<i>Speed</i>	<i>Wait</i>
R1	0.9	1500	0.5	1500	0.9	1500
R2	0.9	2000	0.5	1500	0.8	2000
R3	0.9	2500	0.5	1500	0.7	2500
R4	0.9	3000	0.5	1500	0.6	3000
R5	0.9	1500	0.5	1500	1.0	1500
Analysis result	Assertion failed (collision)		Assertion failed (collision)		Assertion failed (collision)	
States	3074		6816		1740	
Transitions	6602		15293		3727	
Simulations	5		5		5	
Simulation results	2/5 passed 3/5 failed		3/5 passed 2/5 failed		0/5 passed 5/5 failed	

Results – a working case

The screenshot displays the Rebeca IDE interface. The Project Explorer on the left shows a project structure with folders like 'MobileRobots' and 'ros2rebeca', and files such as 'ros2rebeca-5c.property'. The main editor shows C++ code for a robot pathfinding algorithm, including a `main` function that initializes nodes and a map server. The Console window at the bottom shows the execution output, including node priorities and map server initialization. A recording window is overlaid on the right, showing video source settings (Window - ubuntu20-ros2-foxy), video encoder settings (FFmpeg, Mp4), FPS (10), and quality (70%).

```
    }  
    rpath[0]++;  
    rpath[rpath[0]] = path[i];  
    prevx = node[0]; prevy = node[1]; prevdir = dir;  
  }  
  //copy rpath to path  
  for(i=0;i<BUFFER_SIZE;i++){ path[i]=rpath[i]; }  
  } else {  
    //assertion(false,"Path not found");  
  }  
  robot.onNewPath(path);  
}  
  
@main {  
  @priority(3) Node r1(theMap):(1,5,5,45,45,2,0.5,1500,false);  
  @priority(5) Node r2(theMap):(2,5,15,45,35,2,0.7,2000,false);  
  @priority(6) Node r3(theMap):(3,5,25,45,25,2,0.8,2500,false);  
  @priority(4) Node r4(theMap):(4,5,35,45,15,2,0.5,3000,false);  
  @priority(2) Node r5(theMap):(5,5,45,45,5,2,0.3,1500,false);  
  @priority(1) MapServer theMap(r1,r2,r3,r4,r5):(5);  
  
  //working case: stop zone=0.3 scan_rate=100
```

Attribute	Value
SystemInfo	
Total Spent Time	510
Number of Reached States	19264
Number of Reached Transitions	43814
Consumed Memory	462336
CheckedProperty	
Property Name	Deadlock-Freedom and No Deadline Missed
Property Type	Reachability
Analysis Result	satisfied

Results – a non-working case

The screenshot shows the Rebeca IDE interface. The main editor displays the following code:

```
prevx = node[0]; prevy = node[1]; prevdir = dir;
}
//copy rpath to path
for(i=0;i<BUFFER_SIZE;i++){ path[i]=rpath[i]; }
} else {
//assertion(false,"Path not found");
}
}
robot.onNewPath(path);
}
}
main {
@priority(3) Node r1(theMap):(1,5,5,45,45,2,0.9,1500,false);
@priority(5) Node r2(theMap):(2,5,15,45,35,2,0.9,2000,false);
@priority(6) Node r3(theMap):(3,5,25,45,25,2,0.9,2500,false);
@priority(4) Node r4(theMap):(4,5,35,45,15,2,0.9,3000,false);
@priority(2) Node r5(theMap):(5,5,45,45,5,2,0.9,1500,false);
@priority(1) MapServer theMap(r1,r2,r3,r4,r5):(5);

//working case: stop zone=0.3, scan rate=100
/*
//14746 states, 33367 transitions
@priority(3) Node r1(theMap):(1,5,5,45,45,2,0.9,1500,false);
```

The console output shows the following error:

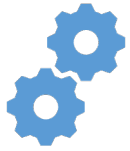
```
Assertion failed: Collision with another robot
```

The video recording window is overlaid on the right, showing the recording interface. The video source is set to 'Window - ubuntu20-ros2-foxy [Running] - Oracle VM VirtualBox'. The video encoder is set to 'FFmpeg' and the video format is 'Mp4 (x264 | AAC)'. The FPS is set to 10 and the quality is 70%. The audio source is set to 'No Webcam'.

Attribute	Value
SystemInfo	
Total Spent Time	135
Number of Reached States	3074
Number of Reached Transitions	6602
Consumed Memory	73776
CheckedProperty	
Property Name	Deadlock-Freedom and No...
Property Type	Reachability
Analysis Result	assertion failed
Message	Collision with another robot

Node	Value
Node.targets	[1, 10051045, 0, 0, 0, 0, 0, ...
Node.moves	[41, 10451006, 10441007, 1...
Node.moveidx	23
Node.isWaiting	false
Node.waits	0

Results: artifacts



**Rebeca model for prototyping
(version 1)**



ROS2 demo code

**Model-based development
Framework for multi robots
systems**



**Rebeca model for verifying
(version 2) → model-based
verification**

1 program – N models



**Coverage: all components of
AMRs problem**

Mapping

Laser-based obstacle detection

Robot physical dimensions

Robot movement characteristics
(rotation & linear, speeds, stopping
distance)

Dynamic path planning

Human-like collision avoidance &
congestion resolution

Challenges

- Discrete model vs. continuous behaviours
 - Discrete state variables vs. real variables (e.g. map data, coordinates, angles)
 - Not a simple 1-1 conversion: retain too much → impossible model checking, drop too much → information loss, inaccuracy
- Heavy computation
 - Exponentially multiplied in model checking
 - Complicated math calculations vs. inequivalent programming facilities in a modelling language



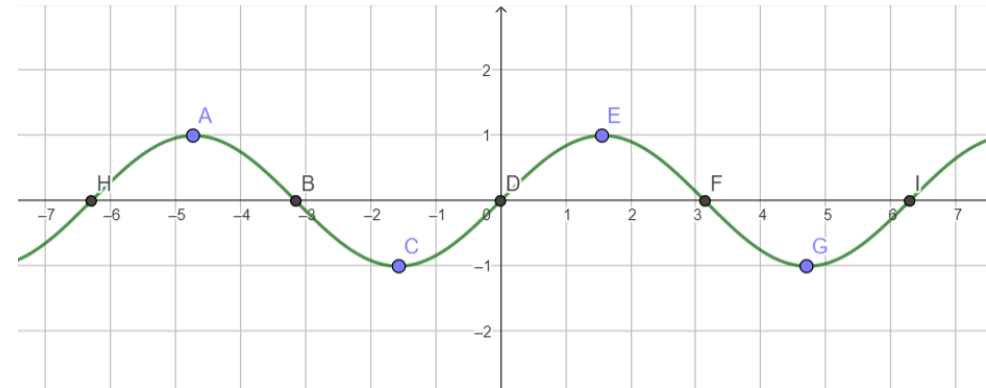
The Bad

- Detail level:
 - Real > Simulation > Model (for model checking)
 - For simulation: more is better
 - For modelling: less is better
- Reality gap: continuous system vs. discrete model
 - Real measurements vs. discrete state variables
 - Incremental, gradual vs. abrupt behaviors
 - Rounding = info. loss. How to sample?

→ Discretization strategies

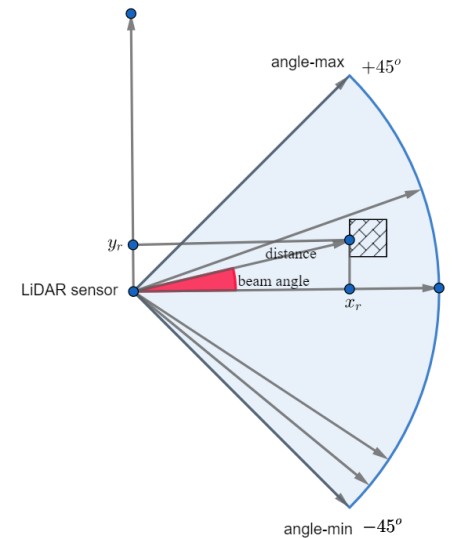
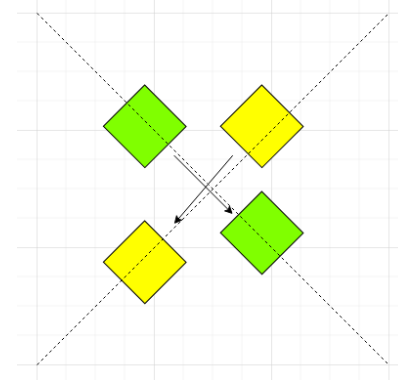
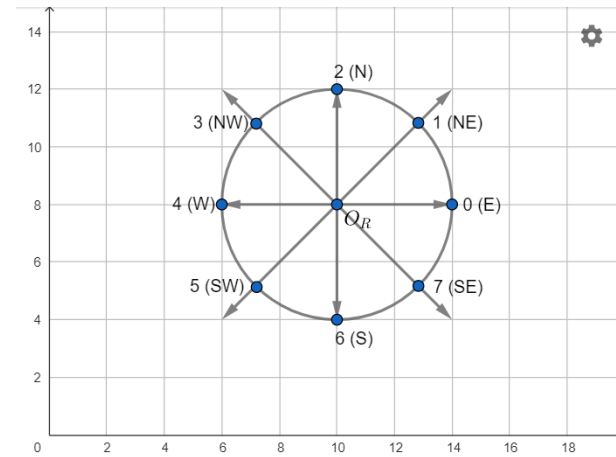
- Robotic complexity:
 - Mapping, sensory data, robot physical structure,
 - Path finding, kinematics (motion science)

→ Simplification strategy (component-wise, retaining system integrity)



The Bad & counter-tactics

- Discretize:
 - 2D projection \rightarrow occupancy grid, footprints/shadows
 - Robot directions: 8 angles
 - Scan step = $2^\circ \rightarrow$ known beam angles 0..360
 - Fine-grained level vs. accuracy
- Simplify:
 - Map size & resolution: 50x50
 - Robot structure: box-bot, one frame
- Just some extra work to de-simplify

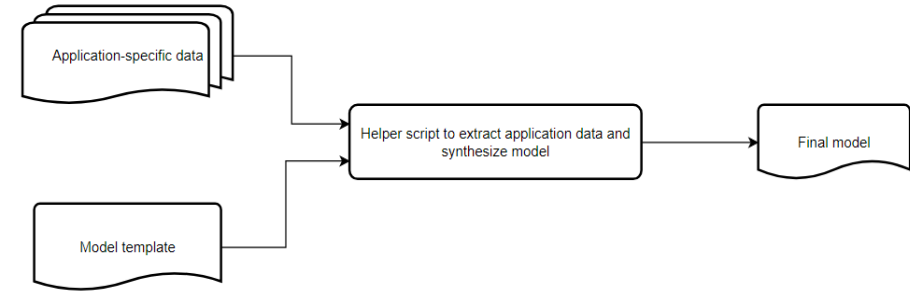


The Ugly - inconveniences

- Inequivalence of programming facilities in Rebeca
 - Not OOP, no inheritance
 - Only state variables, no local variables to each rebec
 - No struct, no string types
 - Fixed sized arrays: *int[100] a;*
 - Uninterpreted calls to common math functions: *sqrt(), cos(), sin(), tan(), atan(), ...*
 - Debugging, visualizing limitations



The Ugly & counter-tactics



- Helper script (PHP): extract & generate data, debug modelling code, visualize states
- Known angles 0..360 → precompute all trigonometric values
 - No more *sin()*, *cos()*, *tan()* !
 - Rule “Don’t repeat yourself” → do once, reuse later
- *Sqrt()* in *Euclidean* distance
 - Use a different heuristic without *sqrt()* – Octile distance, Manhattan distance
 - Square it: $2 = \sqrt{2} * \sqrt{2}$
 - Workarounds

Conclusion

- Two-fold or multiple-fold:
 - Model-based development and verification of ROS2 robotic systems using T.Rebeca
 - Rebeca model template & ROS2 code framework for AMRs
 - Human-like collision avoidance & congestion resolution algorithms
 - Exploratory method, modelling/dev. process, modelling techniques



Results – collision & congestion handling

The screenshot shows a text editor window titled 'mapserver.launch.py' with the following Python code:

```
1 import os
2 import yaml
3 from launch import LaunchDescription
4 from launch.actions import DeclareLaunchArgument
5 from launch.conditions import IfCondition, UnlessCondition
6 from launch.substitutions import Command, LaunchConfiguration
7 from launch_ros.actions import Node
8 from launch_ros.substitutions import FindPackageShare
9 from ament_index_python.packages import get_package_share_directory
10
11 robot_length = 0.2;
12 robot_width = 0.2;
13 safe_margin = 0.02;
14 stop_zone = 0.5;
15
16 pkg_name = 'ros2demo'
17 pkg_dir = FindPackageShare(package=pkg_name).find(pkg_name)
18
19 map_name = 'map50';
20
21 map_pgm = pkg_dir+'/maps/'+map_name+'.pgm'
22 assert os.path.isfile(map_pgm), 'Map file not found at '+map_pgm
23 map_yaml = pkg_dir+'/maps/'+map_name+'.yaml'
24 assert os.path.isfile(map_yaml), 'Map file not found at '+map_yaml
25 with open(map_yaml, 'r') as stream:
26     map_cfg = yaml.safe_load(stream);
27 #print(map_cfg);
28
29 def generate_launch_description():
30     # Create the map server node
31     map_server_node = Node(
```

Overlaid on the right side of the editor is a video recording window. The window title is 'Window - ubuntu20-ros2-foxy [Running] - Oracle VM VirtualBox'. It shows a video source selection dropdown set to the current window, video encoder options (FFmpeg, Gif, SharpAvi, Stream, Discard), and output settings (MP4, AAC, 10 FPS, 70% Quality). The recording path is 'C:\Users\ADMIN\Documents\Captura'.