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

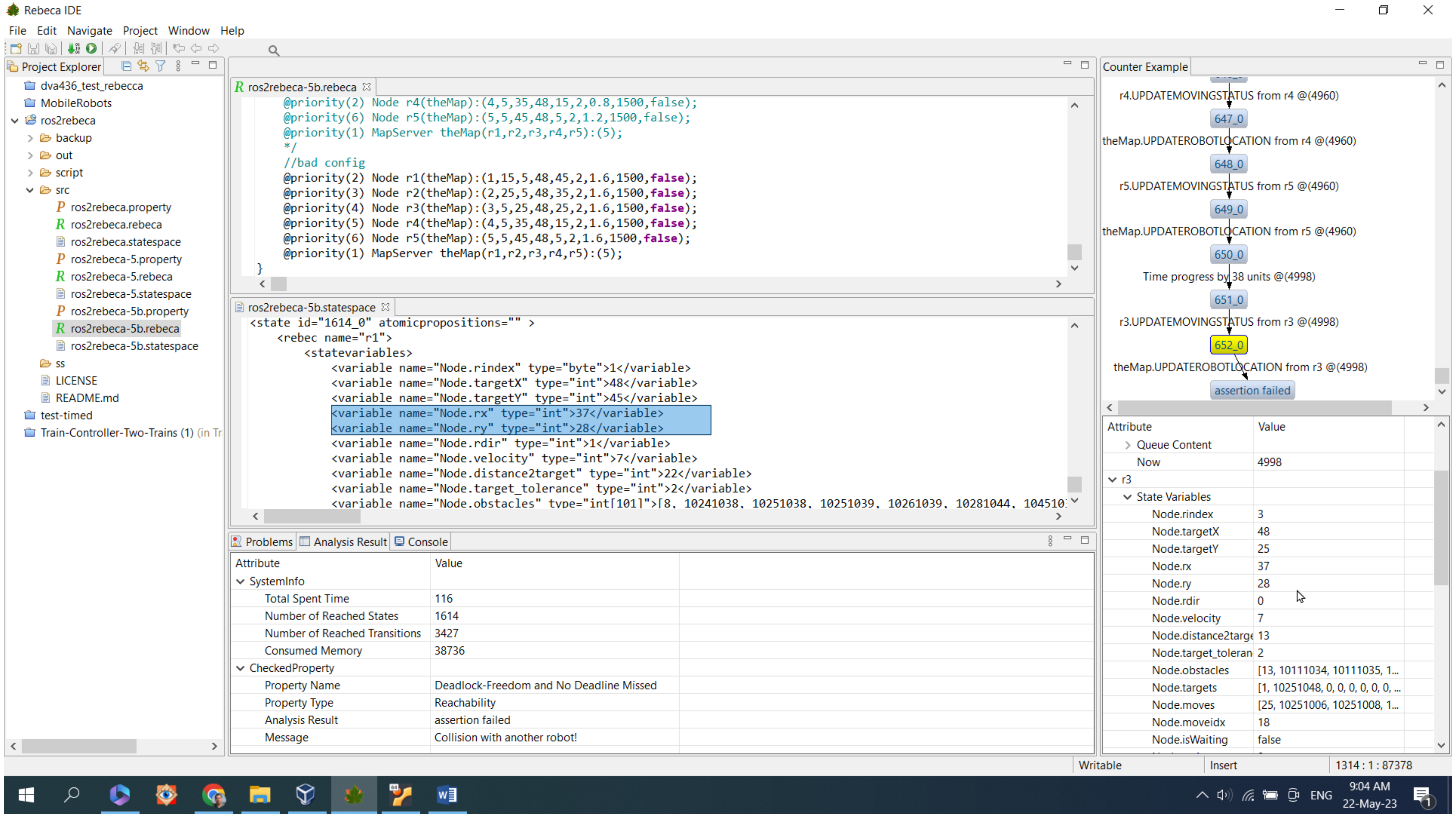
Presenter: **Hiep Hong Trinh**

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

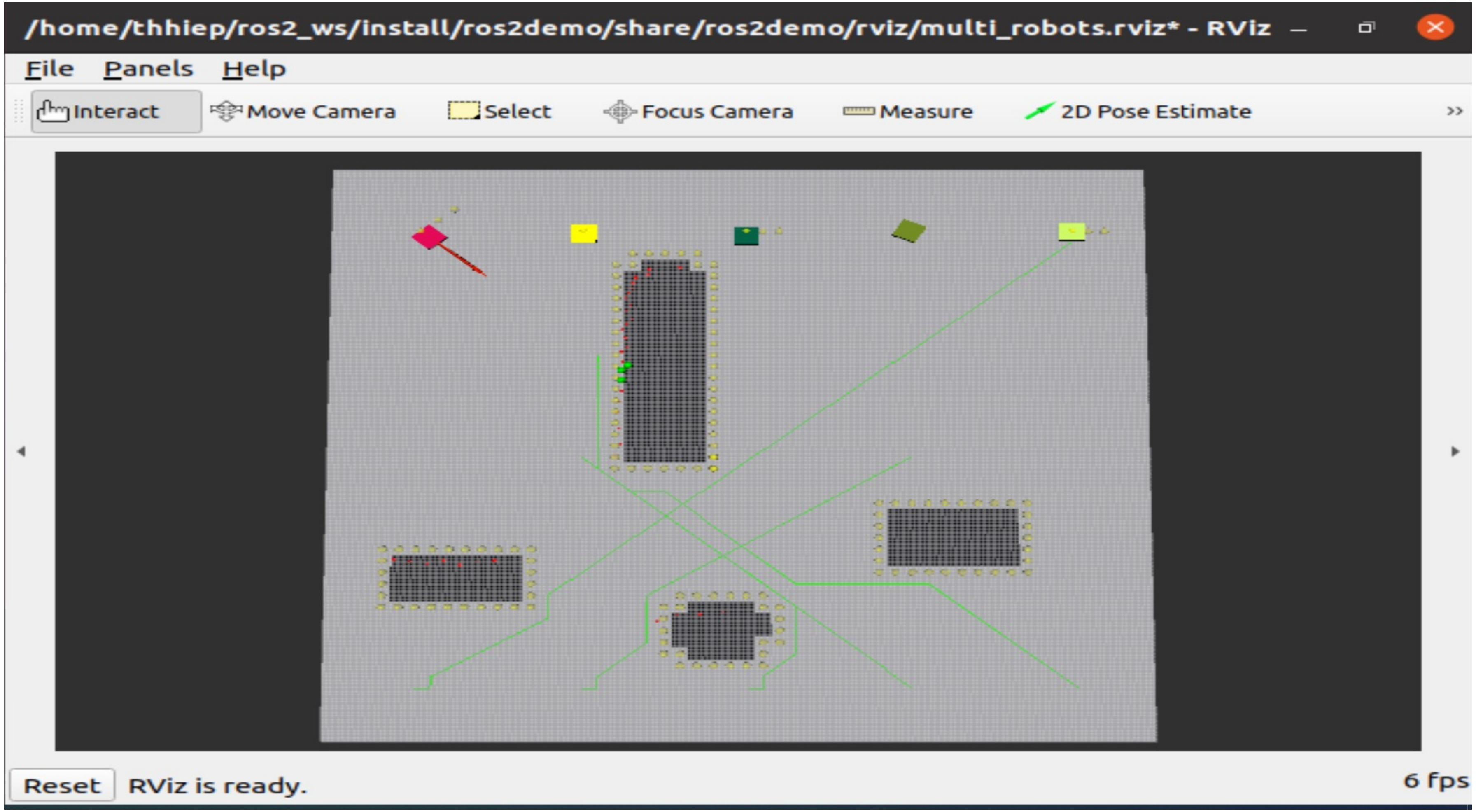
Date: *January 18th, 2024*

Mälardalen University

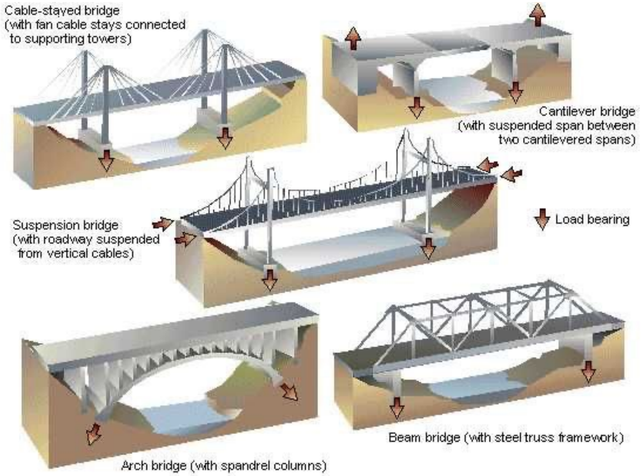
Appetizer: Rebeca model checker



Appetizer: ROS2 simulation with 5 robots



Robotics domain



Several types of bridges
Microsoft Corporation

- 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

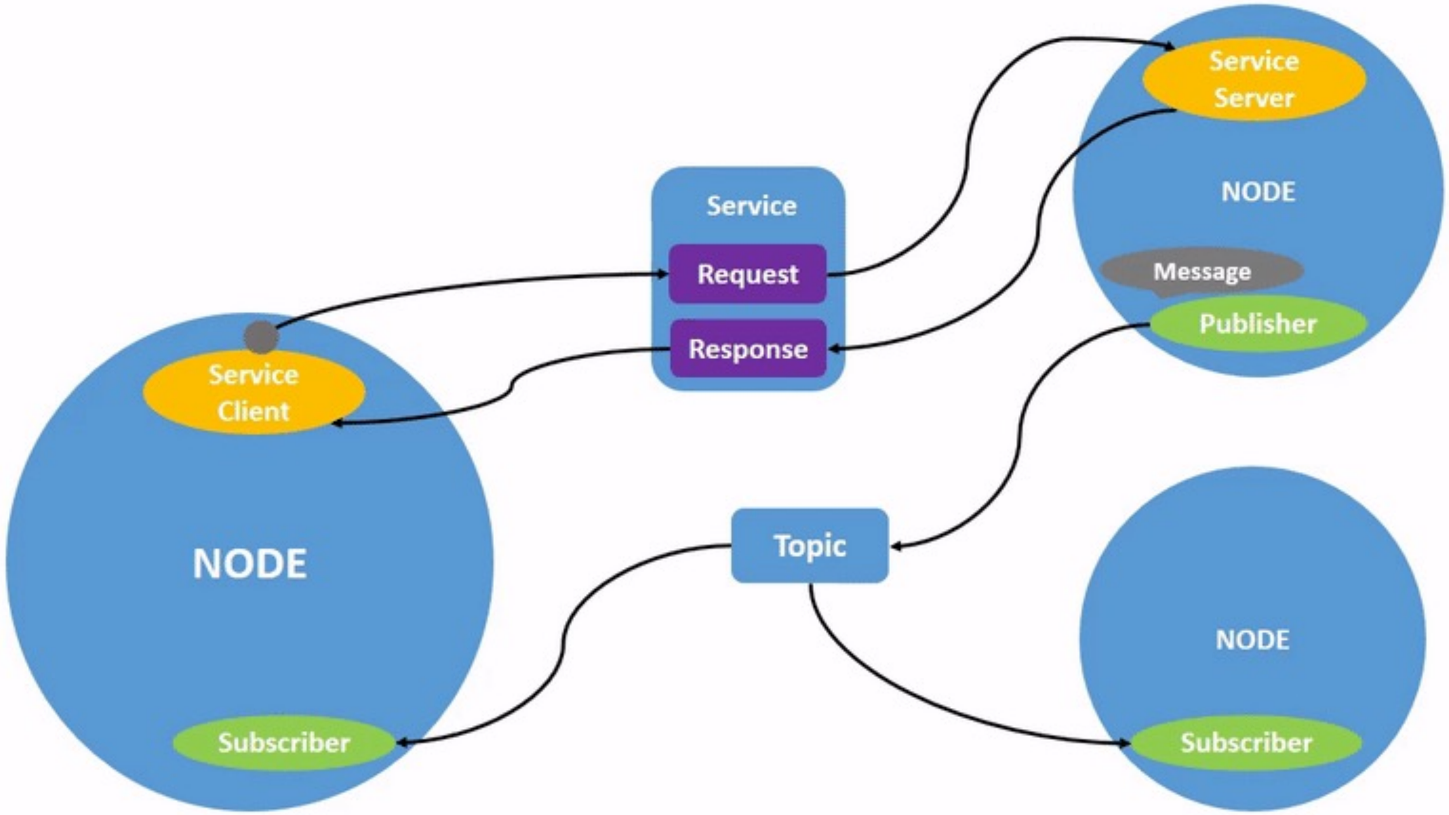
Component-based modelling & dev.

- [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

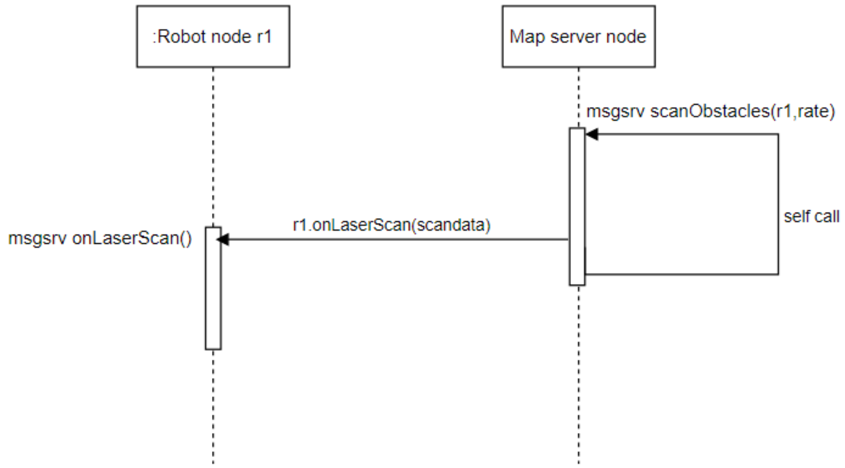
ROS2 architecture – node topography

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



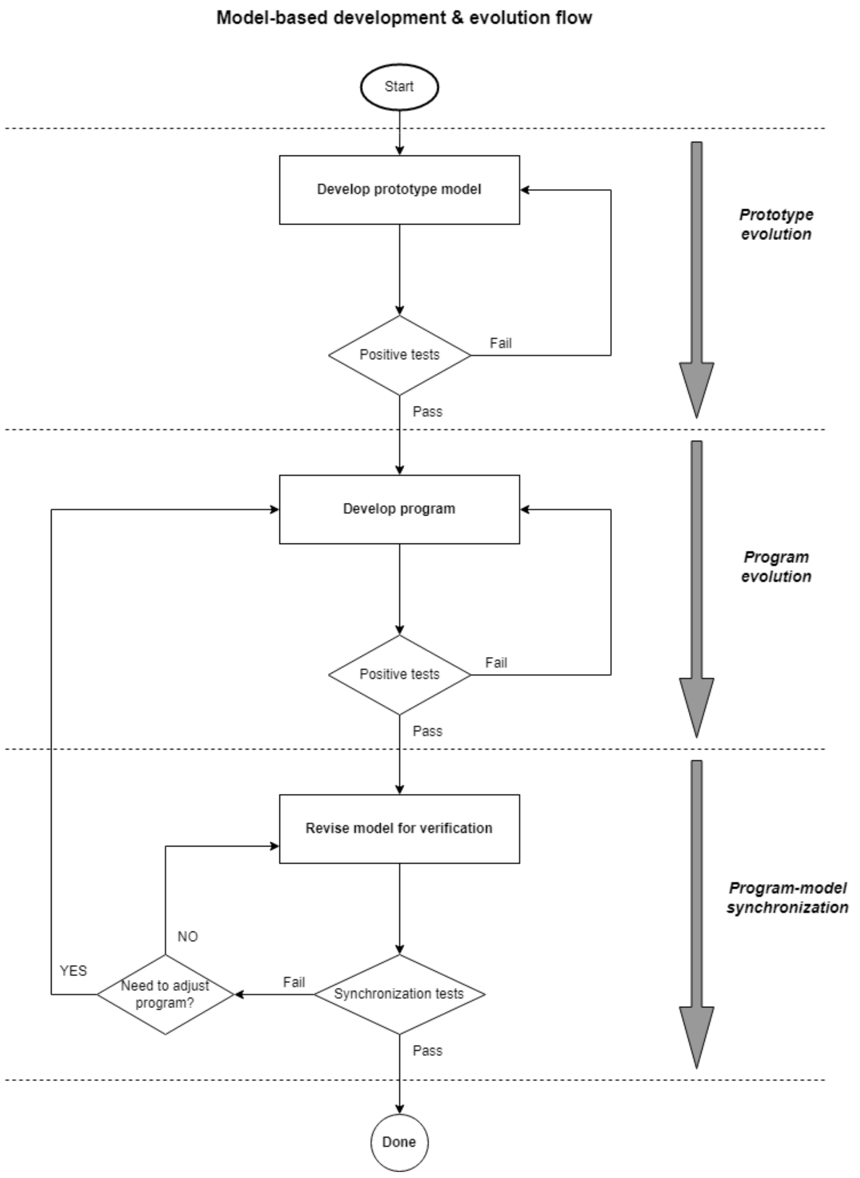
Timed Rebeca

- 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



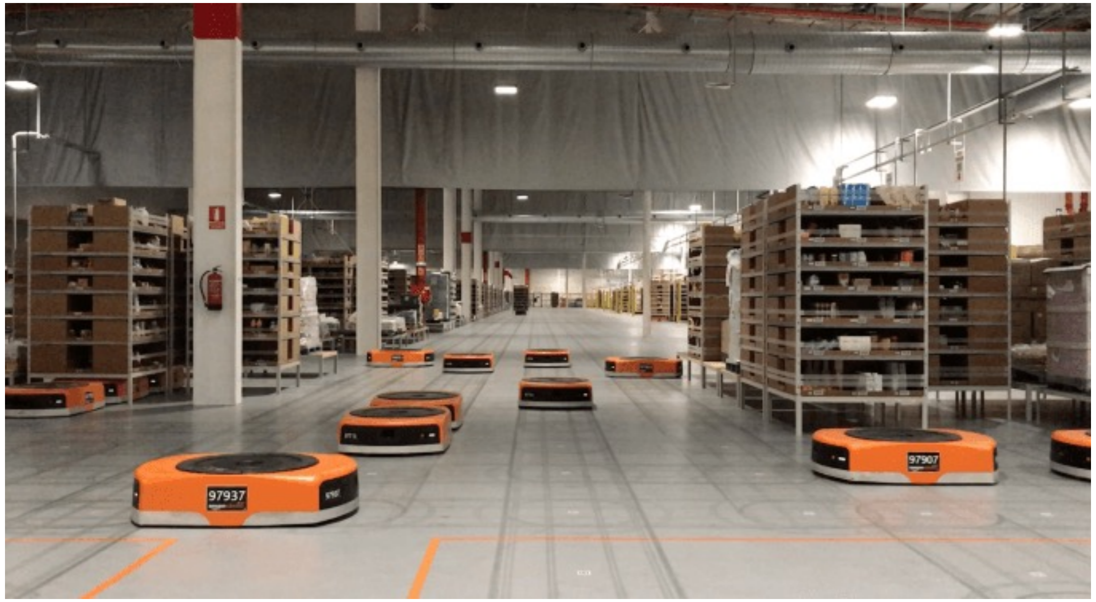
The flow

- 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)

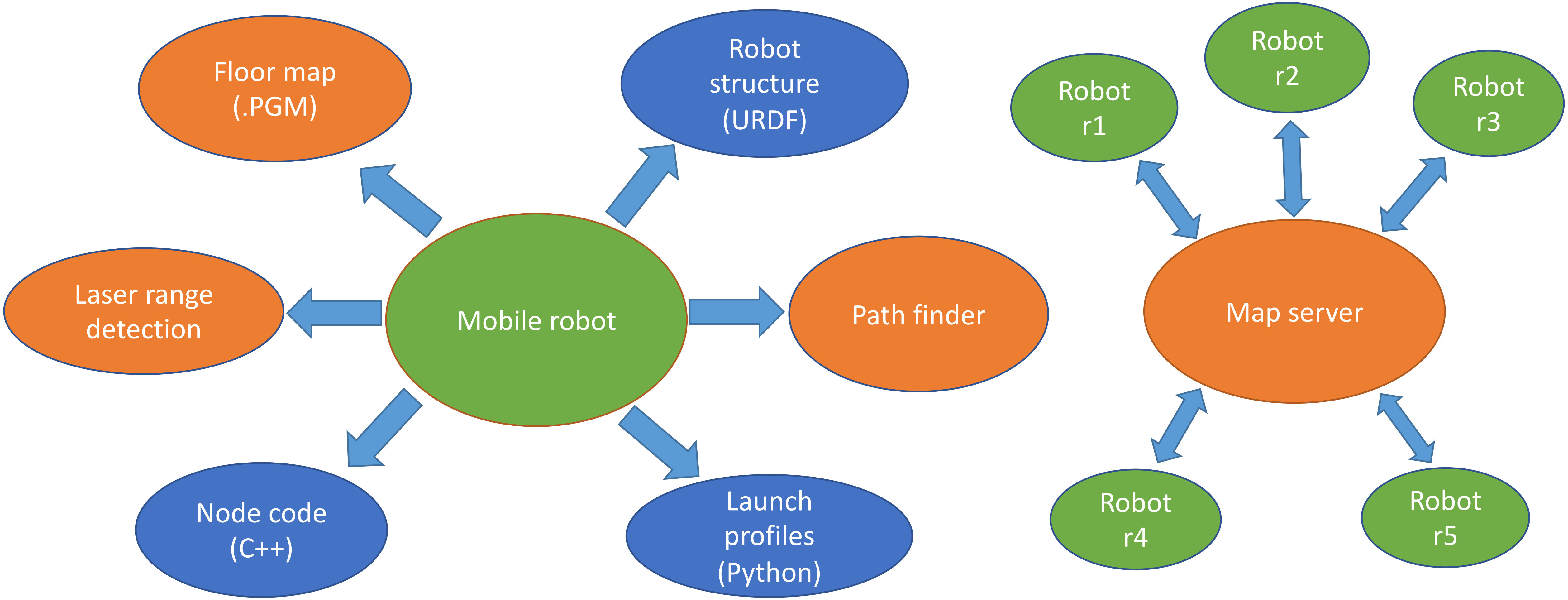


Industrial robotic problem: multiple AMRs

- 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

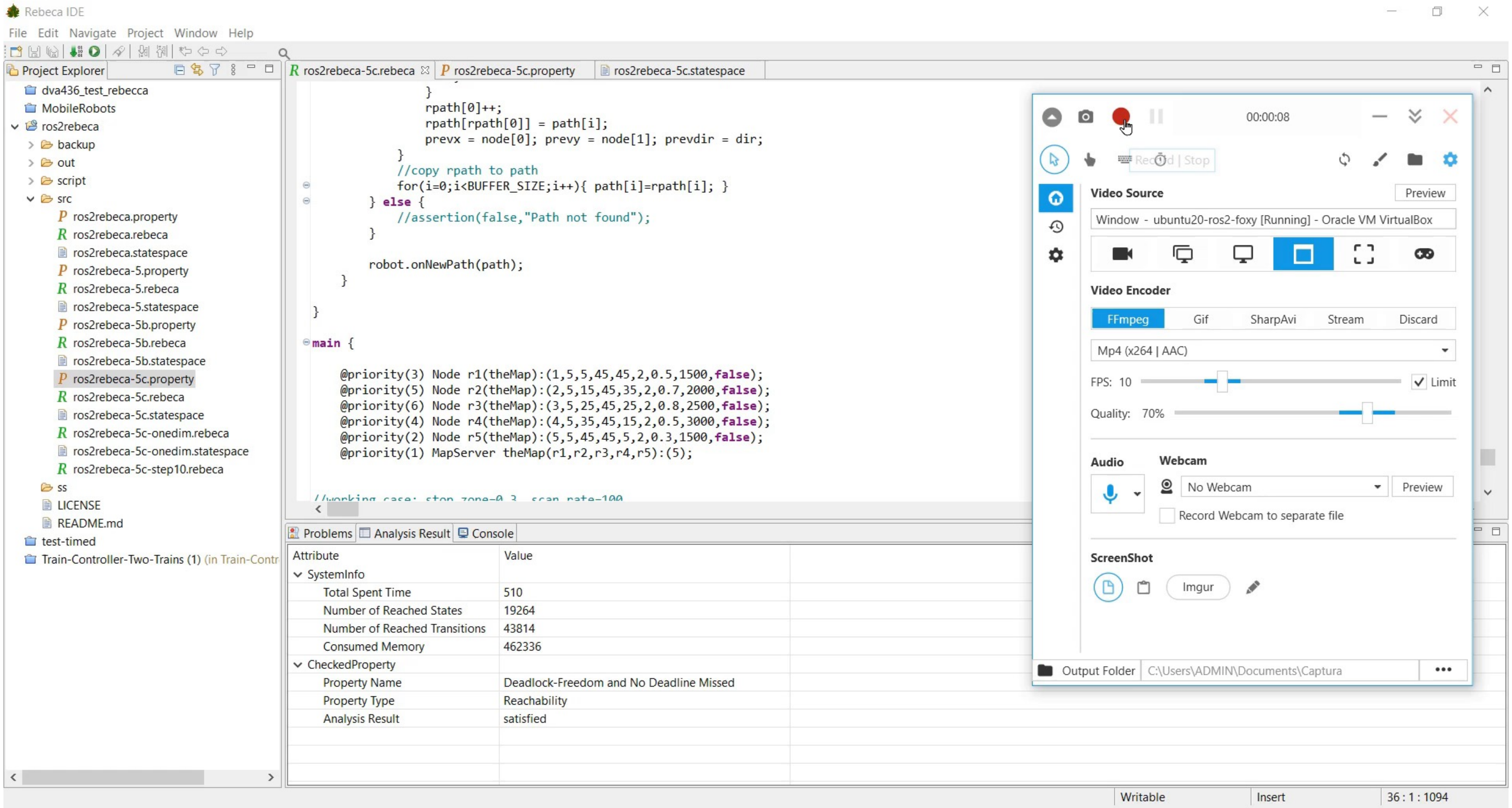


Results: working & non-working

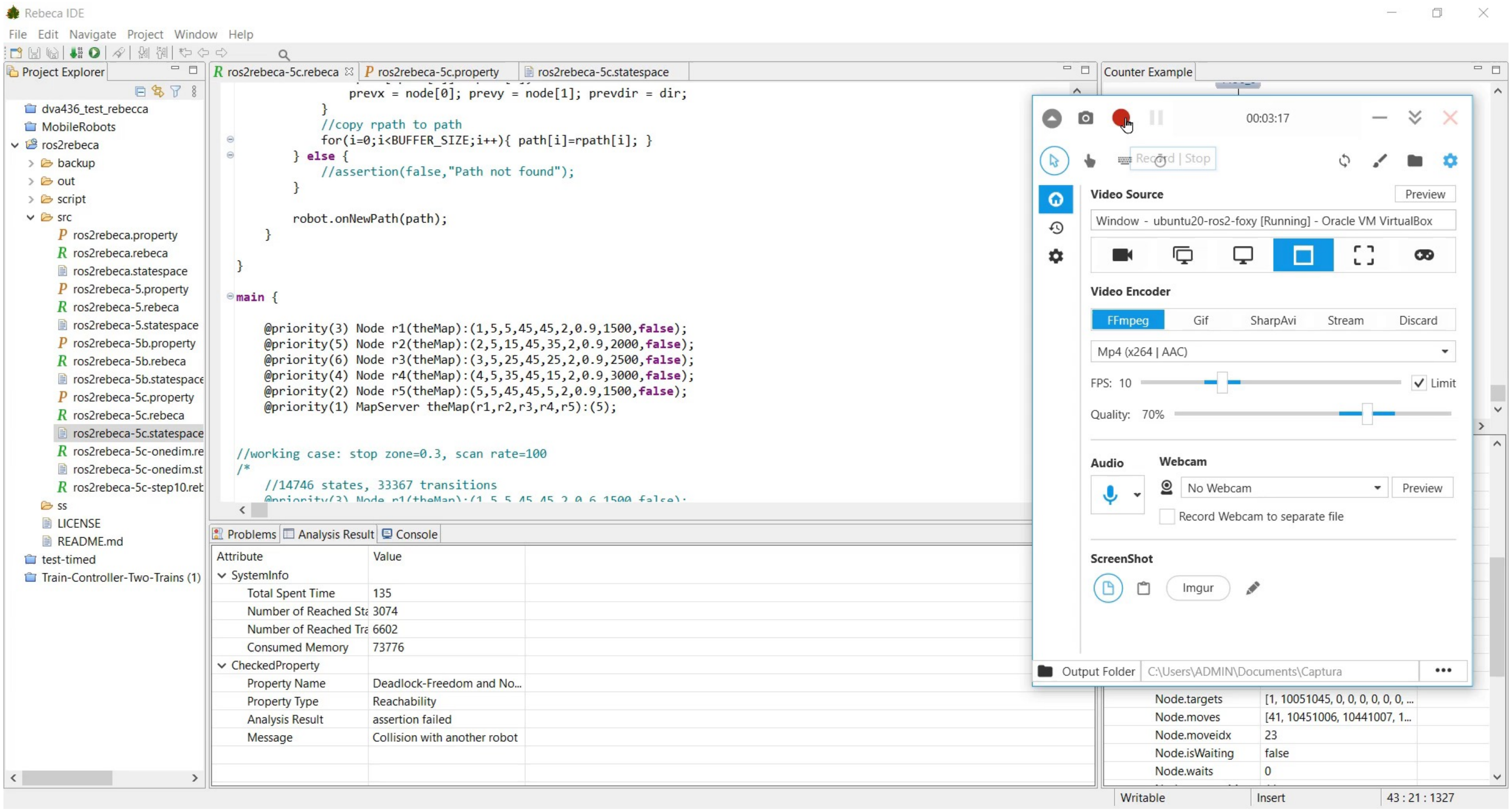
Parameters	Case C1a		Case C1b		Case C1c	
Scan rate	100		100		100	
Speed limit	1.275		1.275		1.275	
Stop zone	0.3		0.3		0.3	
Robots	***Speed***	***Wait***	***Speed***	***Wait***	***Speed***	***Wait***
R1	0.5	1500	0.3	1500	0.4	1500
R2	0.7	2000	0.3	2000	0.5	2000
R3	0.8	2500	0.3	2500	0.7	2500
R4	0.5	3000	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	
Robot	***Speed***	***Wait***	***Speed***	***Wait***	***Speed***	***Wait***
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



Results – a non-working case



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

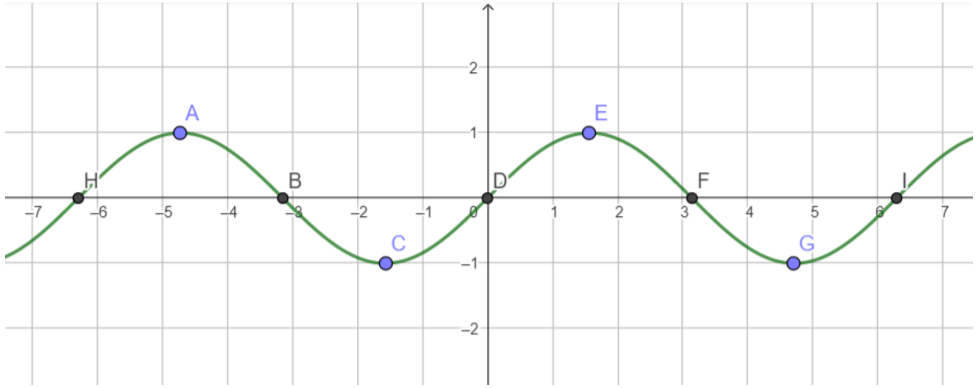
Challenges

- 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?

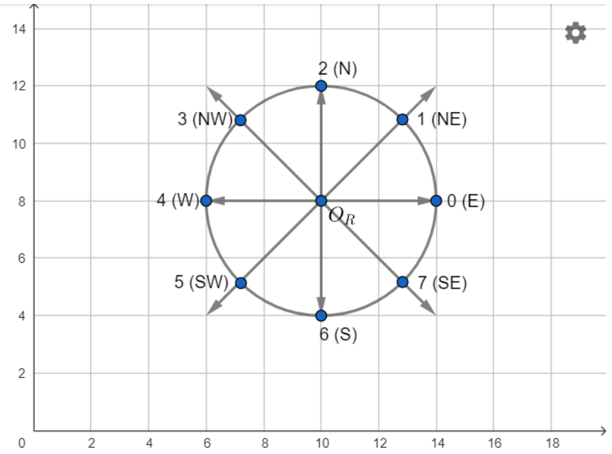
→Discretization strategies



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

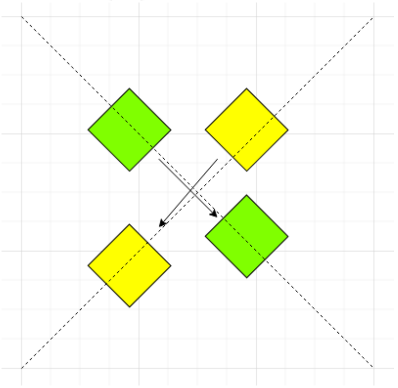
→ Simplification strategy (component-wise, retaining system integrity)

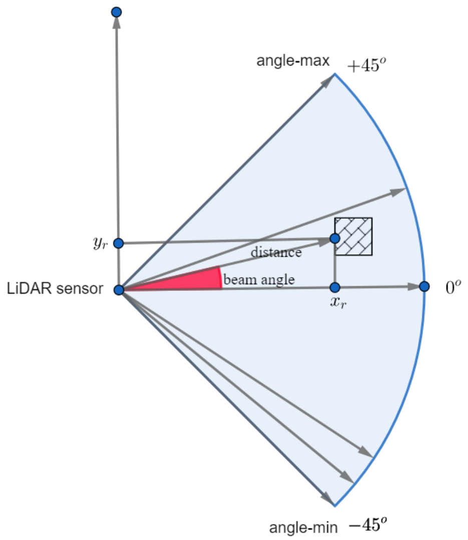
The Bad & counter-tactics



- Discretize:
  - 2D projection → occupancy grid, footprints/shadows
  - Robot directions: 8 angles
  - Scan step = 2º → 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

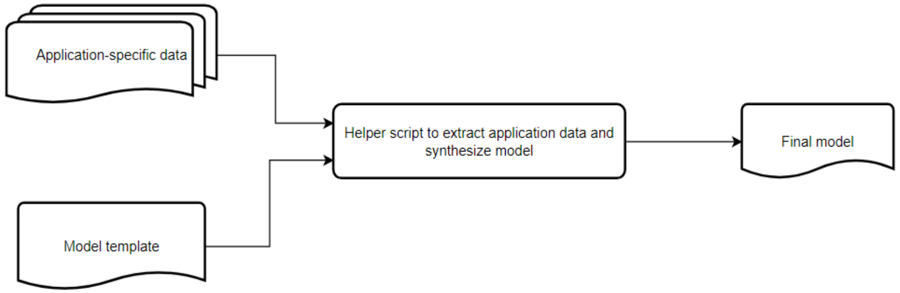




The Ugly - inconveniences

- 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* $Euclide$ 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