# STATECHARTS MODELLING OF TANK WARS

[*Reading Presentation*]

School of Computer Science
McGill University, Montréal, Canada

**COMP 763 / Silvia Mur Blanch**

# Overview



- Introduction
- Context and goals
- Modelling the “game characters”
- Mapping to an execution platform
- Summary

# Acknowledgement



- Entirely based on paper :

*"Model-based Design of Computer-controlled Game Character Behaviour"*

by

**Jörg Kienzle, Alexandre Denault & Hans Vangheluwe**

and their homonym presentation given at the

**10th MoDELS Conference**

5 October 2007

Nashville, TN

# Overview



- Introduction
- Context and goals
- Modelling the “game characters”
- Mapping to an execution platform
- Summary

# About EA Tank Wars competition



- Develop an AI for a tank that lives in a 2D world
- Simulation environment written in C++
- Manage:
  - Resources of the tank (health, fuel)
  - Map out the world
  - Find the enemy tank
  - Destroy the enemy tank

# Computer games nowadays



- Computer games sales have experienced a huge growth in the past few years
- Demand for higher realism in computer games leads to development of better AIs
- Necessity for reusability
- AI specification should be done at a higher level of abstraction: not coded, but modeled!
- Models define reactions to game events, so event-based formalisms seem to be the most appropriate

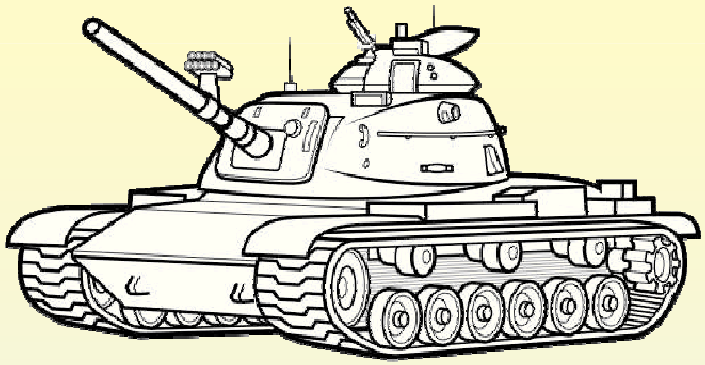
# Overview

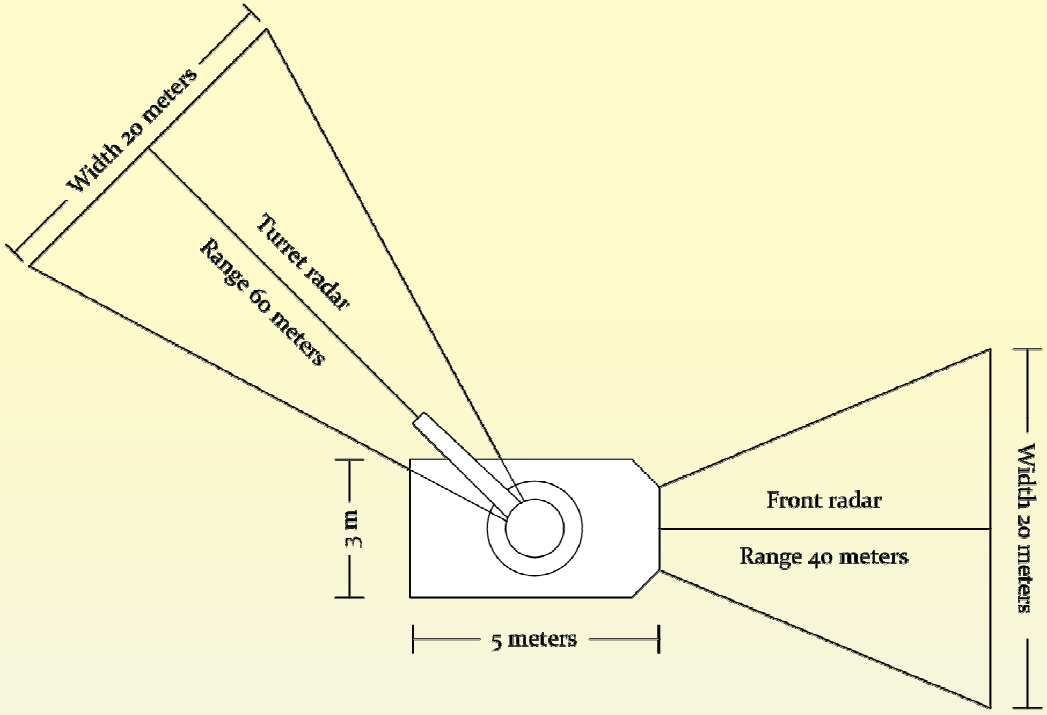


- Introduction
- Context and goals
- Modelling the “game characters”
- Mapping to an execution platform
- Summary

# Abstraction

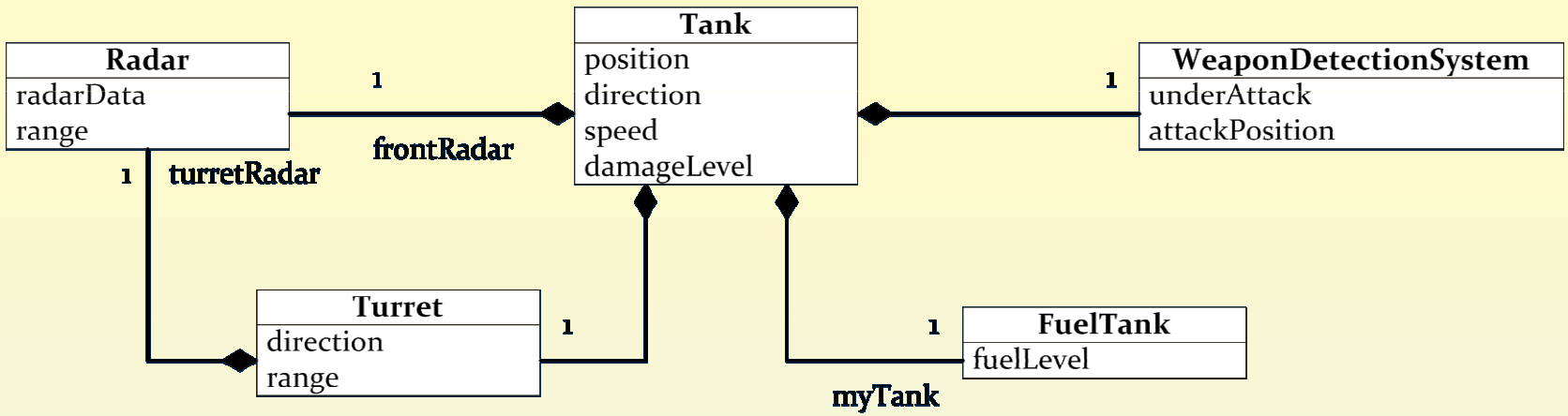






# Abstraction: Tank structure (state)





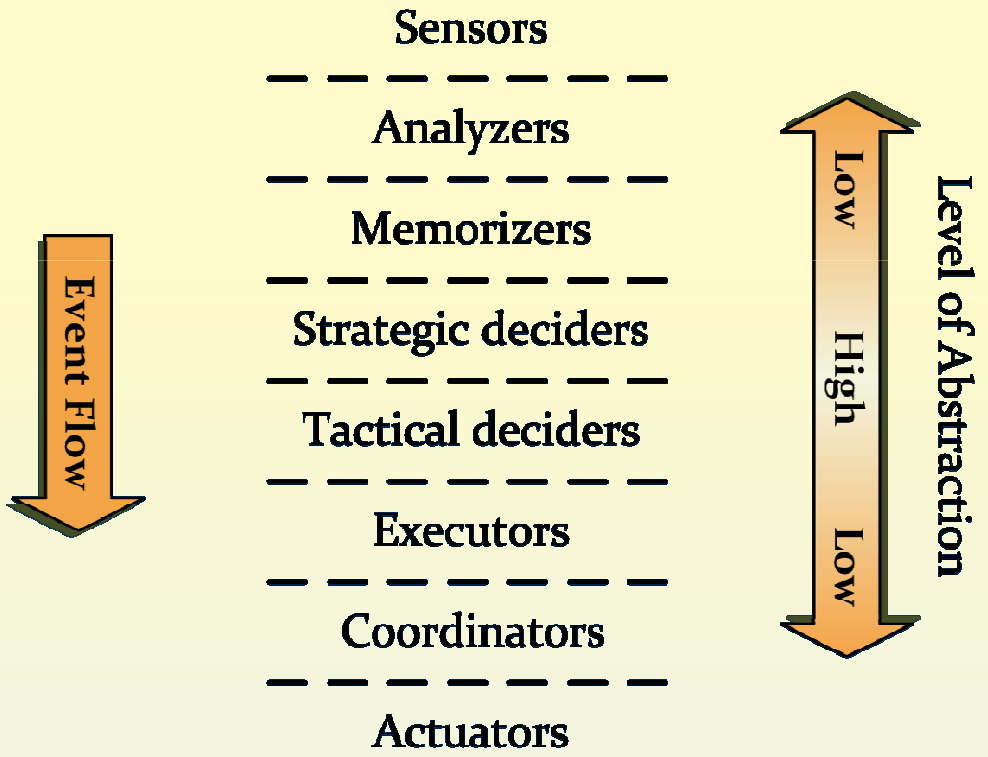
# Optimal formalism(s) to model detailed behavior?



- Time-sliced Vs. Event-based approach (more abstract)
- Individual objects' behavior:
  - State/event based
  - Autonomous/reactive behavior
  - Notion of time
  - Modularity
  - Well known
  - Availability of simulators/code generators

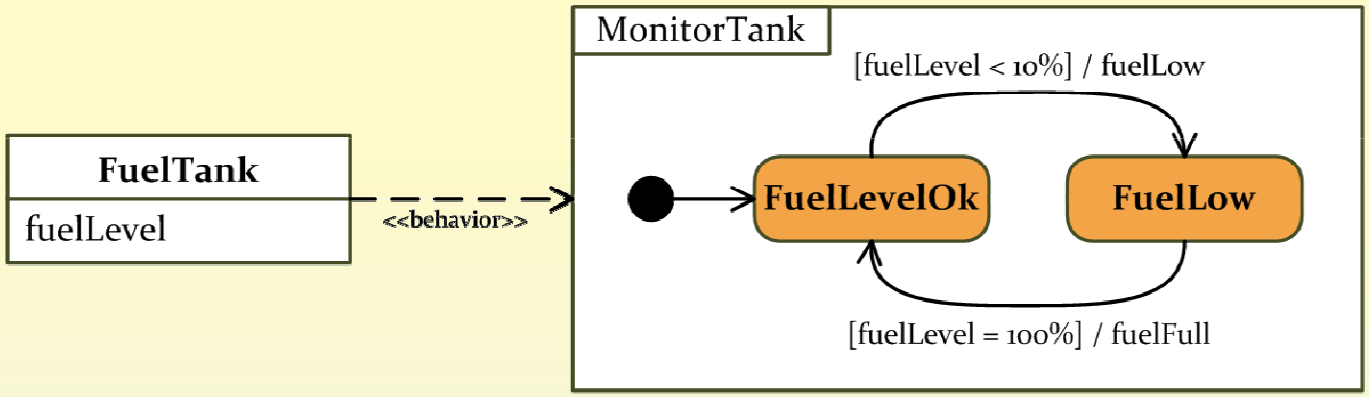
# Abstraction: Tank behavior (high-level)





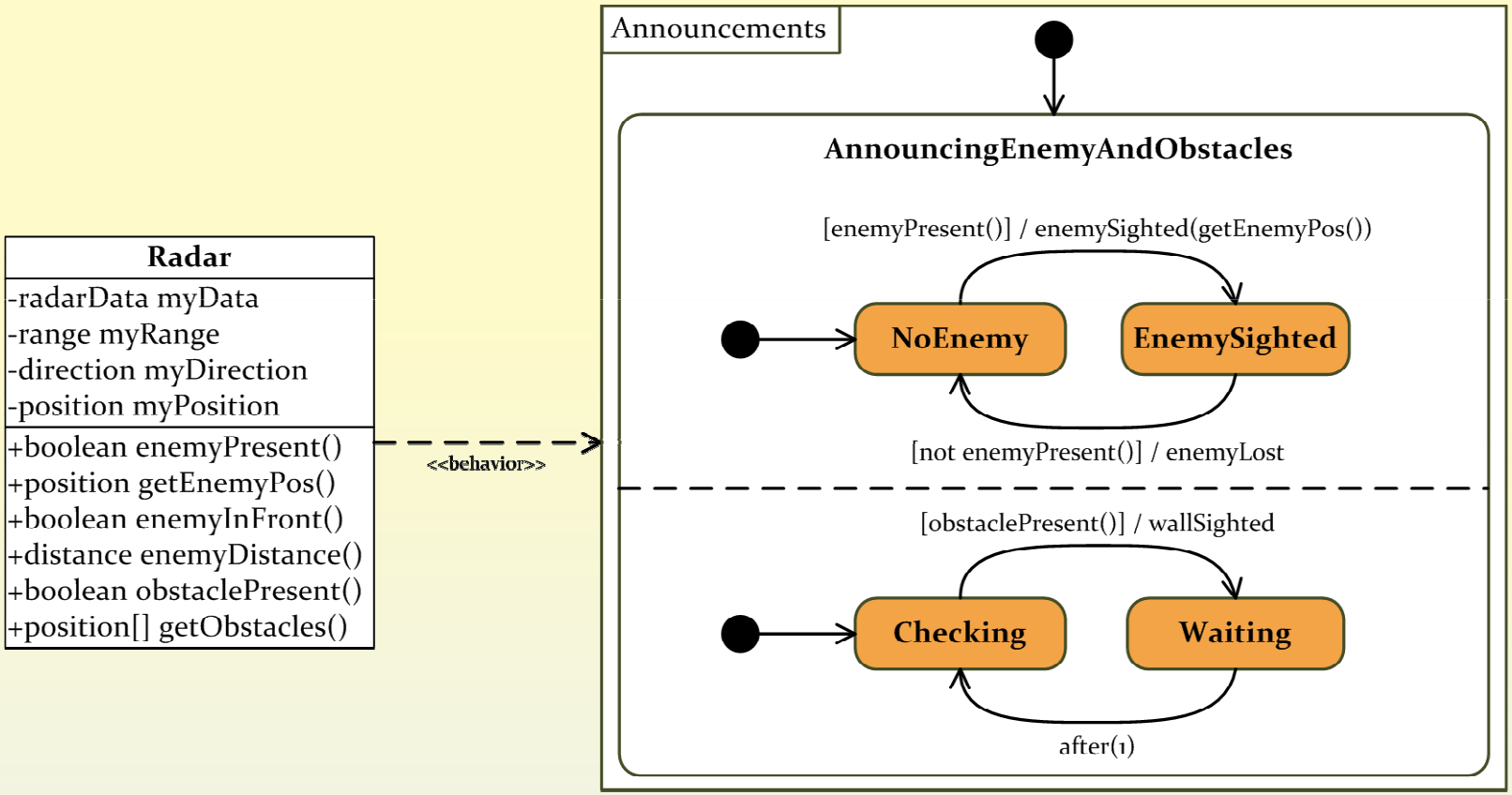
# Sensors – Generating important game events





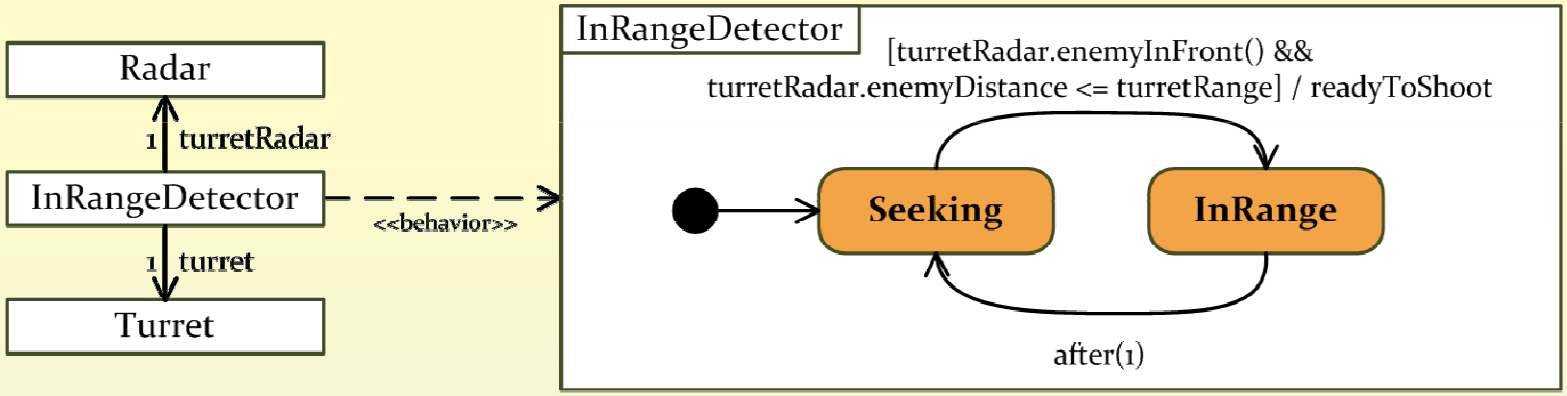
# Sensors – Generating important game events





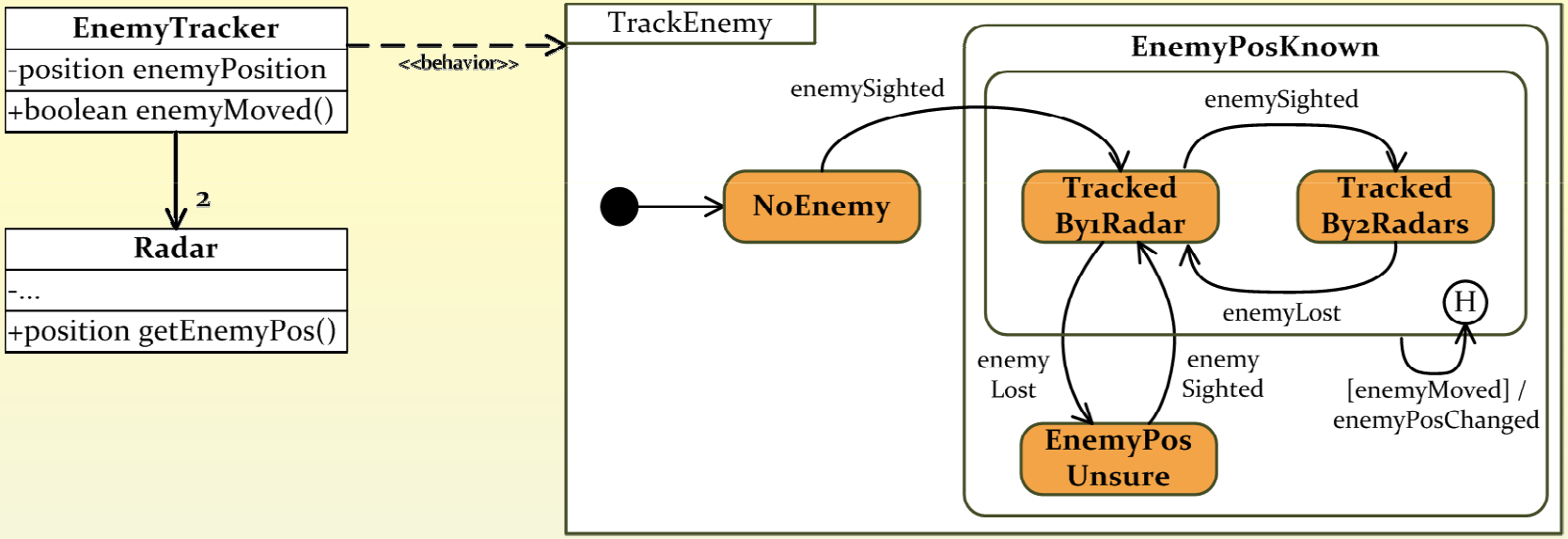
# Analyzers – Correlating sensor events





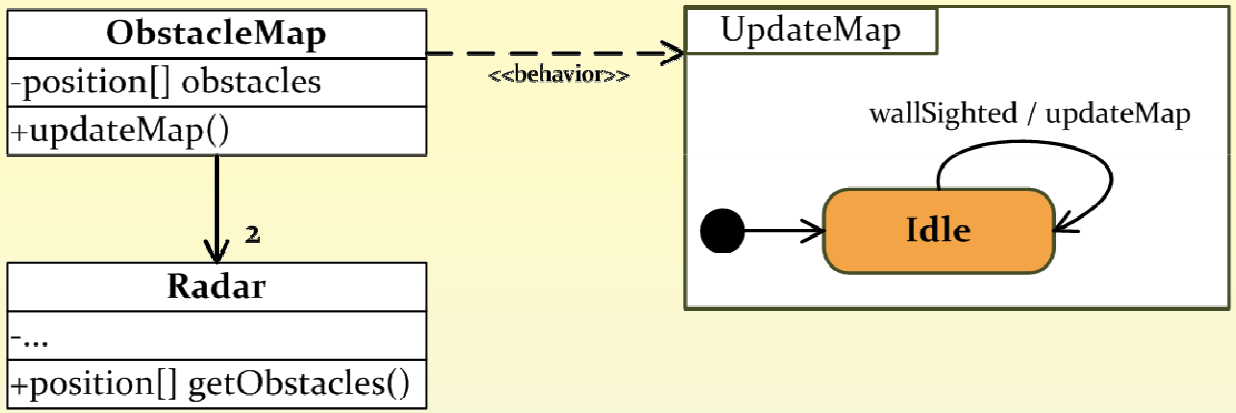
# Memorizers – Modelling memory





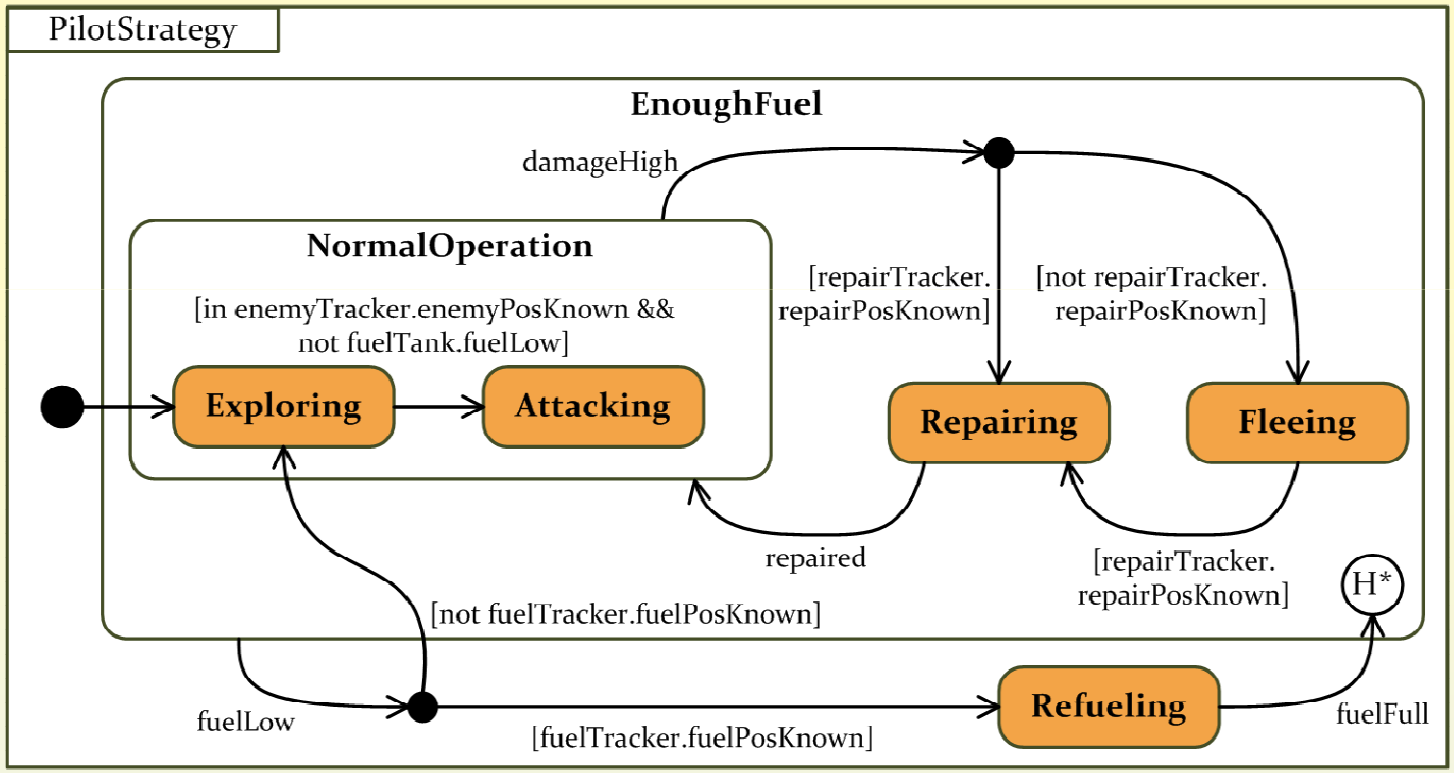
# Memorizers – Modelling memory





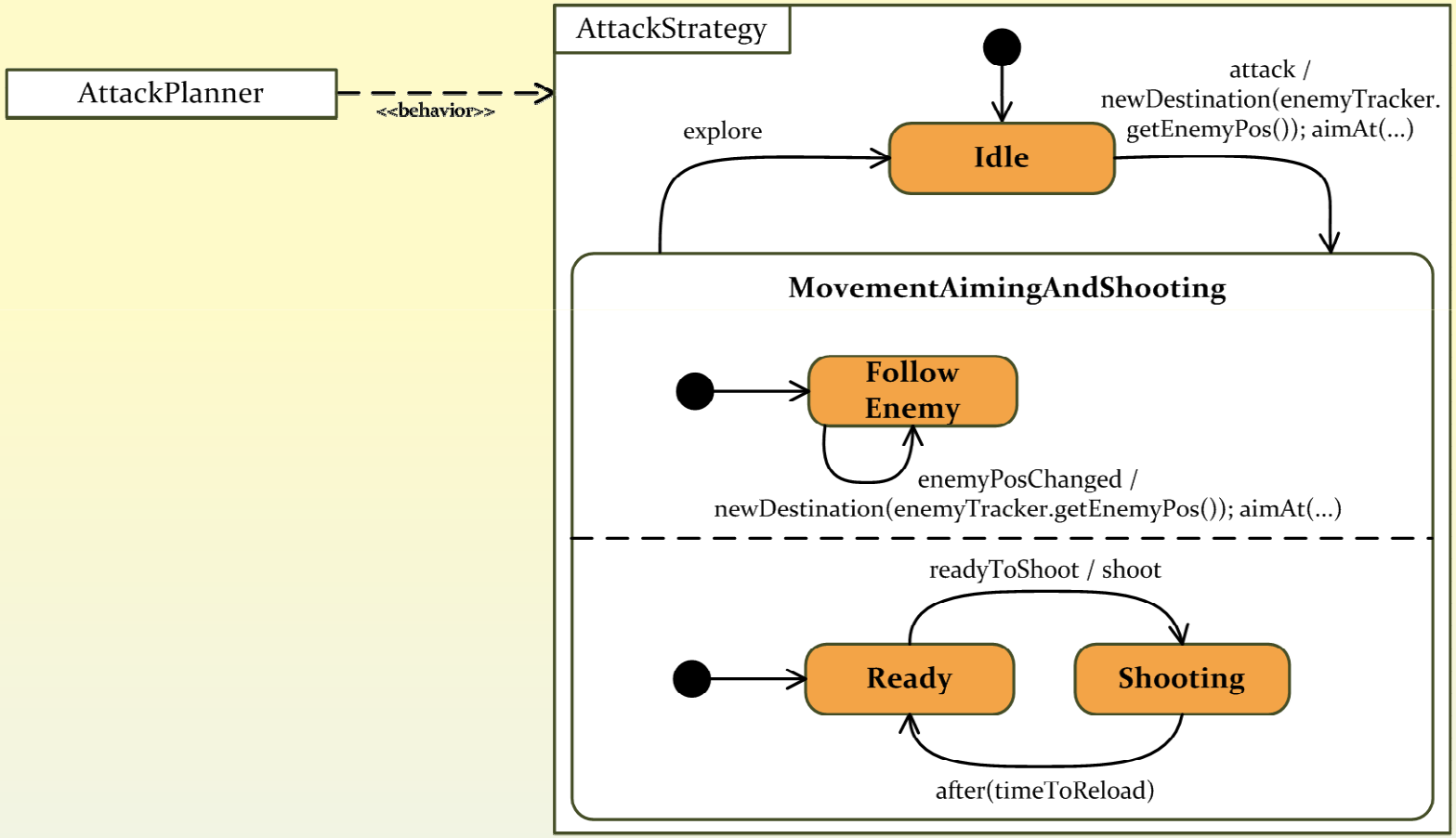
# Strategic deciders – High-level goals





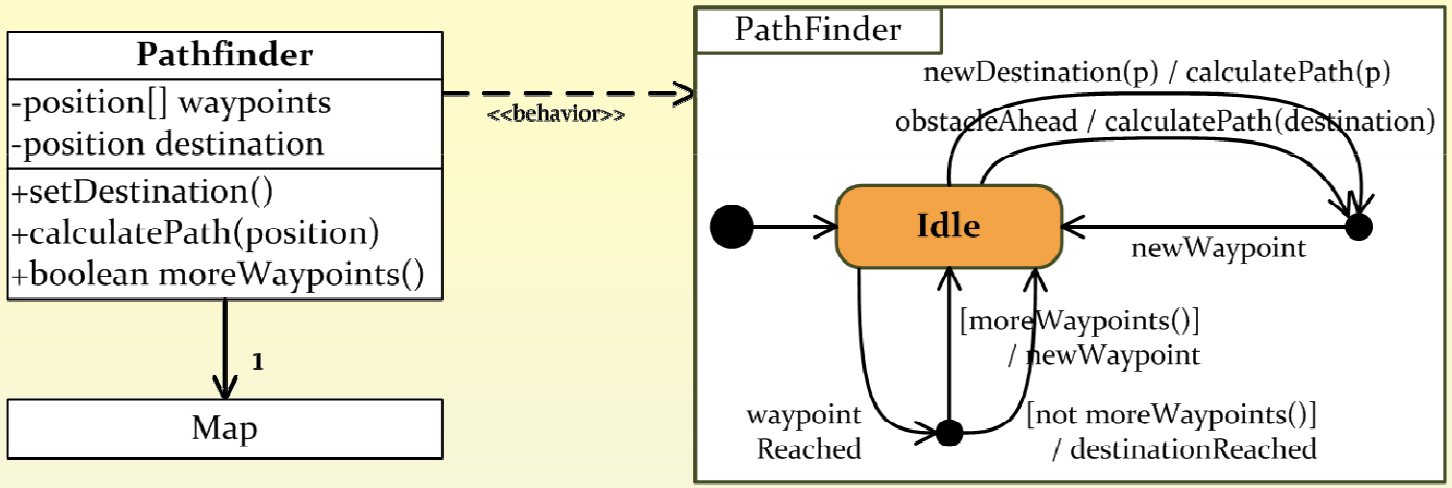
# Tactical deciders – Planning how to achieve goal





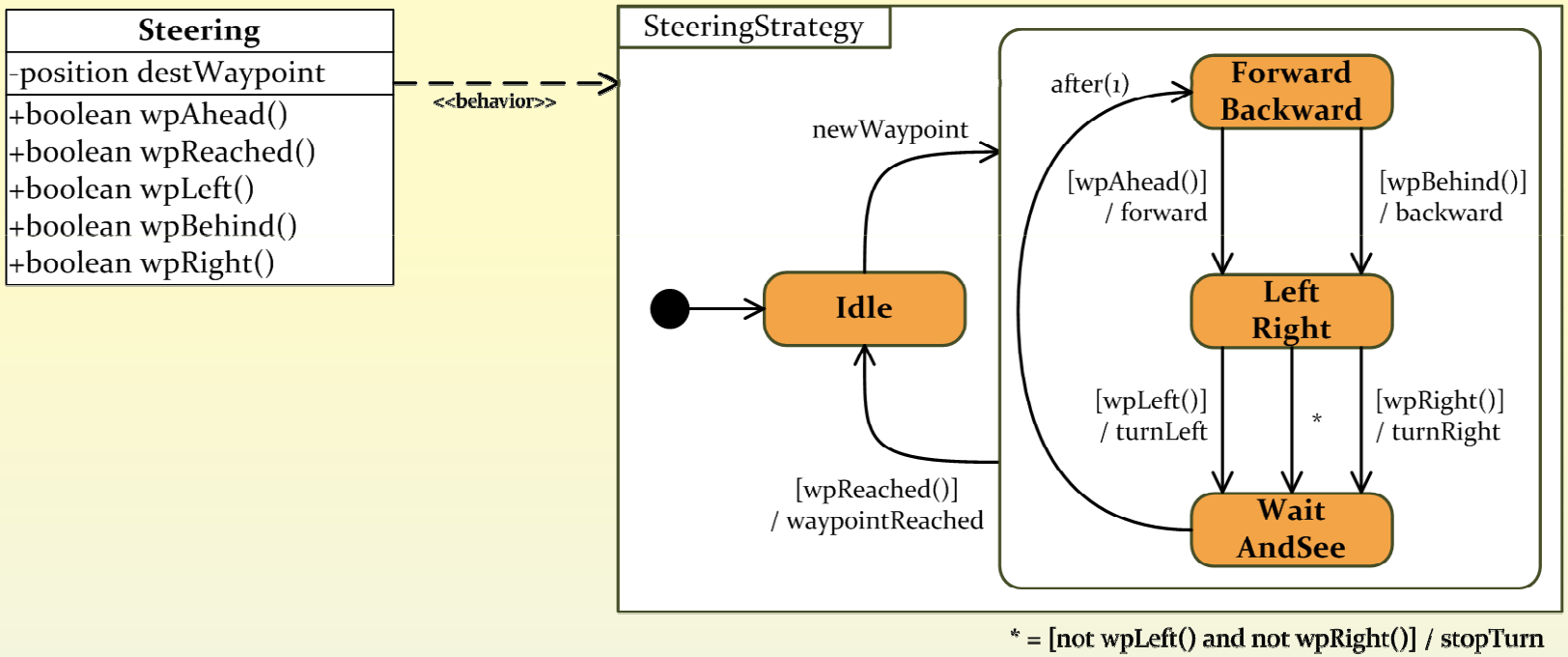
# Tactical deciders – Planning how to achieve goal





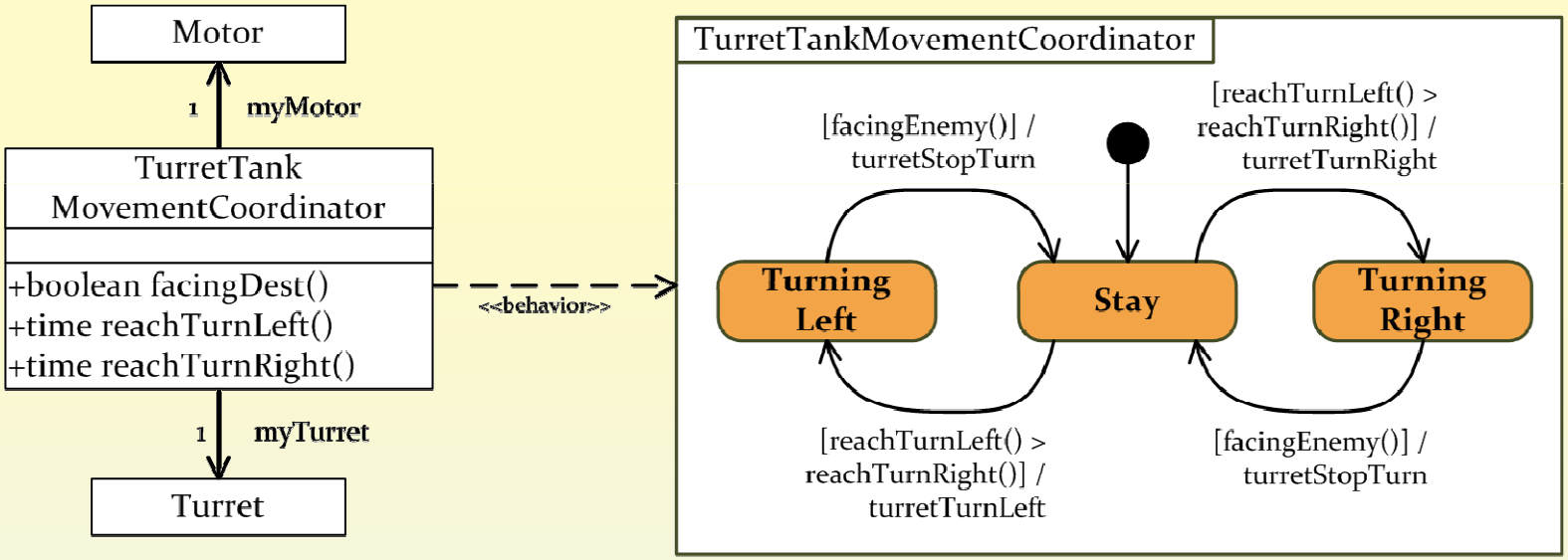
# Executors – Mapping decisions to actuator commands





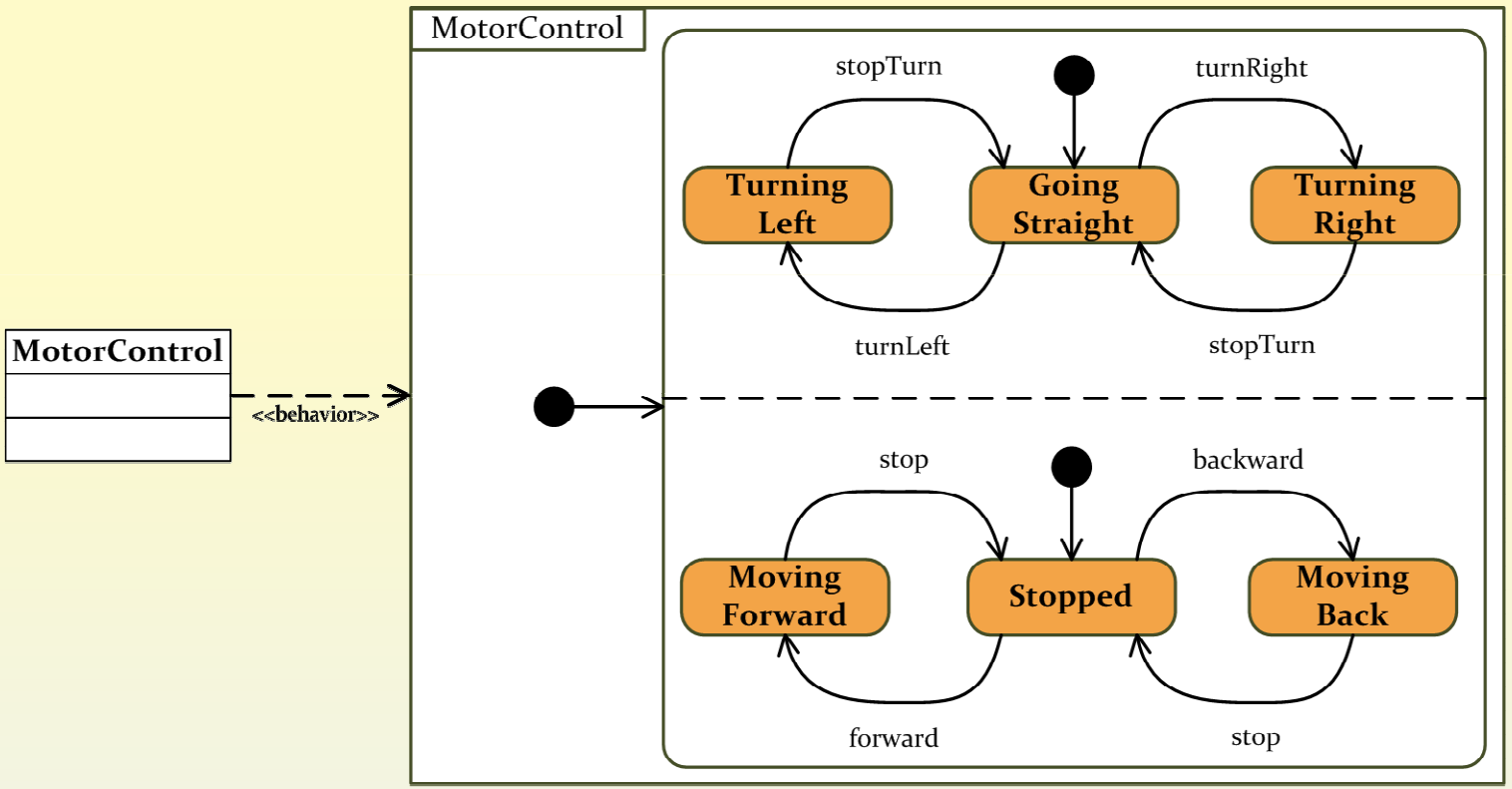
# Coordinators – Resolving actuator interactions





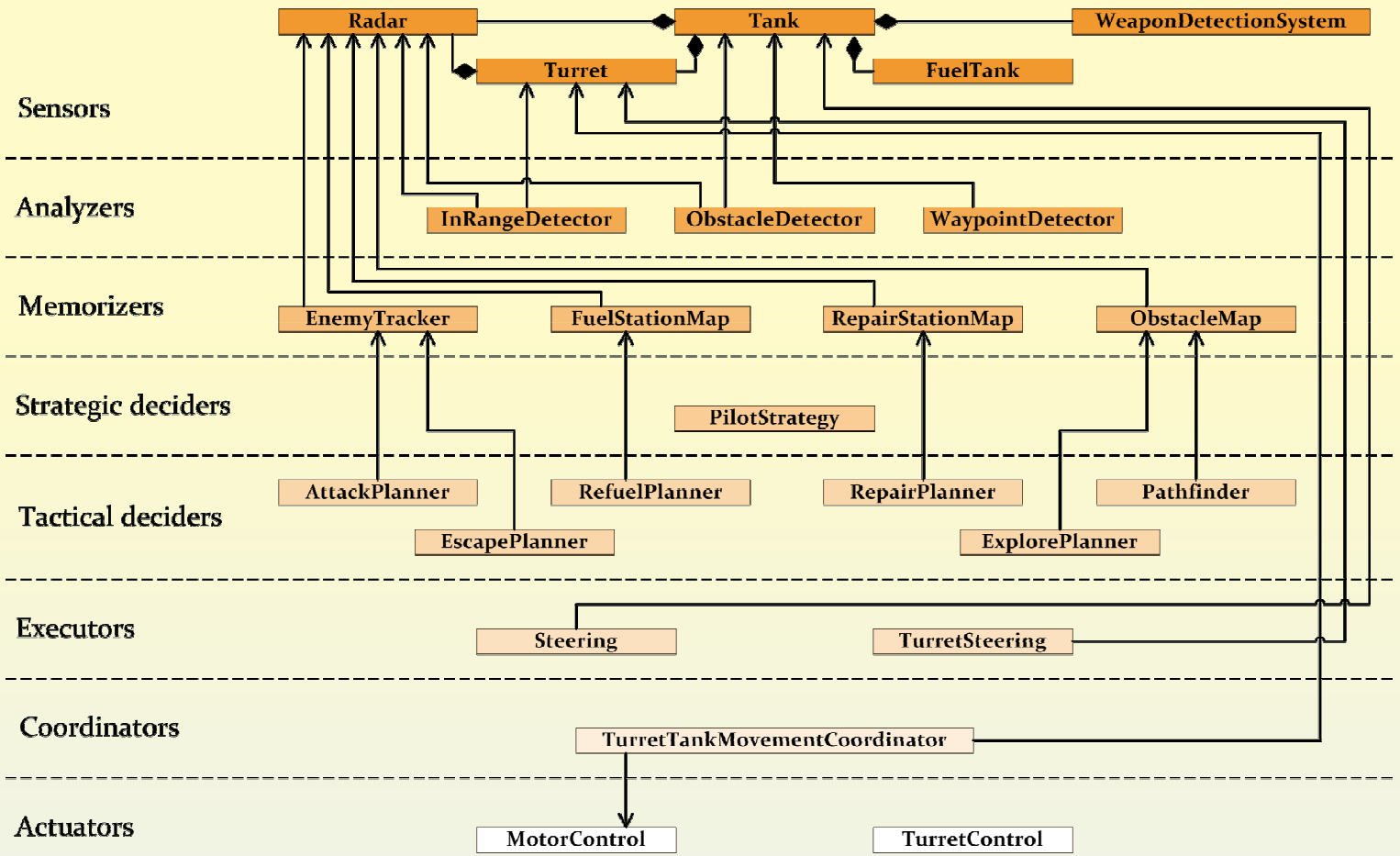
# Actuators – Influencing the game state





# Putting it all together: (a)synchronous communication

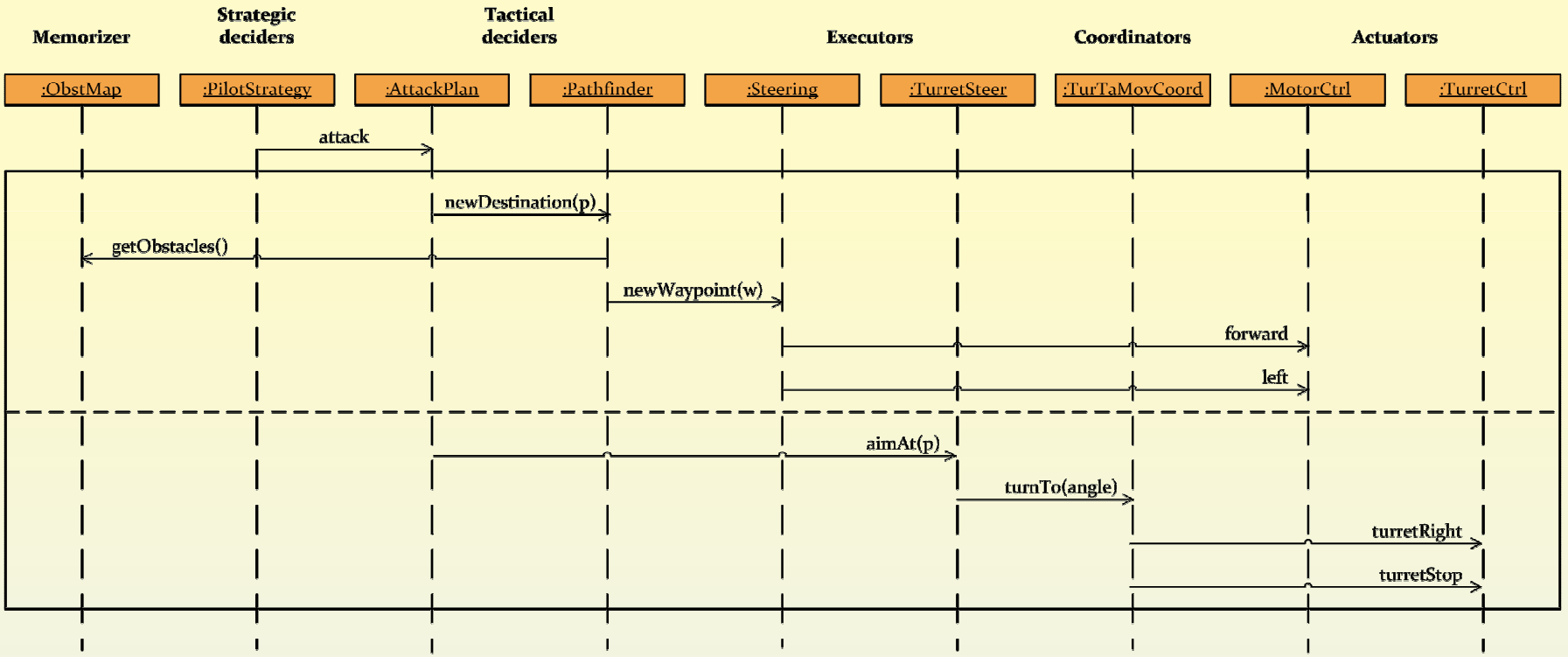





Statecharts modelling of Tank Wars

# Event sequence in case of attack





# Overview



- Introduction
- Context and goals
- Modelling the “game characters”
- Mapping to an execution platform
- Summary

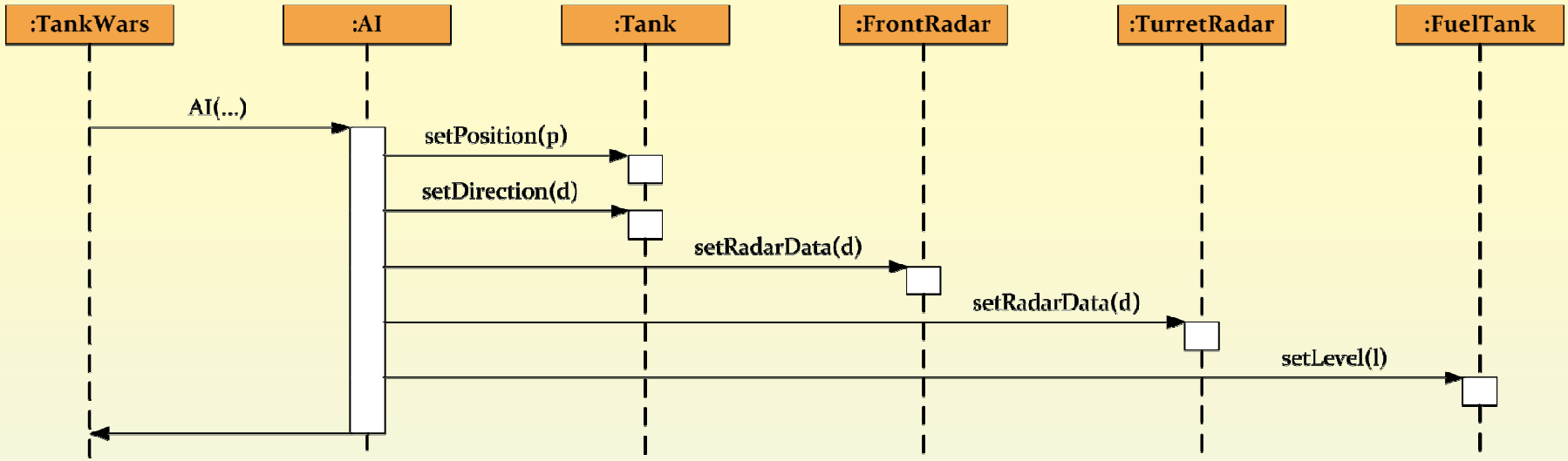
# Time-slicing Vs. Continuous time



- In Tank Wars simulation, new state of the environment is sent to the AI component every 50ms
- Continuous time is most general and most appropriate at this level of abstraction because:
  - Modelling freedom: modeller not unnecessarily constrained by implementation issues, can focus on the logic of the model
  - Symbolic analysis: analyze model using timed logic
  - Simulation accuracy: due to real numbers

# Time-slicing to Event-driven





# Overview

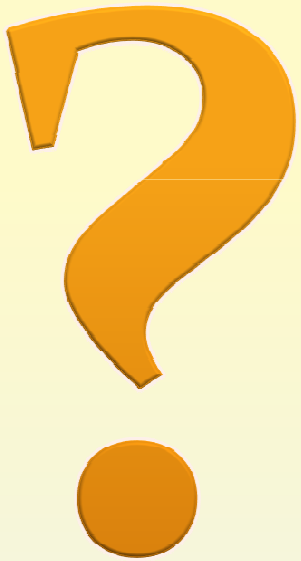


- Introduction
- Context and goals
- Modelling the “game characters”
- Mapping to an execution platform
- Summary

# Summary



- Appropriate level of abstraction:
  - Events
  - Layers
  - Data: filtering
- Appropriate formalism(s):
  - State/event-based
  - Autonomous/reactive behavior
  - Notion of time
  - Modularity
  - Well known
  - Availability of simulators/code generators



Statecharts modelling of Tank Wars