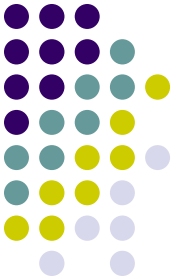


Welcome to SENG 480B / CSC 485A / CSC 586A Self-Adaptive and Self-Managing Systems

Dr. Hausi A. Müller
and Lorena Castañeda
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

Announcements

- Monday, July 6
  - Lorena Castañeda - Models at runtime
- Thursday, July 9
  - Hausi Müller
- Friday, July 10
  - Assignment 3 due
- Monday, July 13
  - Lorena Castañeda - Models at runtime
  - Assignment 3 demos (Time TBA)
- Thursday, July 16
  - Midterm

Reading Assignments

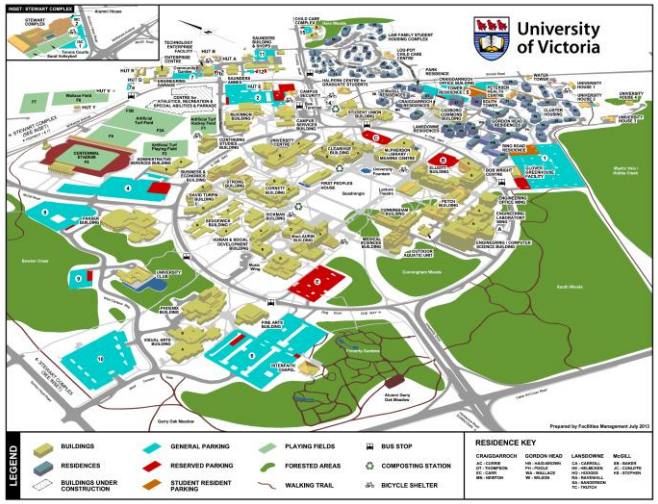
- "Models@run.time" Blair et al. 2009 http://dx.doi.org/10.1109/MC.2009.326
- "Models@run.time to Support Dynamic Adaptation" Morin et al. 2009 http://dx.doi.org/10.1109/MC.2009.327
- "The role of models@run.time in supporting on-the-fly interoperability" Bencomo et al. 2013 http://link.springer.com/article/10.1007%2Fs00607-012-0224-x
- "Living with uncertainty in the age of runtime models" Holger Giese et al. 2014 http://link.springer.com/chapter/10.1007%2F978-3-319-08915-7_3

Other reading material

●Models@run.time Foundations, Applications, and Roadmaps
Editors: Bencomo, N., France, R.B., Cheng, B.H.C., Aßmann, U.
http://www.springer.com/us/book/9783319089140

Definitions

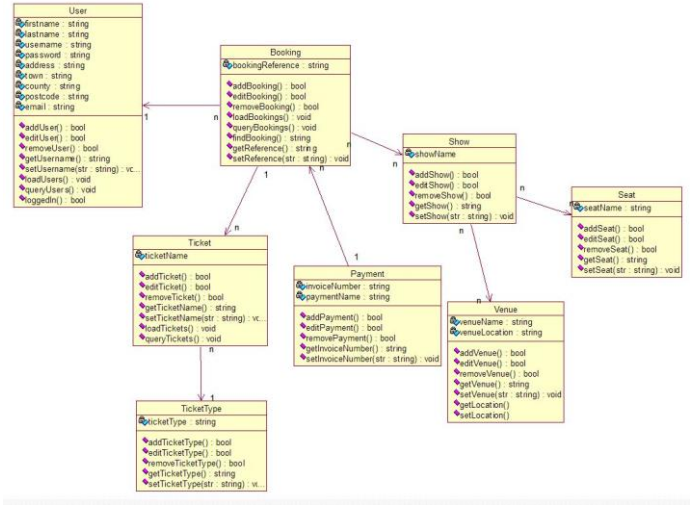
●A ***model*** is a form of representation of an original. It comprises three elements: the original (factual or envisioned), a purpose, and an abstraction function to map the model with the original [Giese 2014]

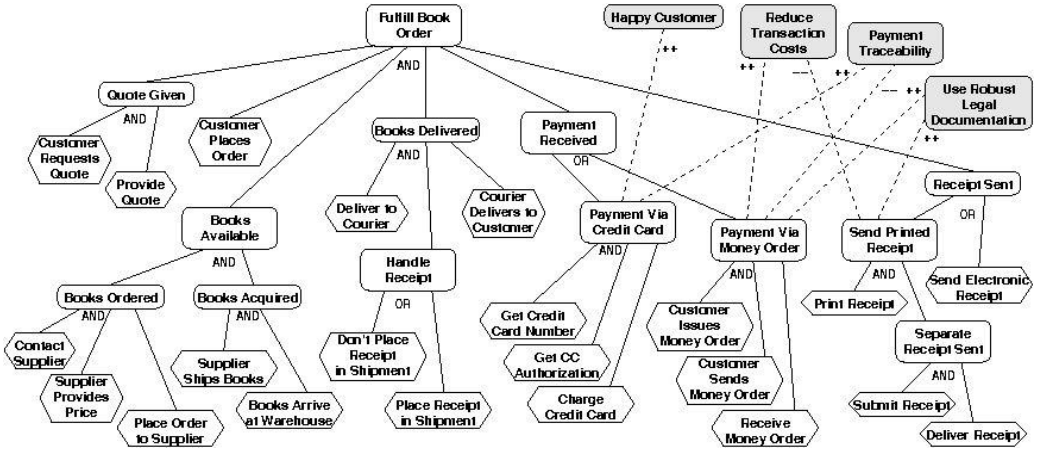


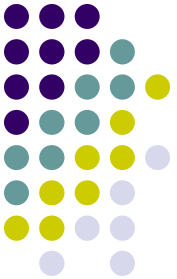
$$v = v_0 + at$$

Definitions

- A **software model** is an abstraction of a system often associated with design time activities such as documentation and analysis [Castaneda 2014]



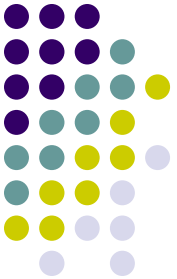




What are problems of software models in modern software?

Model at runtime - MART

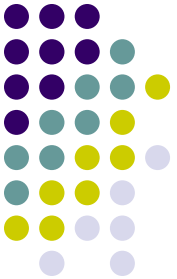
(models@run.time, runtime models)



Models at runtime (MART)

- represent the system's complete environment (possibly more than one MART for a system), up-to-date information (i.e., context, users, and requirements)
- is accessible at runtime by the system, available in the form of software artefacts
- the system must be causally connected, are implemented to support runtime events
- manipulable and capable to evolve during execution time

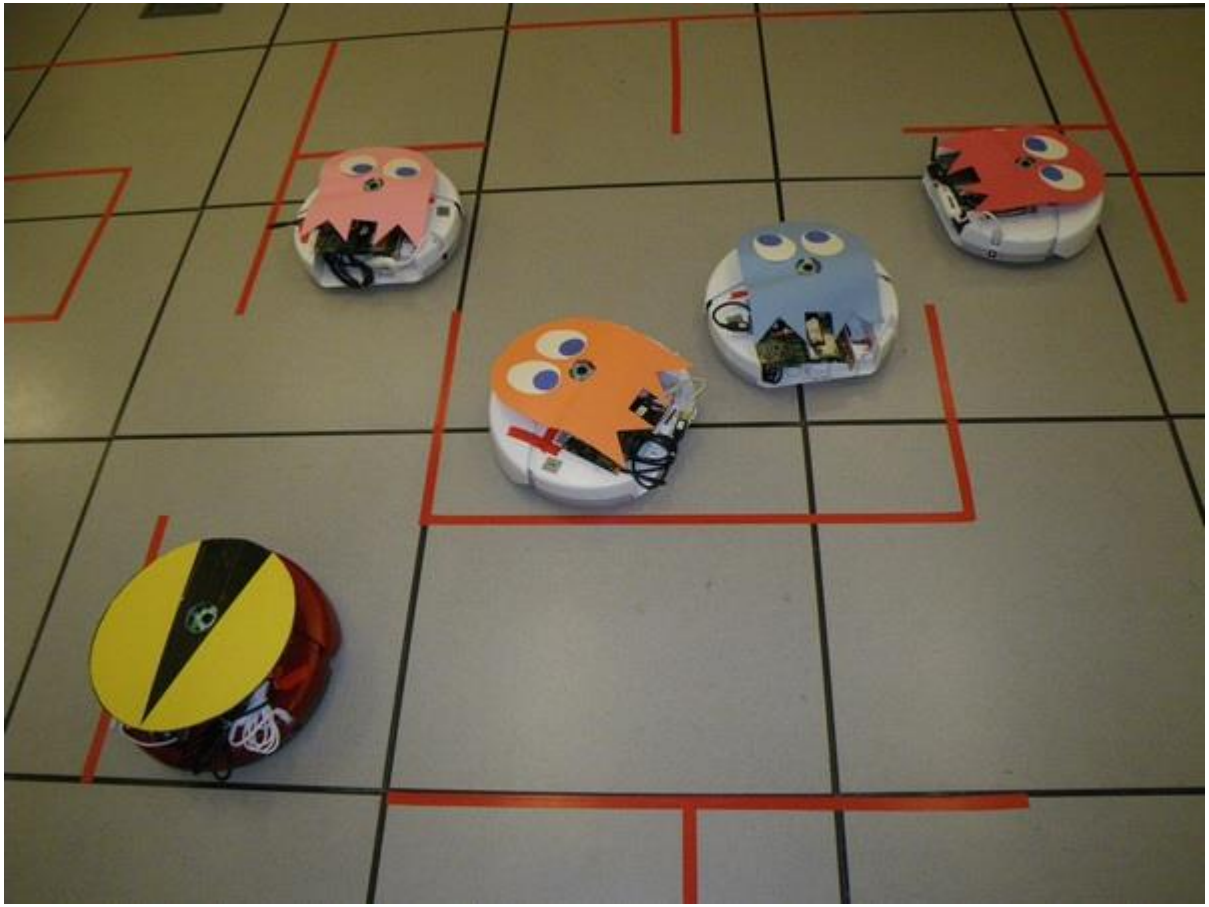
Why MART?



- MART are implemented to deal with runtime concerns for complex systems, such as self-adaptive software systems
- Various purposes including simulating runtime environments, monitoring, policy checking, error handling, and supporting systems adaptation requirements

Roomba Pac-Man

http://pacman.elstonj.com/

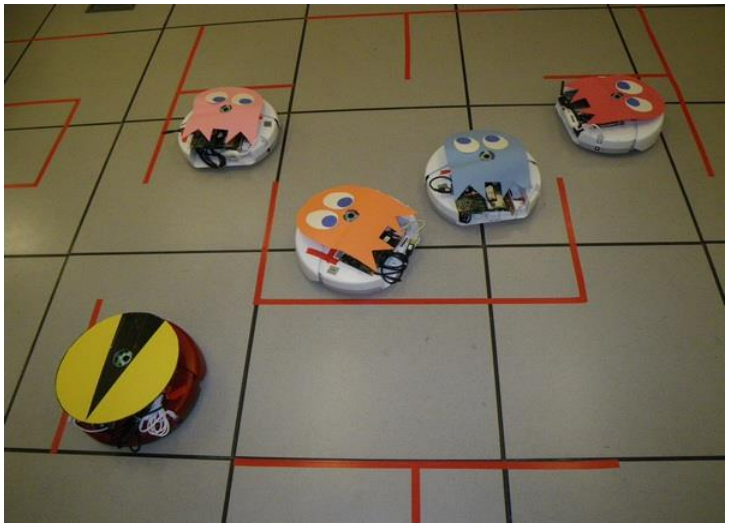


https://www.youtube.com/watch?v=7JHtX2JwZAY

Lets talk about the models for this scenario

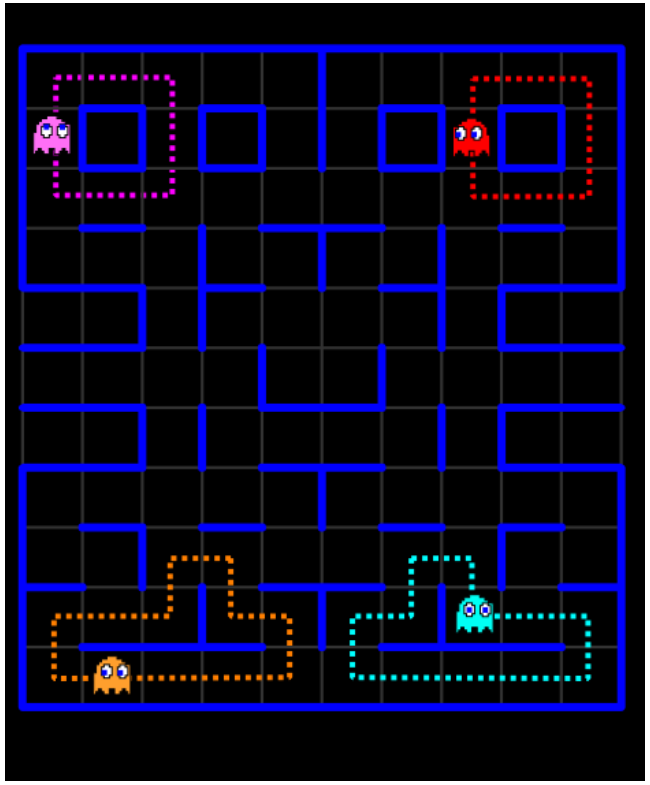
Models differ in their purpose:

1. Requirements (Functional and non-functional)
2. Physical space (walls, cherries, initial place, …)
3. Behaviour of the roombas based on their roles (ghosts and pac-man)

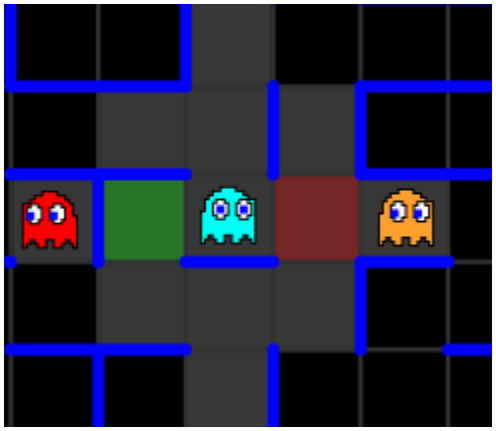


Ghosts

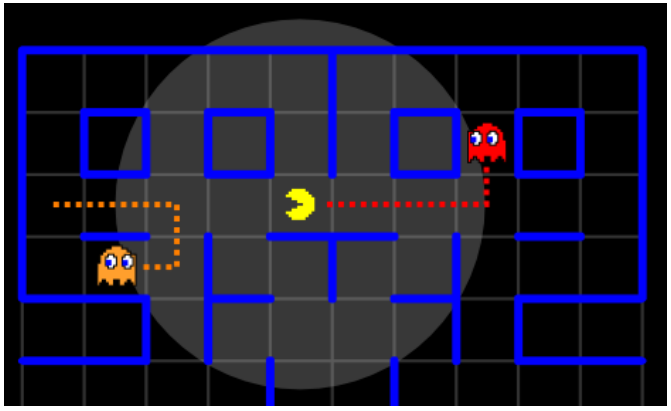
These are autonomous systems that:



Search



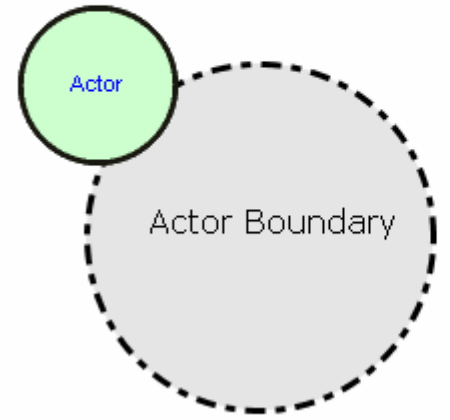
Avoid collisions

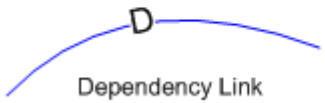


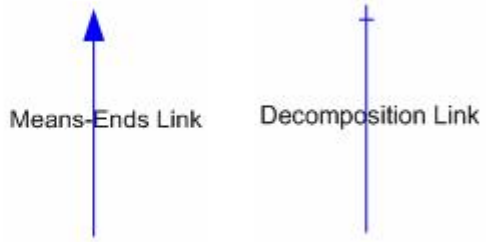
Track

Requirements – GORE

- GORE: Goal-Oriented Requirements Engineering
- **Elements**: goals, soft-goals, tasks, resources, actors, actor boundaries, links …



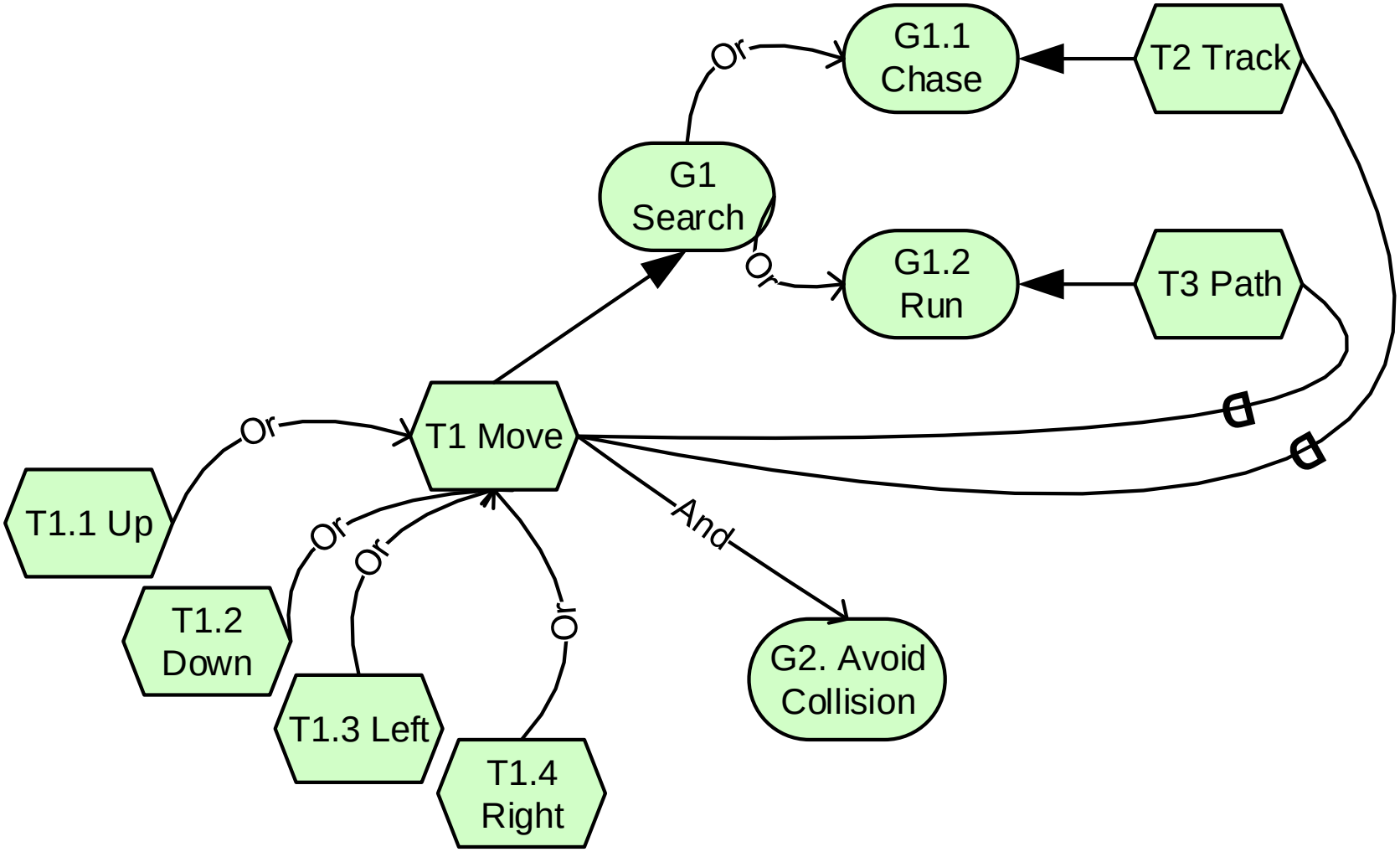




i* Framework is the modelling language suitable for goal models

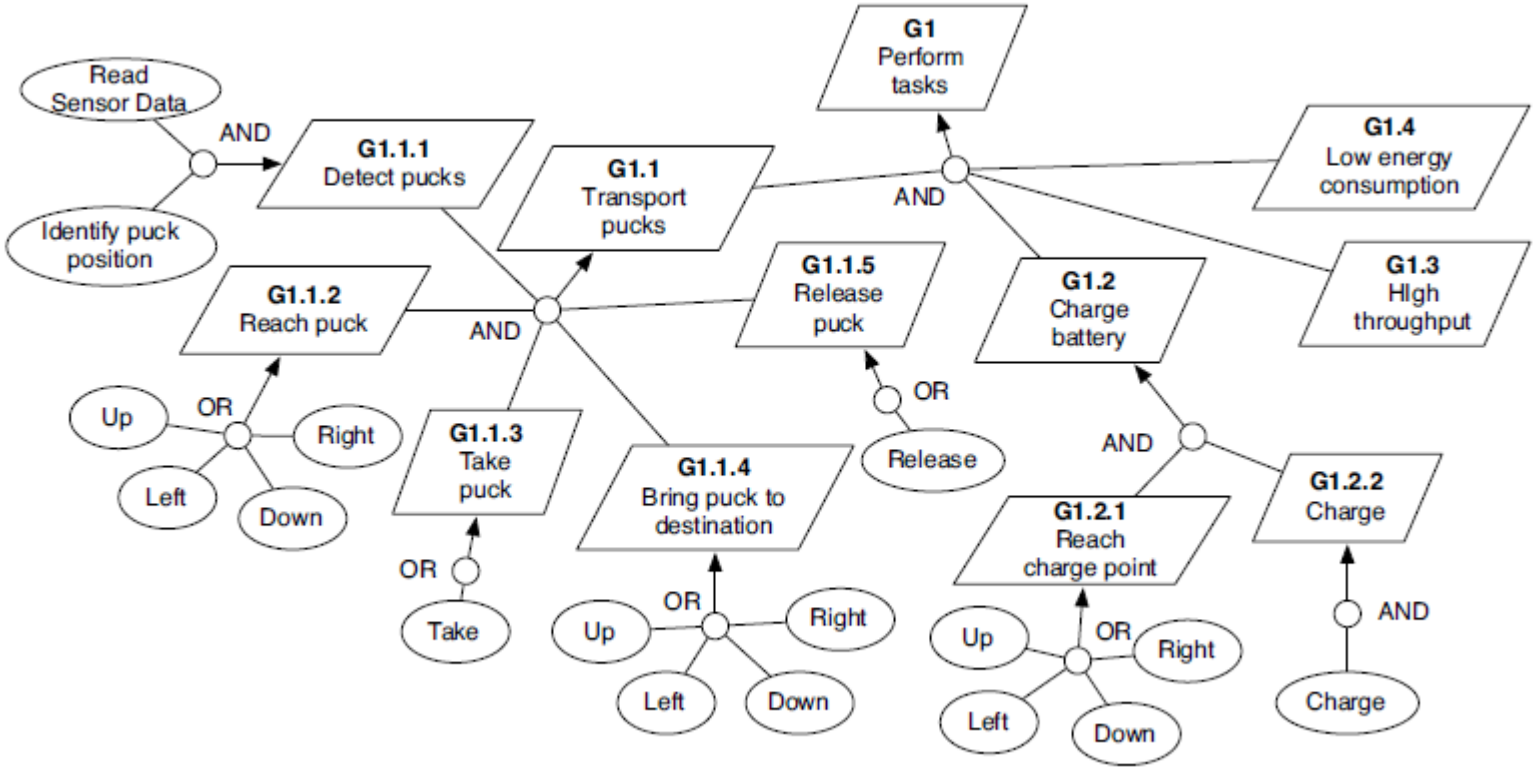
E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", 1997 IEEE

Simple version of a ghost GORE



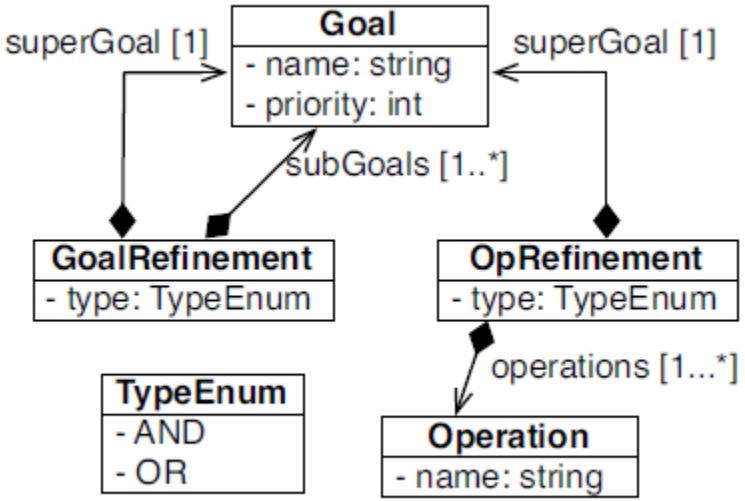
Requirements – KAOS

- KAOS: Keep All Objectives Satisfied
- Extension of GORE with AND / OR operations



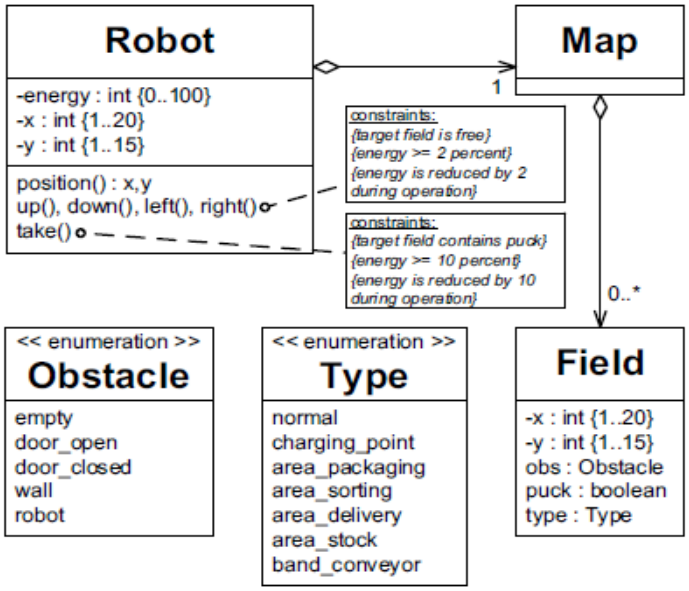
Metamodel of the Elements in KAOS

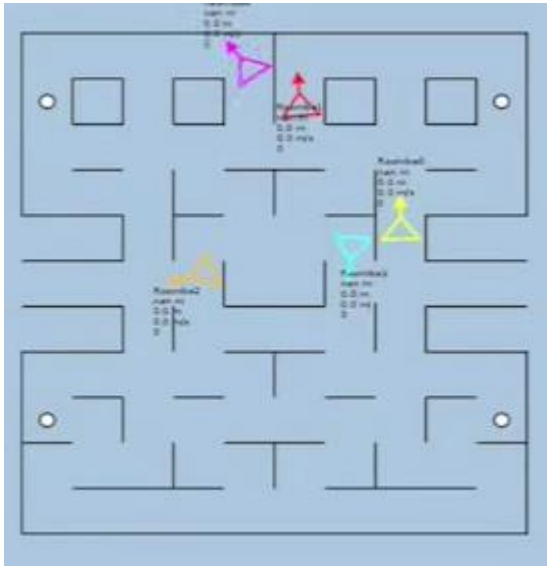
- A MART is available as a software artefact
- Transition between the modelling notation and a piece of software
- KAOS has a metamodel, but other representations can be used for goal models : transformations (i*)



Physical space – Structural context (Grid map)

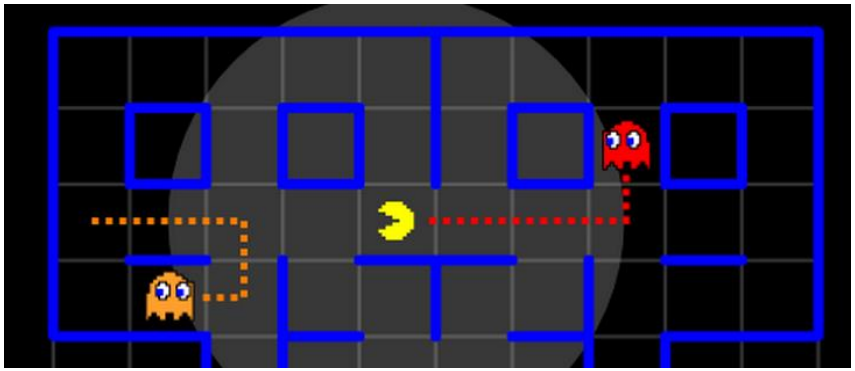
- To represent the environment in which the robot is moving.
- Instances of the model can be enhanced with real world data using other sensors.



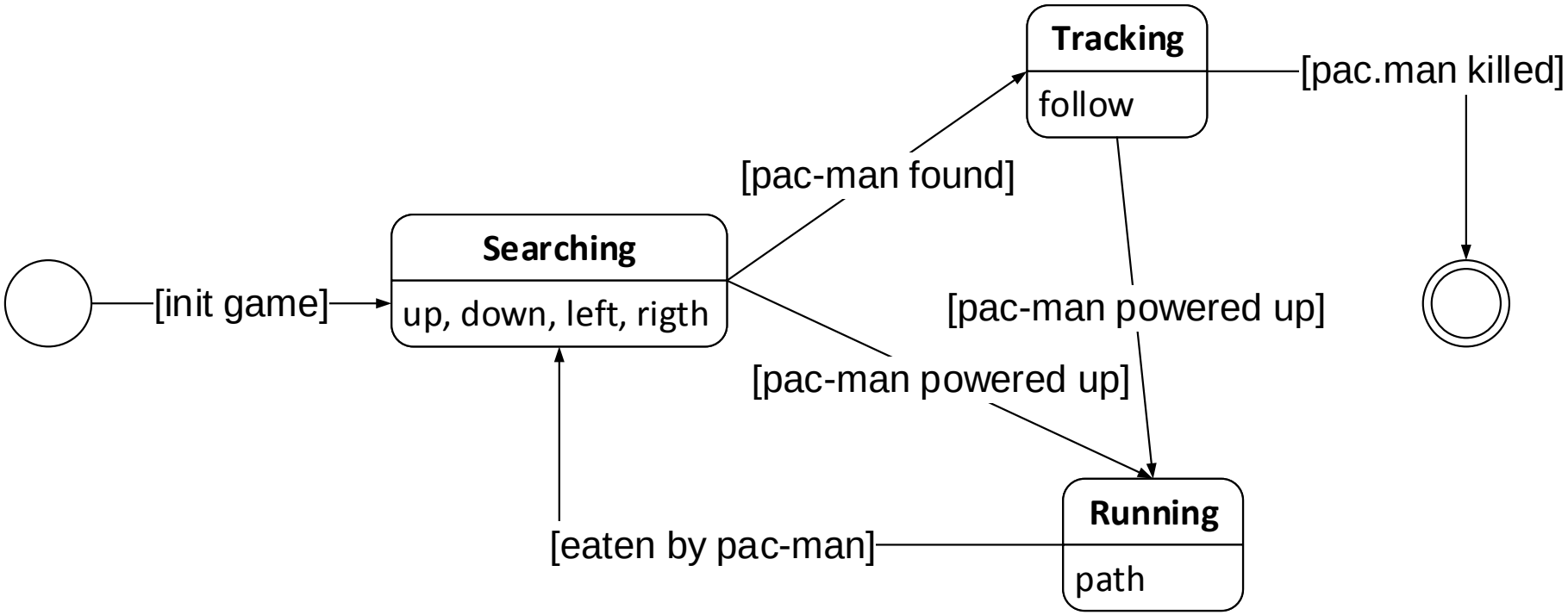


Behavioural – FSM

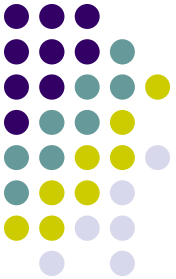
- Finite State Machine is a model to describe the behaviour of the system. In this case, the behaviour of each robot
- Lets consider the states of a ghost
  - Searching for pac-man (initial)
  - Tracking pac-man
  - Running from pac-man
  - Stop (end of the game)
- The transitions
  - Pac-man found
  - Eaten by pac-man
  - Kill pac-man



FSM of a ghost

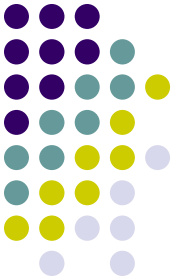


Levels of abstraction



- A model can represent different levels of abstraction
- *Abstraction* implies to eliminate characteristics that are irrelevant for the purpose of the model
  - E.g., how much of the physical space of the real world do we need to model for the rommba pac-man example?
- But MART should be causally connected
  - Hierarchical State Machine (HSM) provide a comprehensive view of models and their levels of abstractions

Properties of the MART



- MART properties are relevant at design-time as well as at run-time
- Validity: the model reflects correctly its original
  - Keep in mind the **abstraction level** when defining “correctly”
- Accuracy: measures the predictions of the model about its original.
- Precision: measures how small the variation is in the prediction made by the model

Uncertainty

Uncertainty within a model is the difference between the amount of information about the original and the information that the model could, in theory, represent about the original at a certain instant in the system lifetime [Giese 2014]

- MART deal with the uncertainty of the context which makes predictions a real challenge
- Dynamic models increase the level of uncertainty over time because of the “possible” continuous updates in order to reflect changes in the original.

Next class

- The feedback loop in systems with runtime models