Doct. of Computer Science and Technology

Ph. D. thesis defense

Multi-Layered Architectures for Autonomous Systems

José Carlos González Dorado

Thesis advisors { *Fernando Fernández Rebollo*, *Ángel García Olaya*

Planning and Learning Group

March 27th, 2020

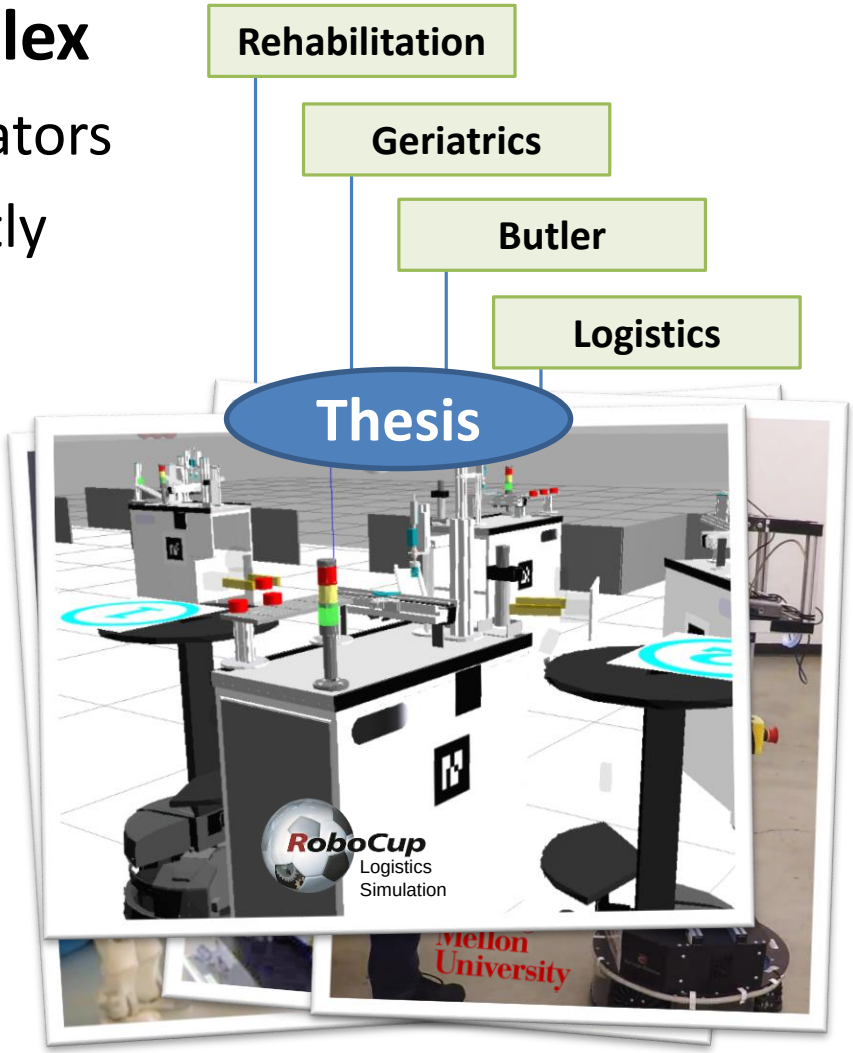
Computer Science Department

Outline

1. **Introduction**
2. NAOTherapist architecture
3. Proposed guidelines
4. Mlaras architecture
5. Mlaras opportunities & failures
6. Mlaras in the logistics league
7. Conclusions

Motivation

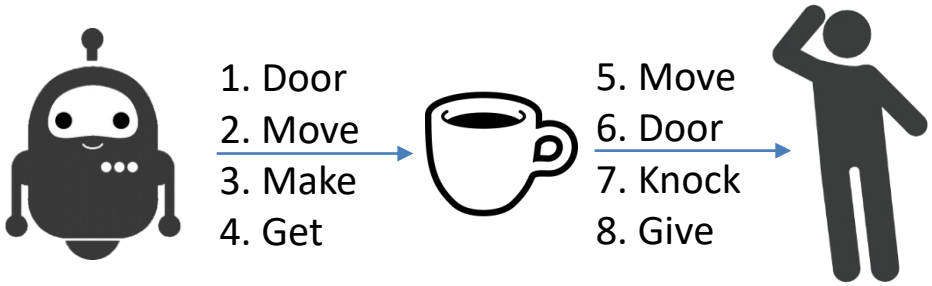
- **Autonomous robotics is complex**
  - Many types of sensors and actuators
  - Deliberation to behave coherently
- **Deliberation is complex**
  - Now feasible for real systems
- **Architecture for coordination**
  [Kortenkamp et al. 2008]
- **Use cases must be defined**
  - How to really implement them?



Managing use cases

- **Use cases can be seen as action plans** [Ghallab et al. 2014]

- **Stochastic environments**
  - May invalidate the plans



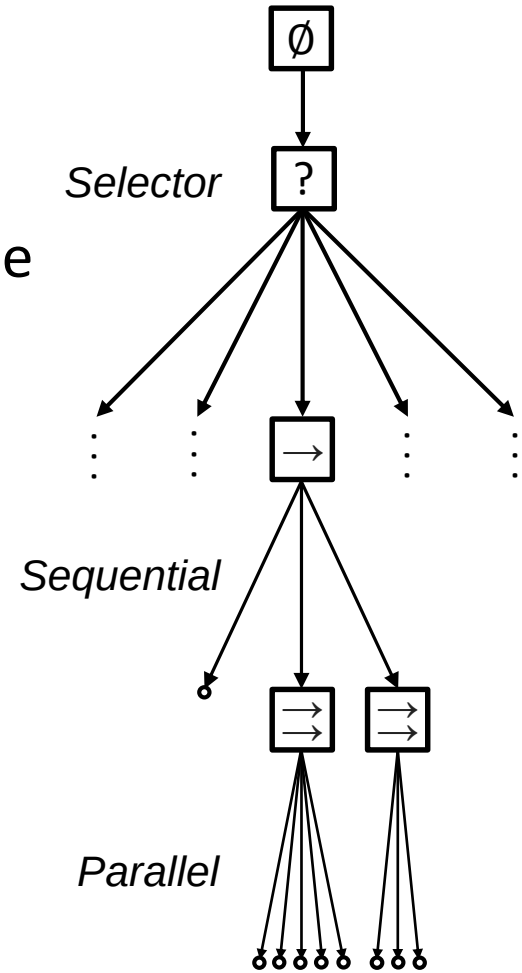
- **Reasoning strategy**
  - Deliberative: all deliberation (sense-plan-act), but slow
  - Reactive: all reactive, fast, but short-term
  - **Hybrid**: joins both, hierarchical, layered

Control strategies

- **Procedural control**
  - Behavioral trees [Ögren et al. 2018]
  - Tree sets can model use cases
  - Action decompositions save deliberation time

- **Deliberation**
  - Decomposed deliberation
  - **Higher** layers: deliberative
  - **Lower** layers: reactive
  - Several simpler problems are easier

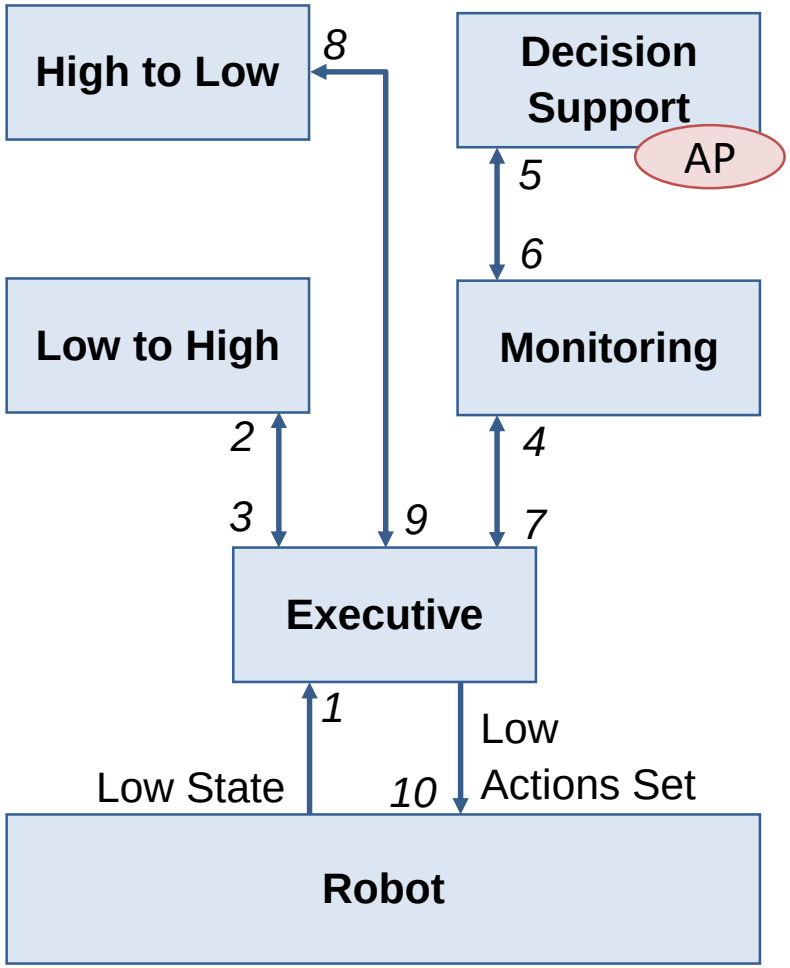
Action decomposition



PELEA

- **Planning and execution system**
  - Focused on classical planning
  - Automated Planner as a black box
  - Modular and extensible
- **Ad-hoc abstraction translation**
  - Actions: High to low
  - States: Low to high
- **Monitors the execution**
  - Replans when state is invalid

[Alcázar et al. 2010]



Layered architectures

	Deliberation	Stochastic	Temporal	Declarative	Multilayer	Middleware
LAAS[1]	Cust. AP	Ad hoc	Cust. AP	Ad hoc	✓	-
T-REX[2]	**AP**	**Replan**	**Timelines**	Partial	✓	-
PELEA[3]	**AP**	**Replan**	**Temp. AP**	Partial	X	-
ROSPlan[4]	**AP**	**Replan**	**Temp. AP**	Partial	X	**ROS**
CORTEX[5]	Ad hoc	Ad hoc	Ad hoc	Ad hoc	X	**RoboComp**

- **Ascending order by year**
- **No one fulfills everything**

1. [Alami et al. 1998]
2. [McGann et al. 2008]
3. [Alcázar et al. 2010]
4. [Cashmore et al. 2015]
5. [Bustos et al. 2019]

Problems of the architectures

- **Lack of guidelines and standards**
- **Difficult to reuse previous works**
- **Use cases are hardcoded by developers**
  - However, they must be defined by end users
- **Hardcoded abstraction conversions**
- **Complex and slow deliberation models**
- **Lack of multilayer deliberation support**

All them slow down the development and advancement of autonomous systems

Objectives of the thesis

- Ease the use of standard deliberative techniques in use-case oriented cognitive architectures for autonomous systems

  I. Design architectures whose structures reflect the use-case
     - Use formalisms to involve the user in the behavior development
     - Ease the use-case modular decomposition into subproblems
     - Design layered architectures to organize knowledge

  II. Define relations among architecture components and layers

  III. Use state-of-the-art deliberative techniques

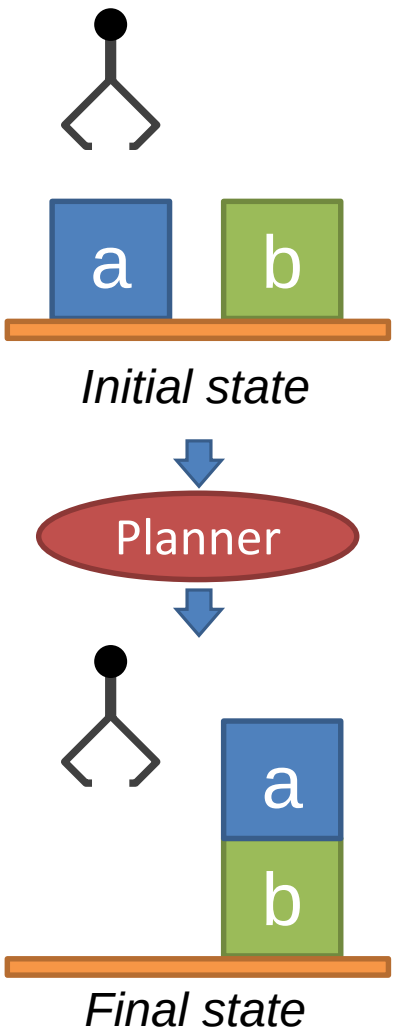
  IV. Carry out objectives I, II and III declaratively

  V. Design guidelines to apply deliberation in these systems

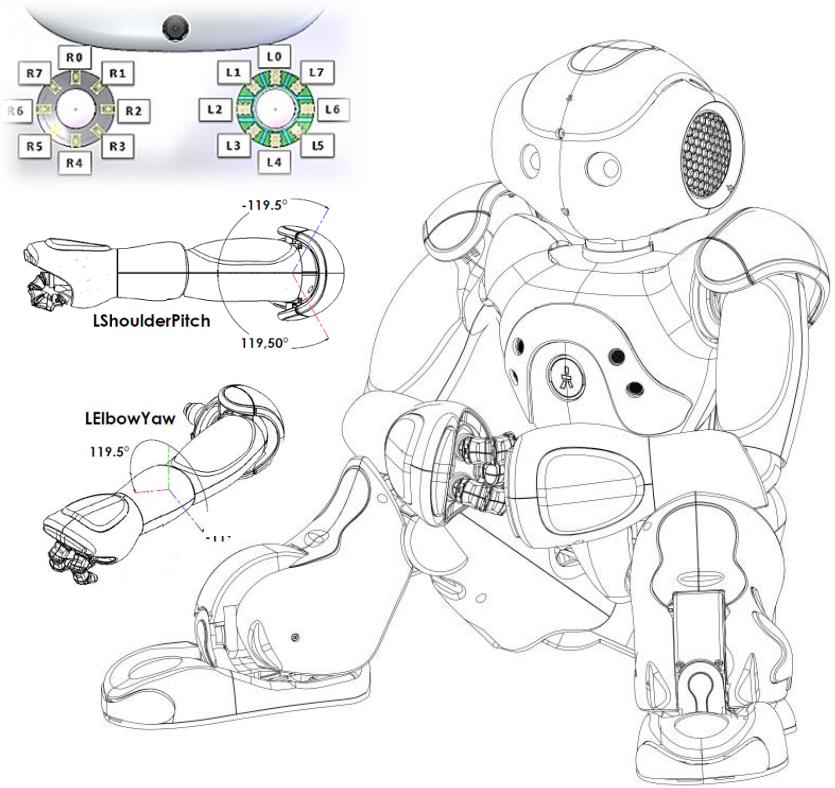
  VI. Evaluate all previous objectives in real systems

Used deliberation techniques

- **Automated Planning (AP)** [Ghallab et al. 2004]
  - Generic planner finds action plans for goals
  - Declarative formal language (PDDL)
  - Need to interleave planning and execution
  - Multiple paradigms
    - **Classical**, probabilistic, temporal
- **Mixed Integer Programming (MIP)**
  - Fast way to reason with numbers and time
  - Declarative rules and problem [Chen et al. 2010]
- **More complex paradigms are slower**



NAOTherapist architecture

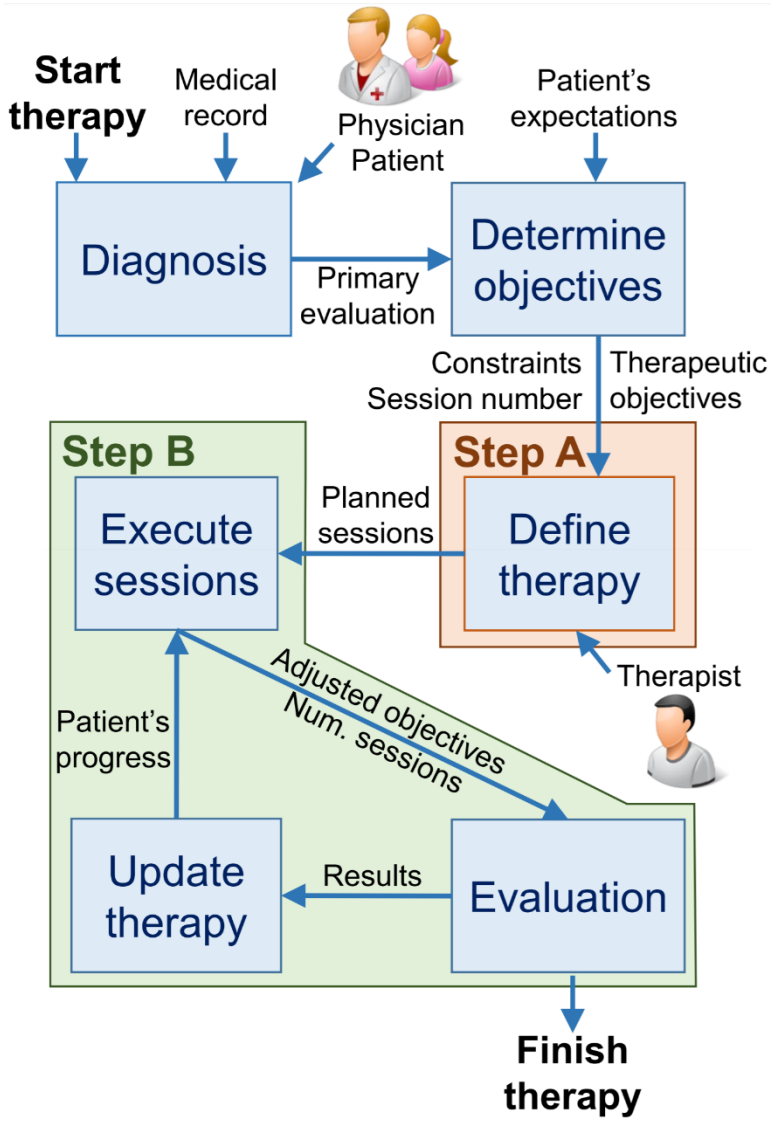


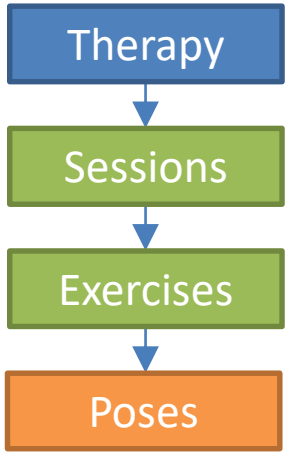
1. Introduction
2. **NAOTherapist architecture**
3. Proposed guidelines
4. Mlaras architecture
5. Mlaras opportunities & failures
6. Mlaras in the logistics league
7. Conclusions

Mirror-game use case

youtu.be/PbfqoILctH4

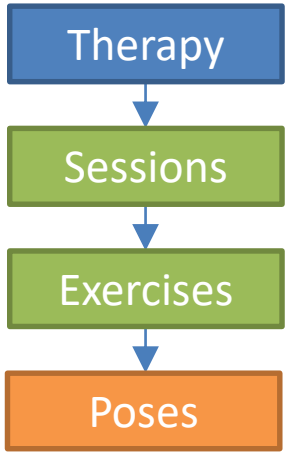
Therapeutic protocol





Use-case planning levels

- **High level: plan all therapy sessions**
  - Classical planning
  - Many complex constraints
  - Offline, manual replannings
- **Medium level: plan the actual execution**
  - Classical planning and PELEA to replan online
  - Problems converted from the high level
  - Centralized ad-hoc abstraction translations
- **Low level: plan each movement**
  - Transparent and independent from the robot and sensor



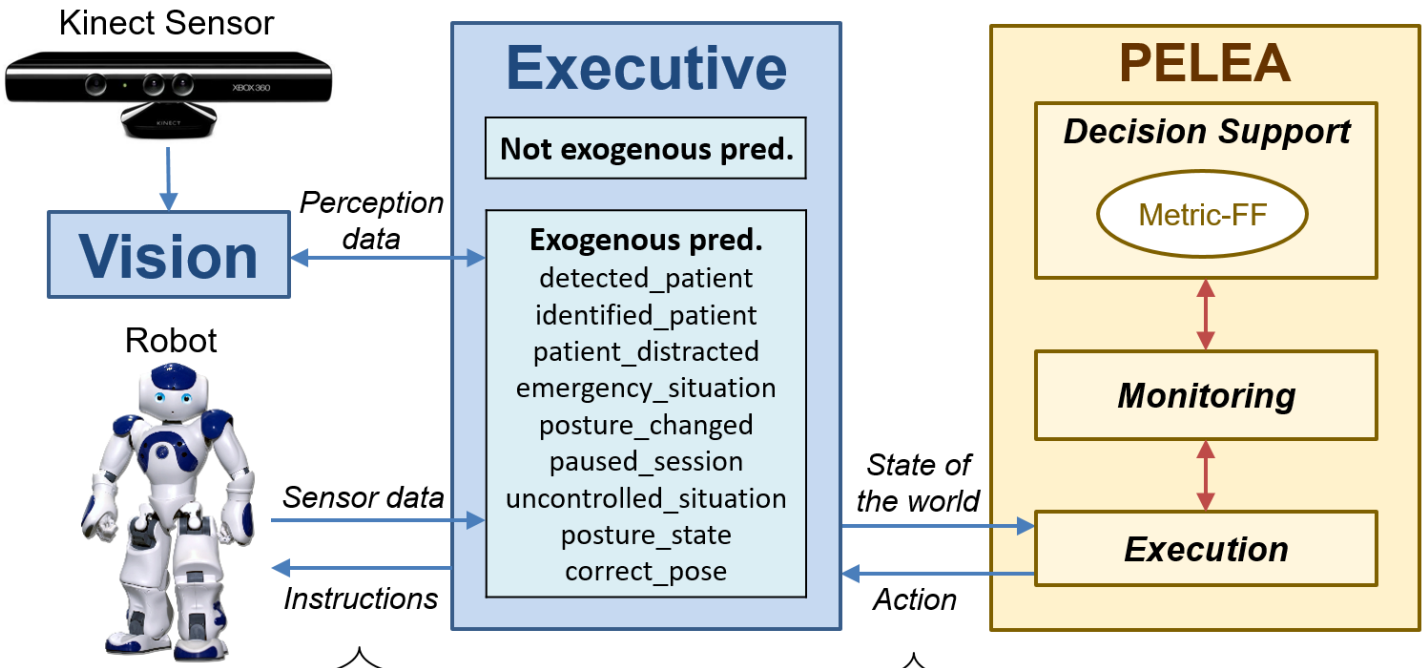
High-level therapy designer

Exercises →

Sessions ↓

	1	2	3	4	5	6	7	8
1	e0	e9	e11	e12	e10	e7	e15	
2	e4	e2	e5	e6	***L19**	***L20**	***L21**	e13
3	e1	e3	e8	***L22**	***L23**	***L24**	***L25**	e16
4	***L26**	***L27**	***L28**	***L29**	***L30**	***L31**	e17	
5	e0	e11	e12	e10	e9	***L32**	e15	
6	e4	e2	e6	L19	e7	e5	L20	e13
7	e1	e3	L24	e8	L23	L22	e16	
8	L25	L26	L30	L27	L28	L29	e17	
9	e0	e12	e10	L31	e11	e9	e15	
10	e4	e2	L19	L20	e6	e7	e13	
11	e1	e3	L22	L23	L24	e8	e16	
12	***L33**	L25	L26	L29	L30	L27	L28	e17
13	e0	e10	L21	e12	e9	e11	e15	
14	e4	e2	e7	e6	L20	L19	e13	
15	e1	e3	L24	e8	L22	L23	e16	
16	L25	L27	L31	L26	e5	L29	L30	e17
17	e0	e11	e12	L28	e10	e9	e15	
18	L32	e4	e2	L19	e6	L20	e7	e13
19	e1	e3	L22	L23	L24	e8	e16	
20	L25	L26	e5	L29	L30	e17		

Medium and low-level architecture



Low-level instructions

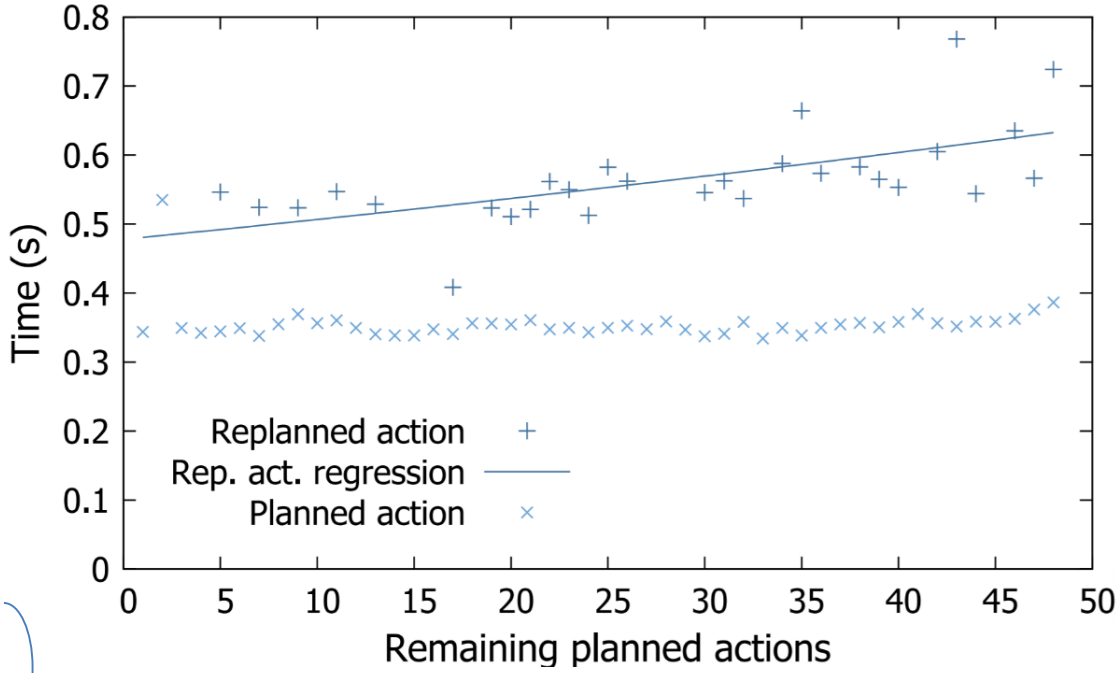
setAnglesFromVision
executeAnimation
say
playAudioFile
setLeds
getPostureFamily
getLastButtonPressed
isConnected
isSimulated
allowAutonomousMovements

Medium-level actions

detect-patient
identify-patient
greet-patient
start-training
introduce-exercise
stand-up
sit-down
start-exercise
execute-pose
correct-pose
finish-pose
finish-exercise
perform-relaxation
finish-training
say-good-bye
finish-session
claim-stand-up
claim-sit-down
claim-attention
pause-session
resume-session
cancel-session

Technical evaluations

- **Planning times**
  - 62 real short sessions
  - 23 poses each



- **Generalization**
  - Simon
  - Reverse Simon
  - Simon says
  - Dancing
  - Teaching movements

[García et al. 2017]

Field tests

- **3 field tests with patients**
  - 3 children, 1 session
  - 8 children, 4 months, weekly
  - 10 children, 15 days, daily
- **Overall**
  - 244 children (21 patients)
  - 429 sessions (206 of patients)
  - Good interactive outcomes
- **Clinical and HRI analysis**

  [Pulido 2020]

Lessons learned

- Multilayered deliberation works well
- Needs guidelines
- Need to fix the monolithic Executive
  - Hard to maintain, small changes hard to apply
- Needs a full declarative configuration
- Needs action interruption
- Needs an online high-level planning
- **The NAOTherapist architecture is not enough**

✓ Rehabilitation

✗ Geriatrics

✗ Butler

✗ Logistics

Proposed guidelines

1. Introduction
2. NAOTherapist architecture
3. **Proposed guidelines**
4. Mlaras architecture
5. Mlaras opportunities & failures
6. Mlaras in the logistics league
7. Conclusions

Objective of the guidelines

- **Face the system design from the use case**
  1. Deliberation strategy: problem decomposition
  2. Planning model: domain and problem modeling
  3. Executive model: interleaving planning and execution

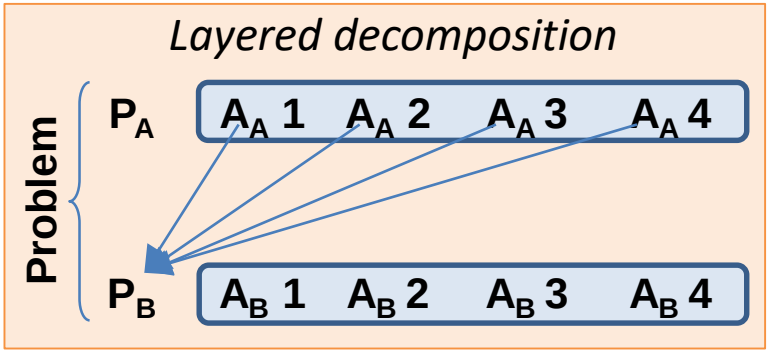
1. Design deliberation strategy	Design layer reasoning	Deliberative
		Subsymbolic
		Reactors
	Define planning paradigm	Classical
		Hierarchical
		Probabilistic
		Temporal
	Design planning decomposition	Planning horizon
		Model partition
2. Design planning model	Action model	Design use case and nominal behavior
		Islands
		Corrective actions
		Interactive steps
	State model	Internal versus external
		Predictable versus unpredictable
		Symbolic versus numerical
3. Design executive model	Continuous monitoring	Replanning scheme
		Interruption scheme
		Failure and opportunity management
		Planning time constraints
	Online knowledge management	High to low decomposition
		Low to high decomposition
	Deployment issues	Retrieving numerical fluents
		Component connection
		Easing the use-case definition

	Probabilistic
	Temporal
sition	Planning horizon
	Model partition
	Design use case and nominal
	Islands

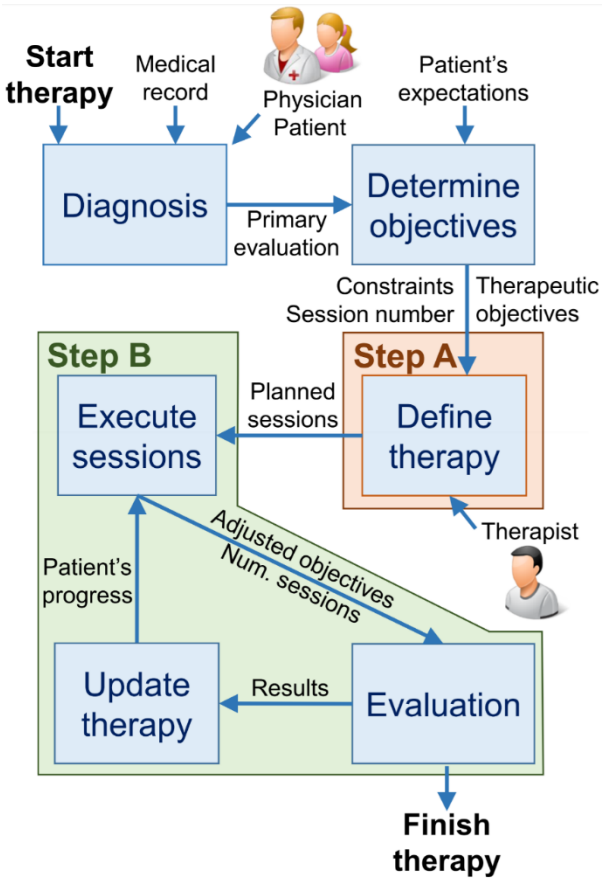
Guidelines for AP and robotics

1. Design deliberation strategy

- Define planning paradigm
  - Classical, temporal, probabilistic…
- Design layer reasoning
  - Deliberative, reactor



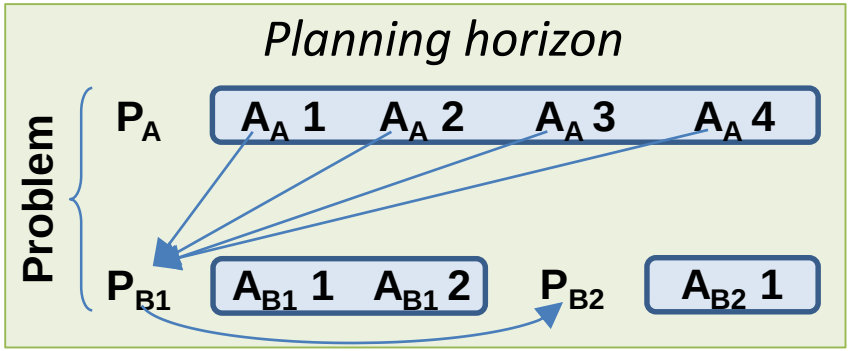
NAOTherapist
- P_A Define therapy
- A_A Session definition
- P_B Execute session
- A_B Interactive actions



Guidelines for AP and robotics

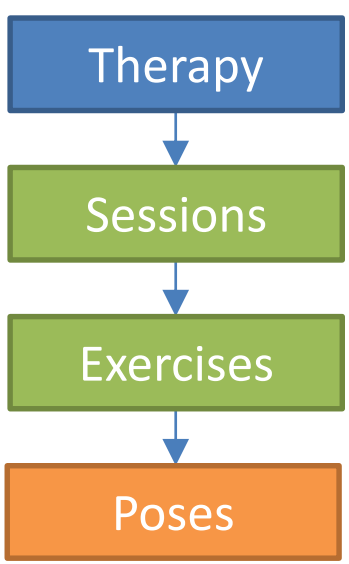
1. Design deliberation strategy

- Design planning decomposition



NAOTherapist
- P_{B1} Session 1
- P_{B2} Session 2
- A_B Exercise

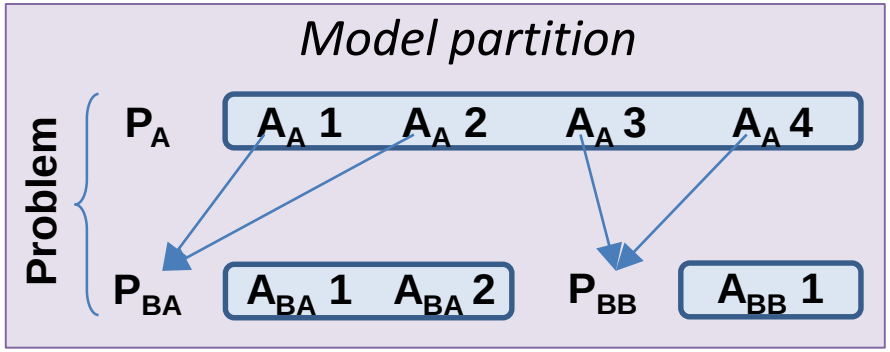
High-level
Divide & conquer



Guidelines for AP and robotics

1. Design deliberation strategy

- Design planning decomposition



NAOTherapist
- P_A Define therapy
- A_A Session definition
- P_{BA} Execute use case A
- P_{BB} Execute use case B
- A_B Interactive actions

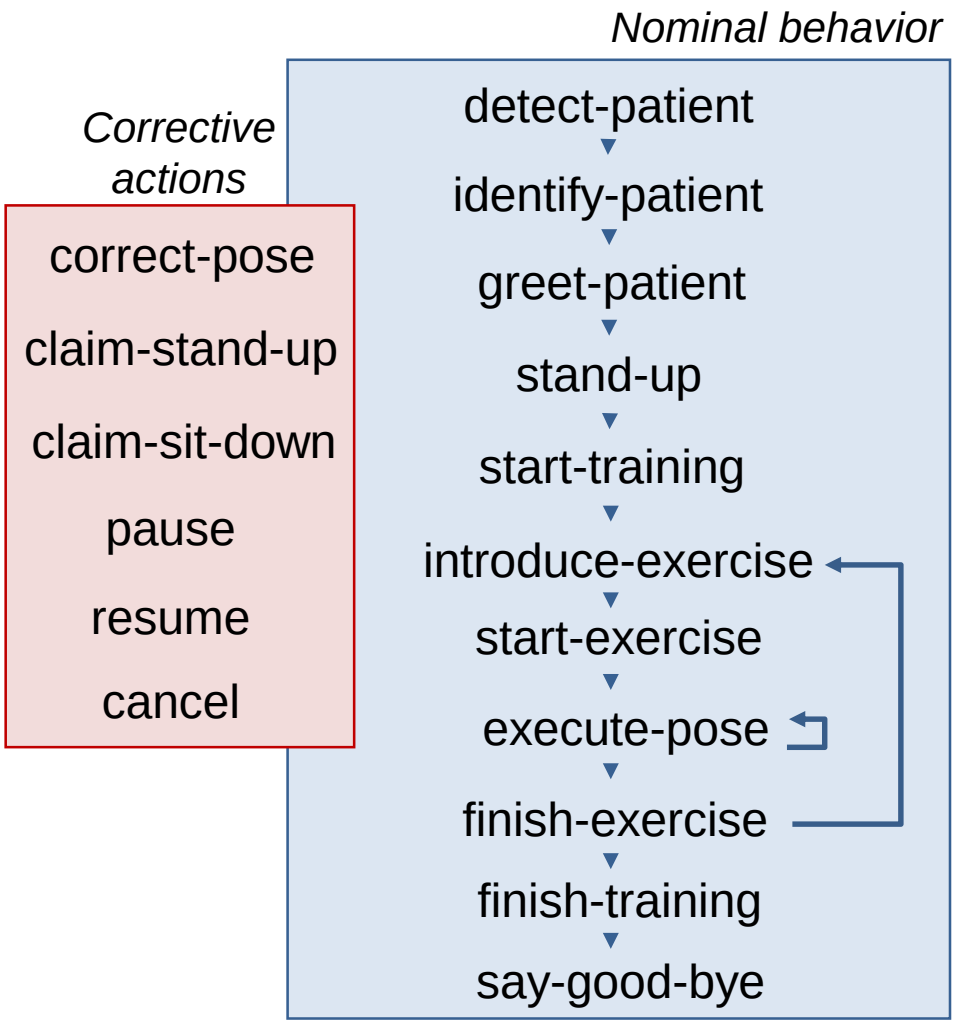
NAOtherapist's use cases

1. Mirror
2. Simon
3. Reverse Simon
4. Simon says
5. Dancing
6. Teaching movements

Guidelines for AP and robotics

2. Design planning model

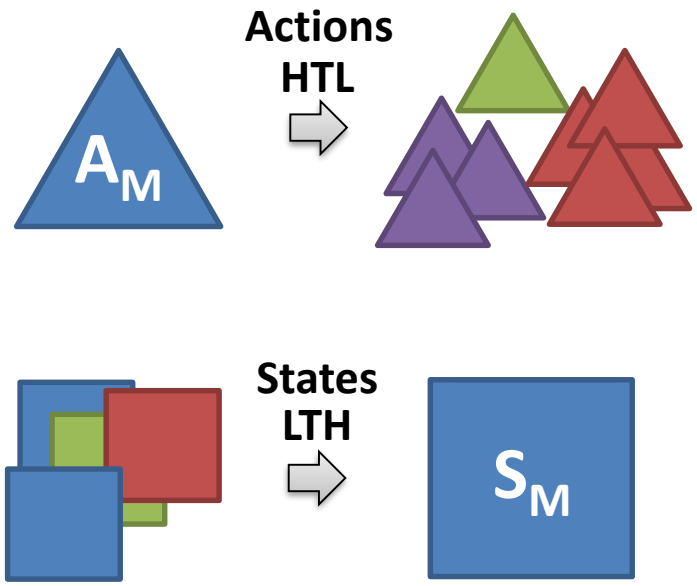
- Action model
  - Nominal behavior
  - Corrective actions
  - Islands
  - Interactive steps
- State model
  - Internal vs. external
  - Predictable vs. unpredictable
  - Symbolic vs. numerical



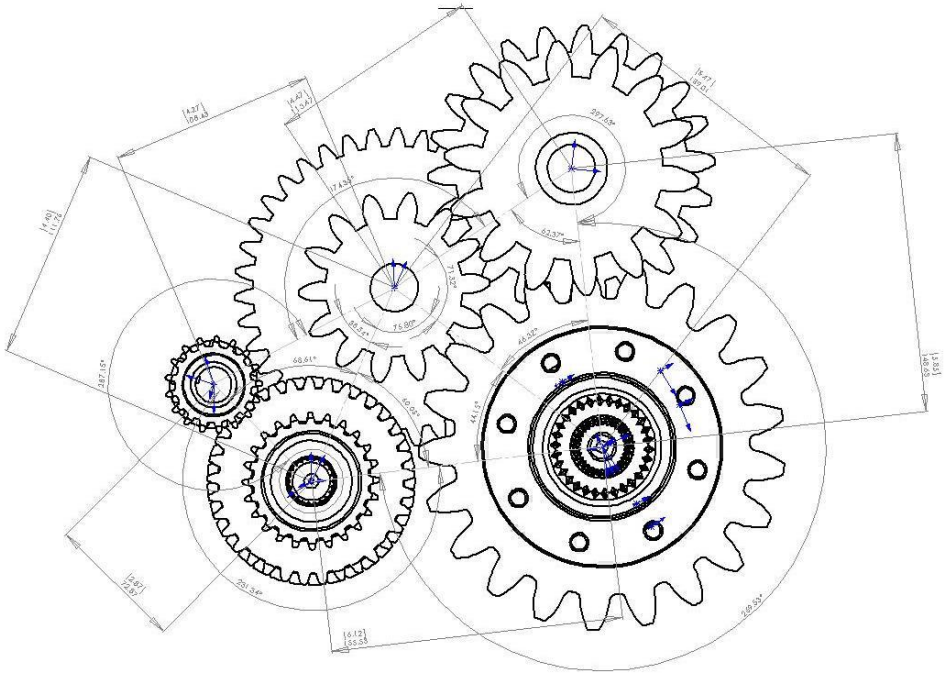
Guidelines for AP and robotics

3. Design executive model

- Continuous monitoring
  - Replanning scheme
  - Interruption scheme
  - Failures and opportunities
  - Planning time constraints
- Online knowledge management
  - High to low decomposition (actions)
  - Low to high generalizations (states)
- Deployment issues
  - Retrieving numerical fluents
  - Component connection
  - Easing the use-case definition



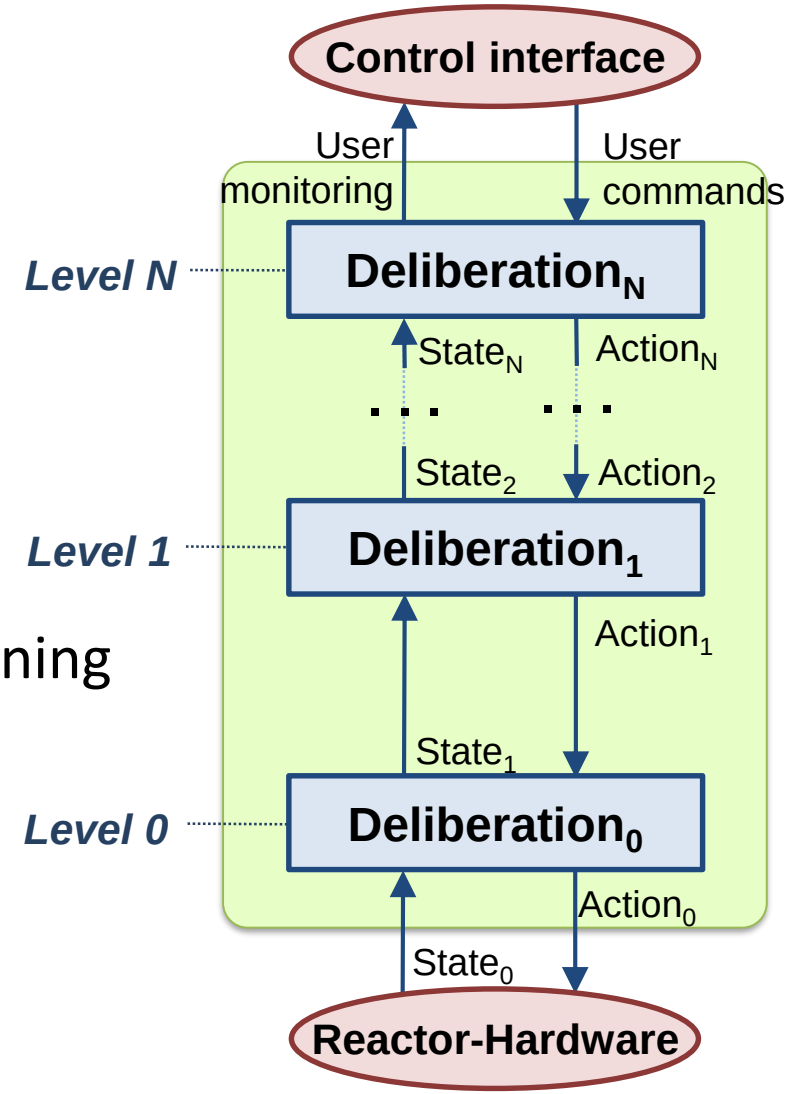
Mlaras architecture



1. Introduction
2. NAOTherapist architecture
3. Proposed guidelines
4. **Mlaras architecture**
5. Mlaras opportunities & failures
6. Mlaras in the logistics league
7. Conclusions

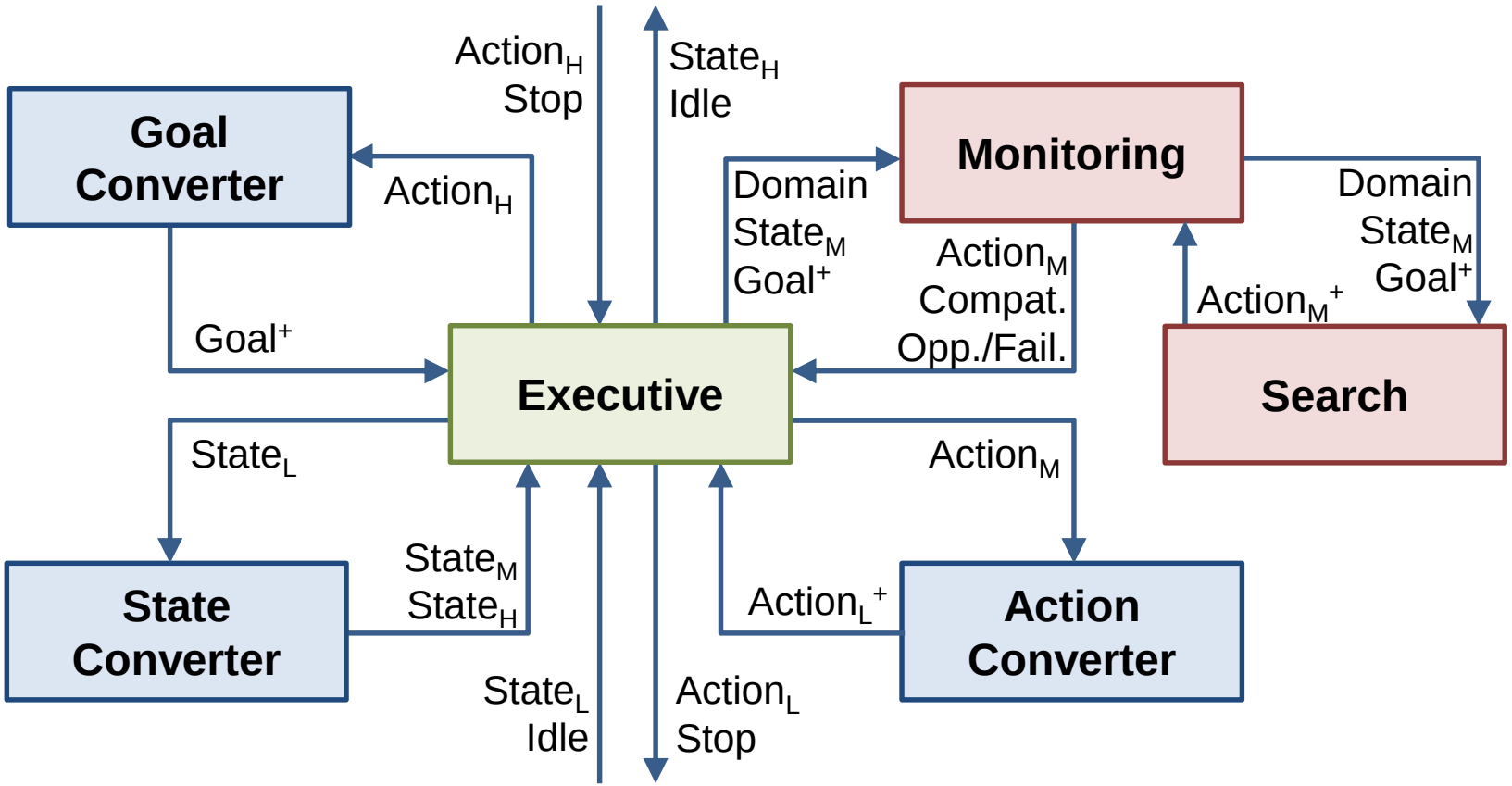
Mlaras architecture

- Multi-Layered ARchitecture for Autonomous Systems
- Focused on classical AP
  - PELEA generalization
  - Temporal: condition annotations
  - Stochastic: monitoring and replanning
  - Hierarchical: layered deliberation
- Explicit abstraction conversions
- Declarative use case definition



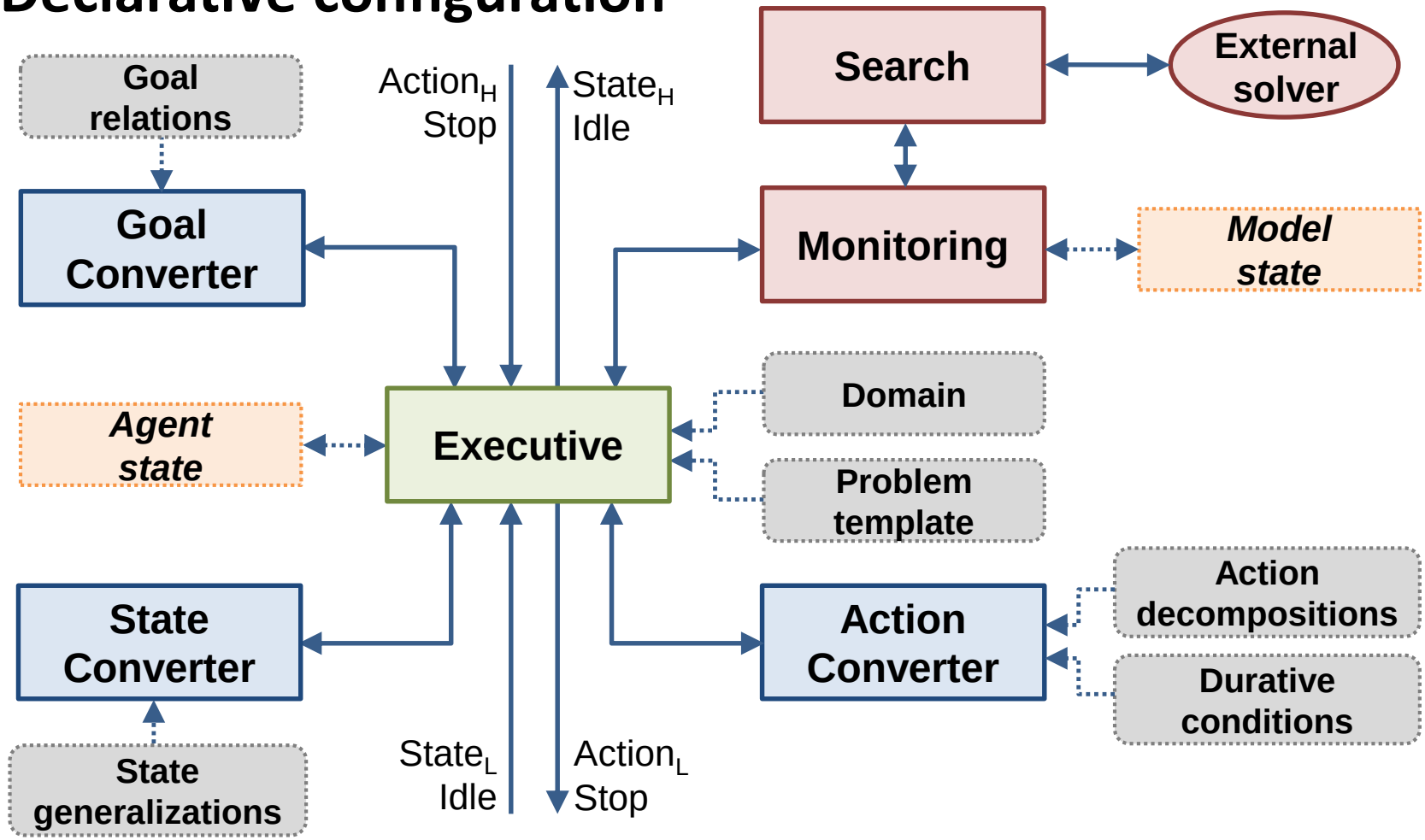
Mlaras modules

- **Deliberation layer overview**

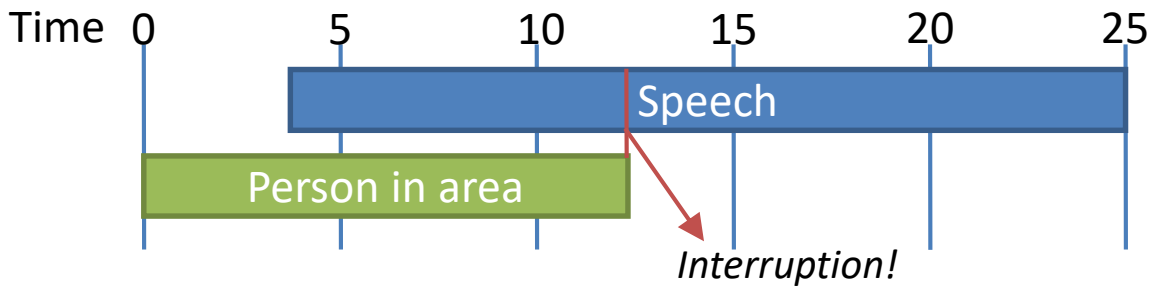


Mlaras modules

- **Declarative configuration**



Managing temporal aspects



- **Actions have duration**
  - Can be unknown
  - Interruption in the middle of their execution
- **Overall and finish-when annotations**
  - Similar to temporal planning without overlapping actions
  - Controlled by Mlaras, not by the planner

uc3m

Mlaras opportunities & failures

Joint work with Prof. Manuela Veloso

CoBots Carnegie Mellon University

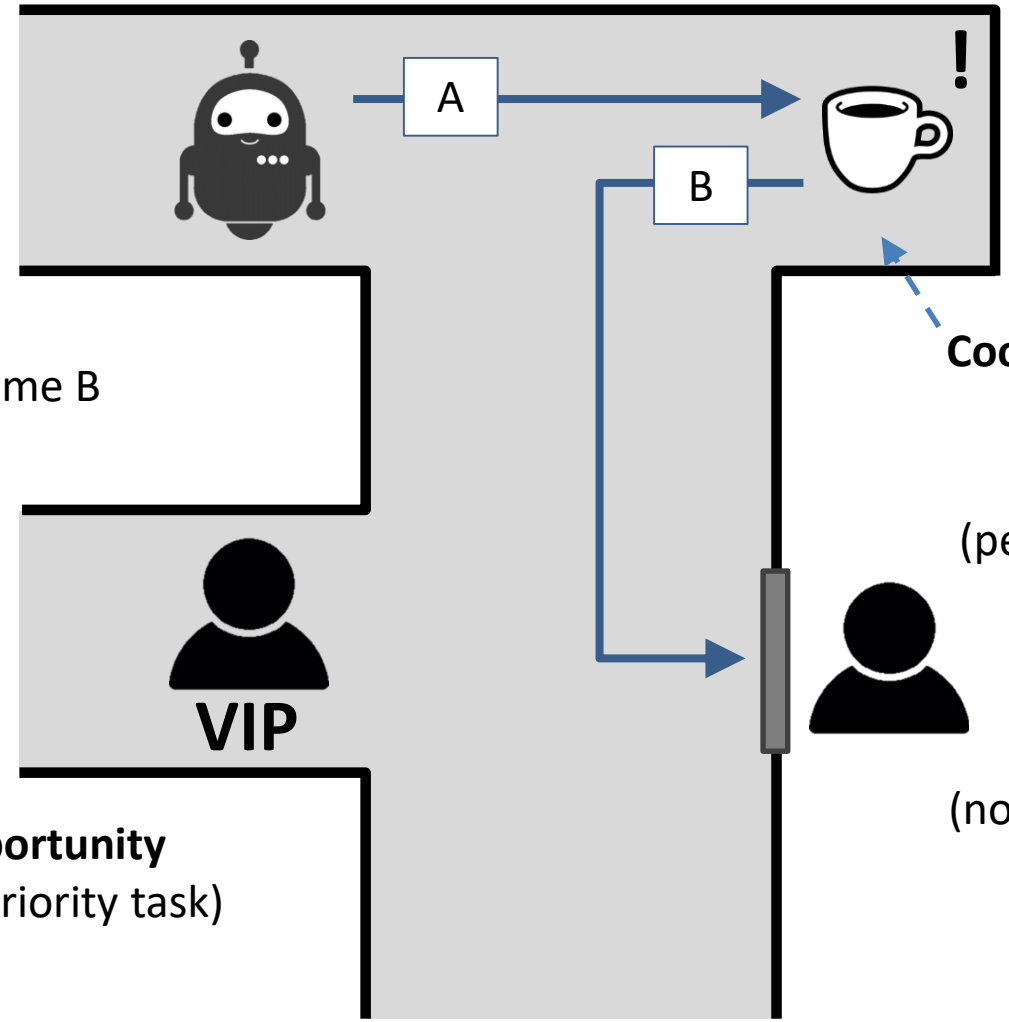
1. Introduction
2. NAOTherapist architecture
3. Proposed guidelines
4. Mlaras architecture
5. **Mlaras opportunities & failures**
6. Mlaras in the logistics league
7. Conclusions

Opportunities and failures

Subtasks: A, B

High-level options

- VIP first, then resume B
- Redo A and B
- VIP after B
- Cancel A and B
- Cancel VIP
- Try a quick VIP



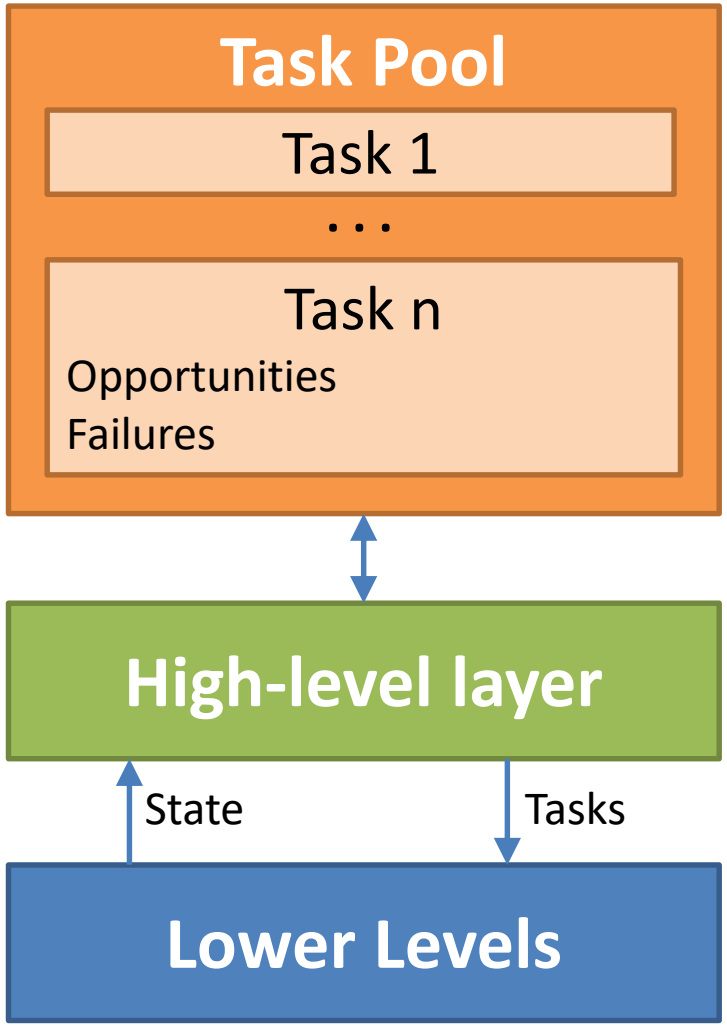
Cooling-down time

Opportunity
(person in the corridor)

Failure
(nobody is in the office)

Opportunity
(high-priority task)

Monitoring model

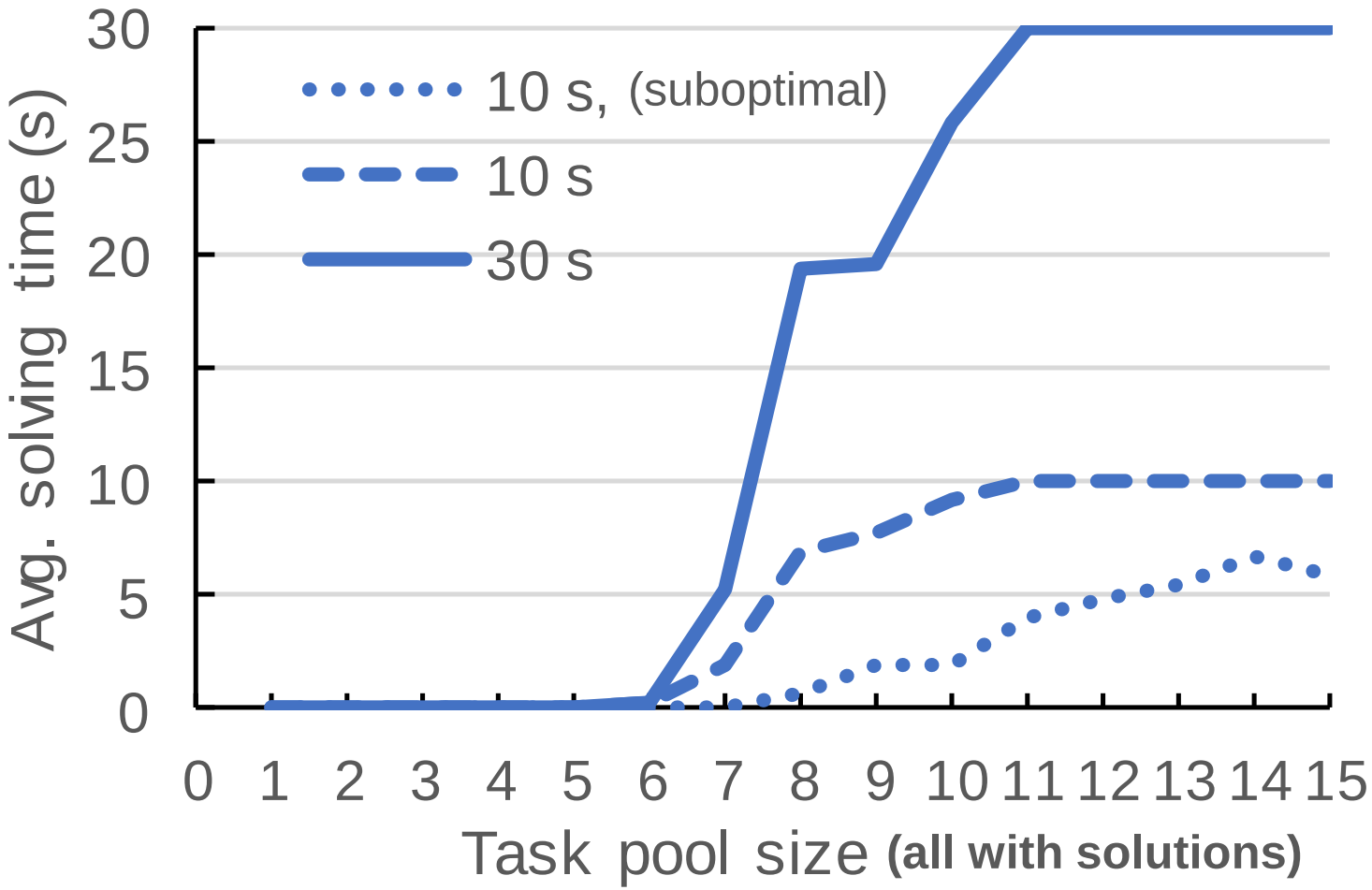


- **Updated states received**
  - While subtasks are being executed
- **Opportunities and failures**
  - Declarative
  - Priority, cooldown-time
- **A rescheduling can**
  - Add or remove tasks in the pool
  - Interrupt the current subtask
- **CoBot's scheduler based in MIP**

CoBot monitoring example

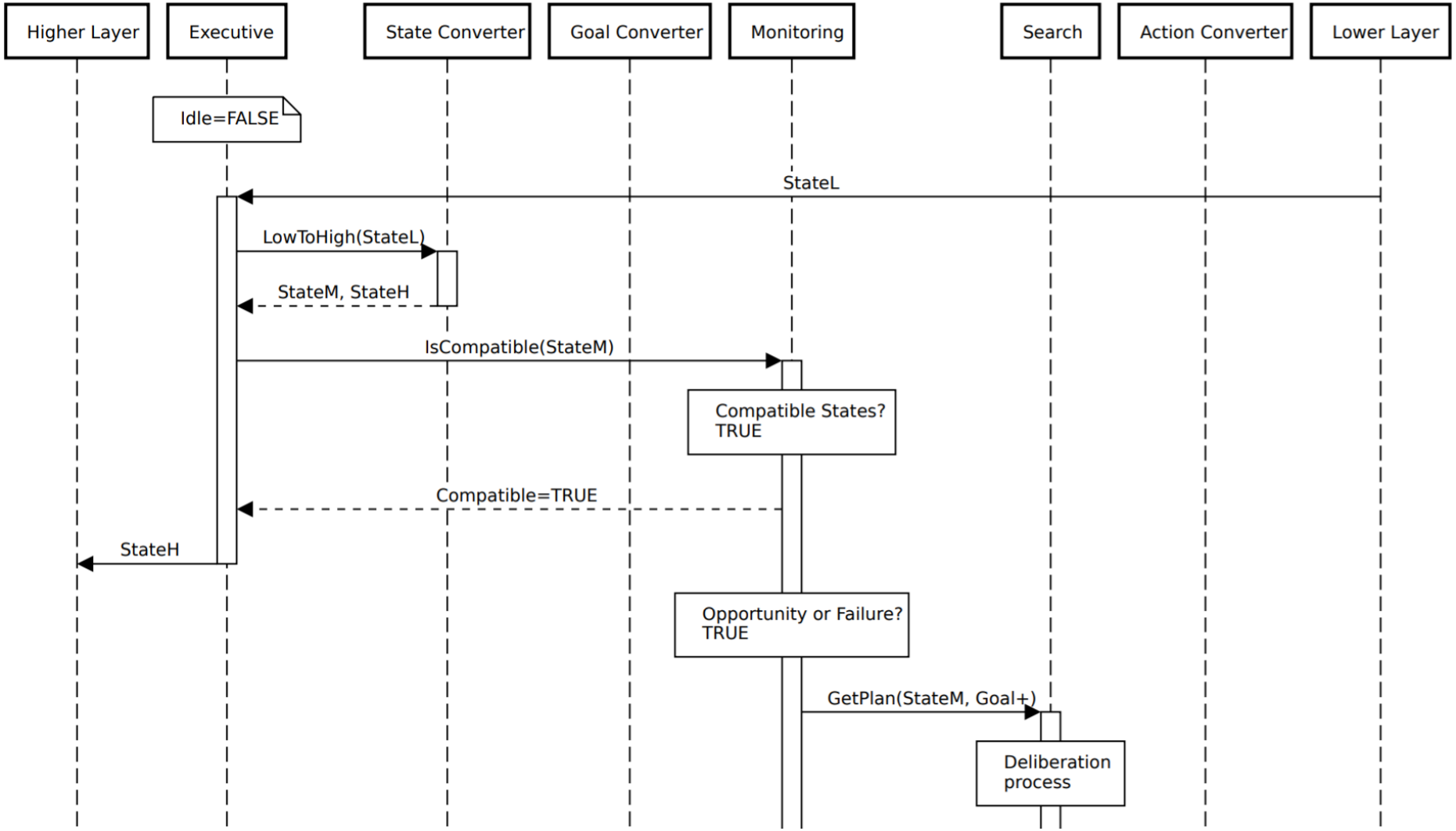
youtu.be/pUCcLHkTgVU

CoBot's solving time

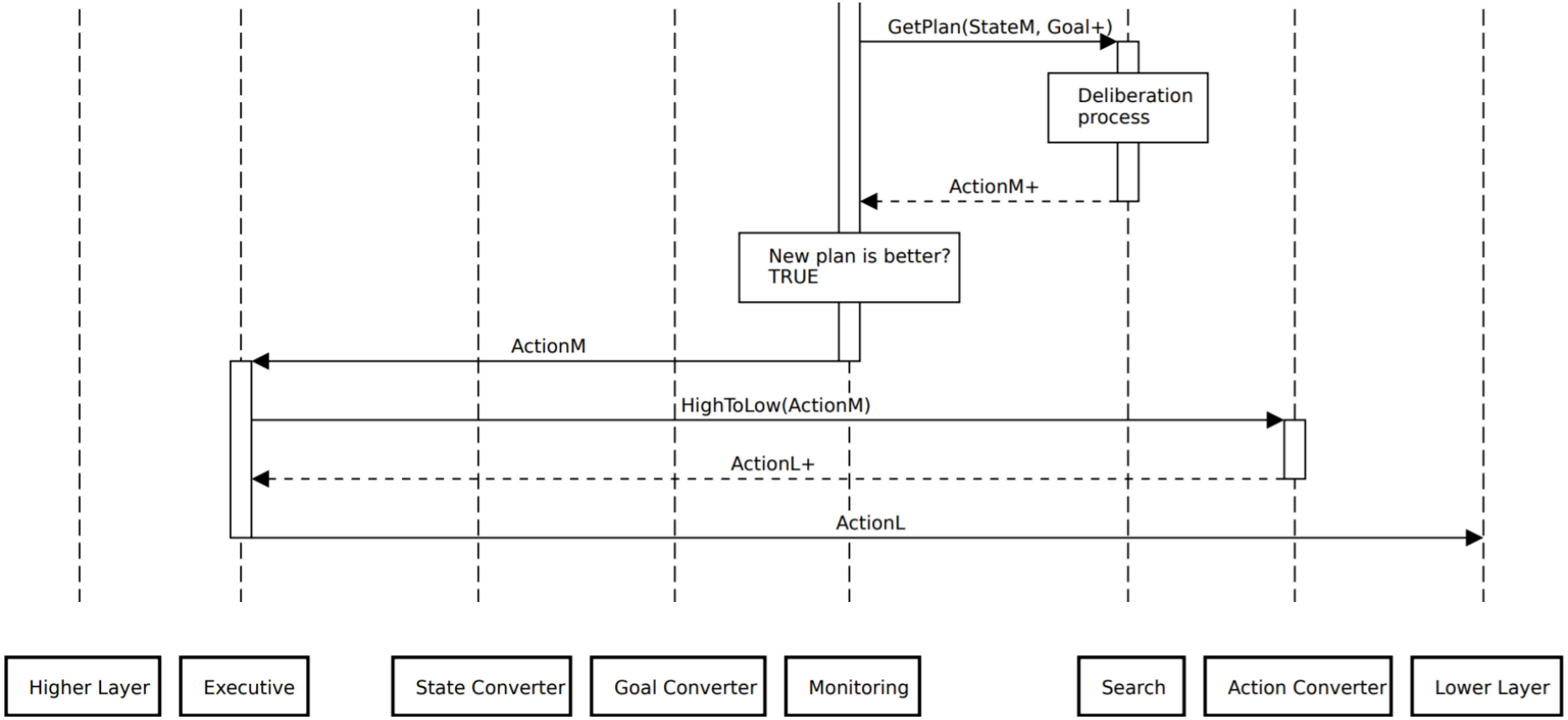


- Quality in "10s suboptimal" is acceptable for the CoBots

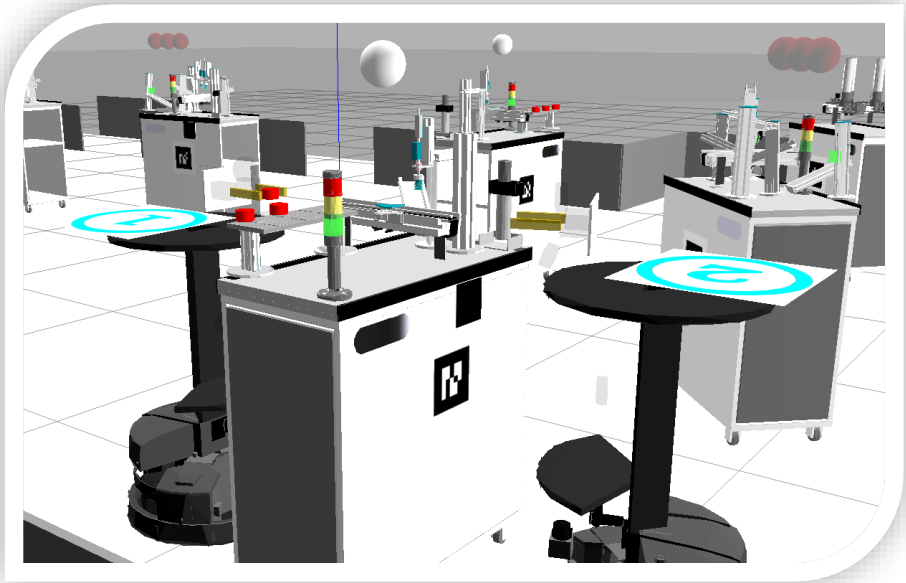
High-level event management



High-level event management



Mlaras in the logistics league

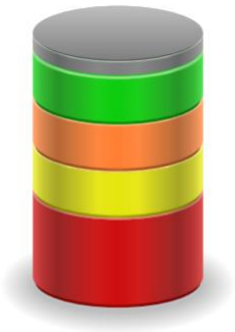


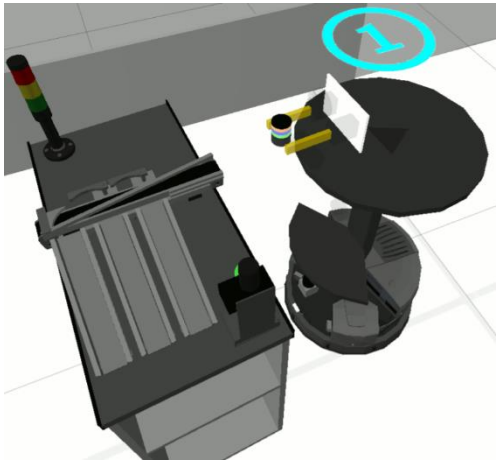
Logistics
Simulation

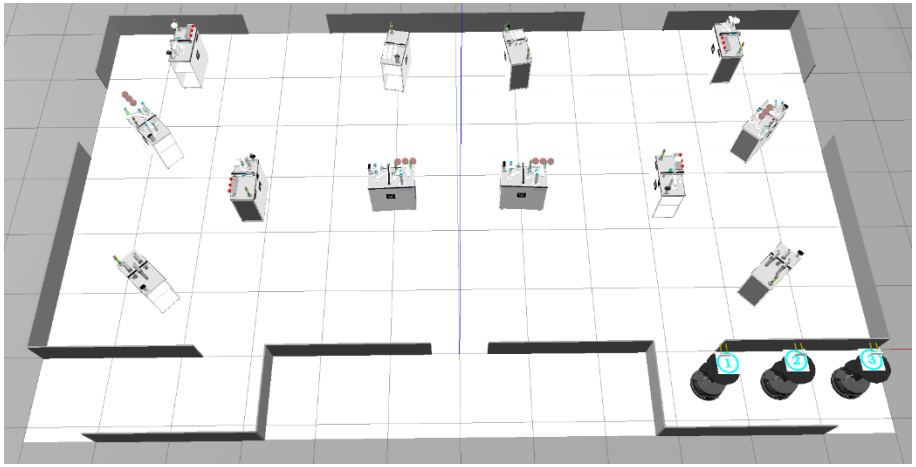
1. Introduction
2. NAOTherapist architecture
3. Proposed guidelines
4. Mlaras architecture
5. Mlaras opportunities & failures
6. **Mlaras in the logistics league**
7. Conclusions

RoboCup logistics simulation league

- **Goal**
  - Product building orders
  - More rings for more score
- **Robots**
  - 2 teams of 3 robots
  - Should cooperate
- **Stations**
  - Base, ring, cap, delivery
  - Errors break them
  - Require bases to work





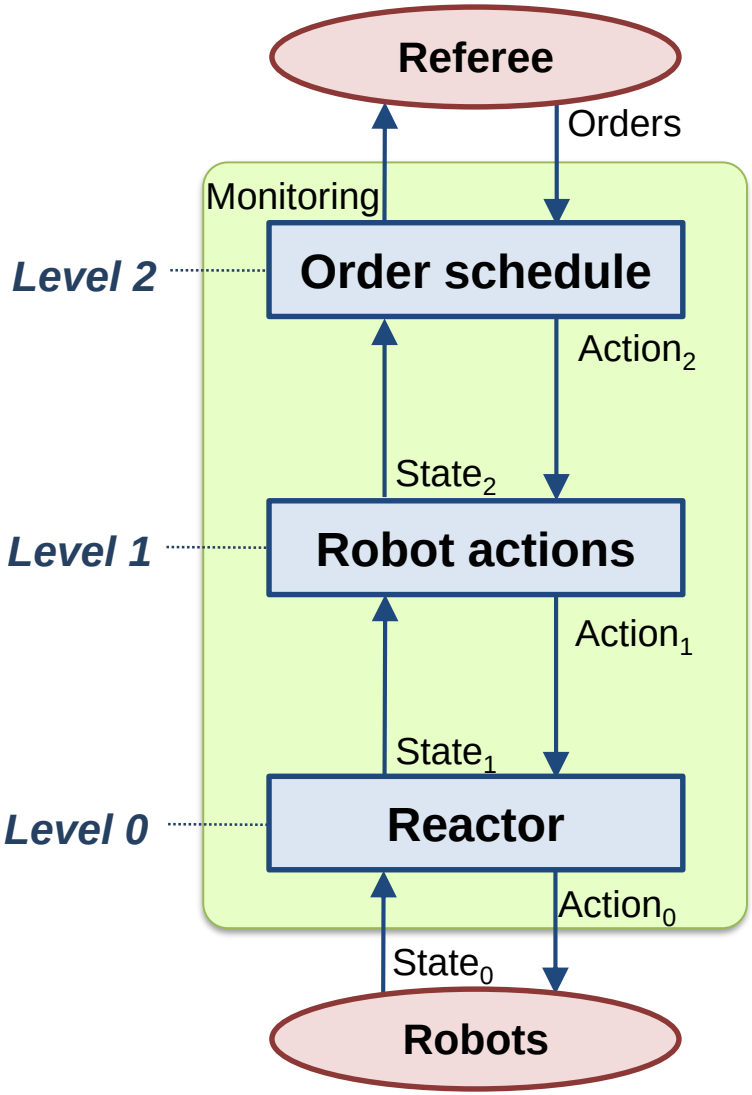


Mlaras logistics for RC logistics

- **Parts replaced**
  - Deliberation
  - Abstraction conversions
- **New PDDL domain**
  - Based on classical planning
  - Centralized plan for the 3 robots
    - Anytime search with 10 seconds
- **Opportunities management**
  - Rescheduling for new orders

```
1 (GO_TO R1 C-CS1 IN START_AREA)
1 (GO_TO R2 C-BS OUT START_AREA)
1 (GO_TO R3 C-CS1 OUT START_AREA)
2 (RETRIEVE_BASE_SHELF R1 C-CS1 IN B_TR)
2 (RETRIEVE_BASE R2 C-BS OUT B_BL)
3 (FEED_CAP_STATION R1 R3 C-CS1)
3 (GO_TO R2 C-RS1 IN C-BS)
4 (RETRIEVE_BASE_SHELF R1 C-CS1 IN B_TR)
4 (FEED_RING_STATION R2 C-RS1 IN B_BL)
4 (GO_TO R3 C-RS2 IN C-CS1)
5 (GO_TO R2 C-BS OUT C-RS1)
5 (GO_TO R1 C-CS2 IN C-CS1)
5 (FEED_RING_STATION R3 C-RS2 IN B_TR)
. . .
```

Mlaras layer overview



Declarative configuration

```
(:action FEED_
    :parameter
                High: MOUNT_CAP(robot_input, robot_o
                Lows: wait(?robot_input, "?station/"  el)
                Lows: prepare_station(?robot_input,
                      bring_product_to(?robot_input,  g

                High: MOUNT_CAP(robot_input, robot_o
                Lows: wait(?robot_output, "?station/
    :precondit  Lows: get_product_from(?robot_output  el)
        (into_
        (robot  High: MOUNT_CAP_ALONE(robot, station  g
        (manag  Lows: wait(?robot, "?station/" + ?st
        (gripp  Lows: prepare_station(?robot, ?stati
        (witho        bring_product_to(?robot, "plac
        (bases                                        el)
        (one_m  High: DELIVER(robot, station, side,
        (ready  High: MOUNT_CAP_ALONE(robot, station  g
    )           Lows: wait(?robot, "?station/" +
    :effect (a  Lows: prepare_station(?robot, ?stati
        (not (        bring_product_to(?robot, "plac  gent_name
        (bases        ?play_sound(etc/ff7fanfare.wav  /1/status
```

External solver

Model state

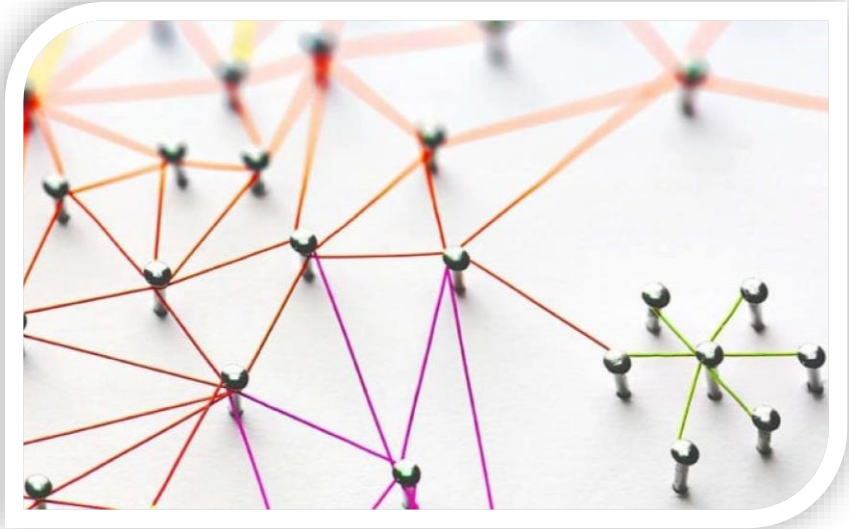
urative nditions

Action mpositions

Results

- **Planning**
  - Planning time of 10 seconds is enough
  - The three agents execute orders in parallel
  - In spite of using classical planning approach
- **Scores**
  - Can finish a 0-ring product and a 3-ring product
  - Over 170 points if there are no obstructions
    - Last winner 139 points, runner-up 32 points
- **Declarative configuration**

Conclusions



1. Introduction
2. NAOTherapist architecture
3. Proposed guidelines
4. Mlaras architecture
5. Mlaras opportunities & failures
6. Mlaras in the logistics league
7. **Conclusions**

Main conclusions

- NAOTherapist was a success as a real project
- Mlaras eases the development of autonomous systems
- Mlaras achieves all thesis objectives
  - Evaluations in 3 real and 1 simulated environments

	Deliberation	Stochastic	Temporal	Declarative	Multilayer	Middleware
LAAS	Cust. AP	Ad hoc	Cust. AP	Ad hoc	✓	-
T-REX	**AP**	**Replan**	**Timelines**	Partial	✓	-
PELEA	**AP**	**Replan**	**Temp. AP**	Partial	X	-
ROSPlan	**AP**	**Replan**	**Temp. AP**	Partial	X	**ROS**
CORTEX	Ad hoc	Ad hoc	Ad hoc	Ad hoc	X	**RoboComp**
Mlaras	**AP**	**Replan**	**Classic AP**	**Use case**	✓	**RoboComp**

Main contributions

- **Guidelines for deliberation in robotics**
  - Focused on automated planning
- **Layered-deliberation implementation**
  - Instantiated in a multiagent logistics environment
- **Mechanisms to widen the classical planning application**
  - Temporal preconditions without reasoning about time
  - Interlayer interruptions for opportunities and failures
- **Declarative languages to ease use case refining**
  - Specially the high to low, which models behavioral trees

Future works

- **Deeper study**
  - **Any-time plan refining**
  - Declarative mechanisms to restart an action
  - Opportunities and failures in automated planning
  - **Graphical tool to generate the declarative files**
  - Automatic opportunities and failures detection
  - **Plan reuse to save deliberation time**
  - Optimize a gain metric to select the best schedules
- **Different use cases**
  - **Improve multi-agent support**
  - **Explore the limits of the layered deliberation of Mlaras**

Publications

- **17 publications since 2014 (4 JCR journal articles)**
  - **2019 JCR journal** *A Socially Assistive Robotic Platform for Upper-Limb Rehabilitation: A Longitudinal Study With Pediatric Patients*: Robotics & Automation Magazine, vol. 26(2), p. 24-39
  - **2019 JCR journal** *Developing a Robot-Guided Interactive Simon Game for Physical and Cognitive Training*: International Journal of Humanoid Robotics, vol. 16(1), p. 1950003
  - **2018** *From High to Low Level and Vice-Versa: A New Language for the Translation between Abstraction Levels in Robot Control Architectures*: in proceedings of the 3rd Workshop on Semantic Policy and Action Representations for Autonomous Robots (SPAR), IROS conference
  - **2018** *Task Monitoring and Rescheduling for Opportunity and Failure Management*: in proceedings of the 2nd IntEx Workshop, ICAPS conference, pp. 24-31
  - **2017** *On the Application of Classical Planning to Real Social Robotic Tasks*: in proceedings of the 5th Workshop on Planning and Robotics (PlanRob), ICAPS conference, pp. 38-47
  - **2017 JCR journal** *Evaluating the Child–Robot Interaction of the NAOTherapist Platform in Pediatric Rehabilitation*: International Journal of Social Robotics, vol. 9(3), pp. 343–358
  - **2017 JCR journal** *A three-layer planning architecture for the autonomous control of rehabilitation therapies based on social robots*: Cognitive Systems Research, vol. 43, pp. 232-249

NAOTherapist accolades

- **2019** *The Collider* entrepreneurship program of the World Mobile Congress
  - Grant of 50,000 €
- **2019** *HealthStart* entrepreneurship program of madri+d
  - Grant of 2,000 €
- **2019** *CaixaImpulse* entrepreneurship program of La Caixa
  - Grant of 70,000 €
- **2017** 1st prize of the act *Implicados y solidarios* of Bankinter
  - Grant of 13,000 €
- **2017** Winner of the *eSalud award* from the Asociación Investigadores en eSalud
- **2017** *National Geographic* exclusive 30-minutes documentary film
- **2016** 3rd prize of the *Yuzz* entrepreneurship program by Banco Santander
  - Grant of 10,000 €

NAOTHERAPIST

References

- **[Alami et al. 1998]** Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An Architecture for Autonomy. The Int. Journal of Robotics Research, 17:pp.315–337
- **[Alcázar et al. 2010]** Alcázar, V., Guzmán, C., Prior, D., Borrajo, D., Castillo, L., and Onaindia, E. (2010). PELEA: Planning, Learning and Execution Architecture. In Proceedings of the 28th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG), Brescia, Italy
- **[Bustos et al. 2019]** Bustos García, P., Manso Argüelles, L., Bandera, A., Bandera, J., García-Varea, I., and Martínez-Gómez, J. (2019). The CORTEX cognitive robotics architecture: Use cases. Cognitive Systems Research, 55:107 – 123
- **[Cashmore et al. 2015]** Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carreraa, A., Palomeras, N., Hurtós, N., and Carrerasa, M. (2015). Rosplan: Planning in the robot operating system. In Proceedings of the Twenty-Fifth International Conference on Int. Conference on Automated Planning and Scheduling, ICAPS'15, pages 333–341.
- **[Chen et al. 2010]** Chen, D.-S.; Batson, R. G.; and Dang, Y. (2010). Applied Integer Programming: Modeling and Solution. John Wiley & Sons
- **[García et al. 2017]** García Estévez, E., Díaz Portales, I., Pulido, J. C., Fuentetaja, R., and Fernández, F. (2017). Enhancing a Robotic Rehabilitation Model for Hand-Arm Bimanual Intensive Therapy. In Proceedings of the 3rd Iberian Robotics Conference.

References

- [Ghallab et al. 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). Automated planning: theory & practice. Elsevier
- [Ghallab et al. 2014] Ghallab, M., Nau, D., and Traverso, P. (2014). The Actor's View of Automated Planning and Acting: A Position Paper. AI, 208:1– 17
- [Kortenkamp et al. 2008] Kortenkamp, D. and Simmons, R. (2008). Springer Handbook of Robotics - Chapter 8, pages 187–206. Springer Berlin Heidelberg
- [McGann et al. 2008] McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., and McEwen, R. (2008). A deliberative architecture for auv control. In 2008 IEEE International Conference on Robotics and Automation, pages 1049–1054.
- [Ögren et al. 2018] Ögren, P. and Colledanchise, M. (2018). Behavior Trees in Robotics and AI: An Introduction. Number 6 in Chapman & Hall/CRC artificial intelligence and robotics series. CRC Press
- [Pulido 2020] Pulido J. C. (2020). Autonomous Socially Assistive Robotics in Pediatric Clinical Practice. Universidad Carlos III de Madrid.

Doct. of Computer Science and Technology

Ph. D. thesis defense

Multi-Layered Architectures for Autonomous Systems

José Carlos González Dorado

Thesis advisors { *Fernando Fernández Rebollo*, *Ángel García Olaya* }

Planning and Learning Group

Thank you for your attention

March 27th, 2020

Computer Science Department

Classical PDDL

Domain (action)

```
(:action pickup
 :parameters (?ob)
 :precondition (and
    (clear ?ob)
    (on-table ?ob)
    (arm-empty))
 :effect (and
    (holding ?ob)
    (not (clear ?ob))
    (not (on-table ?ob))
    (not (arm-empty))))
```

Problem

```
(:init
    (on-table a) (on-table b)
    (clear a) (clear b)
    (arm-empty))
(:goal (and (on a b))))
```

Plan

```
1: pickup(a)
2: stack(a,b)
```

Classical

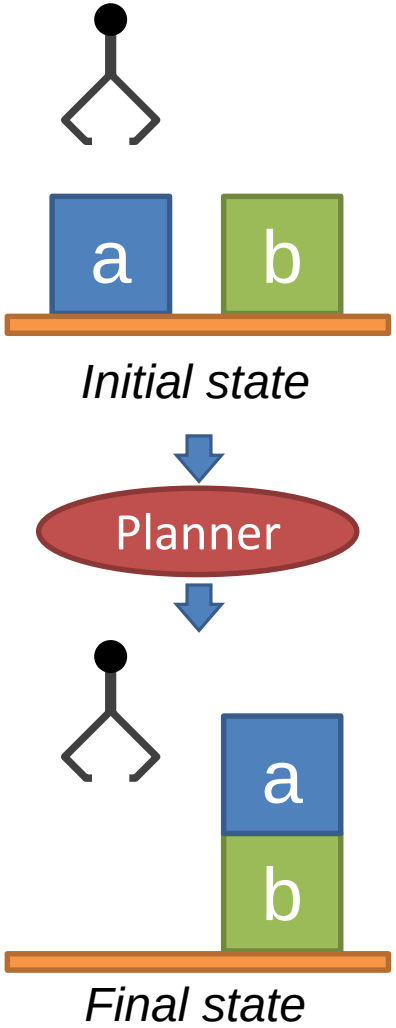
- Simplest
- Fastest

Probabilistic

- Explicit probabilities

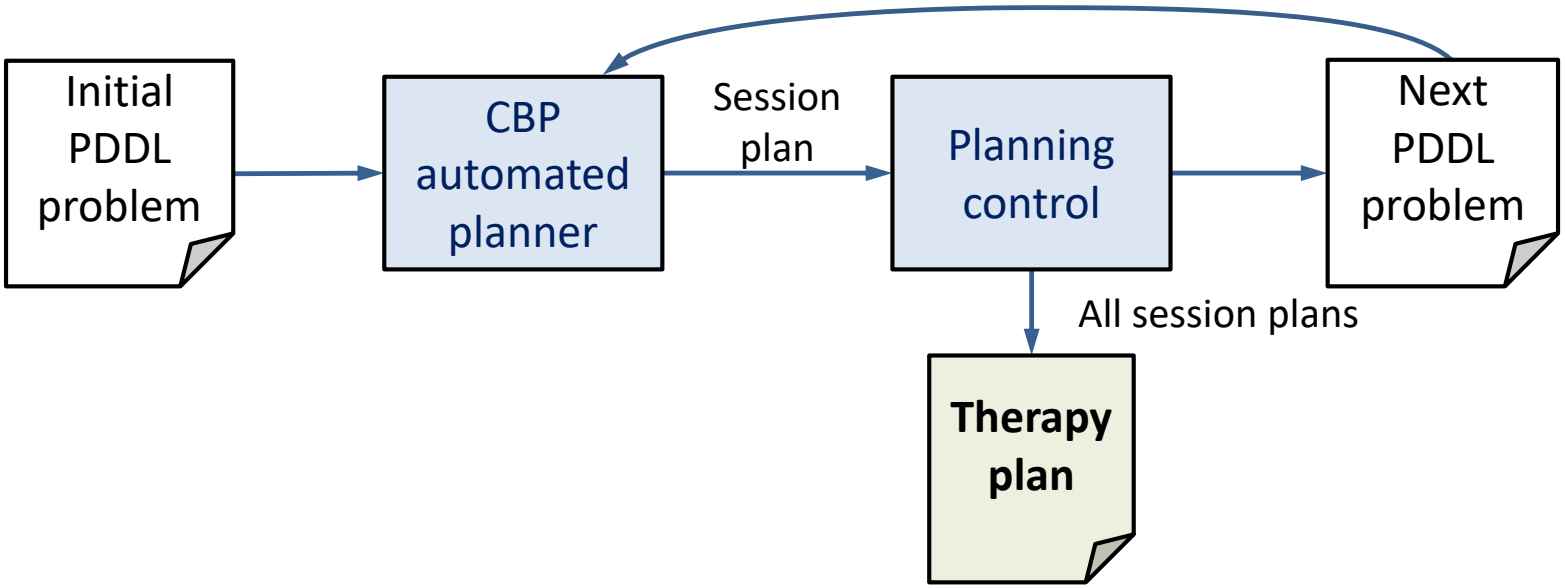
Temporal

- Explicit durations
- Overlapping actions
- Overall and at-end



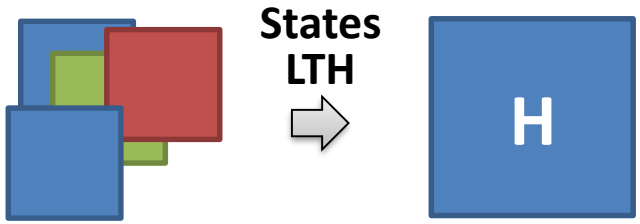
NAOTherapist high-level strategy

- Plan one therapy is harder than individual sessions
- **Divide and conquer strategy**
  - Plan quality is preserved
  - Much less planning time

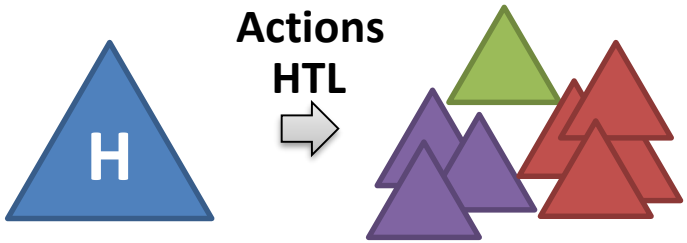


Abstraction layers

- **High-level**
  - High states to deliberate with (PDDL)
  - High actions to define behaviors



- **Low-level**
  - Low states with data from the sensors
  - Low actions are instructions for the robot

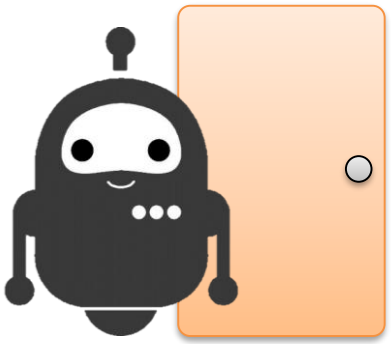


Low state
Image raw data
Depth raw data
. . .

LTH →

High state
Door closed
Handle at right

Deliberation



High actions
Open door
. . .

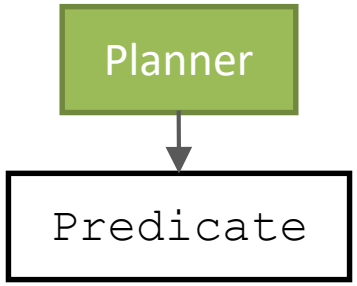
HTL →

Low actions
Approach
Move arm
Grip handle
Turn hand
Push door

State model predicates

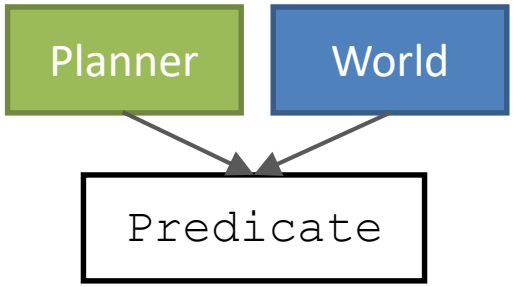
- **Internal**: value is always the expected

```
(total-cost)
(above ?n1 - count ?n2 - Count)
```



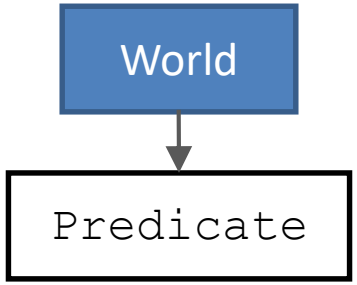
- **Predictable**: value predicted in nominal behavior

```
(validAnswer)
```



- **Unpredictable**
  - Do not appear in any effect
  - Changes triggered externally

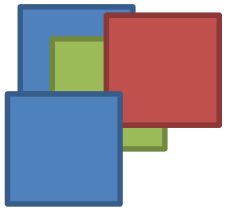
```
(pauseActivated)
```



- **Symbolic or numeric**

State separation

- **Model state**
  - Minimum predicate set to deliberate with in Search
  - Corresponds with $State_L$ and the $State_M$



- **Agent state**
  - Predicates used only to control abstraction conversions
  - Not present in any state (optionally in $State_H$)

- **Layer accesses**
  - Read only: Model state, action parameters
  - Read and write: Agent state
  - Variable resolution order: parameter → agent → model

Rescheduling policy

- **If the scheduler cannot find a suitable plan**
  - **Failures**: Monitoring cancels the next task
    - With the lowest priority first
    - Then the smallest time window that overlaps another
  - **Opportunities**:
    1. Tries to redo the current subtask later
    2. If it cannot: try to redo the whole task
    3. If it cannot: cancel the current task or the new by maximizing the gain

Gain: $g = \sum_{i=1}^{n} p_i$

Sum of the priorities of the scheduled tasks

Durative conditions

- Order is important: First match is selected

```
Medium: deliver-object(object, person, location_a, location_b),
        ?object is coffee
Priority: 10
Cooldown: 360000
Overall: ?held_object is ?object
Opportunity: ?nearest_person is ?person

Medium: introduce-session(pres, robot, pause, 1st_name, 2nd_name)
Overall: ?person and not ?battery and not ?help and not ?stop and
         not ?stop_session
Finish-when: not ?is_speaking

Medium: RETRIEVE_BASE (robot, station, side, color)
Finish-when: ?agent_status == 3 and
             ?agent_last_low_times >= ?pelea/last_low_times
```

Action parameter

Model predicate

Agent predicate

Action decompositions

- Low-level actions of each set are executed in parallel

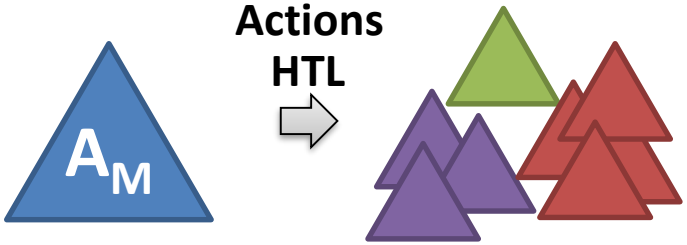
```
High: say(speech,behavior), ?behavior is show_video
High: say(speech,behavior), ?behavior is show_tutorial
Lows: say(?speech)
      show_video(?speech)
      ?pauseatend(600)
```

Internal instruction

```
High: say(speech,behavior)
Lows: say(?speech)

High: finish-pose(), ?counter_skipped_pose < 3
Lows: print("FINISH-POSE (skipped)")
      executeAnimation(green_eyes)
Lows: say(speech_clue)
Lows: executeAnimation(blinking)
      ?set(counter_failed_pose, 0)
      ?increase(counter_skipped_pose, 1)
```

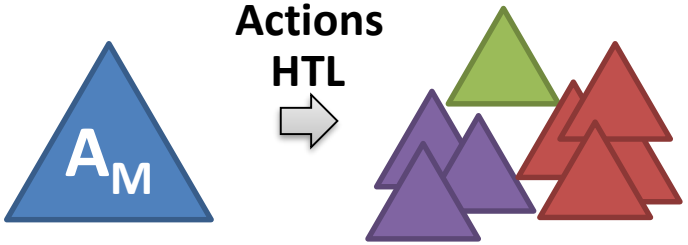
Agent state updates



Action decompositions

- High-level set = Branch of a behavioral tree [?] node
- Powerful and easy tool to refine use cases
- Low-level actions depend on the lower level layer catalog

```
High: finish-exercise-mirror()
High: finish-exercise-simon(mode), ?mode==classic_finish
Lows: print("FINISH-EXERCISE")
      allowAutonomousMovements(true)
      executeAnimation(blinking)
      executeAnimation(initFull)
      say(speech_success)
      ?pauseatend(200)
Lows: say("We have finished this exercise.")
Lows: executeAnimation(blinking)
Lows: say(speech_encouragement)
      ?pauseatend(1000)
```



State generalizations

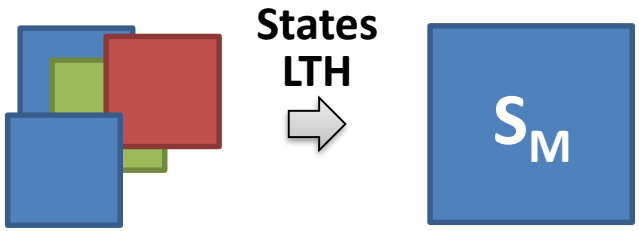
- Order is not important, all conditions are evaluated
- Agent state is only written here

```
If: True
    add(checkPoseResult ?checkPoseResult, model)
    add(exerciseTotalPoses ?exerciseTotalPoses, model)

If: not ?person
If: ?emergency_situation
    delete(can_continue, model)

If: ?checkPoseResult is OK and ?ready
    increase(performedPoses 1, agent)
```

Agent state update



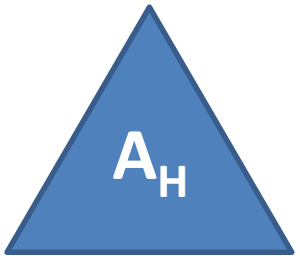
Goal relations

- All *Action*$_H$ received in parallel are checked sequentially

```
High: execute_order(id, base, ring1, ring2, ring3, cap, gate),
          ?ring1 is void and ?ring2 is void and ?ring3 is void
Goal: add(product_piece ?id zero ?base, goals)
      add(product_cap ?id ?cap, goals)
      add(product_gate/~?id ?gate, agents)
      add(delivered ?id, goals)

High: execute_order(id, base, ring1, ring2, ring3, cap, gate),
          ?ring3 is void
Goal: add(product_piece ?id zero ?base, goals)
      add(product_piece ?id one ?ring1, goals)
      add(product_piece ?id two ?ring2, goals)
      add(product_cap ?id ?cap, goals)
      add(product_gate/~?id ?gate, agents, goals)
      add(delivered ?id, goals)
```

Agent state updates



RoboCup logistics simulation league

- **Products**
  - Base (3 colors)
  - From 0 to 3 rings (4 colors)
  - Cap (2 colors)

- **Orders**
  - Supplied by the referee
  - Product, gate and time window
  - Appear randomly
  - **Completion gives score**
  - Only 15 minutes