23rd International Conference on Automated Planning and Scheduling
Rome, Italy 10-14 June 2013
Workshop on Planning and Robotics

Deliberative Systems for Autonomous Robotics: A Brief Comparison Between Action-oriented and Timelines-based Approaches

Pablo Muñoz and María D. R-Moreno
{pmunoz,mdolores}@aut.uah.es

Sponsored by
Under a NPI initiative

Universidad de Alcalá

Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

Contents

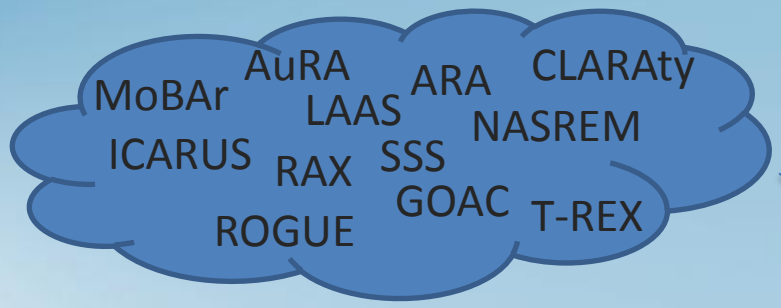
- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

Introduction

- Autonomous control for robotic missions is a complex task
- Involves multiple disciplines:
  - Mechanical: robot design and construction
  - Engineering: implementation of software to interoperate the robot
  - Artificial intelligence: give the robot the ability to decide what to do

Introduction

- The constraints are hard, a robot must:
  - Performs the tasks for which is designed for
  - And do it as efficiently as possible!
  - Deal with uncertainty and dynamic environments
  - And of course, survive!
- These constraints are hard questions and exists a big number of control architectures that try to solve it in different ways



Only a few names…

Introduction

- Also, a lot of different ideas and technologies for the deliberative process
- We will focus on two of them with different approaches:
  - **Action oriented**: based on predicates logic to define the state of the world → MoBAr
  - **Timelines based:** representation of the world as evolution over time for the different attributes of the world → GOAC

Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

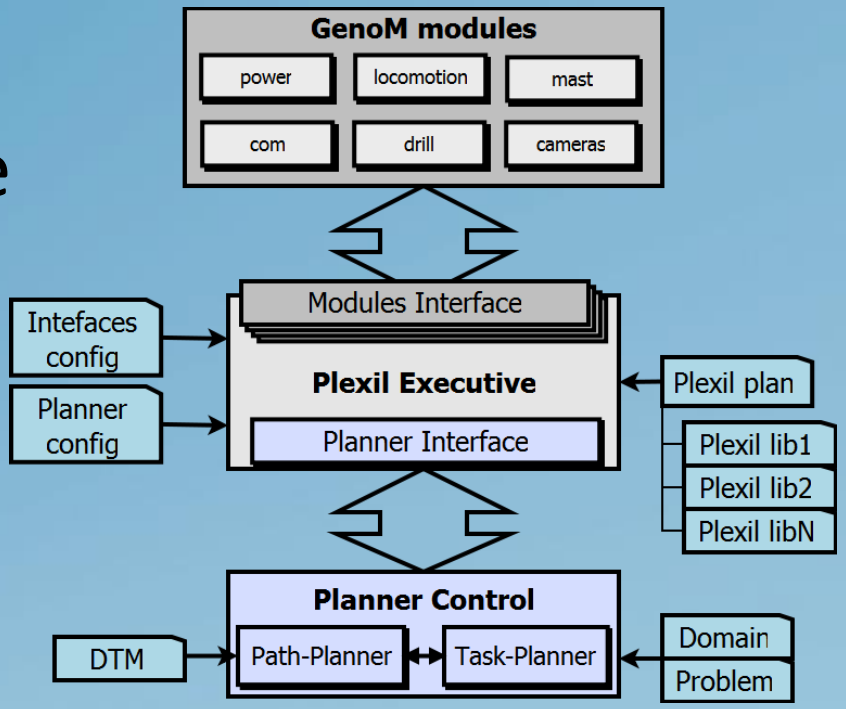
Model-Based Architecture

- Implemented in UAH as a 3 layer architecture with different technologies:
  - Deliberator: modified PDDL-based planner
  - Executor: PLEXIL and Plexil Executive
  - Functional: set of GenoM modules
- The interfaces between layers are fundamental, each layer have a different scope and model

Model-Based Architecture

- An attempt to push the focus on the behavior modeling and swappable components
- In an early stage of development
- You can see it at application showcase:

  → *Wed, June 12 17:00*



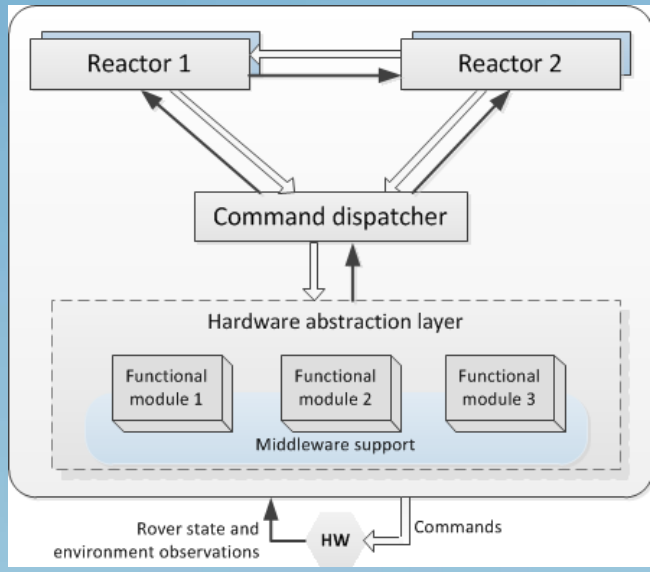
Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

Goal-Oriented Autonomous Controller

- Effort of multiple institutions under ESA-ESTEC contract

- A multi-agent architecture composed of:
  - T-REX for interleaving planning and execution
  - An APSI-based planner for deliberate over a timelines approach into the deliberative reactors
  - GenoM plus BIP to generate reliable, modular and verifiable components for the functional layer

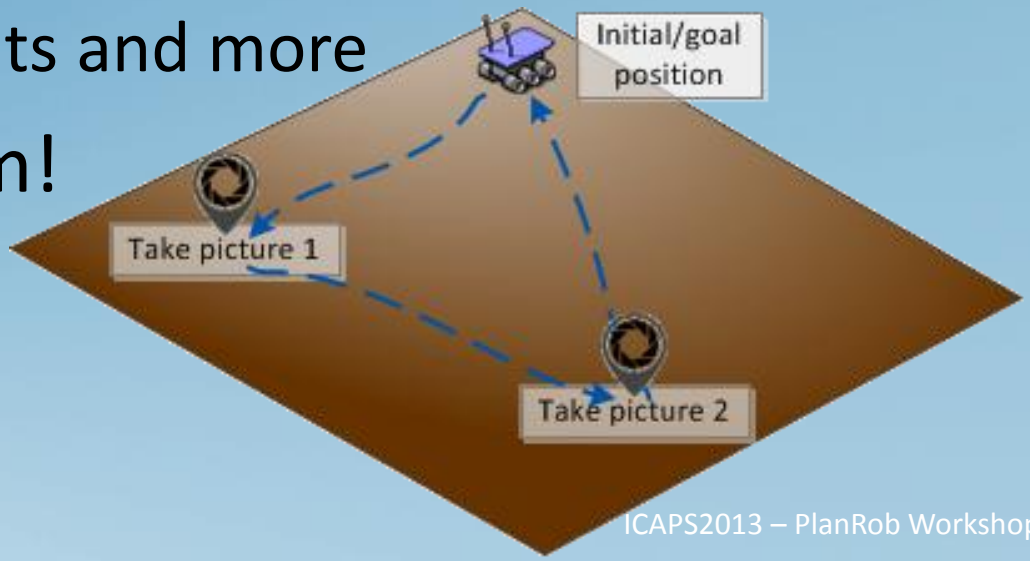


Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

A case of study problem

- A classic scenario for exploration missions:
  - Perform some scientific tasks in different locations
  - Some constraints to consider in the model
  - Implies autonomous navigation, environment monitoring, operate under temporal and energy constraints and more
- Complex problem!



Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

PDDL modeling

- Planning Domain Definition Language
- Model the world as a set of predicates
- The domain specifies a set of actions to change the state of the world
- The problem defines the initial state of the world and the desired goals
- A plan is a sequence of actions to achieve the goals from the initial state of the world

PDDL modeling (Domain)

```
(:durative-action MoveTo
 :parameters (?r - rover ?p1 ?p2 - loc ?n - navmode)
 :duration (= ?duration (/ (distance_to_move ?p1 ?p2) (speed ?r ?n)) )
 :condition (and (over all (has_locomotion ?r))
            (over all (navigation_mode ?r ?n))
            (over all (platine_pos NavCam plat0_0))
            (at start (position ?r ?p1))
            (at start (>= (energy ?r)(*(power_per_m ?r ?n)(distance_to_move ?p1 ?p2)))) )
 :effect (and (at start (not (position ?r ?p1)))
         (at end (position ?r ?p2))
         (at end (decrease (energy ?r)(*(power_per_m ?r ?n)(distance_to_move ?p1 ?p2)))) )
)
(:durative-action MovePlatine
 :parameters (?r - rover ?c - cam ?p1 ?p2 - platpos)
 :duration (= ?duration (time_move_platine ?p1 ?p2))
 :condition (and (at start (platine_pos ?c ?p1))
                 (at start (>= (energy ?r)(*(platine_energy)(time_move_platine ?p1 ?p2)))) )
 :effect (and (at start (not (platine_pos ?c ?p1)))
              (at end (platine_pos ?c ?p2))
              (at end (decrease (energy ?r)(*(platine_energy)(time_move_platine ?p1 ?p2)))))
)
(:durative-action TakePicture
 :parameters (?r - rover ?p - loc ?c - cam ?a - platpos ?m - mode)
 :duration (= ?duration (time_to_picture ?c ?m))
 :condition (and (over all (camera_mode ?r ?c ?m))
                 (over all (position ?r ?p))
                 (over all (platine_pos ?c ?a))
                 (at start (>= (energy ?r) (camera_energy ?c ?m))) )
 :effect (and (at end (picture ?p ?m ?a))
              (at end (decrease (energy ?r) (camera_energy ?c ?m))) )
)
```

PDDL modeling (Problem)

```
(define (problem RoverDemoProb)
 (:domain RoverDemo)
 (:objects
     C0_0 C6_0 C5_-5 - loc
     NavCam - cam  lowRes - mode
     plat0_0 plat15_30 - platpos
     laser stereo - navmode
     dala - rover
 )
 (:init
  ;LOCATIONS INTERCONNECTION
  (= (distance_to_move C0_0 C6_0) 10)
  (= (distance_to_move C6_0 C0_0) 10)
  (= (distance_to_move C0_0 C5_-5) 8)
  (= (distance_to_move C5_-5 C0_0) 8)
  (= (distance_to_move C5_-5 C6_0) 6.4)
  (= (distance_to_move C6_0 C5_-5) 6.4)
  ;ROVER DATA AND CONFIG
  (position dala C0_0)
  (= (energy dala) 1.44)
  (has_locomotion dala)
  (navigation_mode dala laser)
  (= (speed dala laser) 0.2)
  (= (speed dala stereo) 0.1)
  (= (power_per_m dala laser) 0.002)
  (= (power_per_m dala stereo) 0.034)
  (camera_mode dala NavCam lowRes)
  (= (camera_energy NavCam lowRes) 0.008)
  (= (time_to_picture NavCam lowRes) 15)
  (platine_pos NavCam plat0_0)
  (= (platine_energy) 0.003)
  (= (time_move_plat plat0_0 plat15_30) 10)
  (= (time_move_plat plat15_30 plat0_0) 7)
 )
```

```
(:goal (and
 (preference PIC1
   (picture C6_0 lowRes plat15_30))
 (preference PIC2
   (picture C5_-5 lowRes plat15_30))
 (preference POSF (position dala C0_0)) )
)
(:metric minimize (+
 (*(is-violated PIC1) 100)
 (*(is-violated PIC2) 100)
 (*(is-violated POSF) 900)
 (total-time) ))
)
```

Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

DDL and PDL modeling

- Domain/Problem Definition Language
- Model the world as state variables
- The domain model represents the physical interactions between the state variables
- The problem defines the states at the start time, and a set of desired states that must be accomplished at the end of the execution
- Transitions of the state variables define the “plan”

DDL modeling

```
SYNCHRONIZE MissionTimeline.mission_timeline {
 VALUE TakePicture(?file_id1, ?x1, ?y1, ?pan1, ?tilt1) {
   cd1 Camera.camera.TakingPicture(?file_id2, ?x2, ?y2, ?pan2, ?tilt2);
   cd5 Communication.communication.Communicating(?file_id3);
   cd2 MissionTimeline.mission_timeline.Idle();
   CONTAINS [0,+INF] [0,+INF] cd1;
   CONTAINS [0,+INF] [2,2] cd5;
   MEETS cd2;
   cd1 BEFORE [0, +INF] cd5;
   ?x1 = ?x2;
   ?y1 = ?y2;
   ?pan1 = ?pan2;
   ?tilt1 = ?tilt2;
   ?file_id1 = ?file_id2;
   ?file_id1 = ?file_id3;
} }
SYNCHRONIZE RobotBase.robot_base {
 VALUE GoingToLaser(?x1, ?y1) {
   cd2 Platine.platine.PointingAt(?pan1 = 0, ?tilt1 = 0);
   cd3 Terrain.terrain.FlatTerrain();
   DURING [0, +INF] [0, +INF] cd2;
   DURING [0, +INF] [0, +INF] cd3;
} }
```

PDL modeling

```
PROBLEM RoverDemoProb (DOMAIN RoverDemo) {
mtl0 <fact> MissionTimeline.mission_timeline.Idle() AT [0,0] [1,+INF][1,+INF];
comvw1 <fact> Communication.communication_windows.Visible() AT [70,70][90,90][20,20];
comvw2 <fact> Communication.communication_windows.Visible() AT [150,150][180,180][30,30];
pos0 <fact> RobotBase.robot_base.At(?x1=0, ?y1=0) AT [0,0] [1,+INF] [1,+INF];
or0 <fact> Platine.platine.PointingAt(?pan1=0, ?tilt1=0) AT [0,0][1,+INF][1,+INF];
cam0 <fact> Camera.camera.CamIdle() AT [0,0][1,+INF][1,+INF];
com0 <fact> Communication.communication.CommIdle() AT [0,0][1,+INF][1,+INF];
tp1 <goal> MissionTimeline.mission_timeline.TakePicture
    (?gfile_id2=1, ?gx2=6, ?gy2=0, ?gpan2=15, ?gtilt2=30) AT [10,10][80,90][70,80];
tp2 <goal> MissionTimeline.mission_timeline.TakePicture
    (?gfile_id1=2, ?gx1=5, ?gy1=-5, ?gpan1=15, ?gtilt1=30) AT [80,95][160,185][80,90];
gp <goal> MissionTimeline.mission_timeline.At(?gx3=0, ?gy3=0) AT[160,190][161,+INF][1,+INF];
}
```

Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

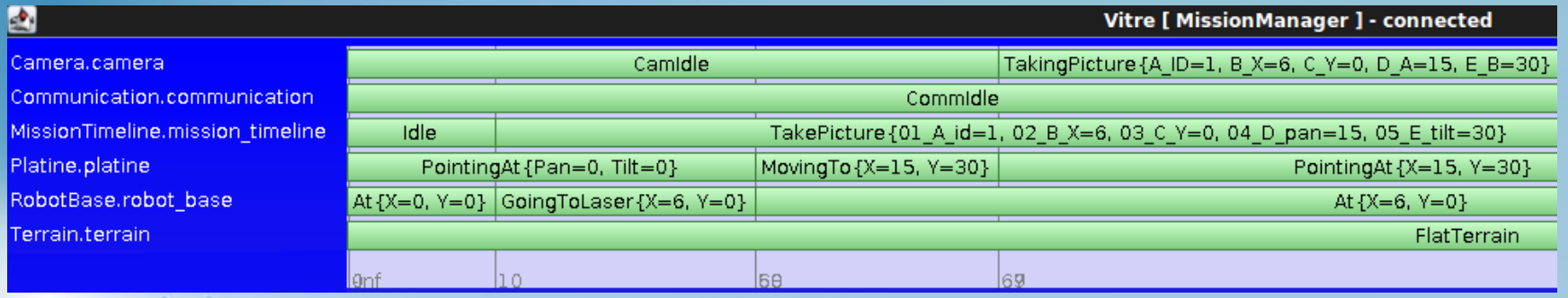
Brief analysis

- Plans generated by each system:

→ PDDL

```
  0.001: (MOVE_TO  C0_0 C6_0 LASER) [50.0000]
 50.002: (MOVEPLATINE NAVCAM PLAT0_0 PLAT15_30) [10.0000]
 60.003: (TAKEPICTURE C6_0 NAVCAM PLAT15_30 LOWRES) [15.0000]
 75.004: (MOVEPLATINE NAVCAM PLAT15_30 PLAT0_0) [7.0000]
 82.005: (MOVE_TO C6_0 C5_-5 LASER) [32.0000]
114.006: (MOVEPLATINE NAVCAM PLAT0_0 PLAT15_30) [10.0000]
124.007: (TAKEPICTURE C5_-5 NAVCAM PLAT15_30 LOWRES) [15.0000]
139.008: (MOVEPLATINE NAVCAM PLAT15_30 PLAT0_0) [7.0000]
146.009: (MOVE_TO C5_-5 C0_0 LASER) [36.0000]
```

→ DDL/PDL



Brief analysis

- In MoBAr PDDL gives a fixed plan
- DDL/PDL in GOAC works as a reactive planner, evolving with the observation of the environment
- So, when an unexpected situation is detected or a goal is added:
  - MOBAR requires a complete new plan from scratch
  - GOAC propagates the change over the timelines and performs replanning over its temporal horizon

Brief analysis

- How do they manage the time?
  - PDDL uses durative actions to define the time required
  - DDL/PDL reason over temporal state variables with flexible bounded periods
- How it affects to the planning process?
  - First one gives a fixed duration time for every action
  - The second has a flexible behavior and transitions are linked via temporal relationships

Brief analysis

- And resources?
  - PDDL uses fluent to represent continuous variables, useful to model basic resources, but planners usually do not work well with it (personal experience)
  - DDL/PDL does not contain a resource model

Contents

- Introduction
- Model-Based Architecture
- Goal-Oriented Autonomous Controller
- A case of study problem
- PDDL modeling
- DDL and PDL modeling
- Brief analysis
- Conclusions

Conclusions

- This is an initial work that tries to characterize the capabilities and performance of different deliberative systems for autonomous control
- It is important to analyze how these systems manages time and resources, a critical point in robotics
- From our experience PDDL is easier to model and understand, but DDL/PDL provides a better way to define the complex interactions of the environment

23rd International Conference on Automated Planning and Scheduling
Rome, Italy 10-14 June 2013
Workshop on Planning and Robotics

Thanks for your attention!

Pablo Muñoz and María D. R-Moreno
{pmunoz,mdolores}@aut.uah.es
Universidad de Alcalá