

From AgentSpeak to C for Unmanned Aerial Vehicles



Samuel Bucheli^a

Department of Computer Science
University of Oxford

May 28, 2015

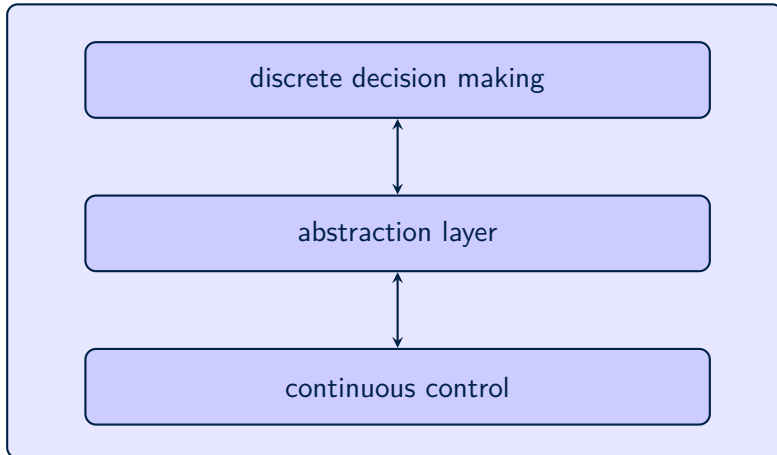
^ajoint work with Daniel, Ruben, and Ashutosh

In this Talk



- autonomous agents as hybrid systems,
- safety considerations: DO-178C and MISRA C,
- AgentSpeak as a modeling language for high-level behavior,
- from AgentSpeak to C: examples and experiments,
- further work: translation validation.

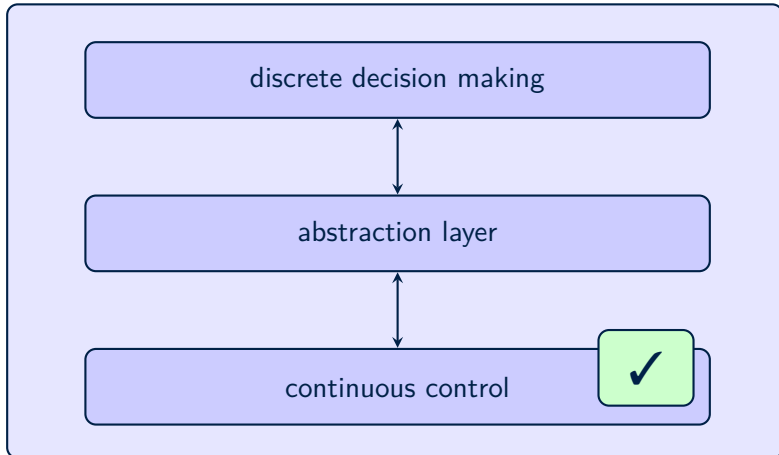
hybrid system



Autonomous Agents

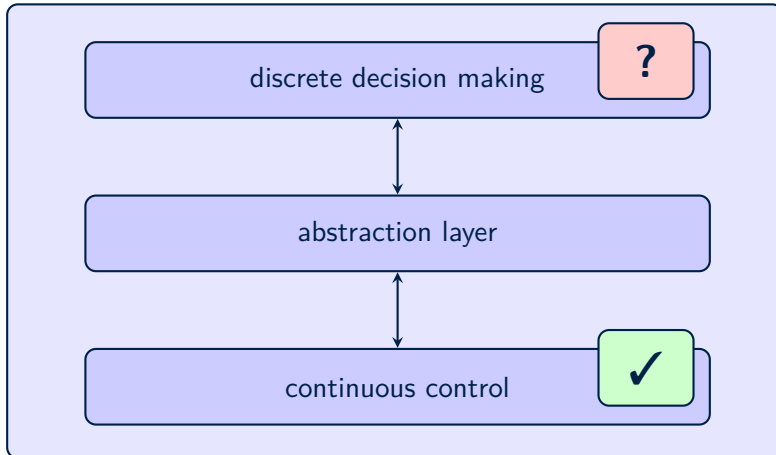


hybrid system



Autonomous Agents

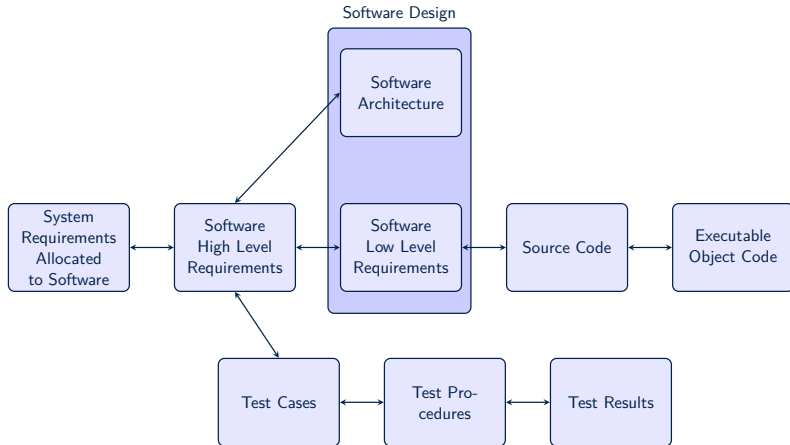
hybrid system



DO-178C



Software Considerations in Airborne Systems and Equipment Certification



MISRA C

Motor Industry Software Reliability Association



DEPARTMENT OF
**COMPUTER
SCIENCE**

...

16.2 (req) Functions shall not call themselves,
either directly or indirectly.

...

20.4 (req) Dynamic heap memory allocation shall not be used.

...



The Autopilot



DEPARTMENT OF
**COMPUTER
SCIENCE**

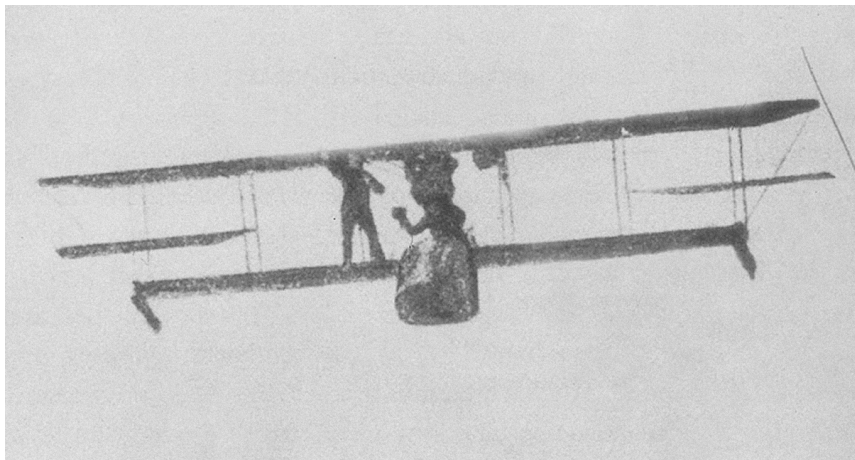


image from <http://fly.historicwings.com/2012/08/george-the-autopilot/>

Sample Flight Plan



takeoff

goto 0 0 1 0

goto 1 0 1 0

goto 1 1 1 0

goto 0 1 1 0

goto 0 0 1 0

land

Sample Flight Plan



engage rotors,
bring aircraft
to altitude

takeoff

```
goto 0 0 1 0  
goto 1 0 1 0  
goto 1 1 1 0  
goto 0 1 1 0  
goto 0 0 1 0
```

```
land
```

Sample Flight Plan

engage rotors,
bring aircraft
to altitude

takeoff

move aircraft
towards target
until reached

```
goto 0 0 1 0  
goto 1 0 1 0  
goto 1 1 1 0  
goto 0 1 1 0  
goto 0 0 1 0
```

land

Sample Flight Plan

engage rotors,
bring aircraft
to altitude

takeoff

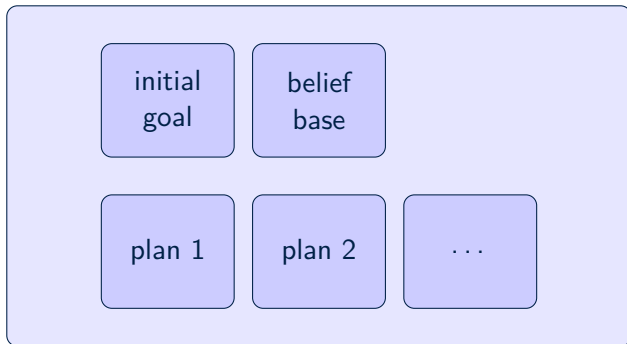
move aircraft
towards target
until reached

```
goto 0 0 1 0  
goto 1 0 1 0  
goto 1 1 1 0  
goto 0 1 1 0  
goto 0 0 1 0
```

lower aircraft,
disengage rotors

land

agent



Plans



```
event
  : condition & condition & ... & condition
  <- formula;
  formula;
  ...
  formula.
```

Plans



triggering events
the plan is
relevant for

event

```
: condition & condition & ... & condition  
<- formula;  
  formula;  
  ...  
  formula.
```

Plans



event

```
: condition & condition & ... & condition  
<- formula;  
   formula;  
   ...  
   formula.
```

preconditions
when the plan
is applicable

A light blue rectangular box containing the text 'preconditions when the plan is applicable'. A black arrow originates from the top-left corner of this box and points upwards and to the left, terminating at the first 'condition' in the event definition's condition list.

Plans



event

: condition & condition & ... & condition

```
<- formula;  
   formula;  
   ...  
   formula.
```

actual body
of the plan to
be executed

A light blue rectangular box containing the text 'actual body of the plan to be executed' is connected by a black line to a light blue rectangular box containing the code 'formula;' lines. An arrow points from the bottom of the code box to the top of the text box, indicating that the code represents the actual body to be executed.

Plans, concretely

```
+!goto(Target) : takenOff
```

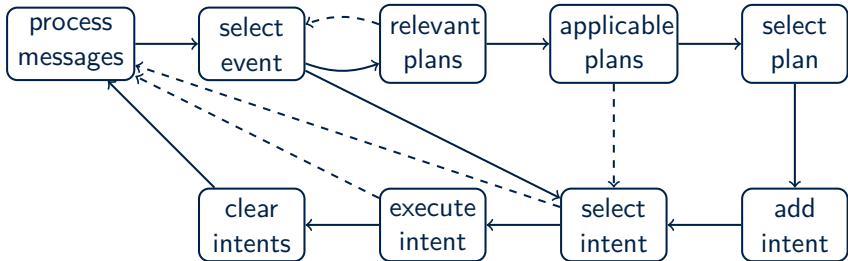
```
<- sendHover();  
+lastTarget(Target);  
!completeGoto.
```

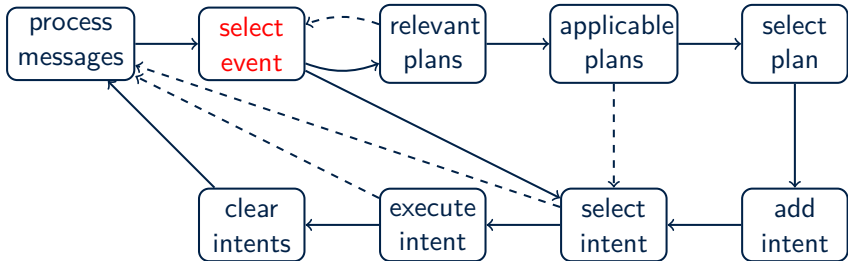
```
+!completeGoto
```

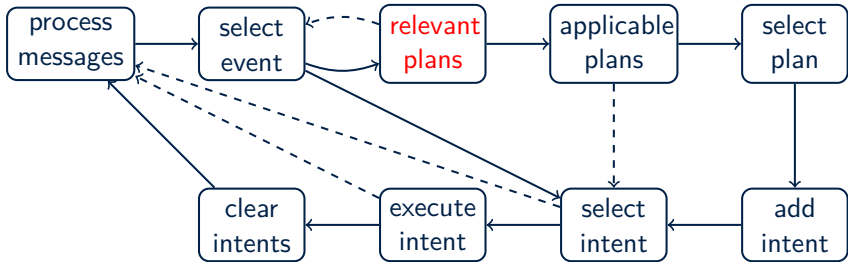
```
: takenOff & myPosition(Pos)  
  & lastTarget(Target) & not closeEnough(Pos, Target)  
<- Movement = calculateMovement(Pos, Target);  
  sendControl(Movement);  
  !completeGoto.
```

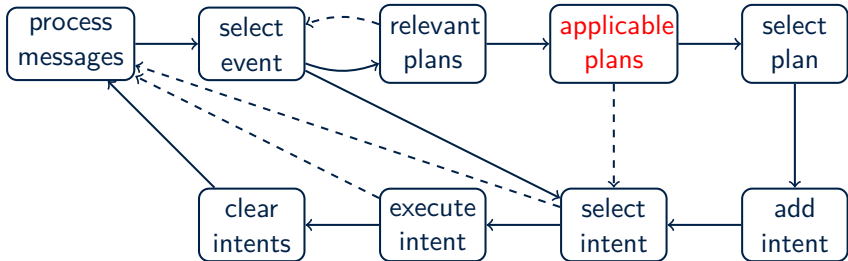
```
+!completeGoto
```

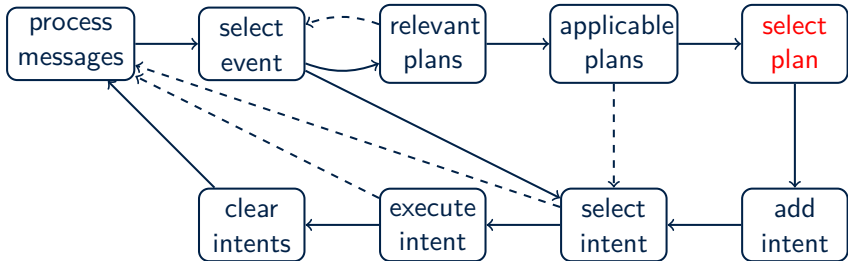
```
: takenOff & myPosition(Pos)  
  & lastTarget(Target) & closeEnough(Pos, Target)  
<- notifyUser("target reached").
```

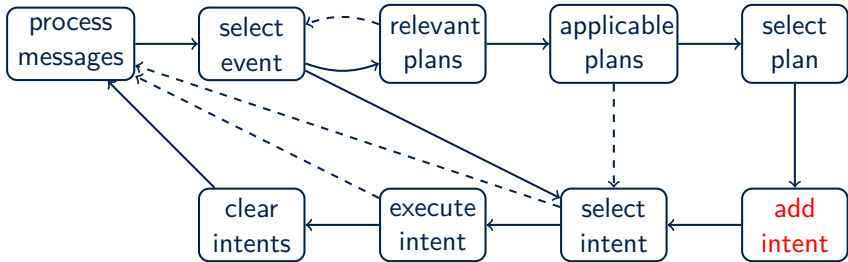


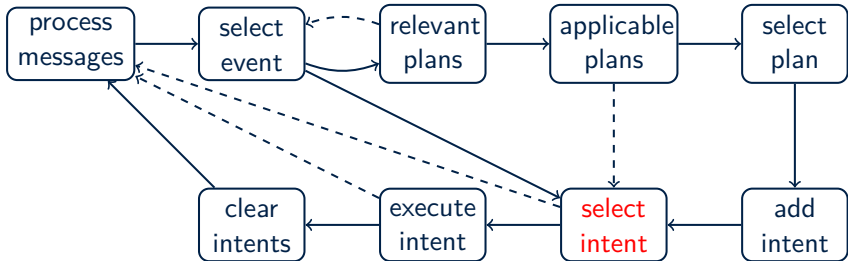


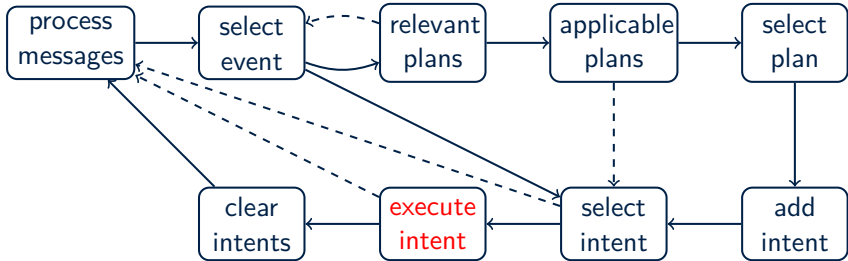


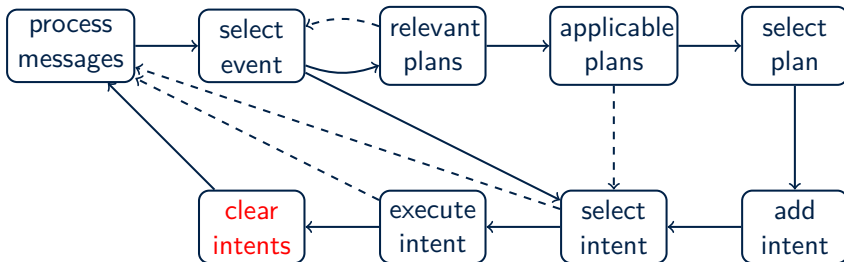












Our Customization



- “run-to-completion” style scheduling,
- only “tail recursion” allowed,
- belief base holds at most one instance of a literal

Translation

Relevant Plans Selection

```
void next_step(void) {
    updateBeliefs();
    eventt event = get_next_event();
    switch (event.trigger) {
        /* ... */
        case ADD_ACHIEVE_GOTO:
            add_achieve_goto(event.goto_param0);
            break;
        case ADD_ACHIEVE_COMPLETEGOTO:
            add_achieve_completeGoto();
            break;
        /* ... */
    }
}
```

Translation

Applicable Plan Selection

```
void add_achieve_goto(positiont param0) {  
    /* try first plan */  
    if (add_achieve_completeGoto_plan0()) {  
        return;  
    }  
    /* try second plan */  
    if (add_achieve_completeGoto_plan1()) {  
        return;  
    }  
  
    /* ... handle the case where no plan is applicable ... */  
  
    return;  
}
```

Translation

Plans



`+!goto(Target) : takenOff`

```
<- sendHover();  
+lastTarget(Target);  
!completeGoto.
```

```
bool add_achieve_goto_plan0(positiont param0) {
```

```
    positiont Target = param0;
```

```
    if (!takenOff_set) { return false; }
```

```
    sendHover();
```

```
    lastTarget_set = true;
```

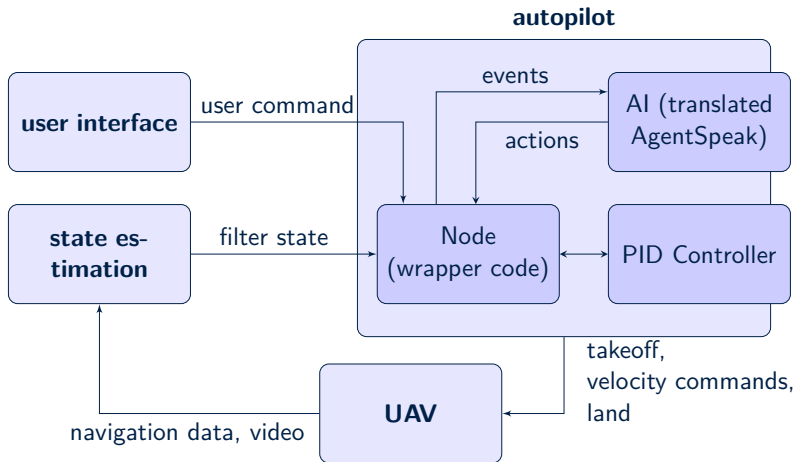
```
    lastTarget_param0 = Target;
```

```
    internal_achieve_completeGoto();
```

```
    return true;
```

```
}
```


Experimental Setup



Test Flight



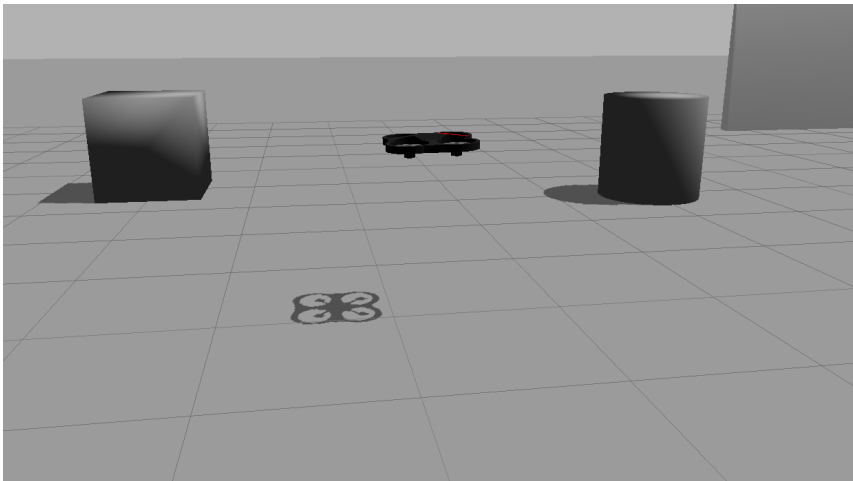
The screenshot displays the PTAM drone GUI with the following components:

- Autopilot Panel:**
 - Node Communication Status: Drone Navdata: 203 Hz, Drone Control: 0 Hz, Pose Estimates: 31 Hz, Pings (RTT): 999 (5008), 999 (20k), Motors: 0.000000 0.000000 0.000000
 - Autopilot Status: (Empty)
 - State Estimation Status: PTAM: Idle, Map: -, Scale: 1.000 (1 in, 0 out), ecc: 0.51, ScaleFovPoint: Fov, Drone Status: Landed (100 Battery)
 - Control Source: Autopilot, None, Ping Drone (every 1s)
 - Buttons: Clear and Send, Clear, Send, Manual, Land, Takeoff, ToggleState, FlatTrim, ToggleCam
 - Messages: PTAM has been reset, Video resolution: 640 x 360
- 3D Simulation:** A 3D environment with a grid floor, a black cube, a black cylinder, and a drone model.
- PTAM Drone Map View:** A 2D grid map showing the drone's position with a red crosshair. Drone Pos: $xyz=(0.66, 0.01, 0.43), rpy=(0.01, -0.03, 0.00)$
- PTAM Drone Camera Feed:** A camera view showing a 3D scene with a blue crosshair. Real Time Factor: 0.31, Sim Time: 00:00. Hint: Point camera at planar scene and press spacebar to start tracking for initial map. Quality: best, scale: 1.000 (ecc: 0.513), PTAM time: 0 ms (0 ms total)

Test Flight



DEPARTMENT OF
**COMPUTER
SCIENCE**



Test Flight



The image shows two side-by-side windows from a ROS-based drone control interface.

tum_arndrone GUI

- Autopilot**: A text area containing flight plan commands: `takeoff`, `goto 0 0 1 0`, `goto 1 0 1 0`, `goto 1 1 1 0`, `goto 0 1 1 0`, and `goto 0 0 1 0`. Below the text area is a `land` button.
- Node Communication Status**:
 - Drone Navdata: 202 Hz
 - Drone Control: 0 Hz
 - Pose Estimates: 31 Hz
 - Pings (RTT): 999 (5008), 999 (20kB)
 - Motors: 0.000000 0.000000 0.000000
- Autopilot Status**: An empty text area.
- Stateestimation Status**:
 - PTAM: Idle
 - Map: -
 - Scale: 1.000 (1 in, 0 out), acc: 0.51
 - ScaleFixpoint: FIX
 - Drone Status: Landed (100 Battery)
- Control Source**:
 - Autopilot
 - None
 - Ping Drone (every 1s)
- Manual**: A row of buttons: `Land`, `Takeoff`, `ToggleState`, `FlatTrim`, and `ToggleCam`.
- Messages**:
 - PTAM has been reset.
 - Video resolution: 640 x 360
 - Load File /home/ros/catkin_ws/src/tum_arndrone/flightPlans/test.txt
- Load File**: A dropdown menu showing `test.txt`.
- Buttons**: `Clear and Send`, `Clear`, and `Send`.

PTAM Drone Map View

- A 2D grid map showing the drone's position. A red crosshair indicates the drone's current location.
- Drone Pose**: `xyz=(-0.00, 0.02, 0.43), rpy=(-0.23, 0.29, -0.00)`

Test Flight



tum_ardrone GUI

Autopilot

```
takeoff  
  
goto 0 0 1 0  
goto 1 0 1 0  
goto 1 1 1 0  
goto 0 1 1 0  
goto 0 0 1 0  
  
land
```

Load File:

Manual

Messages

```
PTAM has been reset.  
Video resolution: 640 x 360  
Load File /home/ros/catkin_ws/src/tum_ardrone/flightPlans/test.txt  
Autopilot: Start Controlling  
notification: taking off  
sent: Takeoff
```

Node Communication Status

```
Drone Navdata: 207 Hz  
Drone Control: 96 Hz  
Pose Estimates: 32 Hz  
Pings (RTT): 999 (500B), 999 (20kB)  
Motors: 0.000000 0.000000 0.000000
```

Autopilot Status:

Stateestimation Status:

```
PTAM: Idle  
Map: -  
Scale: 1.000 (1 in, 0 out), acc: 0.51  
ScaleFixpoint: FIX  
Drone Status: Flying (99 Battery)
```

Control Source:

Autopilot None

Ping Drone (every 1s)

PTAM Drone Map View

Drone Pose: xyz=(-0.11, 0.18, 0.99), rpy=(0.07, 0.39, 3.05)

Test Flight



The image displays a drone flight control interface, split into two main panels.

Left Panel: tum_ardrone GUI

- Autopilot:** A text area containing flight plan commands: `takeoff`, `goto 0 0 1 0`, `goto 1 0 1 0`, `goto 1 1 1 0`, `goto 0 1 1 0`, and `goto 0 0 1 0`. Below this is a `land` command.
- Node Communication Status:** Displays system metrics: Drone Navdata: 209 Hz, Drone Control: 97 Hz, Pose Estimates: 32 Hz, Pings (RTT): 999 (500B), 999 (20kB), and Motors: 0.000000 0.000000 0.000000.
- Autopilot Status:** A large empty text box.
- Stateestimation Status:** Shows PTAM: Idle, Map: -, Scale: 1.000 (1 in, 0 out), acc: 0.51, ScaleFixpoint: FIX, and Drone Status: Flying (98 Battery).
- Control Source:** Radio buttons for Autopilot and None. A checkbox for Ping Drone (every 1s) is also present.
- Manual:** Buttons for `Land`, `Takeoff`, `ToggleState`, `FlatTrim`, and `ToggleCam`.
- Messages:** A scrollable log showing: Video resolution: 640 x 360, Load File /home/ros/catkin_ws/src/tum_ardrone/FlightPlans/test.txt, Autopilot: Start Controlling, notification: taking off, sent: Takeoff, notification: taken off, notification: target initially reached, notification: goto completed, notification: target initially reached, notification: goto completed.

Right Panel: PTAM Drone Map View

- A 3D perspective view of a grid representing the drone's flight path. A green line traces a path that starts at the origin, moves to the right, then up, then left, then down, and finally back to the origin.
- A red crosshair indicates the current drone position.
- At the bottom, the drone's pose is displayed: `Drone Pose: xyz=(0.99, 0.62, 1.02), rpy=(0.30, 0.17, -0.32)`.

Test Flight



The screenshot displays a drone flight control interface with two main windows: 'tum_drone GUI' and 'PTAM Drone Map View'.

tum_drone GUI

- Autopilot**: A text input field containing the following commands:

```
takeoff
goto 0 0 1 0
goto 1 0 1 0
goto 1 1 1 0
goto 0 1 1 0
goto 0 0 1 0
land
```
- Node Communication Status**:
 - Drone Navdata: 198 Hz
 - Drone Control: 0 Hz
 - Pose Estimates: 30 Hz
 - Pings (RTT): 999 (5008), 999 (20kB)
 - Motors: 0.000000 0.000000 0.000000
- Autopilot Status**: An empty text box.
- Stateestimation Status**:
 - PTAM: Idle
 - Map: -
 - Scale: 1.000 (1 in, 0 out), acc: 0.51
 - ScaleFixpoint: FIX
 - Drone Status: Landed (96 Battery)
- Control Source**:
 - Autopilot
 - Manual
 - None
 - Ping Drone (every 1s)
- Manual**: Buttons for 'Land', 'Takeoff', 'ToggleState', 'FlatTrim', and 'ToggleCam'.
- Messages**: A scrollable log showing the following sequence of events:

```
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
sent: land
notification: landing
```
- Load File**: A dropdown menu showing 'test.txt' with 'Clear and Send', 'Clear', and 'Send' buttons.

PTAM Drone Map View

- A top-down grid map showing the drone's path. A red arrow indicates the current heading, and a green outline shows the flight path.
- Drone Pose**: $xyz=(-0.01, 0.18, 0.34)$, $rpm=(-0.03, -0.01, -2.35)$

Test Flight



The image shows two windows from a drone control interface. The left window, titled 'tum_drone GUI', contains the following sections:

- Autopilot:** A text area with commands: 'takeoff', 'goto 0 0 10', 'goto 1 0 10', 'goto 1 1 10', 'goto 0 1 10', 'goto 0 0 10', and 'land'.
- Load File:** A dropdown menu showing 'test.txt'.
- Buttons:** 'Clear and Send', 'Clear', and 'Send'.
- Manual:** 'Land', 'Takeoff', 'ToggleState', 'FlatTrim', and 'ToggleCam' buttons.
- Messages:** A scrollable log with the following text:
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
notification: target initially reached
notification: goto completed
sent: land
notification: landing
- Node Communication Status:**
Drone Navdata: 204 Hz
Drone Control: 0 Hz
Pose Estimates: 31 Hz
Pings (RTT): 999 (500B), 999 (20kB)
Motors: 0.000000 0.000000 0.000000
- Autopilot Status:** An empty text area.
- Stateestimation Status:**
PTAM: Idle
Map: -
Scale: 1.000 (1 in, 0 out), acc: 0.51
ScaleFixpoint: FIX
Drone Status: Landed (96 Battery)
- Control Source:** Radio buttons for 'Autopilot' (selected) and 'None'. A checked checkbox for 'Ping Drone (every 1s)'.

The right window, titled 'PTAM Drone Map View', shows a 3D perspective view of a grid floor. A red line indicates the drone's current position and heading. A green wireframe box represents the drone's field of view or a target area. At the bottom of the window, the drone's pose is displayed: `Drone Pose: xyz=(-0.01, 0.18, 0.34), rpy=(-0.43, -0.10, -2.35)`.

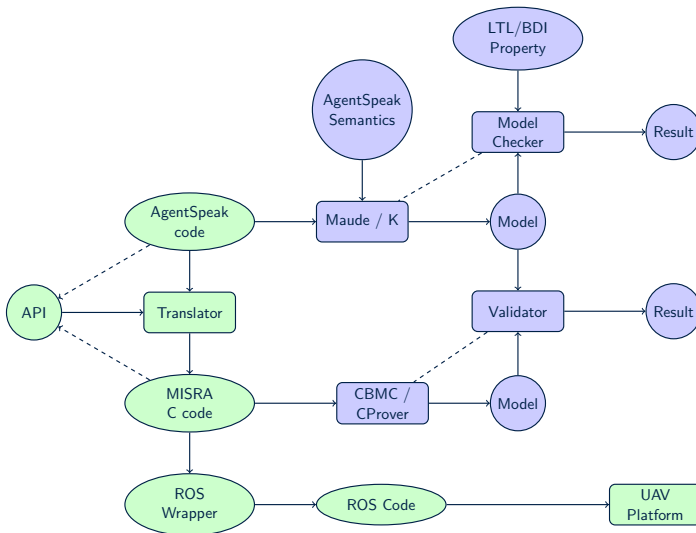
Test Flight



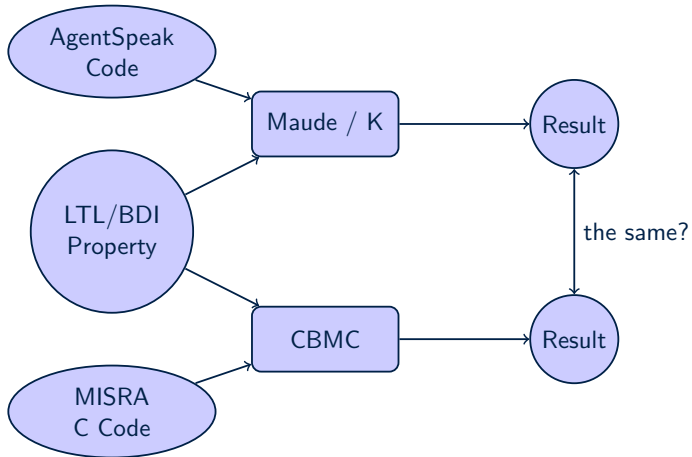
DEPARTMENT OF
**COMPUTER
SCIENCE**



Future Work



A Simple Translation Validation



Conclusions



- translation from agent-oriented programming language to low-level language + translation validation = traceability
- what is the sweet spot for agent-oriented programming languages: expressivity vs. translatability?
- translation validation?