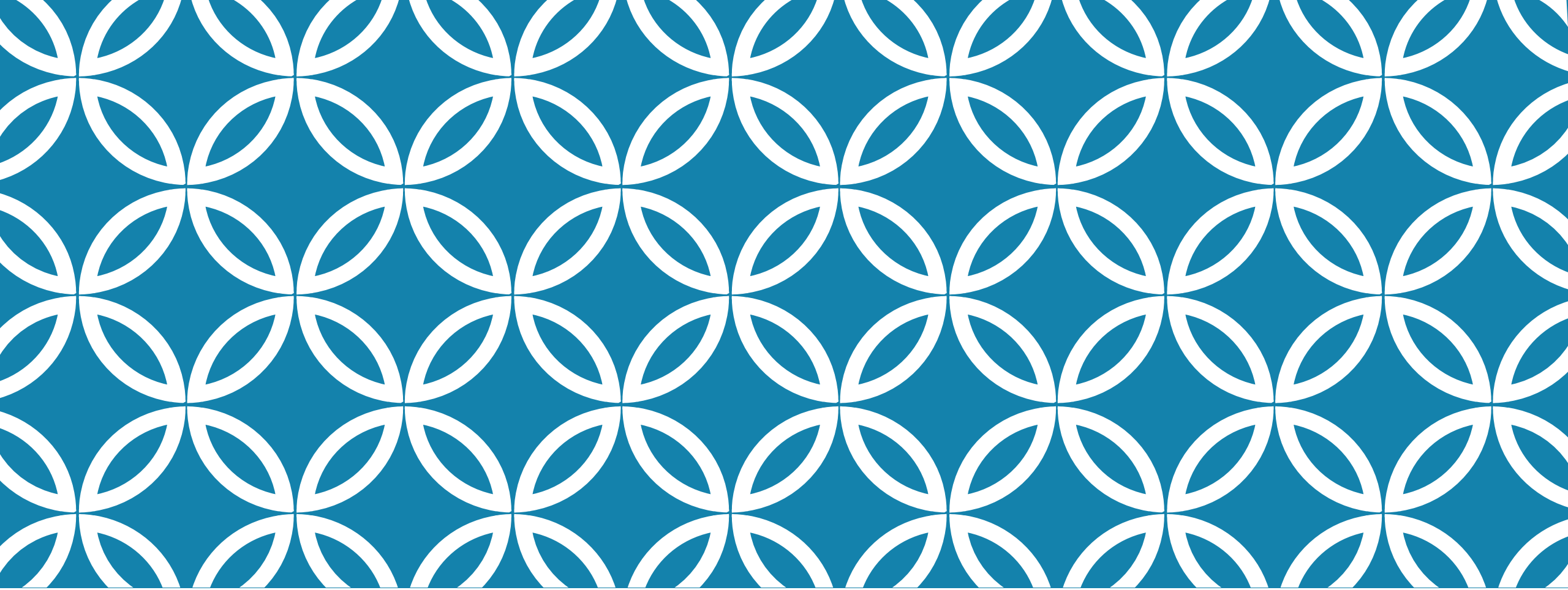


HIPPO | A Formal-Model Execution Engine to Control and Verify Critical Real-Time Systems

SILVANO DAL ZILIO, **PIERRE-EMMANUEL HLADIK**, FÉLIX INGRAND
ET AUSSI ANTHONY MALLET, REYYAN TEKKIN, MOHAMMED FOUGHALI, ROBIN VINCELLE…

ÉCOLE D'ÉTÉ TEMPS RÉEL - POITIERS - FUTUROSCOPE –
20 AU 24 SEPTEMBRE 2021

LS2N

LAAS CNRS

WHAT WILL WE FIND IN THIS PRESENTATION?

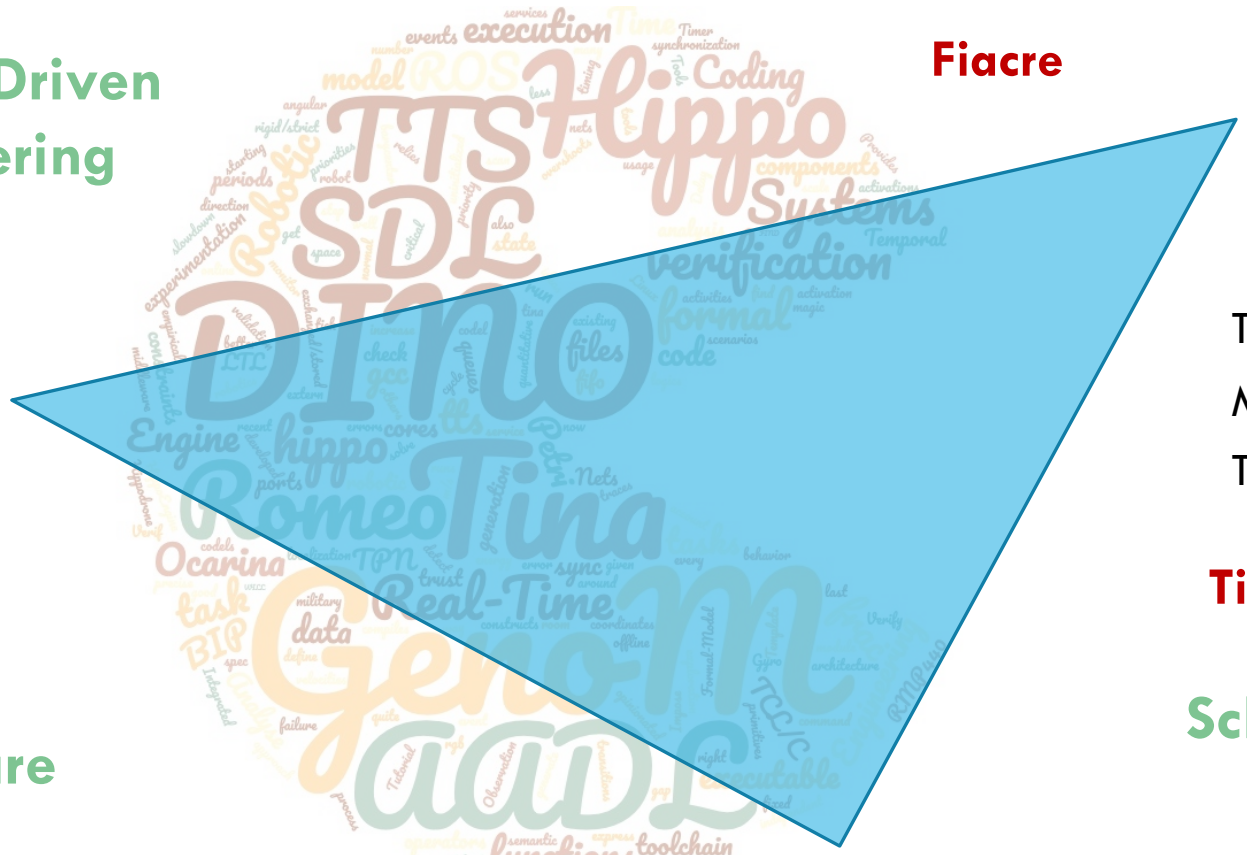
Model-Driven Engineering

GenoM

Fiacre

Formal Methods

Robotic Software



Time Petri Nets

Model-Checking

Temporal Logic (LTL)

Tina

ROS

Pocolibs

Middleware

Schedulability

Real-Time Systems

Dinosaur training

A SHORT STORY

heterogeneous model

weakly formal

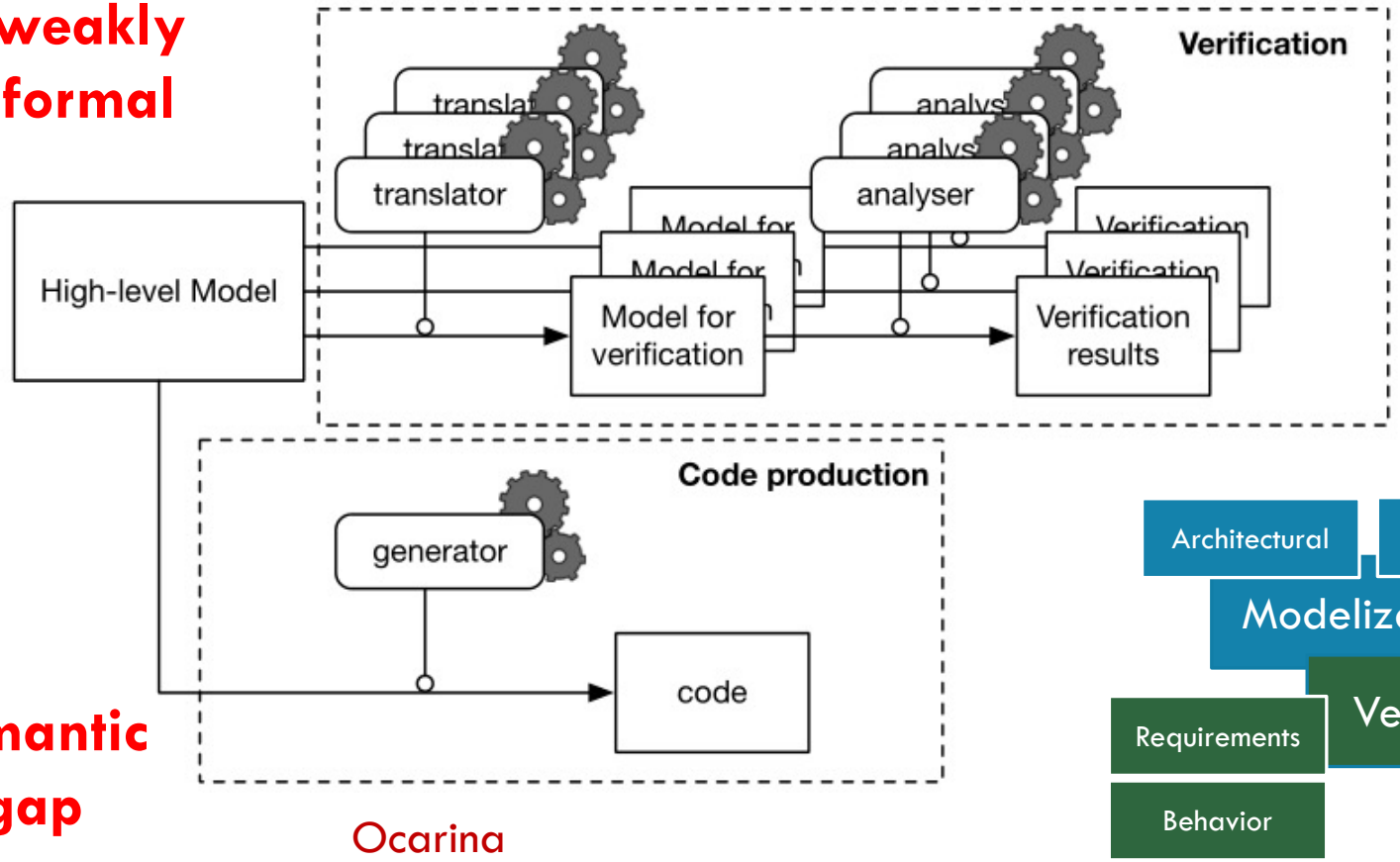
BPMN

SDL

GenoM

SysML

AADL



semantic gap

Ocarina

CADP

SPIN

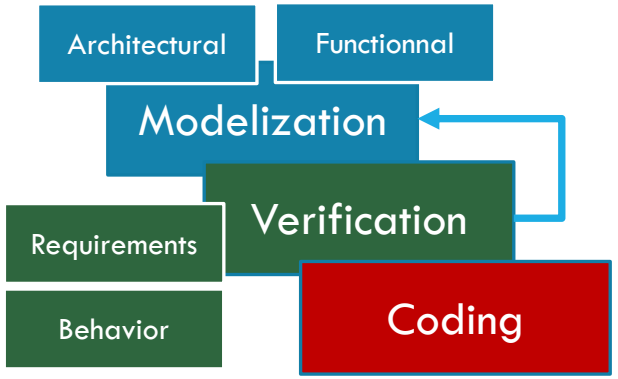
NuSMV

Tina

Cheddar

Uppaal

Romeo



BIP

Giotto

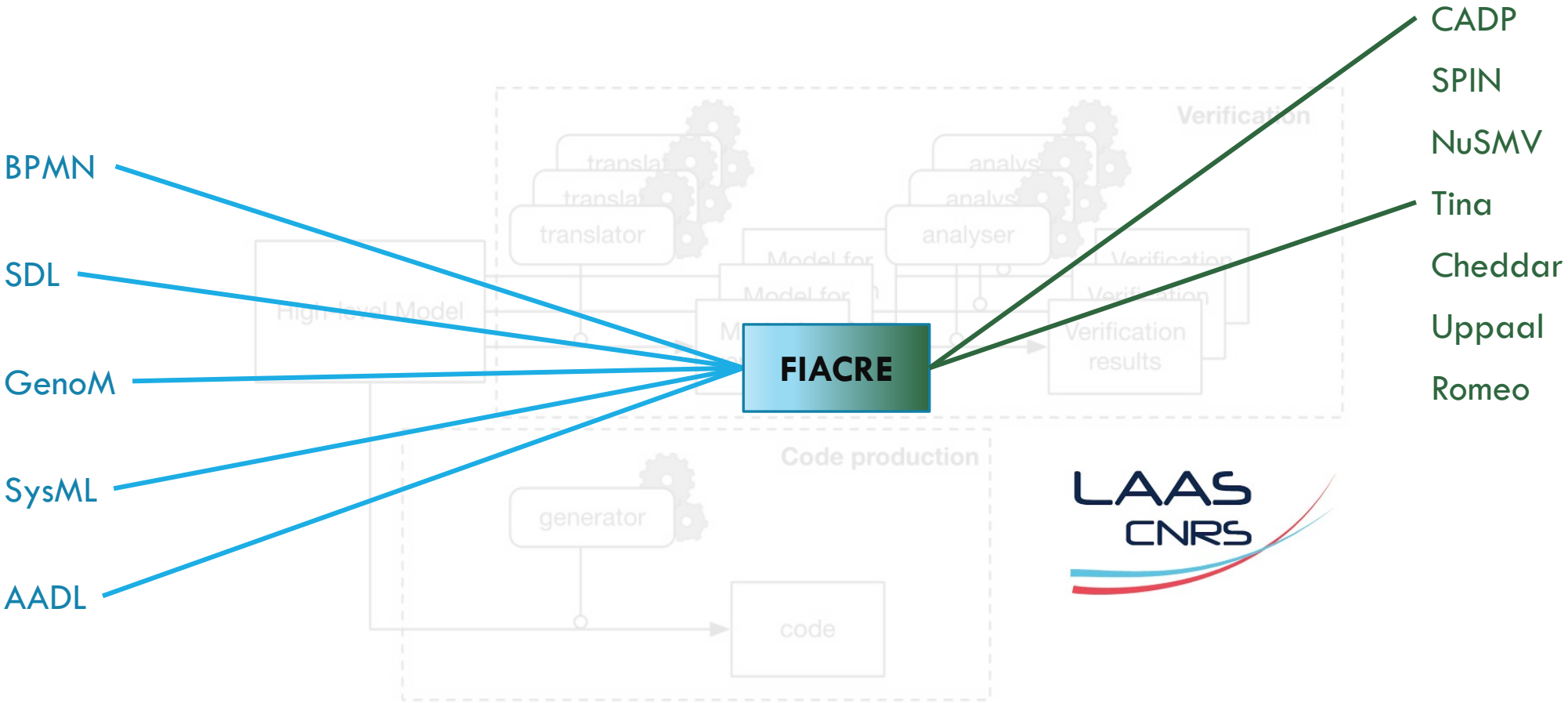
CPAL

SCADE

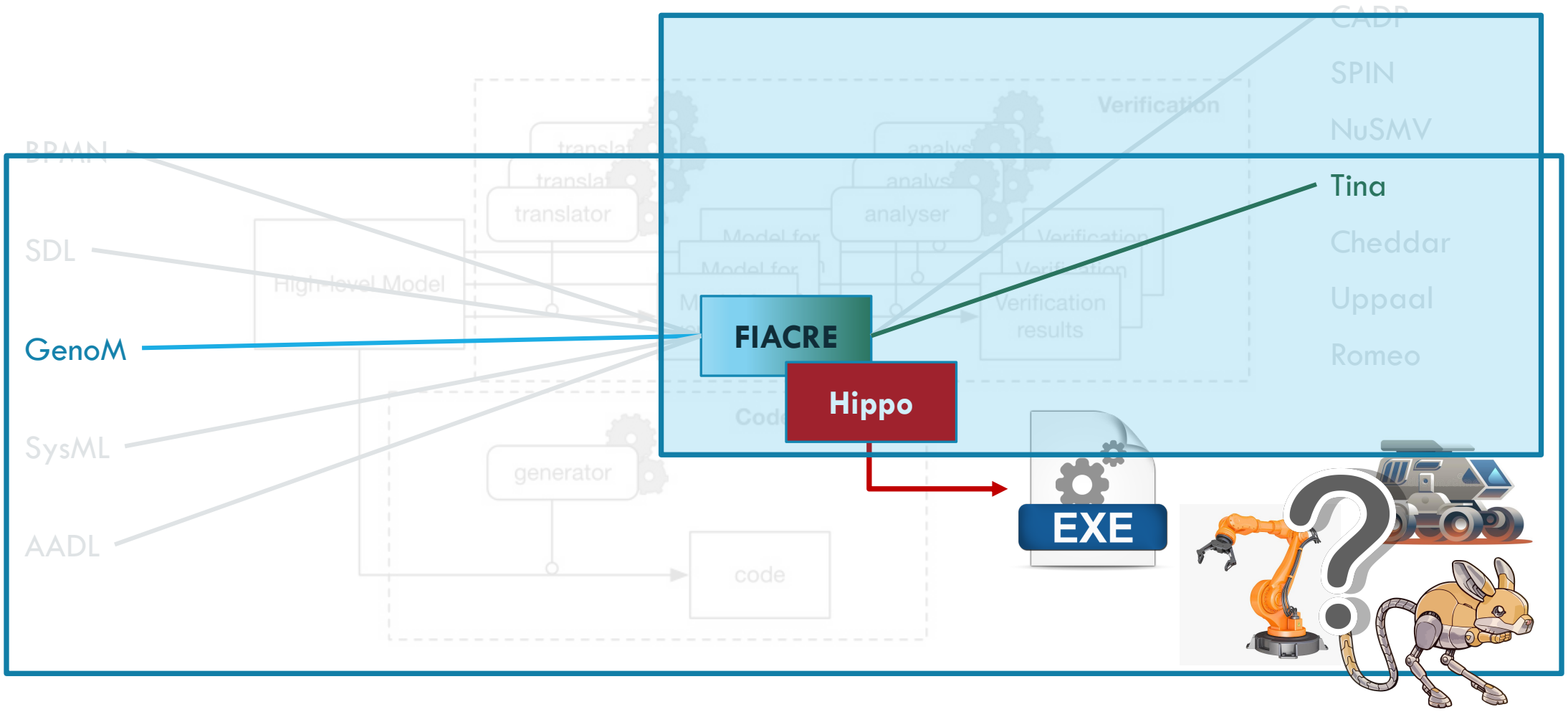
Simulink

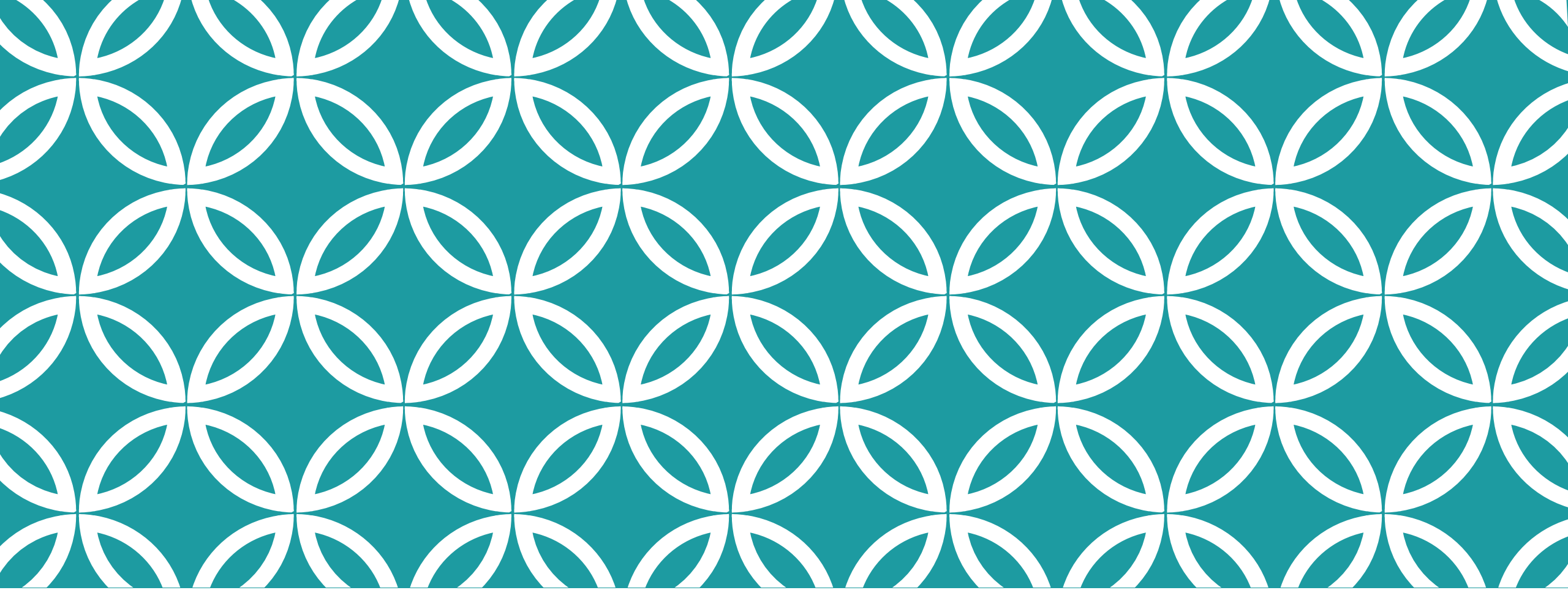
Prelude

A SHORT STORY



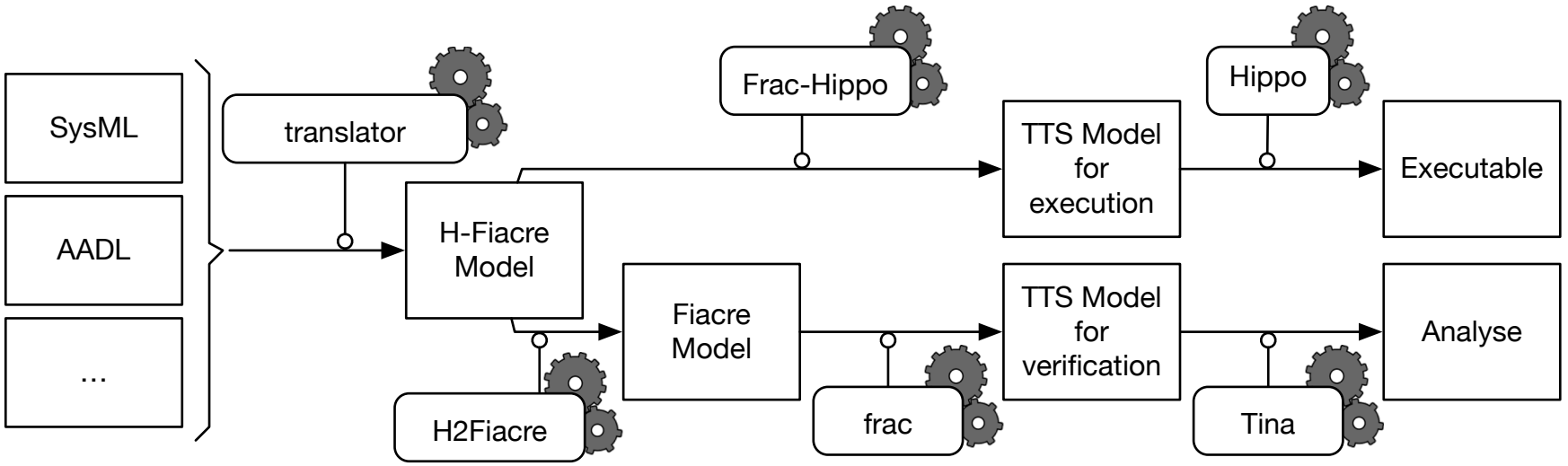
OUTLINE


BPMN
SDL
GenoM
SysML
AADL
FIACRE
Hippo
EXE
CADP
SPIN
NuSMV
Tina
Cheddar
Uppaal
Romeo
translator
analyser
generator
code
Verification
Verification results
High-level Model



HIPPO AN ENGINE FOR FIACRE

OVERVIEW OF HIPPO TOOLCHAIN



TINA

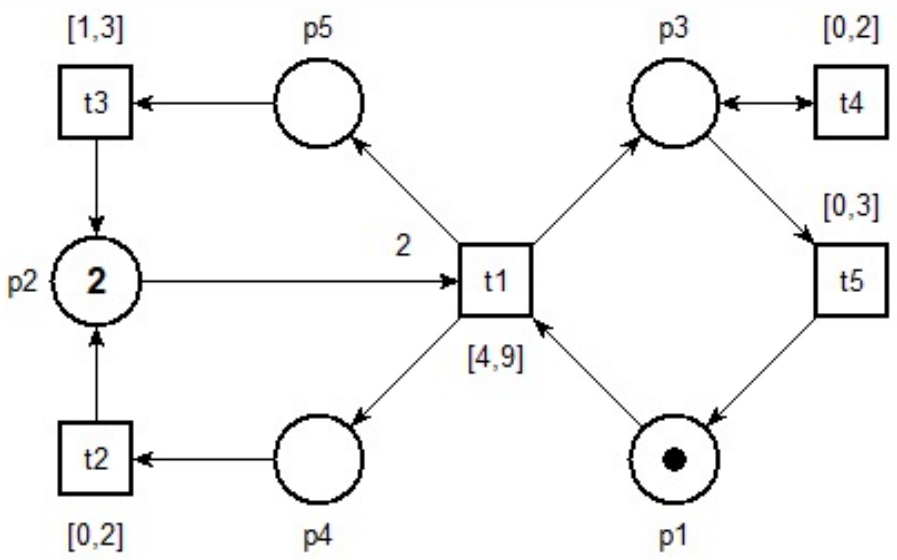
http://projects.laas.fr/tina/home.php

Modelling based on a Time extension of Petri nets (TPN), with priorities, etc.

Historically: checking protocols ; hardware (now SoC) ; architecture exploration ; etc.

Toolbox with multiple abstraction and verification methods

- Reachability analysis
- Simulation
- Model-checking using different temporal logics




```
Selt version 3.6.0 -- 07/07/20 -- LAAS/CNRS
ktz loaded, 12 states, 29 transitions
0.000s

- [] (t1 => <> t5);
FALSE
  state 0: p1 p2*2
  -t1 ... (preserving - t5 /\ t1)->
  state 12: p3 p4 p5
  -t4 ... (preserving - t5)->
* [accepting] state 15: p2*2 p3
  -t4 ... (preserving - t5)->
  state 15: p2*2 p3
0.000s
```

FIACRE ET H-FIACRE

http://projects.laas.fr/fiacre/home.php

Think of Fiacre as TPN with datatypes (arrays, structs, fifo queues, …) and components ⇒ it generates TTS

Hippo adds operators for "runtime" tasks and events ⇒ generates executable code

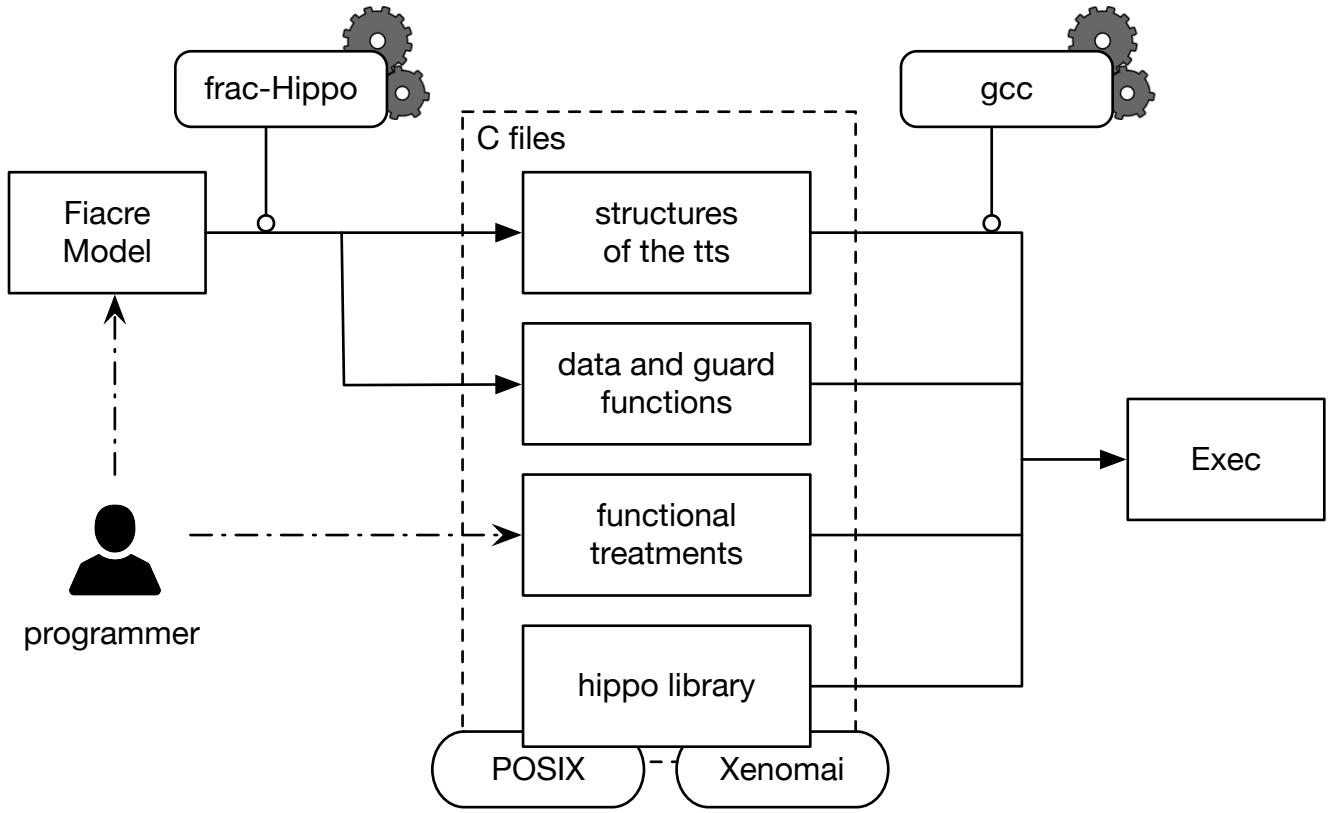
```
type tyEvt is record time : int, id : nat end
type tyDblEvt is array 2 of tyEvt

event e : tyEvt is c_click
task t (tyDblEvt) : nat is c_print

process double_event is
  states wait_first, wait_second, start_print, wait_print
  var tmp : tyDblEvt := [{time=0,id=0}, {time=0,id=0}], ret : nat := 0
  from wait_first
    e?tmp[0]; /* wait first event, assign value to tmp[0] */
    to wait_second
  from wait_second
    select
      wait [200,200];
      to wait_first
    []e?tmp[1];  /* wait second event, assign value to tmp[1] */
      to start_print
    end
  from start_print
    start t (tmp); /* start task t */
    to wait_print
  from wait_print
    sync t ret; /* wait end of task t */
    tmp := [{time=0,id=0}, {time=0,id=0}];
    to wait_first
```

H-FIACRE BEHAVIOR IN FIACRE

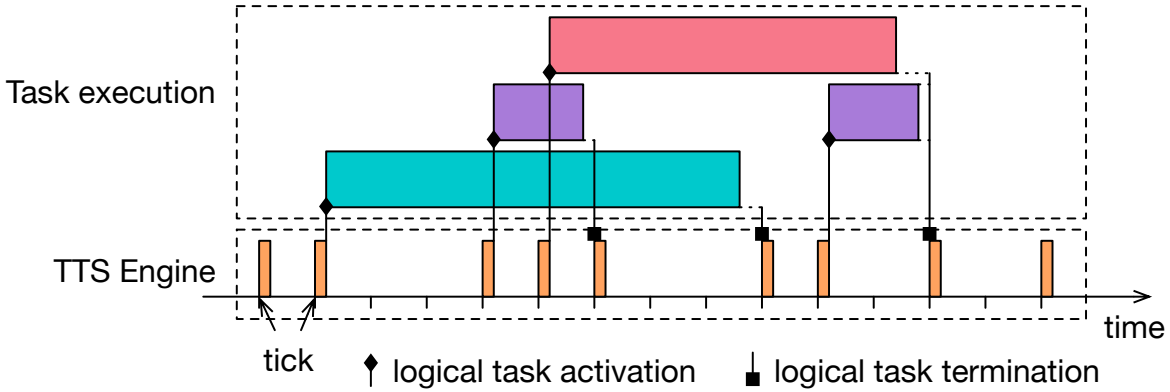
```
type tyEvt is record time : int, id : nat end
type tyDblEvt is array 2 of tyEvt

event e : tyEvt is c_click
task t (tyDblEvt) : nat is c_print

process double_event is
  states wait_first, wait_second, start_print, wait_print
  var tmp : tyDblEvt := [{time=0,id=0}, {time=0,id=0}], ret : nat
  from wait_first
    e?tmp[0]; /* wait first event, assign value to tmp[0] */
    to wait_second
  from wait_second
    select
      wait [200,200];
      to wait_first
    []e?tmp[1];  /* wait second event, assign value to tmp[1] */
      to start_print
    end
  from start_print
    start t (tmp); /* start task t */
    to wait_print
  from wait_print
    sync t ret; /* wait end of task t */
    tmp := [{time=0,id=0}, {time=0,id=0}];
    to wait_first
```

```
process p_task_t [
    t_SyncGlobal : none,
    t_activate_1, t_activate_2, ..., t_activate_n   : tyIn,
    t_terminated_1, t_activate_2, ... t_activate_n  : tyOut
  ] is
  states waiting, running, synchronizing, terminating
  var param : tyIn, ret : tyOut
  from waiting
    select
      t_activate_1?param; to running
    []t_activate_2?param; to running
    ...
    []t_activate_n?param; to running
    end
  from running
    ret := c_foo(param); /* The computational function is called */
    wait[$bcrt, $wcrt];  /* simulate the WCRT */
    to synchronizing
  from synchronizing
    t_SyncGlobal; /* Synchronization with the global tick */
    to terminating
  from terminating
    select /* The return value are written */
      t_terminated_1 ! ret; to waiting
    []t_terminated_2 ! ret; to waiting
    ...
    []t_terminated_n ! ret; to waiting
    end
```

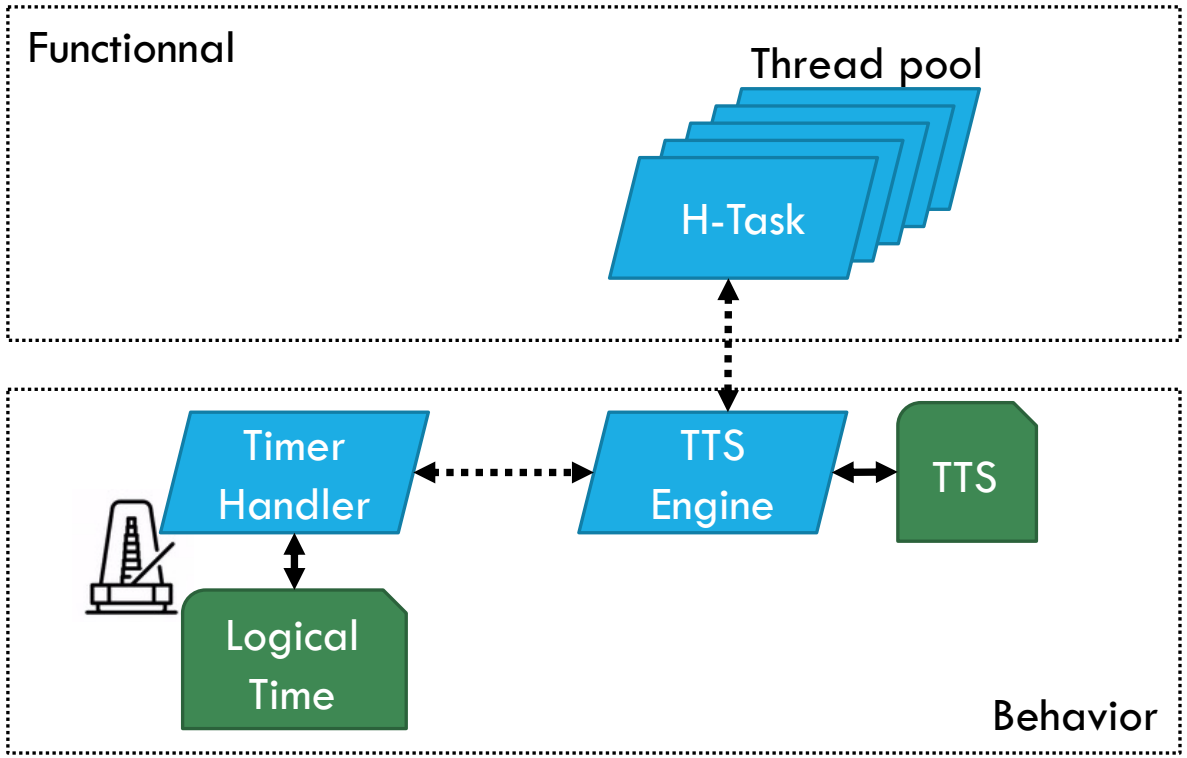
COMPILATION TOOLCHAIN



RUNTIME ENGINE

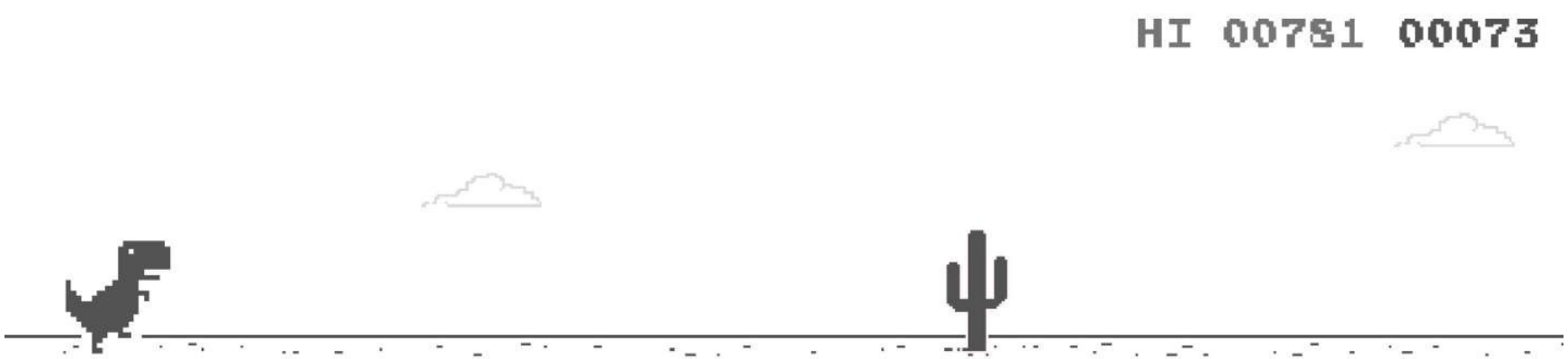


The runtime is implemented on Linux (better with PREEMPT_RT) and uses POSIX services with SCHED_FIFO (~ fixed priority scheduler)

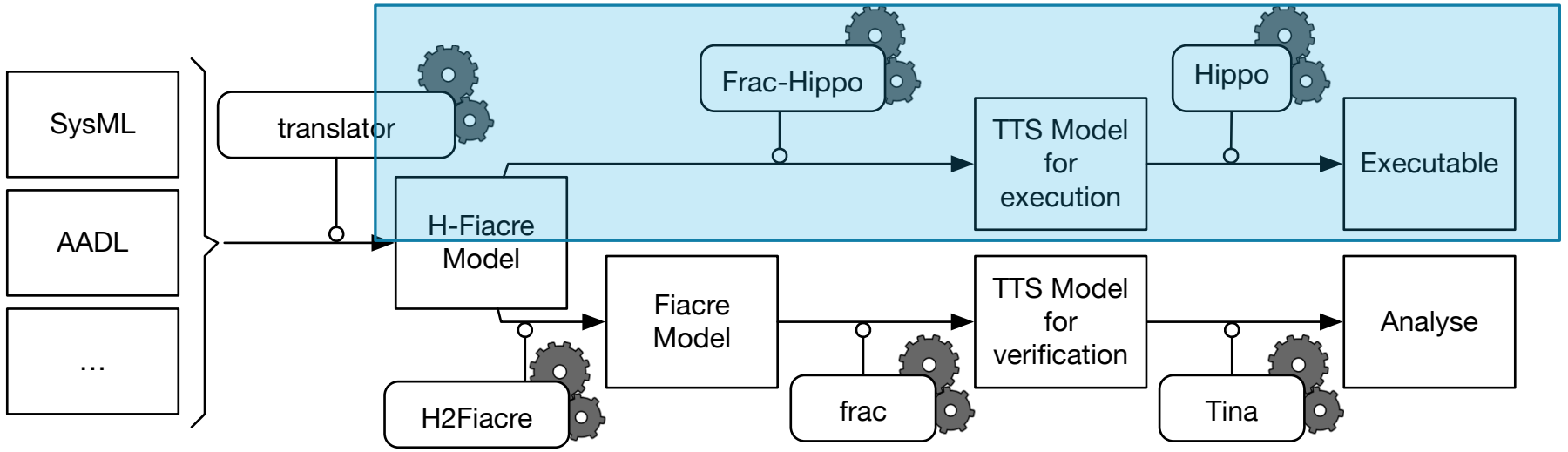


DINO TUTORIAL

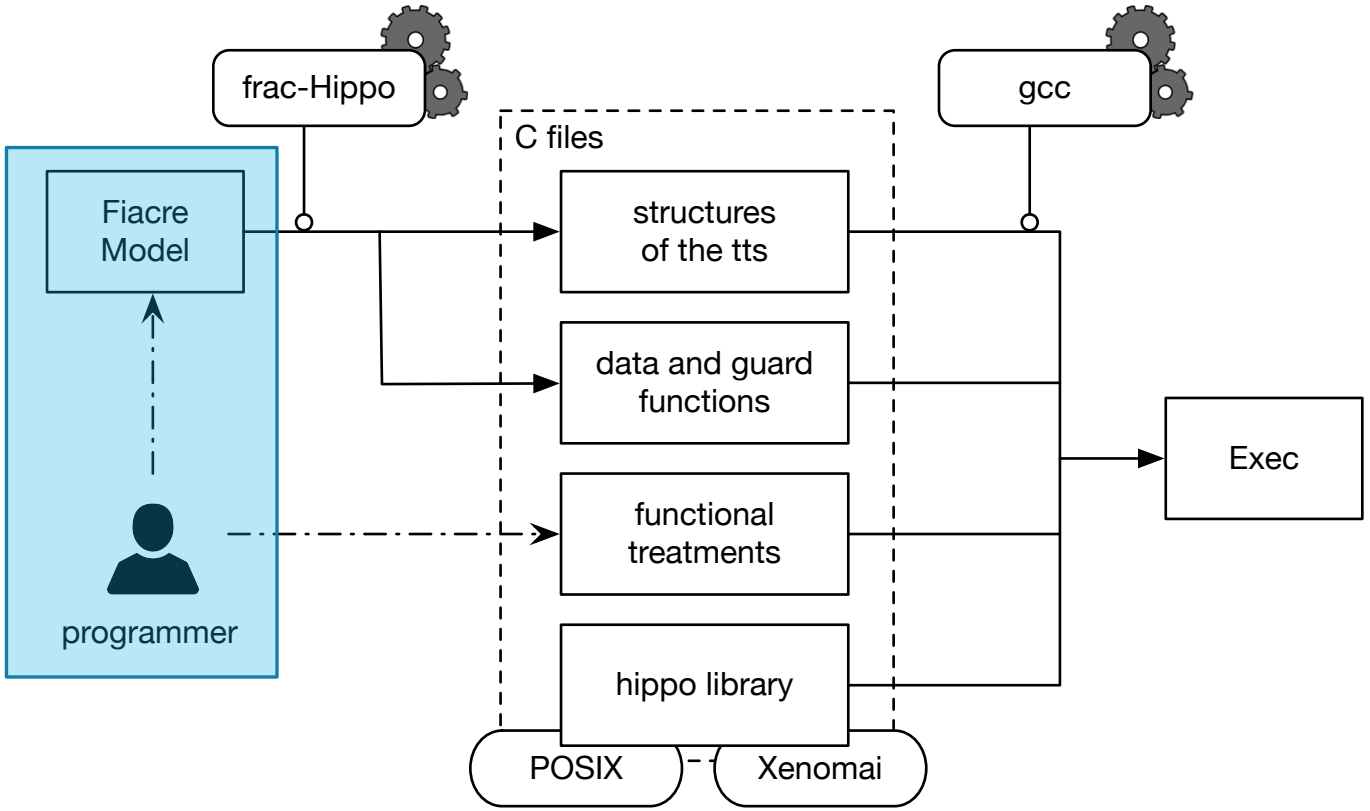
https://gitlab.laas.fr/pehladik/hippo/-/tree/tutorial/software/engine/tutorial



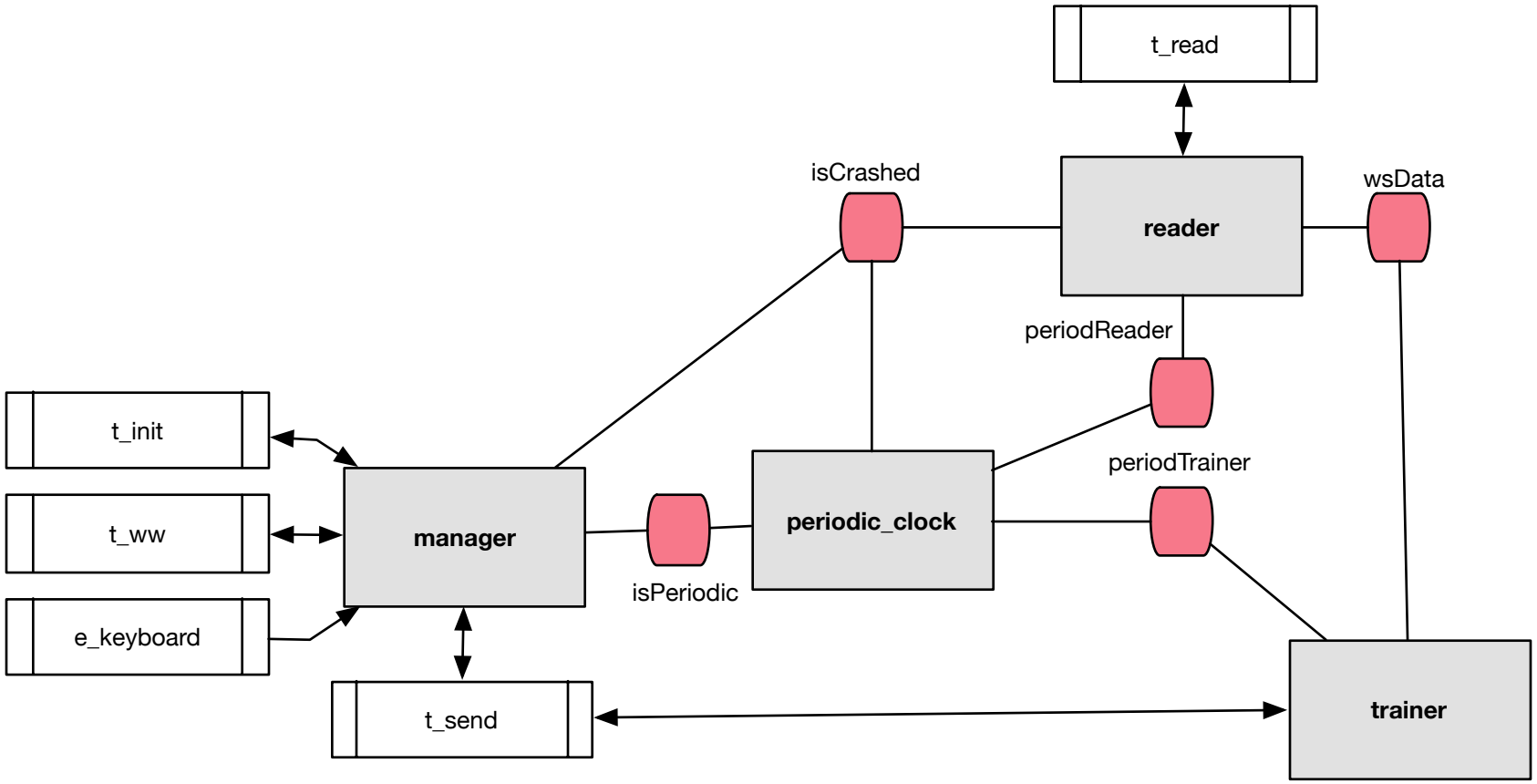
DINO TUTORIAL: GENERATE AN EXECUTABLE



DINO TUTORIAL: WRITE H-FIACRE MODEL

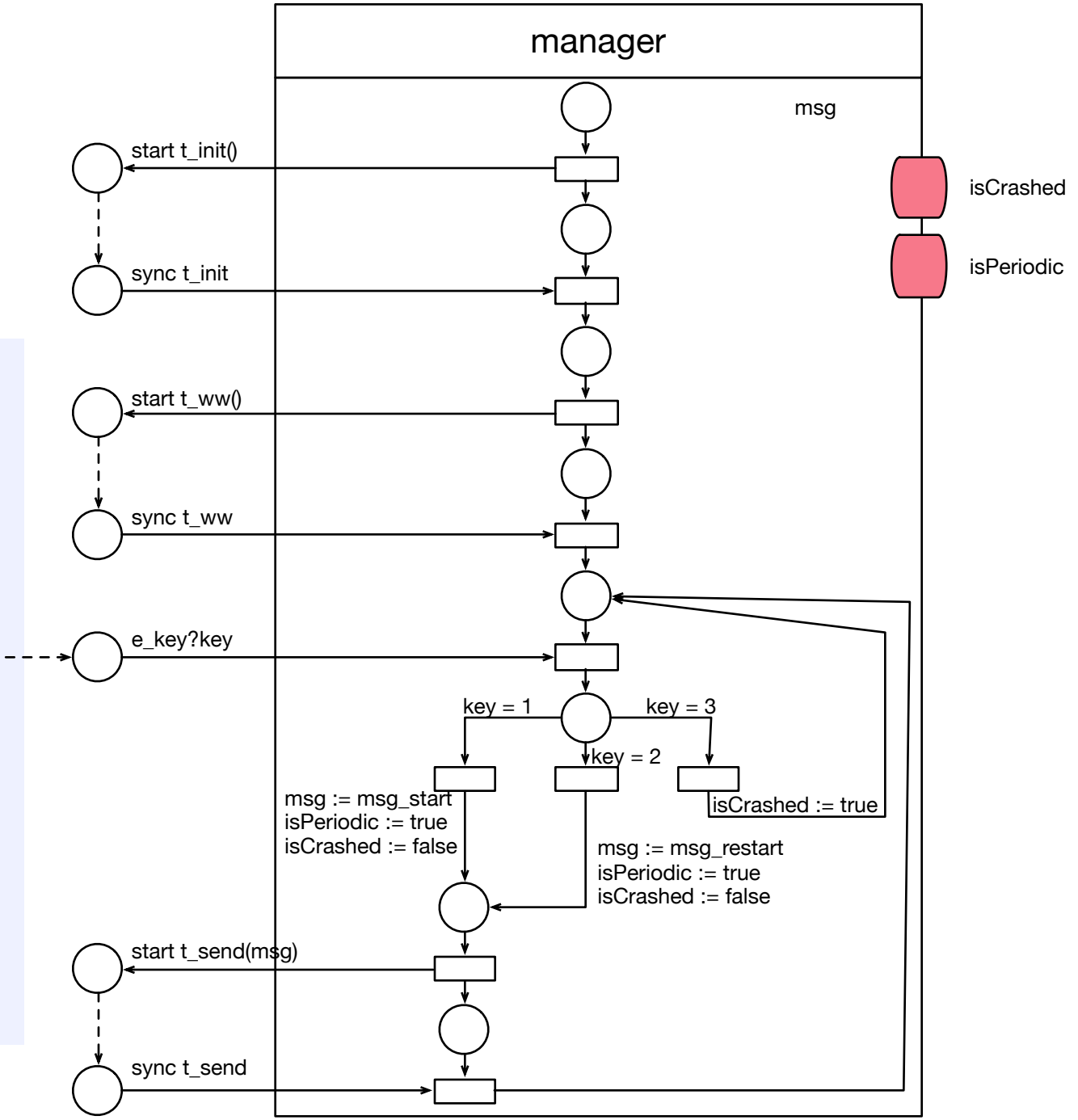


DINO TUTORIAL: MODEL



DINO TUTORIAL: MODEL

```
process manager (&isPeriodic: bool, &isCrashed : bool) is
    states s1, s2, s3, s4, s5, s6, s7, s8
    var     key     : 1..3       := 3,
            t       : int        := 1,
            msg     : msg_id     := msg_restart

    from s1
        start t_init();
        to s2
    from s2
        sync t_init t;
        to s3
    from s3
        start t_ww();
        to s4
    from s4
        sync t_ww t;
        to s5
    from s5
        e_keyboard?key;
        to s6
    from s6
        case key of
              1 -> msg := msg_start; isPeriodic := true; isCrashed := false; to s7
            | 2 -> msg := msg_restart; isPeriodic := true; isCrashed := false; to s7
            | 3 -> isCrashed := true; to s5
        end
    from s7
        start t_send(msg);
        to s8
    from s8
        sync t_send t;
        to s5
```

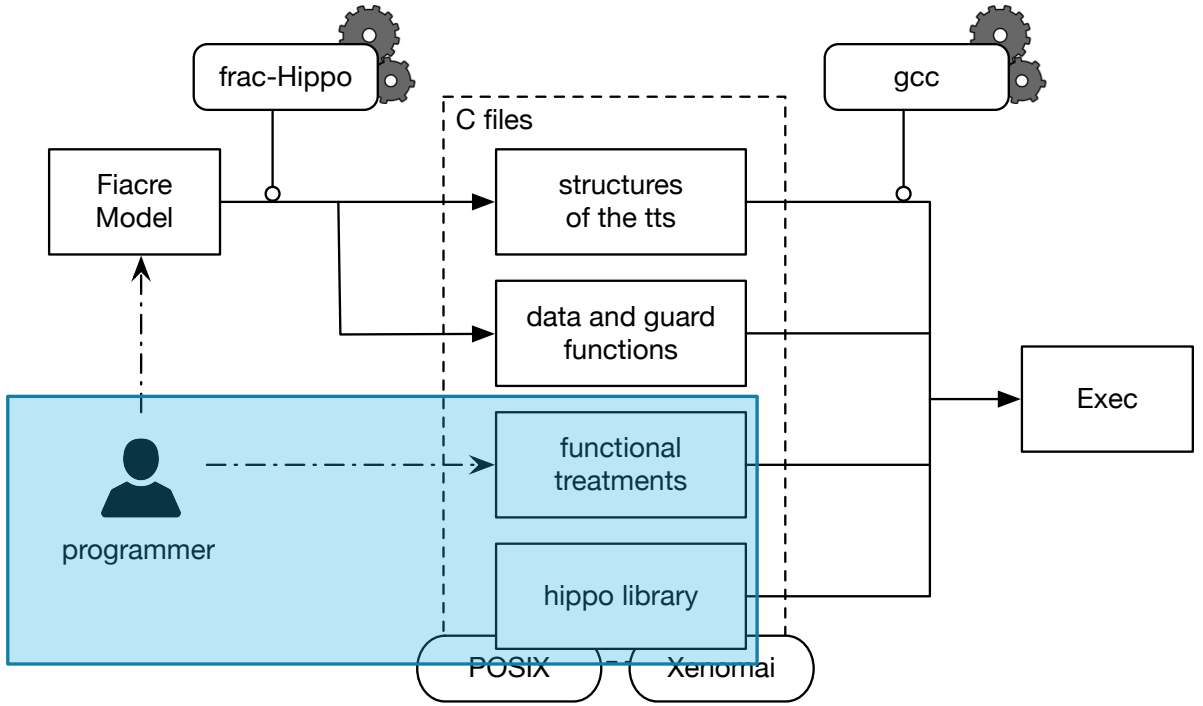


DINO TUTORIAL: FUNCTION

```c
int f_send(msg_id_c msg_id){
    mg_ws_send(mgr.conns, msg_map[msg_id], MSG_SIZE, WEBSOCKET_OP_TEXT);
    pthread_mutex_lock(&mutex);
    valueIsUsed = 1;
    pthread_mutex_unlock(&mutex);
    return 0;
}
```

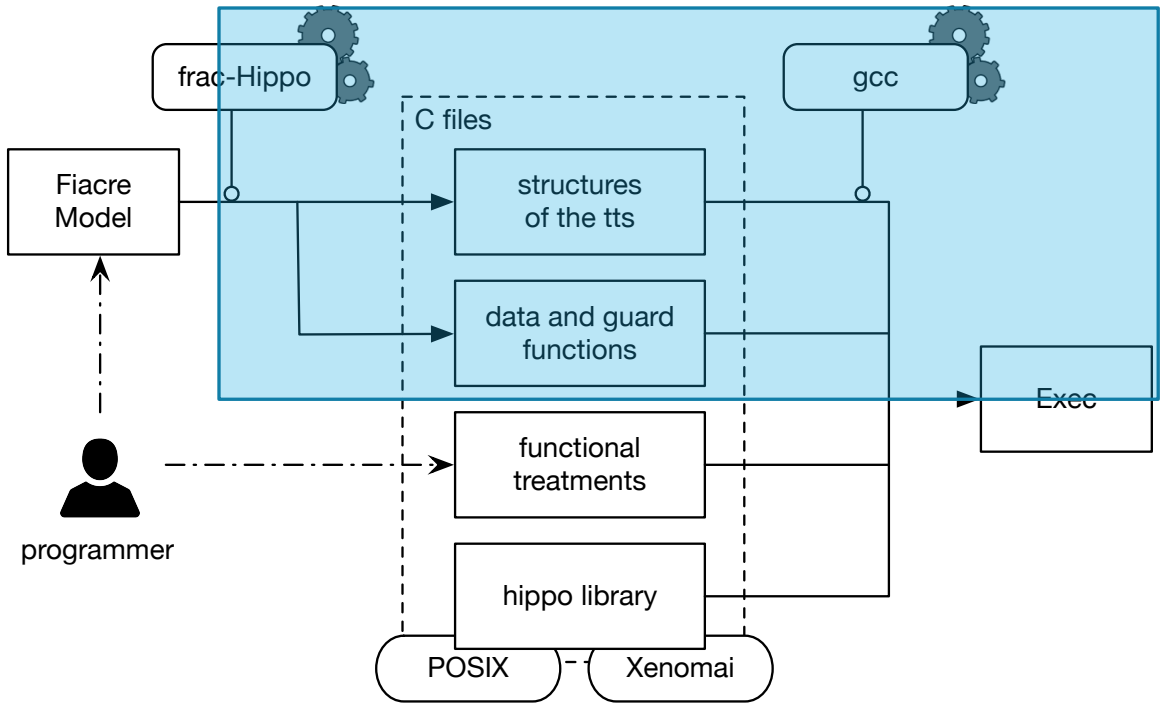
```c
#ifndef _HIPPO_FUNCTIONS_
#define _HIPPO_FUNCTIONS_

#include "mongoose.h"
#include "hippo_app_functions.h"

int f_init_mg();
int f_wait_websocket();
char f_keyboard();
int f_send(msg_id_c);
wsData_c f_read();

#endif
```

```c
#include <stdlib.h>
#include "hippo.h"

int main(int argc, char** argv) {
    initial();
    engine_run();
    return (EXIT_SUCCESS);
}
```

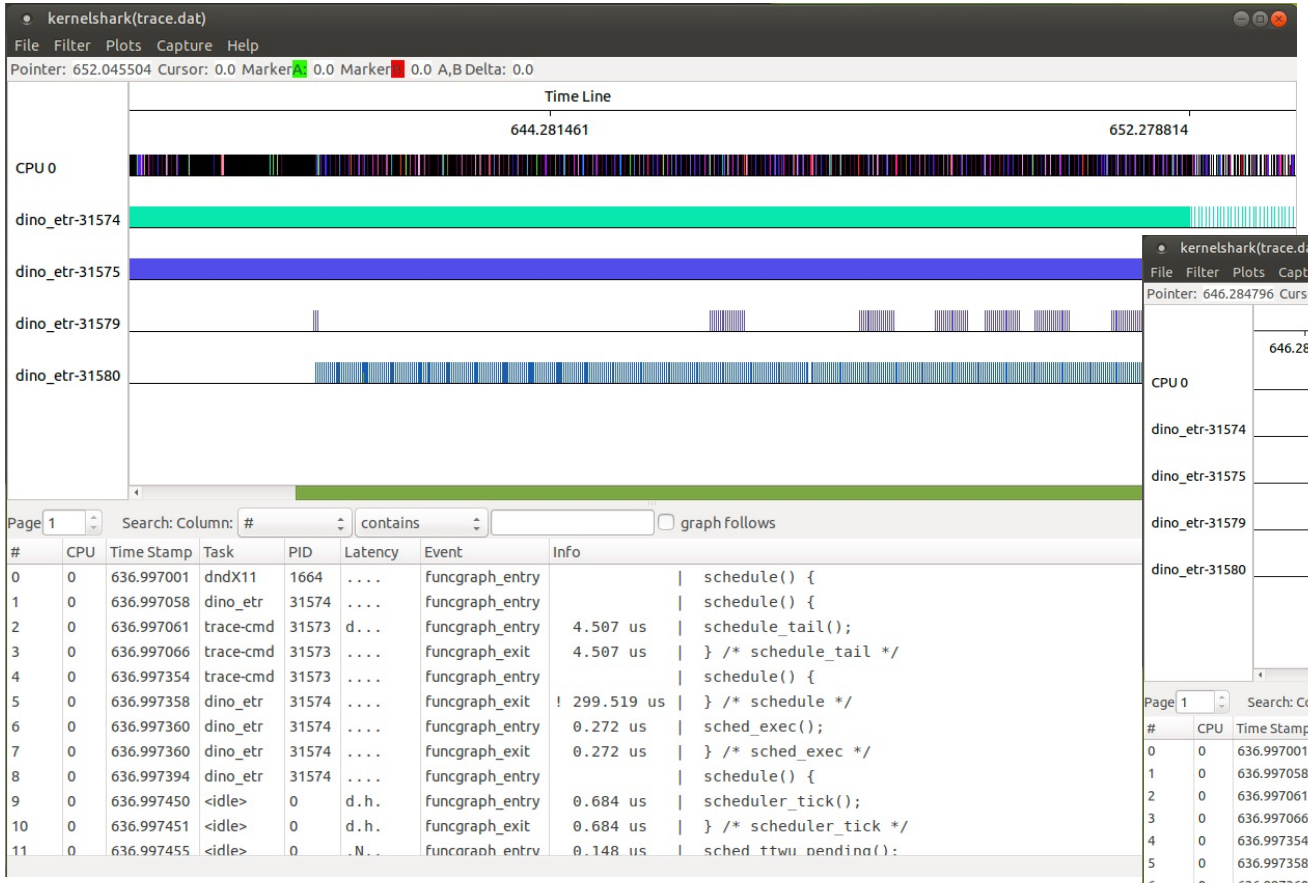


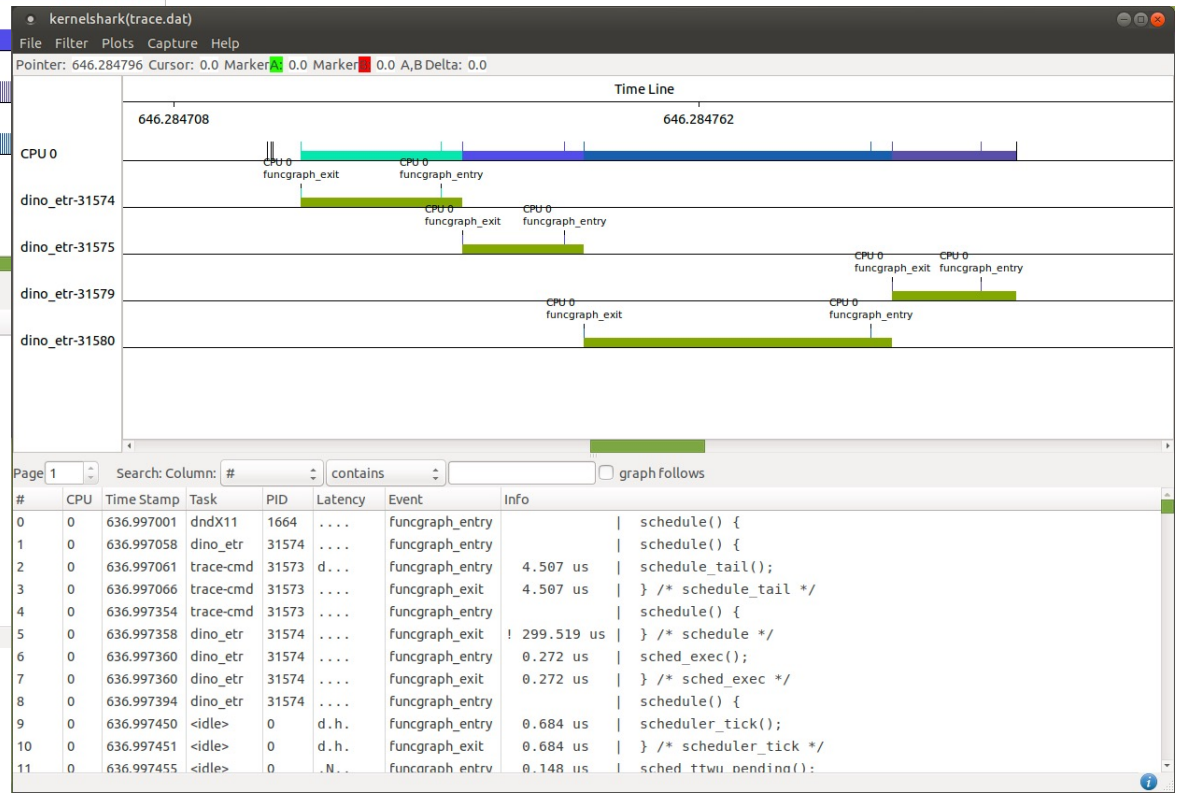
DINO TUTORIAL: CODE GENERATION

```
#!/bin/bash

frac dino_etr.fcr dino_etr.hippo
python3 ../config.py dino_etr -t=posix -u=5ms -h=../lib/hippo/ -a=./lib_etr
make clean
make
```

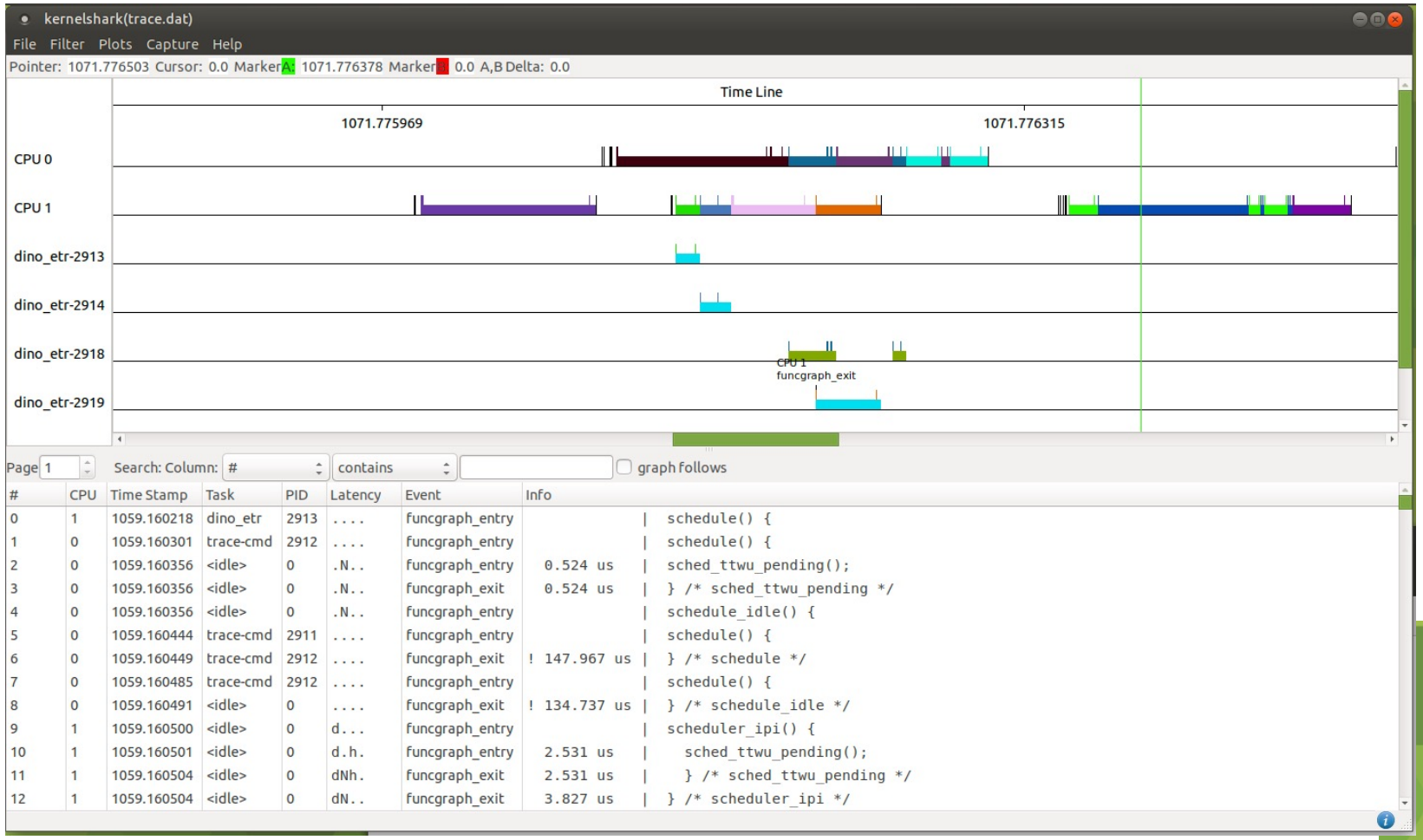


DINO TUTORIAL: EXECUTION AND PERFORMANCE





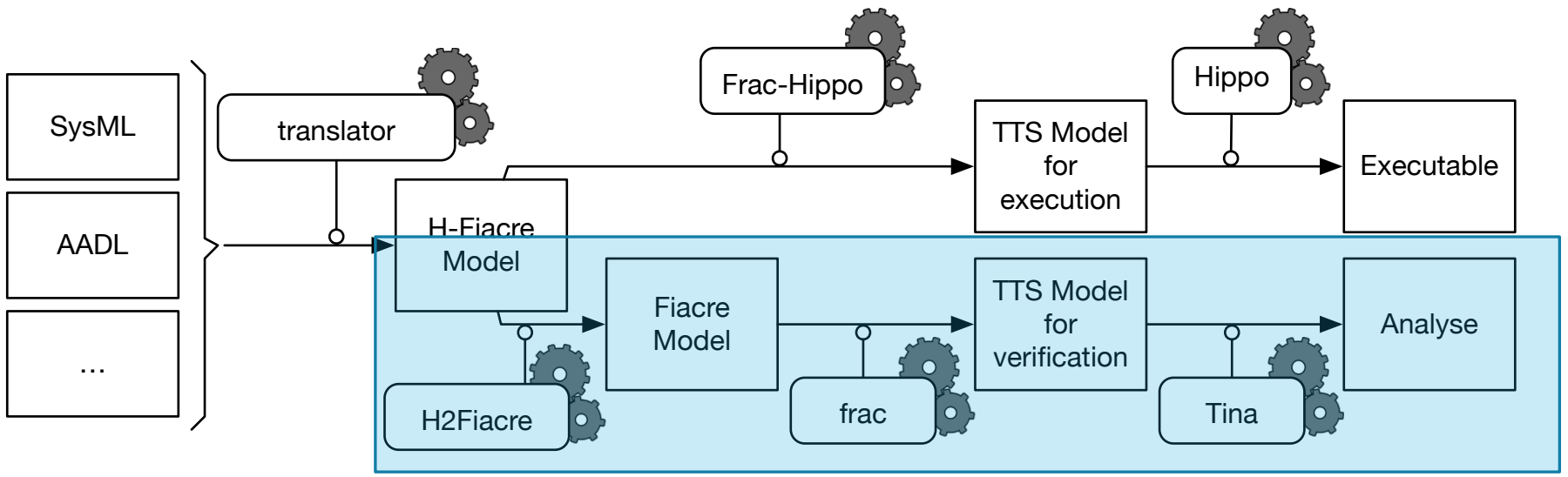
DINO TUTORIAL: DUAL CORE PERFORMANCE



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	1	1059.160218	dino_etr	2913	....	funcgraph_entry	| schedule() {
1	0	1059.160301	trace-cmd	2912	....	funcgraph_entry	| schedule() {
2	0	1059.160356	<idle>	0	.N..	funcgraph_entry	0.524 us | sched_ttwu_pending();
3	0	1059.160356	<idle>	0	.N..	funcgraph_exit	0.524 us | } /* sched_ttwu_pending */
4	0	1059.160356	<idle>	0	.N..	funcgraph_entry	| schedule_idle() {
5	0	1059.160444	trace-cmd	2911	....	funcgraph_entry	| schedule() {
6	0	1059.160449	trace-cmd	2912	....	funcgraph_exit	! 147.967 us | } /* schedule */
7	0	1059.160485	trace-cmd	2912	....	funcgraph_entry	| schedule() {
8	0	1059.160491	<idle>	0	....	funcgraph_exit	! 134.737 us | } /* schedule_idle */
9	1	1059.160500	<idle>	0	d...	funcgraph_entry	| scheduler_ipi() {
10	1	1059.160501	<idle>	0	d.h.	funcgraph_entry	2.531 us | sched_ttwu_pending();
11	1	1059.160504	<idle>	0	dNh.	funcgraph_exit	2.531 us | } /* sched_ttwu_pending */
12	1	1059.160504	<idle>	0	dN..	funcgraph_exit	3.827 us | } /* scheduler_ipi */

```
:8000
DINO >> waiting websocket
DINO >> websocket opened
1
DINO >> enter action (1: start, 2: restart, 3:stop)
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
Lost in the sea
```

DINO TUTORIAL: VERIFICATION



DINO TUTORIAL: VERIFICATION

ADD TEMPORAL PARAMETERS

```
task t_read     ()      : wsData    is f_read
task t_send     (msg_id): int       is f_send
task t_init     ()      : int       is f_init_mg
task t_ww       ()      : int       is f_wait_websocket
event e_keyboard        : 1..3      is f_keyboard
```

```
task t_read     ()      : wsData    is f_read /* @wcrt: 0.001 @bcrt: 0.00005,
                    @ret:$({xObstacle=1, yObstacle=5, speed=10, isCrashed=false})$ */
task t_send     (msg_id): int       is f_send /* @wcrt: 0.0002, @bcrt: 0.0001,
                    @ret:4 */
task t_init     ()      : int       is f_init_mg  /* @wcrt: 0.0002, @bcrt: 0.0001,
                    @ret:4 */
task t_ww       ()      : int       is f_wait_websocket  /* @wcrt: 0.0002 @bcrt: 0.0001,
                    @ret:4 */
event e_keyboard        : 1..3       is f_keyboard /* @wcrt: 0.1, @bcrt: 0.1,
                    @ret: 1 */
```

DINO TUTORIAL: VERIFICATION

GENERATE TASK BEHAVIOR MODEL

```
type tyEvt is record time : int, id : nat end
type tyDblEvt is array 2 of tyEvt

event e : tyEvt is c_click
task t (tyDblEvt) : nat is c_print

process double_event is
  states wait_first, wait_second, start_print, wait_print
  var tmp : tyDblEvt := [{time=0,id=0}, {time=0,id=0}], ret : nat
  from wait_first
    e?tmp[0]; /* wait first event, assign value to tmp[0] */
    to wait_second
  from wait_second
    select
      wait [200,200];
      to wait_first
    []e?tmp[1];  /* wait second event, assign value to tmp[1] */
      to start_print
    end
  from start_print
    start t (tmp); /* start task t */
    to wait_print
  from wait_print
    sync t ret; /* wait end of task t */
    tmp := [{time=0,id=0}, {time=0,id=0}];
    to wait_first
```

```
process p_task_t [
    t_SyncGlobal : none,
    t_activate_1, t_activate_2, ..., t_activate_n   : tyIn,
    t_terminated_1, t_activate_2, ... t_activate_n  : tyOut
  ] is
  states waiting, running, synchronizing, terminating
  var param : tyIn, ret : tyOut
  from waiting
    select
      t_activate_1?param; to running
    [] t_activate_2?param; to running
    ...
    [] t_activate_n?param; to running
    end
  from running
    ret := c_foo(param); /* The computational function is called */
    wait[$bcrt, $wcrt];  /* simulate the WCRT */
    to synchronizing
  from synchronizing
    t_SyncGlobal; /* Synchronization with the global tick */
    to terminating
  from terminating
    select /* The return value are written */
      t_terminated_1 ! ret; to waiting
    [] t_terminated_2 ! ret; to waiting
    ...
    [] t_terminated_n ! ret; to waiting
    end
```

DINO TUTORIAL: VERIFICATION

ADD SCHEDULER

```
dino pehladik$ python3 ../rewrite.py dino_p6.fcr -T 0.005 -p 1 -o dino_rewritten6.fcr
```

```
process scheduler (&ready_list : fifo , &launch : start_tab , &unused_proc : nat) is
    states exec
    from exec
        on (not (empty ready_list)) and (unused_proc > 0) ;
        unused_proc := unused_proc - 1;
        launch [first ready_list]:= true;
        ready_list := dequeue ready_list ;
        wait [0 ,0]; to exec
```

```
process p_task_t_read [SyncGlobal_p_task_t_read : none,
    t_activate_t_read_reader_0 : none,
    t_terminated_t_read_reader_0 :  wsData]
    (id : 0..(numberOfTasks-1),
    &unused_proc : nat,
    &ready_list : fifo,
    &launch : start_tab) is
    states waiting, sched_activate, sched_resume, running, sched_terminated,
    synchronizing, terminating, gen

    var ret : wsData,
        cmpt : int := 0

    from waiting
        t_activate_t_read_reader_0; to sched_activate
    from sched_activate /* task activation : scheduler call */
        ready_list := enqueue(ready_list, id); wait [0,0]; to sched_resume
    from sched_resume
        on launch[id]; launch[id] := false; wait [0,0]; to running
    from running
        if (cmpt > 10) then
            cmpt := 0;
            ret := {xObstacle=900, yObstacle=5, speed=10, isCrashed=false}; wait[0,0];
            to gen
        else
            cmpt := cmpt + 1;
            ret := {xObstacle=10, yObstacle=5, speed=10, isCrashed=false}; wait[0,0];
            to gen
        end
    from gen
        wait[0.01, 0.2];  /* simulate the response time */
        to sched_terminated
    from sched_terminated /* task termination : scheduler call */
        unused_proc := unused_proc + 1; wait[0,0]; to synchronizing
    from synchronizing
        SyncGlobal_p_task_t_read; /* Synchronization with the global tick */
```

DINO TUTORIAL: VERIFICATION

CHEAT CODE: WORLD BEHAVIOR

```
process p_event_e_keyboard [SyncGlobal_p_event_e_keyboard : none,
        e_happened_e_keyboard_manager_0 : 1..3] is
    states waiting, synchronizing, posting, forever
    var ret : 1..3
    from waiting
        wait [20.0, 20.0]; /* timed pattern to produce event */
        ret := 1;
        to synchronizing
    from synchronizing
        SyncGlobal_p_event_e_keyboard;
        to posting
    from posting
        e_happened_e_keyboard_manager_0!ret; to forever
    from forever
        wait[1,1];
        to forever
```

DINO TUTORIAL: VERIFICATION

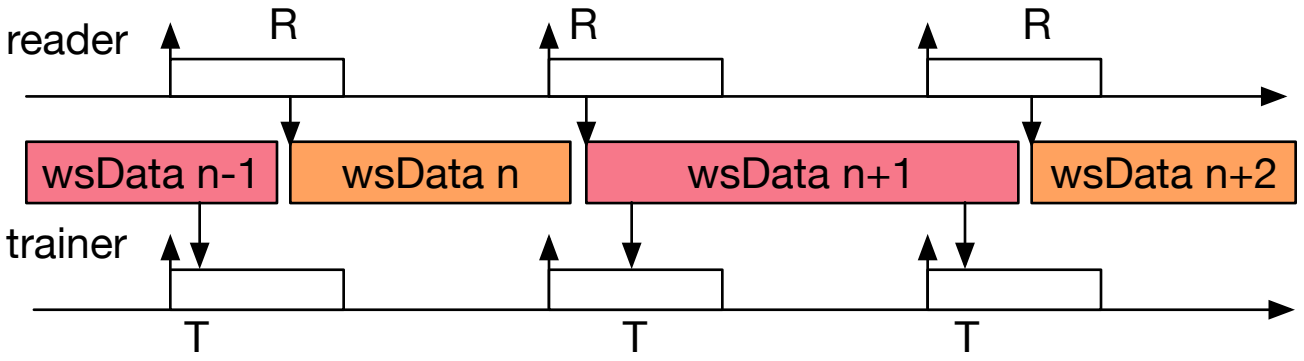
STATE SPACE CONSTRUCTION

```
frac dino_etr.fcr dino_etr.tts
python3 ../extract_transitions.py ./dino_etr.tts/dino_etr.net -w > res.txt
gcc -m64  -O2 -o dino_etr.tts/dino_etr.dylib -dynamiclib -I ../lib/ -I ../lib//avl dino_etr.tts/*.c
```

```
(base) po-hladik-pro13:dino pehladik$ tina dino_etr.tts dino_etr.ktz
# net process_p__task__t__read_1_p__task__t__ww_1_p__task__t__init_1_p__task_
eader_1_periodic__clock_1_manager_1, 56 places, 54 transitions, 140 arcs#
# bounded, not live, possibly reversible                                  #
# abstraction        count      props      psets       dead       live #
#      states         8147         56        844          0        936 #
# transitions        10180         54         54          3         29 #
```

DINO TUTORIAL: VERIFICATION

PROPERTY TO CHECK

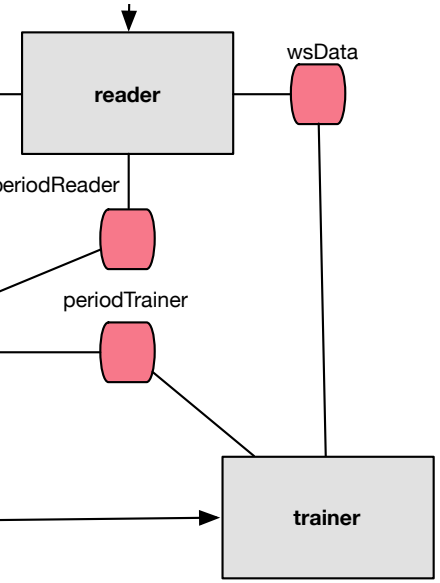


IF T then not T before R

[](T => (() (-T U R))

DINO TUTORIAL: VERIFICATION

WHERE'S WALLY?




```
process reader [t_activate_t_read_reader_0 :
    states s1, s2, s3

     from s1
        on periodReader;
        wait[0,0];
        periodReader := false;
        to s2
    from s2
        t_activate_t_read_reader_0;
        to s3
    from s3
        t_terminated_t_read_reader_0? d;
        isCrashed := d.isCrashed;
        to s1
```

R = p__task__t__read_1_t7_reader_1_t2

T = trainer_1_t2

```
process trainer [t_activate_t_send_trainer_0 : msg_id, t_terminated_
    states s1, s2, s3, s4, s5, s6
    var     duck        : bool  := false,
            ret         : int   := 0,
            msg         : msg_id := msg_jump

    from s1
        on periodTrainer;
        wait[0,0];
        periodTrainer := false;
        to s2
    from s2
        if(d.xObstacle < ((14*(50 + d.speed/10))/10) ) then
            to s3
        else
            to s4
        end
    from s3
        if((d.yObstacle < 76) and(not duck)) then
            duck := true;
            msg := msg_duck;
            to s5
        else
            msg := msg_jump;
            to s5
        end
    from s4
        if(duck) then
            duck := false;
            msg := msg_stop_ducking;
            to s5
```

DINO TUTORIAL: VERIFICATION

AND NOW !

```
const numberOfProcessors : nat is 1
```

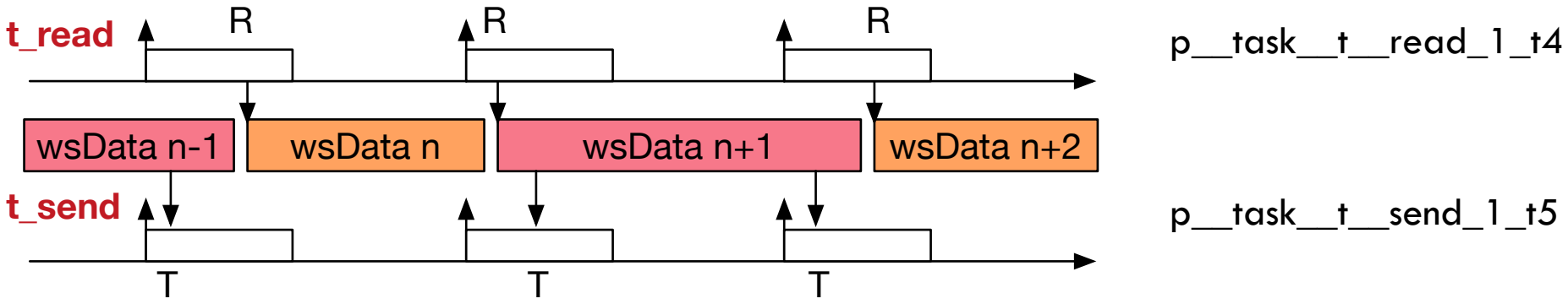
```
(base) po-hladik-pro13:dino pehladik$ selt -p dino_etr.ktz -f "[](trainer_1_t2 => ( () (- trainer_1_t2 ) U p__task__t__read_
1_t8_reader_1_t2 ))"
Selt version 3.5.0 -- 05/29/19 -- LAAS/CNRS
ktz loaded, 8147 states, 10180 transitions
0.011s
TRUE
0.021s
```

```
const numberOfProcessors : nat is 2
```

```
(base) po-hladik-pro13:dino pehladik$ selt -p dino_etr.ktz -f "[](trainer_1_t2 => ( () (- trainer_1_t2 ) U p__task__t__read_
1_t8_reader_1_t2 ))"
Selt version 3.5.0 -- 05/29/19 -- LAAS/CNRS
ktz loaded, 7757 states, 9894 transitions
0.009s
TRUE
0.016s
```

DINO TUTORIAL: VERIFICATION

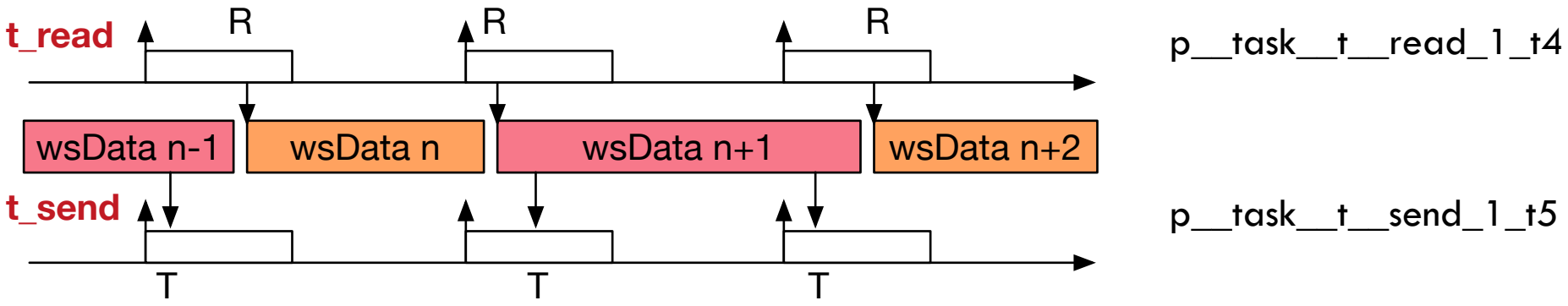
WHERE'S (REAL) WALLY?



… and only after starting!

DINO TUTORIAL: VERIFICATION

WHERE'S (REAL) WALLY?




```
const numberOfProcessors : nat is 1
```

```
(base) po-hladik-pro13:dino pehladik$ selt -p dino_etr.ktz -b -f "[](p__task__t__send_1_t7_manager_1_t9 => ([](p__task__t__s
end_1_t5 => ( () (- p__task__t__send_1_t5 ) U p__task__t__read_1_t4 ))))"
Selt version 3.5.0 -- 05/29/19 -- LAAS/CNRS
ktz loaded, 1177 states, 1468 transitions
0.003s
TRUE
0.005s
```

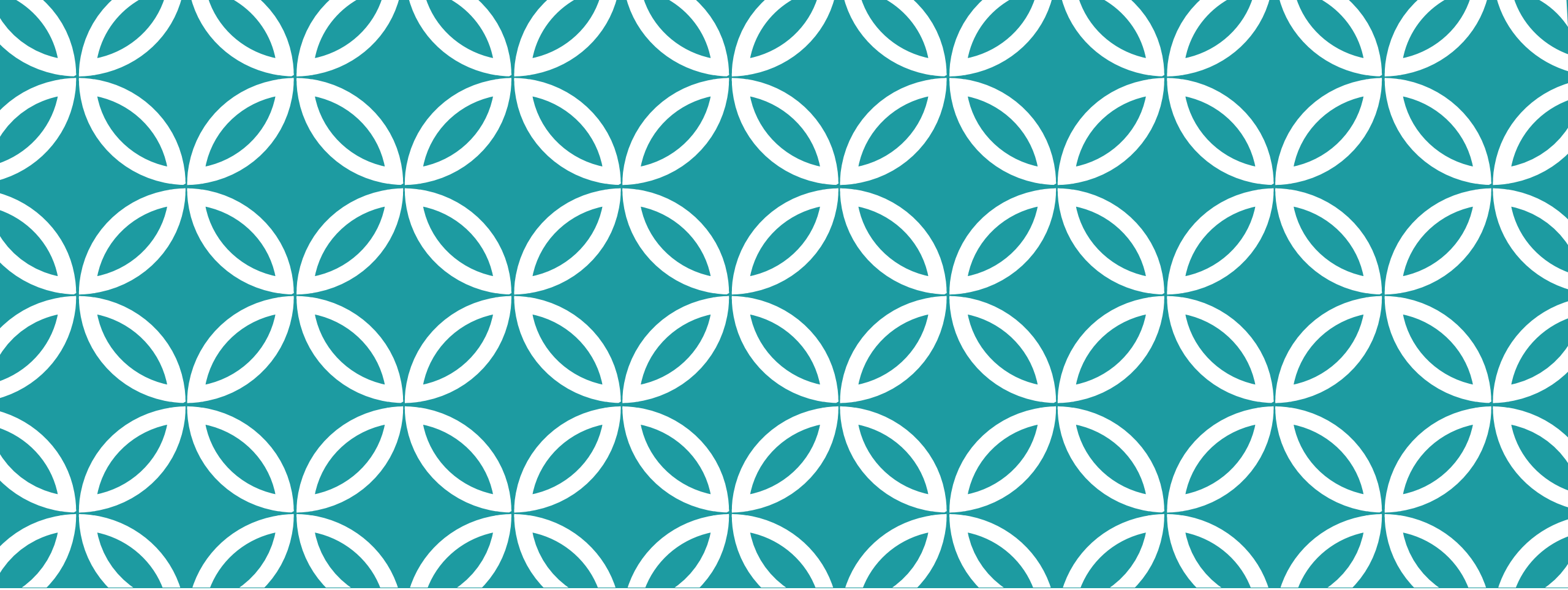
```
const numberOfProcessors : nat is 2
```

```
(base) po-hladik-pro13:dino pehladik$ selt -p dino_etr2.ktz -b -f "[](p__task__t__send_1_t7_manager_1_t9 => ([](p__task__t__
send_1_t5 => ( () (- p__task__t__send_1_t5 ) U p__task__t__read_1_t4 ))))"
Selt version 3.5.0 -- 05/29/19 -- LAAS/CNRS
ktz loaded, 1146 states, 1446 transitions
0.003s
FALSE
0.002s
```

DINO TUTORIAL: VERIFICATION

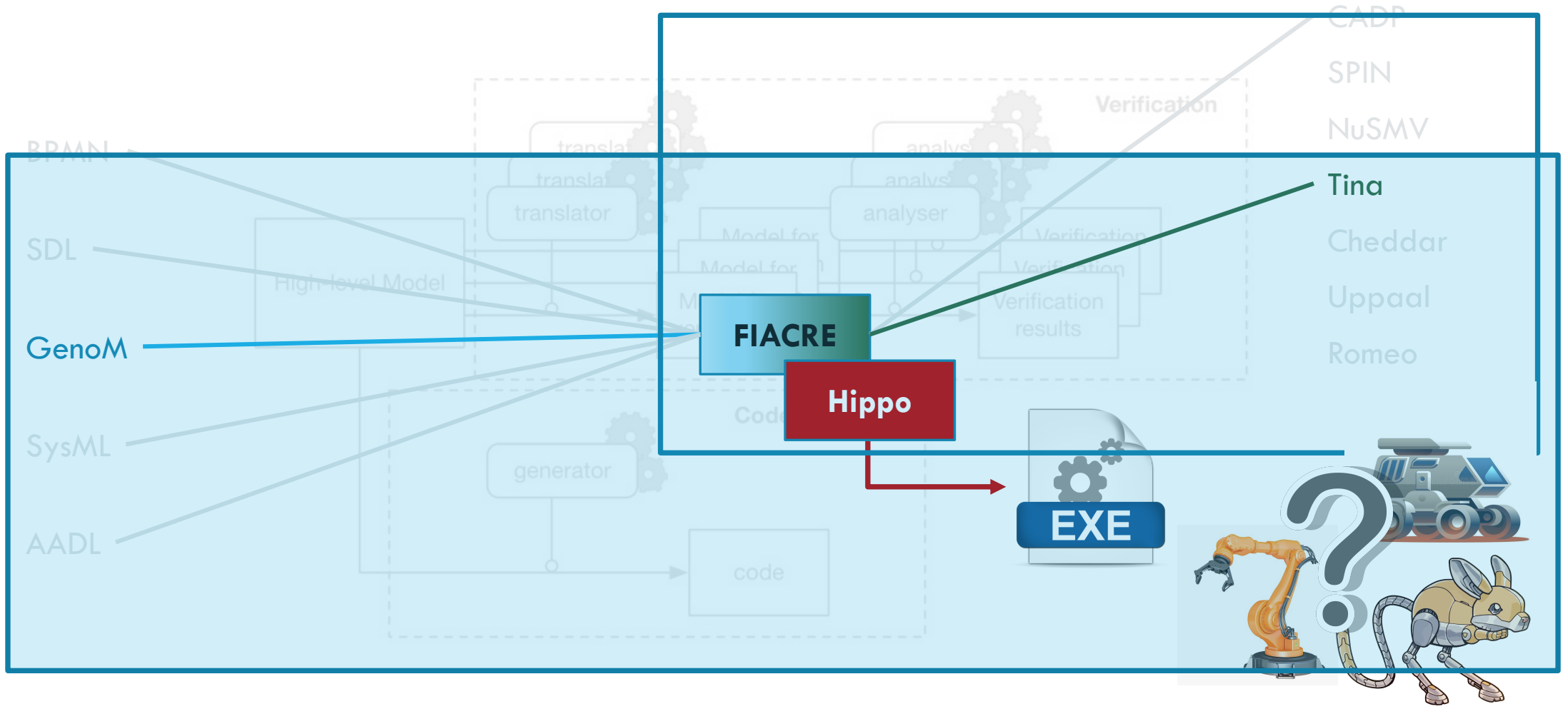
AND A SHORT EXPLANATION

```
(base) po-hladik-pro13:dino pehladik$ selt -p dino_etr2.ktz -c -f "[](p__task__t__send_1_t7_manager_1_t9 => ([](p__task__t__send_1_t5 => ( () (- p__task__t__send_1_t5 ) U p__task_
_t__read_1_t4 ))))"
Selt version 3.5.0 -- 05/29/19 -- LAAS/CNRS
ktz loaded, 1146 states, 1446 transitions
0.003s
FALSE
  state 0: L.scc*541 global__clock_1_stimer manager_1_ss1 p__event__e__keyboard_1_swaiting p__task__t__init_1_swaiting p__task__t__read_1_swaiting p__task__t__send_1_swaiting p__t
ask__t__ww_1_swaiting periodic__clock_1_ss1 reader_1_ss1 scheduler_1_sexec trainer_1_ss1 manager_1_vkey*3 p__event__e__keyboard_1_vret Hossi_1_vunused__proc*2 {Hossi_1_vd.xObstacl
e}*900
  -p__task__t__init_1_t0_manager_1_t0 ... (preserving T)->
  state 71: L.scc*467 global__clock_1_stimer manager_1_ss8 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_swaiting p__task__t__send_1_sterminating
 p__task__t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss1 scheduler_1_sexec trainer_1_ss1 manager_1_vkey p__event__e__keyboard_1_vret Hossi_1_vunused__proc*2 {Hossi_1_vd.xObst
acle}*900
  -p__task__t__send_1_t7_manager_1_t9 ... (preserving p__task__t__send_1_t7_manager_1_t9)->
  state 906: L.scc*462 global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_swaiting p__task__t__send_1_swaiting p_
_task__t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss1 scheduler_1_sexec trainer_1_ss1 manager_1_vkey p__event__e__keyboard_1_vret Hossi_1_vunused__proc*2 {Hossi_1_vd.xObstacl
e}*900
  -p__event__e__keyboard_1_t3 ... (preserving T)->
  state 1031: global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_ssynchronizing p__task__t__send_1_ssched__termin
ated p__task__t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss3 scheduler_1_sexec trainer_1_ss6 manager_1_vkey trainer_1_vduck p__event__e__keyboard_1_vret {p__task__t__read_1_v
ret.speed}*10 {p__task__t__read_1_vret.xObstacle}*10 {p__task__t__read_1_vret.yObstacle}*5 Hossi_1_vunused__proc {Hossi_1_vd.speed}*10 {Hossi_1_vd.xObstacle}*10 {Hossi_1_vd.yObsta
cle}*5
  -p__task__t__send_1_t5 ... (preserving - p__task__t__read_1_t4 /\ p__task__t__send_1_t5)->
  state 1033: global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_ssynchronizing p__task__t__send_1_ssynchronizing
 p__task__t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss3 scheduler_1_sexec trainer_1_ss6 manager_1_vkey trainer_1_vduck p__event__e__keyboard_1_vret {p__task__t__read_1_vret.
speed}*10 {p__task__t__read_1_vret.xObstacle}*10 {p__task__t__read_1_vret.yObstacle}*5 Hossi_1_vunused__proc*2 {Hossi_1_vd.speed}*10 {Hossi_1_vd.xObstacle}*10 {Hossi_1_vd.yObstacl
e}*5
  -p__event__e__keyboard_1_t3 ... (preserving - p__task__t__read_1_t4)->
  state 1067: global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_sgen p__task__t__send_1_srunning p__task__t__ww_
1_swaiting periodic__clock_1_ss2 reader_1_ss3 scheduler_1_sexec trainer_1_ss6 manager_1_vkey trainer_1_vduck p__event__e__keyboard_1_vret {p__task__t__read_1_vret.speed}*10 {p__ta
sk__t__read_1_vret.xObstacle}*10 {p__task__t__read_1_vret.yObstacle}*5 {Hossi_1_vd.speed}*10 {Hossi_1_vd.xObstacle}*10 {Hossi_1_vd.yObstacle}*5
  -p__task__t__send_1_t4 ... (preserving - p__task__t__read_1_t4)->
  state 1069: global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_sgen p__task__t__send_1_ssched__terminated p__ta
sk__t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss3 scheduler_1_sexec trainer_1_ss6 manager_1_vkey trainer_1_vduck p__event__e__keyboard_1_vret {p__task__t__read_1_vret.speed}
*10 {p__task__t__read_1_vret.xObstacle}*10 {p__task__t__read_1_vret.yObstacle}*5 {Hossi_1_vd.speed}*10 {Hossi_1_vd.xObstacle}*10 {Hossi_1_vd.yObstacle}*5
  -p__task__t__send_1_t5 ... (preserving p__task__t__send_1_t5)->
  state 1070: global__clock_1_stimer manager_1_ss5 p__event__e__keyboard_1_sforever p__task__t__init_1_swaiting p__task__t__read_1_sgen p__task__t__send_1_ssynchronizing p__task__
t__ww_1_swaiting periodic__clock_1_ss2 reader_1_ss3 scheduler_1_sexec trainer_1_ss6 manager_1_vkey trainer_1_vduck p__event__e__keyboard_1_vret {p__task__t__read_1_vret.speed}*10
{p__task__t__read_1_vret.xObstacle}*10 {p__task__t__read_1_vret.yObstacle}*5 Hossi_1_vunused__proc {Hossi_1_vd.speed}*10 {Hossi_1_vd.xObstacle}*10 {Hossi_1_vd.yObstacle}*5
  [accepting all]
0.002s
```



HOW TO SAY GENOM IN H-FIACRE

OUTLINE


BPMN
SDL
GenoM
SysML
AADL
FIACRE
Hippo
EXE
CADP
SPIN
NuSMV
Tina
Cheddar
Uppaal
Romeo
translator
analyser
generator
code
Verification
High-level Model
Verification results

WHY MIX ROBOTICS AND FORMAL METHODS

- Formal verification is one approach, among others, to increase the trust we have in robotic systems

- Already used in many critical domains
  - with safety standards: transport, energy, … and without: space, military, etc.
- It does not solve "all the problems"
  - but it is a step in the right direction, and it is very good at challenging preconceived ideas
- It can be integrated in existing frameworks

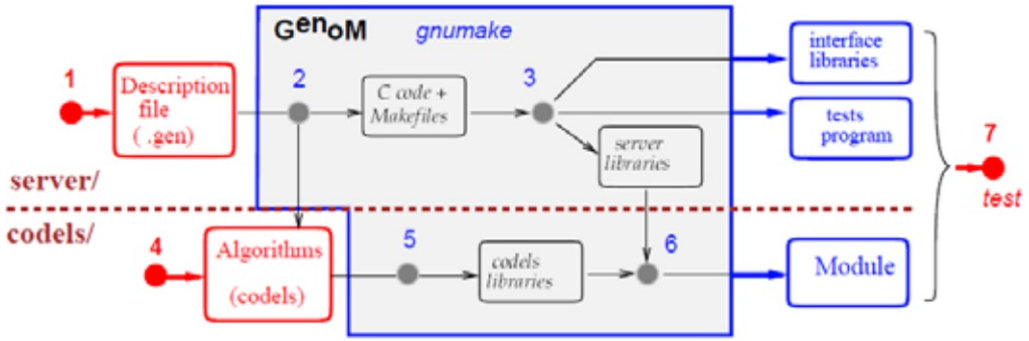
GENOM : GENERATOR OF MODULES

Answer the question: how do you program these ?

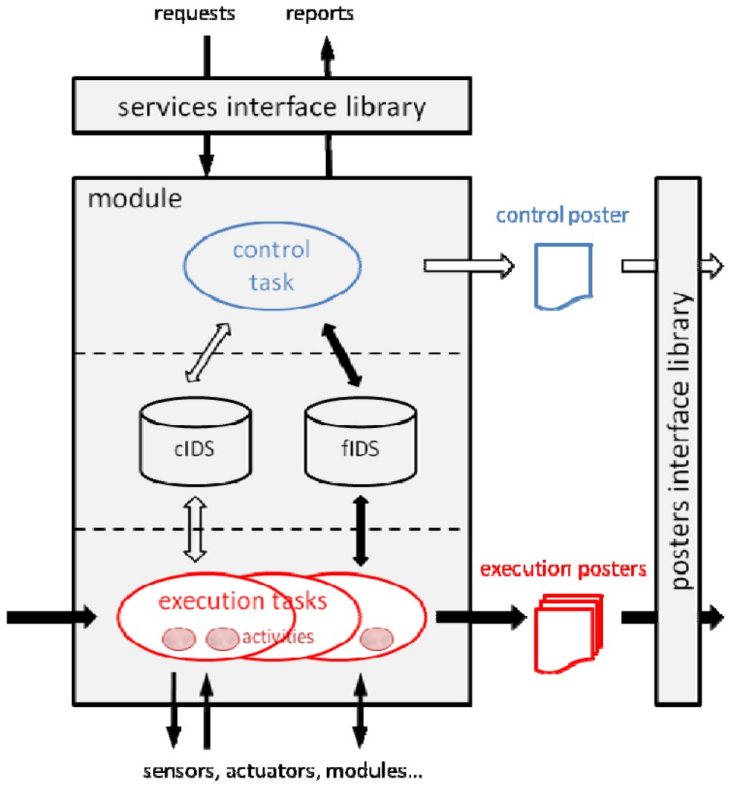
Used and developed by the RIS team, at LAAS, for the last 25 years

LAAS CNRS

GENOM ARCHITECTURE

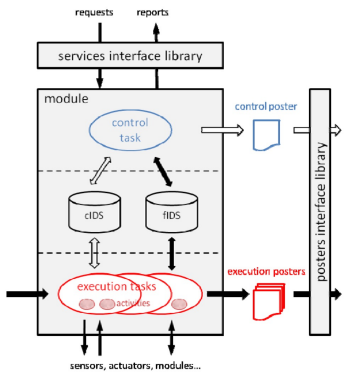


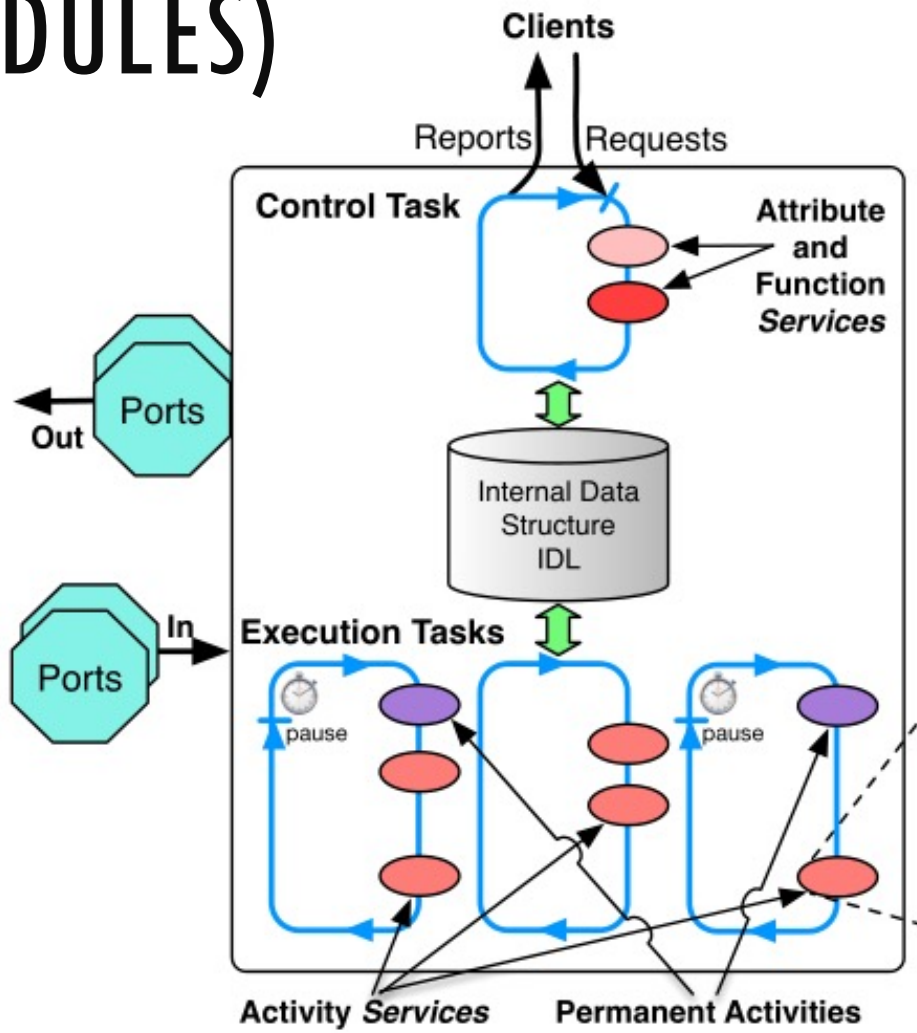
Development cycle

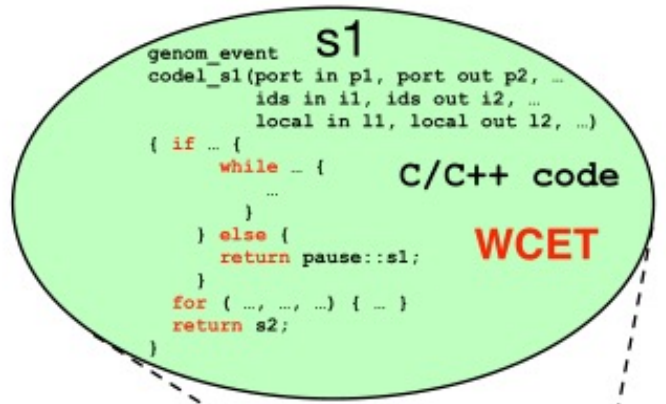


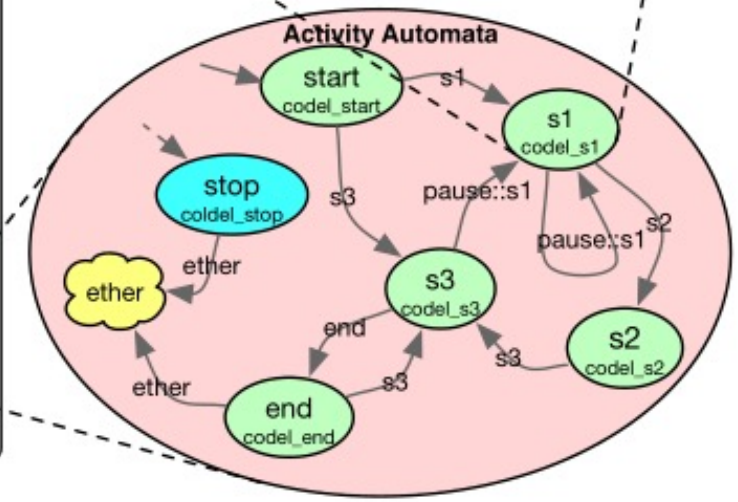
Module structure

GENOM (MODULES)









GENOM (CODE)

velodyne

Task: scan
period: 10ms
Services:
Init, GetScans, GetOneScan, SavePCD

Task: pose
period: 10ms
Services:
StartPoseProcessing, SetFixedSensorPose

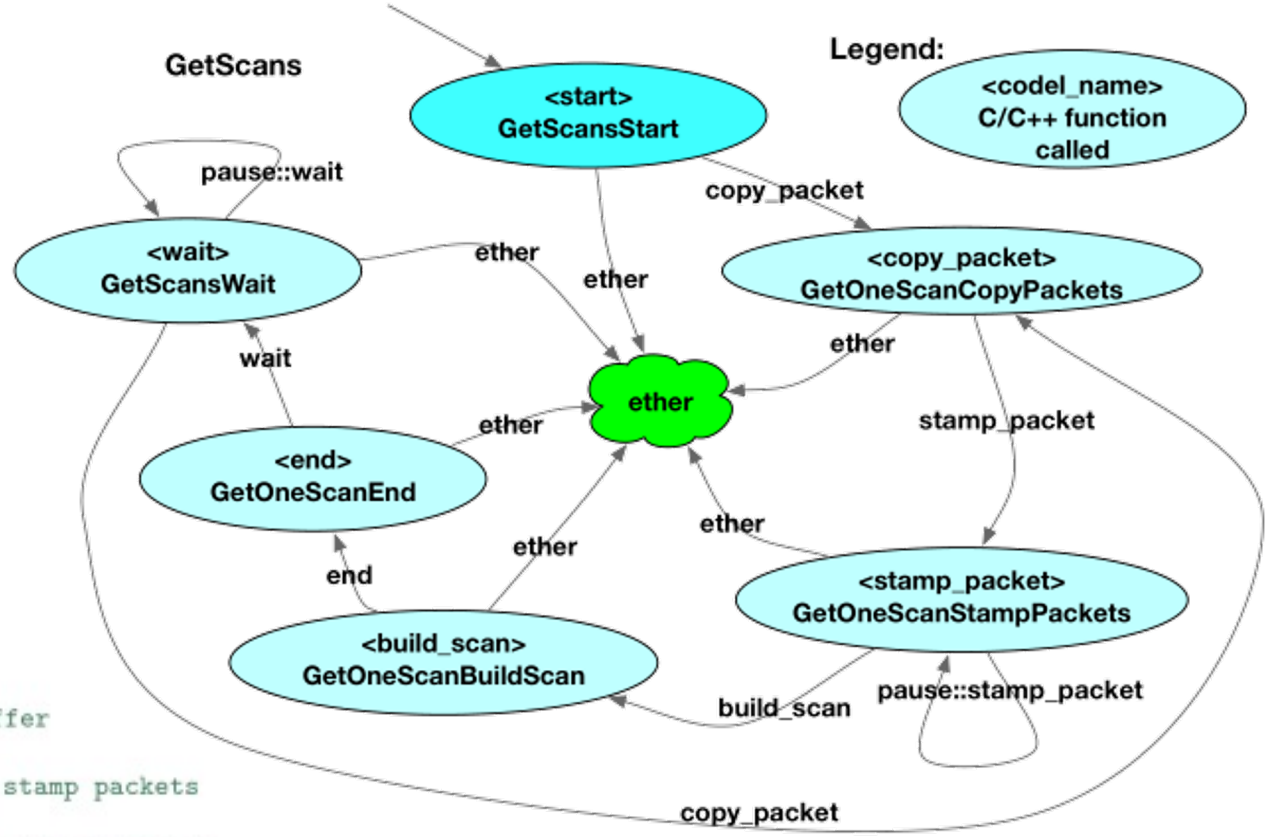
Task: acquisition
aperiodic
Services:
StartAcquisition

SetDelay, StopAcquisition, StopGetScans, {Setup,Stop}PoseProcessing

```
#pragma require "openrobots2-idl >= 2.0"
#pragma require "minnie-idl"

#include "or/pose/pose_estimator.gen"
#include "mi/sensor/pcl.idl"

component velodyne {
  doc               "Provides corrected scans from Velodyne sensors.";
  version           "1.0";
  lang              "c";
  email             "felix@laas.fr";
  require           "genom3 >= 2.99.26";
  codels-require    "velodyne-libs >= 0.7", "eigen3", "pcl_common-1.7", "pcl_io-1.7";

  port in  or_pose_estimator::state robot_pose;
  port out or::pcl point_cloud;

  exception e_sys { string<256> what; };
//...
  exception e_port { string<256> what; };

  /* --- ids ------------------------------------------------------------- */
  ids {
    AcquisitionParams acquisition_params; // Acquisition parameters
    PacketBuffer packet_buffer; // Buffer to store time stamped raw packets
    PoseBuffer pose_buffer;     // Buffer to store time stamped pose.
    long fd;                    // file descriptor to get the raw packets (UDP)

    long usec_delay;            // for fault injection purpose to delay port scan writting
  };

  attribute SetDelay(in usec_delay =  1000000)
  {
    doc         "Set the delay in usec we will delay port update (to test the BIP monitor).";
  };

  /* --- acquisition task ------------------------------------------------ */
  task acquisition {
    codel<start>        velodyneAcquisitionTaskStart() yield ether;
    codel<stop>         velodyneAcquisitionTaskStop()  yield ether;

    throws e_mem, e_grabber;
  };


  activity StartAcquisition() {
    doc                 "Starts the data acquisition";
    task                acquisition;
```

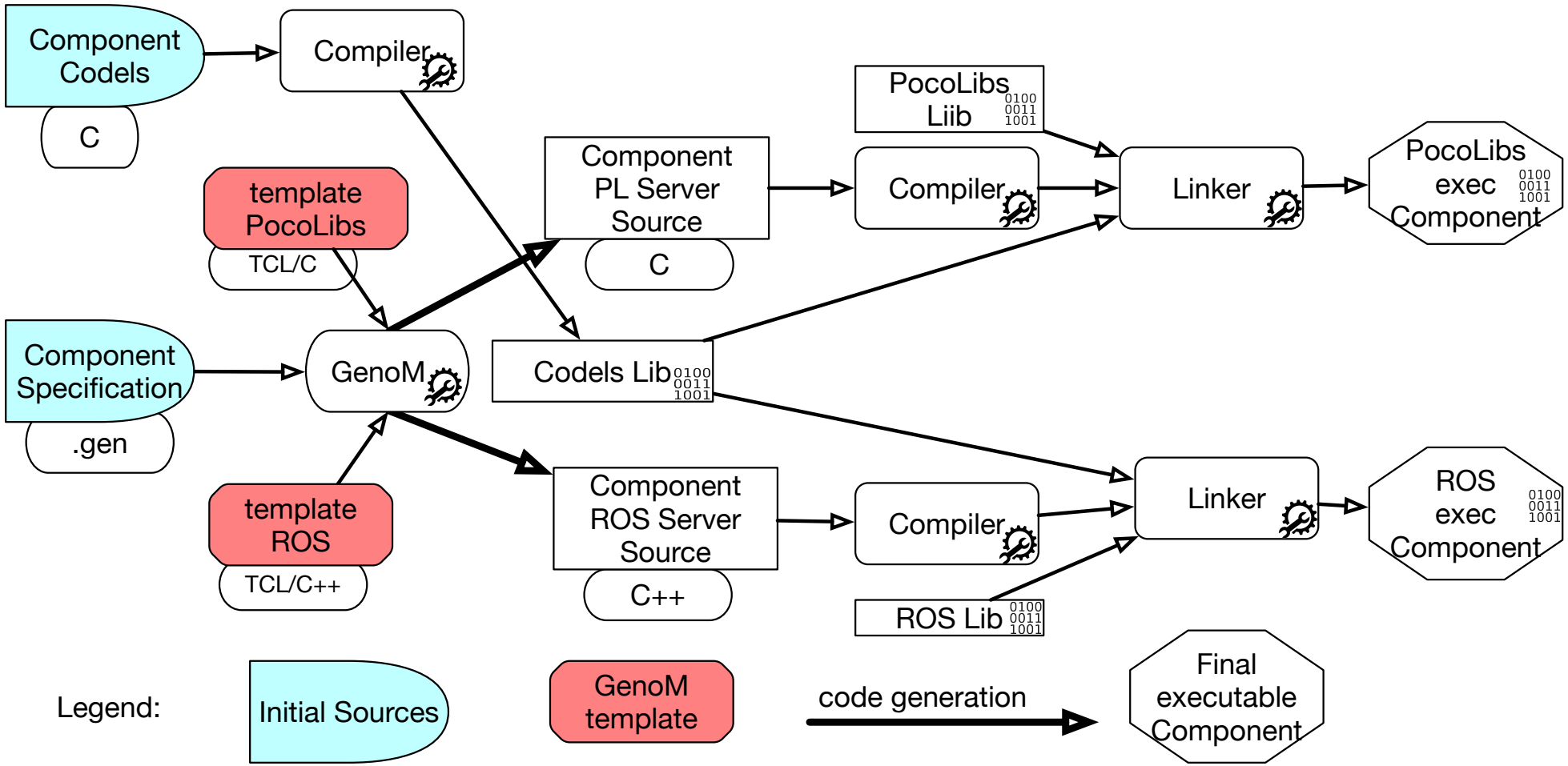
GENOM (CODE)

```
activity GetScans(
     in double firstAngle = :"First angle of the scan (in degrees)",
     in double lastAngle  = :"Last angle of the scan (in degrees)",
     in double period     = :"Time in between two scans",
     in double timeout    = :"Timeout used when stamping packets")
{
 doc  "Acquire full scans from the velodyne sensor periodically";
 task scan;

 validate GetScansValidate(in firstAngle, in lastAngle, in period);

 codel <start> GetScansStart(in acquisition_params)
              yield copy_packets;
 codel <copy_packets> GetOneScanCopyPackets(in acquisition_params,
                 inout scan_buffer) // get packets from acquisition buffer
              yield stamp_packets;
 codel <stamp_packets> GetOneScanStampPackets(in acquisition_params, // stamp packets
                 inout pose_data, in timeout) // with the proper  pose
              yield pause::stamp_packets, build_scan; // pause:: if pose not available
 codel <build_scan> GetOneScanBuildScan(in acquisition_params,
                 in firstAngle, in lastAngle) // build scan repositioning
              yield end;               // individual packet in the first pose.
 codel <end> GetOneScanEnd(in acquisition_params,
                 port out point_cloud, inout usec_delay) //publish the scan in the
              yield wait;  // point_cloud port. usec_delay is for fault injection.
 codel <wait> GetScansWait(in period) // wait next user defined scan period
              yield pause::wait, copy_packets; // then loop back.

 interrupts GetOneScan, SavePCD, GetScans;
};
```



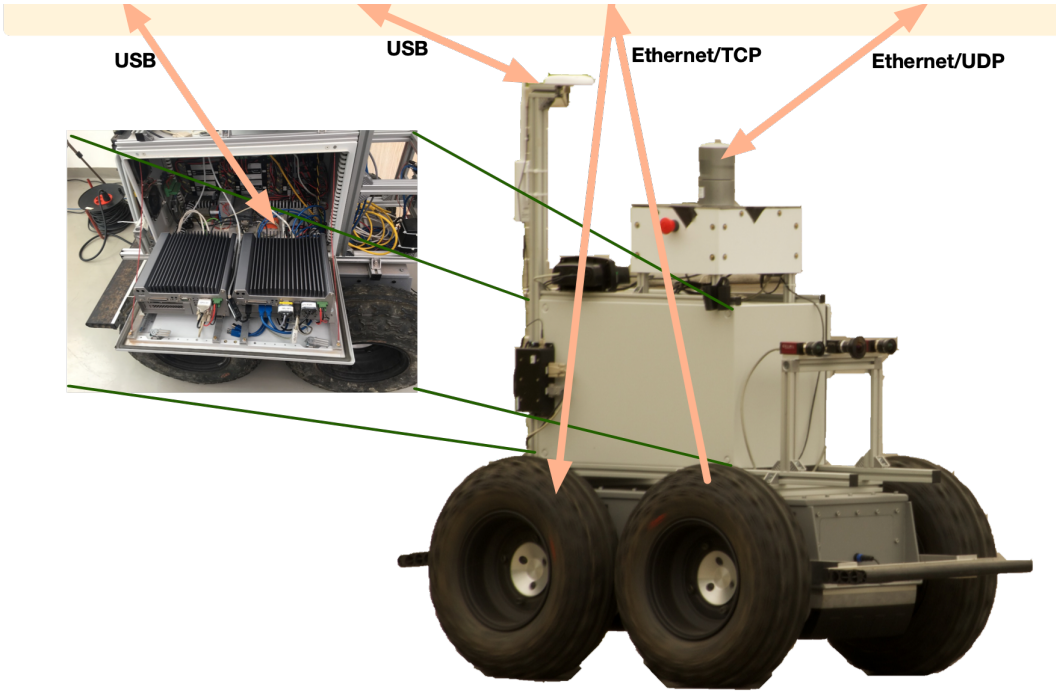
scan task of the velodyne module

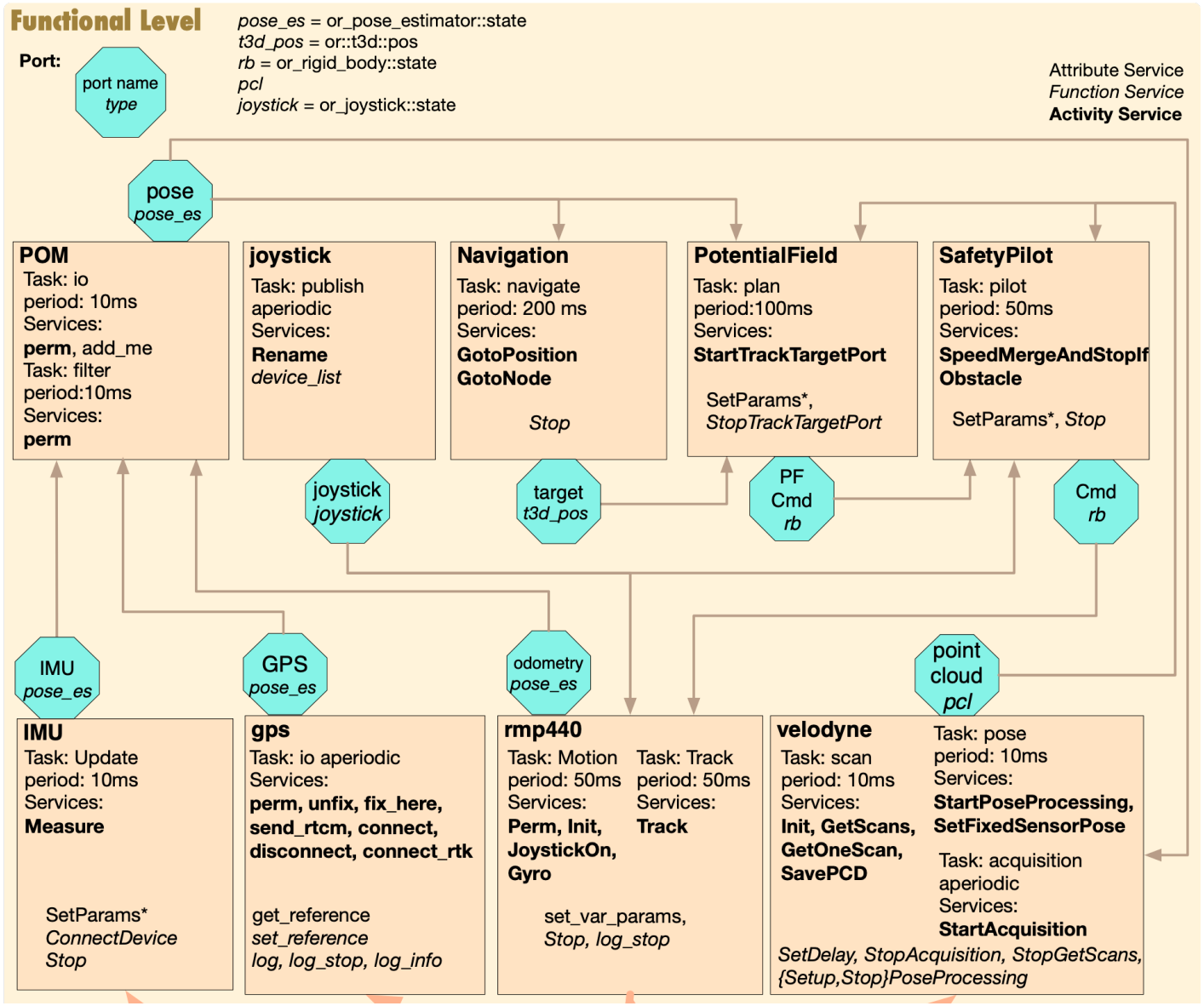
GENOM EXECUTABLE TOOLCHAINS



RMP440: MINNIE

- Segway RMP 440
- Fast (up to 8 m/s)
- GPS ; Gyro ; IMU ; Velodyne LIDAR
- 2 recent CPUs (4 cores)





DRONE

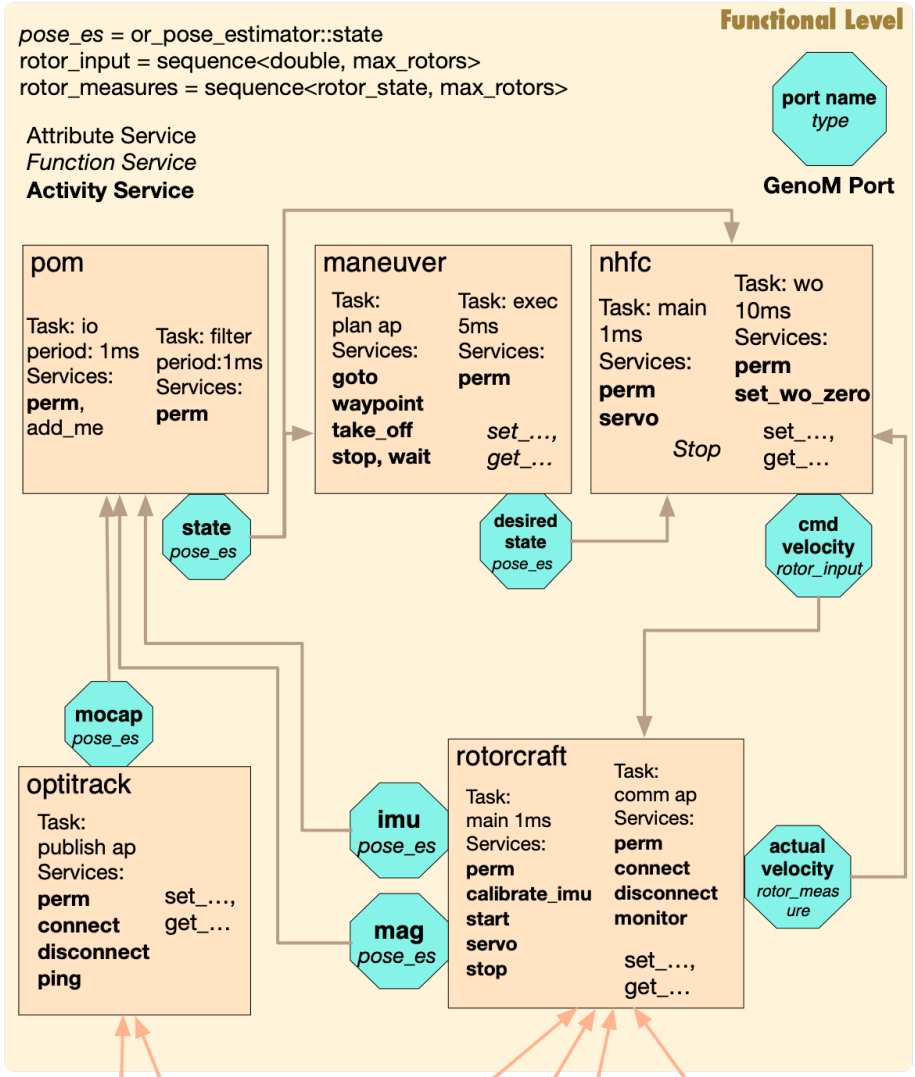
Motion Capture localization

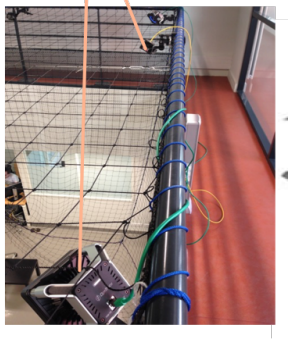
IMU (angular velocities and accelerations)

Control each propeller velocity separately

Only 1 CPU

Update frequency is ~ 1kHz





SOME INTERESTING CHARACTERISTICS OF GENOM

It is **opinionated**

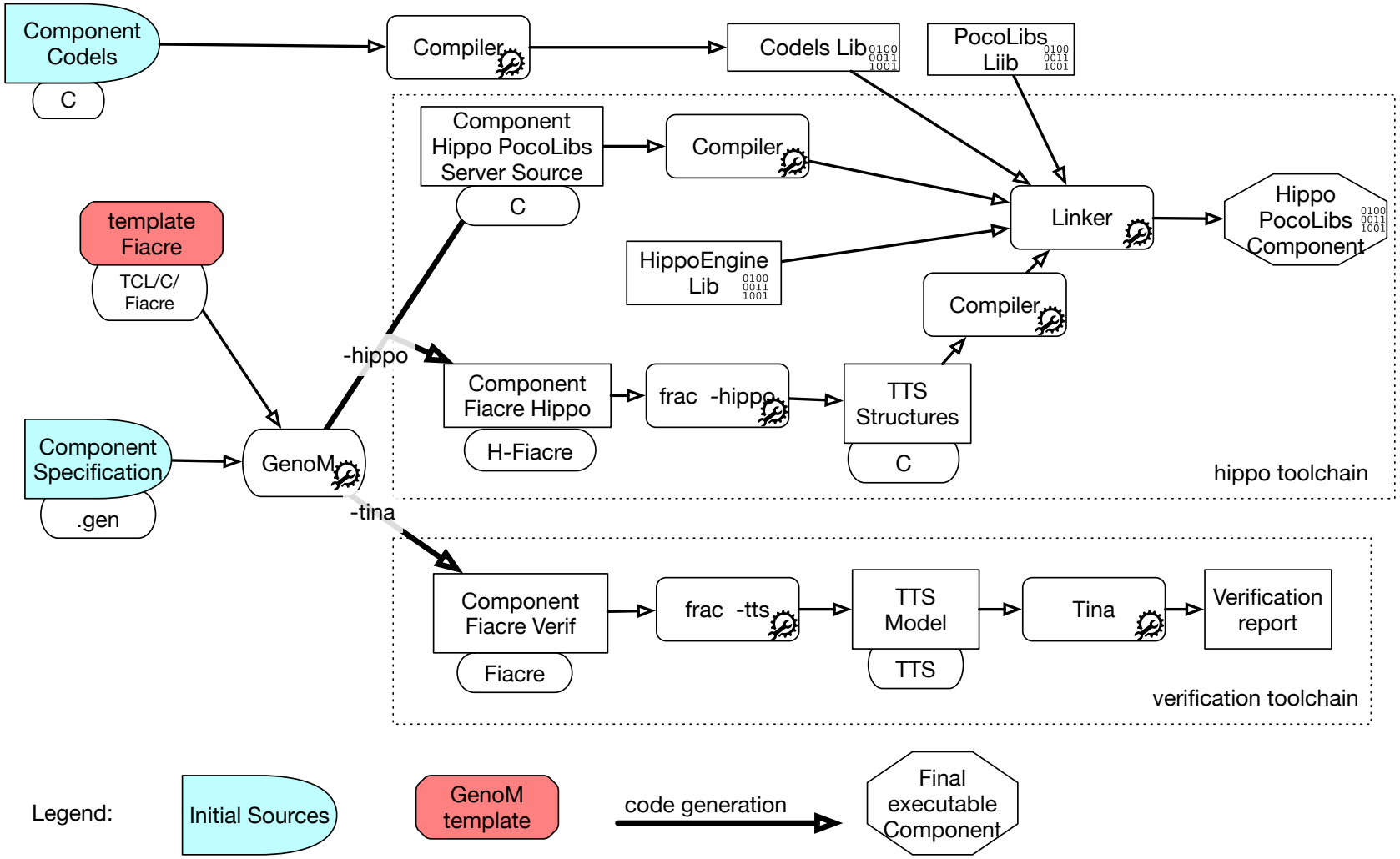
Impose a rather "rigid/strict" way to define components, with little room for fooling/messing around. "Everything is here", as little magic as possible

- It relies on a **templating mechanism** to generate all the artifacts
- It is explicit about **error handling** and possible **failure scenarios**

It is middleware independent

- It uses explicit constructs to express real-time constraints and requirements: tasks, periods, WCET, etc.
- Behavioral description based on state machines and synchronization on ports

GENOM-HIPPO TOOLCHAIN



GENOM TO H-FIACRE

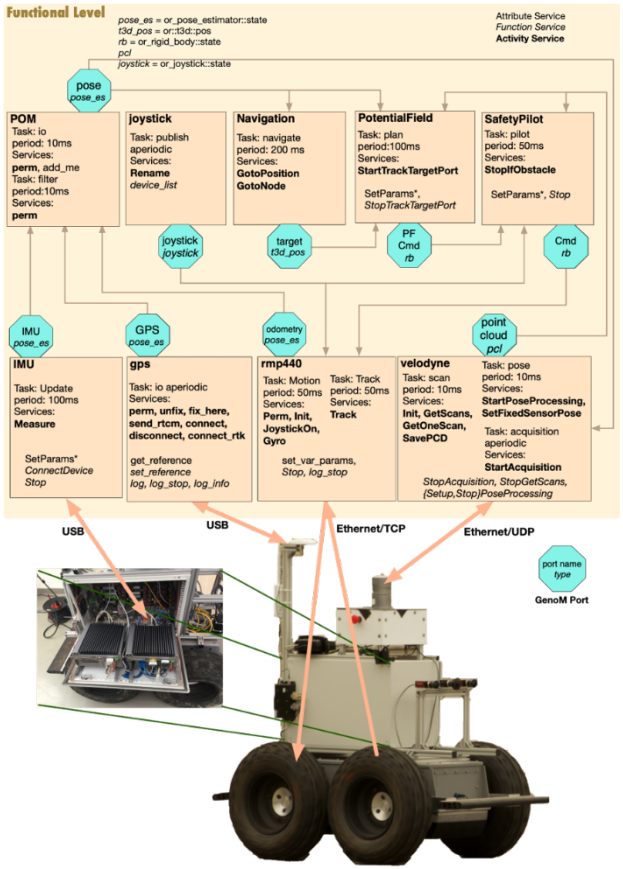
The interpretation of GenoM into Fiacre is quite precise

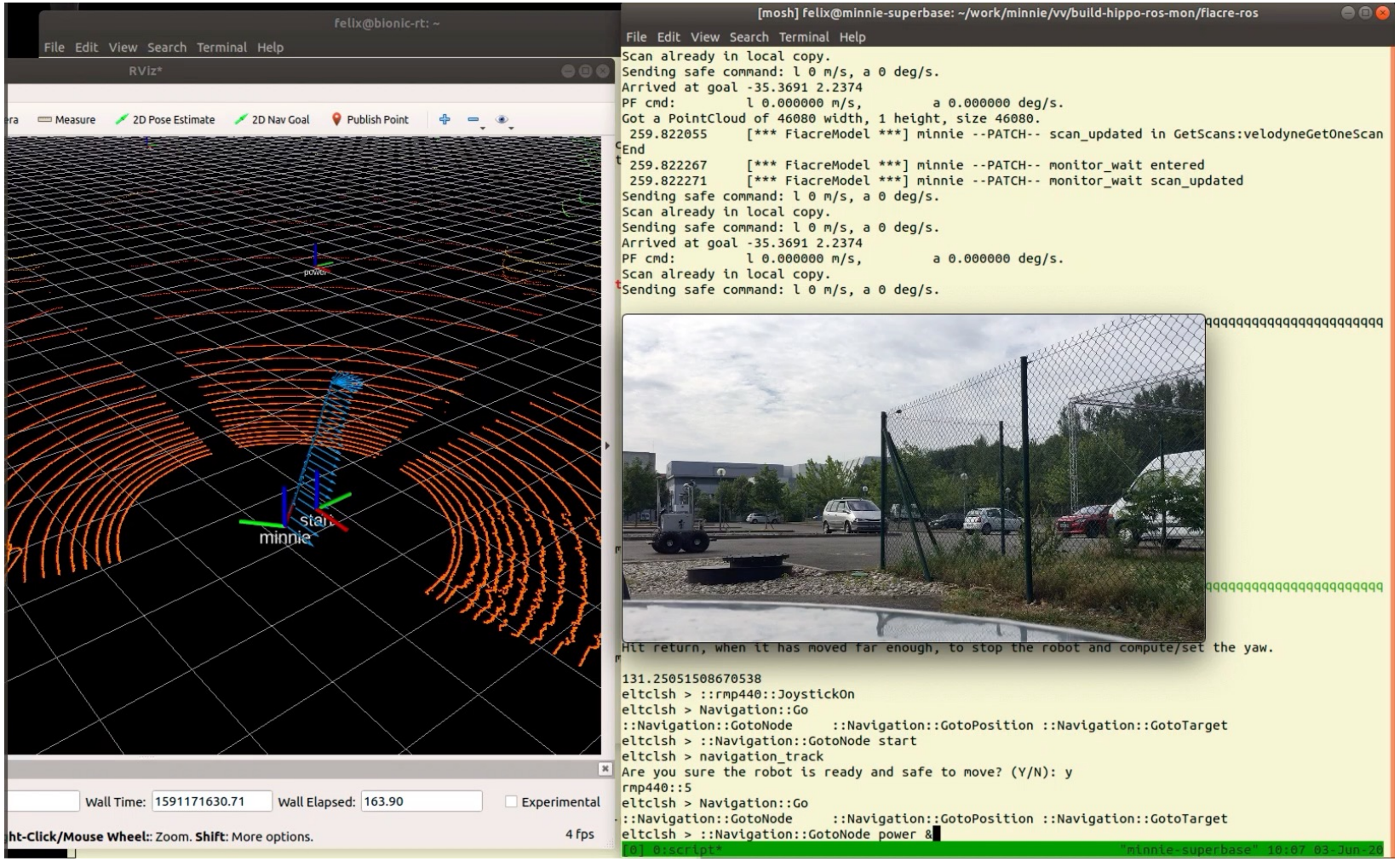
- We get the state of each execution task and activities
- We know what messages are exchanged/stored in the IDS
- We track every timing constraints (timeouts, periods, activations, …)

It only lacks knowledge about the behavior of codels

MINNIE

https://youtu.be/vXZiW5tOG54





CONTROLLING MINNIE WITH HIPPO

- The GenoM spec for Minnie compiles into a Hippo model with **197 tasks, 9 event ports, 441 extern functions, 1780 (Petri) transitions**
- Hippo runs the whole experiment at 10 kHz in one process
- The load is ≈ 5-10% above normal GenoM usage, without noticeable slowdown
  - We report task period overshoots
  - We detect possibly uninitialized port reads

OFFLINE VERIFICATION FOR MINNIE

- Schedulabity: it is an invariant, [] – task_overshoot
  - We can take into account the number of cores (we found scheduling errors when using the Velodyne component with less than 3 cores)
- Mutual Exclusion: (also a safety property)

Scenario	*JoystickOn* then *Track*	*Track* then *JoystickOn*
Time	16 min	10 h
#classes	42,714,945	832,778,752
#markings	5,817,082	44,533,432

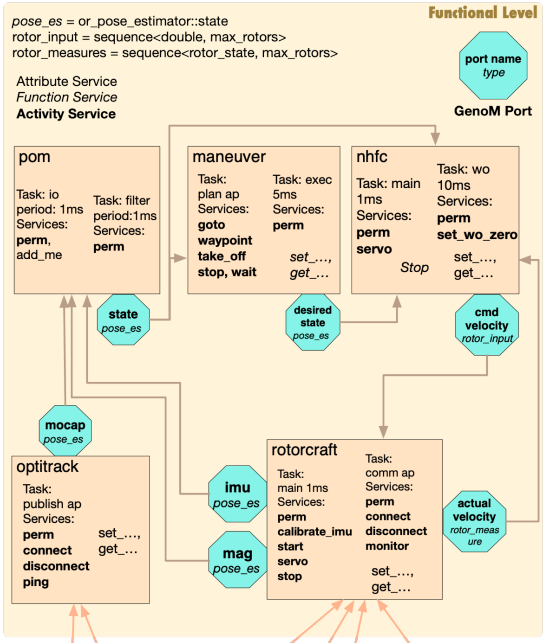
- Delay to Stop: example of quantitative property
  - We found a WCRT of 141 ms (85 cm before we brake),
  - To be compared with a WCET of 43 ms for the slowest codel

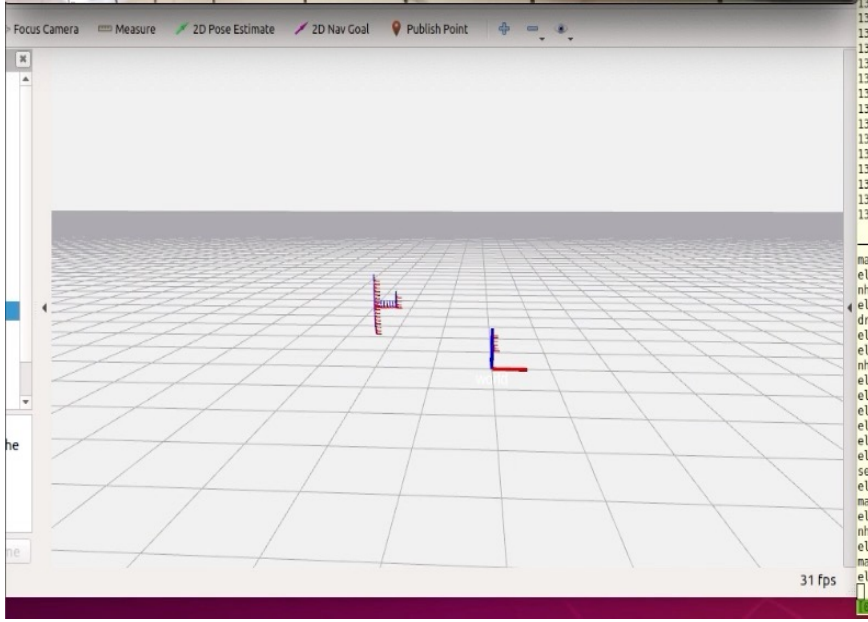
WHAT DID WE DO WITH ALL THIS?

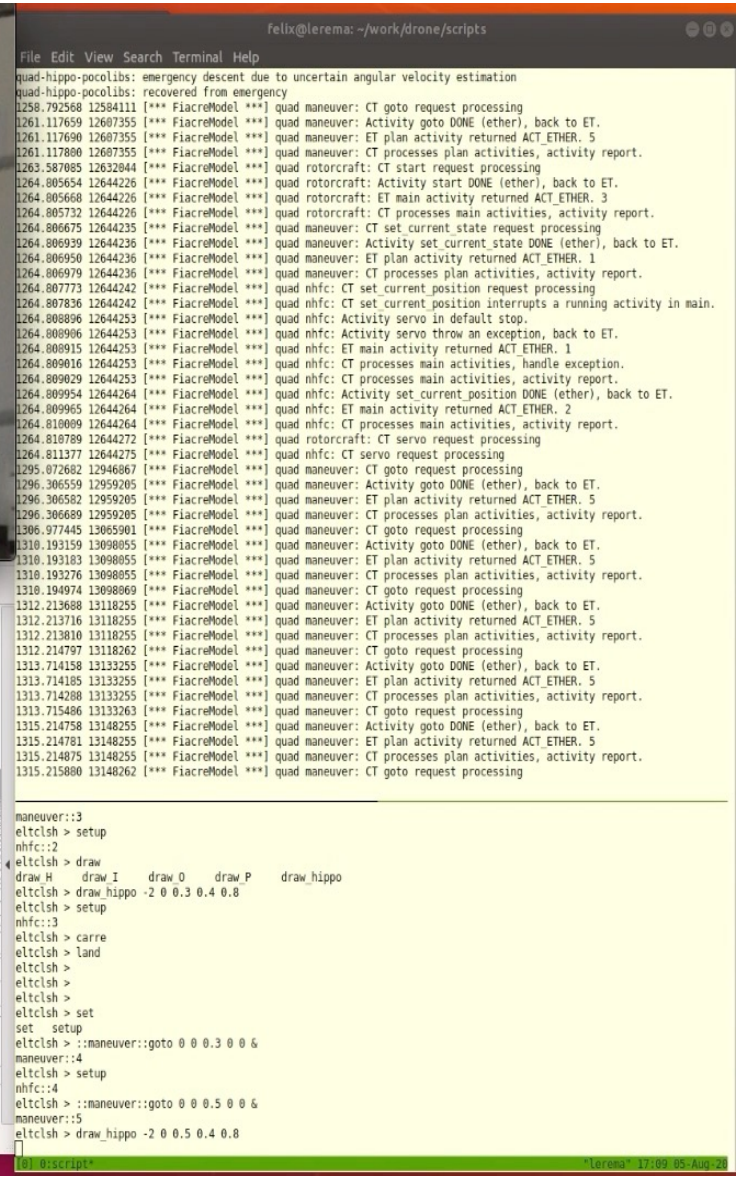
- We used it to check and "run" GenoM specifications (does it run ?)
- We checked properties, "offline" (is it true that … ?)
- A validation of the Hippo Engine (can we trust it ?)
  - We check that *runtime executions* $\subseteq$ *traces in the formal model*
- An empirical analysis of the Hippo Engine (does it scale well ?)
- We checked properties, online (can we monitor it ?)

HIPPODRONE

https://youtu.be/3Ok_c-ATY8I



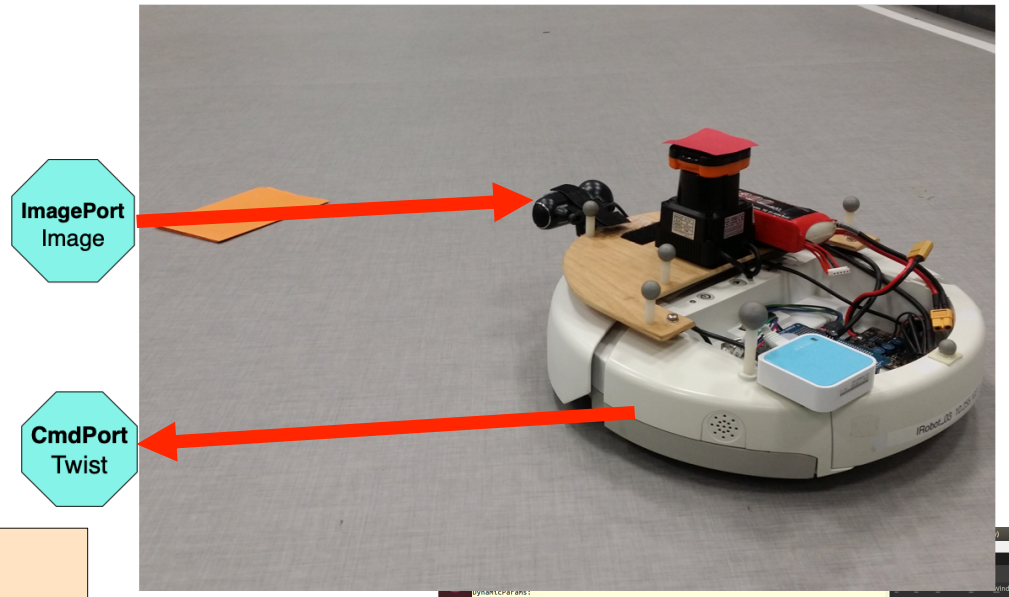




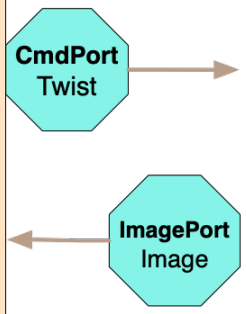
STUDENT LAB WORK: COLORTRACK-ROBOT

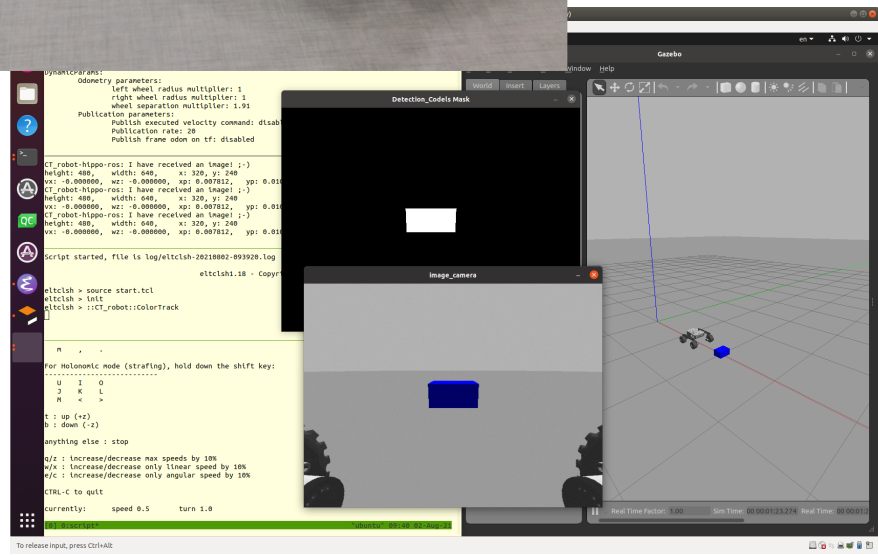
https://hub.docker.com/r/felixfi/hippo-ct-robot

- Provides a service to track a given color (rgb) in the image
- Use simple OpenCV primitives to find the coordinates (x,y) of the barycenter of the color in the image
- Computes a speed command (vx, wz) to keep position (x,y) centered in the image
- Exports the speed in a CmdPort



CT_robot
IDS:
speed, patrol_speed
threshold
x, y, width, height
Task:
track 10ms
Services:
SetThreshold
SetPatrolSpeed
StopTracking
ColorTrack





CONCLUSION

- Operational examples
- Observation of how a roboticist uses formal verification tools
- Confrontation of approaches (formal, real-time, robotics)

[Future] New Tools

- Debuger
- Micro[controller]-engine
- Monitoring

REFERENCES

Hippo	https://gitlab.laas.fr/pehladik/hippo
GenoM3	https://git.openrobots.org/projects/genom3
Template GenoM3 Fiacre (ROS et pocolibs)	https://redmine.laas.fr/projects/genom3-fiacre-template/gollum/index
Minnie RMP440 experimentation	https://redmine.laas.fr/projects/minnie/gollum/fiacre
Hippodrone experimentation	https://redmine.laas.fr/projects/drone-v-v/gollum/index
Tutorial with Color-Track robot	https://hub.docker.com/r/felixfi/hippo-ct-robot
Article: V&V in robotic	hal-02927311
Article: Fiacre/Hippo/GenoM3	hal-03017661