Development of hard real-time systems with Giotto and frameworks

O.Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Pree
University of Salzburg
UC Berkeley (guest)

www.SoftwareResearch.net

SOFTWARE RESEARCH LAB

Contents

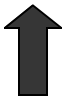
- Giotto: predictable, reusable real-time code
  - concepts
  - case study: helicopter control system
- Reusable framework components for high-level management tasks

SOFTWARE RESEARCH LAB

Giotto concepts

The History of Computer Science: Lifting the Level of Abstraction

High-level languages:
Programming to the application

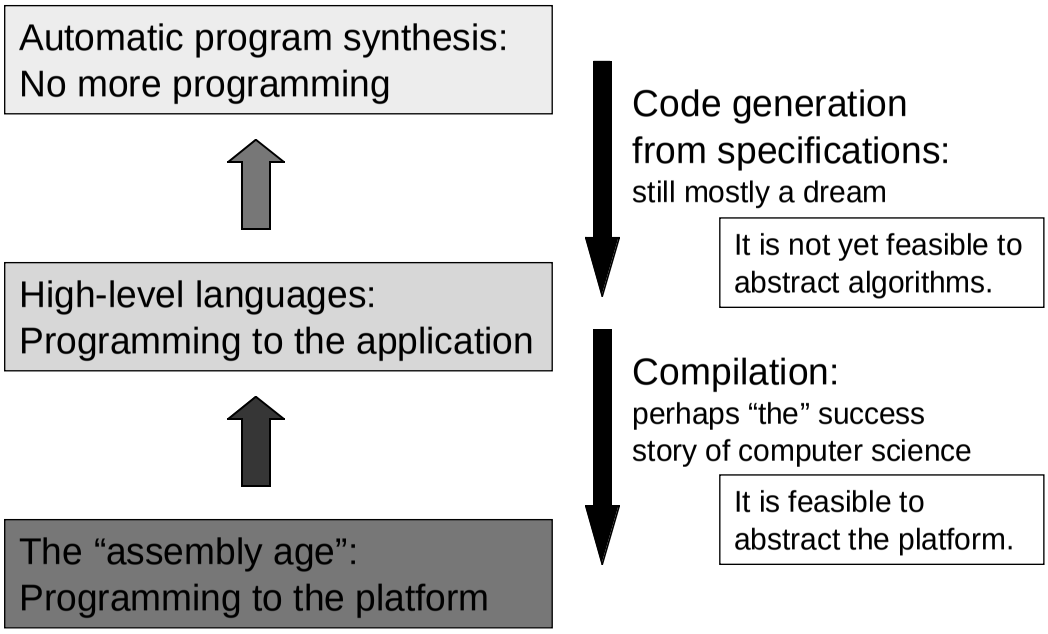


The “assembly age”:
Programming to the platform

Compilation:
perhaps “the” success story of computer science

It is feasible to abstract the platform.

The History of Computer Science: Lifting the Level of Abstraction



SOFTWARE RESEARCH LAB

Current Practice in Control Software

Some automatic
code generation from models

-often inefficient
-often unpredictable

Some manual
programming to the platform

-difficult to reuse
-difficult to verify
-requires systems experts

SOFTWARE RESEARCH LAB

Current Practice in Control Software

Some automatic
code generation from models

-often inefficient
-often unpredictable

The missing link:
platform-independent software

Some manual
programming to the platform

-difficult to reuse
-difficult to verify
-requires systems experts

SOFTWARE RESEARCH LAB

Advocated Practice in Control Software

Mathematical model
e.g. Simulink, HyTech

Control engineer

Platform-independent software
e.g. Giotto

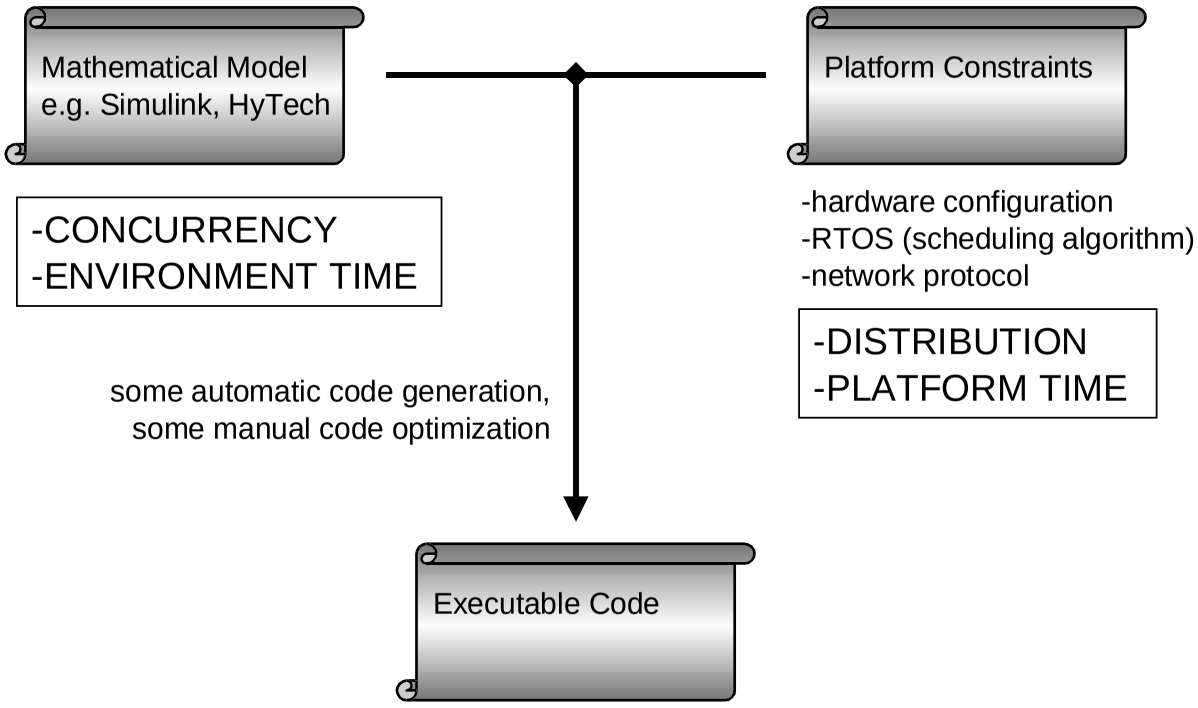
Compiler

Executable code
for a specific platform

-verifiable
-reusable
-efficiently implementable

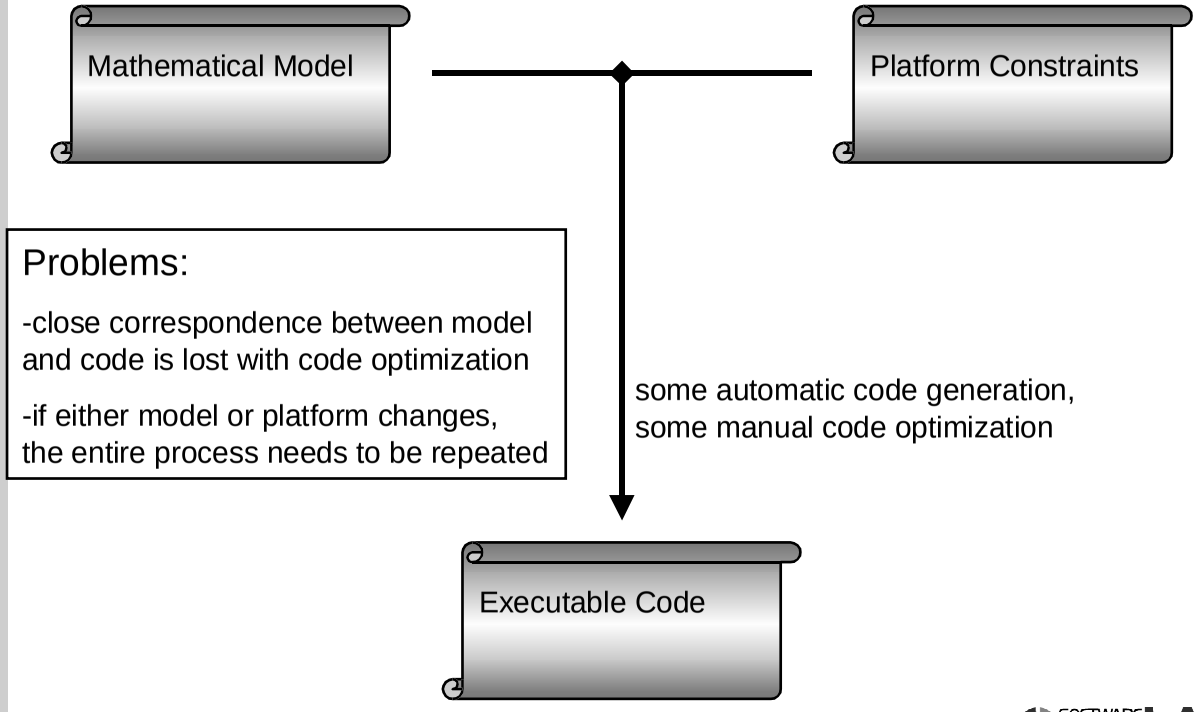
SOFTWARE RESEARCH LAB

Current Control Software Development



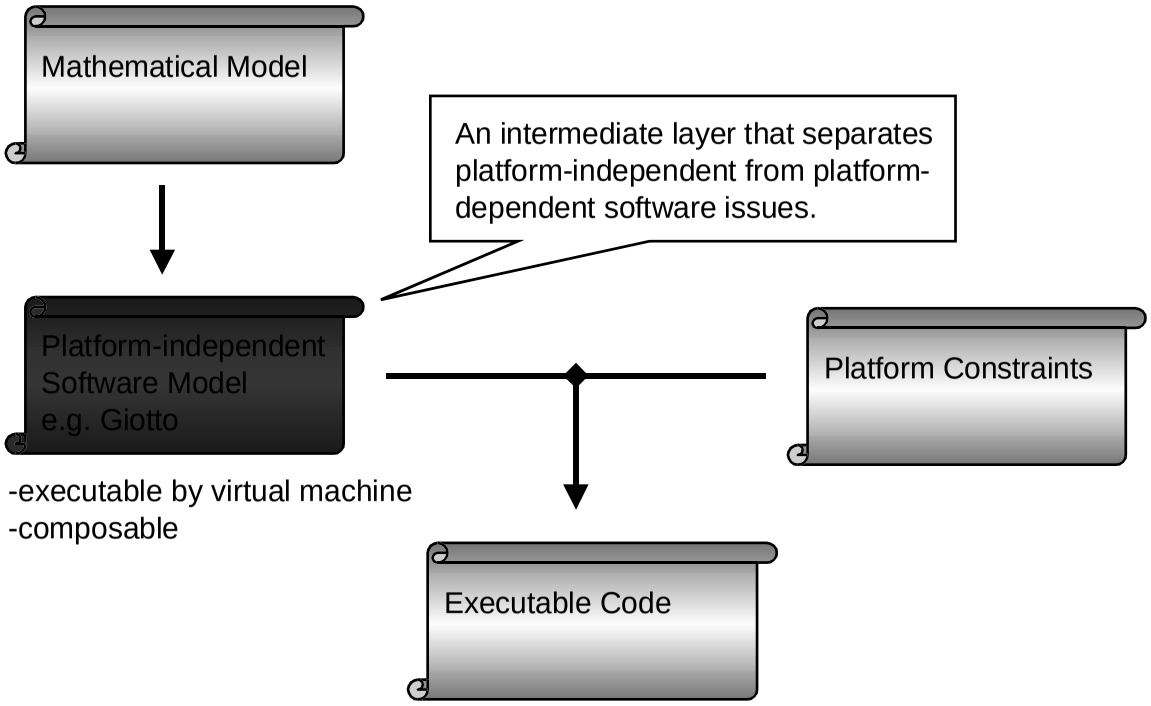
SOFTWARE RESEARCH LAB

Current Control Software Development



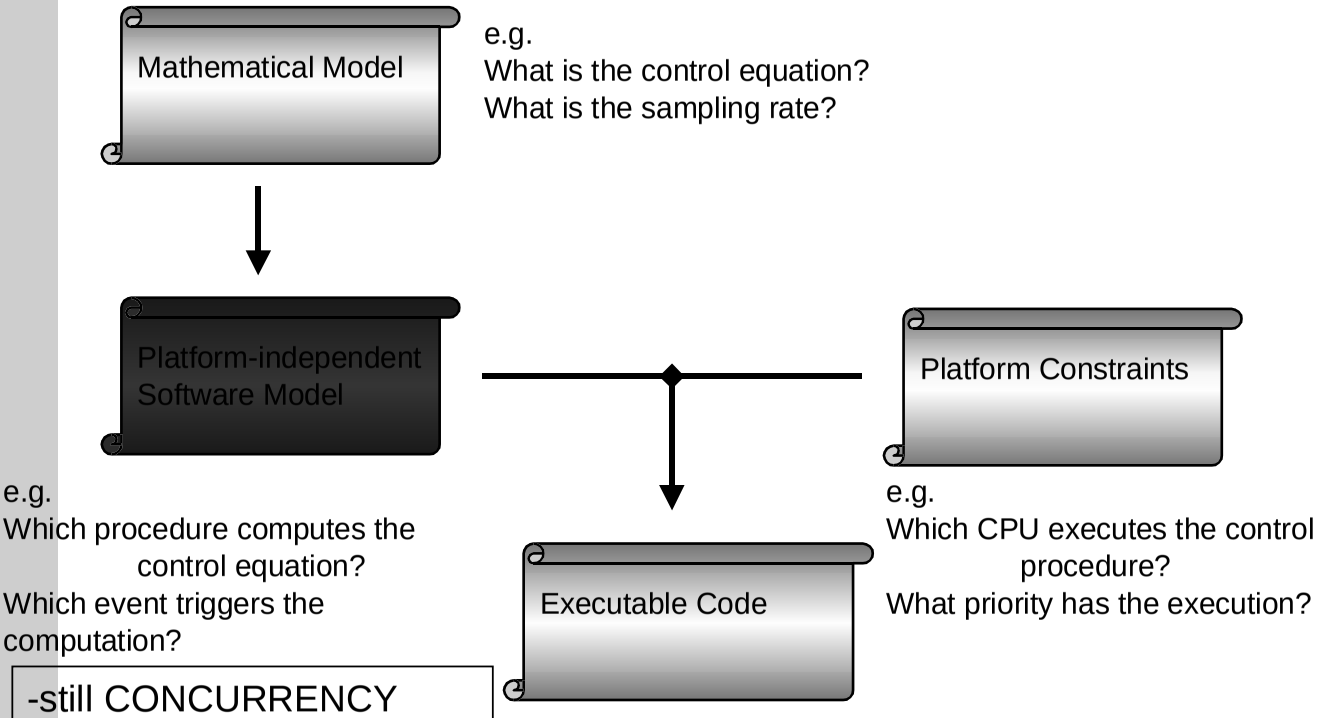
SOFTWARE RESEARCH LAB

Advocated Control Software Development



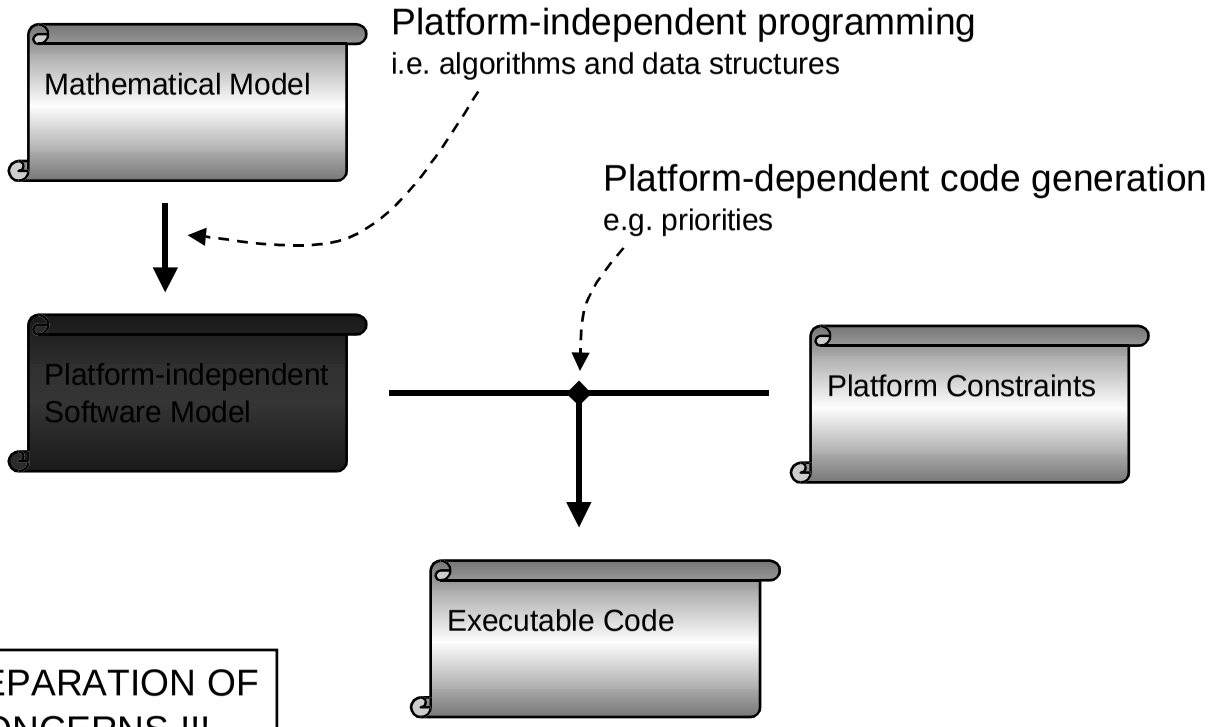
SOFTWARE RESEARCH LAB

Advocated Control Software Development



SOFTWARE RESEARCH LAB

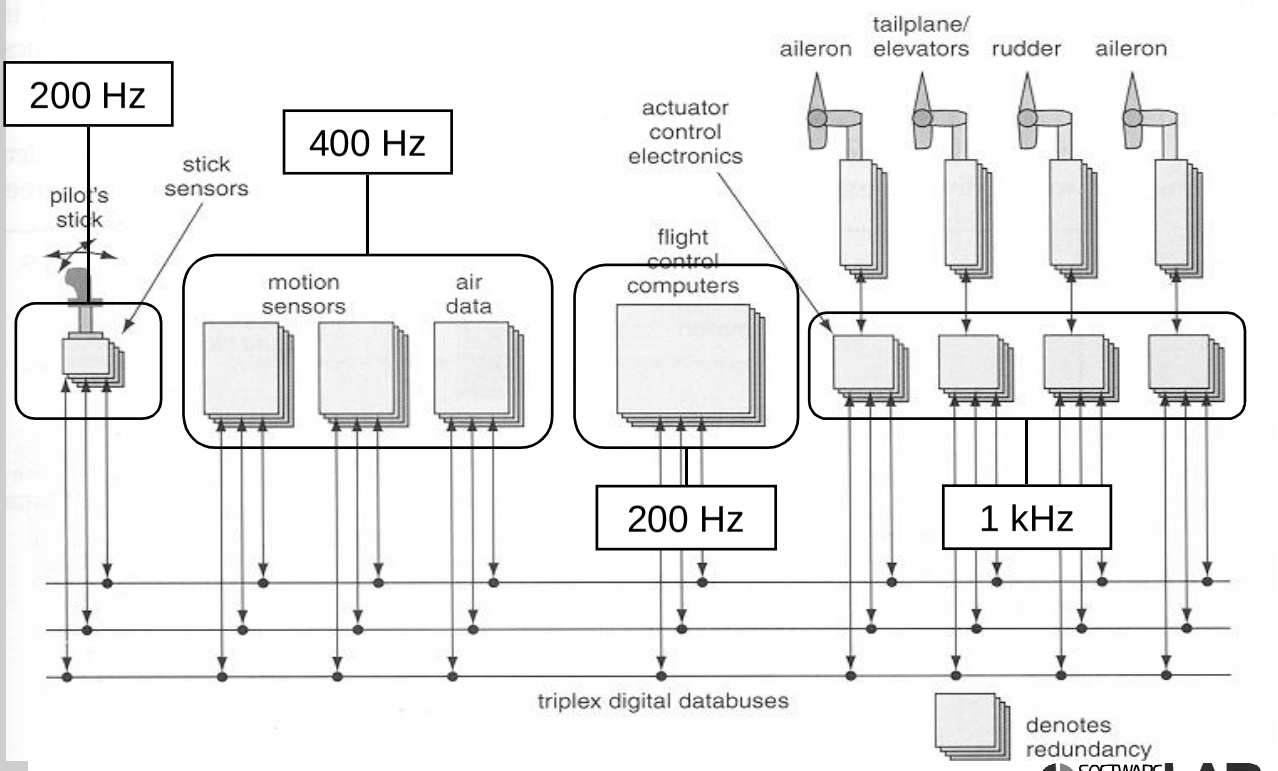
Advocated Control Software Development



Motivation: Flight Control Software

Kirsch, Pree, in cooperation with ETH Zurich (Sanvido, Schaufelberger, Wirth). Single CPU.

Motivation: Flight Control Software



Platform-independent Software Model

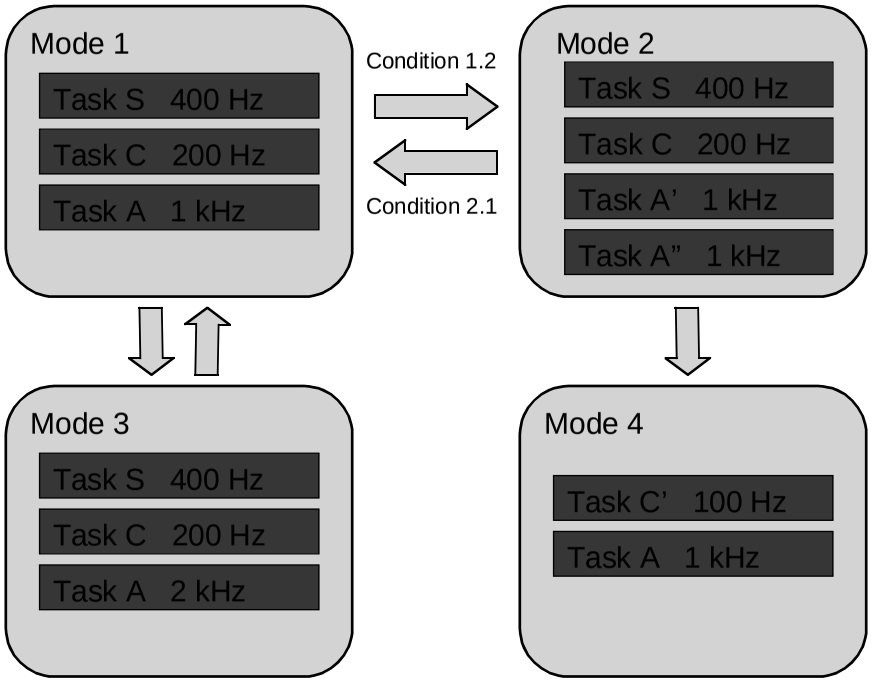
1. Concurrent periodic tasks:

   -sensing
   -control law computation
   -actuating

2. Multiple modes of operation:

   -navigational modes (autopilot, manual, etc.)
   -maneuver modes (taxi, takeoff, cruise, etc.)
   -degraded modes (sensor, actuator, CPU failures)

Platform-independent Software Model



SOFTWARE RESEARCH LAB

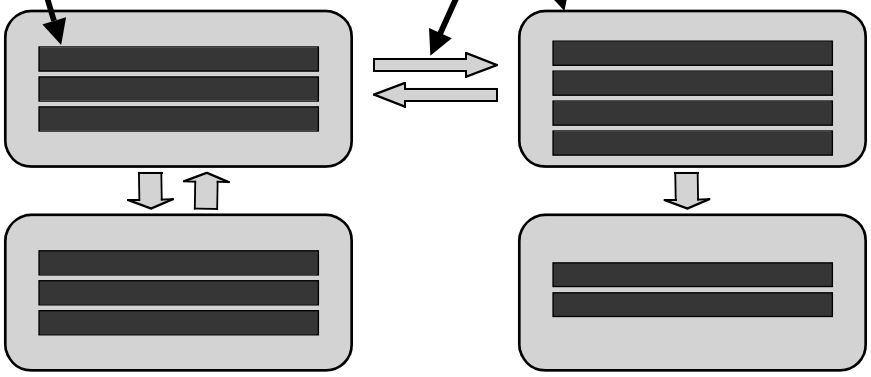
Platform-independent Software Model

Host code
e.g. C

Functionality.
-No time.
-Sequential.

Glue code
Giotto

Timing and interaction.
-Environment time, not platform time.
-Concurrency, not distribution.



This kind of software is understood:
Host code may (sometimes) be generated automatically.

The software complexity lies in the glue code (minimize jitter!) :
Giotto enables requirements-driven rather than platform-driven glue-code programming.

SOFTWARE RESEARCH LAB

The Giotto model

The Giotto Programmer's Model

Programming in terms of environment time:

Programmer's fiction:

-time-triggered task invocation
-tasks are functions with a fixed duration
-platform offers sufficient performance

Implementation in terms of platform time:

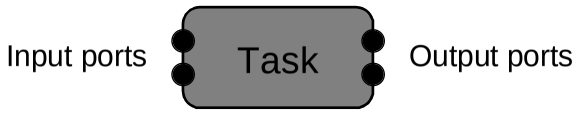
Compiler must maintain programmer's fiction:

-needs access to global time, no other platform requirements
-tasks may finish early, but outputs cannot be observed early
-tasks may be preempted and distributed

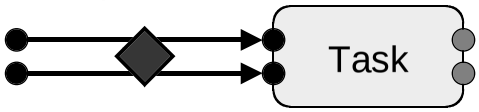
The Giotto Programmer's Model

Given:

1. Units of scheduled host code (application-level tasks).
   e.g. control law computation



2. Units of synchronous host code (system-level drivers).
   e.g. device drivers

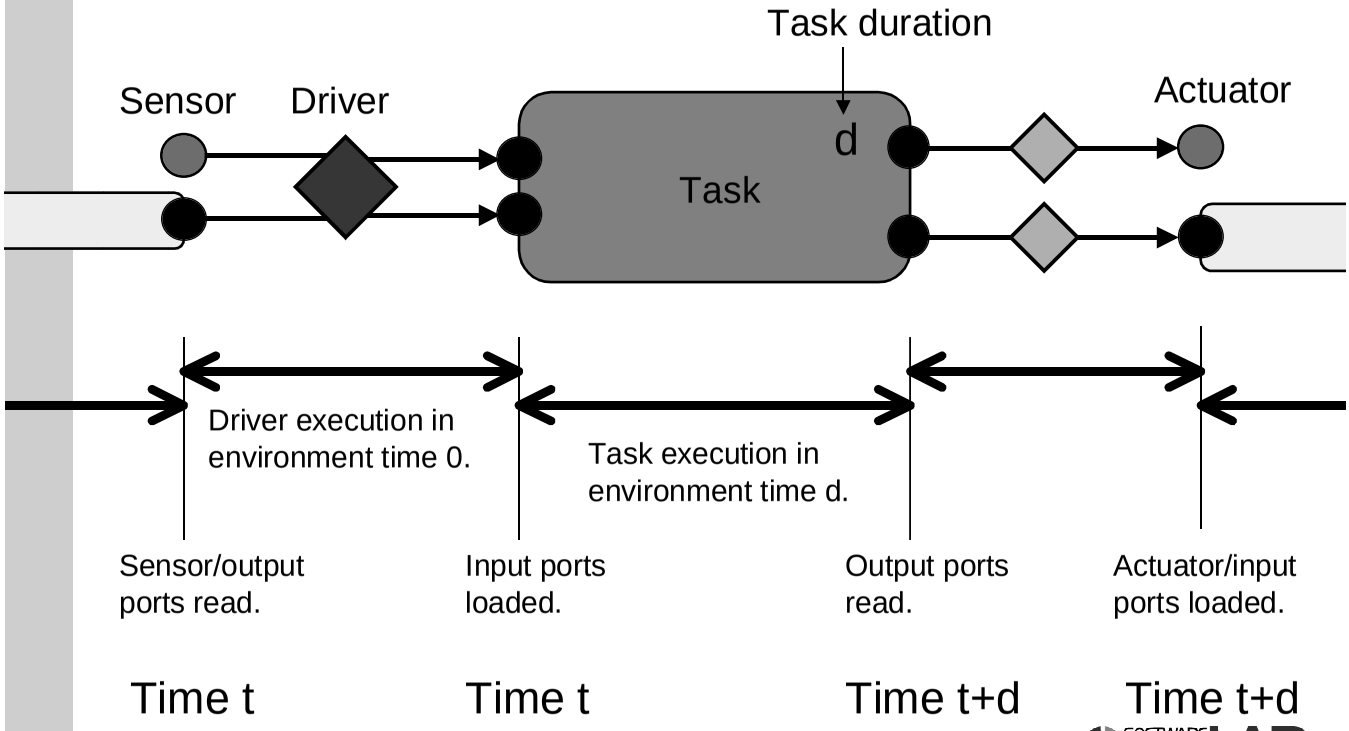


3. Real-time requirements and data flow between tasks.

Giotto: Glue code that calls 1. and 2. in order to realize 3.

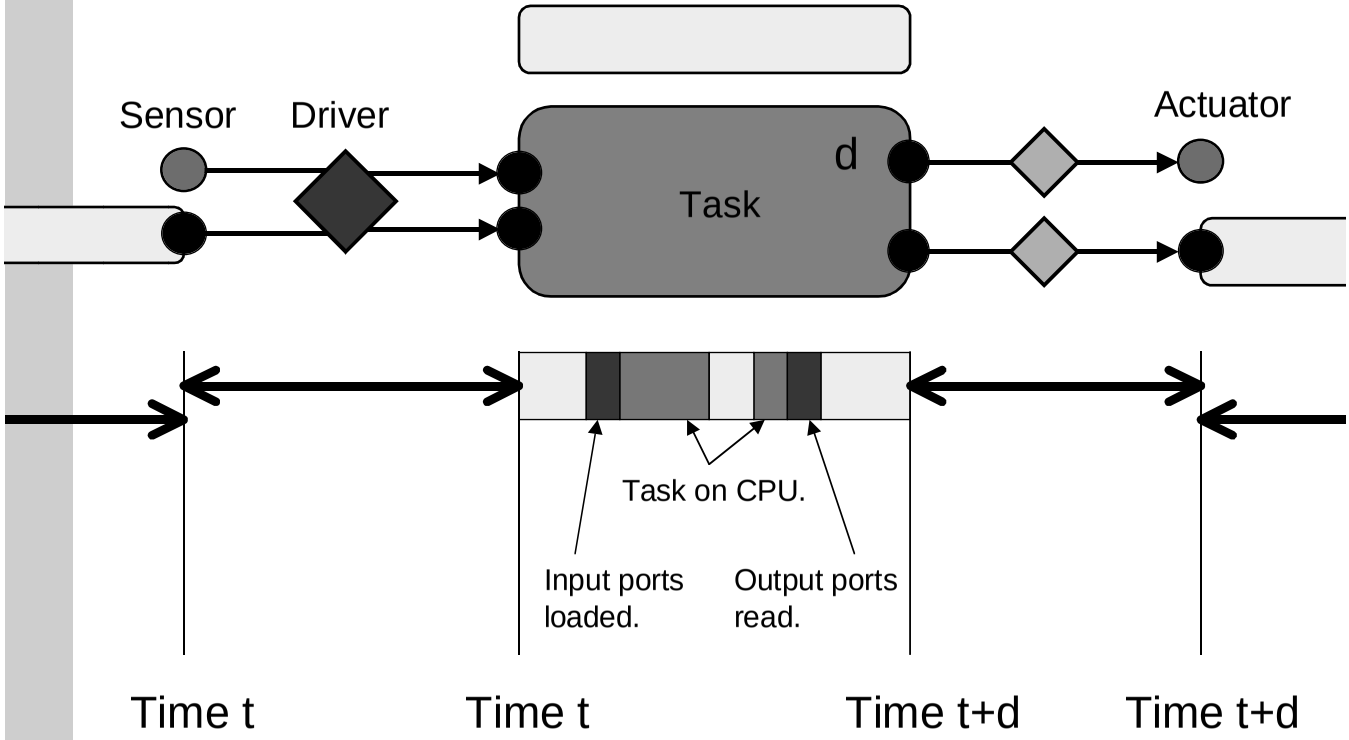
SOFTWARE RESEARCH LAB

Environment Timeline (defined by Giotto semantics)



SOFTWARE RESEARCH LAB

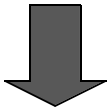
Platform Timeline (chosen by Giotto compiler)



SOFTWARE RESEARCH LAB

Platform Independence ensures Predictability

The Giotto compiler chooses for a given platform a platform timeline that is value equivalent to the environment timeline defined by the Giotto semantics.

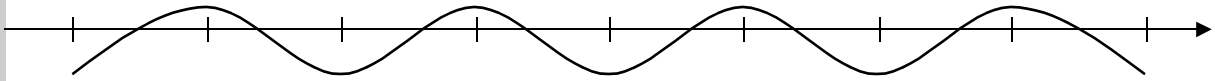


Internal Determinism:

For a given sequence of sensor readings, the corresponding sequence of actuator settings is uniquely determined (i.e., there are no race conditions).

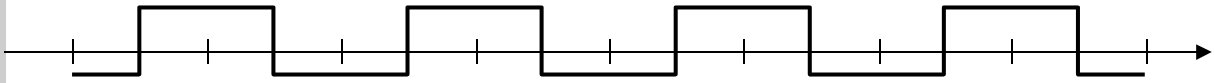
SOFTWARE RESEARCH LAB

Environment



Environment Processes: environment time

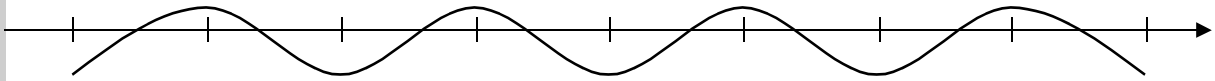
Software Processes: platform time



Software

SOFTWARE RESEARCH LAB

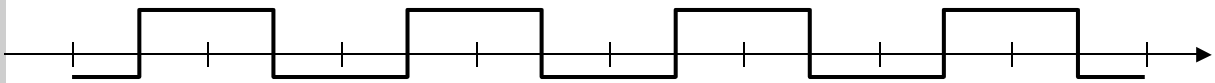
Environment



Reactivity

The Art of Embedded Programming

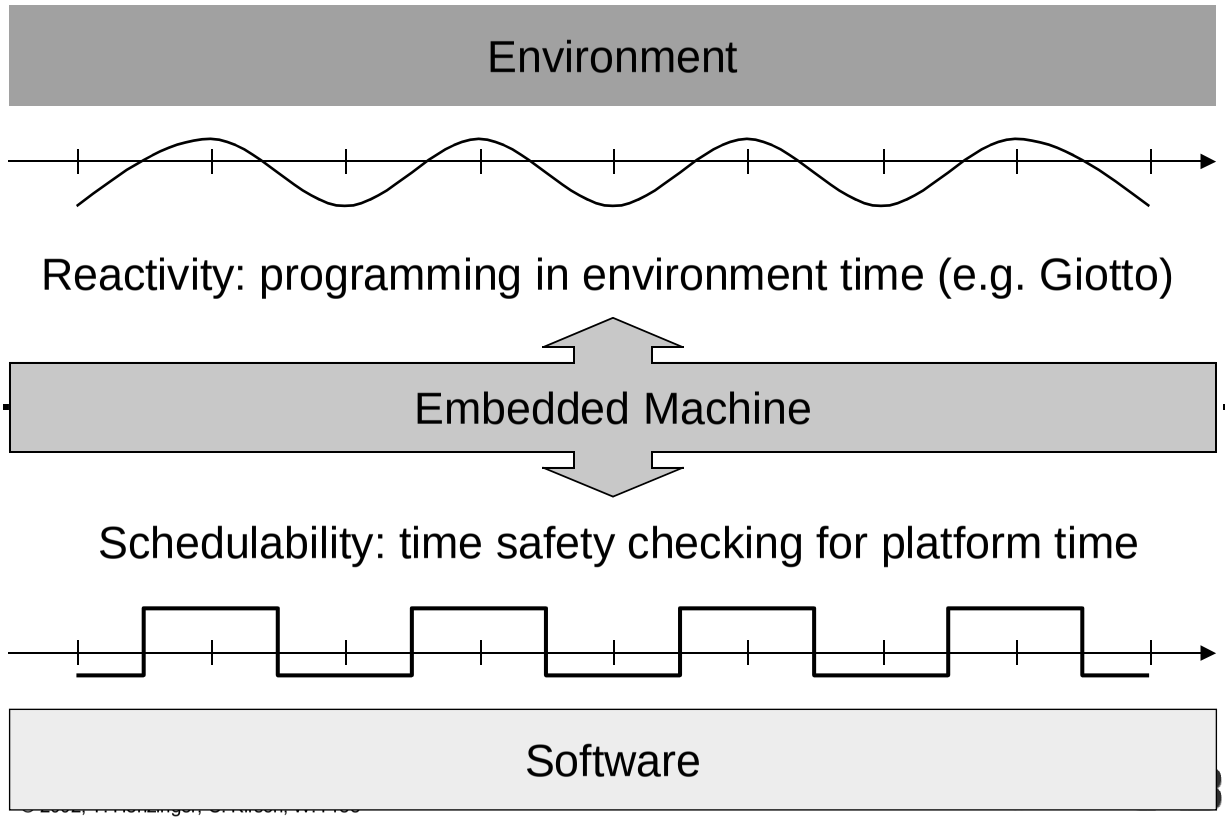
Schedulability



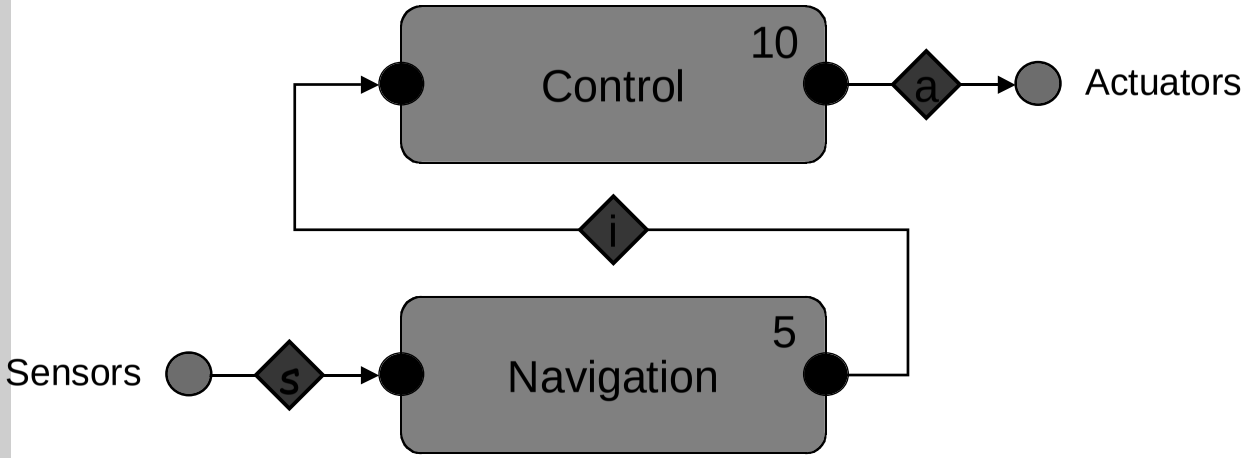
Software

SOFTWARE RESEARCH LAB

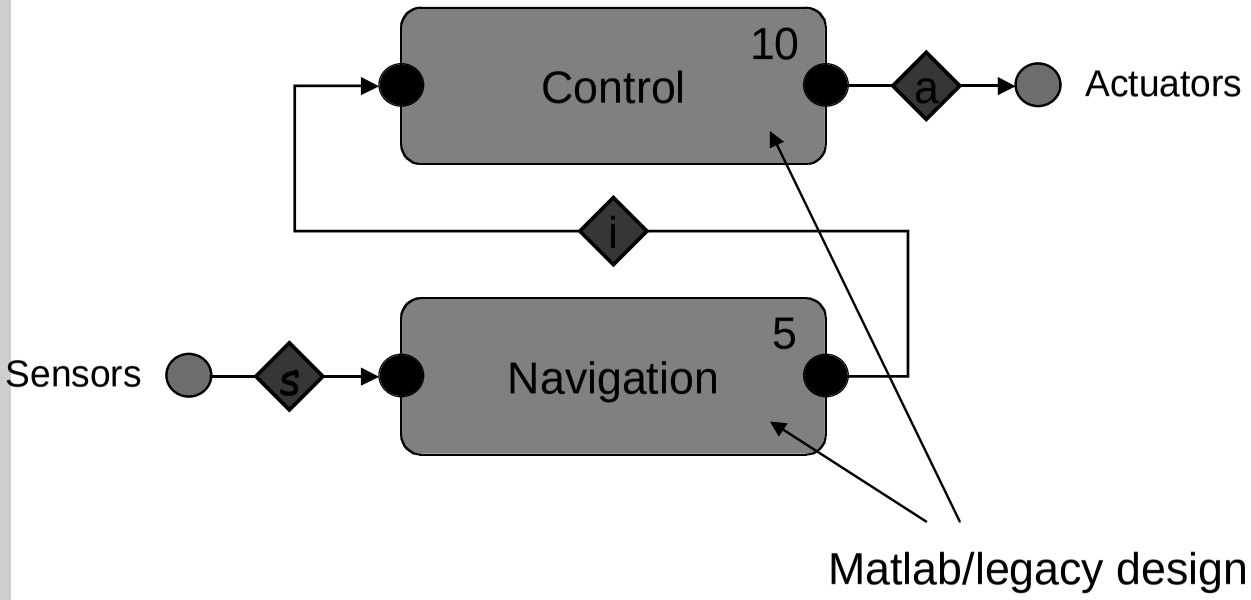
The Embedded Machine: Time is like Memory



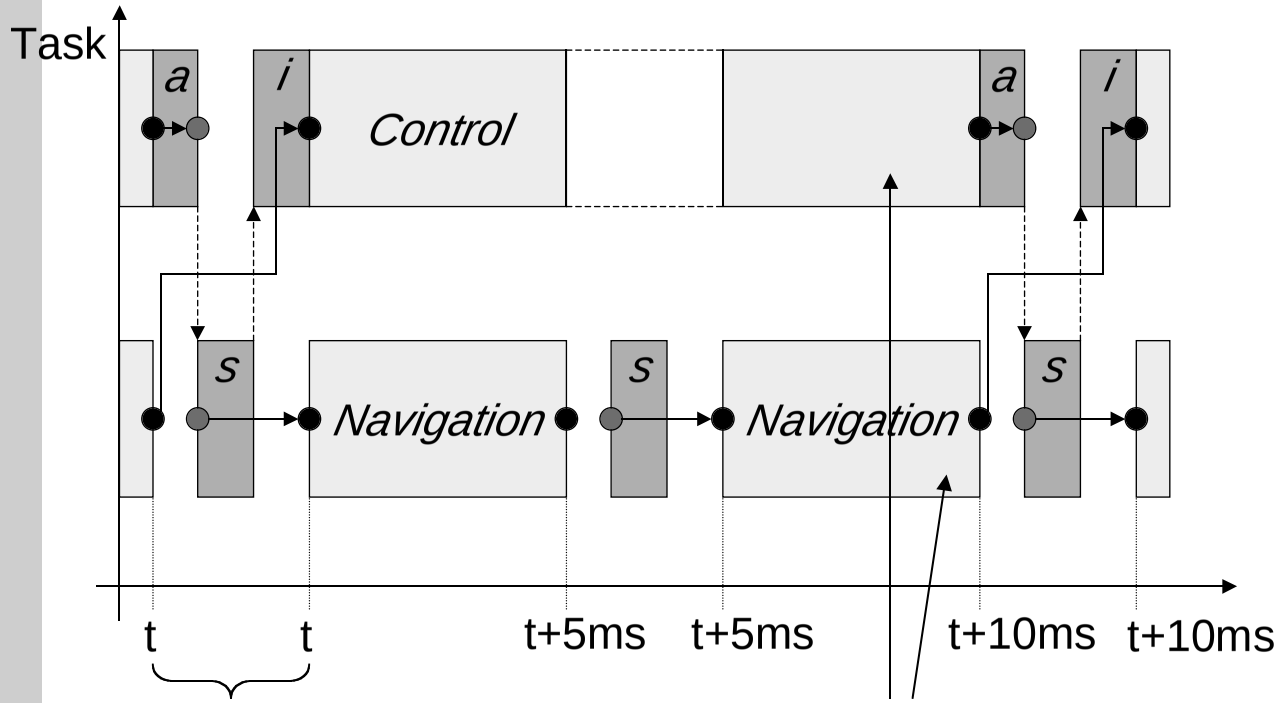
Simplified Helicopter Software



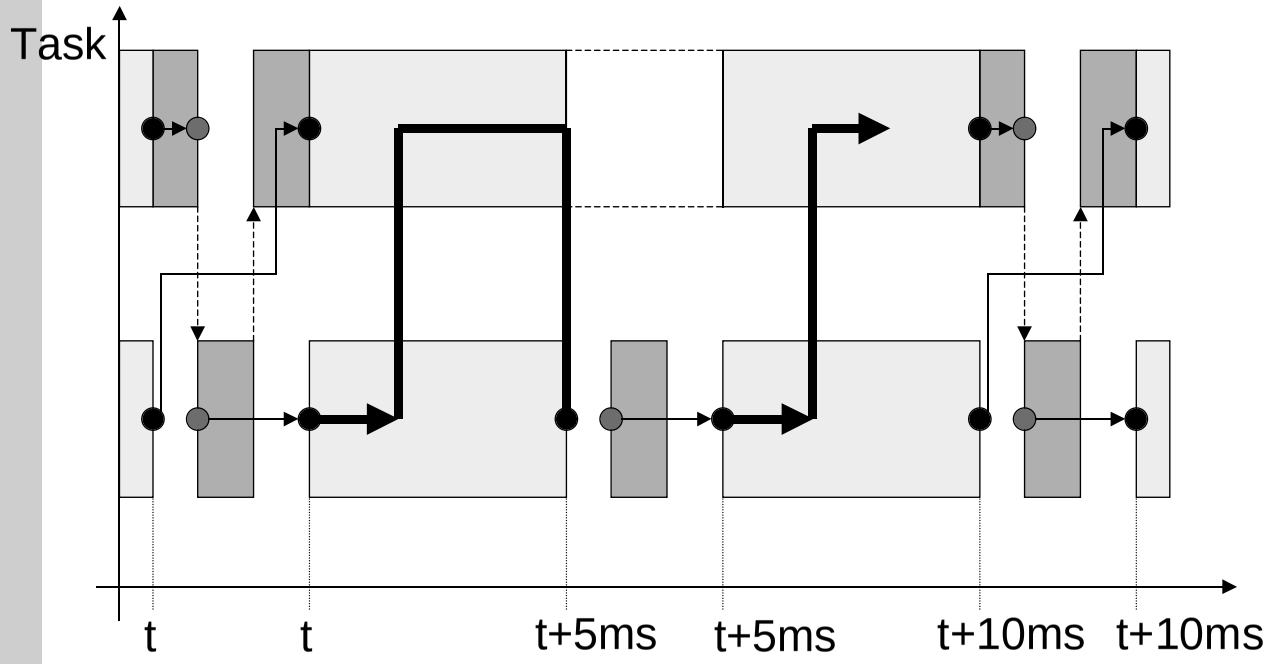
Simplified Helicopter Software



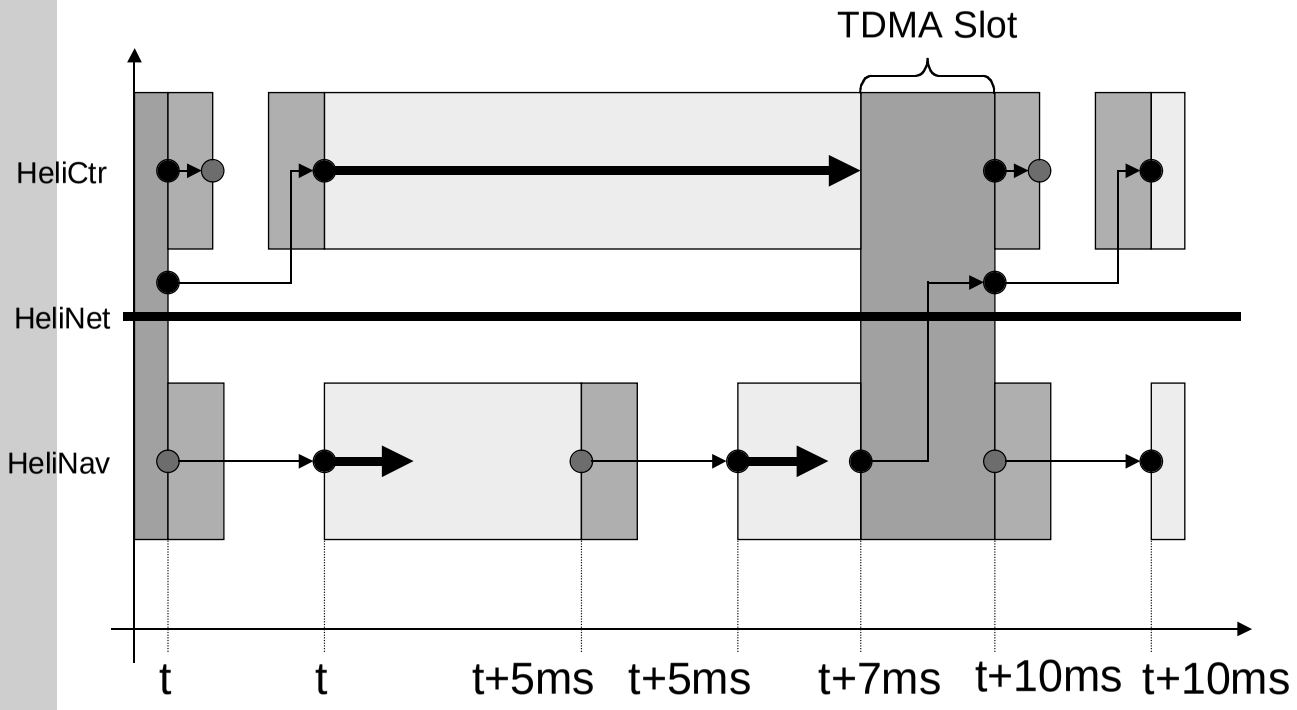
Helicopter Software: Environment Timeline



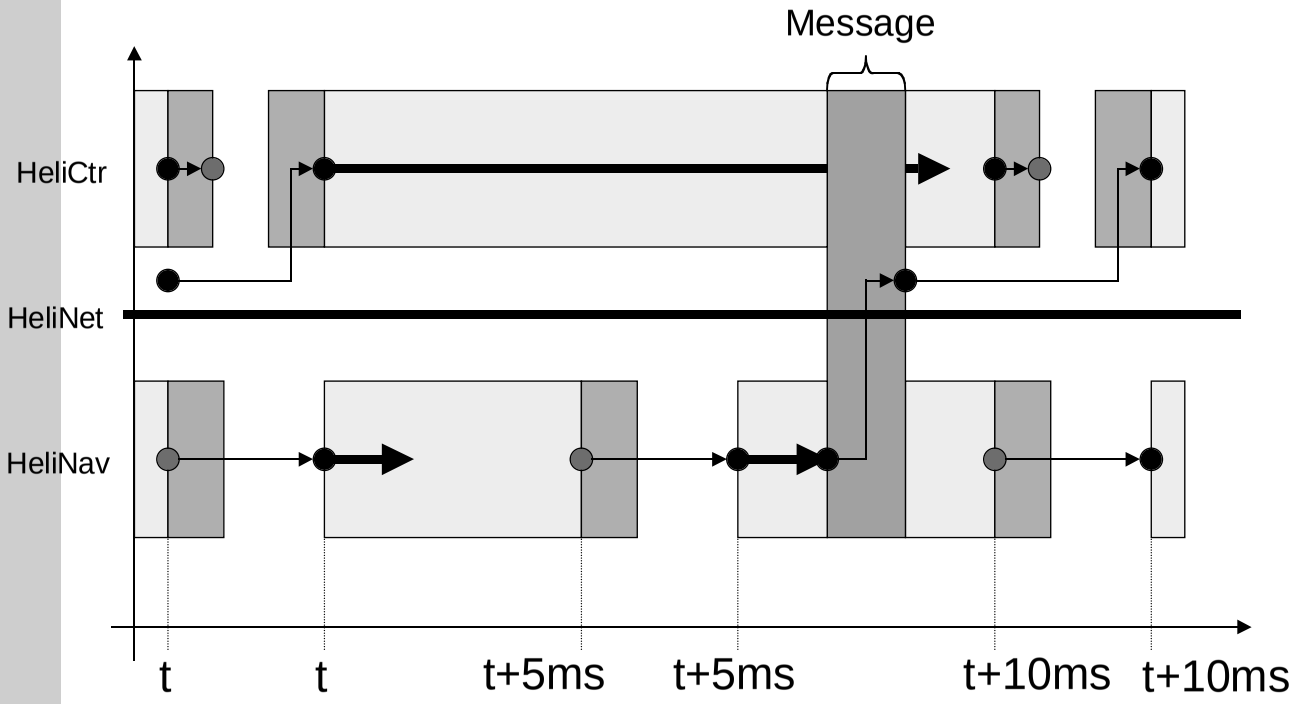
Single-CPU Helicopter: Platform Timeline (EDF)



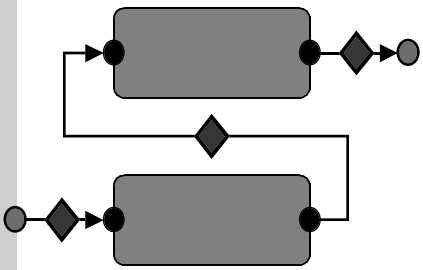
Two-CPU Helicopter: Platform Timeline (Time-triggered Communication)



Two-CPU Helicopter: Platform Timeline (Event-triggered Communication)

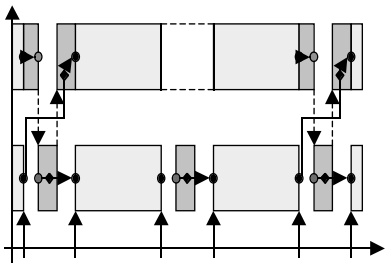


Helicopter Software: Giotto Syntax (Functionality)



```
sensor gps_type GPS uses c_gps_device ;

actuator servo_type Servo := c_servo_init
        uses c_servo_device ;

output

   ctr_type CtrOutput := c_ctr_init ;

   nav_type NavOutput := c_nav_init ;


driver sensing (GPS) output (gps_type gps)
{ c_gps_pre_processing ( GPS, gps ) }


task Navigation (gps_type gps) output (NavOutput)
{ c_matlab_navigation_code ( gps, NavOutput ) }

...
```

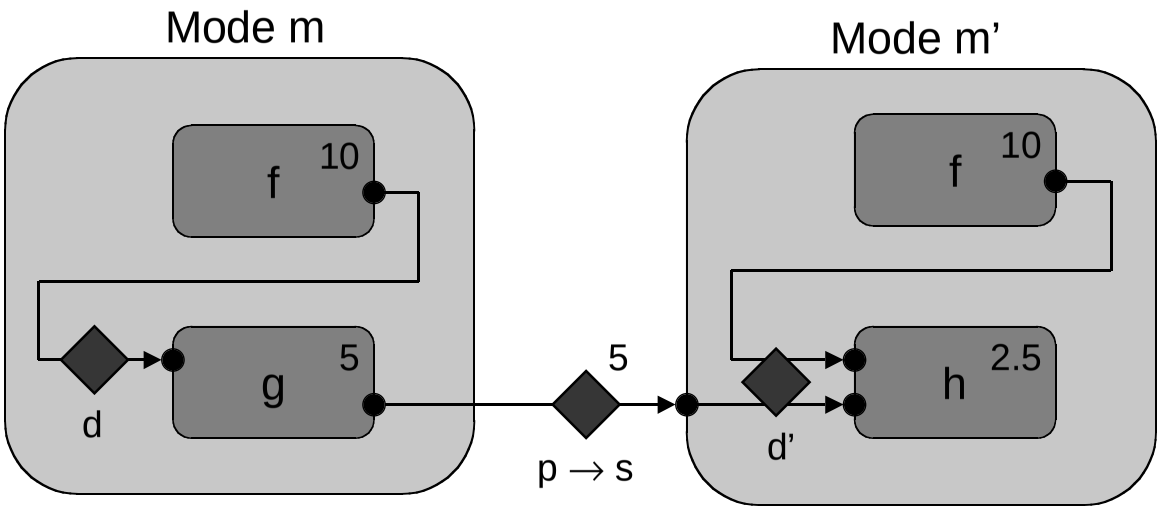
Helicopter Software: Giotto Syntax (Timing)



```
...
mode Flight ( ) period 10ms
  {
      actfreq 1 do Actuator ( actuating ) ;

      taskfreq 1 do Control ( input ) ;
      taskfreq 2 do Navigation ( sensing ) ;
  }
...
```

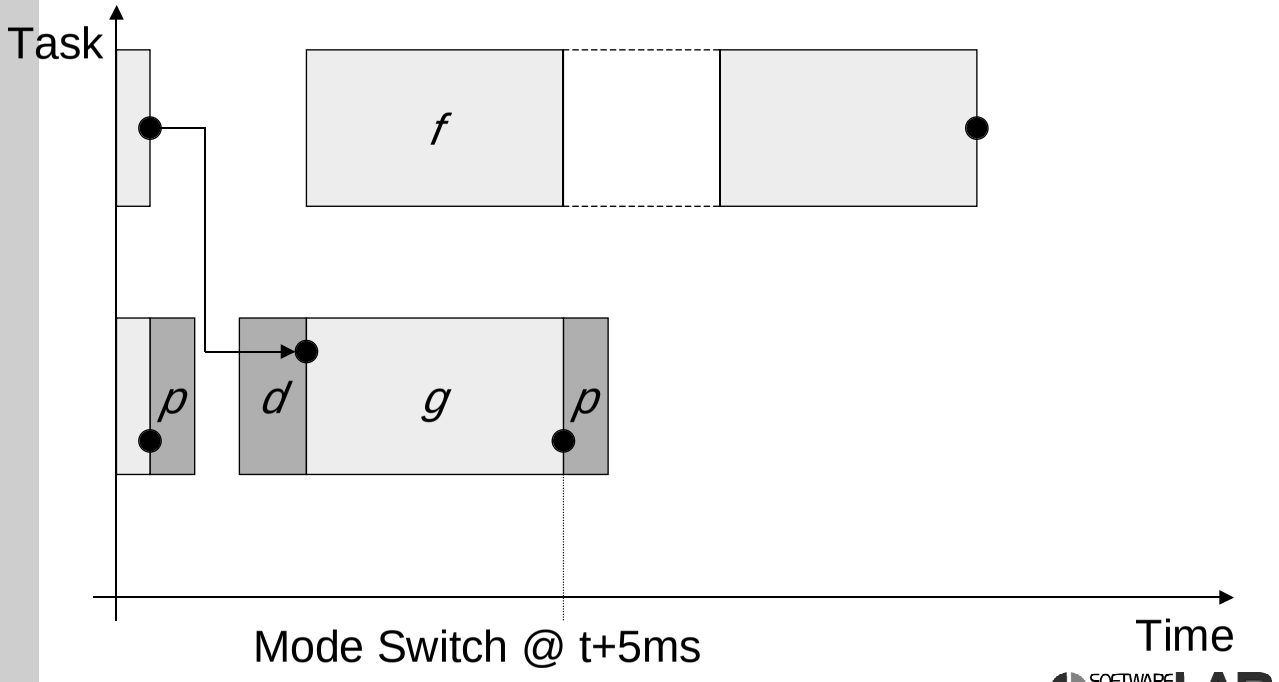

SOFTWARE RESEARCH LAB

Mode Switch

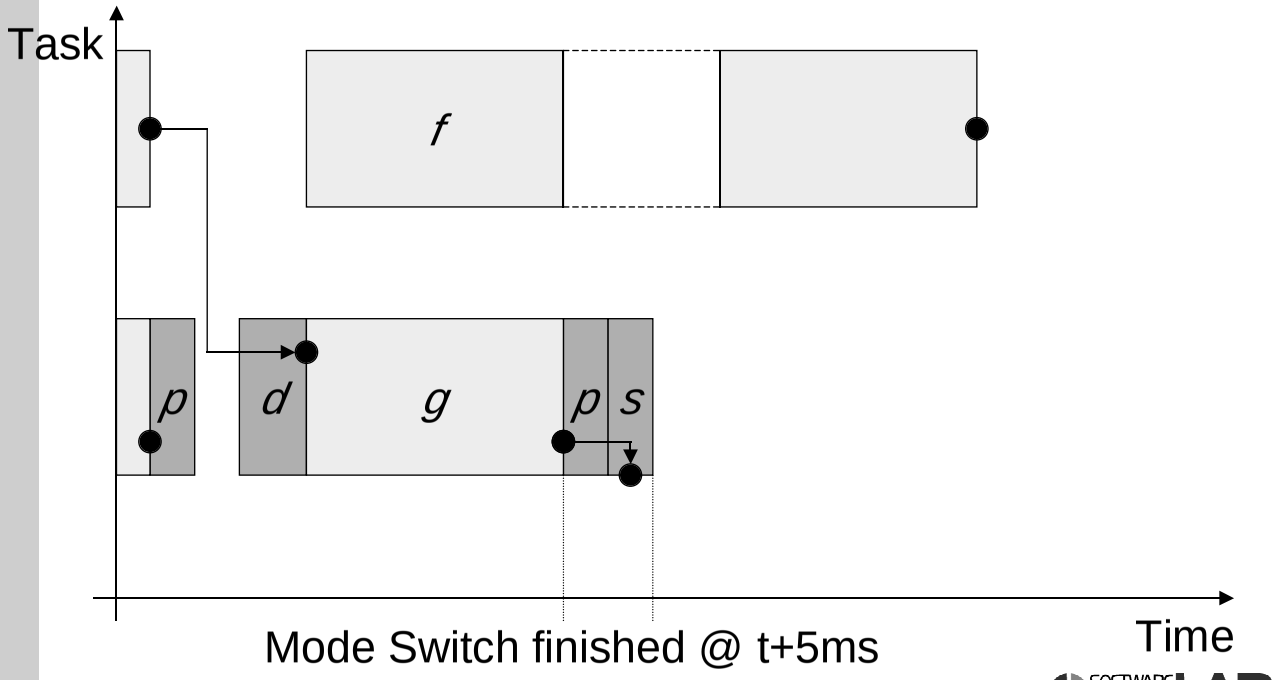


SOFTWARE RESEARCH LAB

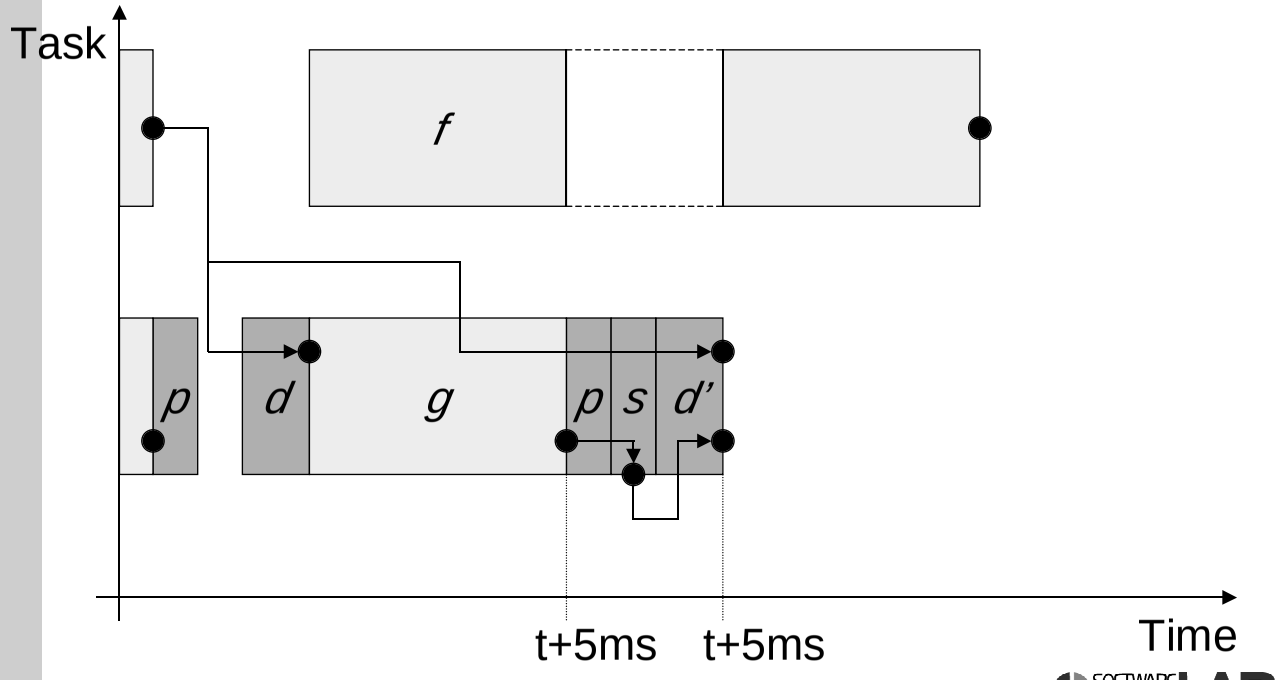
Mode Switch: Environment Timeline



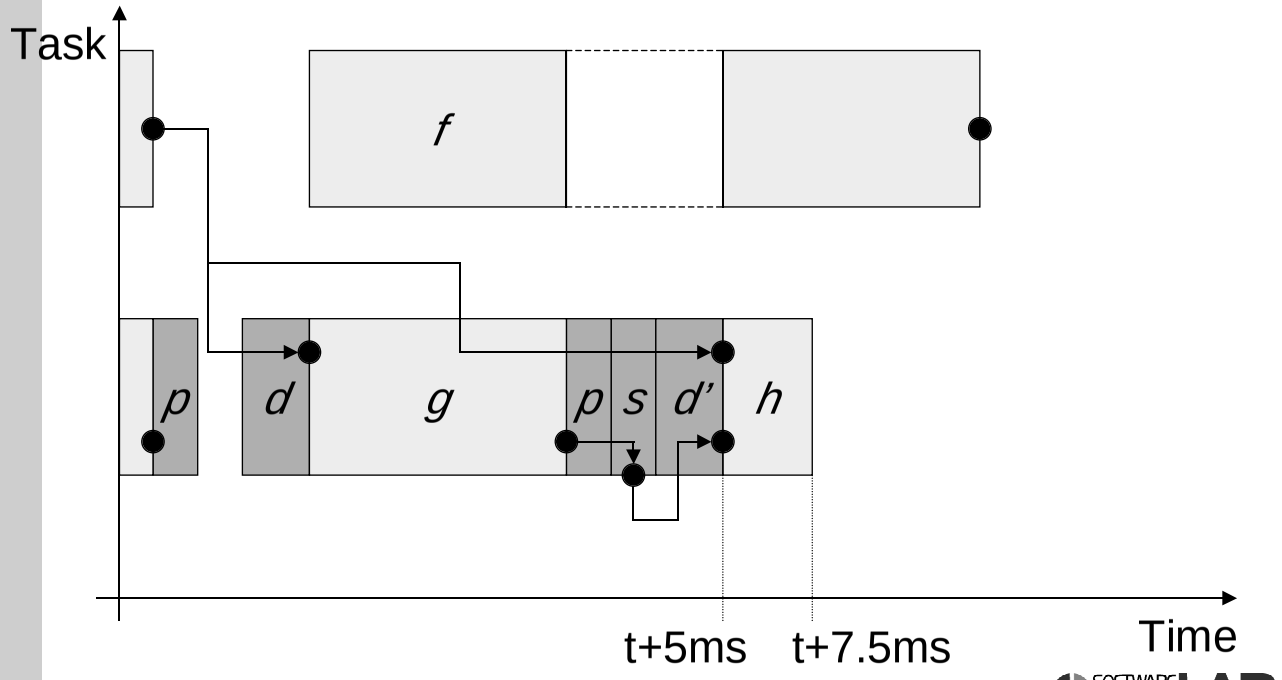
Mode Switch: Environment Timeline



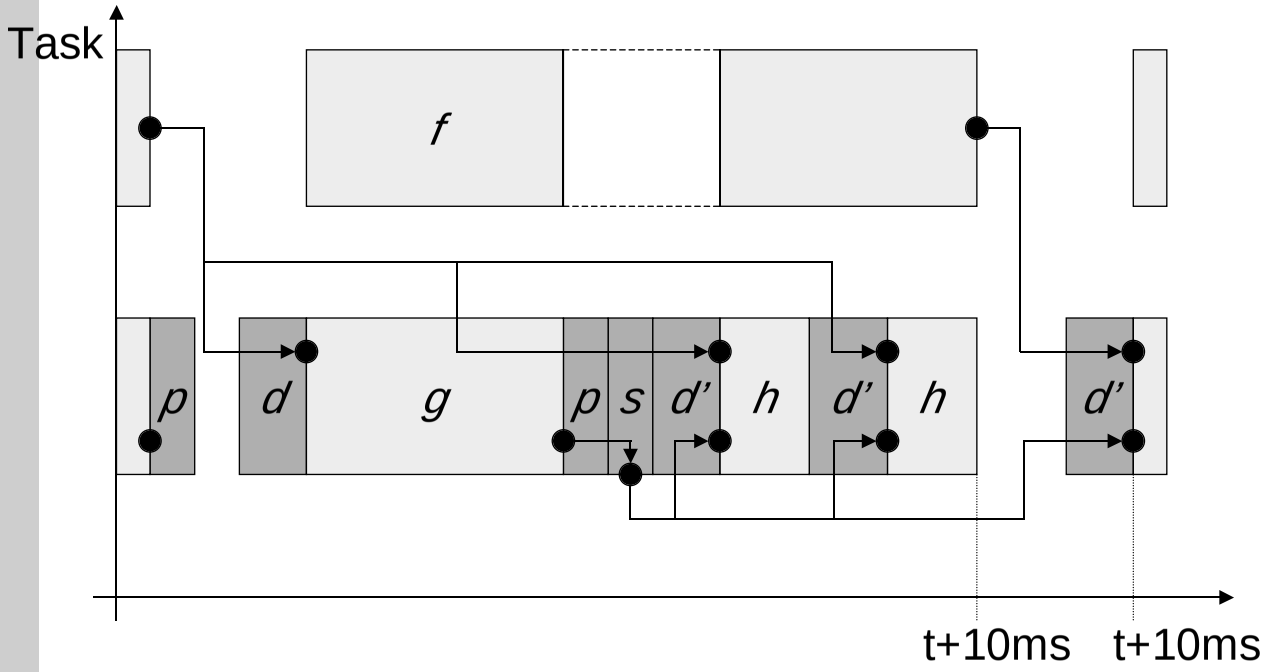
Mode Switch: Environment Timeline



Mode Switch: Environment Timeline



Mode Switch: Environment Timeline



The Giotto Compiler

Functionality — Native Code — Tasks and Drivers

Timing Interaction — Giotto Program

Platform — Giotto-P — Platform Specification

-topology (CPUs, nets)
-performance (WCETs, latencies)
-APIs (RTOSs, protocols)

Giotto Compiler

Executables

or

"Failure"

either Giotto-P overconstrained, or compiler not smart enough (distributed scheduling problem)

Closing the Gap: Annotated Giotto

Functionality — Native Code — Tasks and Drivers

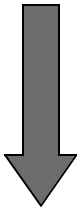
Timing Interaction — Giotto Program

Platform — Giotto-P

Map — Giotto-PM

-topology (CPUs, nets)
-performance (WCETs, latencies)
-APIs (RTOSs, protocols)
-assign tasks to CPUs
-assign connections to nets

Giotto Compiler



Executables or "Failure"

either Giotto-PM overconstrained, or compiler not smart enough (global scheduling problem)

SOFTWARE RESEARCH LAB

Closing the Gap: Annotated Giotto

Functionality — Native Code — Tasks and Drivers

Timing Interaction — Giotto Program

Platform — Giotto-P

-topology (CPUs, nets)
-performance (WCETs, latencies)
-APIs (RTOSs, protocols)

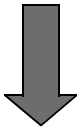
Map — Giotto-PM

-assign tasks to CPUs
-assign connections to nets

Communication — Giotto-PMC

-assign connections to TDMA slots (say)

Giotto Compiler



Executables or "Failure"

Giotto-PMC overconstrained (local scheduling problems solvable)

SOFTWARE RESEARCH LAB

Single-CPU Giotto Scheduling

Why is it simple?

-Static utilization test for each mode
-Mode switches are memory-free

Theorem: Given a Giotto program and WCETs for all tasks, it can be checked in quadratic time if an EDF scheduler meets all deadlines.

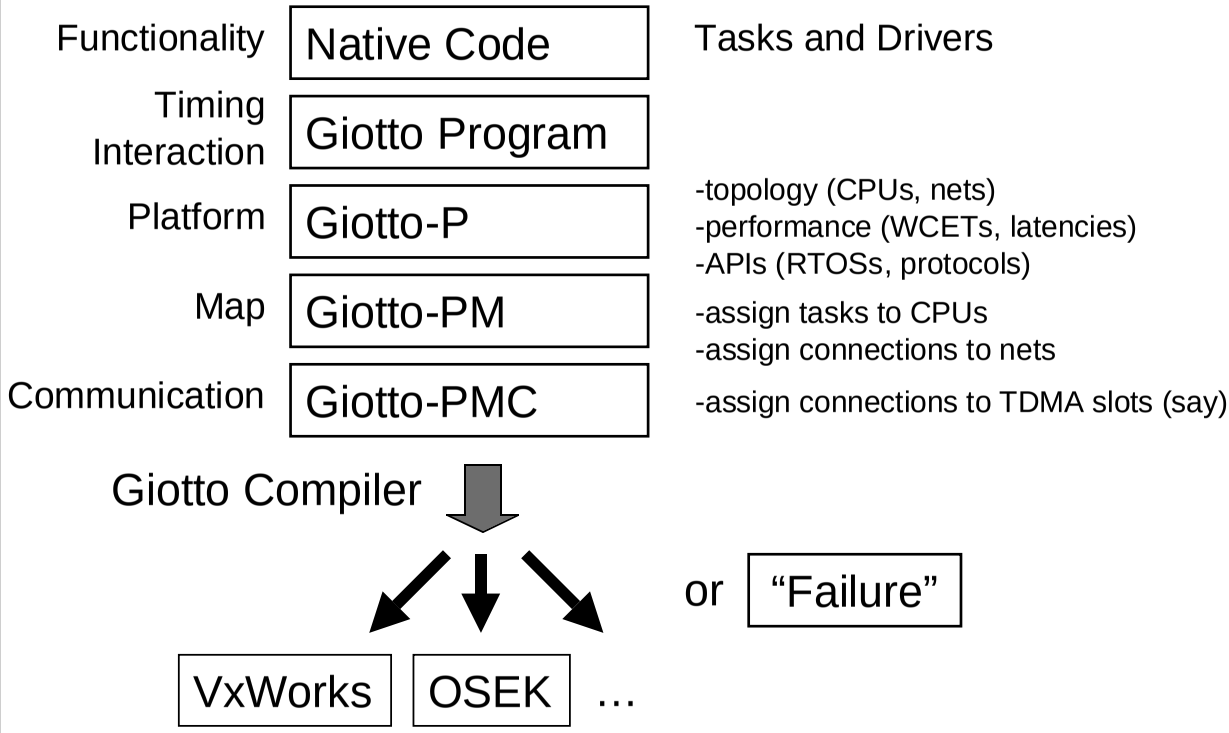
SOFTWARE RESEARCH LAB

Two-CPU Helicopter: Annotated Giotto (Time-triggered Communication)

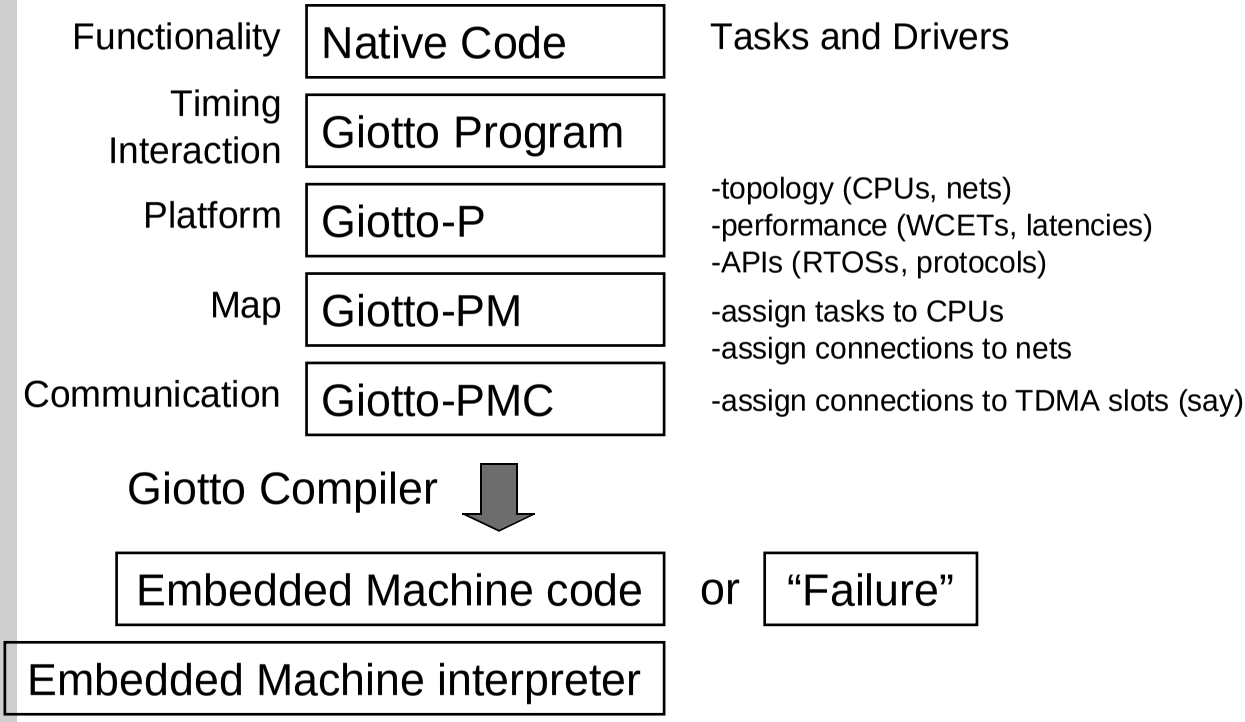
```
[ host HeliCtr address 192.168.0.1;
  host HeliNav address 192.168.0.2;
  network HeliNet address 192.168.0.0 connects HeliCtr, HeliNav ]
...
mode Flight ( ) period 10ms
  {
      actfreq 1 do Actuator ( actuating ) ;

      taskfreq 1 do Control ( input ) [ host HeliCtr ] ;
      taskfreq 2 do Navigation ( sensing ) [host HeliNav;
      push ( NavOutput ) to ( HeliCtr ) in HeliNet slots (7,10) ] ;
  }
...
```


SOFTWARE RESEARCH LAB

Code Generation



Code Generation: The Embedded Machine



The Embedded Machine

-a virtual machine that mediates the interaction of physical processes (sensors and actuators) and software processes (tasks and drivers) in real time

-the Giotto compiler can be retargeted to a new platform by porting the Embedded Machine

SOFTWARE RESEARCH LAB

The Giotto Project

Completed: **`www.eecs.berkeley.edu/~tah/giotto`**

-Software tools:

Simulink to Giotto translator, Part I (Kirsch, Pree, Stieglbauer)

Giotto compiler for single-CPU targets (Kirsch)

Embedded Machine for Linux and VxWorks (Kirsch, Pree)

-Applications:

Lego Mindstorms (Horowitz, Kirsch, Majumdar)

Zurich helicopter (Kirsch, Sanvido, Pree)

In progress:

-Software tools:

Giotto scheduler for distributed platforms (Horowitz)

Simulink to Giotto translator, Part II (Pree, Stieglbauer, Kirsch)

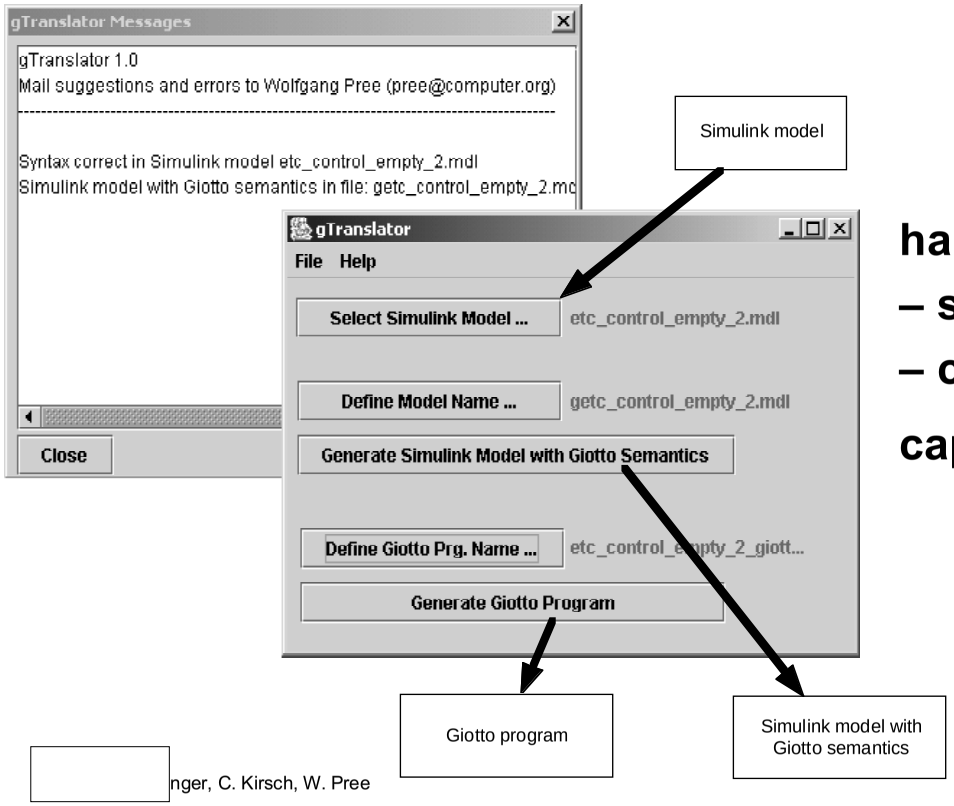
Time safety checker for Embedded Machine code (Kirsch, Matic)

-Applications:

Electronic throttle control (Pree, Stieglbauer, Kirsch)

SOFTWARE RESEARCH LAB

gTranslator tool: seamless integration of Giotto and Simulink



harnesses SL's – simulation and – code gen. capabilities

nger, C. Kirsch, W. Pree

SOFTWARE RESEARCH LAB

Framework components (ESA)

SOFTWARE RESEARCH LAB

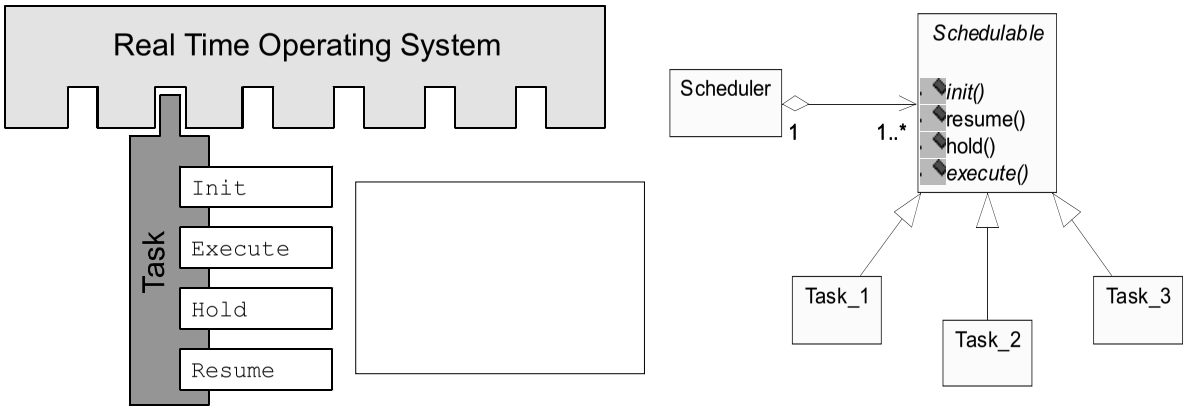
Attitude and Orbit Control System

The AOCS is typically the most complex satellite subsystem

SOFTWARE RESEARCH LAB

Reuse in a RT-OS

RTOS's are examples of reuse in the RT field
→ inspiration for AOCS f/w



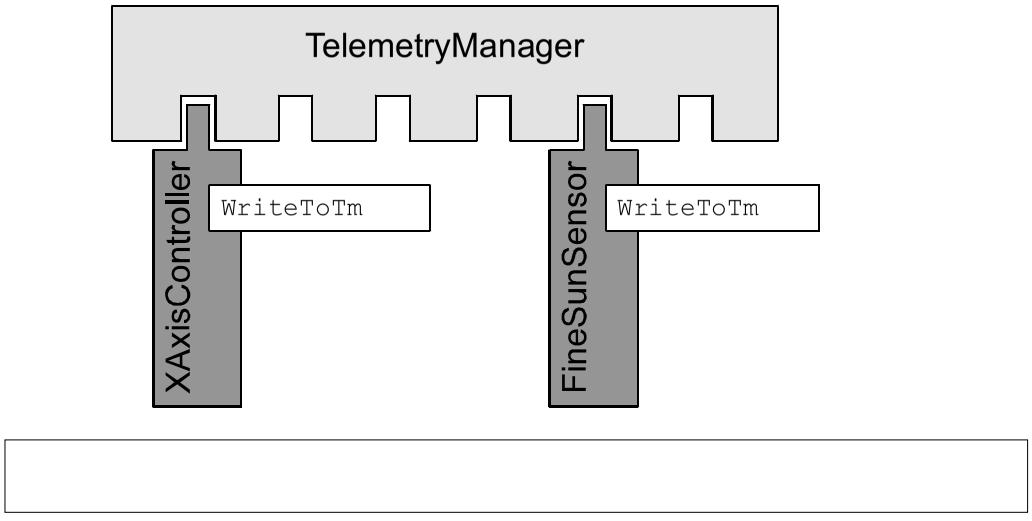
Task *management* is separated from task *implementation* through an abstract class/interface

SOFTWARE RESEARCH LAB

Telemetry Management

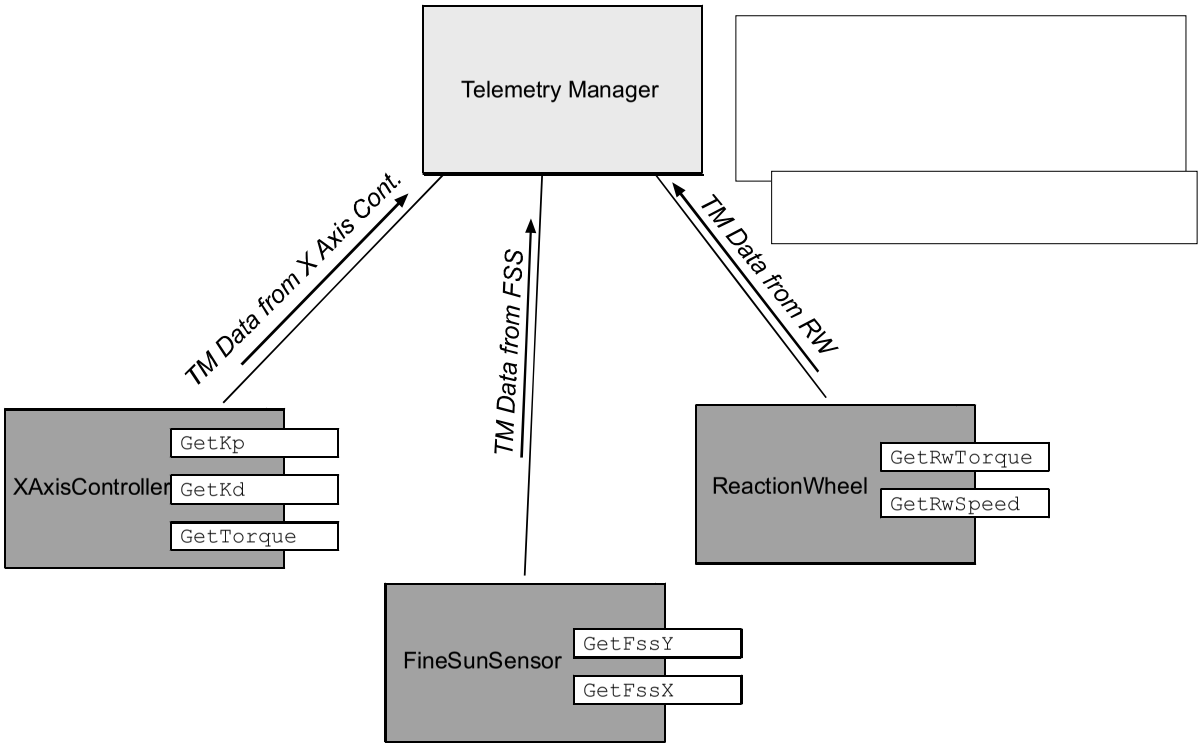
- AOCS periodically sends telemetry (TM) to the ground representing (a subset of) the state of the AOCS objects
- Problem: how can the management of the telemetry functionality be decoupled from the format and content of the telemetry data?
  - Note: We want to design a generic framework, not just a single application
- Proposed solution can be applied to any embedded system where housekeeping data need to be logged on a periodic basis

TM Manager – schematic view




```
Telemeterable* list[N];
TmStream* tmStream;
. . .
for ( i=0; i<N; i++ )
     list[i]->writeToTelemetry(tmStream);
```

Conventional, non-reusable solution



SOFTWARE RESEARCH LAB

The end

Thank you for your attention!

SOFTWARE RESEARCH LAB