Oberon-based Autopilots for Unmanned Aerial Vehicles

Jacques Chapuis

weControl AG
Air Force Center
Überlandstrasse 255
CH-8600 Dübendorf
http://www.wecontrol.ch

Outline

I. Introduction to weControl

II. Autopilot Basics

III. Products/Hardware

IV. Software Design with Oberon

V. Realised Project Examples

I. weControl AG

Formation:	16. May 2000
Structure:	Incorporated (since 1.1.2006)
Founders:	Chapuis/Eck/Kottmann/Sanvido
Chief Executive:	J. Chapuis
Innovative Force:	5 employees
Turnover (2006):	~ 1 M CHF

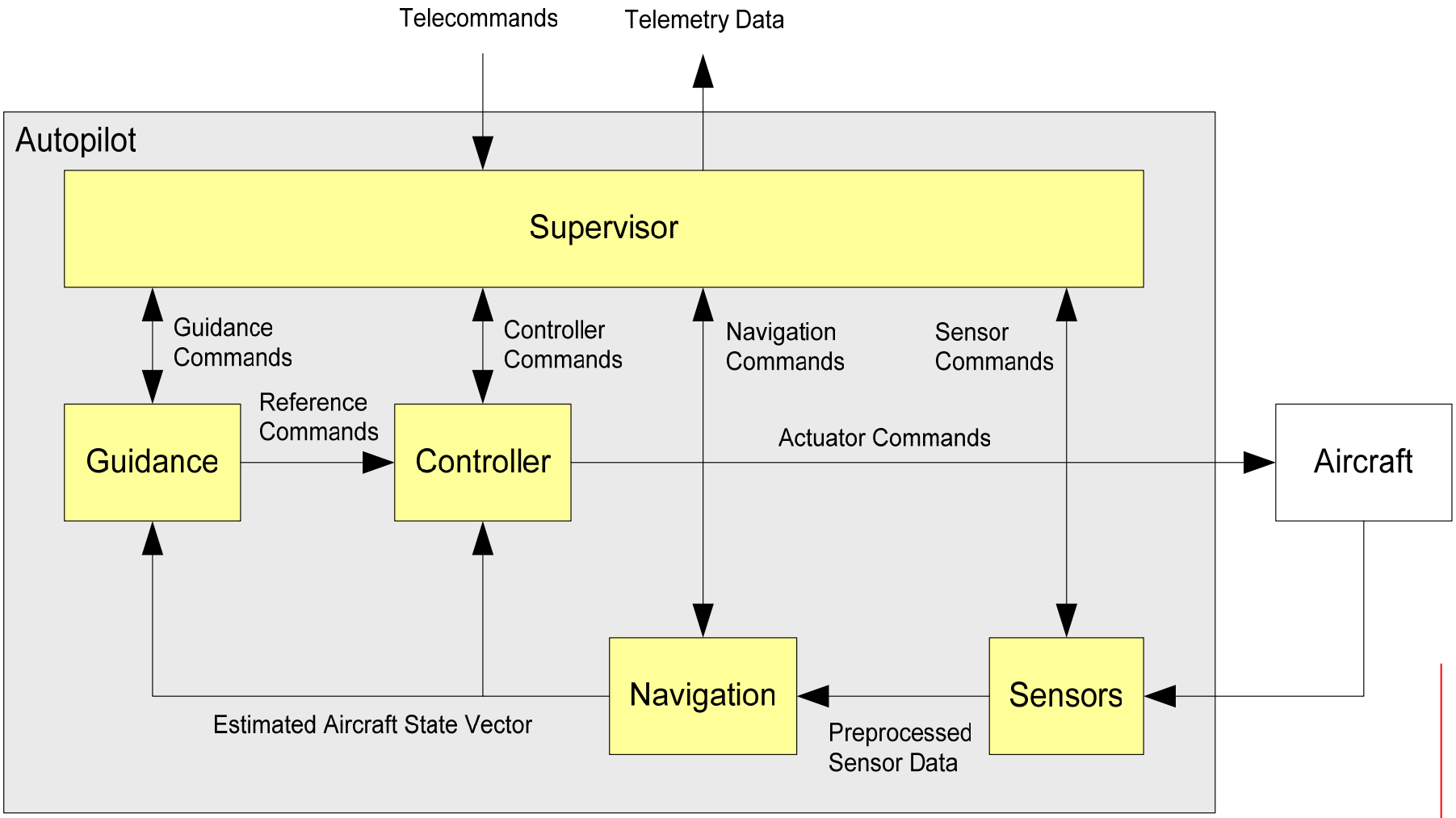
Mission:

weControl provides customized solutions to increase the level of autonomy of miniature unmanned aerial vehicles by combining modern system theory, sensor technologies and state of the art embedded computing

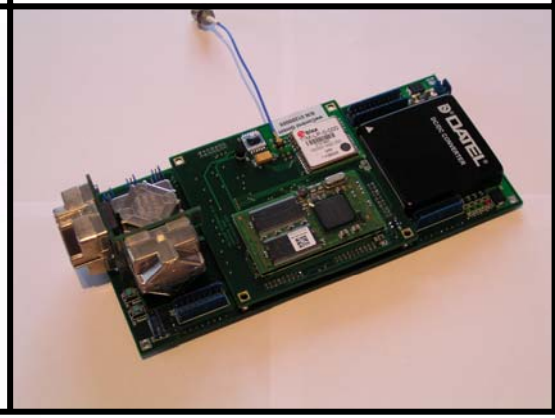
II. Autopilot

- High-level command control: airspeed/course/altitude
- Waypoint guidance
- Geometrical figures
- Automatic launch (hand/bungee/catapult)
- Precision landing (“ILS”)
- Homing in case of data-link loss
- Robust navigation in case of GPS outages
- Flight data logger
- Setup procedure: Collect data, parameter identification, controller synthesis & validation

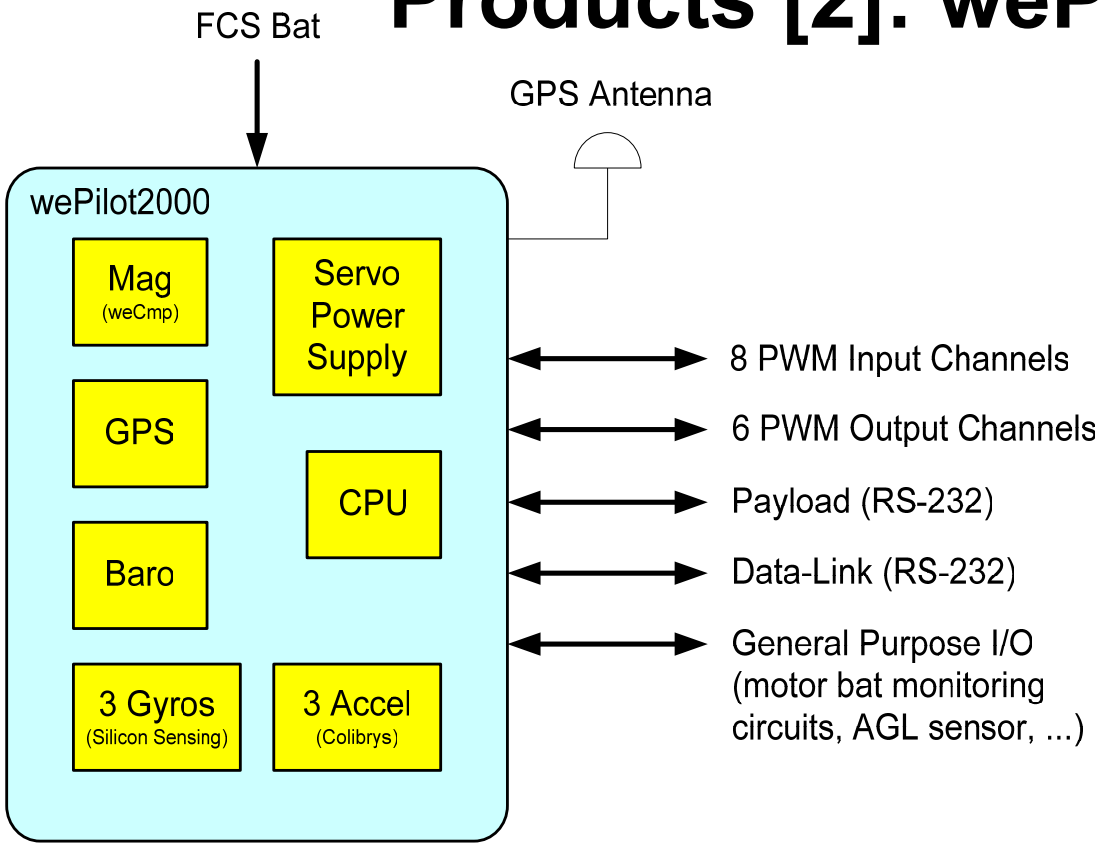
Autopilot [2]

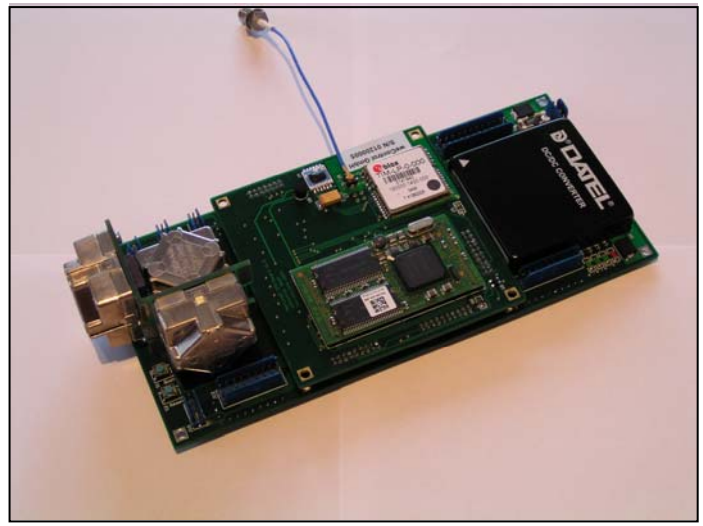


III. Products

wePilot1000 Rotary-wing Aircraft	**wePilot2000** Fixed-wing Aircraft	**wePilot4RMAX** Yamaha RMAX Helicopter
		
• 13 in D/F/NL/CH • 8 in USA • 1 in Asia	• 30 in F	• 1 in Asia • 2 in USA

Products [2]: wePilot2000





- Designed for fixed-wing UAVs
- Integrated 6-DOF IMU
- Total weight: 260 g
- Total power consumption: 250 mA @ 12 VDC
- 30 Watt power supply for servos
- 32-Bit Intel XScale PXA250 CPU @ 400 MHz

IV. Software Development with Oberon

1. Oberon at weControl
2. Design Methodologies
3. Coding Rules
4. Testing

1. Oberon at weControl

- Development of a first embedded FCS (OLGA,1998)
  - Dedicated computer system
  - Oberon compiler for StrongARM CPU
  - Development of Operating System HelyOS

- wePilot1000:
  - Miniaturized CPU-board based on StrongARM
  - Use of Oberon is a logical choice
  - Migration to XScale CPU
  - Adaptation of HelyOS

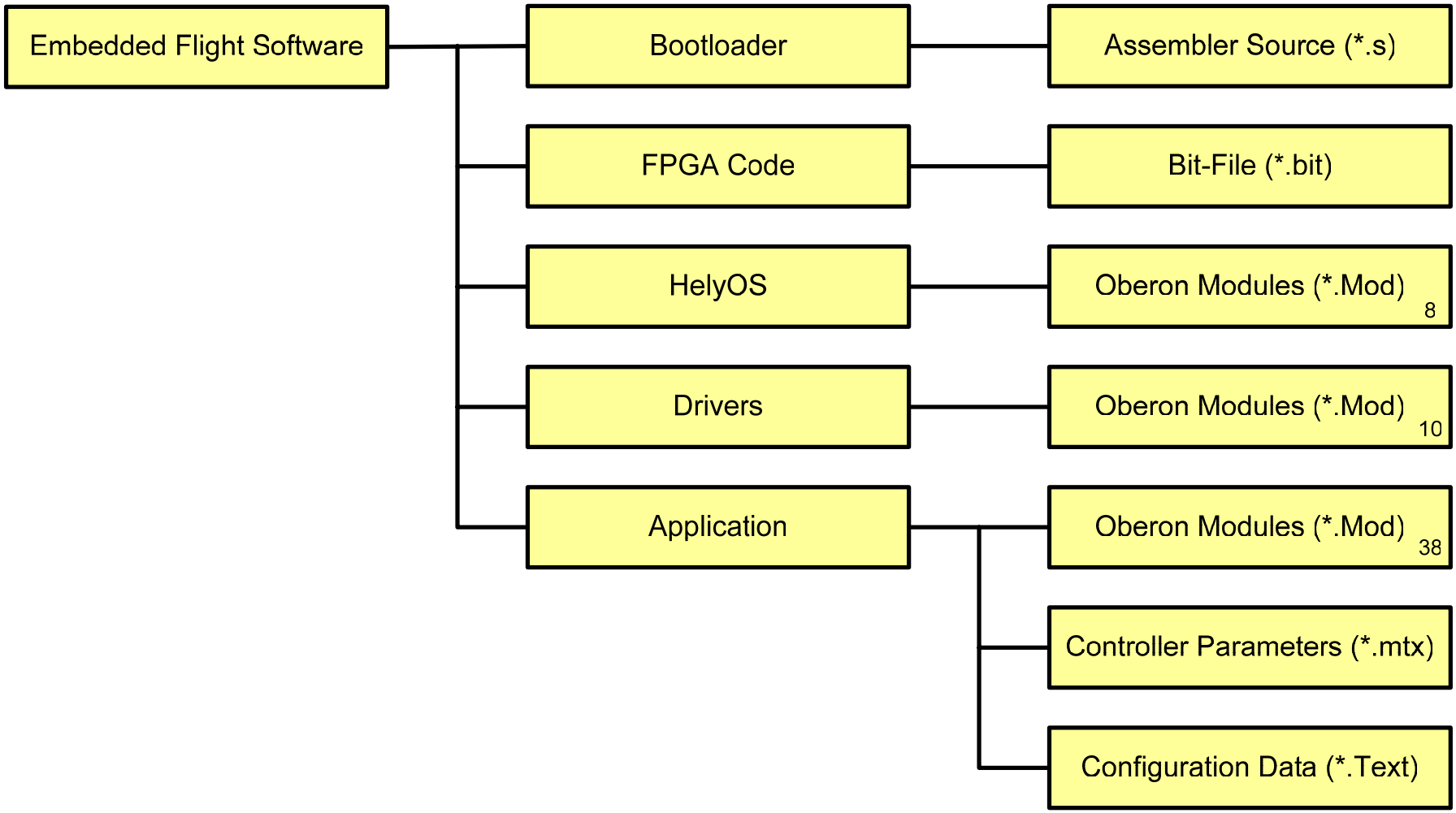
- Newer developments:
  - wePilot2000, wePilot4RMAX

→ Oberon grew with weControl, weControl grew with Oberon

Features and Advantages

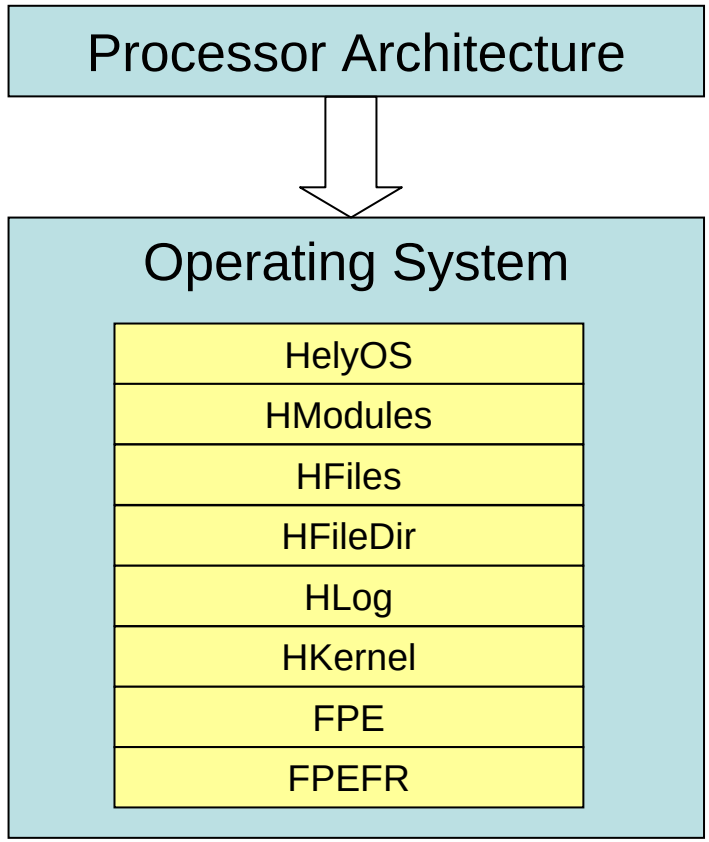
- Rigorously structured
- Efficiently implemented
- Allows systematic approach to the design of programs
- Prerequisite for modular design
- Oberon “programs” are *collection of commands,* which are procedures exported by modules
  - no monolithic executable
- „Safe“ language:
  - Boundary checking
  - Strong type checking
- Single pass compiler
- Lean system: compiler 2350 lines
- No blackbox code in application

Flight Software Structure



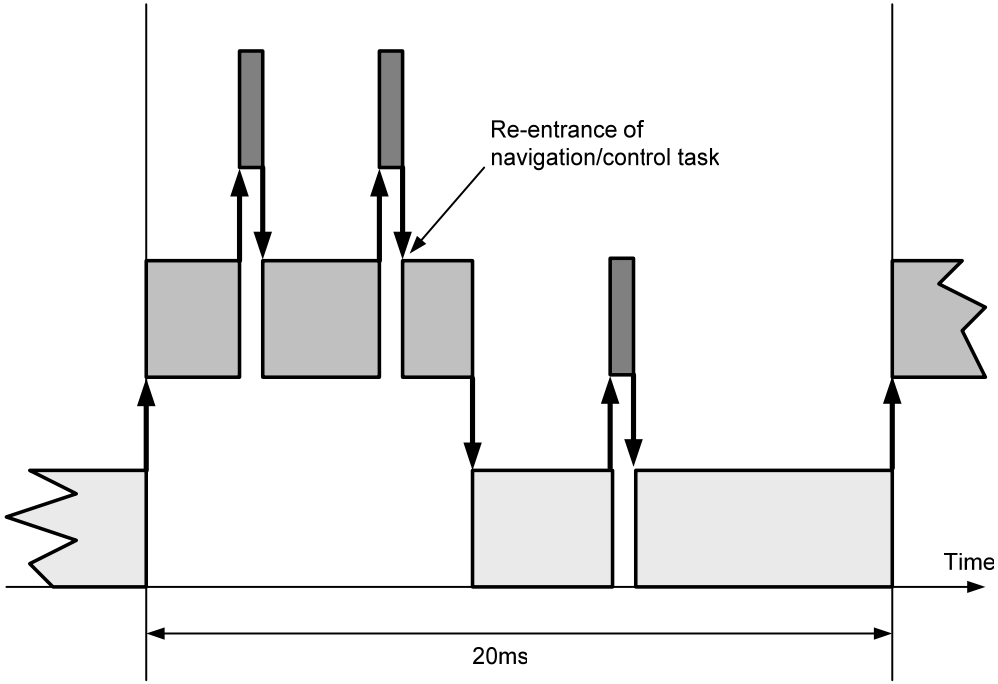
2. HelyOS [1]

- Initially developed by M. Sanvido*
- Highly customized, tailored and tied to HW
  - Initially StrongARM, later adapted for XScale
- Designed for real-time control applications
- Compact (59kB)
- Written in Oberon
- 8 Modules



** M. Sanvido, "A Computer System for ModelHelicopter Flight Control: The Software Core", Technical Report #317, Institute for Computer Systems, ETH, 1999*

HelyOS [2]: Scheduler Scheme



Interrupts	Started tasks	Shown as
IRQ (asynch)	Receive/ transmit char	
IRQ (20ms)	Synchronous tasks	
	Background tasks	

3. Design Methodologies

- Hierarchical Decomposition
- Software as a Dynamical System
- Hardware-in-the-Loop Simulation
- Algorithm Design with Matlab
- Draw upon Experience

Hierarchical Decomposition

Extensible design and *complexity reduction* through system decomposition:

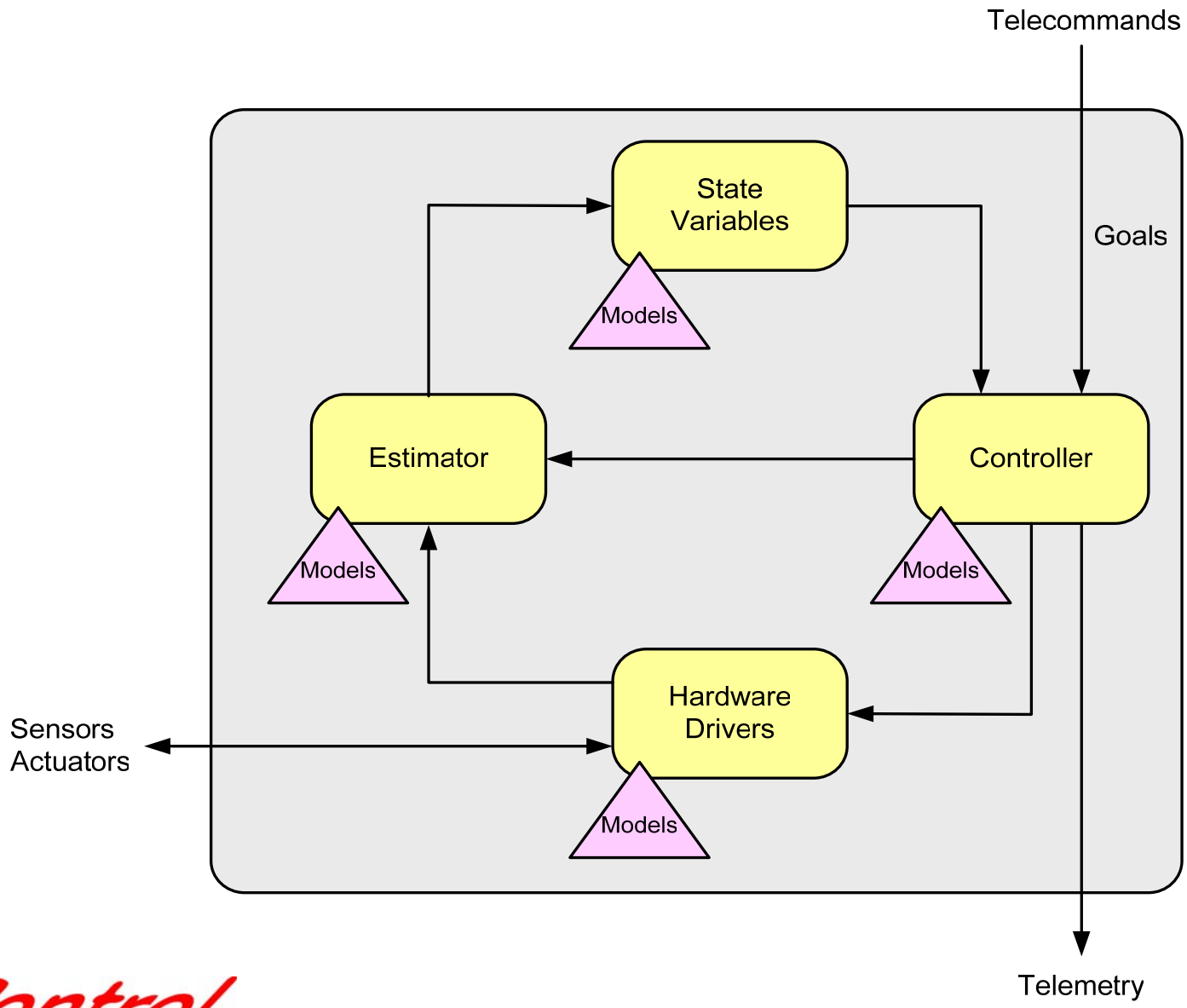
system	=	application
subsystem	=	module
hierarchical dependence	=	module hierarchy (import list)
subsystem interaction	=	information flow (procedure calls, events)
subsystem boundary	=	definition module
system view	=	part of application - navigation - control - telecommand - telemetry - simulator

Result:

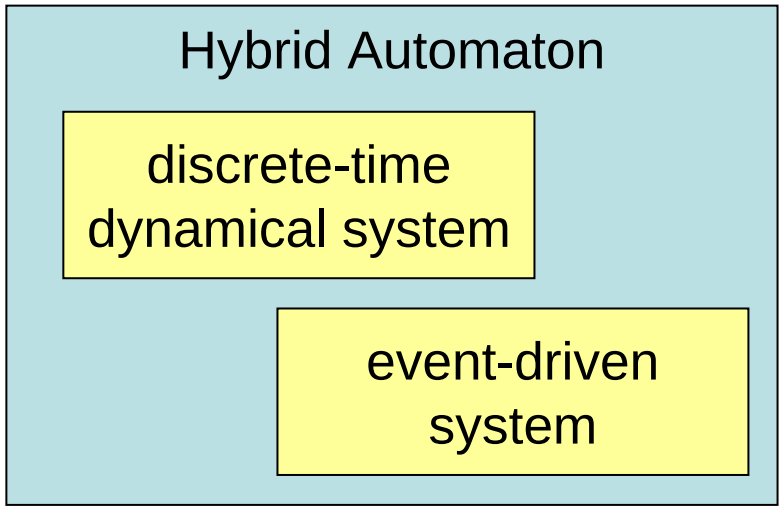
⇒ module hierarchy
⇒ data flow
⇒ control diagrams

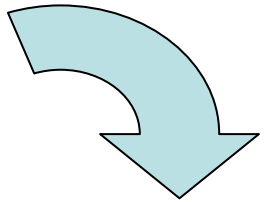
→ ***Software Structure***

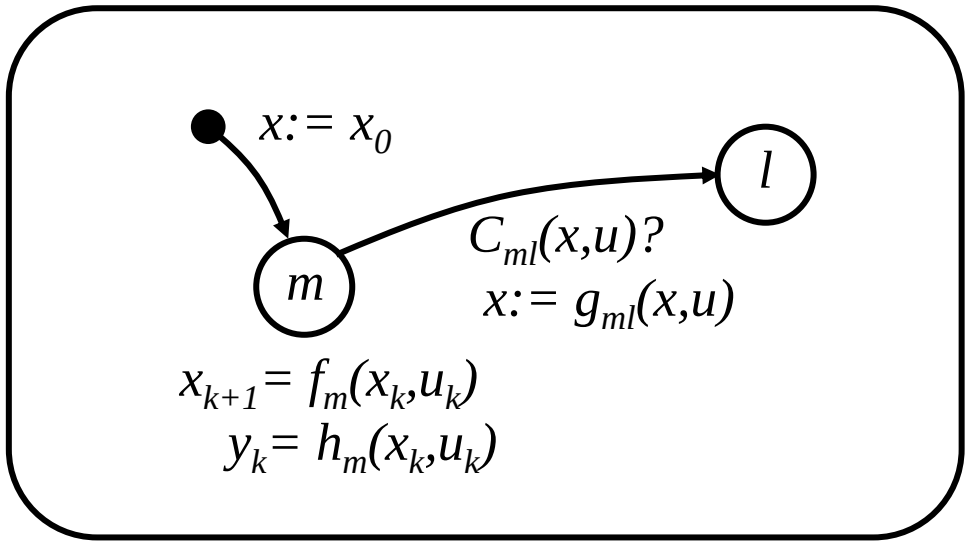
Software as a Dynamical System



Hybrid Control Framework



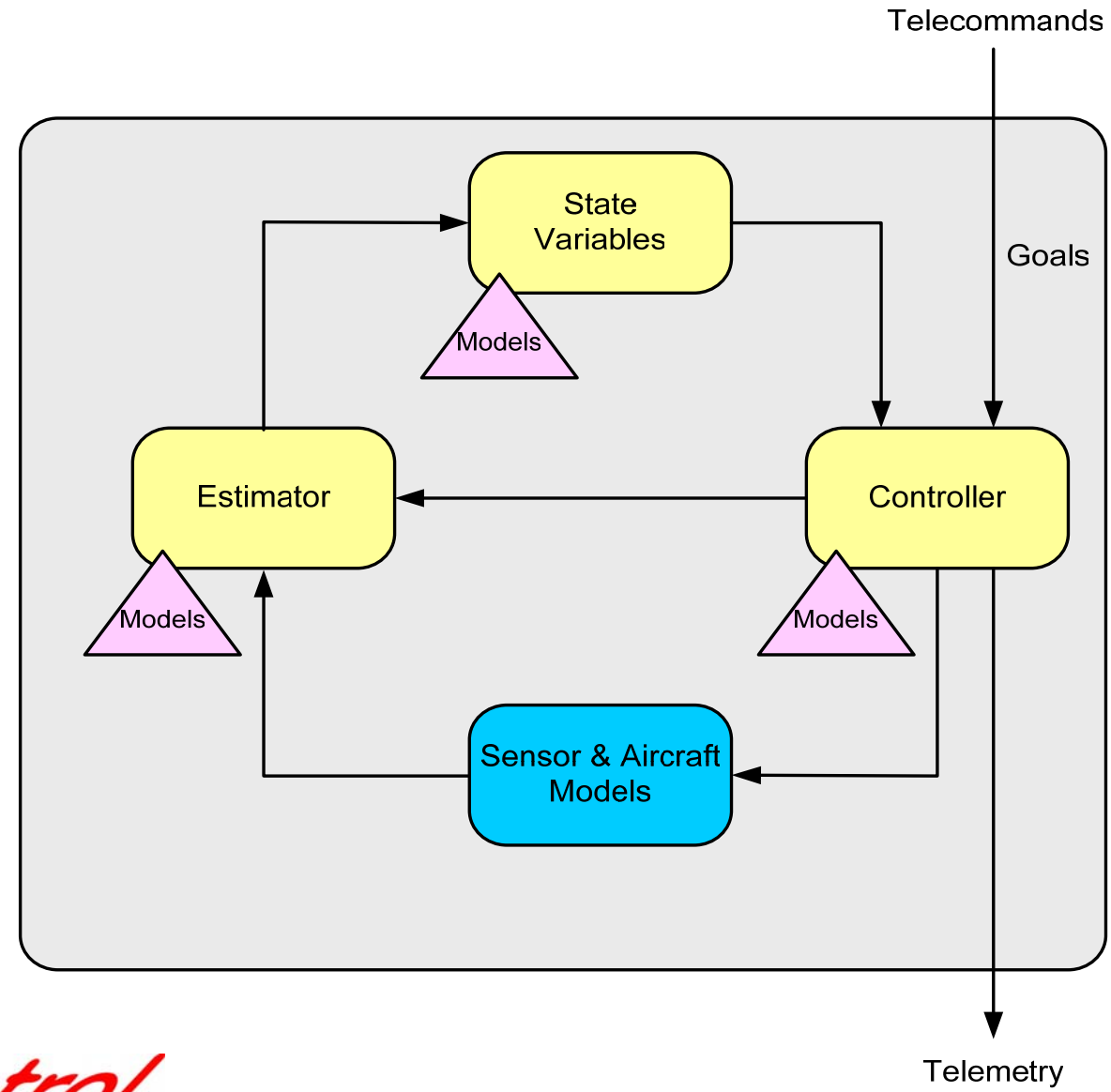




→ ***Time Flow***

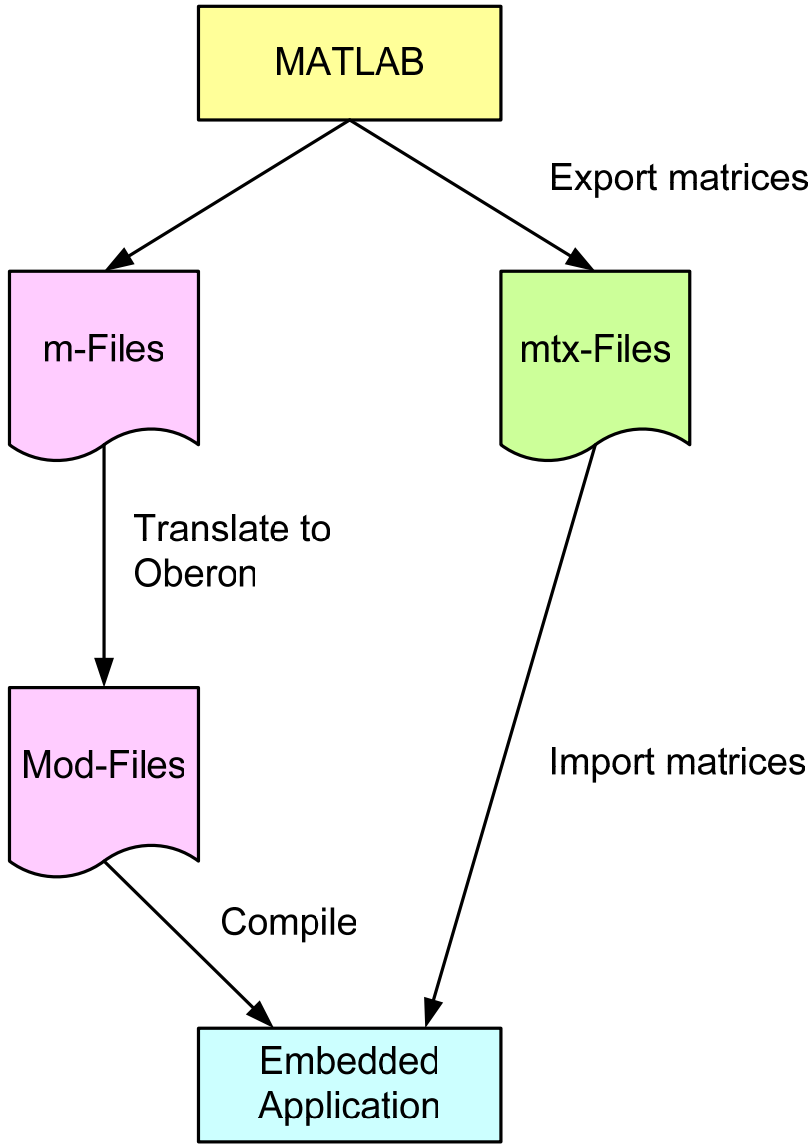
→ *Finite state machine (Module FSMs.Mod)*

Hardware-in-the-Loop Simulation

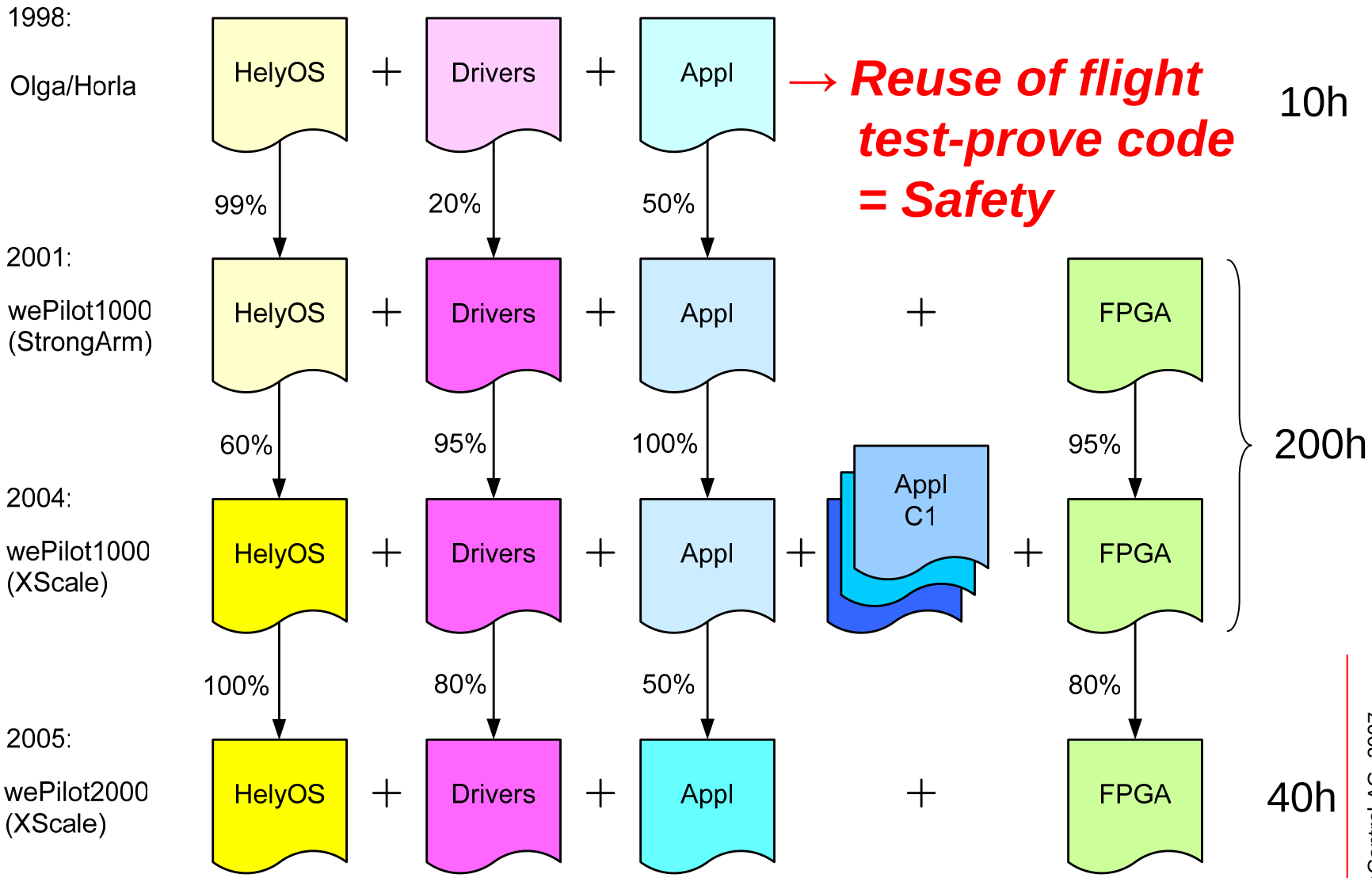


Algorithm Design with Matlab

EKFF.Mod
InnerLoop.Mod
Waypoint.Mod
Figures.Mod
Landing.Mod
Manual.Mod



Draw upon Experience



4. Coding Rules

1. Notation:
   - Constant: capital letter, e.g TIMEOUT
   - Variables: start with lower case letter, e.g. power
   - Procedures: start with capital letter, e.g. ComputePower
   - Procedure returning boolean: starts with Is... Or Has...
2. Use Comments
3. Use local variables instead of globals
4. Do not export any variables except finite state machines and status information. Access variables through procedures
5. Keep definition module (module interface) small
6. Minimize the number of tasks

Coding Rules [2]

7. Modules shall generally have
   - A procedure Start() for initialization
   - A procedure Stop() for termination
   - A procedure Handler() with a finite state machine for action coordination
   - A variable (usually a set) “status“ containing status information
8. Do not waste CPU resources
   - Optimize numerical algorithms
   - Use integers instead of floats
9. Allocate dynamic memory only during initialization (no garbage collection!)
10. No blocking code! Continuation shall never depend on external HW inputs. Use Timeouts!

5. Testing

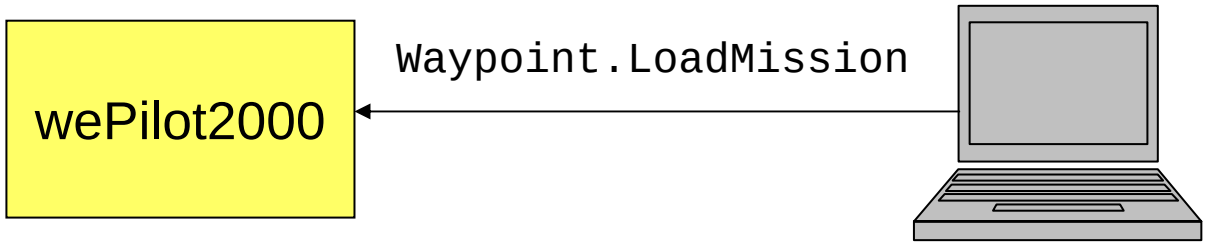
1. Test code within modules

2. Test modules

3. Comparison of Matlab/Oberon output

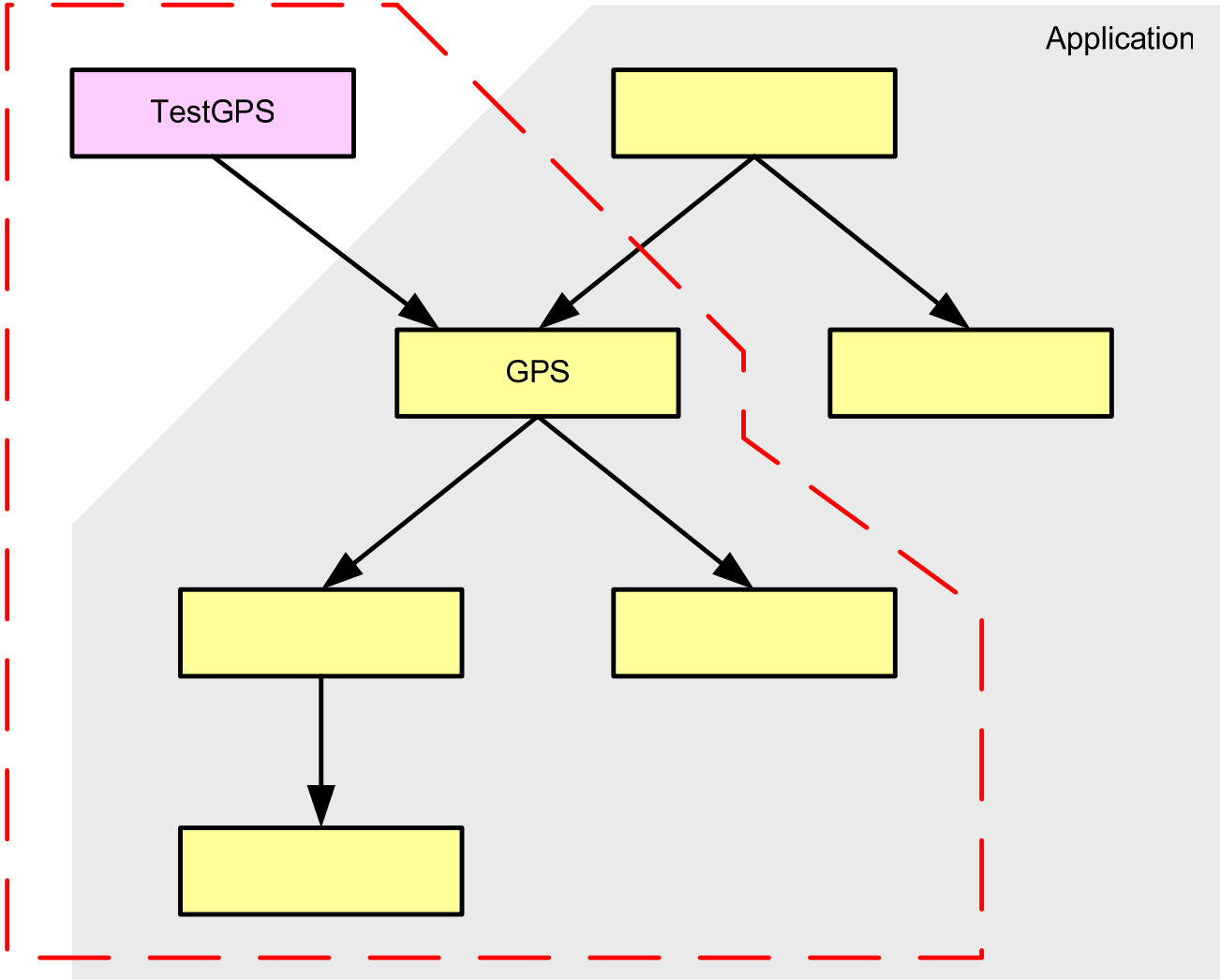
4. HIL tests / ground tests / in-flight tests

Test Code within Module



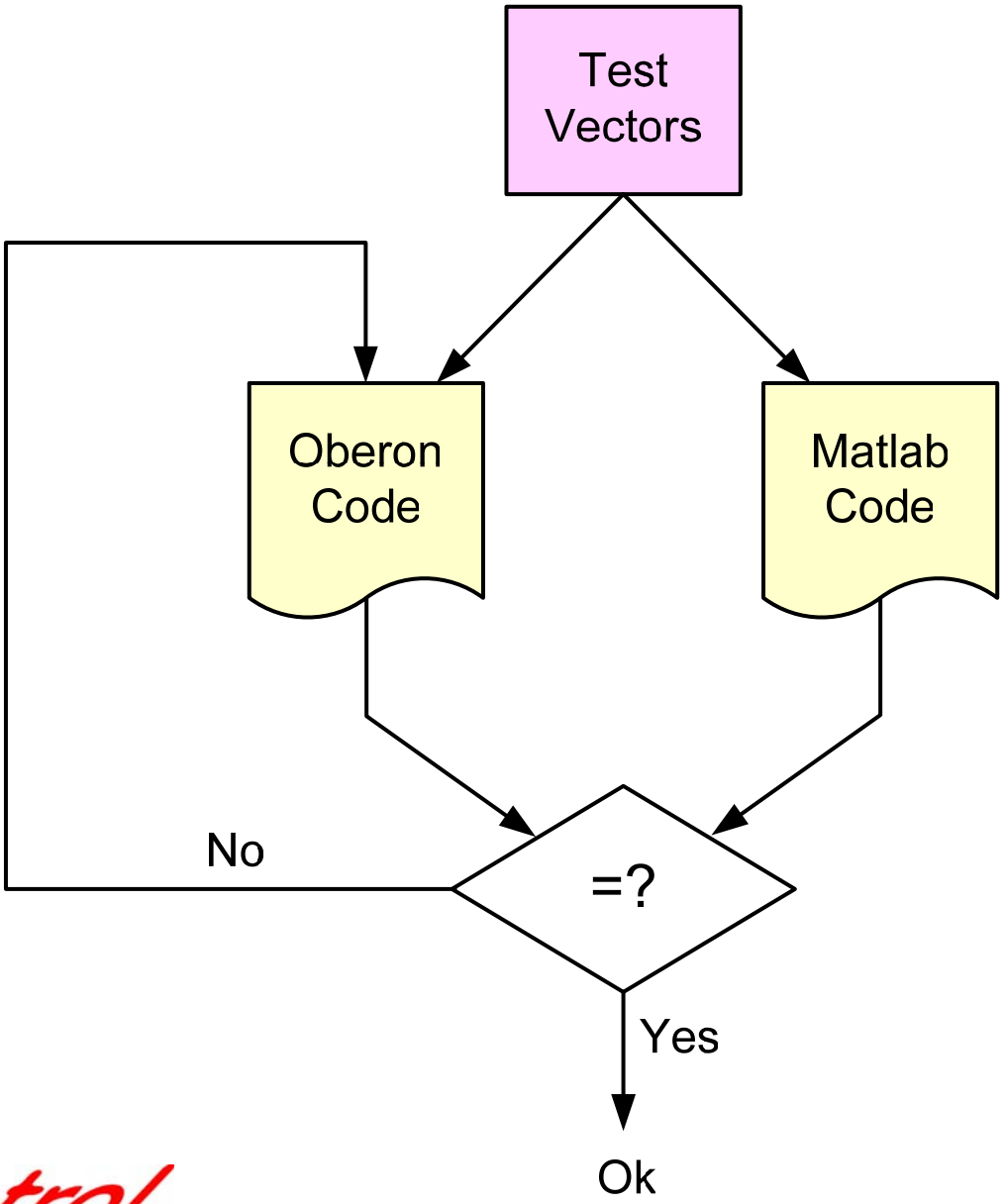

```
MODULE Waypoint;

PROCEDURE LoadMission*;
BEGIN
...
END LoadMission;

BEGIN
END Waypoint.
```

Use of Test Modules

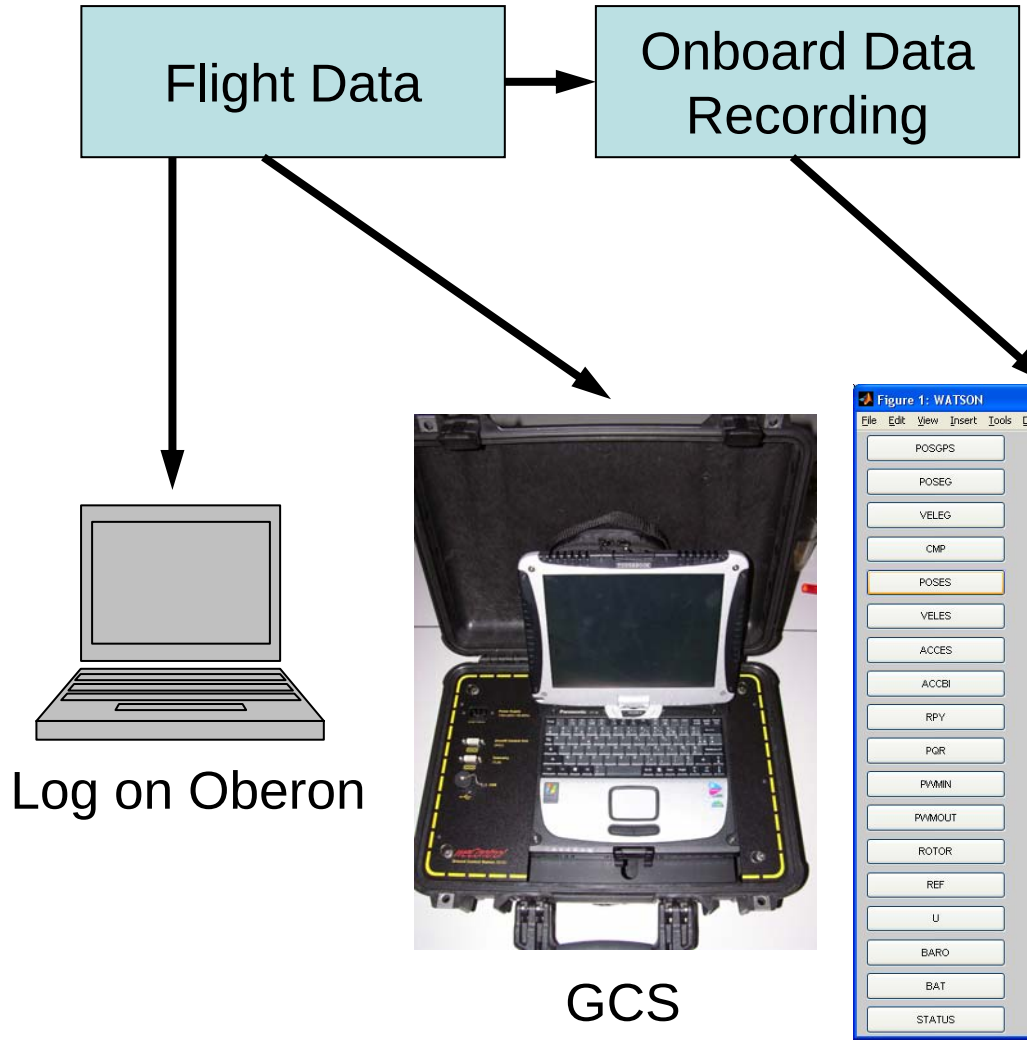


TestADC.Mod
TestBaro.Mod
TestEKFF.Mod
TestFigures.Mod
TestFRAM.Mod
TestGPS.Mod
TestIMU.Mod
TestInnerLoop.Mod
TestMagneto.Mod
TestManual.Mod
TestUART.Mod
TestWaypoint.Mod

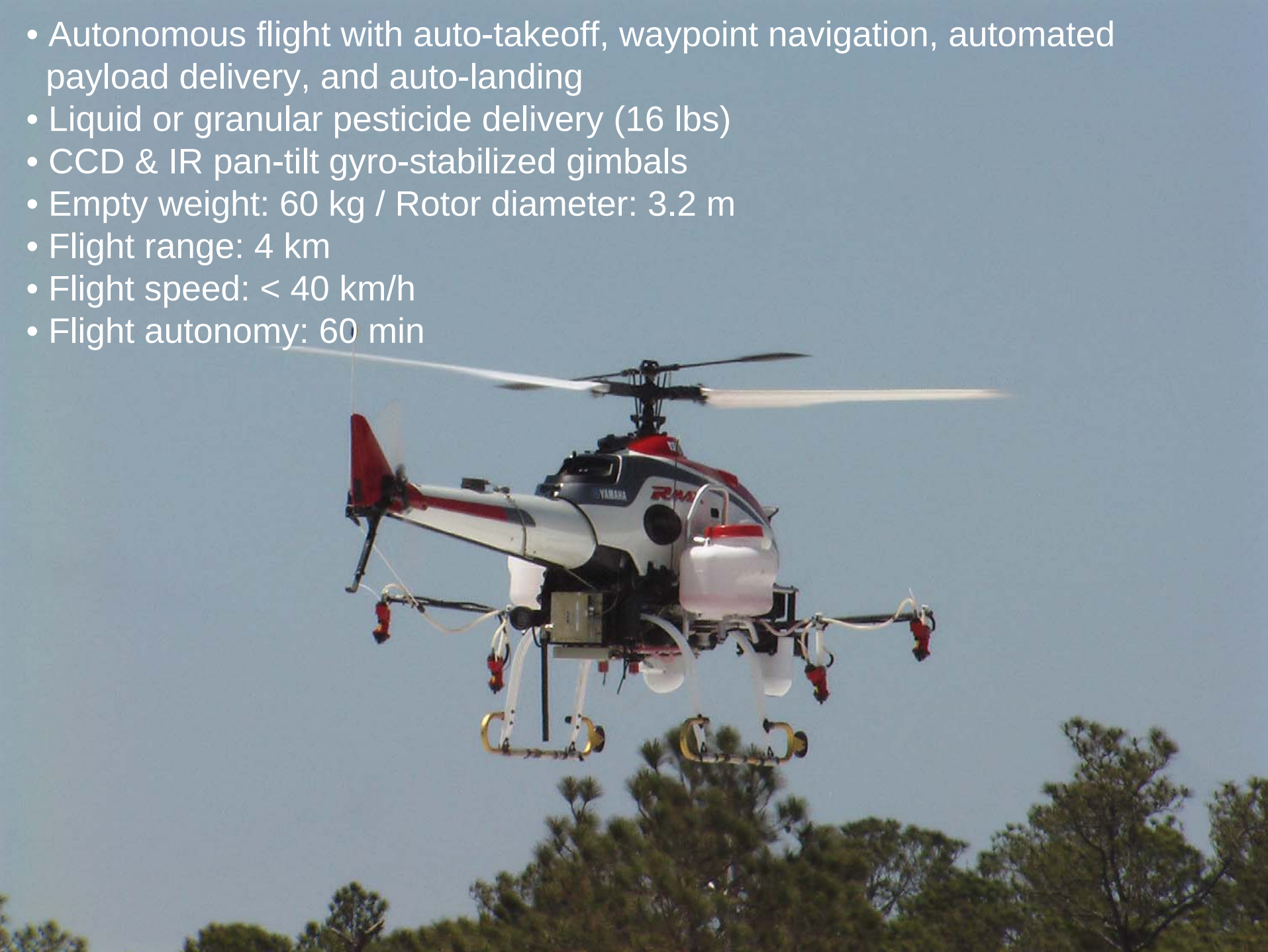
Comparison of Module Output



HIL/ Ground/ Flight Testing



- Autonomous flight with auto-takeoff, waypoint navigation, automated payload delivery, and auto-landing
- Liquid or granular pesticide delivery (16 lbs)
- CCD & IR pan-tilt gyro-stabilized gimbals
- Empty weight: 60 kg / Rotor diameter: 3.2 m
- Flight range: 4 km
- Flight speed: < 40 km/h
- Flight autonomy: 60 min



RCAVAP

Showing Automatic Take-off,
Pesticide Spraying Mission Execution,
and Automatic Landing

AFRL - Tyndall Air Force Base, FL, USA

Jan 2005

Tracker

(Joint development between EADS/Survey-Copter/weControl)

- wing span: 3.4 meters
- total weight: 8.5 kg
- horizontal flight speed:
  - 17 m/s (nominal)
  - 25 m/s (max)
- climb rate: 2 m/s, sink rate: 6 m/s
- endurance: 90 min
- propulsion system: 2 electro motors, LiPo batteries