Integrated Design and Analysis Tools for Software-based Control Systems

Software Enabled Control

OCP Participation Summary
UC Berkeley

Lead Investigators
Shankar Sastry
Edward A. Lee
Tom Henzinger
Alberto Sangiovanni-Vincentelli

Other Investigators
Luca Berardi
Luca de Alfaro
Magnus Egerstedt
Laurent El Ghaoui
Ben Horowitz
Karl Johansson
John Koo
Jie Liu
Xiaojun Liu
John Lygeros
Rupak Majumdar
George Pappas
Santosh Philip
Claudio Pinello
Maria Prandini
Shahid Rashid
Jean-Francois Raskin
Shawn M. Schaffert
Hyunchul Shim
Bruno Sinopoli
Slobodan Simic
Rene Vidal

Objectives

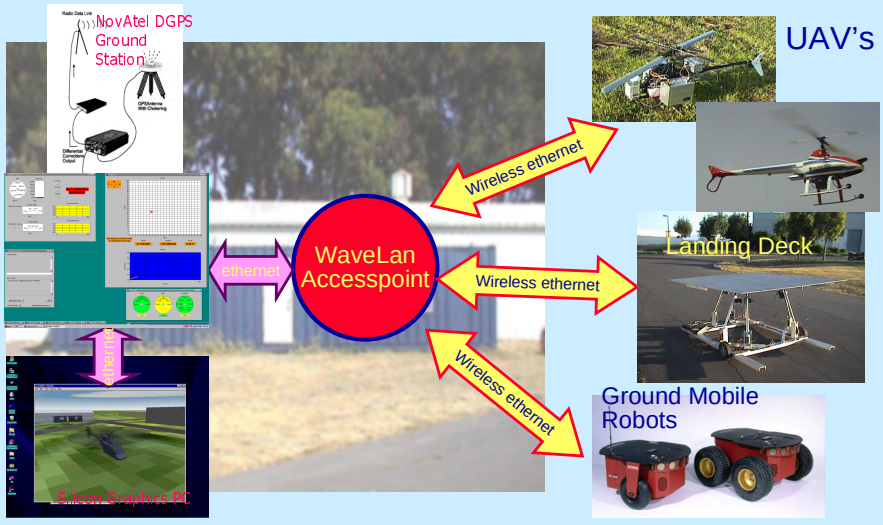
- OCP participation
  - "run-time support methods for hybrid and multi-modal systems."
- Component architectures
  - maintain efficiency
  - compose properties
- Understand designs
  - reduce reliance on simulation
  - correct-by-construction implementations
  - rely on pre-proven frameworks
- Orthogonalize concerns
  - regimes of operation
  - federated coordination

Trajectory Plan

- Study group (weekly++):
  - ...
  - 9/7 CORBA (part I) -- Concepts
  - 9/9 CORBA (part II) -- Java IDL
  - 9/14 TAO and Quality of Service in CORBA
  - 9/17 RT-IDL and Scheduling for Embedded Systems
  - 9/21 Case Study -- Helicopter Control Systems
  - 9/28 Real-time Operating Systems -- QNX/PSOS/VxWorks
  - 10/5 The Time-Triggered Architecture
  - 10/12 Timed Automata Verification
  - 10/19 Planning for kickoff meeting
  - ...
- Architecture for Berkeley AERobots (BEAR) project
- (RT) Corba experimental platform

BEAR Research Platform

thanks to: David H. Shim

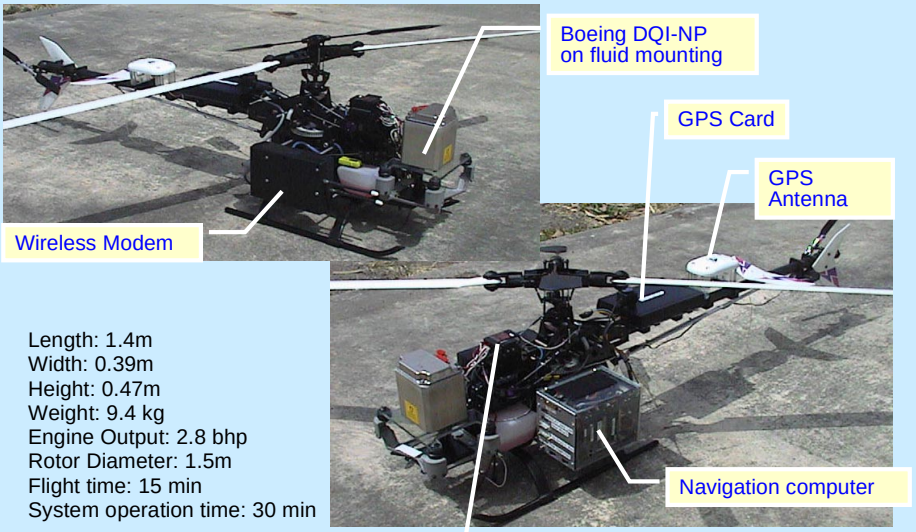


Ground Monitoring System

WaveLAN: T. John Koo
Pioneer mobile rotot: Omid Shakernia, Frank Hoffman
Pitching deck landing pad: Tulio

Ursa Minor3

thanks to: David H. Shim

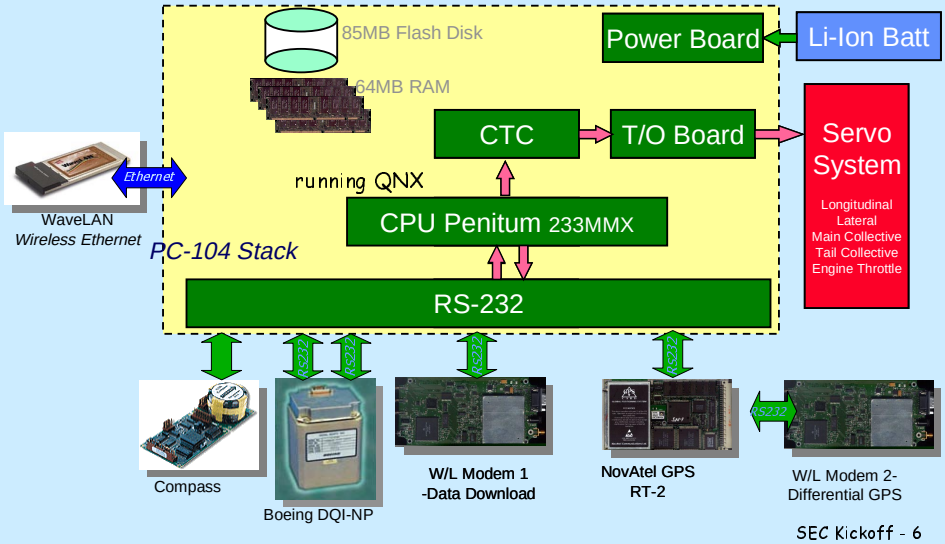


Length: 1.4m
Width: 0.39m
Height: 0.47m
Weight: 9.4 kg
Engine Output: 2.8 bhp
Rotor Diameter: 1.5m
Flight time: 15 min
System operation time: 30 min

Navigation Hardware (Ursa Magnus)

thanks to: David H. Shim

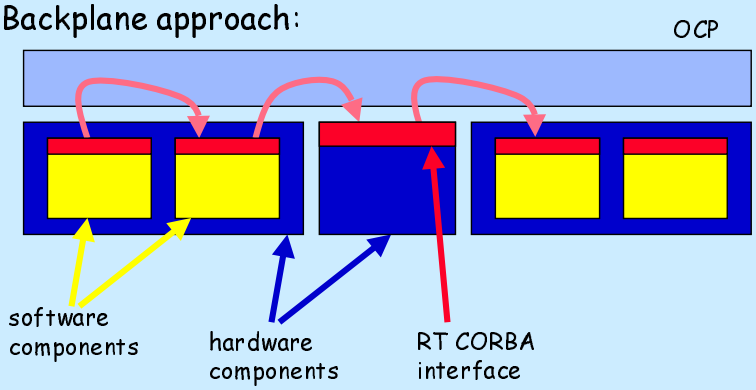
1. Ursa Magnus 2: Boeing DQI-NP based system



OCP = Component-Based Design

First principle: We seek software architectures for modular construction of distributed control systems.

Backplane approach:

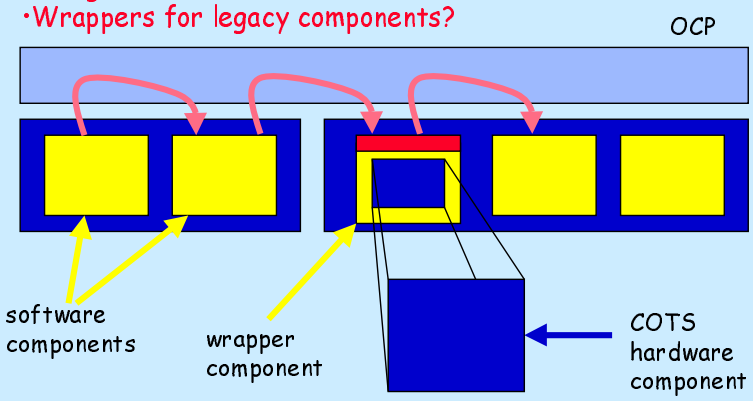


Run-Time Models

Key issue: What interface to expose at the OCP level?

- OO methods?
- Event notification?
- Irregular or low rate?
- Wrappers for legacy components?

Interface definition depends on the model of computation.



Model of Computation

- Component ontology
  - processes? objects? procedures?
  - reactive? active? passive?
- Component epistemology
  - visibility of other components
  - global information, such as time
  - reflection and introspection
- Interaction protocols
  - synchronization? push? pull?
  - delivery guarantees
- Interaction lexicon
  - vocabulary of messages
  - type system

A model of computation is the ontology and epistemology of components together with the protocols and lexicon of their interaction.

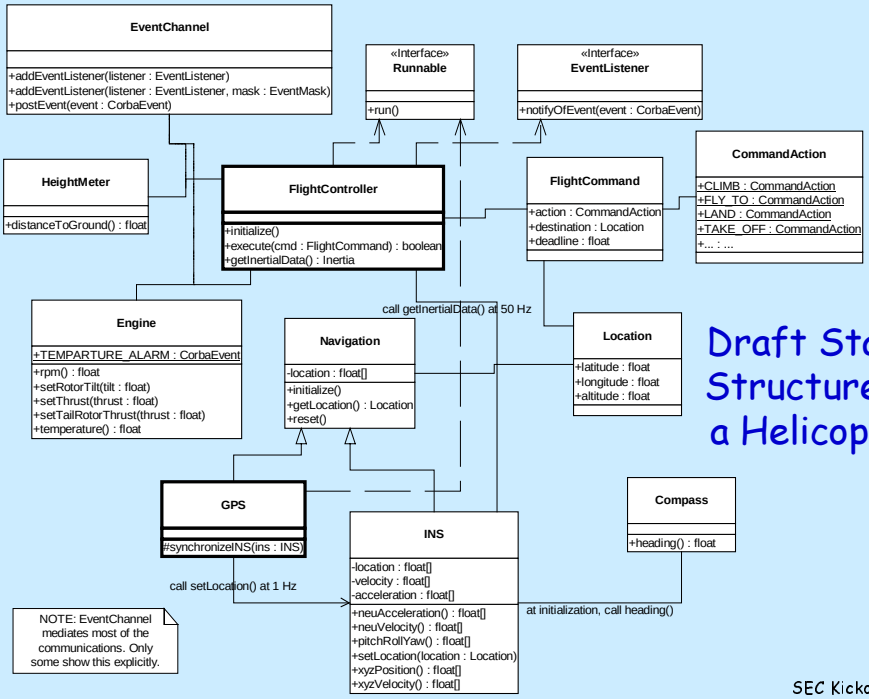
CORBA

- CORBA provides
  - distributed objects with location transparency
  - synchronous (two way) remote method invocation
  - asynchronous (one way) remote method invocation
  - deferred synchronous invocation (at higher cost)
- COS/CES event channel provides:
  - asynchronous notification
  - publish & subscribe
- RT event service:
  - prioritized dispatching
  - periodic event processing
  - active consumers and suppliers

Presumed Model of Computation

- Component ontology
  - Objects (CORBA) and processes (TAO)
- Component epistemology
  - publish & subscribe, with event filtering
  - time is reduced to priorities
- Interaction protocols
  - few constraints … anything is possible.
- Interaction lexicon
  - CORBA method arguments, return values

Similar to Linda and JavaSpaces, but with real-time extensions.



Draft Static Structure of a Helicopter

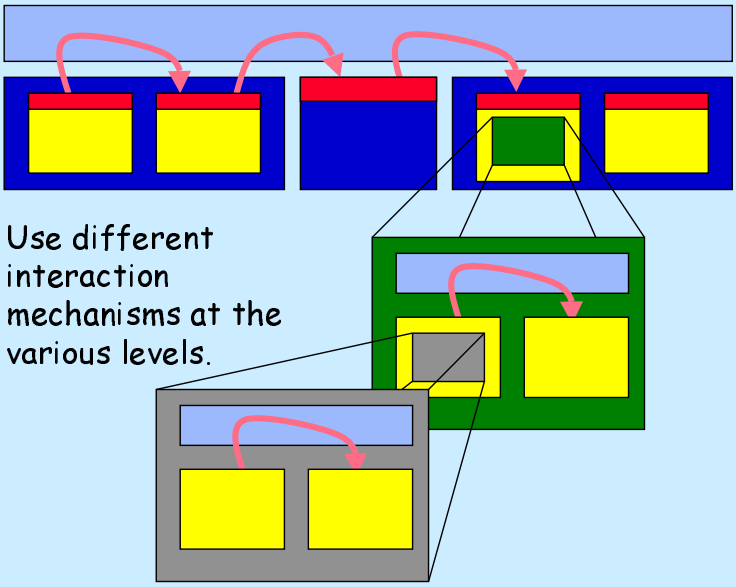
Event Examples

- initialize GPS
- initialize INS
- issue flight command
- GPS re-initialize INS (at 1 Hz)
- flight control reacts to INS data (at 50 Hz)
- sensors notify of landing
- height meter publishes distance to ground

Excluded

- servo loops (control laws)
- all fine-grain interaction
- all continuous interaction

Hierarchical Component-Based Design



Alternative Component Interactions

- Hybrid systems
  - hierarchical nesting of automata and ODEs
  - is the event channel relevant? How to use it?
- Modal models
  - hierarchical nesting of automata and anything else
  - are mode changes events in the event channel?
- Hard-real-time models
  - event channel seems more suited to notification of irregular events than to sampled-data signals.

How can we extend architectural principles to these alternative models?

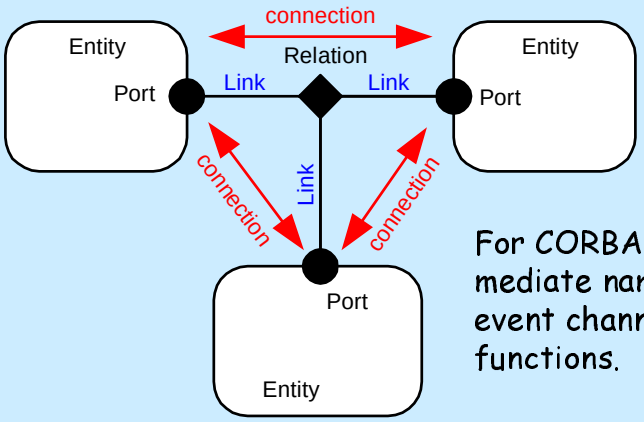
Relevant Models of Computation

- Publish and subscribe (Linda, JavaSpaces)
- Transition systems, state machines...
- Synchronous-reactive systems (SR)
- ODEs and PDEs (continuous dynamics)
- Discrete time (difference equations)
- Discrete-event systems (DE, VHDL, Verilog)
- Sequential processes with rendezvous (CSP)
- Process networks (Kahn)
- Dataflow (Dennis)
- ...

Proposal

- Identify a small suite of MoCs useful for distributed control system design
  - one will not be enough
  - architecture at all levels
- Study inter-domain semantics
  - verifiability
  - comprehensibility
- Emphasize what is common across MoCs
  - abstract syntax for component architecture
  - semantic commonalities (such as type systems)

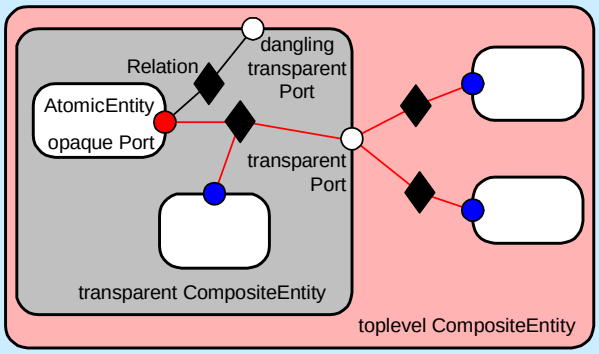
Generic Component Architecture (an abstract syntax)



For CORBA, relations mediate name service, event channel, and RPC functions.

The OCP effort should first agree on an abstract syntax.

Hierarchy & Abstraction

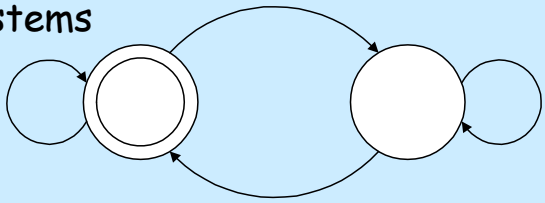


Ideally, aggregations of components behave like primitive components.

homosemantic composition.

Sequential Composition is Homosemantic

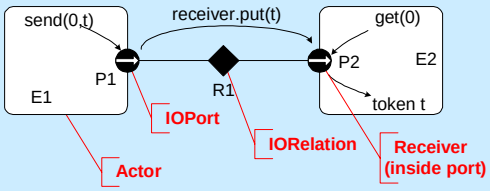
- Statements in imperative languages
- Procedures
- Objects
- State machines
- Transition systems



- Processes and threads are *not* homosemantic

Abstract Semantics

Basic Transport:



Data transport
- broadcast
- publish & subscribe
- multicast
- push/pull
- messages or RPC
- synchronization
- delivery guarantees
- typing
- polymorphism

The OCP effort should focus on defining its abstract semantics - independent of an implementation, what behavior do we want in component interactions?

Key Points

- Heterogeneous hierarchical models can isolate certain sources of complexity, e.g. separating modes from dynamics or events from dynamics.
- Consistent use of input/output views of component models facilitates their hierarchical composition (and is consistent with an event-channel transport mechanism).
- At all levels, there is a component architecture. Share infrastructure.

Mission Plan

- OCP participation (Repeated)
  - "run-time support methods for hybrid and multi-modal systems."
- Understand application area
  - software architecture perspective.
- Realize event-level architecture
  - characterize intercomponent interaction semantics.
- Realize multi-level architecture
  - characterize interlevel semantics.
- Develop validation methods.
  - coupled with intercomponent interaction semantics

Conclusions

- We are about component based design of real-time, safety-critical control systems.
- Dialog should be about models of computation and component architectures.
- Agreement should be about abstract syntax, abstract semantics (first).