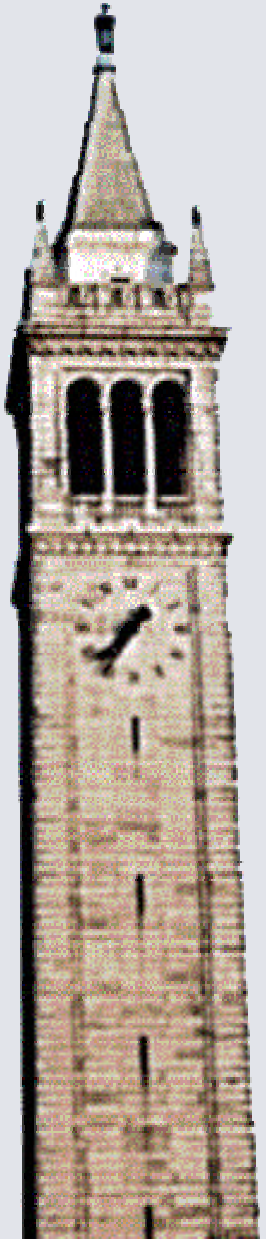
Polymorphic Actor-Oriented Design for Heterogeneous Embedded Software

Edward A. Lee
Professor
UC Berkeley

Seventh Annual Workshop on Embedded Computing
HPEC, Sept. 23, 2003
Boston, MA, USA

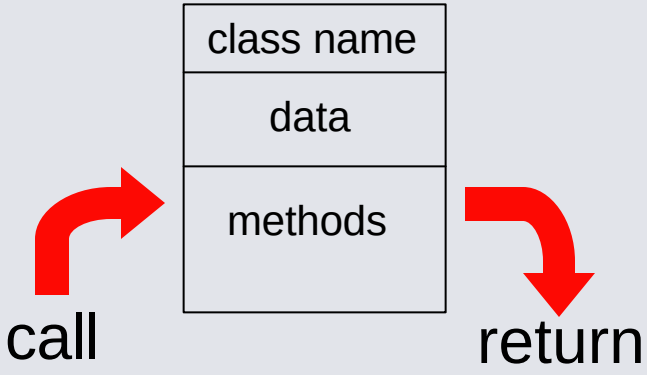


Chess

Center for Hybrid and Embedded Software Systems

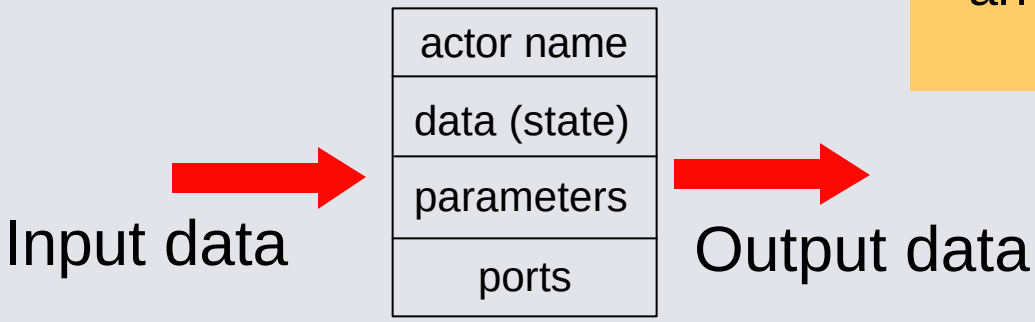
Actor-Oriented Design

- Object orientation:



What flows through an object is sequential control

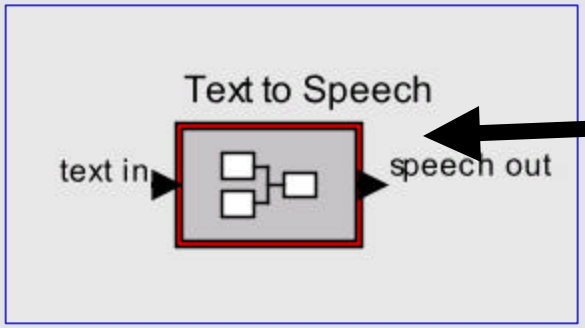
- Actor orientation:



What flows through an object is data streams

Problems and Solutions

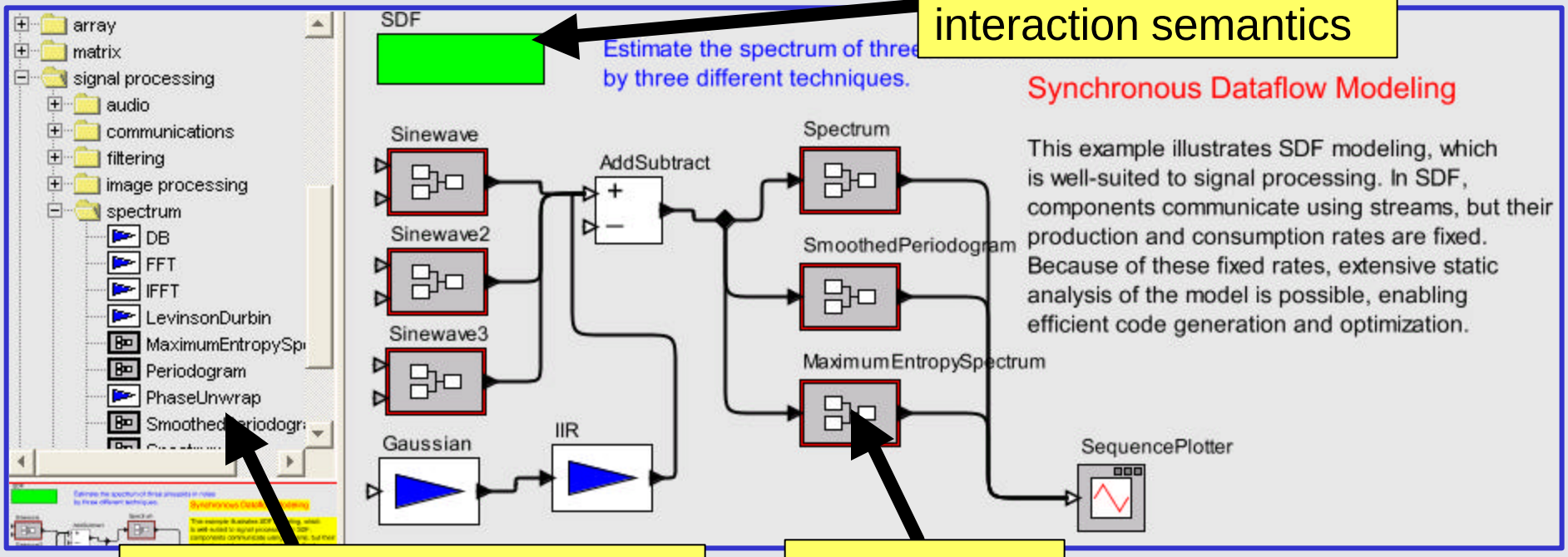
- Some Problems:
  - OO says little or nothing about concurrency and time
  - Components implement low-level communication protocols
  - Components are designed to fixed middleware APIs
  - Re-use potential is disappointing
- Some Partial Solutions
  - Adapter objects (laborious to design and deploy)
  - Model-driven architecture (still fundamentally OO)
  - Executable UML (little or no analyzable structure)
- Our Solution is Based on Actor-Oriented Design with:
  - Behavioral, polymorphic type system
  - Meta modeling of semantics



Interface definition gives requirements and guarantees only, not implementations. E.g.“Give me text and I’ll give you speech” rather than “I have void initialize(), void notify(), boolean isRead(), double[ ] getSpeech() ….”

Example of Actor-Oriented Design (in this case, with a visual syntax)

Ptolemy II example:



Key idea: The model of computation is part of the framework within which components are embedded rather than part of the components themselves.

Model of Computation:
- Messaging schema
- Flow of control
- Concurrency