Are new languages necessary for multicore?

Edward A. Lee
Robert S. Pepper Distinguished Professor and Chair of EECS, UC Berkeley

Position Statement for Panel
International Symposium on Code Generation and Optimization (CGO)

March 11-14, 2007, San Jose, California
“Multicore architectures will (finally) bring parallel computing into the mainstream. To effectively exploit them, legions of programmers must emphasize concurrency.”

The vendor push:

“Please train your computer science students to do extensive multithreaded programming.”
Is this a good idea?
My Claim

Nontrivial software written with threads, semaphores, and mutexes are incomprehensible to humans.
Is Concurrency Hard?

It is not concurrency that is hard…

Lee, Berkeley 5
Threads are sequential processes that share memory. From the perspective of any thread, the entire state of the universe can change between any two atomic actions (itself an ill-defined concept).

*Imagine if the physical world did that...*
Succinct Problem Statement

Threads are wildly nondeterministic.

The programmer’s job is to prune away the nondeterminism by imposing constraints on execution order (e.g., mutexes) and limiting shared data accesses (e.g., OO design).
Do Threads Provide a Sound Foundation for Concurrent Programming?

- Imperative languages are fine.
- Threads change everything (except syntax)
- We can fix the problem at the software component level.

*Note that this whole enterprise is held up by threads*
Component Architecture Alternatives
Object Oriented vs. Actor Oriented

The established: Object-oriented:

- What flows through an object is sequential control
- Things happen to objects

<table>
<thead>
<tr>
<th>class name</th>
<th>data</th>
<th>methods</th>
</tr>
</thead>
</table>

The alternative: Actor oriented:

- Actors make things happen
- What flows through an object is evolving data

<table>
<thead>
<tr>
<th>actor name</th>
<th>data (state)</th>
<th>parameters</th>
<th>ports</th>
</tr>
</thead>
</table>

Input data  Output data

Lee, Berkeley 9
The Solution

Actor-oriented component architectures implemented in \textit{coordination languages} that complement rather than replace existing imperative languages.

See the Ptolemy Project for ongoing research addressing these problems: \url{http://ptolemy.org}