Panel: What Comes After C++ in System-Level Specification

Forum on Design Languages

Workshop on System Specification & Design Languages

September 4-8, 2000 - Tübingen, Germany

Edward Lee
UC Berkeley

What is C++

- Object-oriented language
  - Abstract data types
  - Polymorphism
  - Inheritance
- No concurrency
  - Sequential flow of control

It is *not* a system-level specification language!

What About Java?

- Has (very low level) concurrency
  - threads
  - monitors
- Threads are hard to use! Sun says in the on-line Java tutorial:

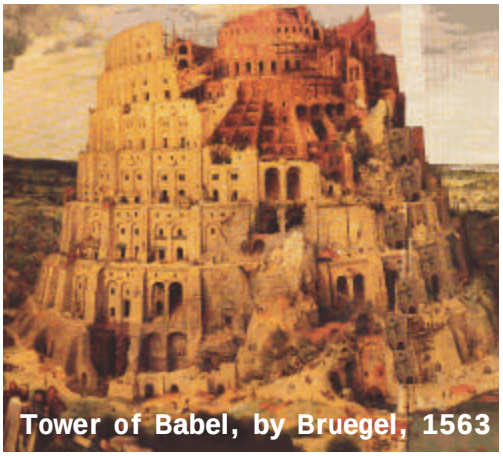
  "The first rule of using threads is this: avoid them if you can. Threads can be difficult to use, and they tend to make programs harder to debug."
- Threads are a poor model for concurrent hardware.

What is System C?

- An elaborate design pattern
  - happens to be defined as a set of C++ classes
  - has semantics well beyond those of C++
  - is more properly viewed as a new language than as C++
- A concurrency model
  - cycle driven (discrete-time)
  - multirate (some discrete-event flavor)

This *is* a system-level specification language, but it is *not* C++.
  - Is it the *right* SLDL?
  - Is it *sufficient*?

Using the *syntax* of C++ does not mean you are programming in C++!

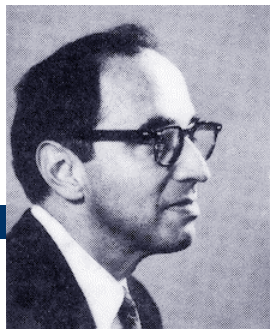


Tower of Babel, by Bruegel, 1563

Many languages use the Roman alphabet, but that does not make them Latin.

The warm, fuzzy familiarity of the *syntax* of C++ is a marketing tool, not a technical tool.

Thomas Kuhn, originator of the paradigm paradigm

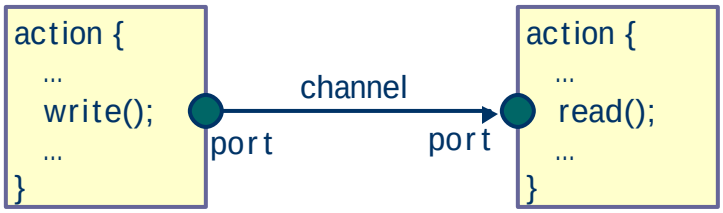


Proposed Focus: Component Frameworks

- What is a component? (ontology)
  - States? Processes? Threads? Differential equations? Constraints? Objects (data + methods)?
- What knowledge do components share? (epistemology)
  - Time? Name spaces? Signals? State?
- How do components communicate? (protocols)
  - Rendezvous? Message passing? Continuous-time signals? Streams? Method calls? Events in time?
- What do components communicate? (lexicon)
  - Objects? Transfer of control? Data structures? ASCII text?

A Class of Concurrent Frameworks: Producer / Consumer

Are actors active? passive? reactive? cycle-driven?



Are communications timed? synchronized? buffered?

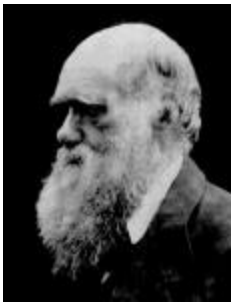
Concurrent Component Frameworks

- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DT – discrete time (cycle driven)
- PN – process networks
- SDF – synchronous dataflow
- SR – synchronous/reactive

There are many more possibilities!

Survival of the fittest is the *only* reasonable way to choose among these.

- A concurrent flow of control
- An interaction protocol.



Interoperability Levels

- Code can be written to translate the data from one tool to be used by another.
- Tools can open each other's files and extract useful information (not necessarily *all* useful information).
- Tools can interoperate dynamically, exchanging information at run time.

Syntax alone achieves *none* of these.

Conclusions

Any nice unified system-level design language is destined to be a drowning cathedral.

- This *is* socket science!
- "Strategy is better than strength"
  - proverb from the Hausa culture of Nigeria

What is UML?

- A family of graphical notations
  - object models
  - interaction diagrams
  - statecharts
  - package diagrams
  - use diagrams
- Very weak at expressing concurrency
  (except its statecharts model, which has a specialized form of concurrency)