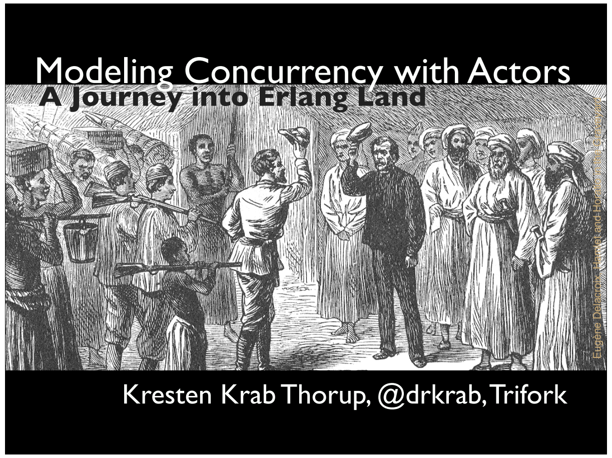
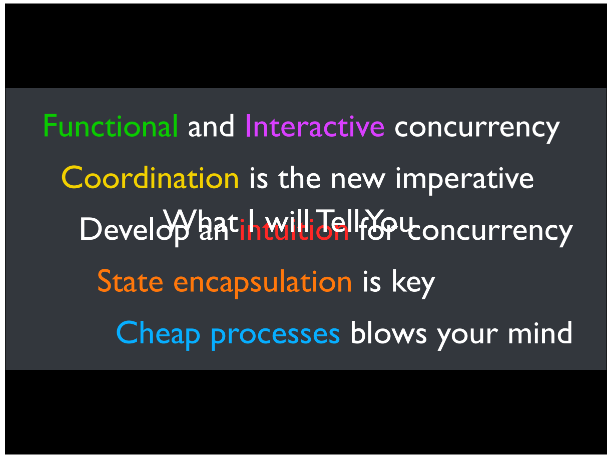


Modeling Concurrency with Actors

A Journey into Erlang Land

Kresten Krab Thorup, @drkrab, Trifork

Eugène Delacroix ... Erfgoed ... Heritage



What I will Tell You

Functional and Interactive concurrency

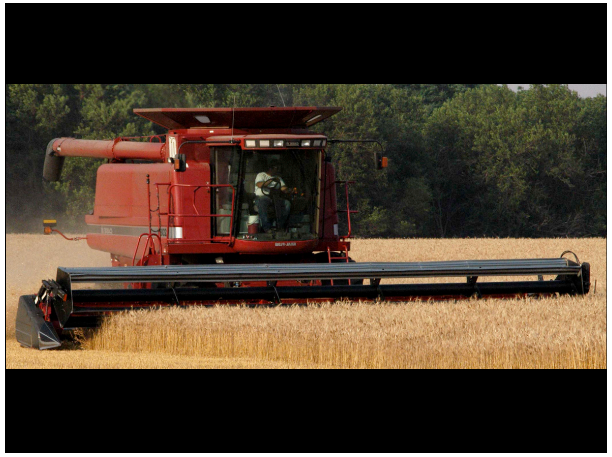
Coordination is the new imperative

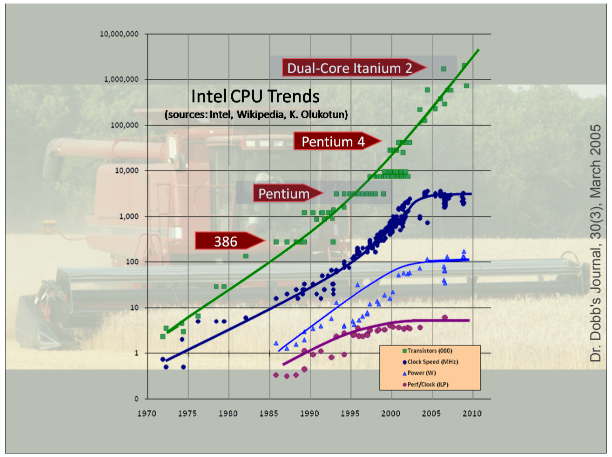
Develop an intuition for concurrency

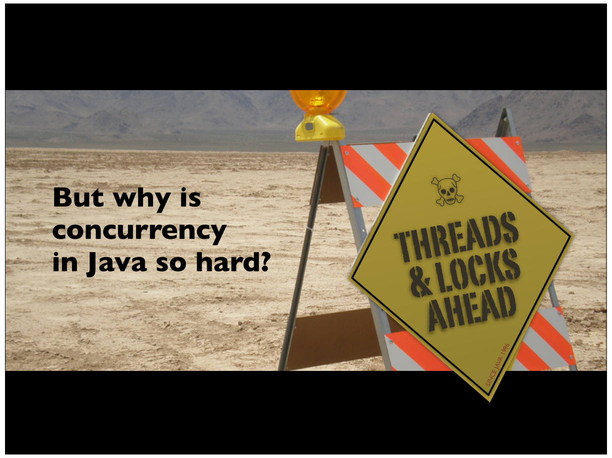
State encapsulation is key

Cheap processes blows your mind

The Free Lunch is Over



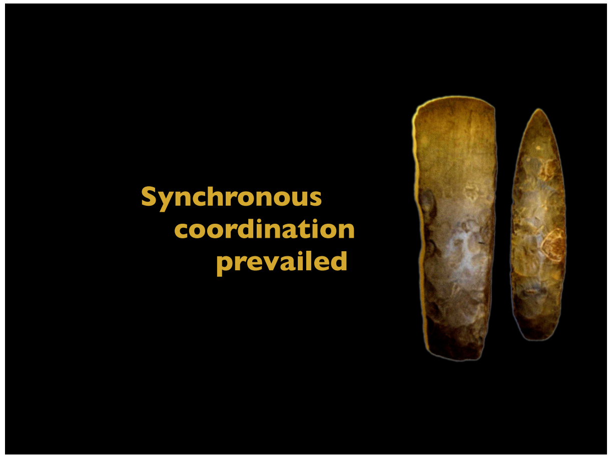




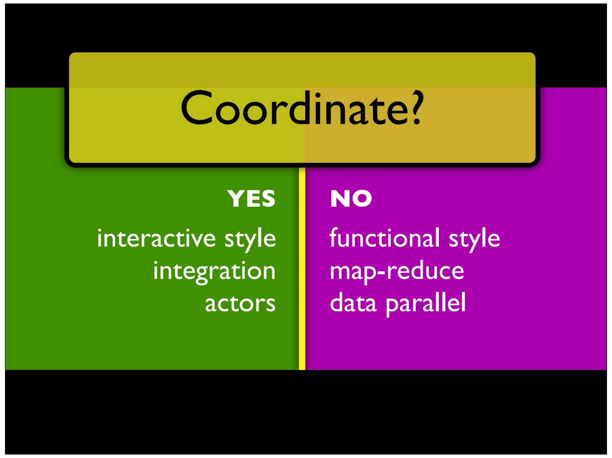
But why is concurrency in Java so hard?

THREADS & LOCKS AHEAD

Java was designed in the client-server age

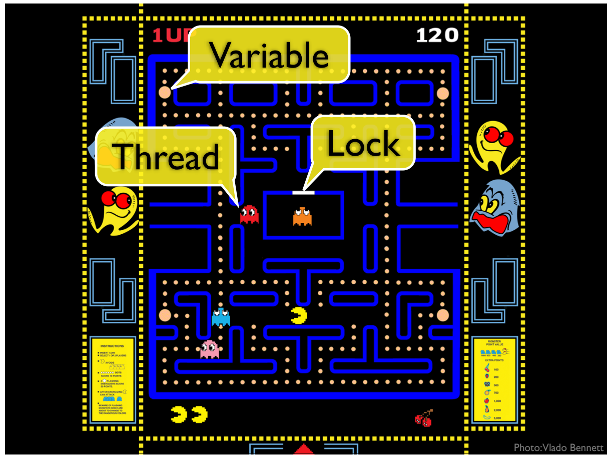


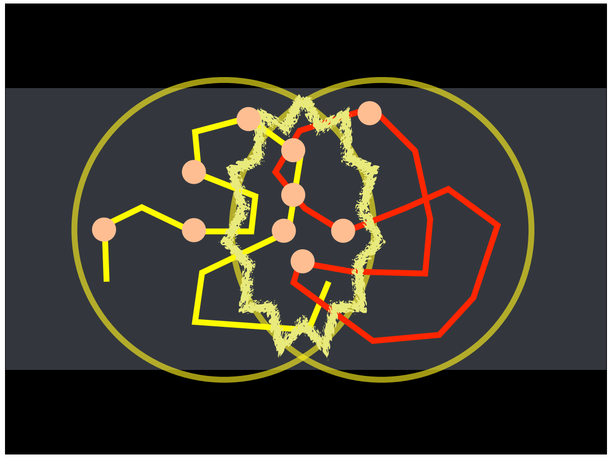
Synchronous coordination prevailed



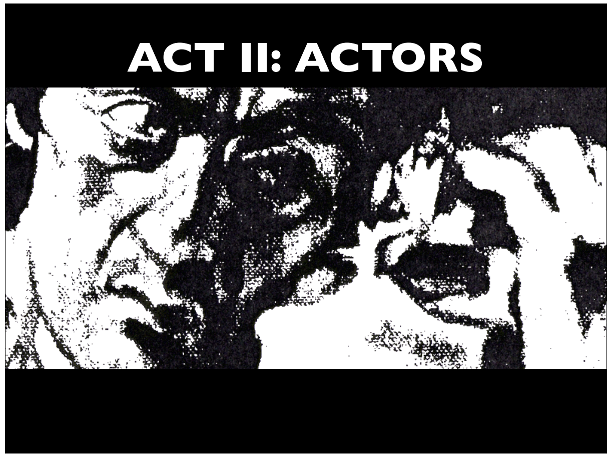
Coordinate?

YES	NO
interactive style	functional style
integration	map-reduce
actors	data parallel

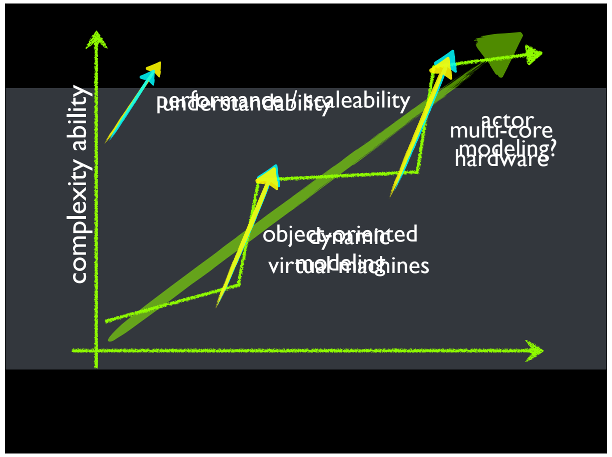


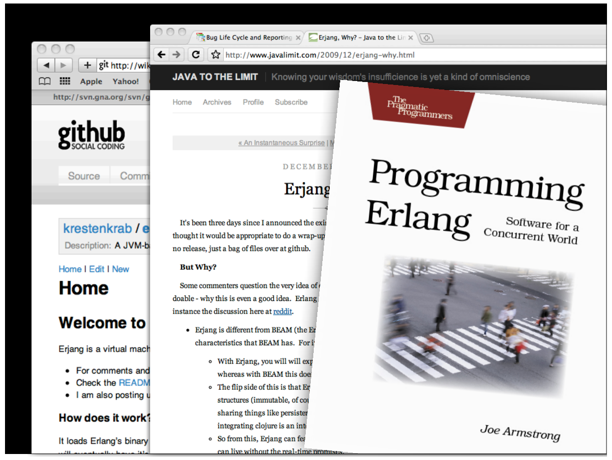


If you think you need threads and locks, think again.
THREADS & LOCKS AHEAD
Java Threads
BRIAN GOETZ
JAVA CONCURRENCY IN PRACTICE
Doug Lea
Concurrent Programming in Java Second Edition
Design Principles and Patterns



ACT II: ACTORS




Bug Life Cycle and Reporting
Erjang, Why? – Java to the Li
http://www.javalimit.com/2009/12/erjang-why.html
git http://wik
Apple
Yahoo!
http://svn.gna.org/svn/g
github SOCIAL CODING
Source
Comm
JAVA TO THE LIMIT
Knowing your wisdom's insufficience is yet a kind of omniscience
Home
Archives
Profile
Subscribe
« An Instantaneous Surprise | M

krestenkrab / e

Description: A JVM-b

Home | Edit | New

Home

Welcome to

Erjang is a virtual mach

- For comments and
- Check the READM
- I am also posting u

How does it work?

It loads Erlang's binary

DECEMBE

Erjang

It's been three days since I announced the exi thought it would be appropriate to do a wrap-up no release, just a bag of files over at github.

But Why?

Some commenters question the very idea of doable - why this is even a good idea. Erlang instance the discussion here at reddit.

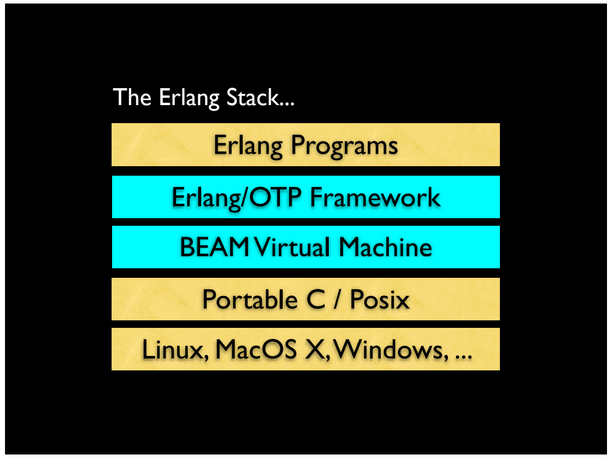
- Erjang is different from BEAM (the Er characteristics that BEAM has. For i
  - With Erjang, you will will exp whereas with BEAM this doe
  - The flip side of this is that Er structures (immutable, of co sharing things like persiste integrating clojure is an int
  - So from this, Erjang can fe

The Pragmatic Programmers

Programming Erlang

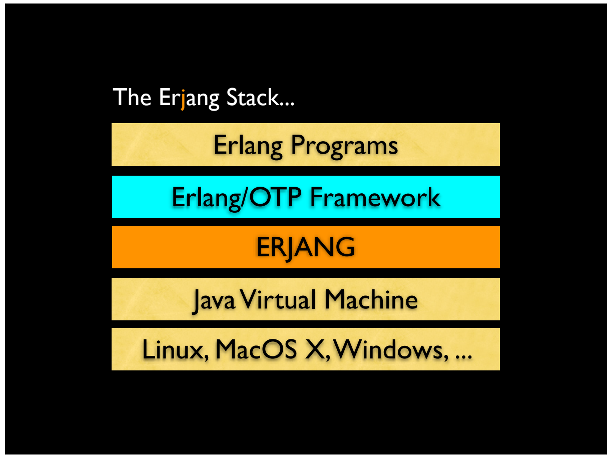
Software for a Concurrent World

Joe Armstrong



The Erlang Stack...

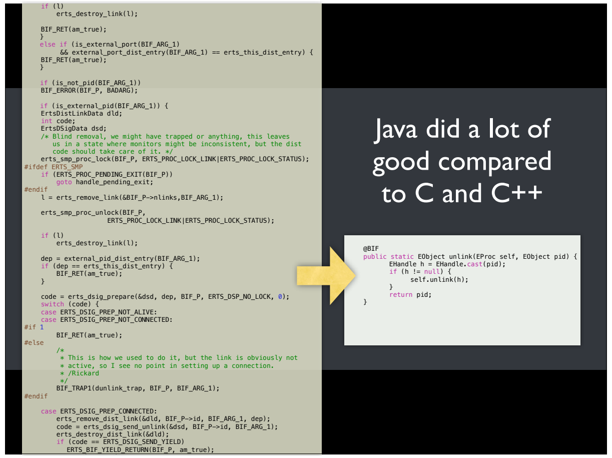
Erlang Programs
Erlang/OTP Framework
BEAM Virtual Machine
Portable C / Posix
Linux, MacOS X, Windows, ...



The Erjang Stack...

Erlang Programs
Erlang/OTP Framework
ERJANG
Java Virtual Machine
Linux, MacOS X, Windows, ...

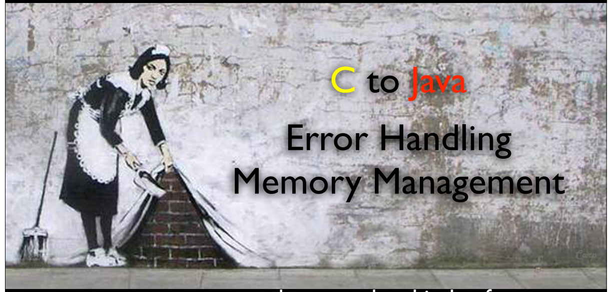
Let's see it in action...



```
    if (l)
        erts_destroy_link(l);

    BIF_RET(am_true);
    }
   else if (is_external_port(BIF_ARG_1)
        && external_port_dist_entry(BIF_ARG_1) == erts_this_dist_entry) {
    BIF_RET(am_true);
    }

   if (is_not_pid(BIF_ARG_1))
    BIF_ERROR(BIF_P, BADARG);

   if (is_external_pid(BIF_ARG_1)) {
    ErtsDistLinkData dld;
    int code;
    ErtsDSigData dsd;
    /* Blind removal, we might have trapped or anything, this leaves
       us in a state where monitors might be inconsistent, but the dist
       code should take care of it. */
    erts_smp_proc_lock(BIF_P, ERTS_PROC_LOCK_LINK|ERTS_PROC_LOCK_STATUS);
#ifdef ERTS_SMP
    if (ERTS_PROC_PENDING_EXIT(BIF_P))
        goto handle_pending_exit;
#endif
    l = erts_remove_link(&BIF_P->nlinks,BIF_ARG_1);

    erts_smp_proc_unlock(BIF_P,
                         ERTS_PROC_LOCK_LINK|ERTS_PROC_LOCK_STATUS);

    if (l)
        erts_destroy_link(l);

    dep = external_pid_dist_entry(BIF_ARG_1);
    if (dep == erts_this_dist_entry) {
        BIF_RET(am_true);
    }

    code = erts_dsig_prepare(&dsd, dep, BIF_P, ERTS_DSP_NO_LOCK, 0);
    switch (code) {
    case ERTS_DSIG_PREP_NOT_ALIVE:
    case ERTS_DSIG_PREP_NOT_CONNECTED:
#if 1
        BIF_RET(am_true);
#else
        /*
         * This is how we used to do it, but the link is obviously not
         * active, so I see no point in setting up a connection.
         * /Rickard
         */
        BIF_TRAP1(dunlink_trap, BIF_P, BIF_ARG_1);
#endif

    case ERTS_DSIG_PREP_CONNECTED:
        erts_remove_dist_link(&dld, BIF_P->id, BIF_ARG_1, dep);
        code = erts_dsig_send_unlink(&dsd, BIF_P->id, BIF_ARG_1);
        erts_destroy_dist_link(&dld);
        if (code == ERTS_DSIG_SEND_YIELD)
          ERTS_BIF_YIELD_RETURN(BIF_P, am_true);
```

Java did a lot of good compared to C and C++

```
@BIF
public static EObject unlink(EProc self, EObject pid) {
        EHandle h = EHandle.cast(pid);
        if (h != null) {
              self.unlink(h);
        }
        return pid;
}
```



C to Java

Error Handling

Memory Management

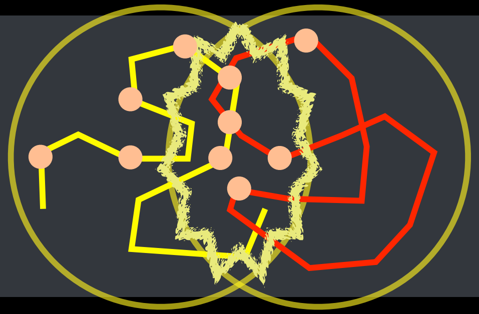
...and many other kinds of messes

Java to Erlang

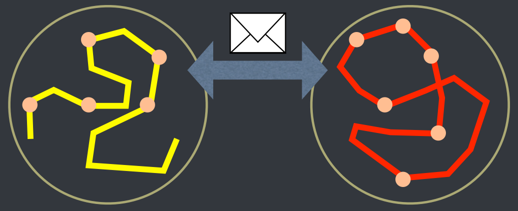
Coordination

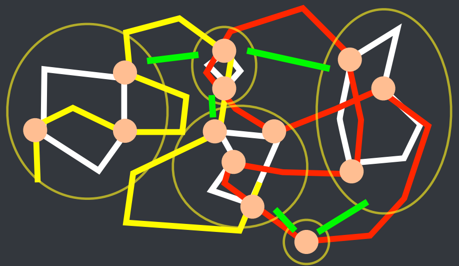
Fault-Tolerance

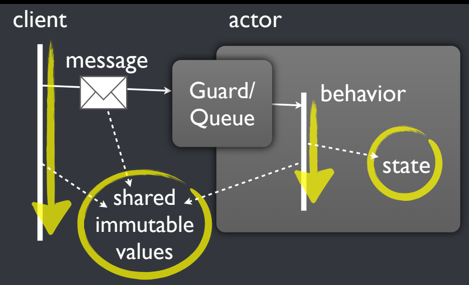
...and many other kinds of messes



http://www.youtube.com/watch?v=NXLuyZMEZbk







FRAMEWORKS

Kilim
Scala Actors
ActorFoundry
JavAct

LANGUAGES

Erlang
E Language
Axum

STATE ENCAPSULATION

SAFE MESSAGING

REAL PROCESSES

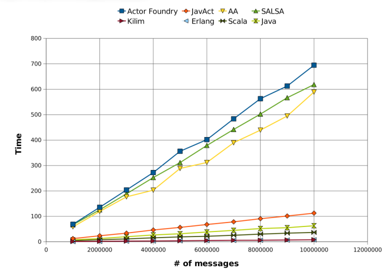
State encapsulation

```
object semaphore {
  class SemaphoreActor() extends Actor {
  ...
  def enter() {
    if (num < MAX) {
    // critical section
    num = num + 1; }}}

def main(args : Array[String]) : Unit = {
    var gate = new SemaphoreActor()
    gate.start; gate ! "enter"
    gate.enter }}
```

Design impact

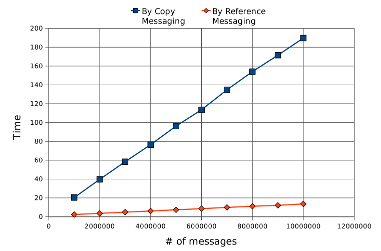
THREADS	**PROCESSES**
Blocking is expensive	Blocking is cheap
Choose: Blocking or non-Blocking interactions	Eat your cake and have it too.

Performance



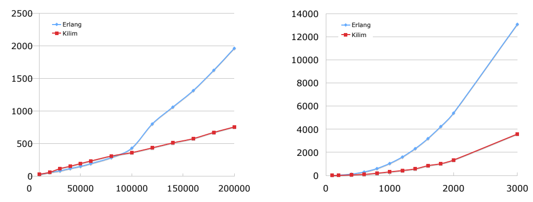
Karmani, Shali, Agha; PPPJ'09

Cost of Safe Messaging



Karmani, Shali, Agha; PPPJ'09

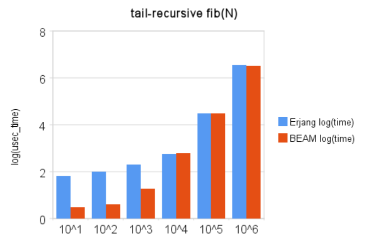
Kilim vs. Erlang



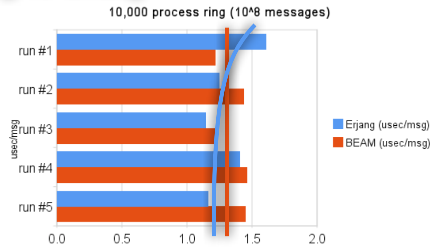
(a) Creation and Destruction (b) Messaging

Srinivasan & Mycroft, ECOOP'08

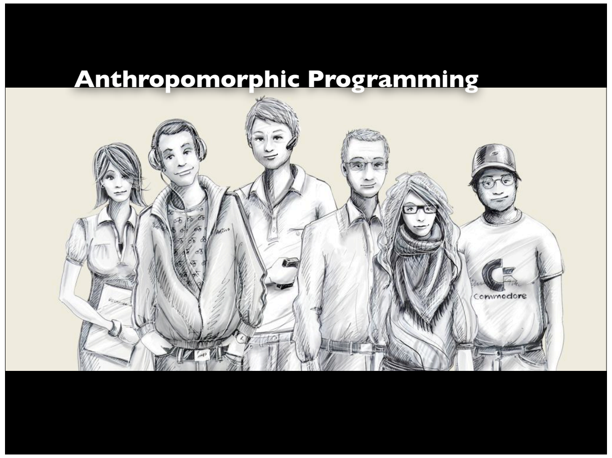
Erjang vs. Erlang



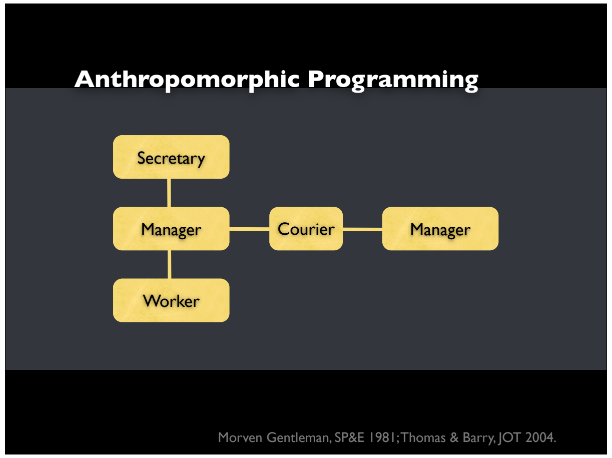
Erjang vs. Erlang



Anthropomorphic Programming

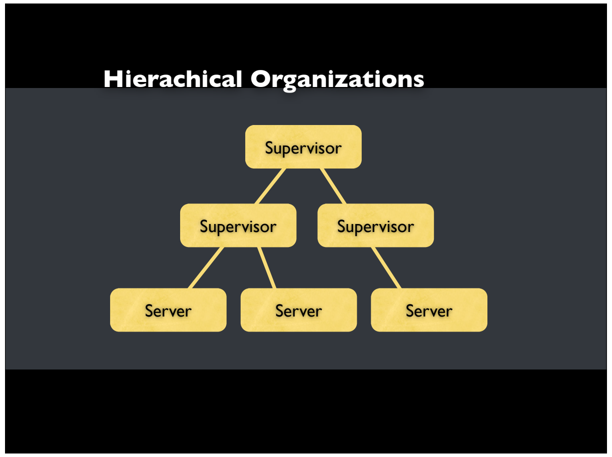


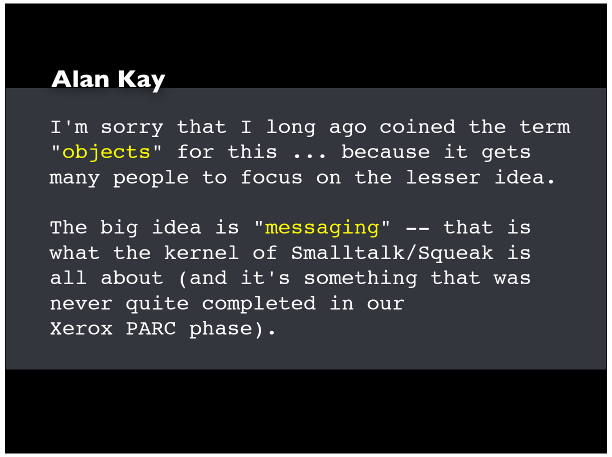
Anthropomorphic Programming



Morven Gentleman, SP&E 1981;Thomas & Barry, JOT 2004.

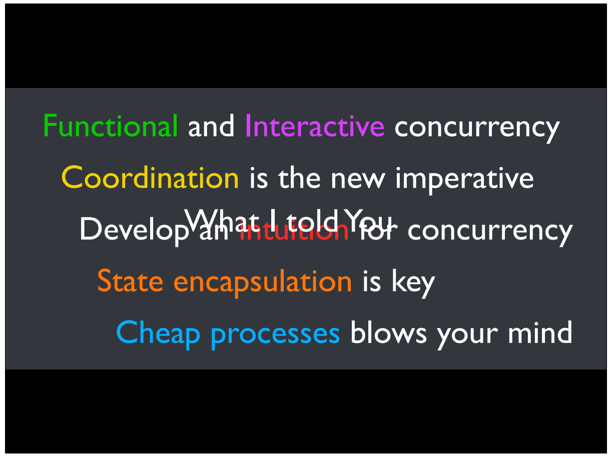
Hierachical Organizations





Alan Kay

```
I'm sorry that I long ago coined the term
"objects" for this ... because it gets
many people to focus on the lesser idea.

The big idea is "messaging" -- that is
what the kernel of Smalltalk/Squeak is
all about (and it's something that was
never quite completed in our
Xerox PARC phase).
```



Functional and Interactive concurrency

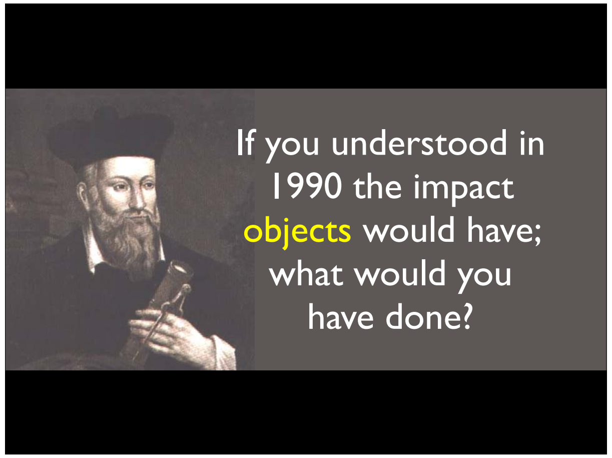
Coordination is the new imperative

Develop an intuition for concurrency

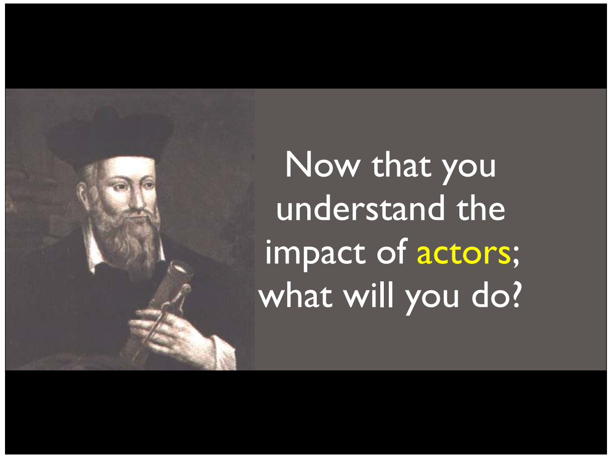
What I told You

State encapsulation is key

Cheap processes blows your mind



If you understood in 1990 the impact objects would have; what would you have done?



Now that you understand the impact of actors; what will you do?