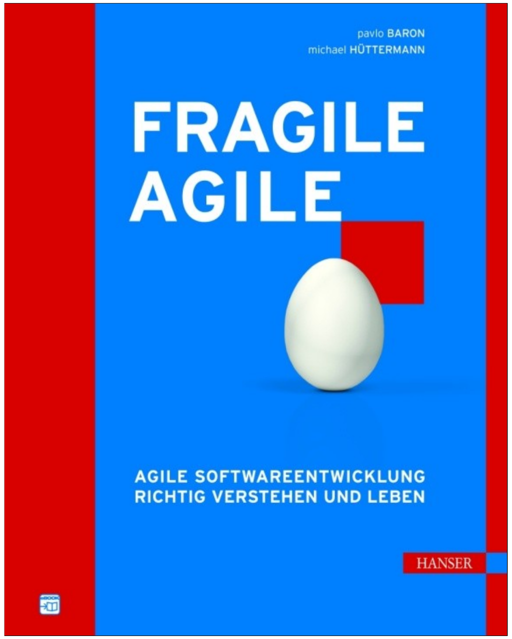
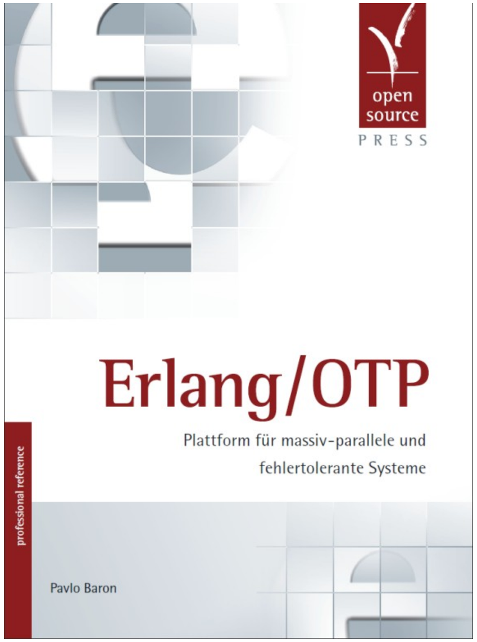
Pavlo Baron

Cruising for a bruising

entwickler.press
Pragmatische IT-Architektur
Schlanke Technologien und Konzepte für dynamische Unternehmen
Pavlo Baron





Geek's
Guide
To The Working Life

Pavlo Baron

pavlo.baron@pbit.org
@pavlobaron

This year I made a tour through German cities showing and explaining Erlang to Java people...

The most important thing I've learned was...

That I have to thank my dad for pulling me by my ears to the boxing club many years back.

And that it's good that I still practice from time to time

Let's consider the genesis

Me, I'm working with Java since 1999 or so.
I wrote a bunch of articles for
German Java magazines.

More or less accidentially I sank in the
Enterprise Java world...

If you're asking yourself what
he means by “Enterprise Java”...

I mean:

In order to have one simple OK-button
on a web page...

You need 10 hardly compatible frameworks,
3000 dependencies from similar libraries
in different versions in parallel,
15 layers,
300 classes,
200 interfaces,
50 factories,
25 factories of factories,
27 patterns (initially) and...

Loads of XML

After 5 weeks of heavy coding,
You're finally done with this thing.

But it's still not perfect –
the test branch coverage
is still under 100%

So you test-cover, refactor, abstract it,
further and further...
until the sponsor cancels
the project

As I started to dig deeper into Erlang,
announced my book,
tweeted / retweeted a lot about Erlang...

I've got unfollowed on Twitter
at least twice a day by
„Java guys”
(good thing I never
had a Facebook acount)

Talking to experts in the course of
German Java conferences,
as I told I'm digging into Erlang,

people sometimes told me...

“Why? Never heard about it.
Nobody uses it!”

"You can't do money
with it, so it's not
worth a thing"

„Oh, yes, Erlang...
It's more like
this Ruby and PHP stuff”

§$&%§”$%§$%$§”§/&%(/$” ???????

So I decided to do a tour
from one Conf / JUG to another
to show what Erlang really is

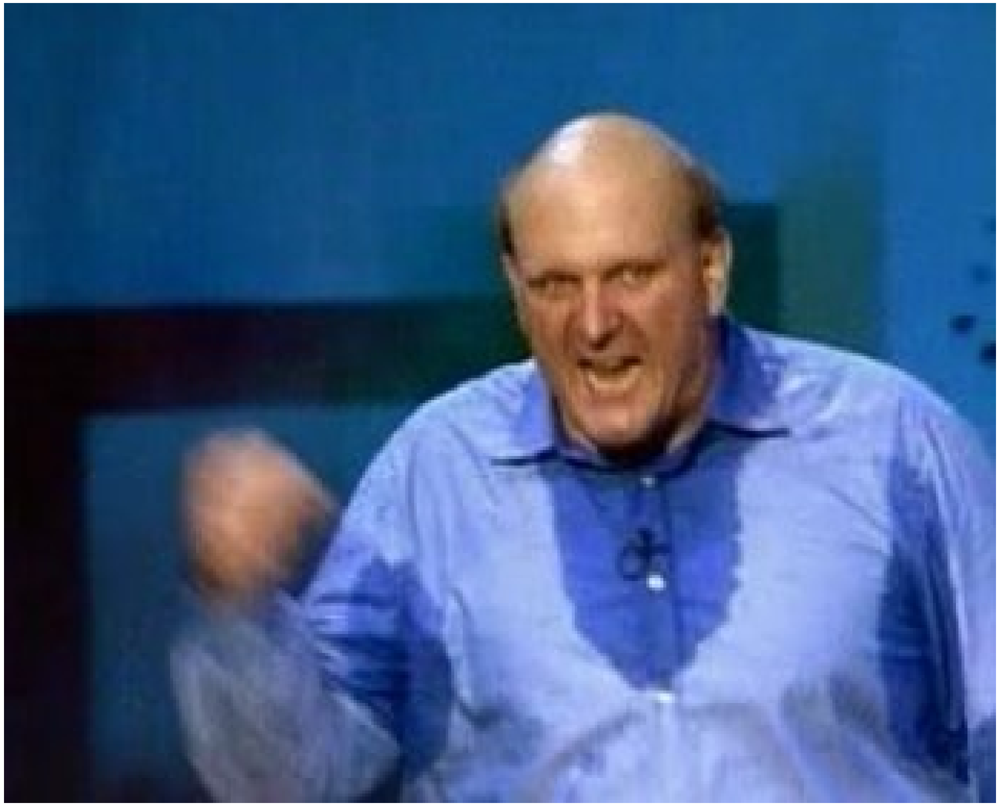
Some of the Confs & JUGs have
contacted me
to do a talk about architecture or so.
I contacted some of them myself.

In all cases I suggested to do
an Erlang talk...

I've never heard again
from some of them

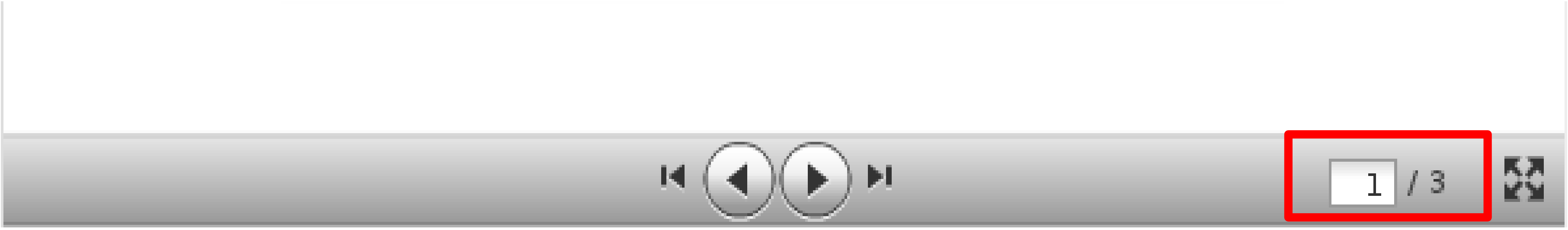
But many of them said „yes”

My first attempt was to explain Erlang through slideware...

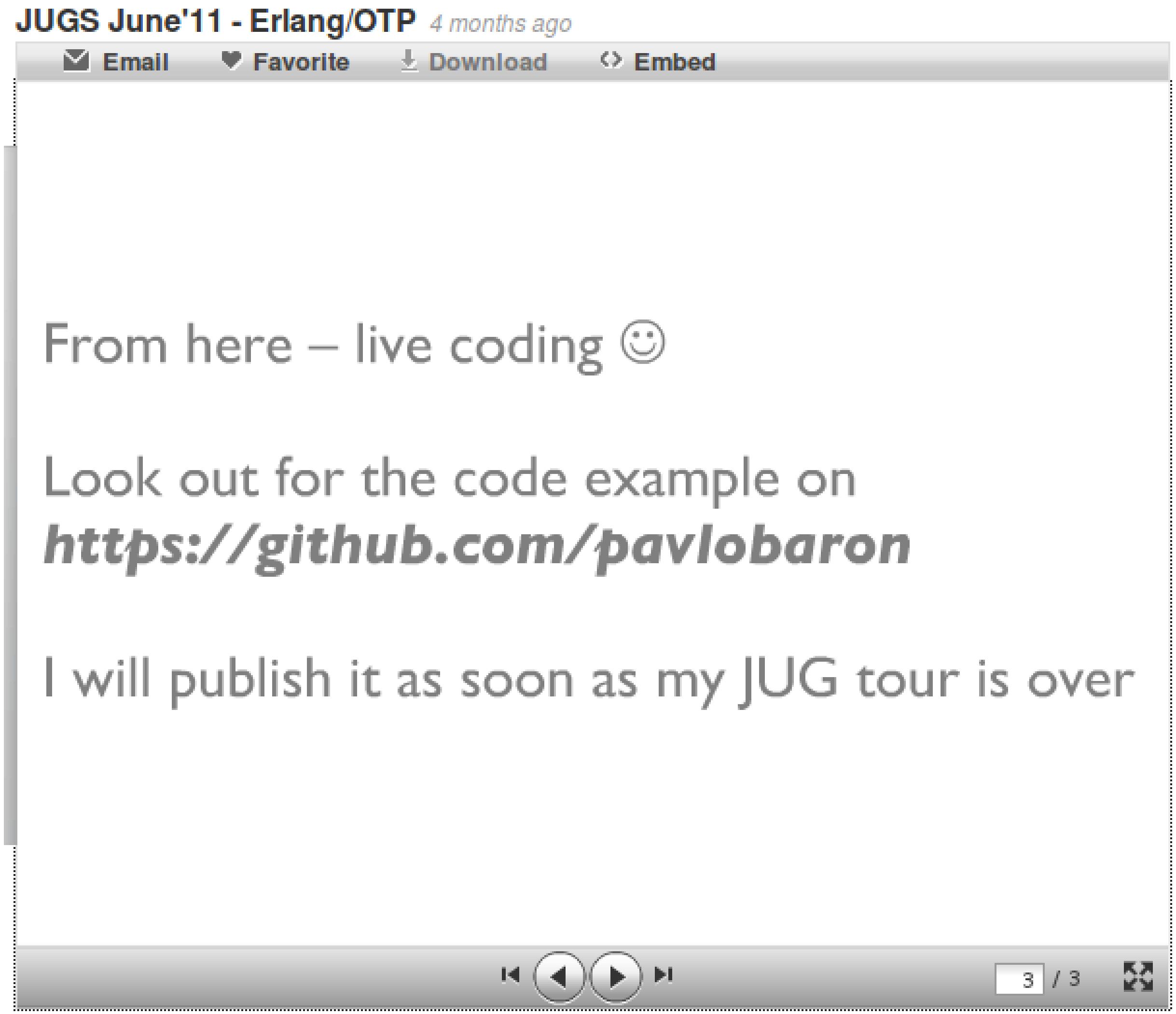


It's for the birds

So I reduced the number of slides a little



With this final slide

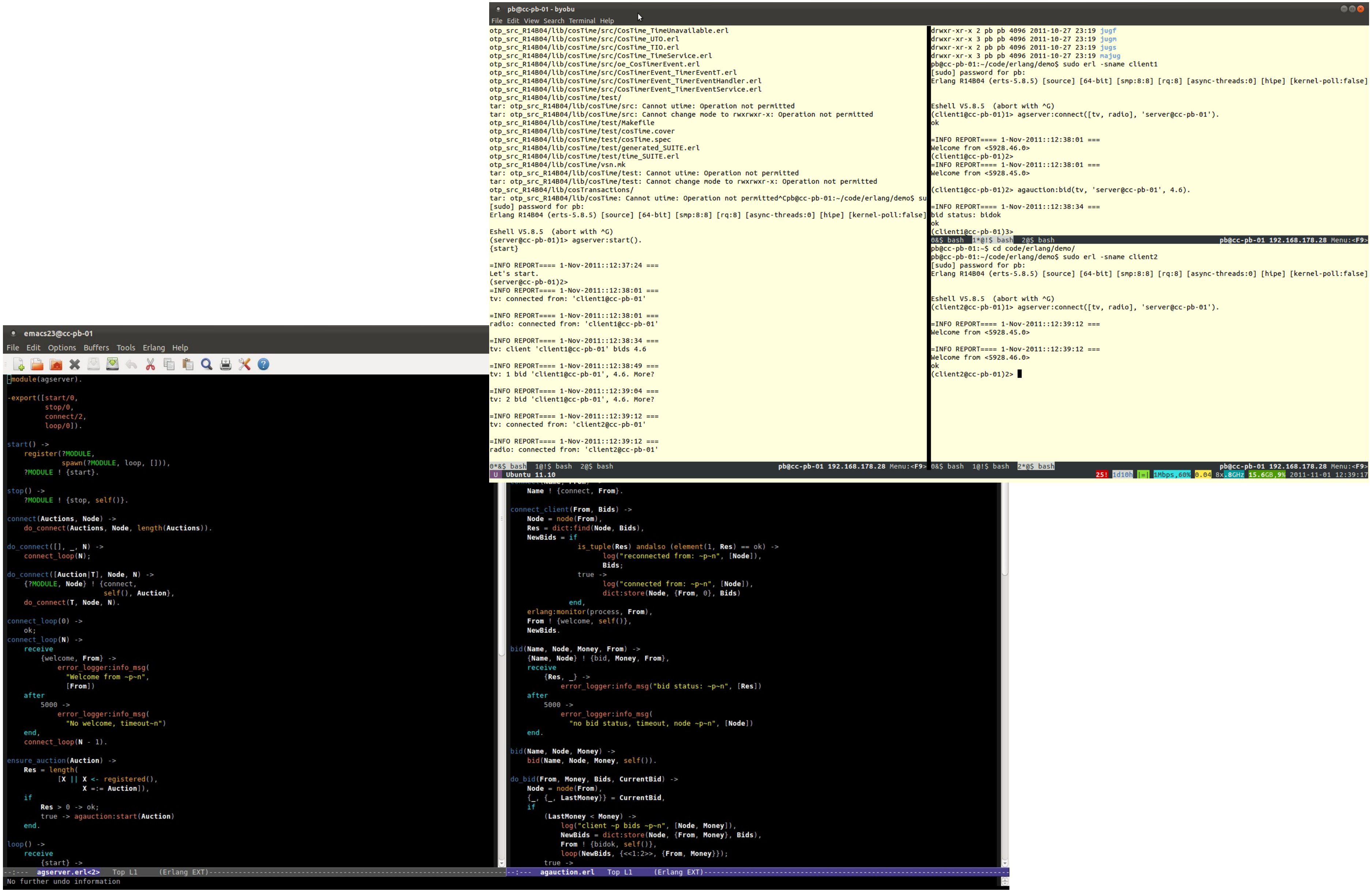


From here – live coding ☺

Look out for the code example on ***https://github.com/pavlobaron***

I will publish it as soon as my JUG tour is over

This was my equipment



```
otp_src_R14B04/lib/cosTime/src/CosTime_TimeUnavailable.erl
otp_src_R14B04/lib/cosTime/src/CosTime_UTO.erl
otp_src_R14B04/lib/cosTime/src/CosTime_TIO.erl
otp_src_R14B04/lib/cosTime/src/CosTime_TimeService.erl
otp_src_R14B04/lib/cosTime/src/oe_CosTimerEvent.erl
otp_src_R14B04/lib/cosTime/src/CosTimerEvent_TimerEventT.erl
otp_src_R14B04/lib/cosTime/src/CosTimerEvent_TimerEventHandler.erl
otp_src_R14B04/lib/cosTime/src/CosTimerEvent_TimerEventService.erl
otp_src_R14B04/lib/cosTime/test/
tar: otp_src_R14B04/lib/cosTime/src: Cannot utime: Operation not permitted
tar: otp_src_R14B04/lib/cosTime/src: Cannot change mode to rwxrwxr-x: Operation not permitted
otp_src_R14B04/lib/cosTime/test/Makefile
otp_src_R14B04/lib/cosTime/test/cosTime.cover
otp_src_R14B04/lib/cosTime/test/cosTime.spec
otp_src_R14B04/lib/cosTime/test/generated_SUITE.erl
otp_src_R14B04/lib/cosTime/test/time_SUITE.erl
otp_src_R14B04/lib/cosTime/vsn.mk
tar: otp_src_R14B04/lib/cosTime/test: Cannot utime: Operation not permitted
tar: otp_src_R14B04/lib/cosTime/test: Cannot change mode to rwxrwxr-x: Operation not permitted
otp_src_R14B04/lib/cosTransactions/
tar: otp_src_R14B04/lib/cosTime: Cannot utime: Operation not permitted^Cpb@cc-pb-01:~/code/erlang/demo$ su
[sudo] password for pb:
Erlang R14B04 (erts-5.8.5) [source] [64-bit] [smp:8:8] [rq:8] [async-threads:0] [hipe] [kernel-poll:false]

Eshell V5.8.5  (abort with ^G)
(server@cc-pb-01)1> agserver:start().
{start}

=INFO REPORT==== 1-Nov-2011::12:37:24 ===
Let's start.
(server@cc-pb-01)2>
=INFO REPORT==== 1-Nov-2011::12:38:01 ===
tv: connected from: 'client1@cc-pb-01'

=INFO REPORT==== 1-Nov-2011::12:38:01 ===
radio: connected from: 'client1@cc-pb-01'

=INFO REPORT==== 1-Nov-2011::12:38:34 ===
tv: client 'client1@cc-pb-01' bids 4.6

=INFO REPORT==== 1-Nov-2011::12:38:49 ===
tv: 1 bid 'client1@cc-pb-01', 4.6. More?

=INFO REPORT==== 1-Nov-2011::12:39:04 ===
tv: 2 bid 'client1@cc-pb-01', 4.6. More?

=INFO REPORT==== 1-Nov-2011::12:39:12 ===
tv: connected from: 'client2@cc-pb-01'

=INFO REPORT==== 1-Nov-2011::12:39:12 ===
radio: connected from: 'client2@cc-pb-01'

0*&$ bash  1@!$ bash  2@$ bash                    pb@cc-pb-01 192.168.178.28 Menu:<F9>
U  Ubuntu 11.10
```

```
drwxr-xr-x 2 pb pb 4096 2011-10-27 23:19 jugf
drwxr-xr-x 3 pb pb 4096 2011-10-27 23:19 jugm
drwxr-xr-x 2 pb pb 4096 2011-10-27 23:19 jugs
drwxr-xr-x 3 pb pb 4096 2011-10-27 23:19 majug
pb@cc-pb-01:~/code/erlang/demo$ sudo erl -sname client1
[sudo] password for pb:
Erlang R14B04 (erts-5.8.5) [source] [64-bit] [smp:8:8] [rq:8] [async-threads:0] [hipe] [kernel-poll:false]


Eshell V5.8.5  (abort with ^G)
(client1@cc-pb-01)1> agserver:connect([tv, radio], 'server@cc-pb-01').
ok

=INFO REPORT==== 1-Nov-2011::12:38:01 ===
Welcome from <5928.46.0>
(client1@cc-pb-01)2>
=INFO REPORT==== 1-Nov-2011::12:38:01 ===
Welcome from <5928.45.0>

(client1@cc-pb-01)2> agauction:bid(tv, 'server@cc-pb-01', 4.6).

=INFO REPORT==== 1-Nov-2011::12:38:34 ===
bid status: bidok
ok
(client1@cc-pb-01)3>
0&$ bash  1*@!$ bash  2@$ bash                    pb@cc-pb-01 192.168.178.28 Menu:<F9>
pb@cc-pb-01:~$ cd code/erlang/demo/
pb@cc-pb-01:~/code/erlang/demo$ sudo erl -sname client2
[sudo] password for pb:
Erlang R14B04 (erts-5.8.5) [source] [64-bit] [smp:8:8] [rq:8] [async-threads:0] [hipe] [kernel-poll:false]


Eshell V5.8.5  (abort with ^G)
(client2@cc-pb-01)1> agserver:connect([tv, radio], 'server@cc-pb-01').

=INFO REPORT==== 1-Nov-2011::12:39:12 ===
Welcome from <5928.45.0>

=INFO REPORT==== 1-Nov-2011::12:39:12 ===
Welcome from <5928.46.0>
ok
(client2@cc-pb-01)2>
0&$ bash  1@!$ bash  2*@$ bash                    pb@cc-pb-01 192.168.178.28 Menu:<F9>
          25! 1d10h |=| 1Mbps,60% 0.04 8x.8GHz 15.6GB,9% 2011-11-01 12:39:17
```

```erlang
-module(agserver).

-export([start/0,
         stop/0,
         connect/2,
         loop/0]).

start() ->
    register(?MODULE,
             spawn(?MODULE, loop, [])),
    ?MODULE ! {start}.

stop() ->
    ?MODULE ! {stop, self()}.

connect(Auctions, Node) ->
    do_connect(Auctions, Node, length(Auctions)).

do_connect([], _, N) ->
    connect_loop(N);

do_connect([Auction|T], Node, N) ->
    {?MODULE, Node} ! {connect,
                       self(), Auction},
    do_connect(T, Node, N).

connect_loop(0) ->
    ok;
connect_loop(N) ->
    receive
        {welcome, From} ->
            error_logger:info_msg(
              "Welcome from ~p~n",
              [From])
    after
        5000 ->
            error_logger:info_msg(
              "No welcome, timeout~n")
    end,
    connect_loop(N - 1).

ensure_auction(Auction) ->
    Res = length(
            [X || X <- registered(),
                  X =:= Auction]),
    if
        Res > 0 -> ok;
        true -> agauction:start(Auction)
    end.

loop() ->
    receive
        {start} ->
-:---  agserver.erl<2>   Top L1     (Erlang EXT)----------------------------------------------------------
No further undo information
```

```erlang
    Name ! {connect, From}.

connect_client(From, Bids) ->
    Node = node(From),
    Res = dict:find(Node, Bids),
    NewBids = if
                  is_tuple(Res) andalso (element(1, Res) == ok) ->
                       log("reconnected from: ~p~n", [Node]),
                       Bids;
                  true ->
                       log("connected from: ~p~n", [Node]),
                       dict:store(Node, {From, 0}, Bids)
              end,
    erlang:monitor(process, From),
    From ! {welcome, self()},
    NewBids.

bid(Name, Node, Money, From) ->
    {Name, Node} ! {bid, Money, From},
    receive
        {Res, _} ->
            error_logger:info_msg("bid status: ~p~n", [Res])
    after
        5000 ->
            error_logger:info_msg(
              "no bid status, timeout, node ~p~n", [Node])
    end.

bid(Name, Node, Money) ->
    bid(Name, Node, Money, self()).

do_bid(From, Money, Bids, CurrentBid) ->
    Node = node(From),
    {_, {_, LastMoney}} = CurrentBid,
    if
        (LastMoney < Money) ->
            log("client ~p bids ~p~n", [Node, Money]),
            NewBids = dict:store(Node, {From, Money}, Bids),
            From ! {bidok, self()},
            loop(NewBids, {<<1:2>>, {From, Money}});
        true ->
--:---  agauction.erl   Top L1     (Erlang EXT)----------------------------------------------------------
```

And this was my live hacking environment

Some reacted like…

§$&%§”$%§$%$§”§/&%(/$” ???????

Can I touch this?

Some reacted like…

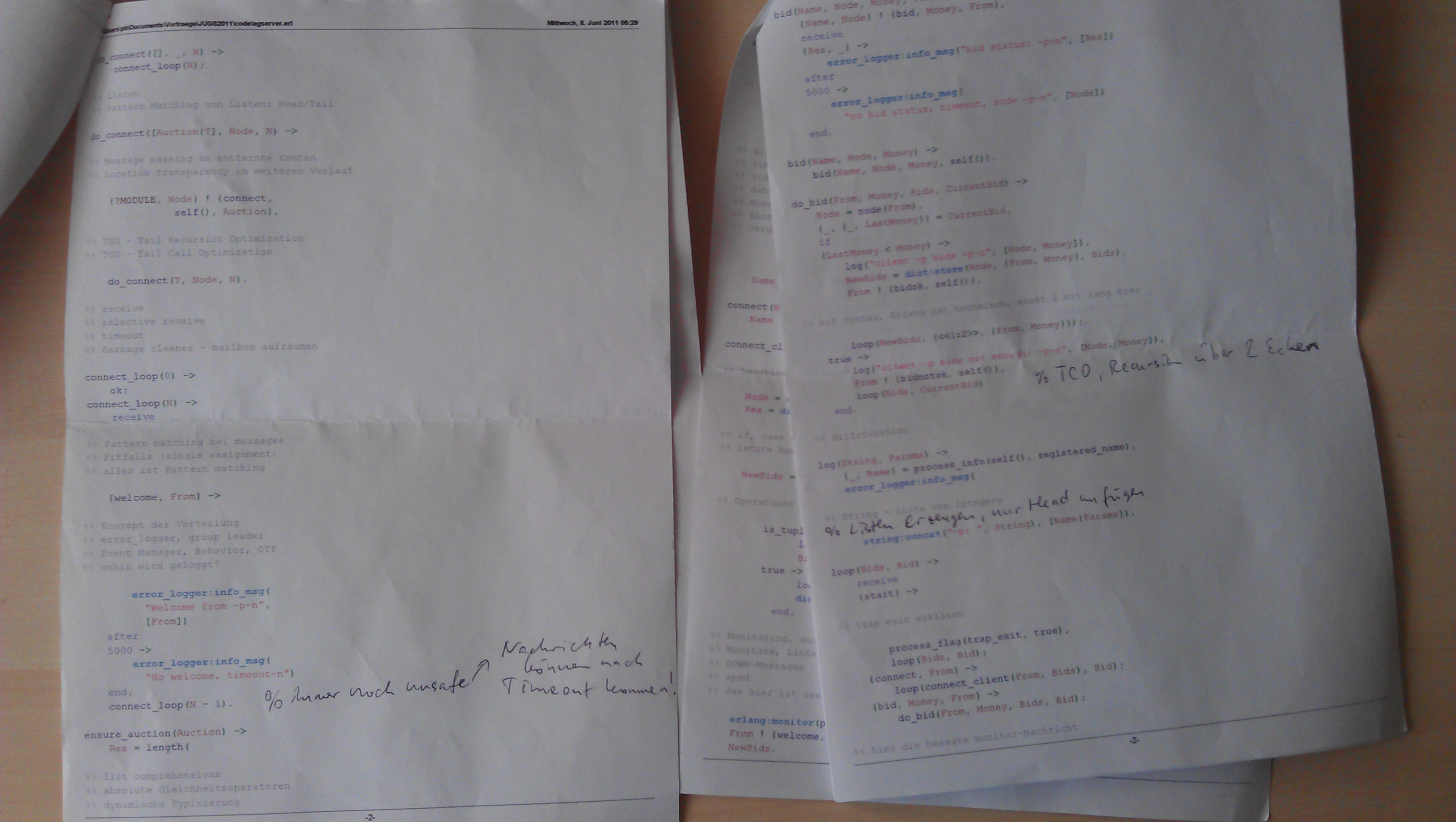
Looks nice. But this is so 80s...

Eclipse on a MacBook would also do the job, wouldn't it?

But the general reaction was like…

• • •

My example: a simple online auction



ers\pb\Documents\Vortraege\JUGS2011\code\agserver.erl — Mittwoch, 8. Juni 2011 05:29

```
do_connect([], _, N) ->
    connect_loop(N);

%% Listen
%% Pattern Matching von Listen: Head/Tail

do_connect([Auction|T], Node, N) ->

%% Message passing an entfernte Knoten
%% Location transparency im weiteren Verlauf

    {?MODULE, Node} ! {connect,
                self(), Auction},

%% TRO - Tail Recursion Optimization
%% TCO - Tail Call Optimization

    do_connect(T, Node, N).

%% receive
%% selective receive
%% timeout
%% Garbage cleaner - mailbox aufräumen

connect_loop(0) ->
    ok;
connect_loop(N) ->
    receive

%% Pattern matching bei messages
%% Pitfalls (single assignment)
%% alles ist Pattern matching

    {welcome, From} ->

%% Konzept der Verteilung
%% error_logger, group leader
%% Event Manager, Behavior, OTP
%% wohin wird geloggt?

        error_logger:info_msg(
          "Welcome from ~p~n",
          [From])
    after
    5000 ->
        error_logger:info_msg(
          "No welcome, timeout~n")
    end,
    connect_loop(N - 1).        % Immer noch unsafe ↗ Nachrichten können nach Timeout kommen!

ensure_auction(Auction) ->
    Res = length(

%% list comprehensions
%% absolute Gleichheitsoperatoren
%% dynamische Typisierung
```

-2-

```
bid(Name, Node, Money,
    {Name, Node} ! {bid, Money, From},
    receive
    {Res, _} ->
        error_logger:info_msg("bid status: ~p~n", [Res])
    after
    5000 ->
        error_logger:info_msg(
          "no bid status, timeout, node ~p~n", [Node])

    end.

bid(Name, Node, Money) ->
    bid(Name, Node, Money, self()).

do_bid(From, Money, Bids, CurrentBid) ->
    Node = node(From),
    {_, {_, LastMoney}} = CurrentBid,
    if
    (LastMoney < Money) ->
        log("client ~p bids ~p~n", [Node, Money]),
        NewBids = dict:store(Node, {From, Money}, Bids),
        From ! {bidok, self()},

%% bit Syntax, Erlang ist technisch, exakt 2 bit lang hier

        loop(NewBids, {<<1:2>>, {From, Money}});
    true ->
        log("client ~p bids not enough: ~p~n", [Node, Money]),
        From ! {bidnotok, self()},
        loop(Bids, CurrentBid)        % TCO, Recursion über 2 Ecken
    end.

%% Hilfsfunktion

log(String, Params) ->
    {_, Name} = process_info(self(), registered_name),
    error_logger:info_msg(

%% String = Liste von integers        % Listen Erzeugen, nur Head anfügen

      string:concat("~p: ", String), [Name|Params]).

loop(Bids, Bid) ->
    receive
    {start} ->

%% trap exit erklären

        process_flag(trap_exit, true),
        loop(Bids, Bid);
    {connect, From} ->
        loop(connect_client(From, Bids), Bid);
    {bid, Money, From} ->
        do_bid(From, Money, Bids, Bid);

%% hier die besagte monitor-Nachricht
```

-2-

```
%% di
%% Si
%% Bid
%% dah
%% Mon
%% Eins
%% Verg

    Name

connect(N
    Name

connect_cl

%% Location

    Node =
    Res = d

%% if, case
%% letzte An

    NewBids =

%% Operatoren

        is_tupl
            l
            B
        true ->
            lo
            dic
        end,

%% Monitoring, au
%% Monitors, Links
%% DOWN-Messages
%% epmd
%% das hier ist das

    erlang:monitor(p
    From ! {welcome,
    NewBids.
```

I prepared some cheet sheets and started

The typical course of the talk was…

0′ – 5′

Me.
Narcissistic

6′ – 15′

Doing my huge slideware. Still narcissistic

16′ – 20′

Realizing that I didn't bring up the environment.

Staying totally cool

21´ – 30´

Finally, the environment is set.

But the colors and display resolution need to be ajusted.

Still absolutely cool

31′ – 37′

I'm starting to show and explain the example and the concepts: processes, message passing distribution etc.

Looking very competent

38′ – 40′

First 2–3 guys stand up and leave the room.

I'm trying to joke about it

41´ – 60´

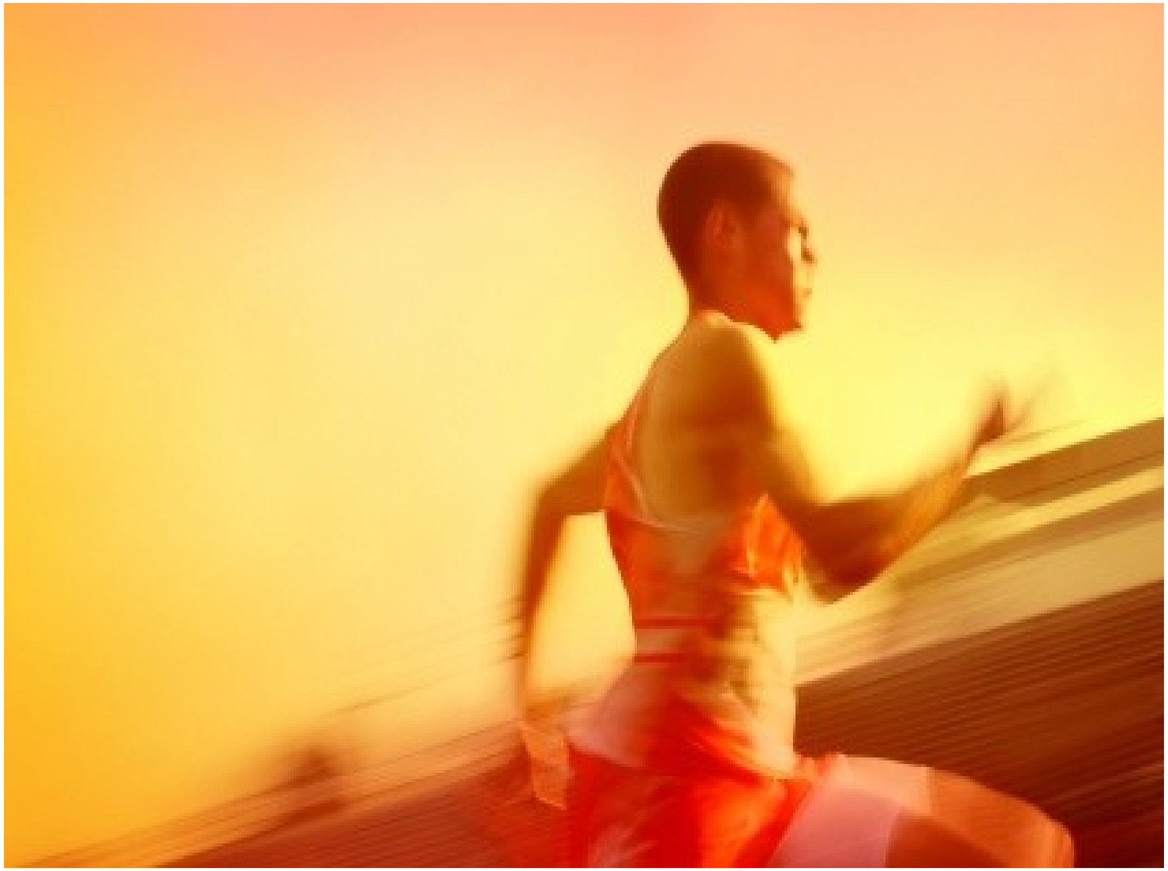
Of course, some random things go wrong. Every time different ones.

But I'm cool, I'm cool

61´ – 62´

I'm completely
out of time. Checking if we have
half an hour more. Some more guys
stand up and leave

I'm cold as ice...



63′ – 90′

Finishing my example.
Seeing some guys close their eyes.
Some are just silent. The others look interested.

Doing it very slowly

90´ – 120´

I'm done.
Now the official discussion starts.

I'm as fit as a fiddle.
We all are pretty fit.

Questions, answers and reactions

Q: why is the syntax so horrible?

A:

```
[X || X <- lists:seq(1, 10), X rem 2 == 0]
```

Now count in your head
the number of lines in Java for that.

Even when you do it as a one-liner...

Reaction

Q: wait, it's not a pure functional language? It doesn't have monads? Even Haskell does!

A:

I'm sorry, I'm absolutely no expert
with this topic. You do use
Haskell in production systems,
don't you? No? Hm. Ok, I'm sorry again,
I'm no expert with it...

Reaction
(it's ok)

Q: wait, the language is dynamically typed? Isn't it dangerous?

A:

it is, of course. But only if you're
doing blind and not checking upfront.
And it's not more dangerous
than throwing null objects
into methods

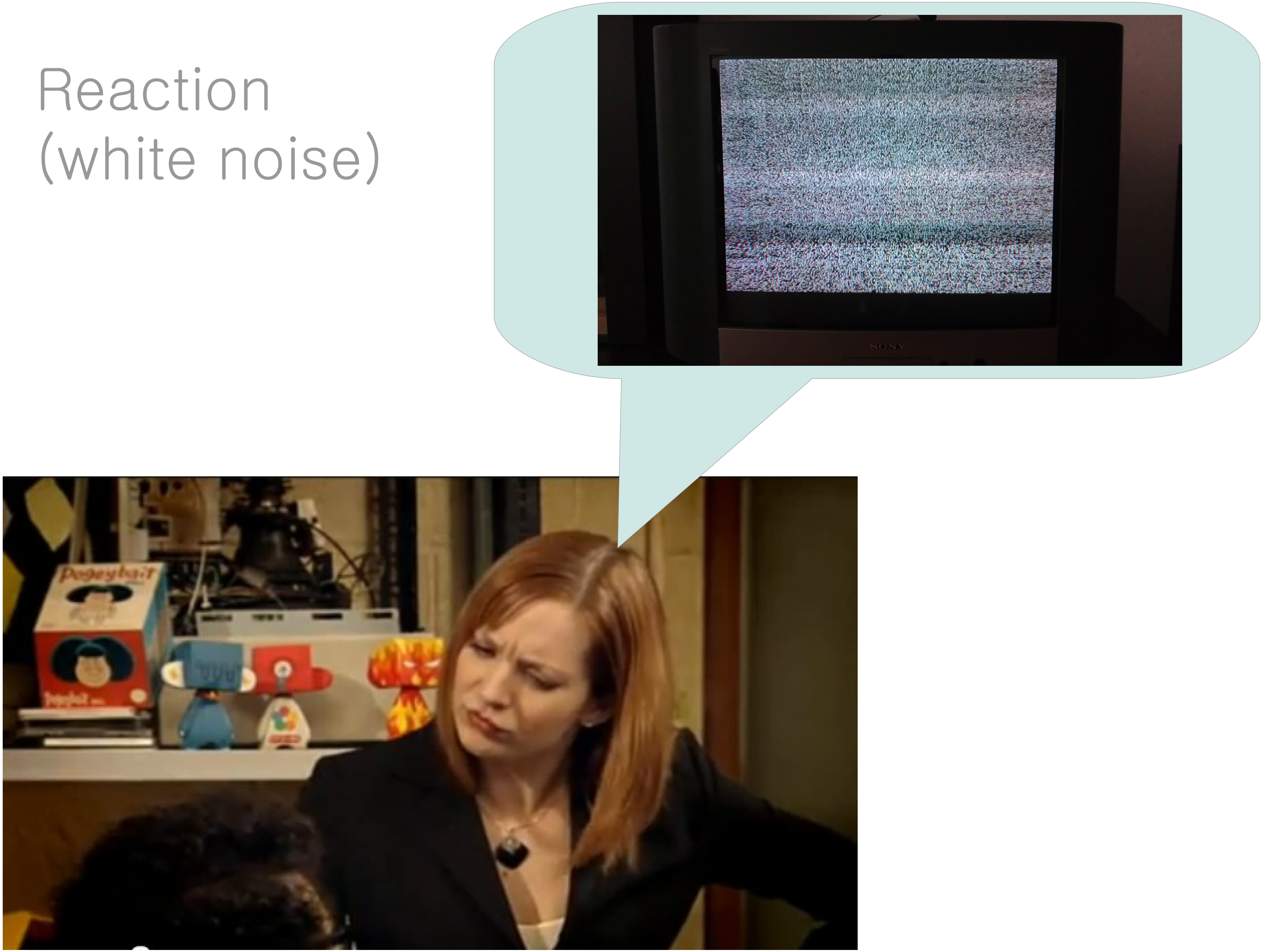
Reaction

Q: does the VM have a garbage collector and all the optimizations the JVM has?

A:

Not all of them, but some. It has one straightforward copying generational GC instead of ~10 GCs of the JVM. You don't have a JIT compiler. But you have HiPE, schedulers on top of threads, run queues, event based I/O...

Reaction (white noise)



Q: does it perform well?

A:

It's for high-performance,
massive parallel solutions.
Millions of processes and events are
possible in parallel.
Such impressive numbers you might know
from LMAX on the JVM.
It's all about mechanical sympathy

Reaction
(white noise,
not my
use cases)

Q: can I do parallel programming with it?

A:

Sure, the concept of an actor (process) is integral component of the platform. Even beyond the borders of one single machine you can benefit from the location transparency and do work distribution

Reaction
(not really
impressive – the
JVM also gives
me comfortable
threads and
threading models)

Q: does it integrate with other technologies?

A:

Of course, you have several layers
of integration, depending on how native
you want to be. It even runs with and on
the JVM.
Often, a bigger
Erlang solution also contains
many lines of C code

Reaction
(yes, I remember.
I did some C
during my
studies)

Q: how and where does Ericsson use Erlang these days?

A:

§$&%§”$%§$%$§”§/&%(/$” ???????

How can I know this?

Reaction
(it's ok,
you loser)

Why do you tell us all this?

Let me tell you what
I've really learned during
my tour

From my experience:

In Germany, you can learn and like what ever language or platform you want. It doesn't matter which...

'Cause you're likely to
end up doing
Enterprise Java

So many around who don't want to change it

Programming language can't score,
since when they want another one,
then they want one which
continuously tickles the brain

Platform, concurrency and distribution can't score since not many have use cases for that

Technology can't score since
not many want to look that deep
and instead blindly trust in the JVM. Even not
everyone can strictly separate
Java from the JVM

Library can't score since
Enterprise Java owns
the black belt with stars
in library production

Exact explanation of typical Erlang
use cases and specialization
can score,
though it would mean to
lose contact to the
big fat enterprise business

Pushing Erjang can score.
But Scala/Akka are
likely to win this race. Of
course only for the cases where
JVM is a must

Going to universities and
forming of young people's
opinion and winning them
over to Erlang can score.
Though it's nothing
that will work immediately

Cool stuff like Riak, Couch, Rabbit, online gaming backends can score. Cool stuff sells

Road shows can score

There is a need for an image change (?)

Thank you

images originate from:
istockphoto.com
wikipedia
product pages
direct movie shots