> I have been asked to hold a talk at a Smalltalk conference <gulp>

well, you've been challenging them for some time now (object- vs
concurrency-oriented programming) <:-)>

> So I have to convince a conference room full of Smalltalkers to
> become Erlangtalkers.

if there's a room full of hackers, you might be able to use that
to your advantage: describe the salient features of Erlang/OTP.
then challenge them to add an "Erlang/OTP" class to their favourite
Smalltalk system (Squeak?). what are the fundamental issues
they will run into and need to take care of? how does Erlang
solve these?

I'm not a Smalltalker myself, but I doubt that the usual "Erlang is
the better language" approach is going to be fruitful in that kind of
community. You do *not* have to convince them to become
Erlangers, nor would I rate your chances highly if you tried.

You ought to be able to convince them that Erlang represents
certain ideas whose time has come, tackling issues that Smalltalks
may not have addressed yet (check carefully before making such
claims!-), but will have to address sooner rather than later, too.

In other words, accept that they'll keep doing things the
Smalltalk way, but convince them that there are problems they
need to take care of, and that Erlang represents one coherent
and well-tested set of solutions.

What might work is to look at what current Smalltalks do well,
check what Erlang does better, and then to highlight the differences
in terms of areas that Smalltalkers may not have focussed on yet
(multithreading/distribution/fault-tolerance?). You'll probably find
lots of things that Erlang does not do as well as Smalltalk (guis/
reflection/ease of tool-building?). Be honest about those, too..

just as Erlang has limitations that make some tasks more difficult
than in Smalltalk (reflection/meta-programming, for instance), so
Smalltalk has limitations that make some tasks more difficult than
in Erlang. Working out what these limitations are, and why they
are important, would give you something more welcome than the
usual sales talk for a not-invented-here language: a well-informed
talk for a well-informed audience, addressing real issues and
offering intelligent appropaches to their solution.

> As part of the lecture I have promised to hold a "5 second Erlang
> course"

I thought Erlang was soft realtime, why the artifical hard time limit?-)

The gold-standard for conciseness there might not be the feature
bulletpoint slide, nor the 5-second sales pitch, but the Lisp-in-Lisp
or Smalltalk-in-Smalltalk executable definitions. Can you give an
Erlang-in-Erlang implementation on a single page of code?

> "To make a fault-tolerant system you need two computers because if you
> only have one and it crashes you're sunk so you'd better use Erlang cos

> all other languages suck."

you might as well not attend, then. perhaps they'll give you some credit
for bravado, but none for trying to understand their way of doing things.

by all means, highlight Erlang's successes in that area, but be careful to
understand Smalltalk's as well. eg., some decades ago, there was a famous
cover page depicting a Smalltalk system as a ship going to see while parts
of it were under (re-)construction (pre-dating Erlang's hot code replacement).

if you leave out the "cos all other languages suck" part, you have a point,
but I'd phrase it differently, presenting Erlang's ideas and contributions as
a natural extension and generalisation of Smalltalk's ideas and contributions
(to the multi-processing/distributed case, with implicit and hardware fault
tolerance, and with large teams of programmers when needed).

you need to figure out what they do already have, where Erlang offers
advantages, and where inherent Smalltalk features might get in the way
of exploiting these advantages (eg. is message passing in a typical
Smalltalk system these days loosely coupled and suited for distribution,
or tightly coupled imperative, update-everywhere, sequential?).

compare with this (from "The Early History Of Smalltalk", 1993 by Alan Kay):

> In computer terms, Smalltalk is a recursion on the notion of computer itself.
> Instead of dividing "computer stuff" into things each less strong than the
> whole--like data structures, procedures, and functions which are the usual
> paraphernalia of programming languages--each Smalltalk object is a recursion
> on the entire possibilities of the computer. Thus its semantics are a bit like
> having thousands and thousands of computer all hooked together by a very
> fast network. Questions of concrete representation can thus be postponed
> almost indefinitely because we are mainly concerned that the computers
> behave appropriately, and are interested in particular strategies only if the
> results are off or come back too slowly.
>
> http://www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Introduction.html

I have the feeling that this gives an almost ideal starting point: taking the
ideas as given, what happens if we insert some harsh reality:

- what if this recursion is not idealised, but real, with thousands and
  thousands of *concurrently operating* processes, possibly on many
  real computers, or on a single computer with multiple cores, or both?
- what if the network isn't "very fast"?
- what if some results do not come back at all?
- what if some of those computers fail or have to be rebooted?
- what if debugging a running system is a fine idea, but you simply
  can't afford to stop the world while you're doing it?

(enter Erlang)

In other words, present Erlang as a natural next step in thinking about
some of the same ideas from which Smalltalk emerged. Don't try to
turn Smalltalkers into Erlangers, but do try to make Smalltalkers think
about the issues Erlang addresses. Ideally, convince them to "borrow"
what is best about Erlang, instead of adopting conventional threads
(again, check carefully what they've got). Mix with your usual spiel
on objects vs processes, perhaps tie the loop back to Smalltalk's

roots in Simula's process libs..

just one partially informed outsider's view..

good luck,
claus

ps. I'd certainly like to see a video of this. I'm trying to imagine
    Alan Kay and Joe Armstrong in the same room, in violent
    agreement about how everybody else never seems to get it!-)