

Ask not what your Erlang
can do for you

 Klarna™

An Erlanger will fight to start Erlang projects, and make others start them as well

However, Erlang projects “can” fail

Oh No! People blame us
and Erlang
for the failure!

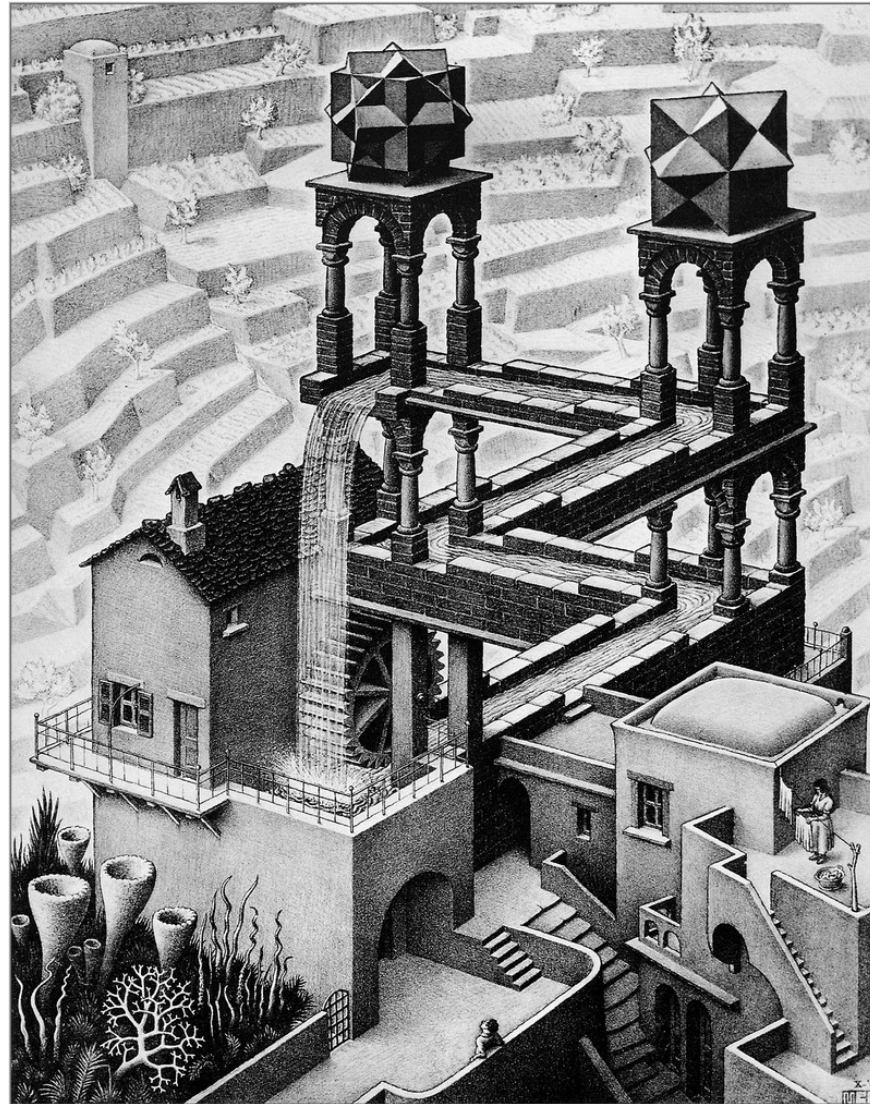
Using Erlang is not enough! Proper
engineering *might* also help

Imagine we have a golden opportunity, our company is going to start a new project to solve a big problem

The current version of the software
is a disaster

Written in a vastly inferior language

The code is so complicated that is unreliable and unmaintainable



Obviously, it is heavy and slow

We have to rebuild it in Erlang!

**It is fault tolerant! No more bugs in
production!**

Erlang scales to the infinity!

It is functional! We will add new features in hours! If not minutes!

And it has hot code loading! No
more maintenance downtime!

So you create an awesome piece of software in even less time than you promised!

... well... we've got some production
issues ...

And we might have problems
handling the expected load for this
year ...

About doing that change... it might
take a while...

Erlang is a *tool* for writing software,
but writing good software is still hard

Erlang *helps* building fault tolerant
systems

Can't we just “let it crash”?

Erlang will not design your error handling strategy for you, it just helps to implement it

It is not trivial to design a good
application and supervision
hierarchy

Erlang helps to use SMP CPUs
efficiently

Erlang gives you a very decent
model to utilise SMP CPUs
efficiently

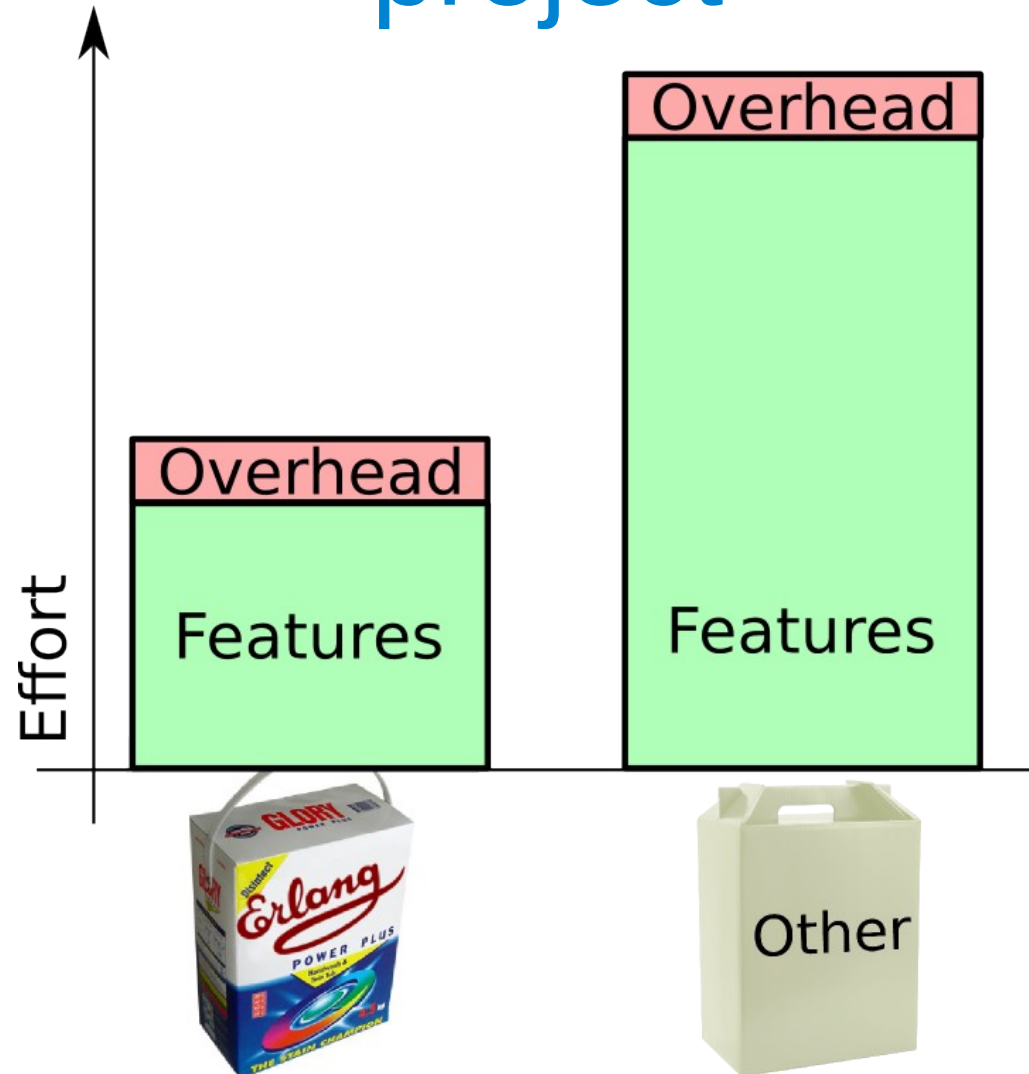
But today “scaling” usually
means scaling out

“Thinking Erlang” helps designing
scalable systems

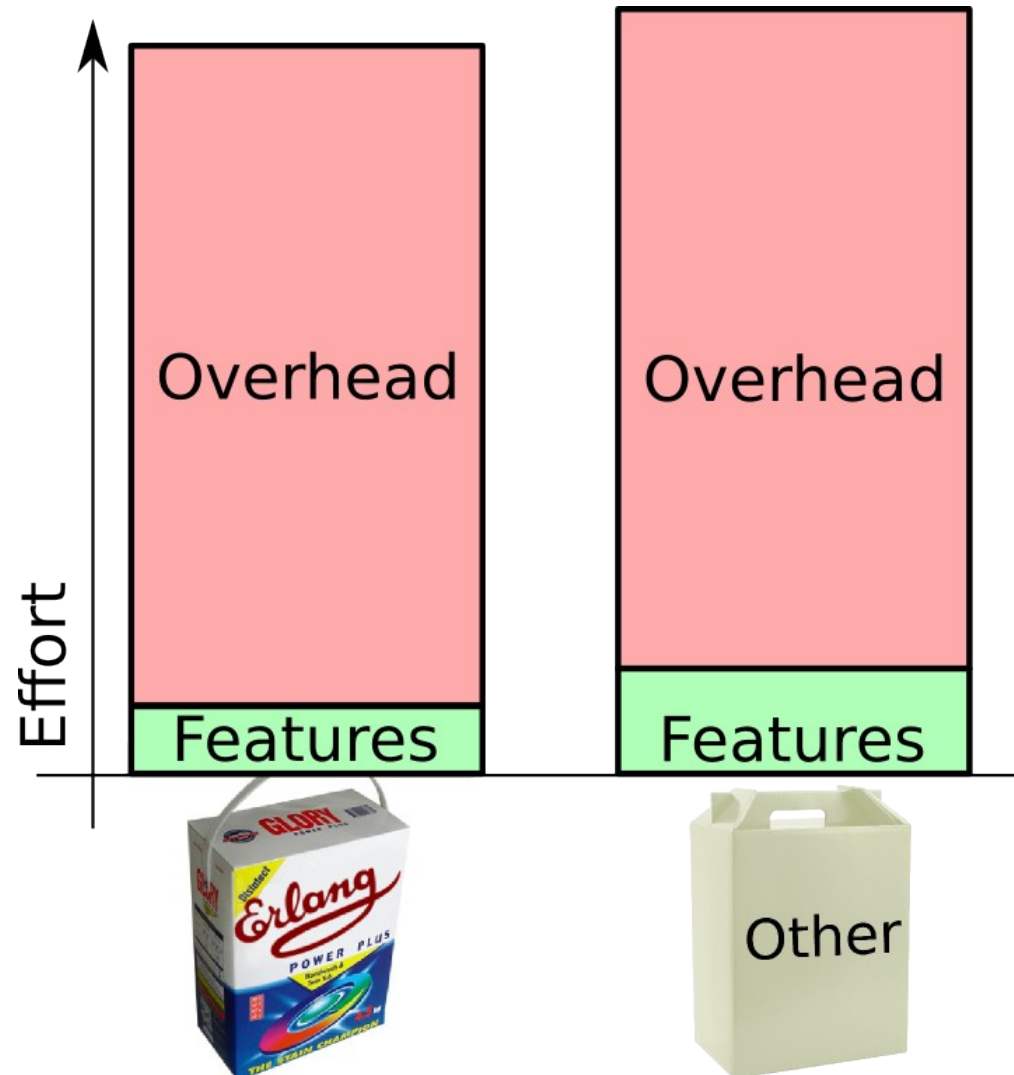
Erlang *helps* writing concise and
easy to understand code

Good developers write good code in
any language, and viceversa

The impact on productivity is significant at the beginning of the project



Coding speed becomes less relevant in the long term compared to other overheads



Hot Code Loading

It helps a lot when debugging

Implementing an application that is *always* running requires much more than just hot code loading

It is often simpler to write distributed applications where individual nodes can be taken down, rebooted, etc

There are Bad Things™ that we
need to avoid or mitigate

Dynamic Typing



Dynamic typing is not an advantage,
it is a compromise

As the software matures, you'll be fixing more and more bugs that are actually typing errors

Work your software so that it is
robust against type errors

Tooling is weaker than for more popular languages

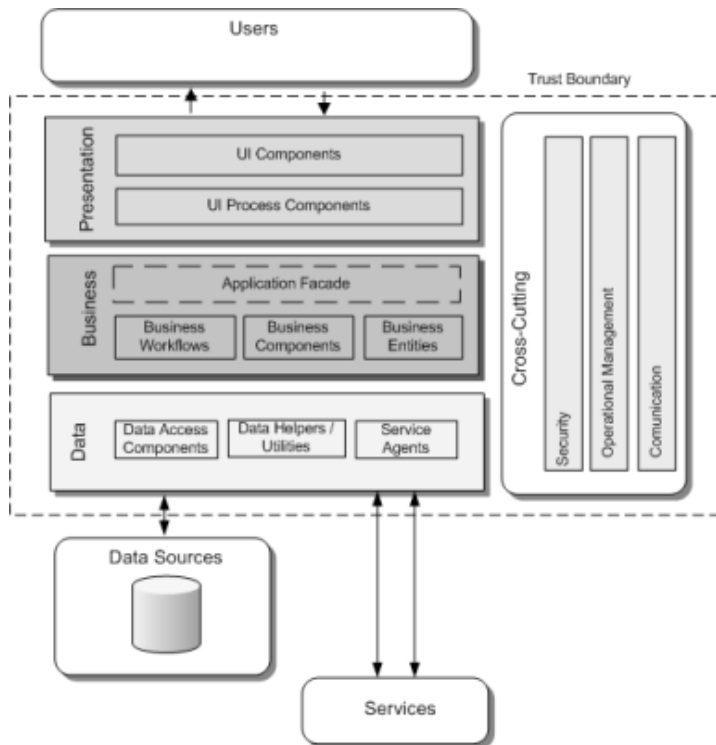
Library support might be immature
or simply missing

Releasing and packaging in the
large is typically painful

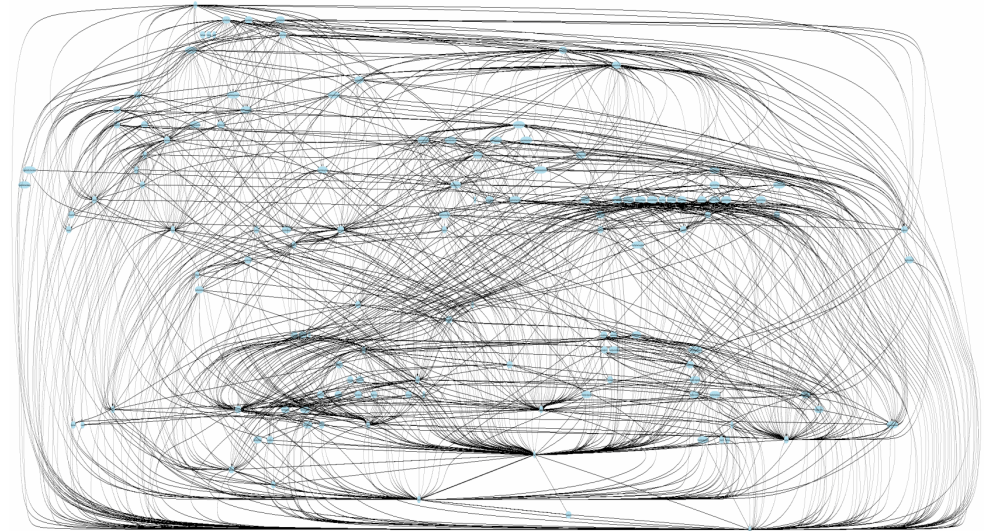
Rudimentary encapsulation

Erlang has a flat module name space, and only public/private visibility for functions

You'll need to be serious about architectural structure and encapsulation conventions



VS



You could use tools to prevent this
weakness to degenerate into code
rot

My fellow erlangers, ask not what
your Erlang can do for you. Ask
what you can do for your Erlang

Klarna™

About Klarna:

<http://engineering.klarna.com/>
SignMeUp@klarna.com

About me:

samuel.rivas@klarna.com
[@samuel_rivas](#)