ERICSSON

DEPLOYING AN EMBEDDED ERLANG SYSTEM

A case example

CONTENT

- What are Autotools and why use them?
- What is Yocto/Bitbake and why use it?
- A case example using Yocto/Bitbake to deploy an Erlang system
- Why is it difficult to deploy an Erlang system?
- How can we make it easy to deploy an Erlang system?

WHAT ARE AUTOTOOLS?

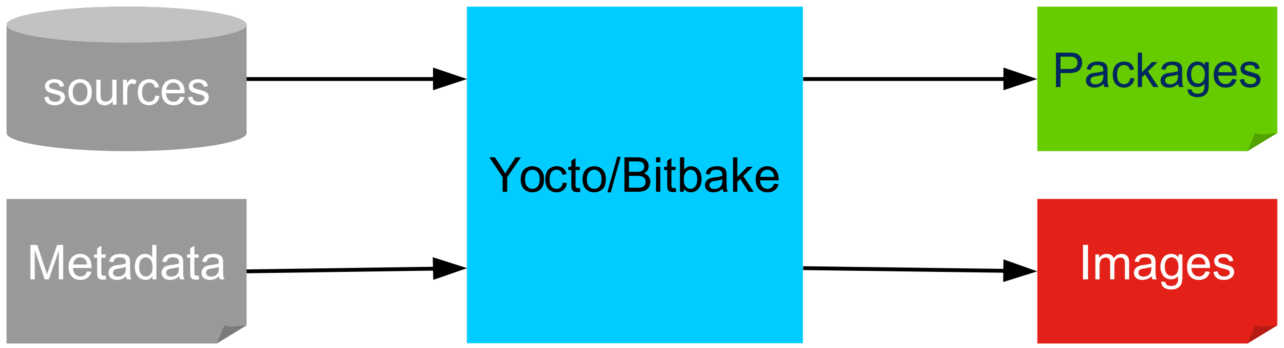
- The GNU Build System
  - designed to assist in making source code packages portable to many Unix-like systems
- Components:
  - GNU Autoconf
  - GNU Automake
  - GNU Libtool
  - Gnulib

WHY USE AUTOTOOLS?

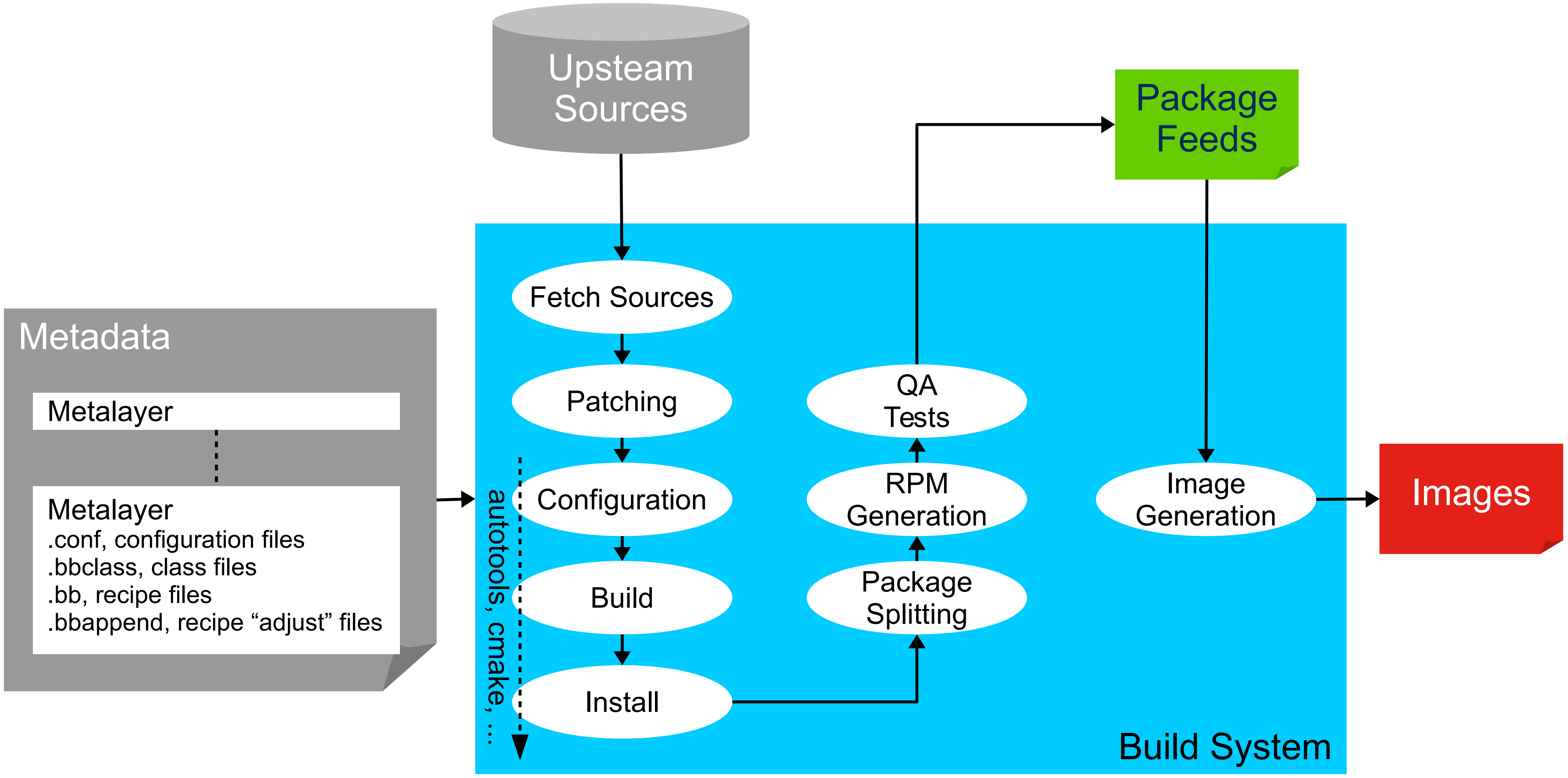
- Your code will be portable and easy to deploy on any Unix-like system
  - Without any extra effort on the users of user code
- Autotools adhere to the GNU Coding Standards
  - It is easy to make your code distributable
    - make dist, make distcheck
  - It is easy for others to build an install your distributed code
    - ./configure & make & make install
  - and so on ...

WHAT IS YOCTO/BITBAKE?

- Yocto is an open source collaboration project with:
  - templates, tools and methods for creating *embedded* linux products regardless of hardware architecture
- Bitbake is a generic task execution engine that:
  - allows shell and Python tasks to be run efficiently and in parallel while working within complex inter-task dependency constraints



THE YOCTO/BITBAKE FLOW



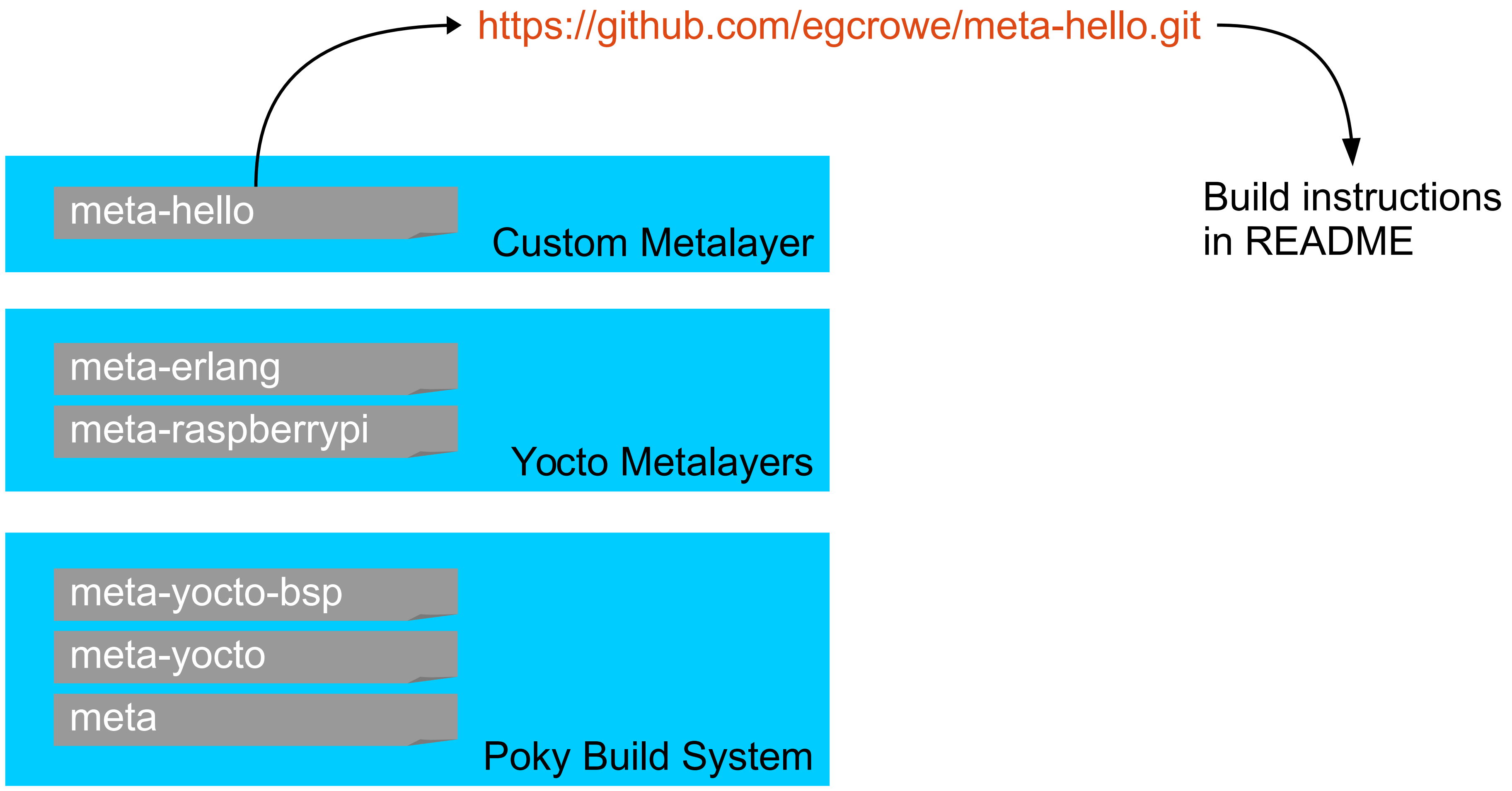
WHY USE YOCTO/BITBAKE?

- You want to deploy a customized *embedded* linux system
  - Other tools are available for server linux systems, chef, puppet, ...
- You want to automate the production of images and packages
  - Reproduceable, efficient system level builds
- You want to be able to deploy images and packages on multiple hardware architectures

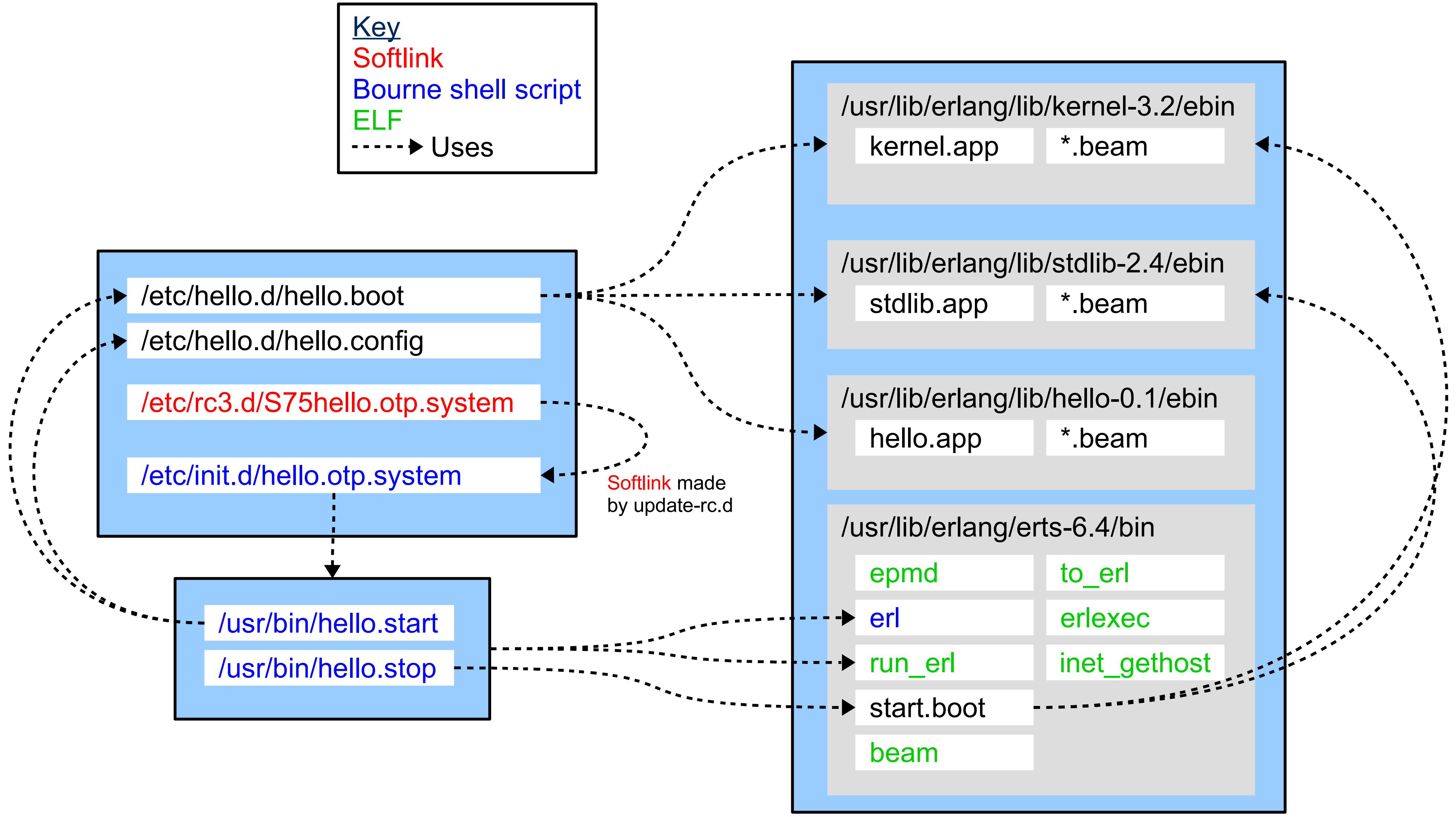
A CASE EXAMPLE, “HELLO”

- A system that periodically writes “Hello” to log file, /tmp/hello.log
- Minimal code that adheres to the OTP Design Principles
- Minimal deployment of files on target images
- Bourne shell scripts for bootstrapping this Erlang system with the Linux init system (System V)
  - Embedded Systems User Guide

BITBAKE METALAYERS



164 FILES DEPLOYED ON IMAGE


Key
Softlink
Bourne shell script
ELF
Uses
/etc/hello.d/hello.boot
/etc/hello.d/hello.config
/etc/rc3.d/S75hello.otp.system
/etc/init.d/hello.otp.system
Softlink made by update-rc.d
/usr/bin/hello.start
/usr/bin/hello.stop
/usr/lib/erlang/lib/kernel-3.2/ebin
kernel.app
*.beam
/usr/lib/erlang/lib/stdlib-2.4/ebin
stdlib.app
*.beam
/usr/lib/erlang/lib/hello-0.1/ebin
hello.app
*.beam
/usr/lib/erlang/erts-6.4/bin
epmd
to_erl
erl
erlexec
run_erl
inet_gethost
start.boot
beam

ERLANG/OTP INSIGHTS

- Erlang/OTP packaging is monolithic
  - No distinction between runtime system, libraries, tools and applications
- Erlang/OTP is not autotools compliant
  - This explains why Erlang/OTP is not as ubiquitous as it ought to be
- These root problems propagate down the chain
  - Meta-erlang is more complex due to Erlang/OTP being monolithic and non-Autotools compliant

META-ERLANG INSIGHTS

- João Henrique Ferreira de Freitas has made a great contribution
- The monolithic Erlang/OTP is split into smaller packages
  - The erlang package depends on erlang-erts, erlang-stdlib, erlang-kernel and erlang-sasl
  - However the contents of the erlang and erlang-erts packages are wrong in my opinion
- There are cross compiling workarounds to solve the non-autotools compliance problem
- Includes tools widely used in the Erlang community
  - rebar, relx, erlinit, ...

AUTOCONF INSIGHTS

- Romain Lenglet gave a EUC presentation in 2006 about new Erlang specific autoconf macros
- These are very useful
- I always use these macros for any Erlang code I now write
- These have helped me considerably and I have had no issues, until I started using bitbake
  - Unfortunately not all Erlang specific autoconf macros are “cross-compile” safe

AUTOMAKE INSIGHTS

- Erlang code is often enhanced with C code
- Autotools provides support for packaging C code for portability across Unix-like systems
- Automake ensures the build system adheres to the GNU Coding Standards
- For these reasons it is worth using Automake for Erlang code
- It is possible to write Makefile.am files for Erlang code
- However it can be tricky for beginners

SIMPLIFICATION ROADMAP

- Unravel the Erlang/OTP monolith
  - Enabler for simplifying meta-erlang Yocto metalayer
- Document recommended packaging and distribution practices
- Fix broken Erlang autoconf macros that are not cross compile safe
- Fix meta-erlang Yocto metalayer package splitting (erlang & erlang-erts)
- Design and implement automake primaries for erlang
- ...

ERICSSON