

BUILDING WPE



FOR AN EMBEDDED DEVICE

Supported hardware, WPEBackends, BSPs

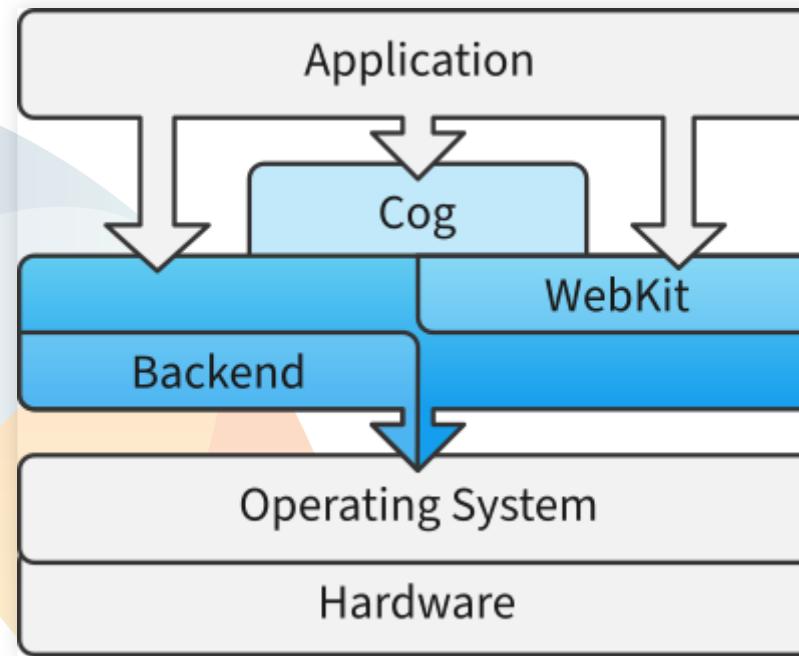
clopez@igalia.com

psaavedra@igalia.com

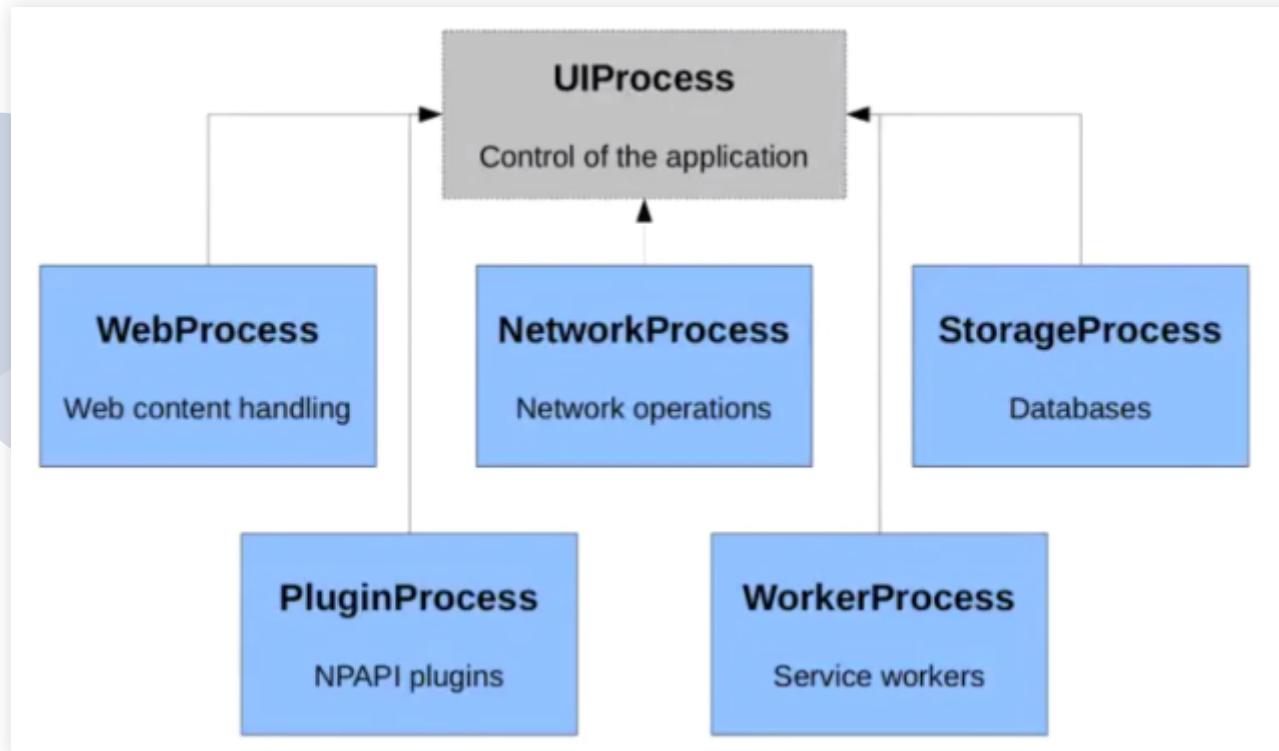
<https://people.igalia.com/psaavedra/slides/webengines-hackfest-2021-wpe-embedded>

WPE ARQUITECTURE

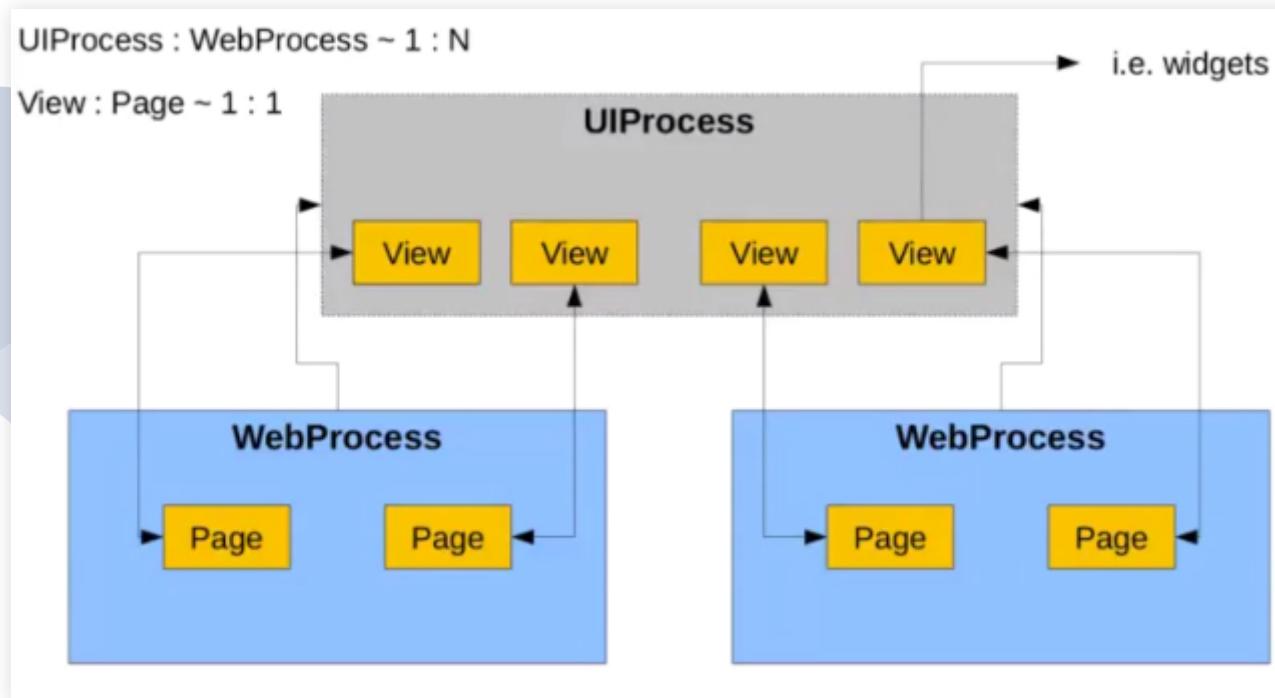
WPE STACK



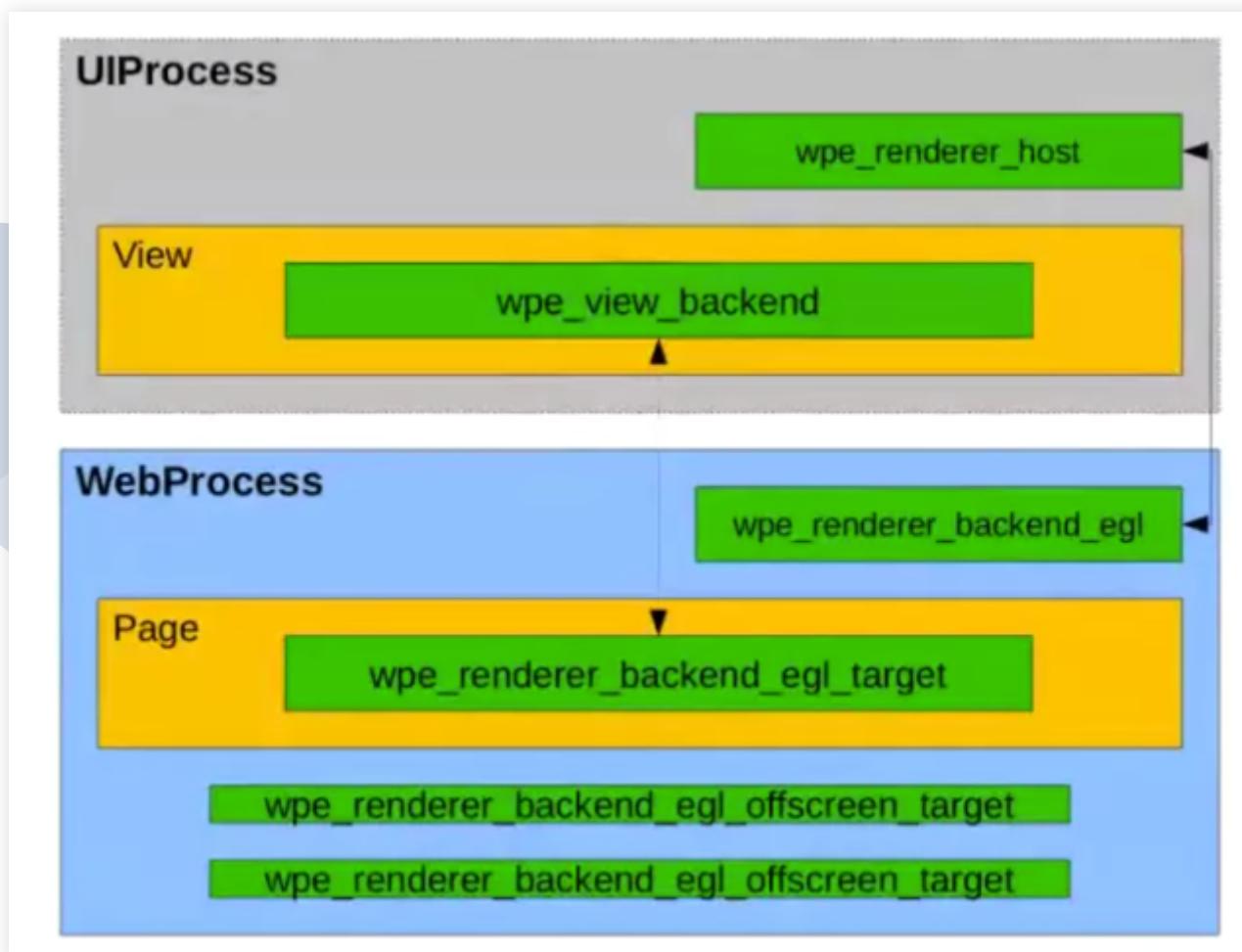
WEBKIT PROCESS MODEL (1/2)



WEBKIT PROCESS MODEL (2/2)



LIBWPE INTERFACES



WPE BACKENDS: LIBWPE IMPLEMENTATIONS

- Used by the WPE port
- Provides the implementation of the interfaces defined by the `libwpe` for rendering and input handling
- Sets EGL resources as requirement for the graphical output consumption (OpenGLv2)
- Several implementations but the most relevant are:
`wpebackend- rdk`, `wpebackend- fdo`, ...

WPEBACKEND-RDK

- RDK is a Set-top boxes consortium
- Covers different STB hardware and prototype boards
- Uses a proprietary API (Dispmanx) to lowest level access to the GPU
- It is supported by the proprietary RPi Broadcom driver

WPEBACKEND-FDO

- Uses Wayland protocol to coordinate the operations among the interface implementations
- Depends on the Wayland EGL support (`EGL_WL_bin_wayland_display`)
- Relies in GLib as IPC mechanism for communication in between the host and the backend
- In theory, compatible with any Mesa driver implementation

- Conclusion 1: Several libs combinations (`libwpe`, `libwebkit`, `cog` ...) and several backend implementations that makes a bit difficult the setup.

WEBKIT'S JAVASCRIPT (JSC) SUPPORT

- Depends on the CPU architecture
- Fully operational for JSC: armv7, arm64, x86
x86_64, mips32
- With limitations for 32bits architectures: *JIT and WebAssembly are disabled.*
- Other architectures risc-v, mips64, powerpc
... not supported

- **Conclusion 2:** WPE configuration is sensitive to the underlying hardware and software stack where it has to work

WHAT MAKES A HARDWARE PLATFORM

- System-on-Chip (SoC)
- GPU
- CPU

EXAMPLE (1/3): NXP I.MX 6

- SoC: i.MX6Q
- GPU: Vivante GC2000 / GC320
- CPU: NXP i.MX 6 - Cortex-A9 - quad-core

EXAMPLE (2/3): RASPBERRY PI 4 B

- SoC: BCM2711B0
- GPU: Broadcom VideoCore VI 500MHz
- CPU: A72 - quad-core

EXAMPLE (3/3): QUALCOMM

- SoC: APQ8017
- GPU: Adreno 306
- CPU: Qualcomm - Cortex-A53 CPU

... MORE SUPPORTED HARDWARE

wpewebkit.org/about/supported-hardware

- Conclusion 3: WPE works in the top of several multiple different hardware platforms

BOARD SUPPORT PACKAGE (BSP)

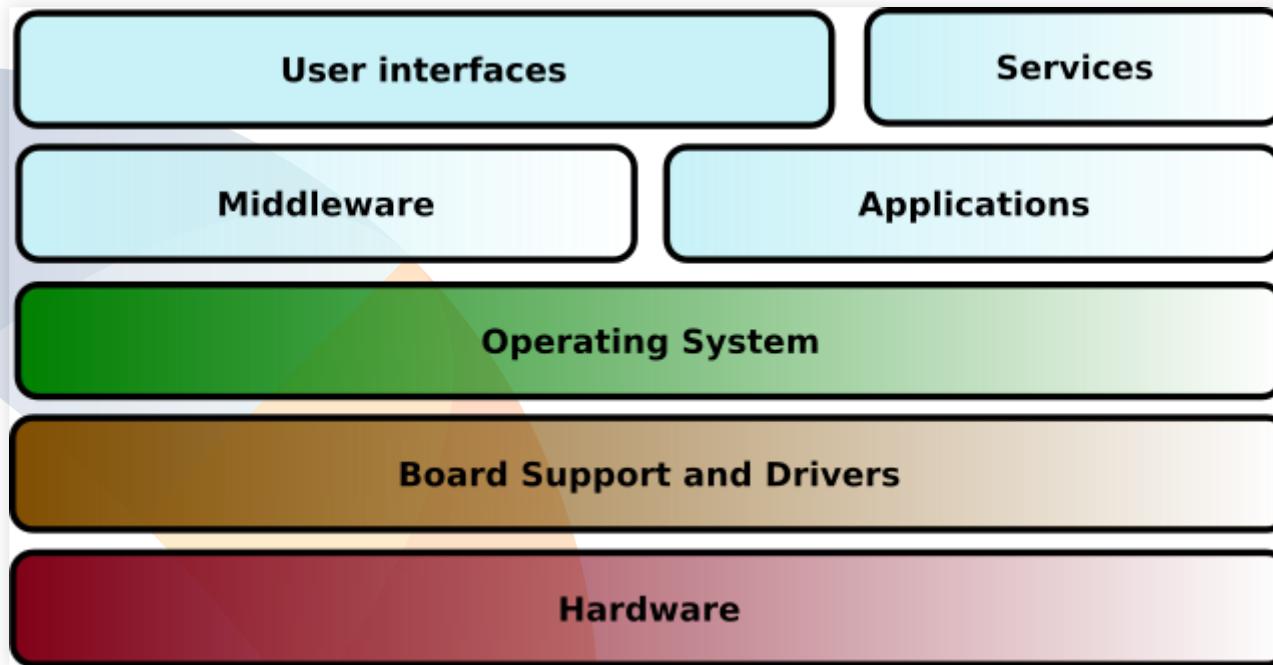
BOARD SUPPORT PACKAGE (BSP)

- **Problem 1:** Several libs and dependencies that makes a bit difficult the the setup.
- **Problem 2:** WPE is sensitive to the underlying hardware and software stack where it has to work
- **Problem 3:** WPE works in the top of several multiple different hardware platforms
- **Solution:** A software layer that enables an hardware-specific platform: BSP

BOARD SUPPORT PACKAGE (1/3)

- Bootloader and Linux kernel
- SoC operative system support:
 - SoC support (peripherals, storage, network, ...)
 - Graphics stack support
- Userspace tools and interfaces
- WebKit stack:
 - `libwpe`
 - WPE backend implementation
 - WebKit WPE runtime
 - WPE browser (`cog`)

BOARD SUPPORT PACKAGE (2/3)



BOARD SUPPORT PACKAGE (3/3)

- Assembling all the user space components needed for the system, configure them, develop the upgrade and recovery mechanisms, etc.
- Application development: write the company-specific applications and libraries.
- Building from source
- Cross-compilation
- Recipes for building components

YOCTO VS BUILDROOT

- **Yocto/OpenEmbedded:**
 - Builds a full Linux distro with binary pkgs.
 - Powerful, but somewhat complex, and quite steep learning curve.
- **Buildroot:**
 - Builds a root filesystem image, no binary pkgs.
 - Much simpler to use, understand and modify.
 - WPE recipe in upstream buildroot (thanks [aperezdc](#)!)

YOCTO (1/2)

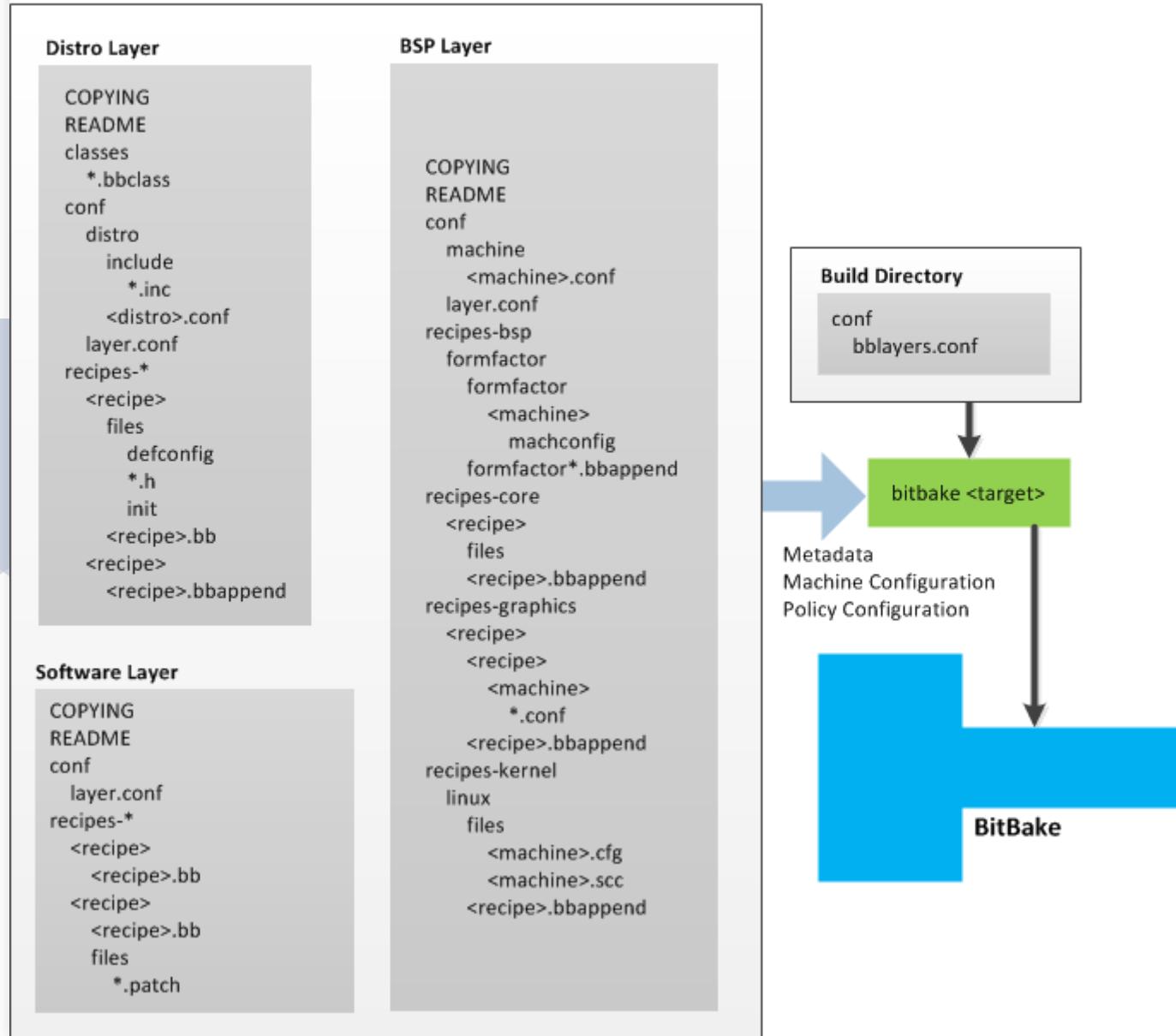
- YP is not a distro but **is something that allow you to build your own distro ...**
- *Combines, maintains and validates three key development elements: ...*

YOCTO (2/2)

1. A set of integrated tools to make working with embedded Linux successful, including tools for automated building and testing: [Bitbake](#), [Wic](#) ...
2. [Poky](#): A reference embedded distribution
3. The OpenEmbedded build system, co-maintained with the [OpenEmbedded Project](#)

The Yocto build environment is structured in layers. Let's see the layers like a set of recipes, classes and definitions that extend the base distribution.

Layers



Build Directory

conf
bblayers.conf

bitbake <target>

Metadata
Machine Configuration
Policy Configuration

BitBake

META-WEBKIT

- Created on Oct 2015 by Carlos López ([blog](#)).
- `meta-webkit` is an compatible Yocto BSP meta-layer which provides recipes for `WebKitGTK` and `WPE`:
 - The runtime and libraries for `wpe` and `webkitgtk`
 - The `libwpe`
 - The WPE backends implementations: `wpebackend-fdo` and `wpebackend-rdk`
 - and the reference WPE browser: `cog`

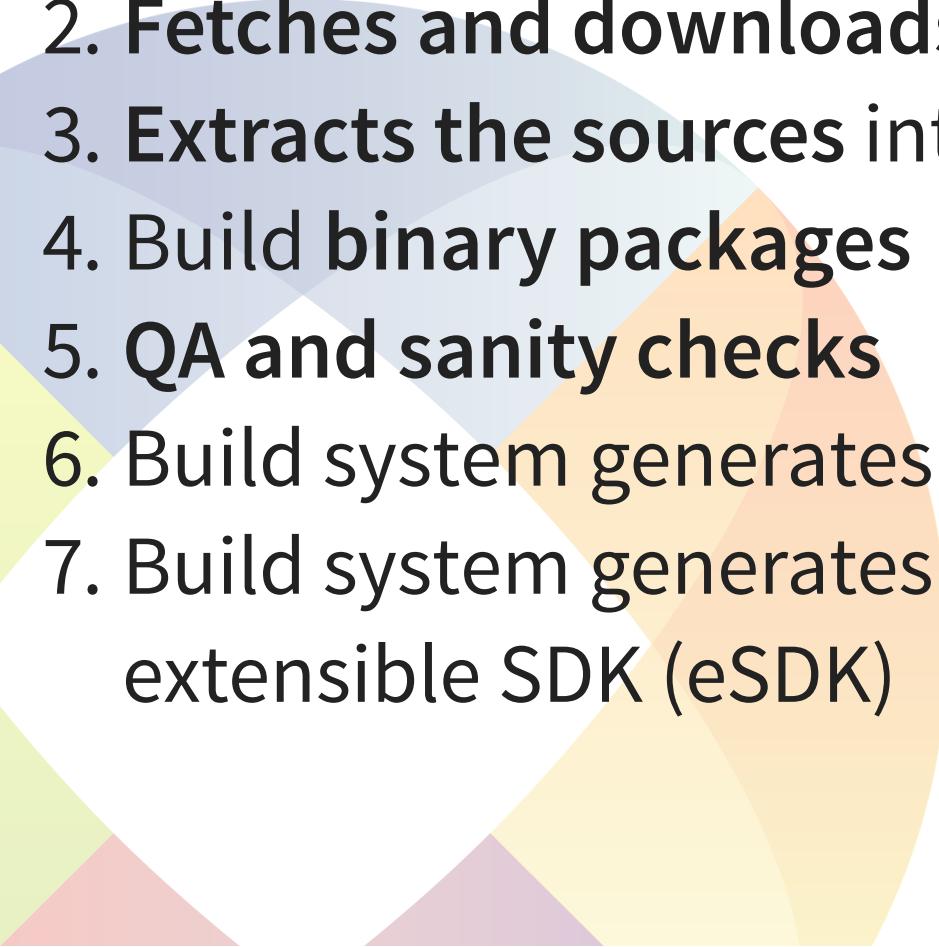
meta-webkit:

```
|- conf
  |- layer.conf
|- recipes-browser
  |- cog
    |- cog_0.8.0.bb
  |- libwpe
    |- libwpe_1.8.0.bb
  |- webkitgtk
    |- webkitgtk_2.32.0.bb
  |- wpebackend-fdo
    |- wpebackend-fdo_1.8.3.bb
  |- wpewebkit
    |- wpewebkit_2.32.0.bb
...
...
```



HANDS-DOWN

WPE ON A RPI (META-WEBKIT)

- 
- 1. Specify architecture, policies, patches ...**
 - 2. Fetches and downloads the source code**
 - 3. Extracts the sources into a local work area**
 - 4. Build binary packages**
 - 5. QA and sanity checks**
 - 6. Build system generates root file image**
 - 7. Build system generates the system image and the extensible SDK (eSDK)**

GETTING THE SOURCES AND ACTIVATE THE ENVIRONMENT

```
cd ${HOME}/yocto-rpi3-wpe
git clone https://git.yoctoproject.org/git/poky -b hardknott
git clone git://git.openembedded.org/meta-openembedded -b hardknott
git clone https://github.com/OSSystems/meta-gstreamer1.0 -b master
git clone https://git.yoctoproject.org/git/meta-raspberrypi -b
    master
git clone https://github.com/Igalia/meta-webkit -b master
```

```
$ source poky/oe-init-build-env
```

bblayers.conf:

```
$ cat conf/bblayers.conf
```

```
BSPDIR := "${@os.path.abspath(os.path.dirname(dgetVar('FILE',
    True)) + '/../..')}"  
  
BBLAYERS = " ${BSPDIR}/poky/meta \
${BSPDIR}/poky/meta-poky \
${BSPDIR}/poky/meta-yocto-bsp \
${BSPDIR}/meta-openembedded/meta-oe \
${BSPDIR}/meta-openembedded/meta-python \
${BSPDIR}/meta-gstreamer1.0 \
${BSPDIR}/meta-raspberrypi \
${BSPDIR}/meta-webkit \
"  
"
```

local.conf:

```
$ cat local.conf
MACHINE = 'raspberrypi3'
MACHINE_FEATURES_append = " vc4graphics"
GPU_MEM_256 = "128"
GPU_MEM_512 = "196"
GPU_MEM_1024 = "396"
EXTRA_IMAGE_FEATURES = "debug-tweaks"
IMAGE_FEATURES_append = " ssh-server-dropbear hwcodecs"
DISABLE_VC4GRAPHICS = "1"
PREFERRED_PROVIDER_virtual/wpebackend = "wpebackend-fdo"
PREFERRED_PROVIDER_virtual/libwpe = "libwpe"
IMAGE_INSTALL_append = " cog wpewebkit"
```

RUN BITBAKE

```
$ bitbake core-image-weston
Loading cache: 100% |#####| Time: 0:00:00
Loaded 3376 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:20:00
Build Configuration:
BB_VERSION          = "1.36.0"
BUILD_SYS           = "x86_64-linux"
NATIVELSBSTRING     = "universal"
TARGET_SYS          = "arm-linux-gnueabi"
MACHINE             = "raspberrypi3"
DISTRO              = "poky"
DISTRO_VERSION      = "1.0.0"
meta
meta-poky
meta-yocto-bsp      = "hardknott"
```

```
$ ls tmp/deploy/images/raspberrypi3/*wic
tmp/deploy/images/raspberrypi3/core-image-weston.wic
```



RUNNING COG IN RPI

```
root@raspberrypi3:~# export WAYLAND_DISPLAY=wayland-0  
root@raspberrypi3:~# export XDG_RUNTIME_DIR=/run/user/0  
root@raspberrypi3:~# cog -P fdo http://wpewebkit.org
```



THANKS

DISCUSSION / QUESTIONS