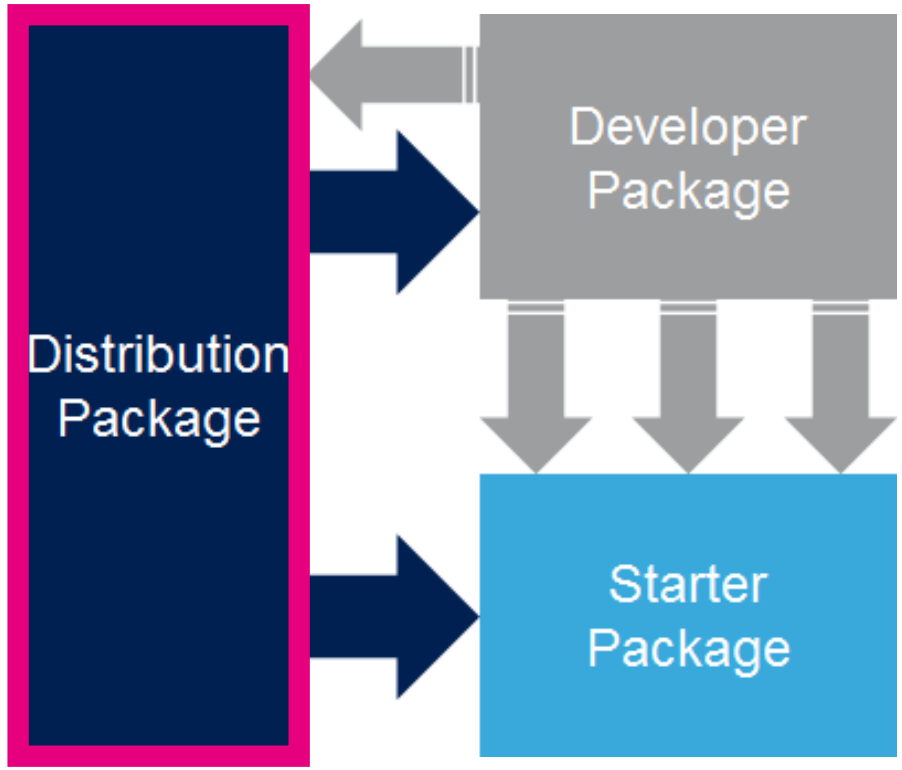
life.augmented

Openstlinux : distribution package workshop

Bossen WU

Distribution package

- The distribution package is able to generate the full starter package and also the SDK needed in the de developer package.
- The distribution package is based in Yocto/open embedded tools
- The main role of the distribution package is to manage the packages needed by the customer in the Linux envirnnement
- Able to generate:
  - All the file system (bootfs, vendorfs, rootfs, userfs)



Distribution package : installation

- Please follow the wiki page:
  - https://wiki.st.com/stm32mpu/wiki/
- Same distribution package for all the boards.
- Compile each binaries and update your board with the result.

Distribution package : user application

- Integrate your user land application in STM32 MP1 distro
  - Remind the application in "openstlinux-hands.c" you developed with the Developer Package. We are going to create a new generic layer for this application and add it in the standard build image (st-image-Weston).
  - Use devtool (see [OpenEmbedded - devtool]) to develop and validate your recipe (openstlinux-hands): **PC $>** devtool add openstlinux-hands <working_dir>/hands-on/openstlinux-distribution/openstlinux-hands-appli
    - Take back your hands-on delivery installed in your <working_dir> directory.

- ✓ Devtool has created a recipe based on what it has analyzed from your <working_dir> path. It has also added a workspace layer to be able to work with it.
- ✓ Check conf/bblayers.conf (see the new "workspace" layer) then "workspace" directory content.
- ✓ Check the proposed « openstlinux-hands.bb » recipe:
  - ✓ SRC_URI is empty: Check the bbappend recipe to understand how the src are found.
  - ✓ Oe_runmake was proposed to the compile and install tasks (as a Makefile was present in the source directory). Yocto will call it to build and install the component.

Distribution package : user application

- Try to build it, load and test:

```
PC $> devtool build openstlinux-hands
PC $> devtool deploy-target openstlinux-hands root@<board ip@>
```

```
Board $> openstlinux-hands
```

- The recipe is now validated with devtool, we will have to integrate it in the final layer

- Create your new layer: See [How to create a new open embedded layer].
  - Create a new layer named "meta-open-hands" in meta-st directory and add it to your build system ("-p 6" for priority and "–e openstlinux-hands" for recipe name).

✓ Check content (from build dir):

```
PC $> tree ../layers/meta-st/meta-open-hands/
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-openstlinux-hands
    └── openstlinux-hands
        └── openstlinux-hands.bb
```

life.augmented

Distribution package : user application

- ✓ Check "conf/bblayers.conf":
  A new layer was added to the BBLAYERS variable (if you did "bitbake-layers add-layer" as recommended).
- ✓ Check your layer "conf/layer.conf": There might be a bug in the "create-layer" command. You need to replace the "\\" with single one "\" (if bug not solved). It will be required for the next steps.

- The layer you have created is a template (so almost empty). Now we have to finalize this layer so that it can be used like others.
- Copy the recipe designed with devtool (in "workspace" directory) to this new layer, edit this recipe.
- For the SRC_URI, You need to tell Yocto what to process and where to find them:
  - There are numerous solutions for that and "Openembedded-core" package is full of examples especially the "meta-skeleton" layer.

Distribution package : user application

- Most of the SRC_URI rely on a git repository but for this hands on, we will store the files directly inside the layer. The method below is commonly used:

```
SRC_URI = "file://openstlinux-hands.c \
           file://Makefile"
S = "${WORKDIR}"
```

- "$S" variable is the source directory for Yocto.
- Then the sources will be stored like below (directory name is recipe name):

```
└── openstlinux-hands
    ├── openstlinux-hands
    │   ├── Makefile
    │   └── openstlinux-hands.c
    └── openstlinux-hands.bb
```

Distribution package : user application

- Copy your Developer Package files like mentioned above (new "openstlinux-hands" and copy files) in your "meta-open-hands-layer" layer

- Remove your devtool workspace:

```
PC $> devtool reset openstlinux-hands
```

Distribution package : user application

- Test your new layer
  - Build your recipe (for a quick check) then your full image:

```
PC $> bitbake openstlinux-hands
PC $> devtool build-image -p openstlinux-hands st-image-weston
```

  - Flash the full image and test on the board:

```
Board $> openstlinux-hands
```

  - OR, You may use « devtool deploy-target » as a quicker way like previously

Distribution package : user application

- Finalize your layer
  - Update your recipe with the license information (LICENSE and LIC_FILES_CHKSUM): GPLv2 in our case. The recipe license is MIT, do not mix it with the source license. The checksum is the one from the license text and it is always checked by bitbake before building.
  - Add your recipe to the selected image:
    - Create a new bbappen file: st-image-weston.bbappend.
    - Add IMAGE_INSTALL_append = " openstlinux-hands" in this file.
  - Add your layer to ST distribution:
    - Check: meta-st/meta-st-openstlinux/conf/template/bblayers.conf.sample.
    - Add your layer to ADDONSLAYERS variable like done for others.

Distribution package : user application

Lab 1 TIPS:

- Your Developer Package application makefile contains this line:

"DESTDIR ?= ./kernel_install_dir/usr/local/bin"

This is very common and match how devtool/yocto is working (defining DESTDIR and bindir).

- You can get reference of "meta-skeleton" layer to get licensing recipe information up to date in your recipe
- After you have successfully integrated your application or framework, you can regenerate your complete Developer Package:
  - Generate SDK see [How to create an SDK for OpenSTLinux distribution].
  - Generate Starter and Developer Packages see [Generating your own Starter and Developer Packages].
  - The use case is to go back in a Developer Package cycle with your own distribution or a customized ST distribution.

Distribution package : device driver

- Lab 2: Integrate your device driver in STM32 MP1 machine
  - As for lab 1, use devtool to make a recipe template but using kernel driver code
  - From yocto build dir:

PC $> devtool add openstlinux-hands-machine <working_dir>/hands-on/openstlinux-distribution/openstlinux-hands-driver/

- ✓ Check the recipe, you will see:
  - ✓ KERNEL_DIR variable automatic setting
  - ✓ inherit module (to inherit module.bbclass)

- Build and deploy it like on lab 1.
- It is not working, Why?

life.augmented

Distribution package: device driver

- Previous trial was not working because we did not handle the device tree modifications: we will use for that "devtool modify" tool.
- In your build directory:

```
PC $> devtool modify linux-stm32mp
PC $> tree -L 3 workspace/
```

  - You will see a new sources directory with the kernel sources and the linux-stm32mp directory contains a .git directory: you are on a git extraction of the ST kernel code !

- Patch the device tree with the patch from the Developer Package hands on, and commit your modification (you may “git am” to do it in one step).

✓ You can see (for example with gitk tool): a new commit has been added on "devtool" branch. You can also check with "git status" command, there is nothing left to commit.

life.augmented

Distribution package : device driver

- Build and test the new device tree modifications on your board (“devtool build” then “deploy-target”)

- Do not forget to build the new dtb files (not obvious in yocto)
- Now, you have validated your modifications with devtool, you need to integrate them in your layer:
  - Integrate "openstlinux-hands-machine" recipe in your layer like done in lab 1 (add directory tree and files)
  - Add your "openstlinux-hands-machine" recipe to "st-image-Weston" like lab 1
  - Integrate your modified recipe in your layer:

PC $> devtool finish linux-stm32mp meta-open-hands

✓ Check what the previous command did for you. This command is magic !

- Test build then deploy on target and test

life.augmented

Distribution package : device driver

Lab 2 TIPS

- List what devtool will deploy on target (when you are working with "workspace"):

```
PC $> devtool deploy-target -n linux-stm32mp root@<board ip@>
```

- “Deploy-target” of the kernel leads to error for the moment: you have to remove wmlinux (automatically generated by the kernel recipe) from the image directory before the deploy.
- List all the tasks supported by a recipe:

```
PC $> bitbake -c listtasks linux-stm32mp
```

- To force the rebuild of the kernel device tree, you can "touch" the kernel recipe and rebuild it with devtool (when you are working with "workspace").
- "%" wildcard for recipe version means it will apply to all the versions.
- You can cleanup your build space with bitbake command:

```
PC $> bitbake openstlinux-hands-machine -c cleanall
```

Distribution package : device driver

- You may redefine "STAGING_KERNEL_BUILDDIR" variable (to build outside source code directory).
- OR you can use "--no-same-dir" option of devtool.

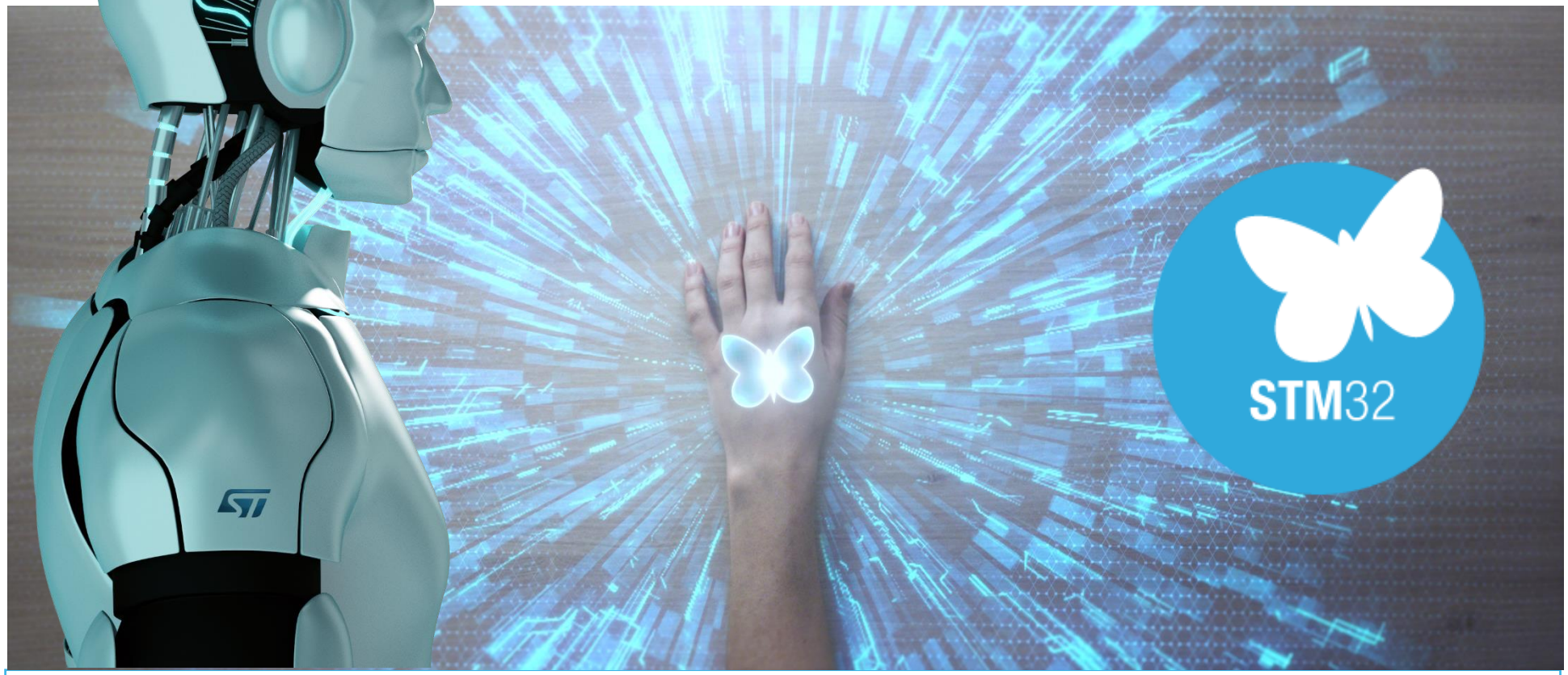
Distribution package : create image

- Please follow the wiki page:
  - https://wiki.st.com/stm32mpu/wiki/How_to_create_your_own_image
- Following this wiki page and the information inside the wiki create a new image “ST_customer”
- This ST_customer image must be include a new core image ST_cutomer core image that is the same that the ST core image without the python package.
- Generate and use the new created image

Distribution package : create SDK

- Please follow the wiki page:
  https://wiki.st.com/stm32mpu/wiki/How_to_create_an_SDK_for_OpenSTLinux_distribution
- Create a SDK link to the new image that you just create : ‘ST_customer”.
- Install the SDK generated
- Build one application using this SDK (Kernel, TF-A or user application)

Releasing your creativity with the STM32



/STM32 @ST_World community.st.com

Famous video here

www.st.com/stm32