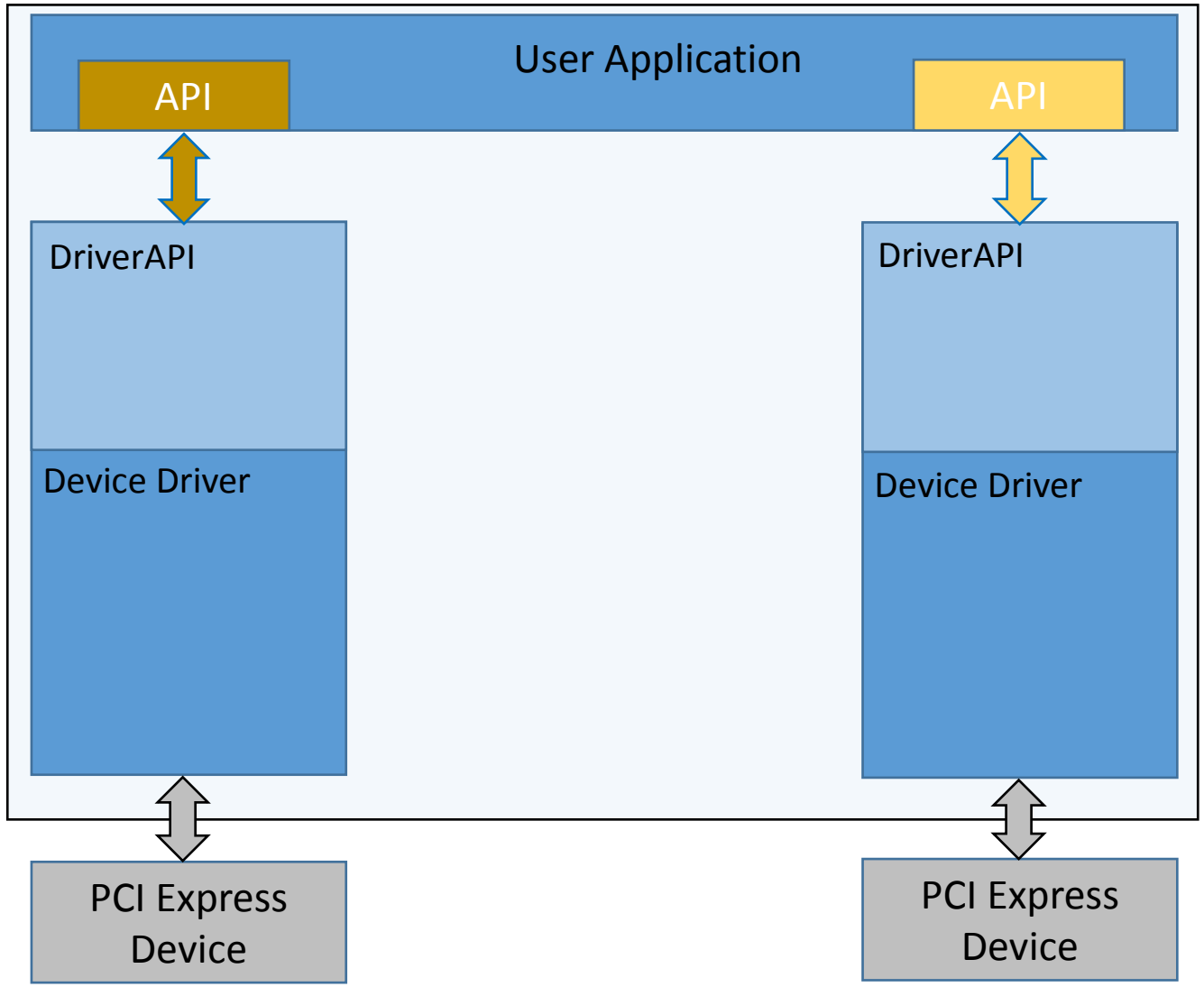
The universal PCI Express Device Driver for MTCA.4

L.Petrosyan

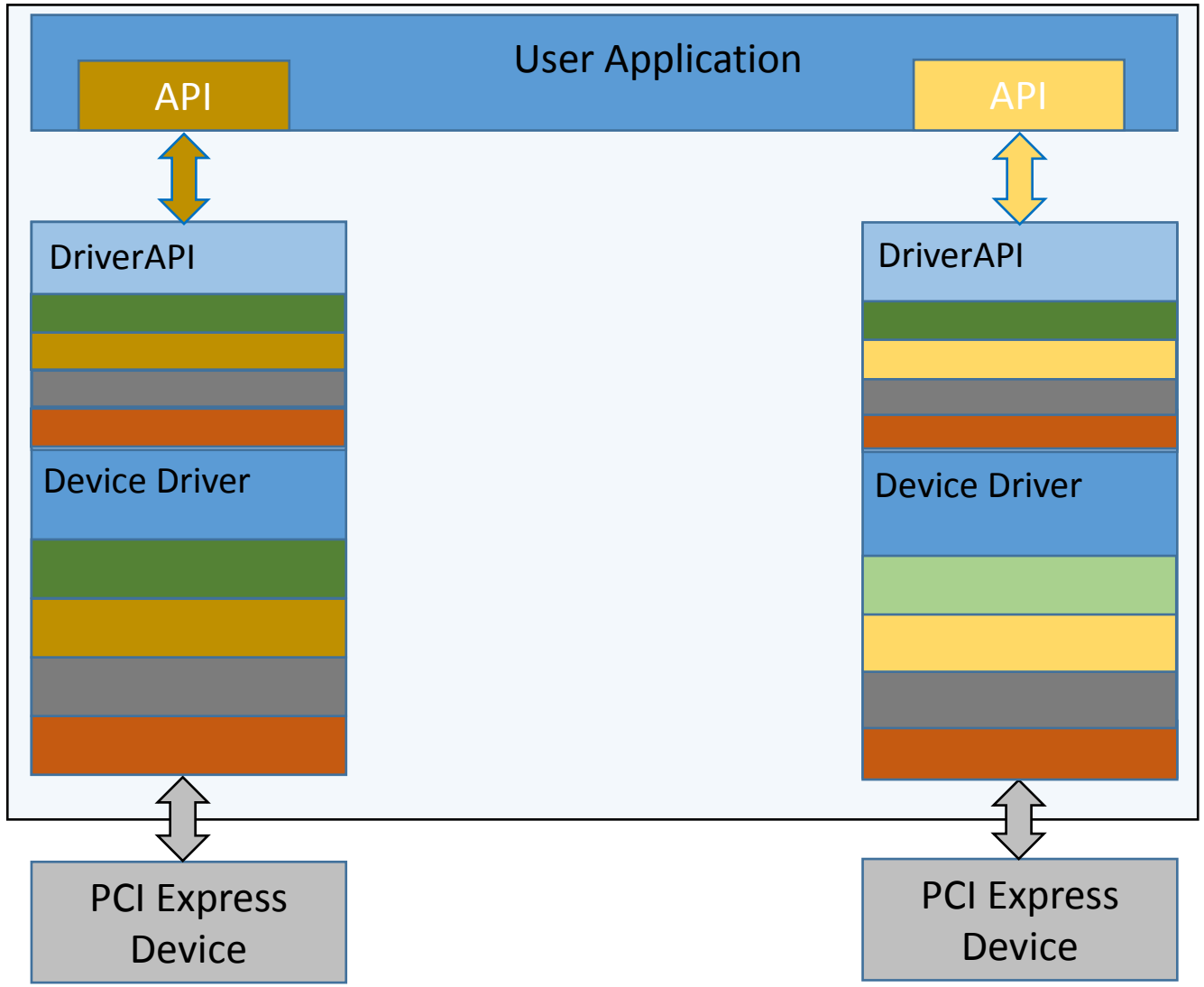
The Universal PCI Express Device Driver

- A Device Driver is loadable kernel Module that provides access to the particular device attached to a computer (PCI Express Bus)
- User Application use the Device Driver API to access to the Device
- More Devices -> More Drivers
- Different Drivers -> different APIs



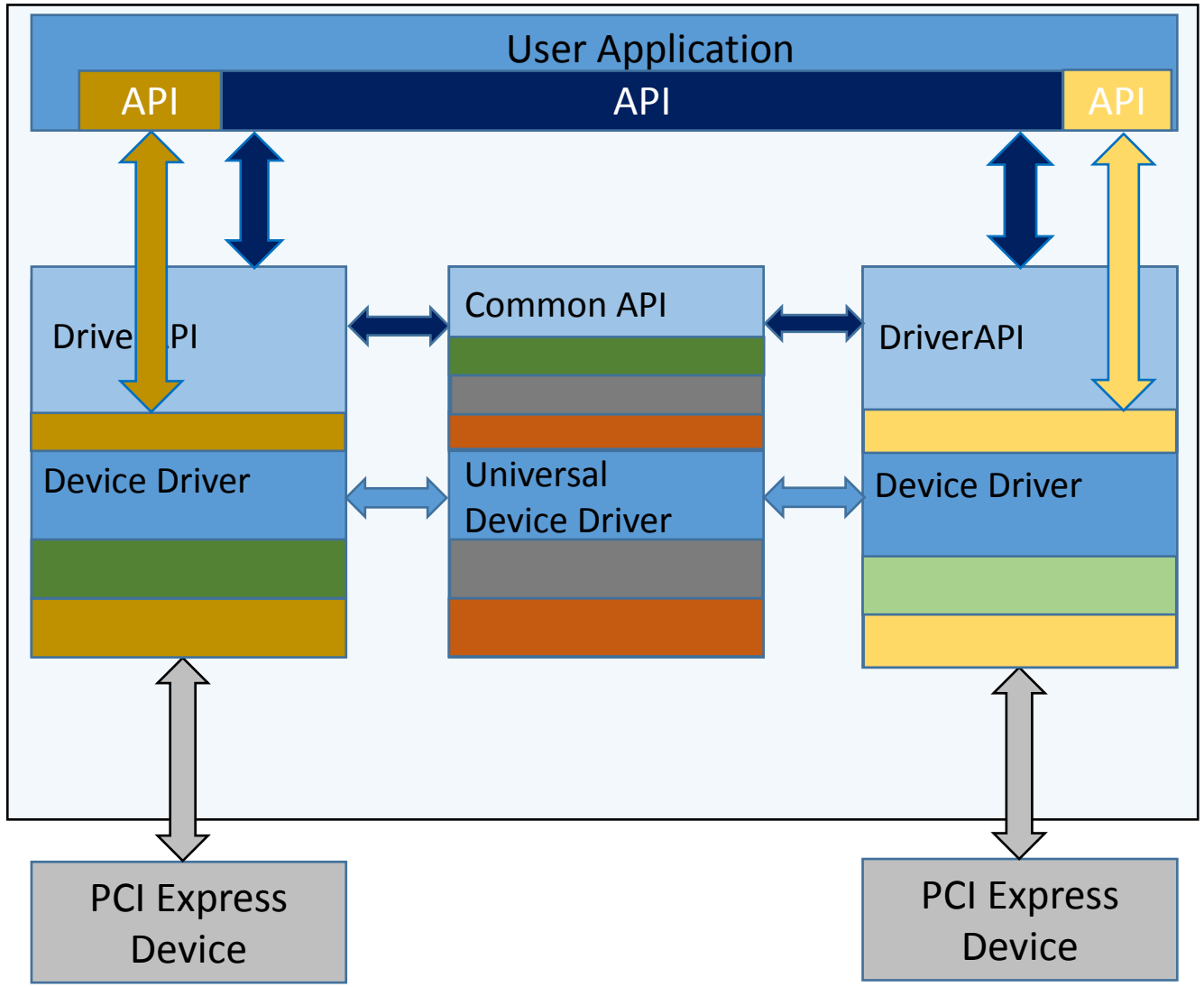
The Universal PCI Express Device Driver

- Basic PCI Express functionality
  - Mapping memories
  - **Read**, **write** and some common **ioctl**
  - Error handling
  - Hot Plug
- Standards or Guidlines functionalty
  - Standard Registers Set
  - SHAPI Registers Set (PICMG)
- Device specific functionality
- Device specific but has common API
  - DMA



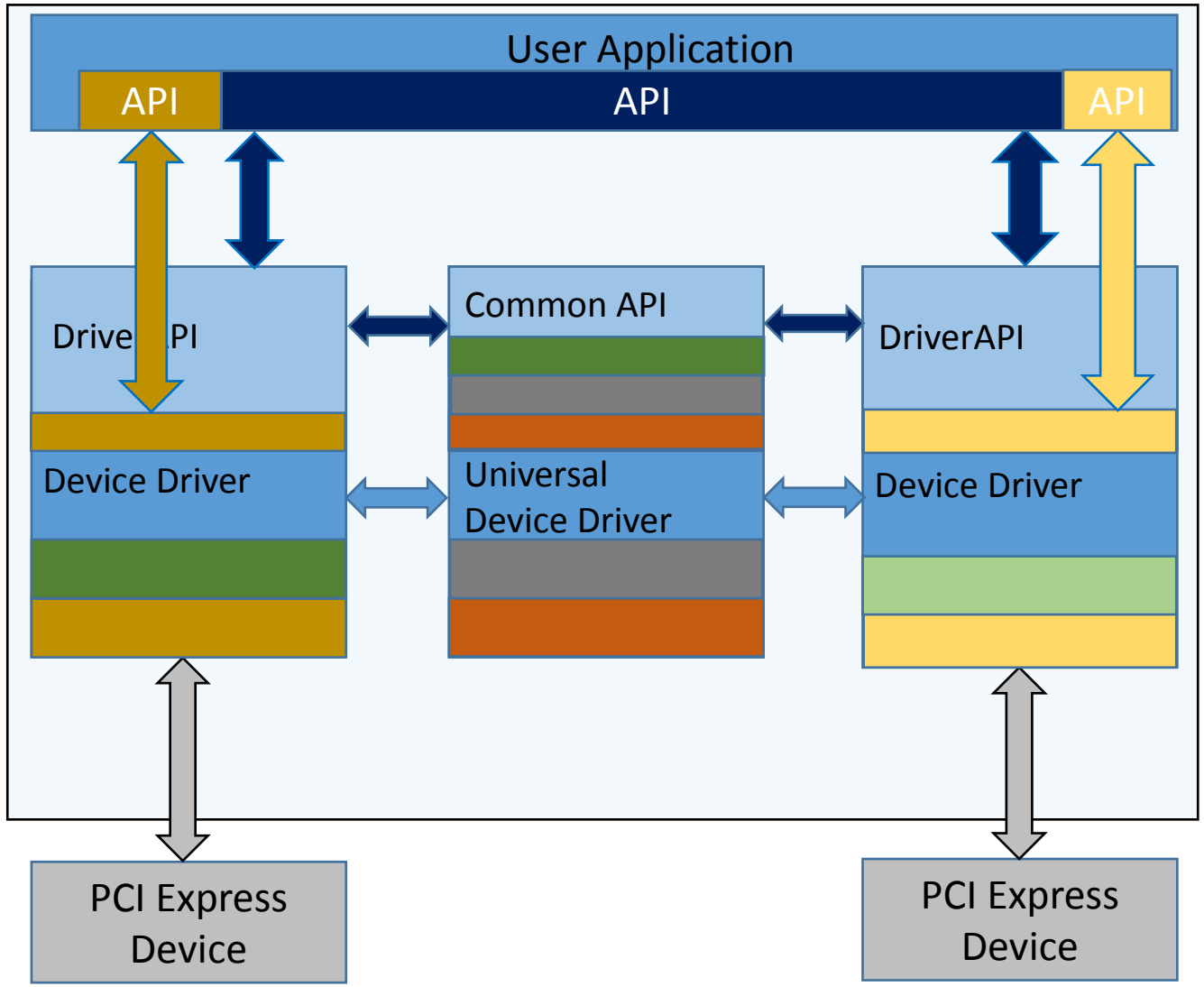
The Universal PCI Express Device Driver

- Split Device Driver into two parts follow the Linux Device Driver stacking Model
- Add all common functionality and API into universal part
  - Basic PCI Express functionality
  - Standards or Guidlines functionalty
- Add Common API for Device specific functionality into universal part but keep functionality in Device Driver side
  - Device specific but has common API



The Universal PCI Express Device Driver

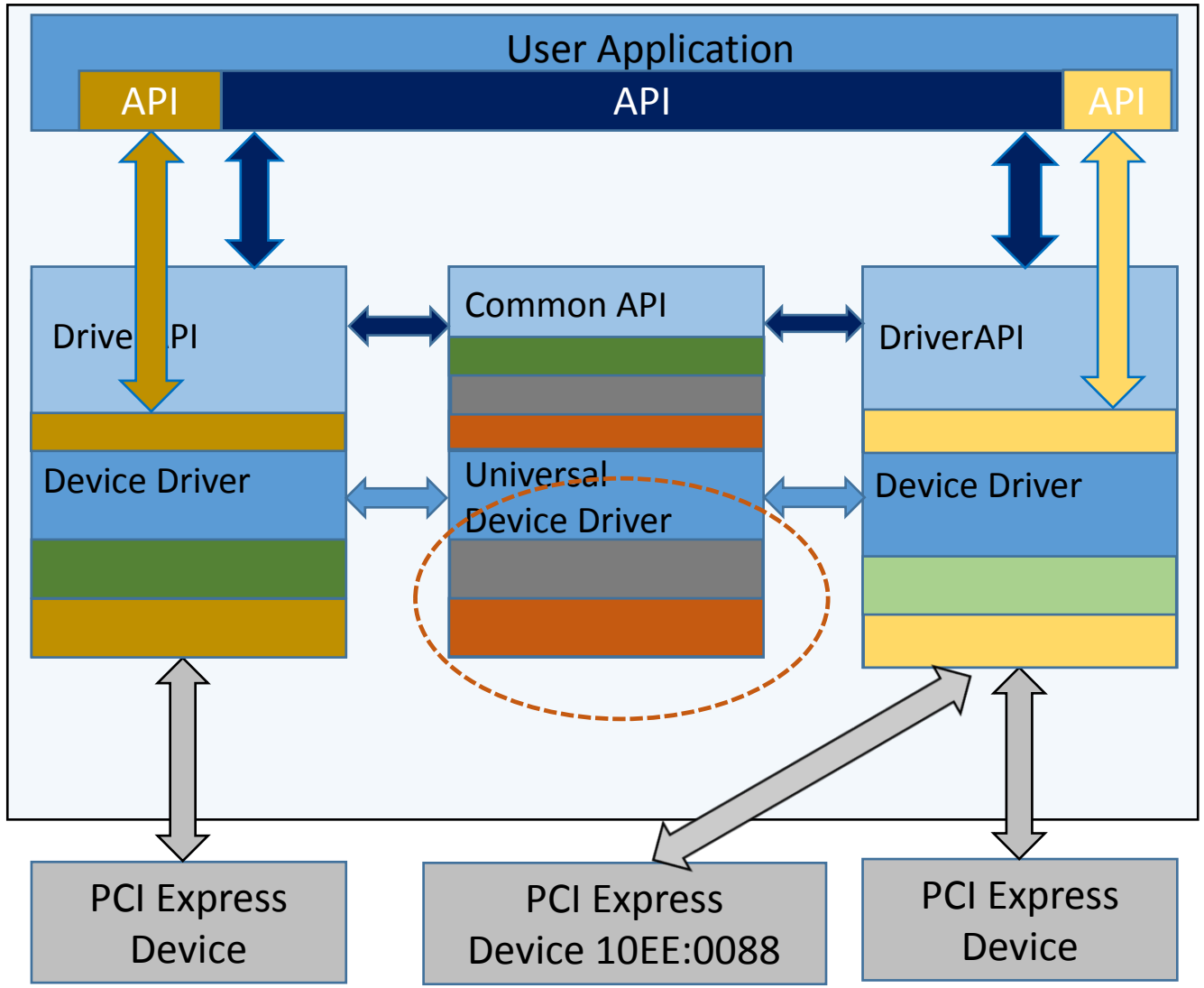
- this approach facilitates creation of new drivers, user applications and integration of new devices into the existing software



The Universal PCI Express Device Driver

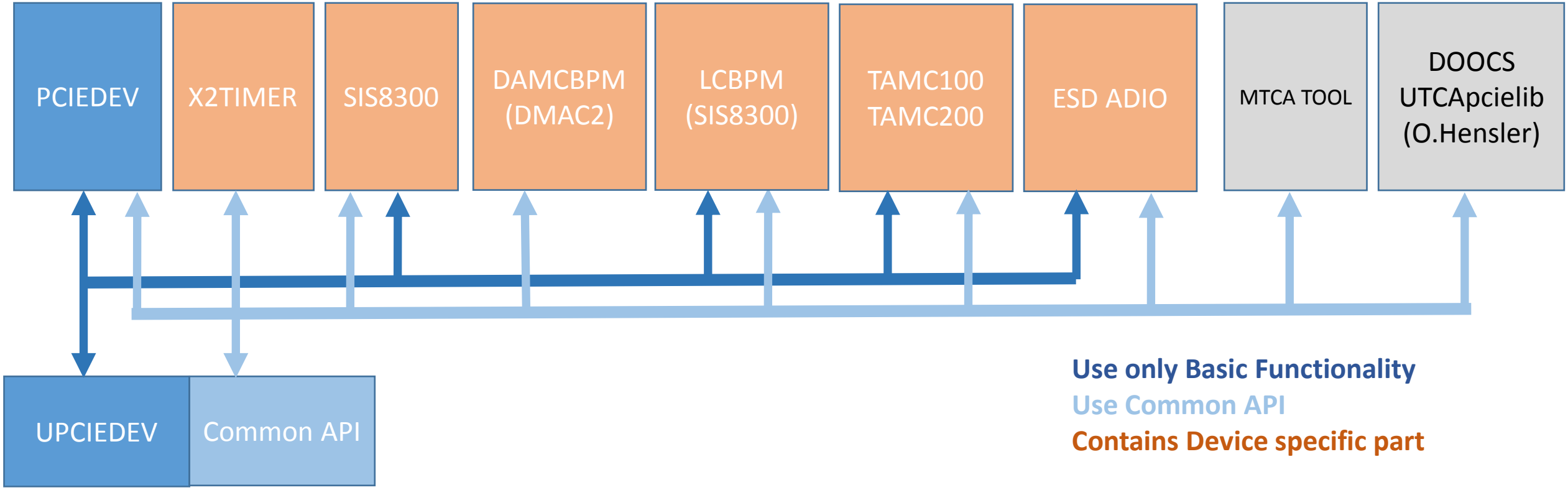
- this approach facilitates creation of new drivers, user applications and integration of new devices into the existing software
- The Device Driver created on the top of ***universal*** driver has all necessary PCI Express functionalty
- It could be binded to any PCI Express Device

echo „10EE:0088“ > /sys/bus/devices/xxx/driver/new_id



Status

- this architecture was developed in DESY group MCS4 and till today with success is used
- the following drivers, tools and libraries are developed used



Status

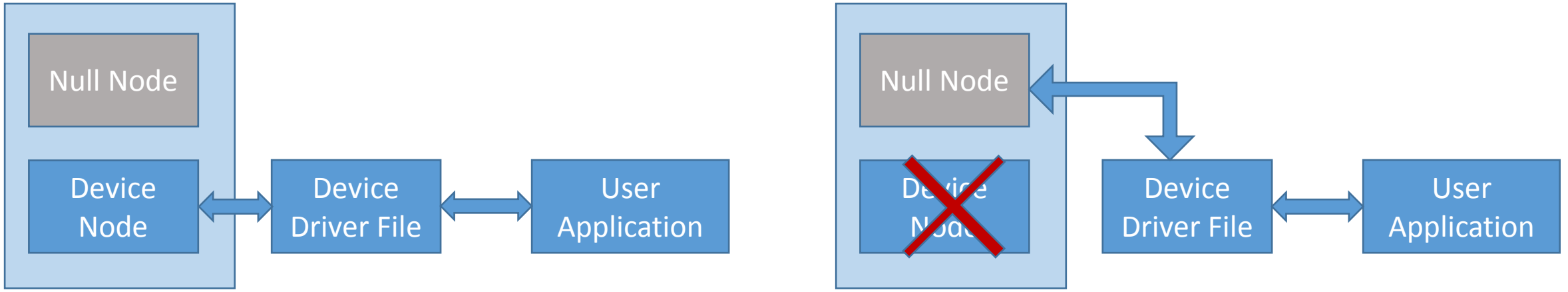
Added new functionality

- Block Read/Write
- Read error handling based on the readed value (0xFFFFFFFF)
  - If readed value is 0xFFFFFFFF read from the known register to check the Device
- Device Driver File name based on the Standard Registers Set
  - Driver creates the Device File with the name coded in the Standard Registers Set
- Added DMA for the DESY/XFEL StartUp projects for DESY DAMC2 and STRUCK SIS8300 boards

Status

Improvements of the Hot Plug

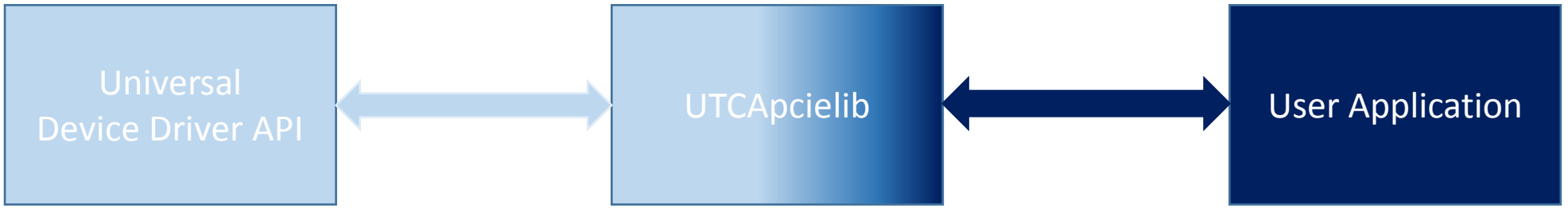
- Device Driver and User Application handshaking during Device Hot Swapping
- Save Removing the Device File and the Device which is in use by the user application
- All Opened by the User Applications Device Driver Files redirected to the ***„Null Node“,*** which returns ERROR on all Device Accesses



Status

User Application
UTCApcielib (developed by O.Hensler)

- The basic API as C++ classes which allow to use all Device Driver functionality without knowledge about Device Driver API details
  - Which allow to change Driver part keeping User Application unchanged
- Last yers all DOOCS servers switched from the direct Driver API to Libraries API



Collaboration with **COSYLAB**

DMA performance tests,
code cleaning,
Bug reports and future improvements

- Different DMA transfer algorithms were tested
  1. Simple DMA read
     - Each DMA read is done into freshly allocated kernel buffer. Data is then copied to user-space and kernel buffer is freed
  2. DMA read using ring buffer
     - This algorithm uses a ring of kernel buffers and is capable of executing DMA to one buffer in parallel to copy-to-user-space from another buffer

Collaboration with **COSYLAB**

DMA performance testing results

(***The detailed result report one can get from T.Susnik COSYLAB or M.Killenberg DESY***)

- The best compromise between complexity of a driver code and performance is split DMA reading using ring buffer with parallel copy-to-user-space .
- One can gain performance up to 20% depends of DMA size.

Open questions

- increase in number of interrupts
- Tests are done with one board
- This algorithm will be tested with more boards and higher rate (make DMA with 10-25 Hz repetition rate)

Plans

Add more functionality to the universal driver

- More PCI Express specific tasks (PCI Express Error Handling, Transaction Ordering ...)
- Hide more Linux Kernel calls in the universial driver (top level driver does not depend on Kernel Version)
- Code cleaning, better documantation
- ...

Info

- All sources of the Drivers and Libraries are Open Source
- The source codes can be found on a DOOCS web page ***http://doocs.desy.de***
  - Go to *Source Code Repository*
    - ***source/unixdriver/utca/linux/upciedev*** and ***pciedev*** (all others as an example)
    - ***source/UTCApcielib***

- The packages can be download from ***http://doocs.desy.de/pub/doocs/***
- Mail ***doocs@desy.de***

THANK YOU