Generic Software Platform

Mathias Punger

January 19th, 2005

Agenda

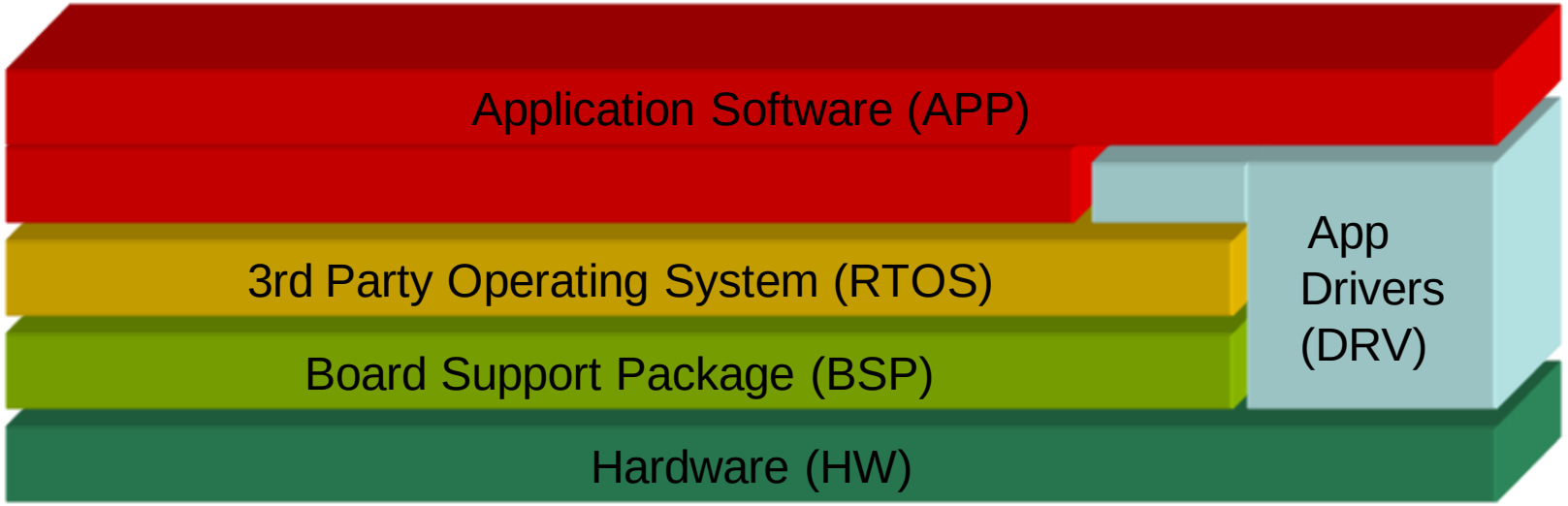
Software Architecture

Components of the Generic Software Platform

Test Features of the Generic Software Platform

Current Status

Embedded Systems Software Architecture



APP: Software running on device (protocol stack, multimedia, PIM)

DRV: High-level application drivers (use RTOS services)

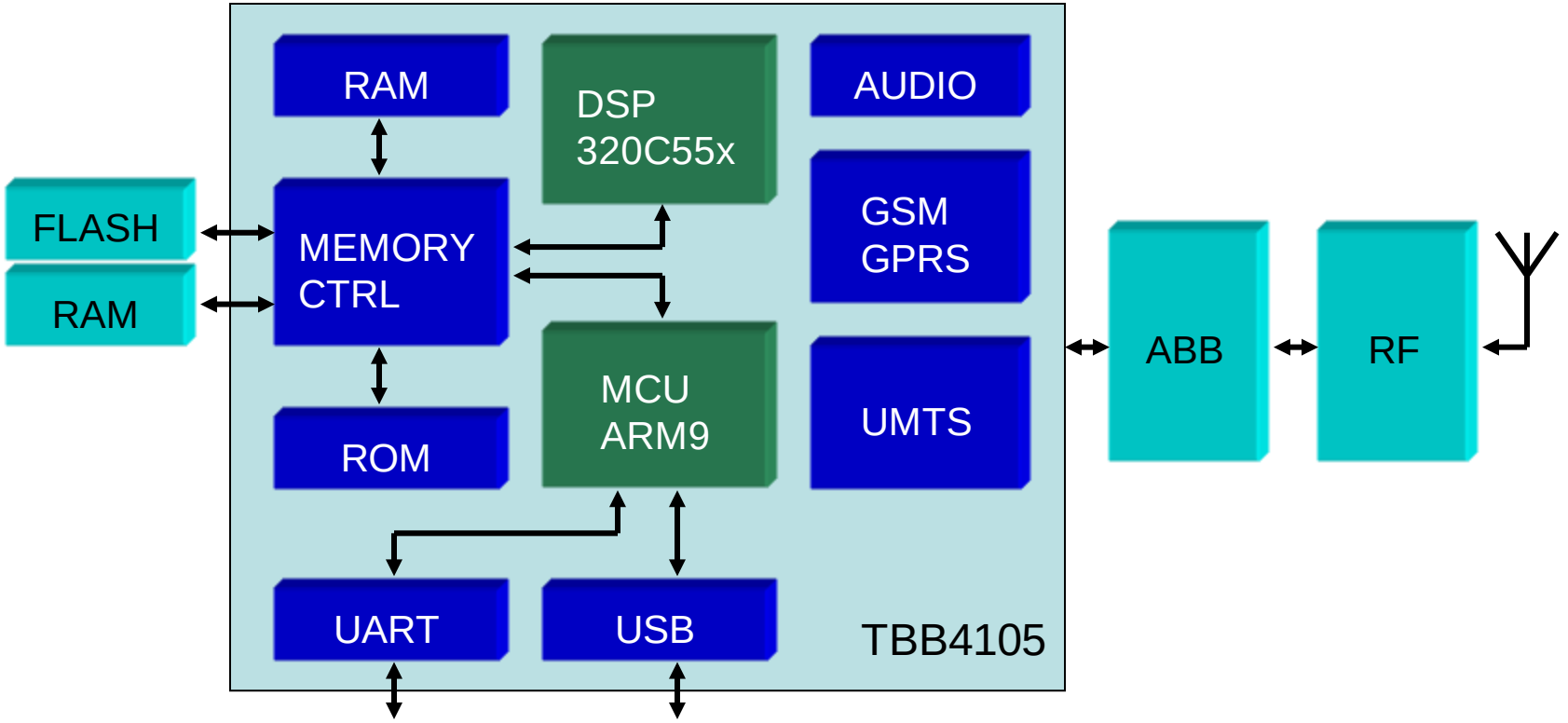
RTOS: Abstracts from HW and manages resource sharing

BSP: Basic HW support to boot system, low-level drivers

HW: Mobile phone chipset including MCU, DSP and Peripherals

Hardware Example

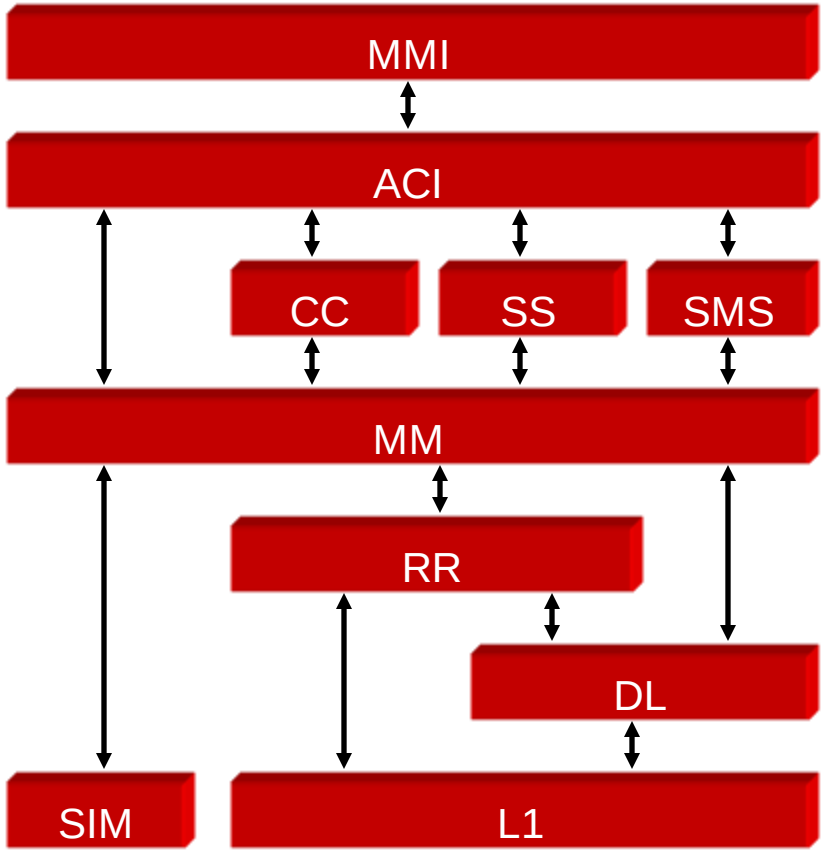
Digital Baseband Chip for GSM/GPRS/UMTS



The software in a mobile phone is executed on the MCU and DSP of the Digital Baseband chip.

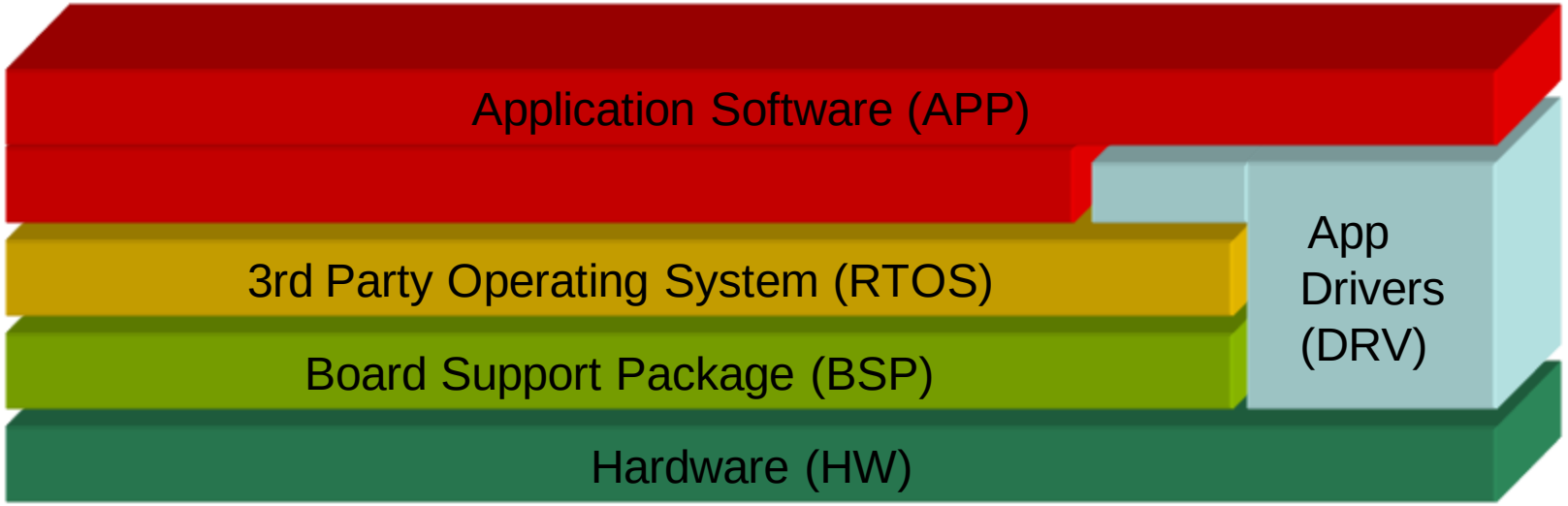
Application Software Example

GSM Protocol Stack for Mobile Phones



- Software Entities running in different RTOS threads implement the GSM protocols
- Message based Communication between software entities
- Nearly all software entities use RTOS services like timers, memory,...
- Extensive tests of software entities mandatory

Embedded Systems Software Architecture



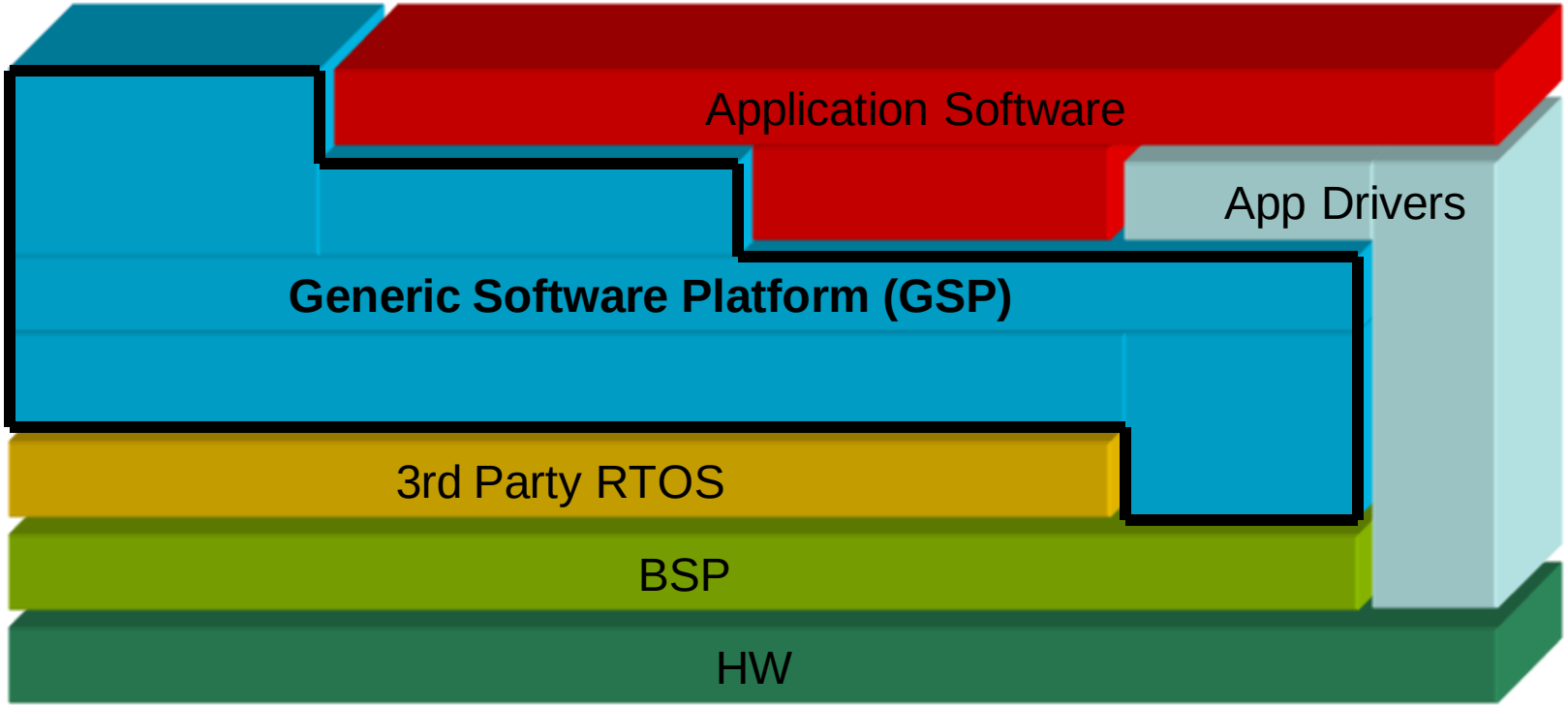
Challenge: Different operating systems need to be supported (Nucleus, Linux, Symbian, ...)

Drawback: The application software is dependent on the RTOS.

Objective: Keep application software operating system independent.

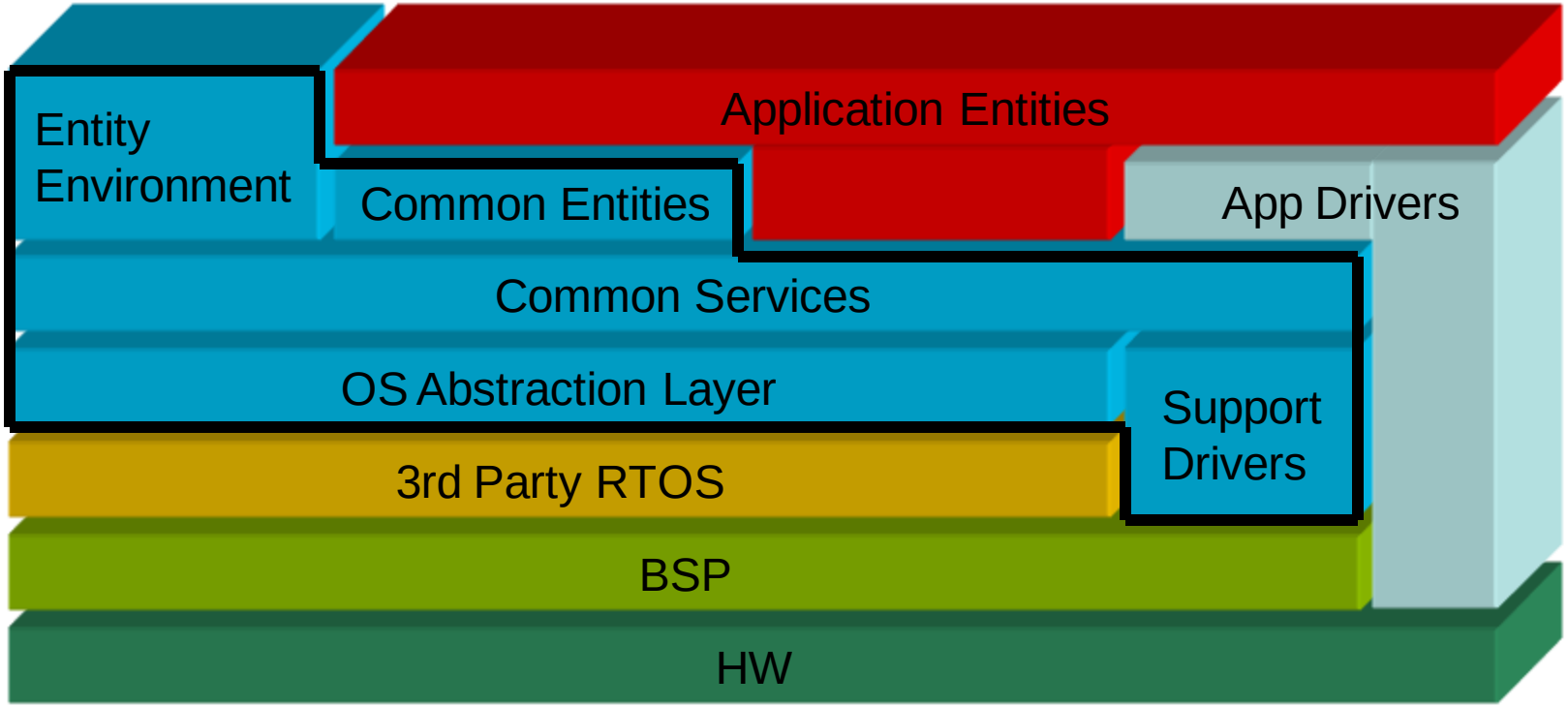
Conclusion: Introduction of the Generic Software Platform (GSP)

Embedded Software Architecture with GSP



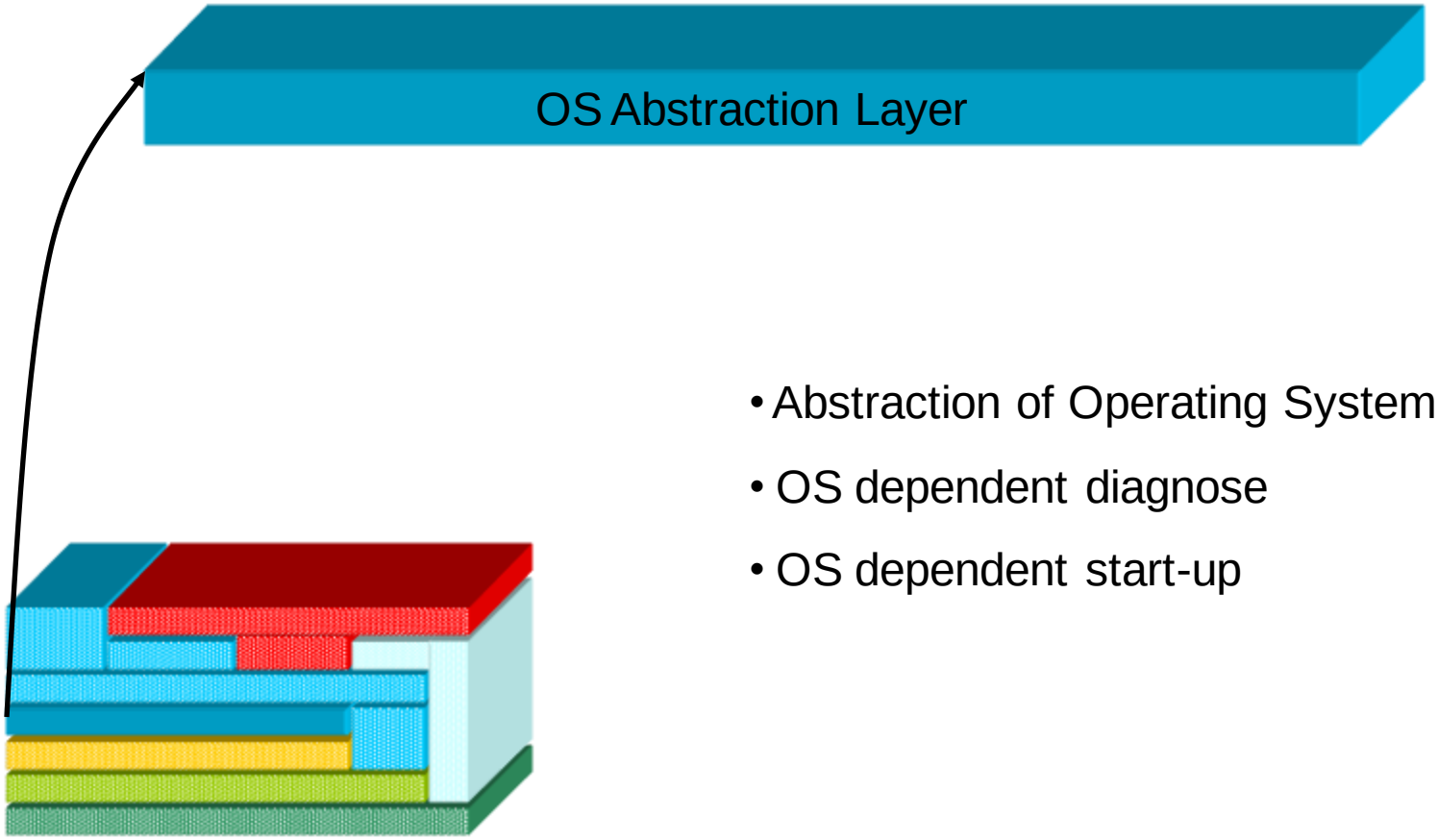
Application Software accesses the Operating system through the Generic Software Platform (GSP) and remains OS independent.

Software Architecture with GSP



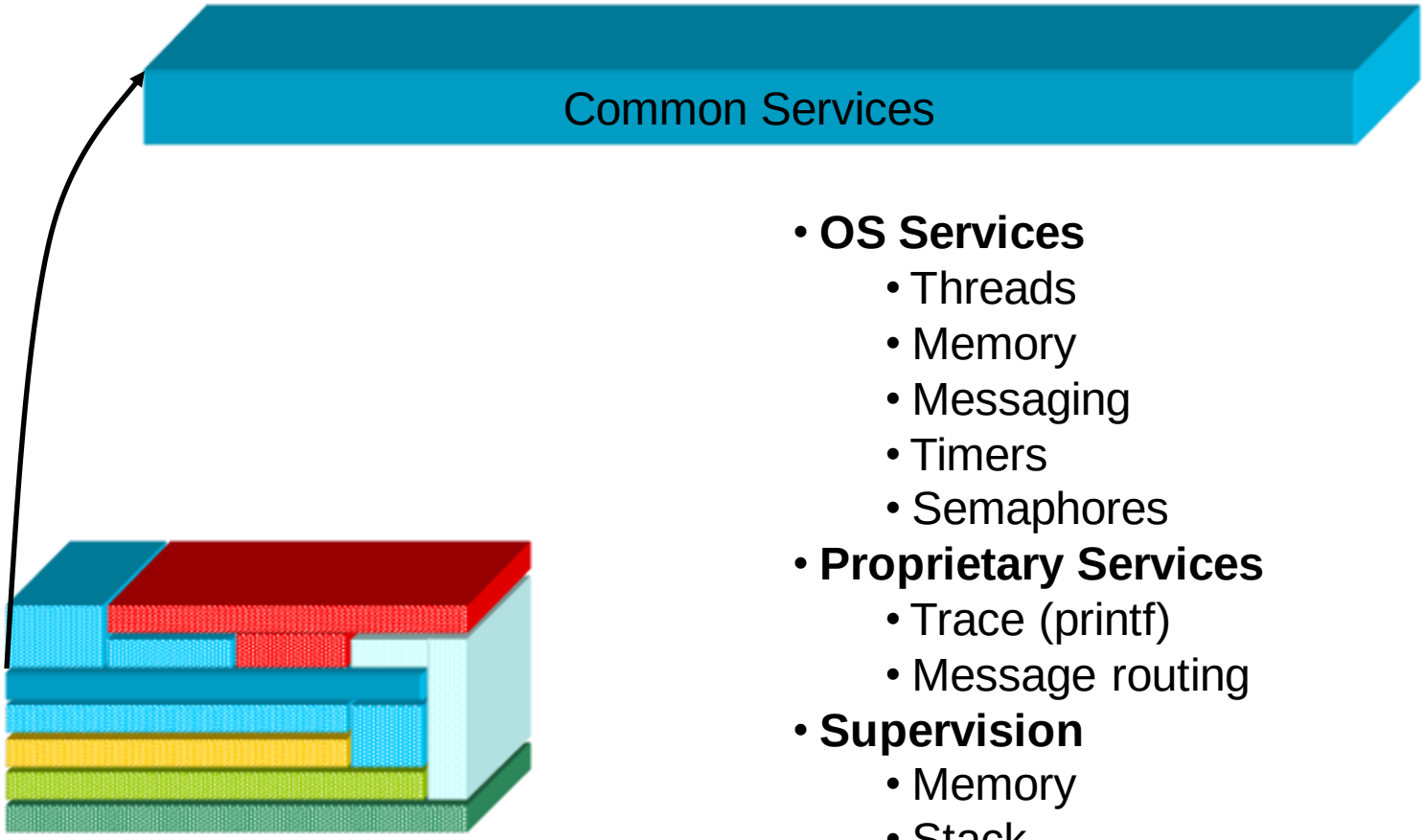
In addition to the RTOS abstraction GSP provides several services needed to run and test the application entities.

GSP Components



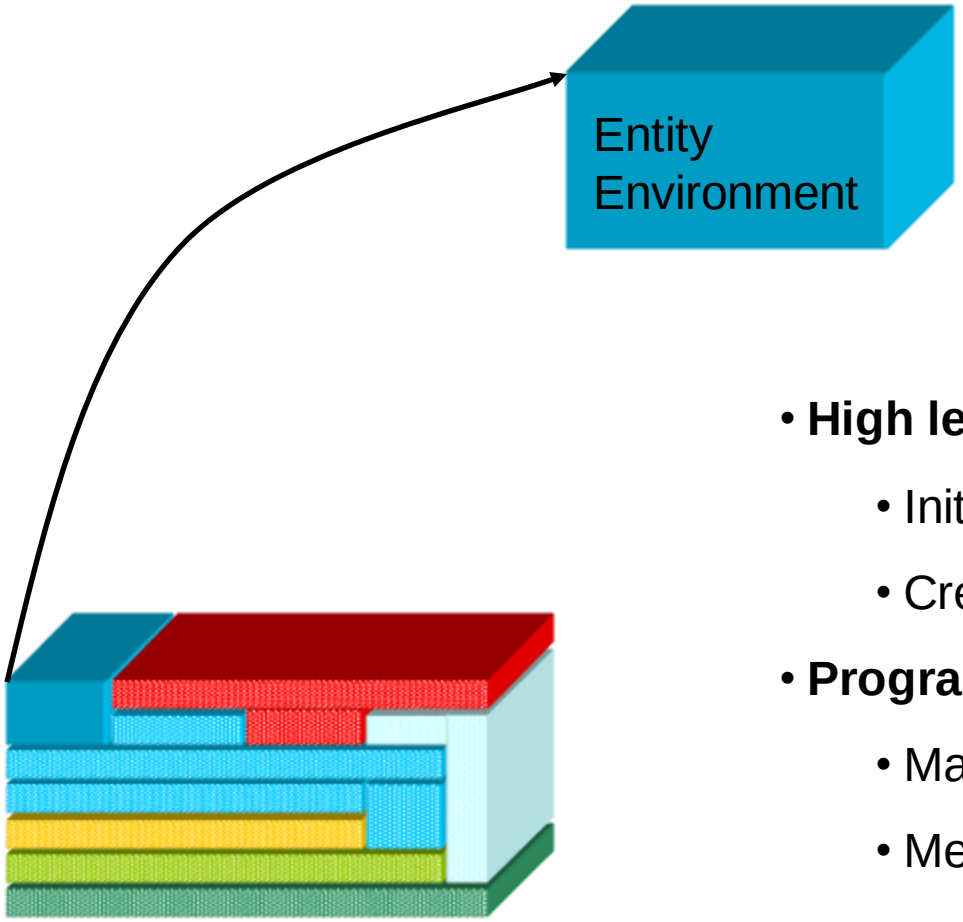
- Abstraction of Operating System API
- OS dependent diagnose
- OS dependent start-up

GSP Components



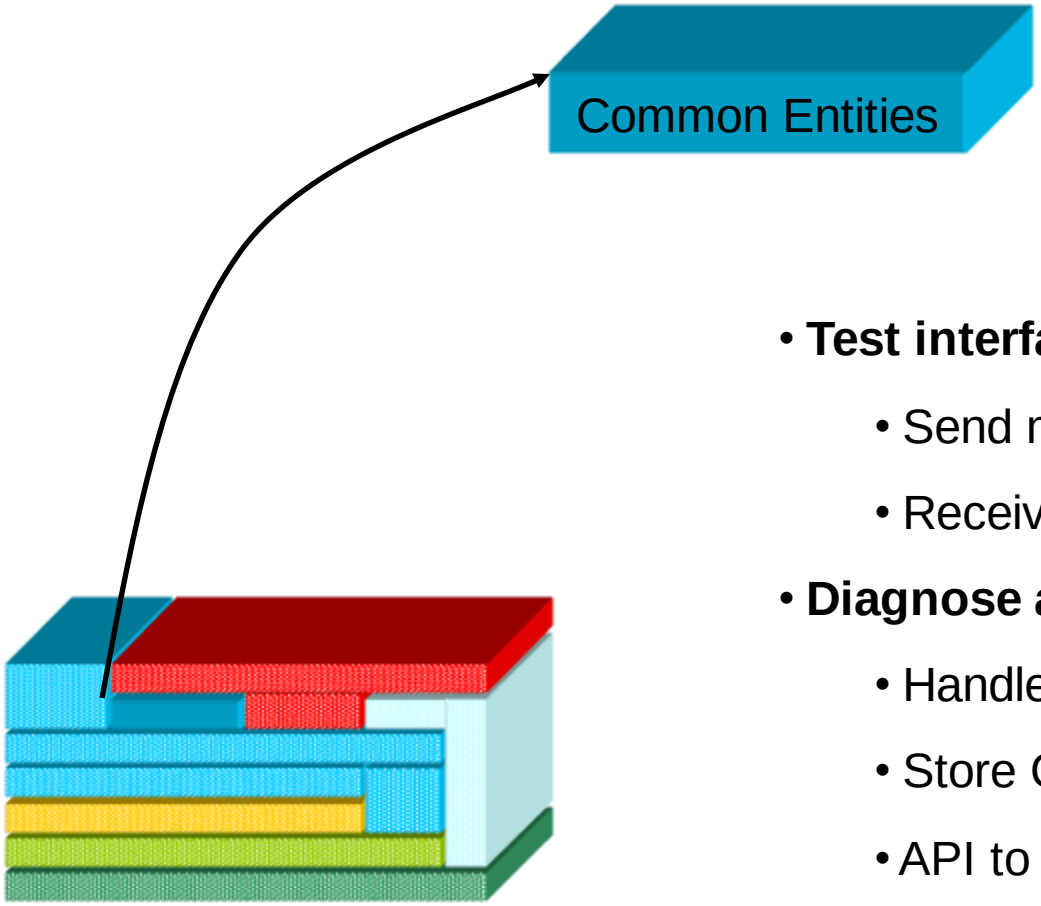
- **OS Services**
  - Threads
  - Memory
  - Messaging
  - Timers
  - Semaphores
- **Proprietary Services**
  - Trace (printf)
  - Message routing
- **Supervision**
  - Memory
  - Stack

GSP Components



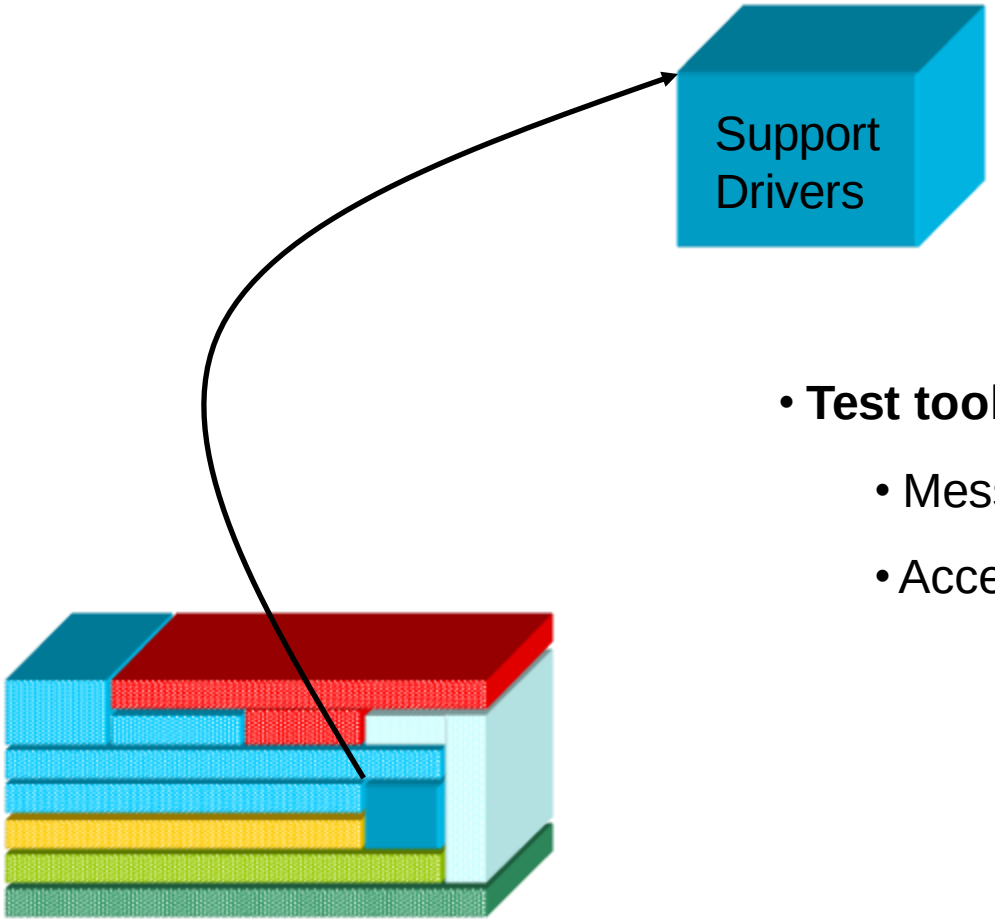
- **High level startup**
  - Initialization
  - Creation/Start of threads
- **Programming environment**
  - Main loop of threads
  - Message dispatcher

GSP Components



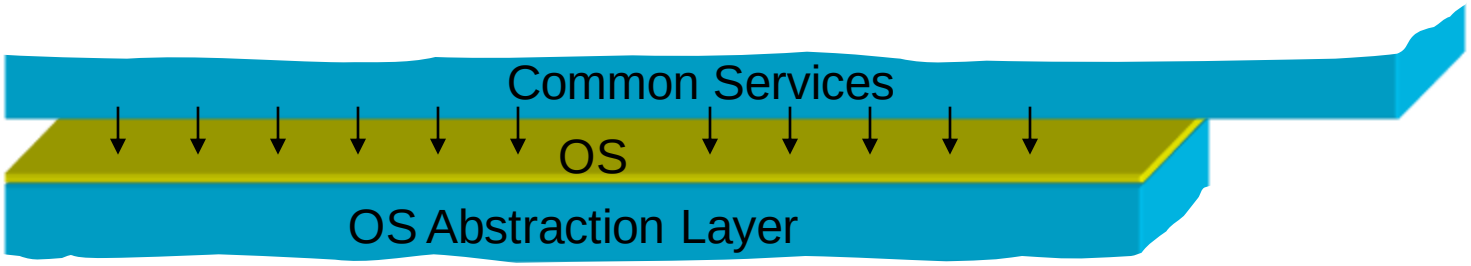
- **Test interface**
  - Send messages to tools
  - Receive messages from tools
- **Diagnose and crash handling**
  - Handle CPU exceptions
  - Store CPU state at crash time
  - API to handle errors/warnings

GSP Components



- **Test tool connectivity**
  - Message protocol
  - Access to UART, USB, Ethernet

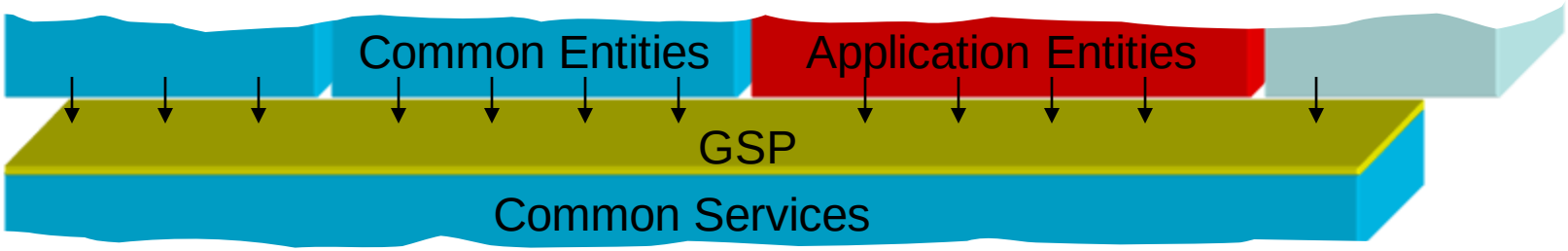
Generic Software Platform - Interfaces



OS Interface

- provides interface to RTOS services
  - Threads
  - Low-level memory management
  - Queues
  - Timers
  - Semaphores
  - Interrupts
  - Diagnosis

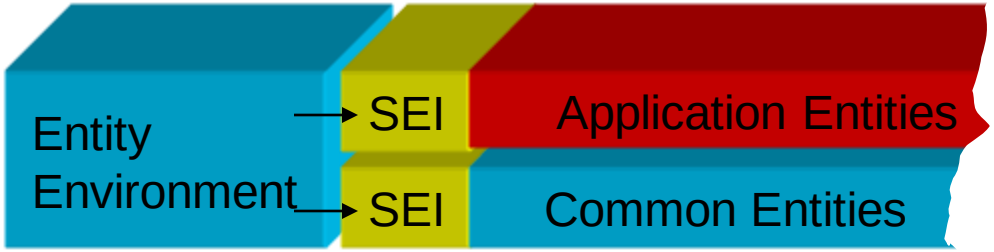
Generic Software Platform - Interfaces



GSP Interface

- Entity/thread services
- High-level memory management
- Messaging (send, receive)
- Timer services (single shot, periodic)
- Semaphore/mutex
- Trace services

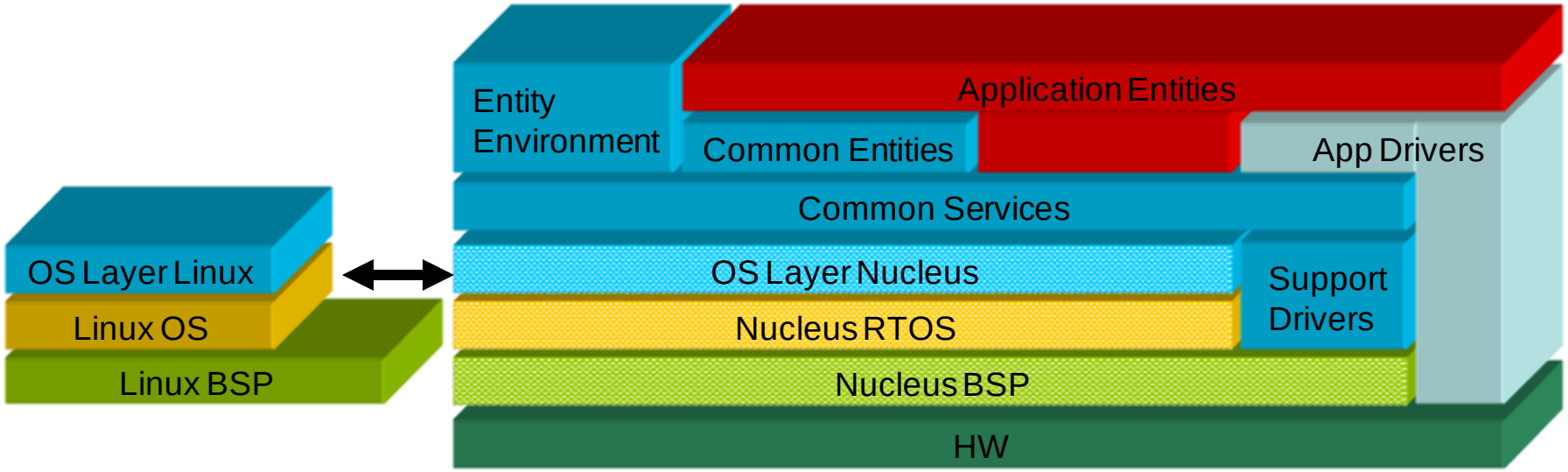
Generic Software Platform - Interfaces



Software Entity Interface (SEI)

- Provides interface of a software entity
  - creation
  - initialization
  - message passing
  - timeouts
- The SEI needs to be implemented in each GSP based software entity
- Interface functions are exported by the software entity (not statically linked)

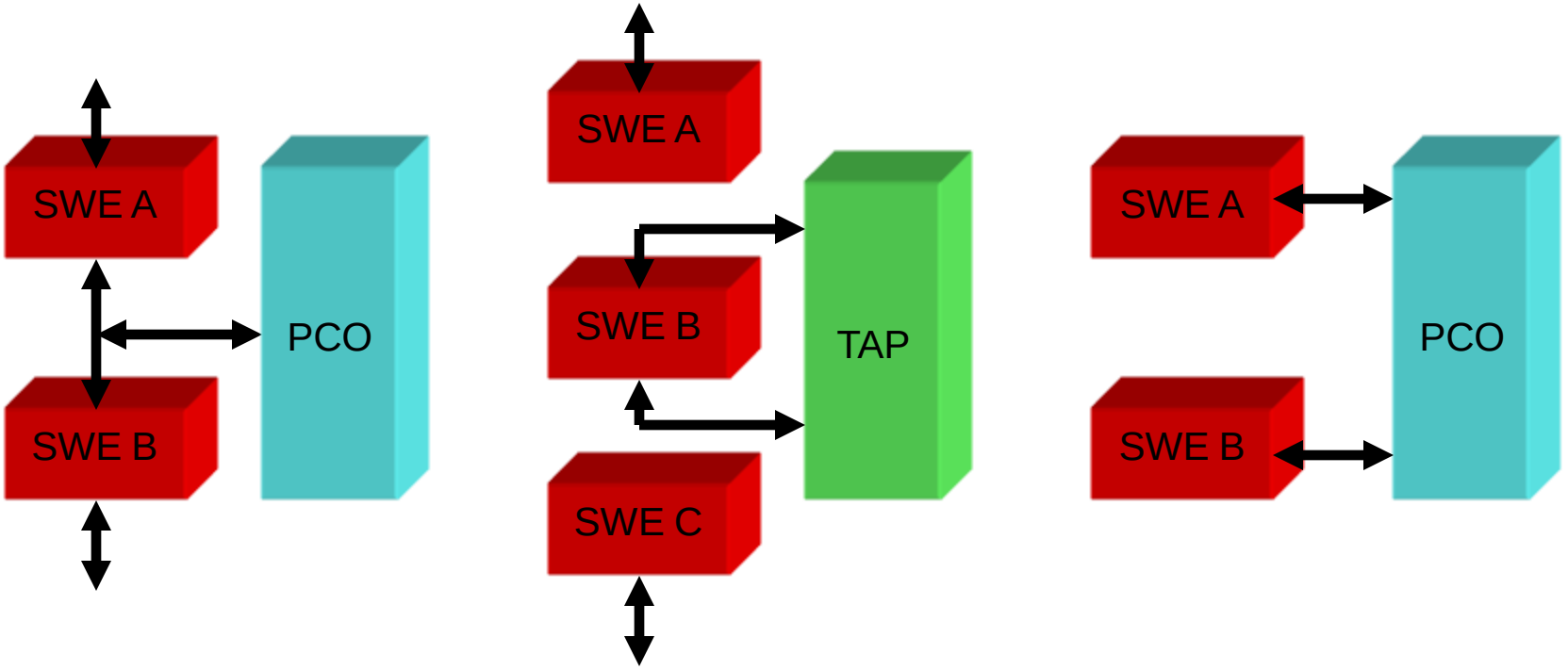
Exchange of Operating System



Changing from Nucleus to Linux

- Exchange OS Layer, RTOS and BSP
- Application software and major parts of GSP remain unchanged

GSP Test and Debug Functionality

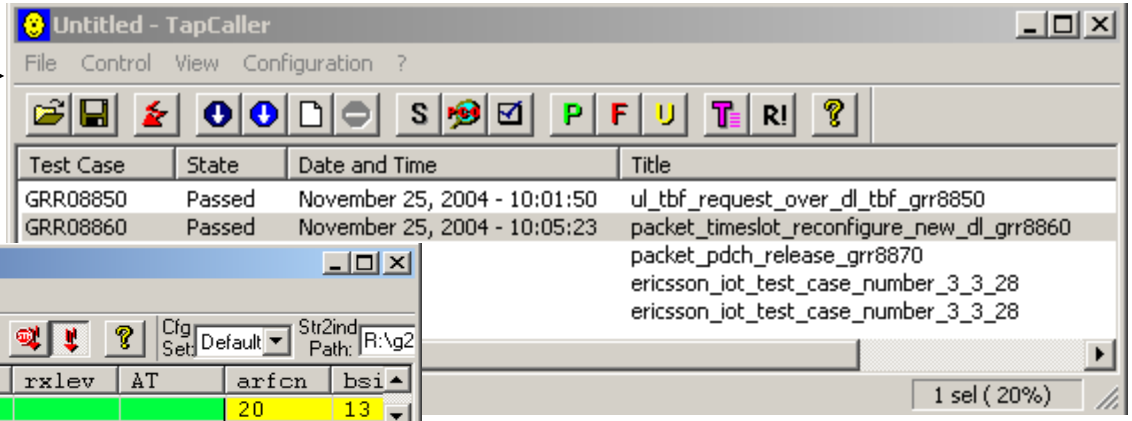


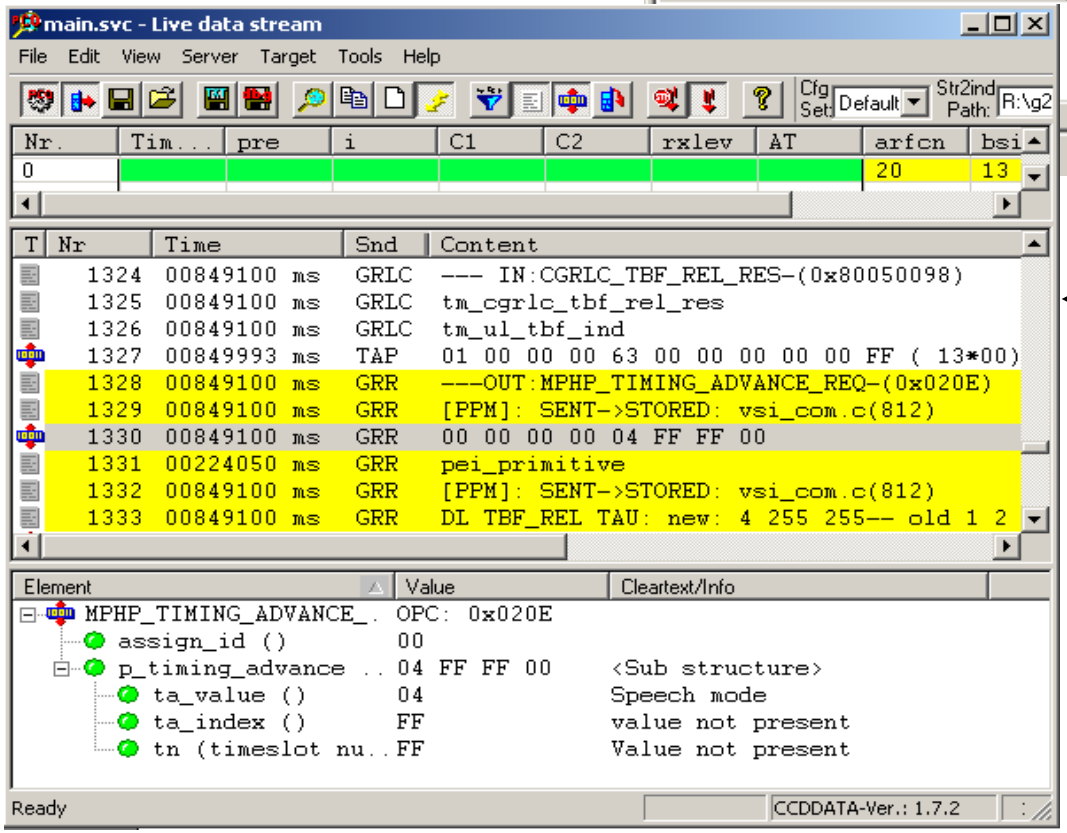
- **Message Duplication**, monitor communication in PCO
- **Message Redirection**, run test scripts with TAP
- **Tracing**, view debug information in printf() format in PCO
- all filters are set dynamically (not at compile time)

GSP Tools for Testing

Test Application (TAP)

- run test scripts

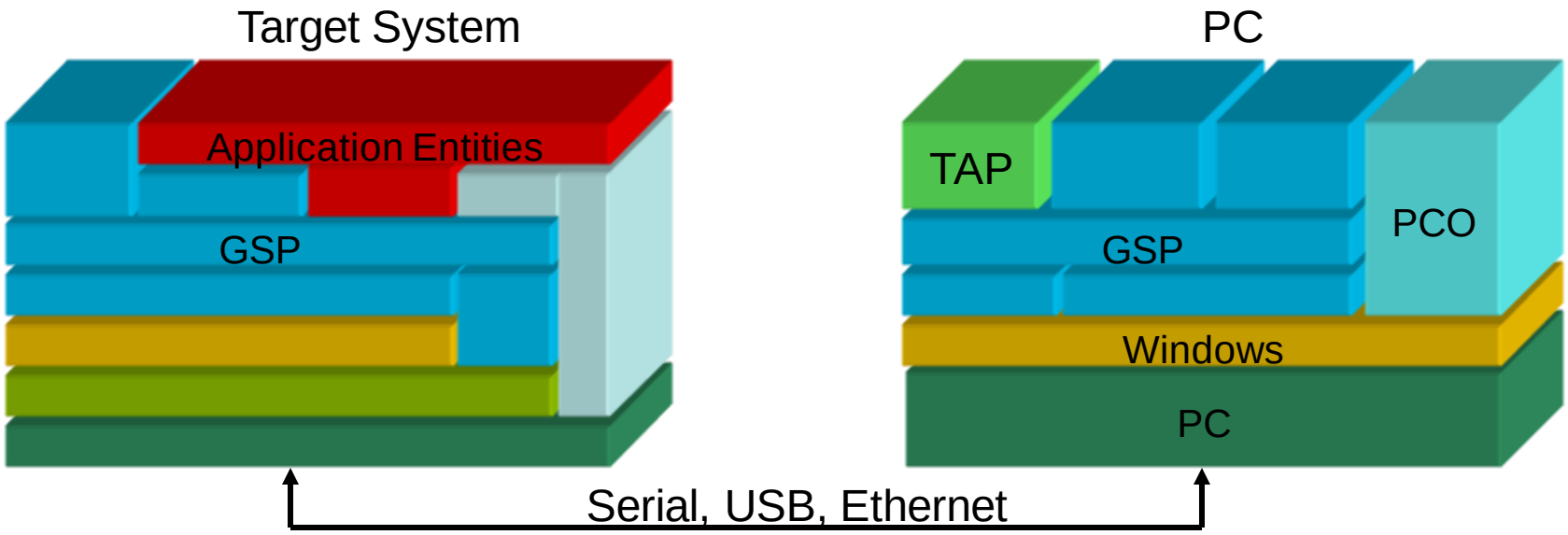




Trace Viewer (PCO)

- display traces
- decode routed messages
- set dynamic configuration

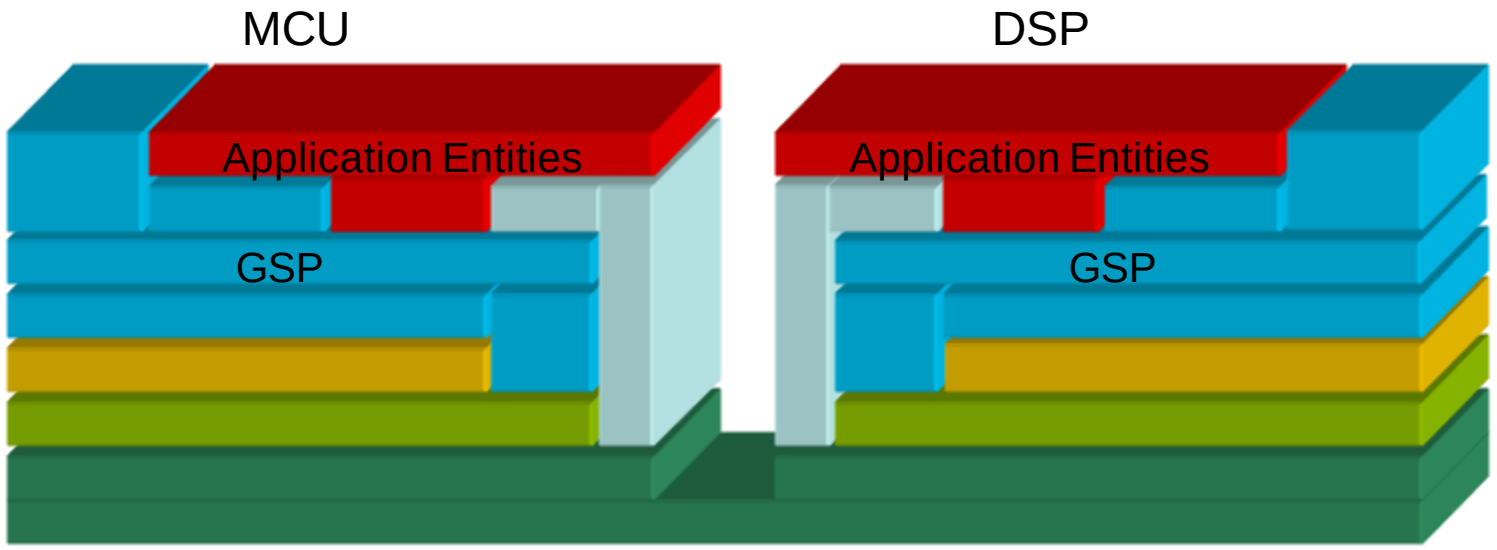
GSP in Target and Tools



Link target system to PC via test interface and connect TAP and/or PCO

- Log message exchange between the software entities in target system
- Log traces and routed messages sent by the software entities in target system
- Run testcases (send a messages into the target and check the response)
- Additional GSP based tools can be designed to run on PC

GSP on MCU and DSP



GSP running on MCU(ARMx) and DSP(TMS320C5x)

- GSP-'light' on DSP due to memory constraints
- Identical GSP API on MCU and DSP for services existing on both
- Transparent location (no need to know own location for most services)
- Transparent communication (no need to know location of partner)

State of Generic Software Platform

Operating Systems:

- Nucleus (ARM, MNT)
- Linux (kernel space, user space, RT Linux)
- Symbian (feasibility study done)
- WinCE (feasibility study planned)
- Win32 for test tools
- pSOS, Vxworks (currently no support)
- DSP/BIOS (planned to run on DSP)

Test interface drivers supported:

- UART
- Simulated UART (Shared memory)
- Socket
- USB

Conclusion

GSP supports you in

- designing operating system independent software
- reducing time to market by reuse of existing software
- increasing software quality by using built-in test functionality