SAND2011-7588 C

A Reference Architecture for Payload Reusable Software (RAPRS)

2011 Workshop on Spacecraft Flight Software

Richard D. Hunt
Sandia National Laboratories
P.O. Box 5800 M/S 0513
Albuquerque, NM 87185-0513
(505) 844-3193
rdhunt@sandia.gov

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Goals & Approach

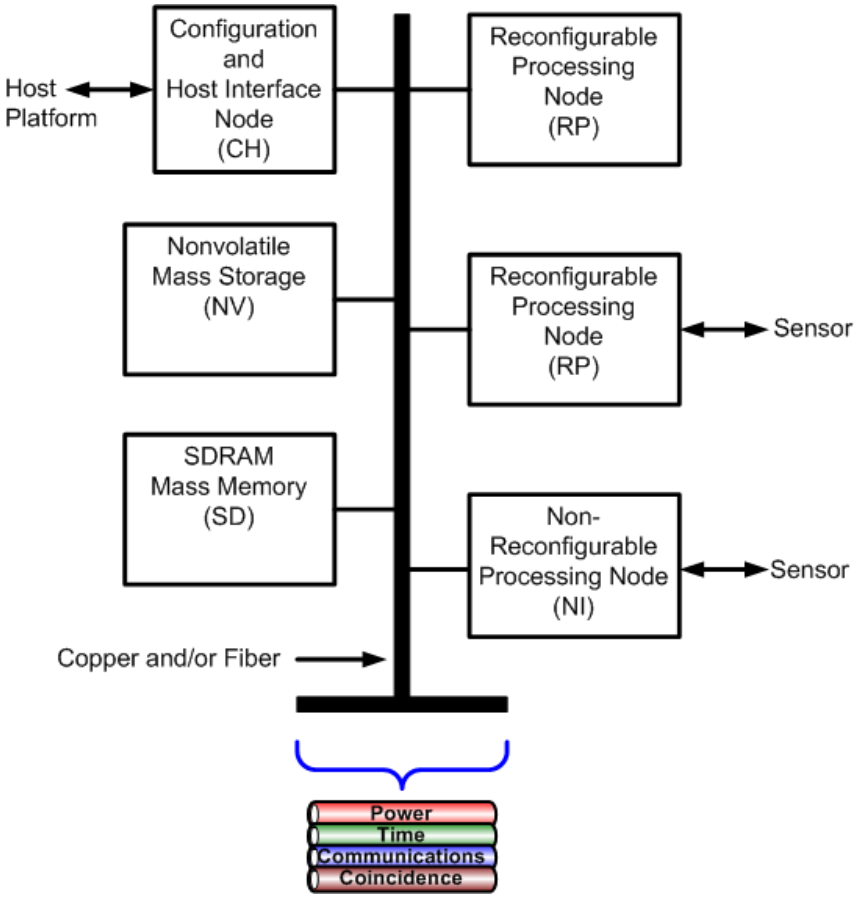
- **Goals**
  - Design a software architecture that can be reused across a broad range of payloads
    - →Eliminate custom, single-use-only designs
    - →Provide synergy by sharing software, developers, and tools
- **Approach**
  - Apply modern software architecture design styles
    - →Support distributed processing architectures
  - Identify current technologies and standards for implementing in highly-embedded systems
    - →Systems that have a fraction (<1%) of the processing resources of desktop computers

Motivation

- SNL is developing a standard data processing architecture called Joint Architecture Standard (JAS) and a new software architecture is needed to support it
- This would lower cost and risk to all programs by providing a mature platform to build payload software on
- Internal Research & Development funding was available
- We have a unique opportunity in which some of our payloads needed to be redesigned

Features of JAS

- **JAS is a modular, node-based architecture that uses**
  - High-speed serial data interfaces
  - Industry standard protocols
  - Hardware and software building blocks
- **Nodes**
  - Several processing nodes
  - Mass SDRAM & non-volatile memory nodes
  - Have 2 common HW components
    - → System monitor & communications (SMAC) port
    - → Point-of-load (POL) power converters
  - Number and type are determined by system requirements
- **JAS supports**
  - Rad-hard ASIC processors
  - FPGA-based soft-core signal processors
  - HDL for custom logic designs
- **JAS offers COTS-based development and test environment for rapid system demonstration.**



Sandia National Laboratories

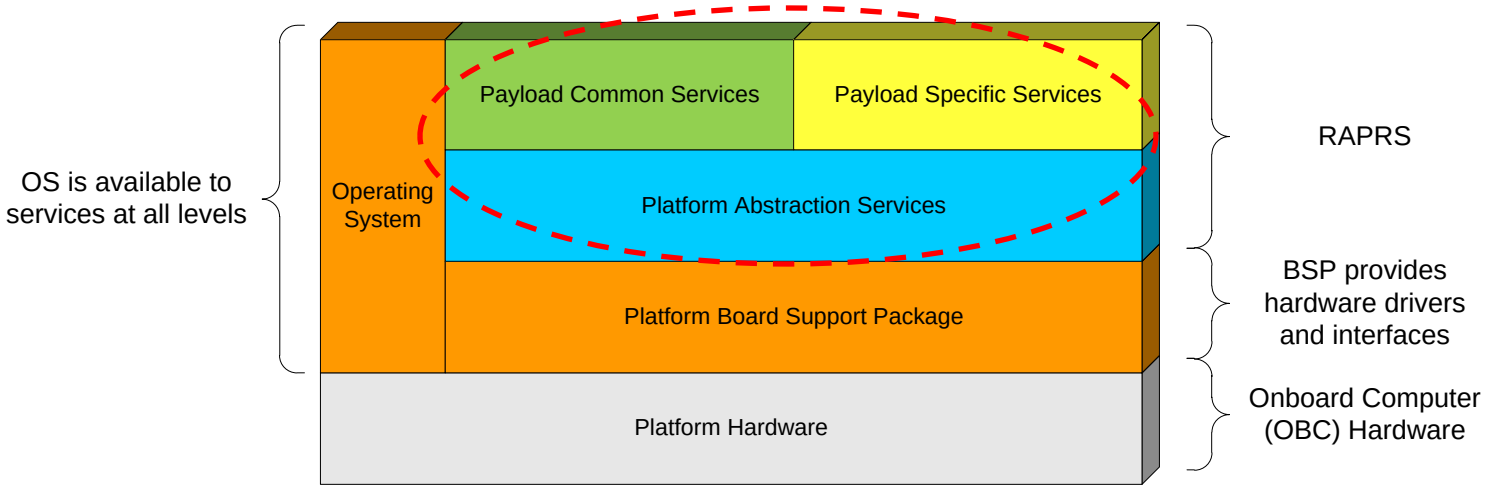
Software Requirements

- **Required**
  - Design a modular, scalable, and reusable platform
  - Support payloads with significant onboard processing needs
    - → Command and Data Handling (C&DH)
      - – Support tens of thousands of command and telemetry parameters
    - → Complex instrument control
    - → Sensor data processing from kbits/s to Mbits/s
  - Support payloads in a variety of orbits from LEO to GEO
    - → Real-time and intermittent ground system interaction
  - Support new and legacy ground systems
- **Desired**
  - Dynamically update software without system interruption
    - → Support new missions through software reconfiguration

Minimum Hardware Configuration

- **CPU – Hard or soft-core processors**
  - 32-bit processor with hardware floating point unit, 75+ MHz
    - →LEON3 SPARC, PPC-603, PPC-750
- **RAM – Depends on application requirements**
  - 16MBytes to 64MBytes
- **NVRAM – Store hardware and software applications**
  - 1Mbyte to 64Mbytes
- **PROM – Boot loader and applications**
  - 128Kbytes to 256Kbytes
- **Network – Depends on mission requirements**
  - Nominally >10Mbps for C&DH

Layered Architecture



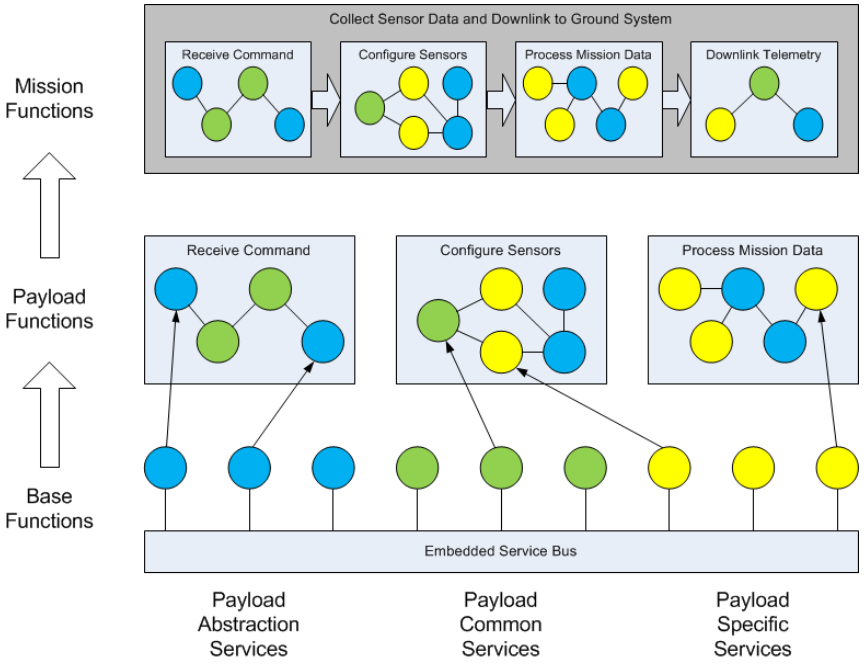
- **Platform Abstraction Services (PAS) provide a standardized and abstracted communication interface to all payload hardware**
- **Payload Common Services (PCS) provide many of the functions necessary for controlling a payload and communicating with the ground system**
- **Payload Specific Services (PSS) are the functions that are unique for each payload**

A layered architecture allows RAPRS to be reusable on different hardware

Sandia National Laboratories

Event-Driven Service Oriented Architecture

- **Combines modular services of SOA with an event-driven architecture to create a "reactive system"**
- **ED-SOA decouples software services and simplifies the interfaces between them**
  - Services send and receive "events" through a publish/subscribe interface
  - Events are routed based on "topics" to subscribing services



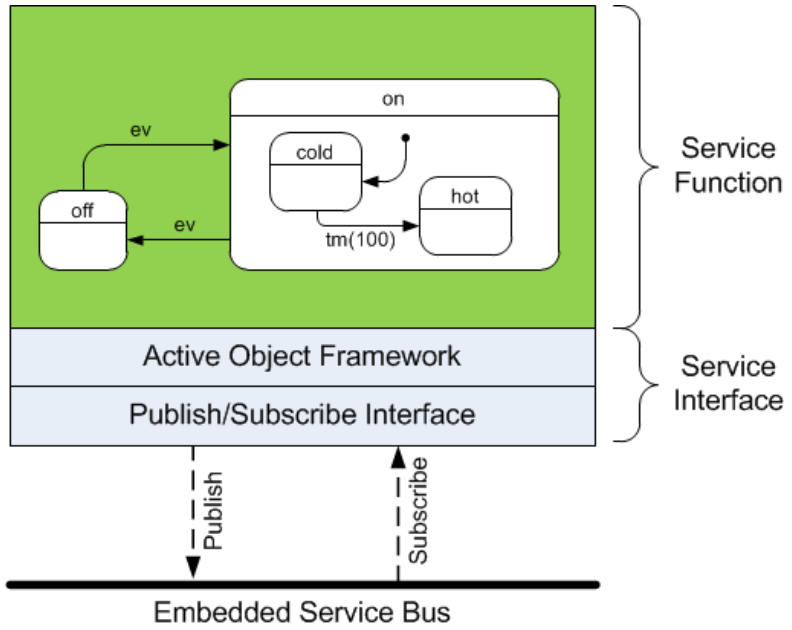
- **An Embedded Service Bus (ESB) allows services to be <u>seamlessly</u> distributed across JAS nodes**

Even-Driven SOA provides the <u>modular</u> and <u>scalable</u> framework for RAPRS

Sandia National Laboratories

A Software Service

- **Services are defined based on architecture analysis**
- **Implemented in terms of a standard framework**
  - Service Function
  - Service Interface

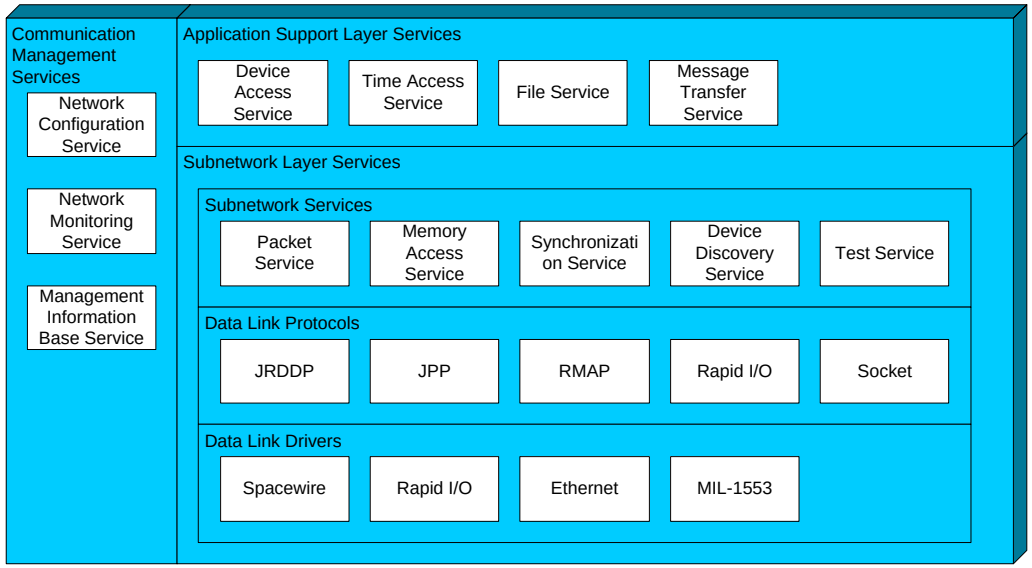


- **The framework is implemented with:**
  - An Active Object (AO) Framework provides the event-driven execution framework
  - Data Distribution Service (DDS) provides the ESB

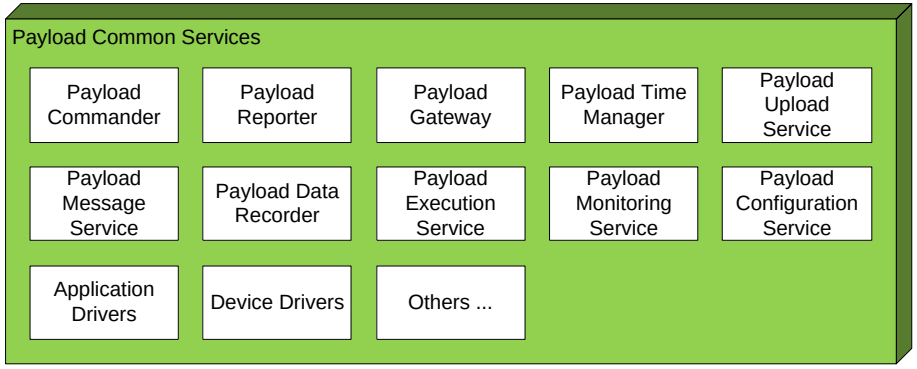
Sandia National Laboratories

Platform Abstraction Services

- **Provide standard interfaces to upper-layer services for accessing payload hardware**
- **Access payload hardware regardless of location or communication interface**
  - Hides the details from upper-layer services
- **Based on CCSDS-SOIS**
  - Services APIs are implemented as events
- **Subnetwork Layer Services can be implemented in hardware (VHDL) or software**



Payload Common Services



- Translate between F/G protocol and payload events
- Execute commands
- Collect and report SOH data
- Manage and distribute time
- Upload and store HW/SW apps.
- Collect and report messages
- Read/Write to a data recorder
- Deploy HW/SW apps.
- Monitor payload HW/SW state
- Distribute configuration data
- Templates for Application and Device Drivers

Programs can pick and choose the services they need

Embedded Scripting Languages

- An embedded scripting language allows services to be customized at runtime making them more reusable
- How it works
  - Develop the service with basic functions and hooks to run scripts
  - Implement program-specific functions in the scripting language
  - At run-time a reference to the appropriate scripts is passed into the service
- Examples
  - Commanding Service runs scripts to perform “complex” commands
  - Gateway Service executes scripts to translate between events and the F/G communication protocol
- Advantages
  - Services are more reusable
  - Scripts can change without having to recompile the software
- Disadvantages
  - Performance
  - Capability is limited by the features of the scripting language
- Lua can extend programs written C, C++, Ada, Java and others

Payload Configuration and Interface

```xml
<object>
  <name>Red Hill</name>
  <position>
    <lat>-33.69</lat>
    <lon>18.83</lon>
  </position>
</object>
```

```
object* obj = ...; // Parse XML.
const char* name = obj->name ();
position& pos = obj->position ();
float lat = pos.lat ();
float lon = pos.lon ();

delete obj;
```

- **The payload is defined in terms of XML data files**
  - CCSDS-XTCE is used as the schema
- **An XML Data Binding compiler converts XML elements to software objects**
  - Creates the classes as well as parsing and serialization code
  - Code Synthesis XSD/e XML Schema to C++ Compiler
- **Flight software uses the software objects to manage the configuration and state of the payload**
- **Ground software uses the software objects to send commands and process telemetry data**

Sandia National Laboratories

Status

- **Current R&D efforts have focused on proving the JAS hardware**
- **Software activities focused on developing infrastructure and demonstrating the architecture**
  - Demonstrated the layered architecture and abstraction
  - Demonstrated the event-driven framework using Rhapsody OXF
  - Implemented many of the Platform Abstraction Services
    - → Spacewire network configuration and monitoring
    - → RMAP and CCSDS packet protocols
    - → Time distribution
    - → Remote file system access
    - → Loading bit-files applications in FPGAs
  - Testing with RTEMS and Linux running on the LEON3
  - Porting Opensplice DDS to RTEMS to demonstrate an ESB
  - Gathering resource and performance data

Future Plans

- New programs are planning to use JAS/RAPRS and we are working on architectural concepts for them
- Continue to mature the framework and develop a library of new services over the next 1-2 years
  - Finish implementing the Payload Abstraction Services
  - Detailed design of the Payload Common Services
- As programs adopt this architecture, the library will grow so others can take advantage of existing services

References

- Opensplice DDS
  - http://www.prismtech.com/opensplice
- IBM Rational Rhapsody OXF
  - http://www.ibm.com/software/awdtools/rhapsody/
- Quantum Platform
  - http://www.state-machine.com/qp/
- XML Data Binding
  - http://www.artima.com/cppsource/xml_data_binding.html
- CodeSynthesis XSD/e
  - http://www.codesynthesis.com/products/xsde/
- XTCE
  - http://www.omg.org/space/xtce/
  - http://public.ccsds.org/publications/archive/660x0b1.pdf
- Lua
  - http://www.lua.org/about.html