

## The Implementation of a Plug-and-play Satellite Bus

Maurice Martin, Don Fronterhouse, Jim Lyke

Air Force Research Laboratory, Space Vehicles Directorate

AFRL/RVSE; (505) 846-5812

### ABSTRACT

Several years ago, the Air Force Research Laboratory (AFRL) began a research program to understand the complexity of aerospace systems and how, through technology, it would be possible to create them much faster. To underscore the ambitions of this work, we referred to this research as the pursuit of the “six-day” spacecraft. The six-day interval is marked starting with identification of a mission need and ending with a fieldable spacecraft ready to integrate onto a launch vehicle. This body of work culminated in the creation of a plug-and-play satellite bus (PnPSat) as a “clean-sheet” approach to spacecraft architecture. With PnPSat, we demonstrate how a complete spacecraft can be developed, integrated, and tested based on plug-and-play components and supporting software, design, and simulation technologies. This paper reviews the architecture and current status of the PnPSat project.

### INTRODUCTION

Complexity begets cost and expense, at least it has with DoD frontline space systems. A recent study found that all ten of ten major space systems each suffered from staggering (> \$1B US) cost overruns and multiple year schedule slips [1]. We are led to wonder if it is possible to reverse this trend, and we conclude that we must, since the systems of tomorrow will certainly not be simpler than the ones we build today. We find inspiration from many technologies on the “ground”, such as the personal computer, in which brand new components can be quickly integrated without rewriting software or hacking hardware. Or the scalable web-based systems, in which thousands of distributed servers can deliver service seamlessly, while having hardware and software being constantly upgraded.

We began several years ago to study how we might make complex systems, especially space systems, more quickly, coining terms like “responsivonics” (electronics for quick systems) and “peel-and-stick” or appliqué sensors. Under the influence of a study done by the Air Force Research Laboratory (AFRL) to study operationally responsive space (ORS) [2], we converged to a set of core ideas to make systems that could have a “plug-and-play” ease of assembly and integration that we take for granted in desktop computers. This approach was called Space Plug-and-play Avionics (SPA). The ideas grew more-or-less official since 2004 into a suite of open systems technologies that we have been convinced hold at some of the keys to combating the factors in complex systems that cause them to cost more and take longer than they “should”. We began to study them in ground testbeds, then a sounding rocket, eventually as part of a

spacecraft (TacSat 3). But we felt it was important to demonstrate more boldly that it would be possible to not just develop a system with a plug-and-play “port”, but to create an entire system from the ground up using plug-and-play principles. This system became the Plug-and-play Satellite (PnPSat), the focus of the present paper. We have created not simply the infrastructure of a plug-and-play system but its first prototype in a launch-ready form.

This paper is organized as follows. We will review the background and principles of SPA that we believe distinguish it among other modular open system architecture (MOSA) approaches. We next overview the basic architecture of PnPSat, and comment on current status. As PnPSat is a proof-of-feasibility exercise (that happens to be a flyable spacecraft), it provides insights on how we might scale the creation of many such plug-and-play systems based on the same principles.

### BACKGROUND

Much of the present work in PnP stems from attempts to create high-performance processing architectures for space systems [3]. Initially, our work focused on raw miniaturization through the use of advanced packaging technologies, but we later found that much of the problem in creating high-performance systems was embroiled in interfaces. For space experiment testbeds, which we developed for the MightySat 1 and STRV-1d (and for C/NOFS in its initial development), we began to try to constrain interfaces through the application of standards. We found that this was not straightforward. Simply invoking standard interfaces like RS-422, MIL-

STD-1553, even Ethernet did help reduce variations in components. It did not, however, make systems plug-and-play. The problem in plug-and-play was much deeper than these surface effects. At the same time, we were very successful in building scalable processing systems that employed switch fabric (i.e., Myrinet, later Spacewire) and eventually concluded that some combination of these ideas and a few others would be useful in reducing complexity in systems by embedding principles of scaling and some common approaches in configuration and message passing interfaces. Even before our work for Operationally Responsive Space (ORS), we converged on a set of principles that became the eventually technology we now refer to as space plug-and-play avionics (SPA). We initiated our first workshop on SPA in 2004. By 2005, we had established the responsive space testbed as a proving ground for ideas like SPA, and within a year had demonstrated rudimentary networks. By 2007, we launched the first sounding rocket using a four-port SPA network and had created an experiment for TacSat 3 (also four ports). The ideas of SPA evolved over this time, after nearly a dozen workshops and reviews.

SPA is a suite of technology concepts:

**SPA interfaces** (SPA-U, SPA-S, and future SPA-x). These are overlays onto commercial standards like USB (SPA-U), Spacewire (SPA-S), with accompanying protocols that extend the commercial approaches to support PnP better (especially Spacewire). Ideally, every component would use one interface to the spacecraft. As a single-point interface, SPA provides command, data, synchronization and power. This model works well for simple components, most of the ones outside of complex payloads and radio-frequency (RF) equipment, in which more complex interconnection relationships are required (we have studied routable manifolds for electromagnetic signals in anticipation of a SPA-like treatment of RF [4]). The definition of any SPA-x interface includes hardware, pin/connector definitions, the protocols of devices and how they join the network, and the messaging infrastructure. A SPA network is a collection of endpoints and routers. In the case of hosted networks, such as USB, hubs perform the same role as routers, and at least one distinguished node must exist to support the hosting function. Any endpoint can exist at any compatible point in the network. Theoretically, we should be able to mix-and-match SPA devices (topology agnosticism), as well as interchange and mix networks through bridge nodes. The ideas of SPA networks and interfaces have been captured in draft standards, and we have demonstrated the ability to freely interchange components in SPA networks in our various demonstrations in the responsive space testbed.

Supporting these interfacing requirements in a standard way is important and straightforward, but not trivial. In analogy, the developers of peripherals in personal computers face a similar challenge. A keyboard developer would be hard pressed to implement an integrated circuit that implements the USB standard from scratch, yet must somehow bridge the unique circuitry of the keyboard to the uniform plug-and-play infrastructure of the personal computer (i.e., the USB interface). This is resolved by procuring a pre-built component that contains the USB interface. This pre-built component (or intellectual property core) was developed by a third party in painstaking detail, and complies with the USB interface standard. It is no longer necessary for a keyboard developer to understand a very involved specification, but a much simpler one.

These were the thought processes underlying the creation of the ASIM. Therefore, the ASIM is a "worked out" embodiment of a SPA interface with a lot of useful hooks to simplify the chore of making a non-SPA component into a SPA-compliant component. With ASIMs, we do not "see" the raw interfaces of components, but launder those indigenously interfaces through the ASIM to make a "non-SPA" device operate effectively *as* a SPA device. We call this process "legacy conversion", since it means that we take components designed for other missions and engineer plug-and-play overlays on them (i.e. through ASIMs, xTEDS, etc.). (Over 100 ASIMs have been employed in the development of PnPSat.) The ASIM is not the standard, it implements the standard. The difference is important, as we believe the emergence of third party USB developers has proven in the personal computer market. Developers are free to implement SPA in a custom way, or to use pre-developed solutions, as needs dictated. Proprietary advantages must be confined to the interior of SPA devices, not in custom interfaces.

The research in interfaces is an ongoing activity. We consider SPA-U (USB) and SPA-S (Spacewire) most mature, but have investigated a number of interface supplements, including higher speed, wireless, and high-determinism approaches.

**Electronic datasheets and ontology.** Electronic datasheets define the "PnP universe". At a simple level, the electronic datasheets, which are referred to as extended Transducer Electronic DataSheets (XTEDS), provide a listing of the services each SPA device can provide. Everything that is worth knowing about a device should be in the XTEDS. If it is not in the XTEDS, it should not be important to the system. Consistency in the construction of the datasheets is very important. Without it, we have in effect, a "Tower of Babel", and the PnP universe is broken. For example,

"temperature" must always be "temperature". This attention to consistency speaks to the need for a "Rosetta stone" of plug-and-play. Even for the PnP/Sat we do not yet have this ultimate Rosetta in any robust sense, but our first generation ontology is adequate for our purposes.

#### **The software side of PnP, Satellite data model.**

Software infrastructure is fundamental to effective construction of PnP systems. Our approach combines features reminiscent of object resource brokers (such as CORBA), remote procedure calls (RPCs), and even web services. The resulting synthesis is called the "satellite data model" (SDM). This SDM provides two key features present in any plug-and-play system: "discovery-and-join" and "look-up services". The former is about the automatic finding of devices, extraction of their xTEDS descriptions, and registration of services. The latter is comparable very roughly to an embedded "Google". The SDM provides support for all of these feature and more, such as the innate infrastructure to manage network topology, distribute SDM on multiple processing systems, and much more.

Structurally, SDM consists of five major components. The Data Manager provides a query and discovery mechanism (i.e., the embedded "Google") whereby other applications can determine what data is available in the system, who provides it, and how to get a hold of it. The ability to be able to find single data elements within the data system is a key capability of a data centric architecture. The Data Manager maintains a database of all xTEDS that have been registered by components and applications (yes, software modules also have electronic datasheets!). If the data network is composed of two or more sub-networks (for example SPA-U and SPA-S), a Sensor Manager is used to bridge the two networks. A Network Manager is used to discover the elements of the data network, their addresses, and in the case of Spacewire the path routing between any two elements on the data network. A Processor Manager resides on each processor in a distributing computing system and provides for the orderly execution of tasks on that processor. It is responsible for the underlying messaging services, and for dynamically selecting applications (tasks) to execute that are compatible with its resources. Tasks to be executed are posted to a Task List maintained by the Task Manager. The Processor Manager periodically reviews the current Task List and requests those that match its available resources. The Task Manager then assigns the task to one of the responding Processor Managers for execution. The Processor Manager also provides a heartbeat to the Task Manager to guard against processor failure. We call the Satellite Data Model sideware because it is only used to discover the

network and the data elements in it. Once a message has been subscribed to the data flows from producer to consumer as peers and SDM does not get in the way.

#### **The advent of automated (pushbutton) toolflows.**

Plug-and-play approaches support rapid design approaches with a goal of translating user needs into the specification of a buildable spacecraft. Our goal is to achieve toolflows that employ an open framework, in which different groups can make tools that can interoperate and be shared. We ultimately consider the analogy to internet browsers, which process documents created in an open language (html). Individuals are free to choose browsers, which may themselves be open source or proprietary (for example, MSIE, while given freely, employs a proprietary code base). Browsers have evolved to support a number of other web formats and the provision for third-party extensions /add-ins (often installed for the user on demand). These concepts constitute a powerful open framework.

Similarly, we view that it is essential not to be locked into one group's tools to manage the design process, but to have the freedom to ultimately mix and match tools. In such a milieu, it will eventually be possible to employ web-driven automated design flows for satellites that link directly to vendors. Pushbutton toolflows for satellite design automation unify the concept of component libraries and will ultimately bring together a supplier community, who will be able to publish hardware and software modules in the form of electronic component libraries. It could be a tool that generates not just a bill of materials, but some test scripts, the console for running the spacecraft, and other useful documentation and auxiliary software components. Their PnP components would appear as selectable options in a rapid design flow. This will make it possible to build PnP/Sats without having every component (or maybe any components) on one's shelf. It doesn't take a lot more imagination to see that one could link this infrastructure to rapid prototyping shops, so that custom brackets, etc. could be ordered online, from specifications automatically generated as by-products of this framework- We presently have created the Mission Spacecraft Design Tool in the responsive testbed as a first step for PnP/Sat and a step towards the ultimate push-button toolflow.

*Test bypass.* Building a satellite very quickly requires more (not less) test infrastructures. The concept of test bypass allows us to test SPA devices *in situ* without special added hardware / software. It does at present require a special debug port (4pins), but even this could be eliminated (studying wireless test bypass, which

could make it simple to log in to individual components). The spacecraft might "think" it's even flying while sitting in the testbed. This is a very powerful idea that provides an entire system with a facility similar to a software debugging tool.

## THE PNPSAT SPACECRAFT APPROACH AND DEVELOPMENTAL PHILOSOPHY

PnPSat represents the first spacecraft of its kind, not from outward appearance but from first principles as platform based on a self-organized network of self-describing components. It is modular, but the application of modular approaches in spacecraft is not a new concept. PnPSat can be viewed as the combination of modularity and complexity hiding. Even the panels of PnPSat are SPA devices, and they contain routing infrastructure and power management to the spacecraft. Most of its wiring harness will be invisible, recessed within panels. PnPSat is a technology experiment to establish the feasibility of software defined systems based on PnP. Over the last several years we have touched every aspect of satellite design and construction test and operation to find those areas that inhibit the six-day spacecraft. What we found is that we must simplify the interfaces by hiding complexity. In PnPSat we are applying the principles of plug-and-pay to the mechanical, electrical and software interfaces.

What is a plug-and-play satellite? It is a modular satellite with open standards and interfaces, self describing components, and auto configuring system. This results in system integration and testing tasks that can be automated and are themselves simplified. Modular spacecraft structures that allow components to be mounted either on the inside or outside on regular grids. We are currently using a 5 cm x 5 cm grid. Modular flight software that is both easy to maintain and can be reused for various satellites and is intrinsically autonomous. Our goal is to have a satellite capable of maintaining its own health and status and only needs to talk to the ground by exception and for user tasking. High-performance-computing-on-orbit (HPCOO) provides processing to the user to support both the autonomy and on orbit processing of sensor data. We want to be able to provide to the user not only raw data as appropriate but also the ability to process information as mission needs dictate. We have worked out concepts for tactical user interfaces that allow the user to directly task a satellite to extract mission products. We have worked with a distributed community of experimenters to develop plug-and-play experimental payloads. Distributed power systems support plugging in a battery onto one panel and solar arrays onto another. Main bus power and charging grids are distributed throughout the spacecraft allowing

access to the main power grid from anyplace on the spacecraft. This access is protected with circuit breakers and intelligent, plug-and-play power management. We have also investigated plug-and-play launch vehicle interfaces.

**Requirements.** PnPSat requirements fall into three basic categories: the overall Responsive Space program requirements, the PnPSat program requirements, and the primary system capabilities that need to be demonstrated. The Responsive Space program requirements include demonstration of the viability and maturity of a modular plug-and-play architecture. It is important to be able to transition technologies to other satellite programs, and despite objections to the contrary, spaceflight appears to be an important prerequisite to major spacecraft adoption of new technologies.

The primary system capabilities include being able to demonstrate rapid design, assembly and test. Of course, we must be able to demonstrate modular plug-and-play, including both SPA and the Satellite Data Model. We must demonstrate distributed systems including power, thermal, computing, and control. On the software side, we must be able to demonstrate robust autonomy including both dynamic schedules and activities.

**PnPSat Architecture.** One way to look at the PnPSat architecture is as a personal computer with 48 USB ports. More realistically, we view the spacecraft as having three basic partitions. First, we have the basic bones of the spacecraft, the substrate upon which all components are attached. These "bones" includes the spacecraft structure, the power grids (both main and charging), the SPA infrastructure, and thermal control. As the second partition, we add components that provide robust performance including the autonomous flight software; the quantity of high-performance computing; power generation and storage; guidance, navigation, and control components; and the communications radios for both tactical and TT&C. The first two partitions pertain to the spacecraft, the conveyance that delivers the third partition to space – the payload – to perform its mission. We add then those mission sensors that provide customization for warfighter needs. From the perspective of building and testing the satellite, we must consider assembly, integration, and test; the ground systems that control the system and provide user access; and the launch systems.

The PnPSat structure features modular panels to support quick assembly and the flexibility to mount components in multiple places. There are standard plug-and-play mechanical and electrical interfaces that

can accommodate 48 experiments are components located on either the interior or exterior surfaces. A tactical satellite requires approximately 25 to 28 components, which provides us with sufficient flexibility to mount the components based upon mass, thermal, power, and FOV requirements, among others. Electronics infrastructure and harnessing is



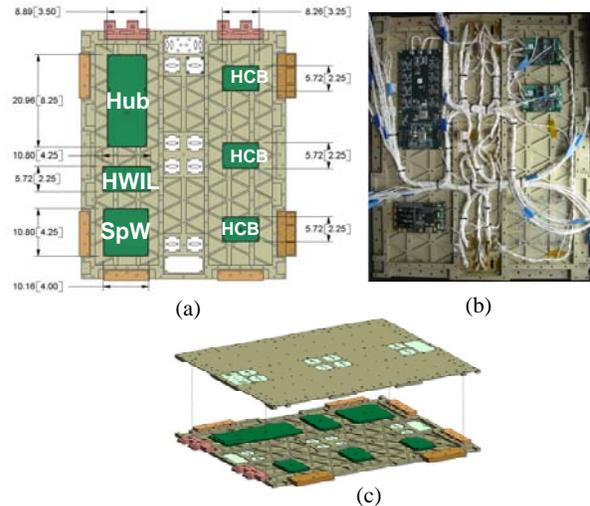
recessed within each panel to increase available footprint and volume for the plug-and-play components and experiments. Locking hinge joints allow panels to rotate about the hinge line for easy access to the interior. Inter-panel jumper the harnesses across joints allowing the plug-and-play electrical network to remain intact throughout assembly, integration, and test. This means that we can determine if a component is working as it is assembled on the spacecraft. Currently, the panels are machined from 6061-T6 aluminum. The current structure is 51 x 51 x 61.2 cm and weighs 34.7 kg excluding the launch vehicle adapter.

One of the advantages of the folding PnPSat concept is that it can be changed easily to various configurations to support requirements for different stages of the project. At first it can be opened up into a flat configuration for internal components to be mounted and tested. Then it can be closed, while still active, for the external components to be mounted and tested. Panel to panel joints are pinned to allow panels to be rotated from the horizontal flat to vertical folded configuration. Then securing the joints with bolts

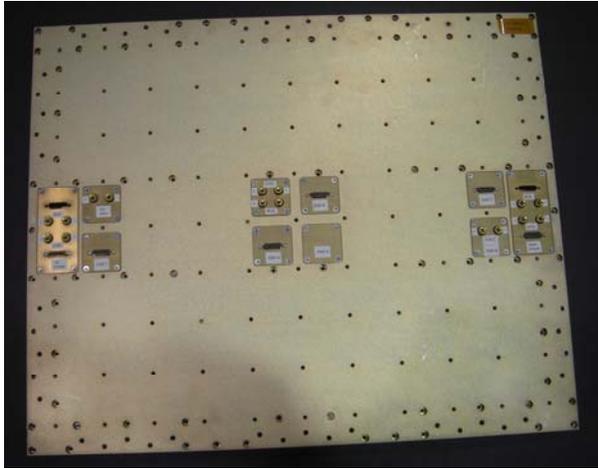
provides for a rigid structure. Individual panels or sets of panels can be integrated and tested in parallel.

One of the modularity keys is to have a standard simple mechanical interface between the components and the structure. We have established a simple, standard mechanical interface to increase the flexibility and to speed integration. We have initially selected a 5 x 5 cm grid pattern that goes completely across the internal and external surfaces of all panels. The holes are threaded to support #8-32 fasteners. The hope is that eventually new components and experiments will be designed to accommodate this interface. In the meantime, existing components can be integrated with a simple adapter plate. This is the approach we are using on PnPSat to match legacy components to the modular structure.

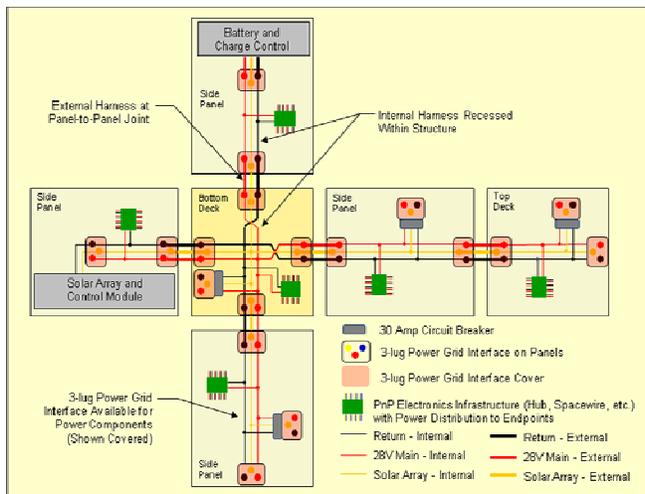
The SPA electronics infrastructure is recessed within the interior of each panel including boards and inter-board harnessing. The power and data services provided to each of the eight SPA endpoints on each panel are handled by the robust hub. Panels are networked together, including power and data using the inter-panel harnessing. Once the SPA infrastructure has been installed and tested the panel halves are bolted together to form an EMI tight enclosure.



Each of the eight SPA endpoints has a standard electrical interface for components and experiments. For PnPSat the standard electrical connector is a 25-pin micro-D containing data (both Spacewire and USB), power (up to 4.5 amps @ 28v), time synchronization pulse, test bypass interface, and single point ground. Endpoints can be located on either the interior or exterior surface of the panel. Batteries, solar arrays, and power supplies have access to the power grids through 2-lug interfaces.



**PnPSat Components.** There are 25 PnPSat components plugged onto the structure. These include two coarse sun sensor assemblies, three reaction wheels, three magnetic torque rods, a fine sun sensor, a magnetometer, two batteries, FITS solar array, GPS radio, two packages of HPCOO processors, an Intelligent Data Store, and a TT&C radio. We believe all components that plug onto the structure should be plug-and-play. Our initial studies have shown that by recessing the electrical infrastructure and harnessing inside the panels, we significantly increased flexibility for component and experiment mounting.



To enable a plug-and-play power system, the bus power grid is composed of two separate grids: the main power grid, and the battery charging grid. These grids extend across all of the panels allowing batteries, and the solar arrays to be connected to their grids from anywhere on the bus. High power components can gain access to the main power grid via 30 amp circuit breakers. The battery charge control electronics and the solar array controller are also SPA components. By separating the charging and main power grids, we enable a Phoenix

mode, where even if we disconnect the main power grid due to low battery charge, we can still use opportunistic photons to charge the batteries. After the batteries reach sufficient charge, the battery and charge control electronics reconnect the battery to the main power grid and the satellite reboots.

The SPA infrastructure consists of the ASIM, robust hub, hardware in the loop router (for ground testing only), the Spacewire router, and the high power circuit breakers. The ASIM is used to interface legacy components to the SPA network. The ASIM has two major functions. First, it is charged with the care and feeding of the attached component. Second, it presents a standard plug-and-play interface to the SPA network. The ASIM contains the xTEDS that defines the devices' data products, accepted commands, supported interfaces, and services provided. This allows each component to be self describing to the SPA data network. In addition, the ASIM provides a very accurate, real-time clock, and the hardware-in-the-loop test bypass interface.

One of the fundamental changes being implemented in PnPSat is the concept of a data centric architecture. Traditional systems engineering is component centric, relying upon a detailed component interface control document (ICD) to enable system configuration. SPA enables us to focus more on the data rather than the details of the component. Data can be described, moving from the more fundamental to the more specific, as the basic physics, measurable quantities measured through a measurement process yielding variables and qualifiers that we provide names and formats, and gather all of this up into the ICD. Now if we were able to agree upon the meaning of measurable quantities - for example, attitude or position or pressure or temperature - and place that in a Common Data Dictionary (CDD) for all to share and place the variable names and qualifiers and their formats in the xTEDS. Then we could implement a standard SPA interface and get rid of the Interface Control Document. In this way, we have defined both a plug and a play interface, where that data interface is based upon a common standard (CDD) of what data means that is distributed to all, a standard data interface expressed in a standard language (XML), and the electrical interface based upon a common SPA standard.

The robust hub provides both a USB hub, and endpoint power distribution and monitoring. Each SPA endpoint can be supplied up to 4.5 A @ 28 V protected by a circuit breaker. In addition, there is a current monitor on each endpoint with a soft breaker that can be set based upon the power required for that component as

described in its xTEDS. In addition, the robust hub provides control of the high power circuit breakers. The robust hub uses an ASIM to provide power interfaces and control functionality, much like any other component.

*HPCOO Components.* To host the SDM and processing infrastructure of PnPSat, as well as provide on orbit data storage, we are developing an Intelligent Data Store (IDS) that is fully SPA compliant. The IDS uses a Vertex 2 FPGA, with up to 4 32-bit Microblaze processors and 512 MB (with a potential of 11 GB) of error corrected flash and 128 MB of error corrected RAM. The IDS resides on the Spacewire, high-speed data network and runs SDM applications and provides file storage and retrieval for system configuration data, application executables, and telemetry data. We had previously explored the use of a more powerful processor, but were unable to address a number of development concerns for PnPSat

*Autonomous Flight Software.* Assembling a spacecraft in two to three days means does not provide occasion to write much if any custom software manually. We have endeavored to engineer modularity / reuse in software inasmuch as we have in hardware. This engineering includes the ability to develop software applications before the satellite mission or the specific components of the satellite are known. To facilitate the independent and concurrent development of hardware devices and software applications, we have developed a sideware application called the Satellite Data Model (SDM). SDM allows for the last-minute integration of independently developed hardware and software while supporting self configuration and self discovery. SDM is the play side of modular plug-and-play. It also provides a support model for fault tolerance to loss of devices, loss of software applications or services, and loss of SDM components.

There are two ways to look at the flight software architecture. The SDM discovery model provides for a flat architecture, where any application can get or provide data from/to any other application or from/to any component as necessary. This is an extremely flexible architecture, but more difficult to manage. We also have a more hierarchical architecture that is composed of controllers, agents, and managers that is conceptually easier to manage. It is important to remember that controllers do not own the devices they use to provide control. For example, the ADCS Controller does not own the reaction wheels, but does use them to control spacecraft attitude.

The PnPSat flight software core functionality is implemented as a group of autonomous activities. An activity is defined as a function that requires

coordination of multiple subsystems and needs to be scheduled. Implementing flight software using SDM provides usability beyond just PnPSat. There are five basic categories of flight software in the hierarchical model. Subsystem controllers (for example Power, Communications, Computing, Thermal, ADCS, and Sensor) support both planning and commanding interfaces. System order is maintained by an Activity Manager that keeps the schedule and places activities to be executed in the schedule based upon time window and priority. We break priority into both a base priority associated with an activities importance to the satellite mission and an urgency that is time-dependent. For example, an activity to charge the batteries becomes more urgent the greater the depth of discharge. When it is time for an activity to be executed, the Activity Agents enables the associated Activity Agent. Activity Agents implement the basic activities of the satellite such as charging batteries, maintaining thermal control, collecting imagery, and safe mode. Activity Agents provide the heavy lifting to get things done and are required whenever more than one subsystem must be coordinated. Utility support applications such as coordinate transforms, orbit propagators, and celestial almanac, provide general-purpose support. Finally, there are the general purpose applications, such as satellite protection, image processing, etc. that are not associated with any specific activity.

Perhaps a PnPSat separation timeline will help to illuminate how all the various controllers, agents, and managers work together to bring the satellite up from cold at launch a fully functional system. First, the separation switch closes as the satellite leaves the launch vehicle allowing the battery ASIM to provide battery power to the bus at which point the robust hubs boot providing power to the endpoints. As each ASIM boots, it provides control to its attached device. The WSSP ASIM boots the WSSP processors and loads and executes SDM. After network discovery by the Network Manager, the Task Manager is started and retrieves the initial Task List from the IDS. The list includes the Subsystem Controllers, Activity Agents, Activity Manager, utility applications, etc. The Solar Array Activity Agent will place a deployment activity in the schedule via the Activity Manager and when executed by the Activity Manager will reduce tip off rates, deploy the solar arrays, and request ADCS go to sun point mode. And then the normal activity agents take over and the satellite is up and running.

*Building PnPSat.* At the time of this writing, the PnPSat is near final integration in the responsive space testbed and is to enter environmental testing. The schedule is amazingly aggressive, though certainly not a six-day schedule (do not confuse “doing something fast” with

“creating the ability to do something fast”. We have been trying to do both!). After we complete environment testing and final integration, we will disassemble the spacecraft and undergo more representative time trials in which we hope to assemble the entire spacecraft from primitive elements in less than four days.

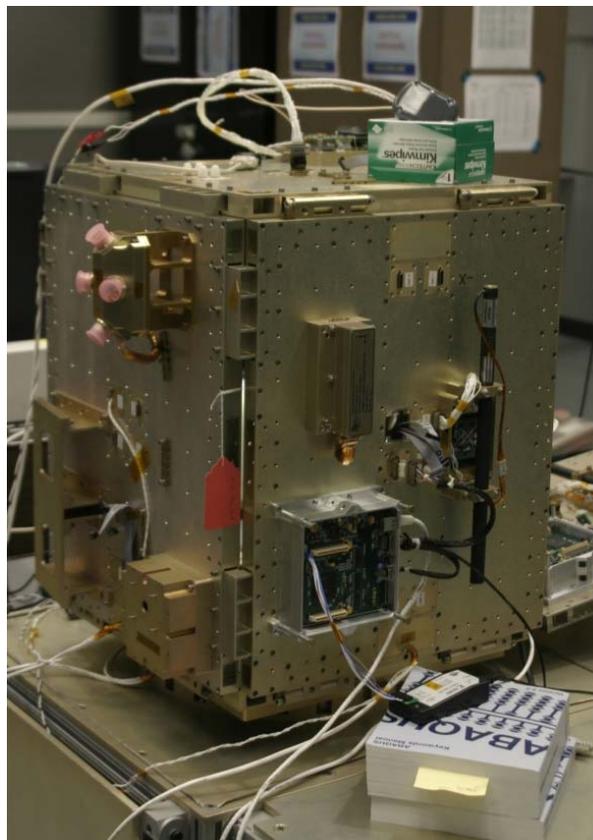
PnPSat is unusual in that it has been a spacecraft built before the idea of what payloads would be flown had been settled. At this point, we have several modest payloads, including two visible imagers (both SPA devices), a SPA AIS receiver, a SPA version of USAFA’s MESA instrument, and a SPA version of an industrial beam steering mirror (BSM). One of the cameras will view the solar array deployment, and the second will image and track stars. The BSM will examine some of the issues in rapidly integrating SPA devices with precision mechanical features. More significantly, we will demonstrate with MESA and BSM the ability for external organizations to develop compliant SPA devices. We have several other experiments in development as backups, such as a high-performance processing system developed by AFRL referred to as the Wafer Scale Signal Processor.

## CONCLUSIONS

In this paper, we have described the first spacecraft ever developed based on a modular open system architecture (MOSA) framework, and we have outlined the core principles of the underlying space plug-and-play avionics (SPA) technology. The spacecraft (at the time of this writing) is nearing completion at the Kirtland Air Force Base in the responsive space testbed. We have demonstrated a number of key technology concepts relating to the creation of a scalable PnP infrastructure, modular hardware and software.

## ACKNOWLEDGMENTS

The authors would like to thank the AFRL Space Vehicles Directorate, especially the Responsive Space technical area leadership, the Small Business Innovative Research program, and the Operationally Responsive Space thrust for its strong support of this effort. We are grateful for a talented and distributed development team who believe in the core principles of plug-and-play and its “message of hope” for battling complexity in systems.



---

<sup>1</sup> Marco Cáceres , “Cost overruns plague military satellite programs”, Aerospace America (publication of AIAA), January 2006, pp 18-20, 23.

<sup>2</sup> Wegner, P. and Kiziah, R. "Pulling the Pieces Together at AFRL – Space Vehicles Directorate", Proceedings of the 4th AIAA Responsive Space Conference, Los Angeles, CA, 24-27 April 2006

<sup>3</sup> Lyke, J., “Space-Plug-and-Play Avionics (SPA): A Three-Year Progress Report,” Proceedings of the AIAA Infotech Conference, 7-9 May 2007, Rohnert Park, CA.

<sup>4</sup> Wilson, W., Lyke, J., and Contino, P. “MEMS-based Reconfigurable Manifold”, presented at MAPLD 2001, Johns Hopkins University.