Government-owned CubeSat Next Generation Bus Reference Architecture

Vincent Riot, Lance Simms, Darrell Carter, Todd Decker
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550; 925-422-9798
riot1@llnl.gov

Jim Newman, Lara Magallanes, Jim Horning, David Rigmaiden
Space Systems Academic Group, Naval Postgraduate School
1 University Circle, Monterey, CA 93943; 831-656-2487
jhnewman@nps.edu

Meagan Hubbell, Dave Williamson
National Reconnaissance Office
14675 Lee Road, Chantilly VA 20150; (703)808-6892

ABSTRACT

The number of CubeSats and small satellites placed in orbit has been growing exponentially since 1999 as demonstrated by more than 40 CubeSats being launched in the last quarter of 2013 from the USA alone. While CubeSats were initially used for academic purpose and generally tailored towards technology demonstration, it has become more evident that small satellites can play a role in some operational contexts such as earth observation, space weather, or situational awareness, to name just a few. In the past, each institution involved in Small Satellites has often designed their own proprietary system with regards to communication, software, avionics, and command and control, with incremental improvements based on previous successes. While this may make sense in an academic environment, where it provides students with a wide range of learning opportunities, it distracts teams exploring scientific or operational missions from focusing primarily on the payload technology. Building upon previous work funded by the National Reconnaissance Office (NRO) and known as the Colony I and Colony II bus programs, the Lawrence Livermore National Laboratory (LLNL), in partnership with the Naval Postgraduate School (NPS) is developing a CubeSat bus reference architecture and a set of minimum specifications useful for government applications. The architecture has application to software, electrical, and mechanical interfaces and aims at providing a flexible platform that can be endorsed by industry, supporting interchangeability of components while retaining customization for payload integration. We intend to present the framework of the architecture and its first embodiment in a flat satellite prototype.

INTRODUCTION

CubeSats and nano-satellites have increasingly become a platform of choice for various proof-of-concepts demonstrations and large distributed constellation missions[1]. The development cycle for a simple CubeSat is generally expected to be one to two years long[2], but this tends to be much longer when advanced capabilities are implemented such as precise avionics, high data-rate links, etc … While private enterprises[3] have developed their own proprietary small-satellite busses, they generally do not make them commercially available. Government and educational programs are heavily distributed across many institutions and traditionally, each institution has worked on their own bus and components to support their very specific payload. This approach limits rapid payload and technology demonstration cycles as a significant effort has to be dedicated towards developing the overall bus capabilities. Bus capabilities generally encompass the following: structure and thermal management, power system and solar panels, radio, attitude control and determination system (ADCS), command and data handling system (C&DH) as well as any deployment mechanism related to any of those sub-systems. A general mechanical, electrical and software standard is expected to allow replaceability of each of those sub-systems from various sources based on the complexity needed for the mission under consideration. This allows teams to focus on the payload and the mission, as well as building a shared community heritage. The NRO, LLNL and NPS are developing a CubeSat bus reference architecture aimed at allowing interchangeability of components and reducing design cycles.

REFERENCE ARCHITECURE GOALS

Programatic goals

Previous work funded by the National Reconnaissance Office (NRO) and known as the Colony I[4] and Colony[5] II bus programs were reviewed to guide the CubeSat Next Generation Bus Reference Architecture effort (CNGB). The main high level goals of the program were summarized as follows:

- Ensure modular, non-proprietary solutions to support open competition: this goal is intended to promote detailed open documentation on mechanical, electrical and software interfaces that can be endorsed by industry. It also drives the need for open-source firmware and software.
- Enable mission flexibility at lower cost: this goal pushes the need for bus configuration flexibility with limited restrictions in component location, access to space, orbital regime, etc.
- Provide extensibility to larger form factor: while the intent of this effort is focused on a 3U form factor, this goal drives the architecture to not be locked to interfaces heavily dependent on the mechanical form factor.
- Drive the state-of –the art and meet demands of upcoming mission applications: this goal pushes the need to see ahead and support upcoming capabilities such as propulsion, fine pointing, high data rates, etc. In particular, it aims at adding enabling functionality covering both the technical aspects as well as the regulatory aspects (number and type of inhibits, safety concerns, radiation mitigations, etc.) of CubeSats.

Technical goals

Flowing from the programmatic goals as well as lessons learned from past CubeSat programs, several technical goals have been identified to be addressed by the CNGB architecture. In particular, the following requirements have been identified:

- Mechanical interface shall support slip-in of components and payload to limit dis-assembly during integration.
- Mechanical interface shall support mounting of components and payload at a wide range of locations. A 1cm granularity was deemed appropriate.
- Side panels shall support maximum access to space and limit multi-use (magneto-torquer, sensors, etc.)
- Staking of fasteners should be limited and alternate reversible locking should be used whenever possible to allow flexibility during integration.
- Data and power interfaces shall provide multiple payload connections in order to reduce the need for break-out interface sub-systems
- Data interfaces shall allow sub-systems to communicate directly with each other when permissible. (Payload to ADCS, etc.)
- Power interface shall provide maximum capability to payloads and sub-systems.
- Software interface shall provide self-documentation of the sub-systems capabilities for ease of integration.
- Architecture should support at least three independent inhibits for advanced capability (propulsion, high power transmitters, etc.)
- Architecture should support modes of operation able to support various failure modes (depleted battery at ejection, solar panel deployment failure, etc.)

Each of these requirements have flowed down to the design and incorporated in the CNGB architecture. The following sections will describe the current CNGB architecture.

MECHANICAL INTERFACE

Various trades were evaluated to establish the baseline structure and mechanical interfaces. A tray concept implementing the full 10cm x10cm was studied, but was assessed to add un-necessary complexity on any spacecraft deployable attachment as it requires the tray enclosures to be custom designed to accommodate hinges or requires predetermined heights. Additionally, it was found that it limits integration of payload and subsystems of non-standard shapes and may drive high fabrication and assembly tolerances to meet all dispenser requirements. Monolithic structures (4 walls) were also studied, and, while providing the maximum volume space, they were deemed to limit access to space and flexibility late in the integration process. The CNGB is therefore defining a mechanical interface based on a rail concept with a set of well-defined mounting locations. The mechanical interface also allows slip-in of sub-systems without the removal of the rails by implementing removable end-feet crossbar and side cross-bars.

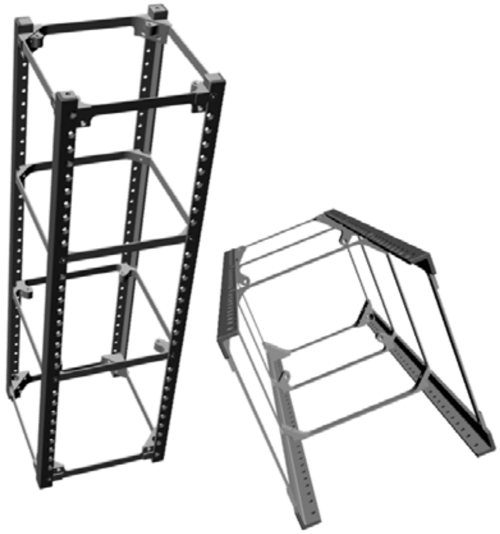

Figure 1: CNGB structure showed with end-feet crossbar removed on the right.

The structure was designed such that once assembled it is fully symmetrical except for the fastener location. This allows components to be rotated by 90 degrees without affecting the fit. Standard mounting hardware has been designed to fasten the various sub-systems to the rail.

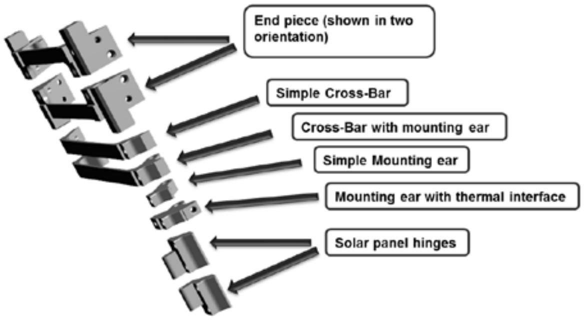
End piece (shown in two orientation)
Simple Cross-Bar
Cross-Bar with mounting ear
Simple Mounting ear
Mounting ear with thermal interface
Solar panel hinges

Figure 2: CNGB structure mounting hardware.

It should be noted that the crossbars can be placed anywhere along the rail to allow maximum flexibility or not used at all if a sub-system provides sufficient structural stiffness. Preliminary finite element analysis shows that adding a single simple sub-system in the rails can increase the natural frequency of the structure by 25%.

The overall CNGB mechanical interface has been summarized into a stayclear definition as shown below in Figure 3.

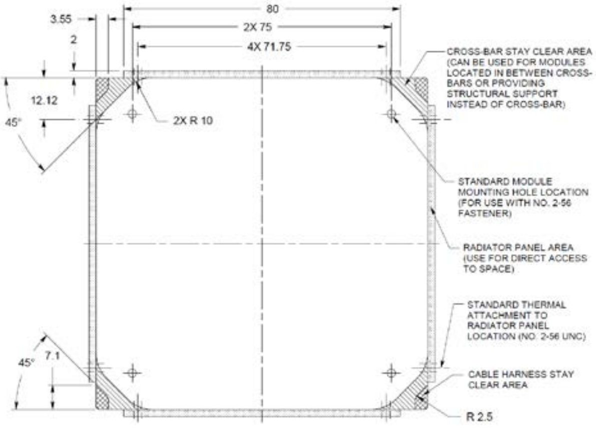
80
2X 75
4X 71.75
3.55
2
12.12
45°
2X R 10
CROSS-BAR STAY CLEAR AREA (CAN BE USED FOR MODULES LOCATED IN BETWEEN CROSS-BARS OR PROVIDING STRUCTURAL SUPPORT INSTEAD OF CROSS-BAR)
STANDARD MODULE MOUNTING HOLE LOCATION (FOR USE WITH NO. 2-56 FASTENER)
RADIATOR PANEL AREA (USE FOR DIRECT ACCESS TO SPACE)
STANDARD THERMAL ATTACHMENT TO RADIATOR PANEL LOCATION (NO. 2-56 UNC)
45°
7.1
CABLE HARNESS STAY CLEAR AREA
R 2.5

Figure 3: CNGB stayclear definition.

Additionally the mounting locations have been captured in an interface definition drawing including basic rail dimension. Mounting is done using #2(.086)-56 UNC x 0.3125 inch long x 82 degrees flathead cap screws, hex socket drive, A286 (iron base superalloy) stainless steel.

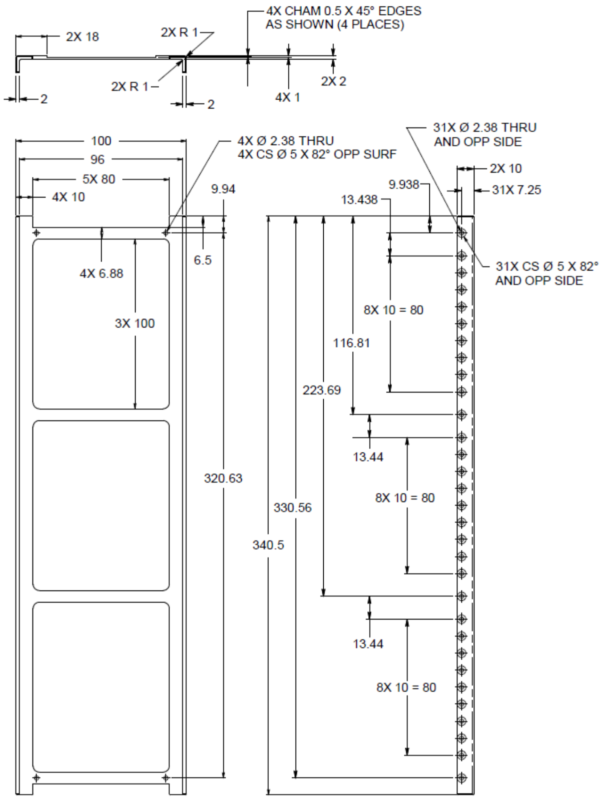

Figure 4: CNGB Interface Definition.

Fasteners are locked using thread inserts, nominally Emhart Helicoil #2(.086)-56 x 0.172 inch long, Nitronic 60 stainless steel. A trade-study was conducted regarding the choice of metric versus SAE fastener. SAE was down-selected as it provided more robust options for equivalent sizes. As an example a metric M2 fastener provides only 138kg of yield strength as opposed to 159kg yield strength for the equivalent SAE number 2 fastener.

ELECTRICAL INTERFACE

An extensive trade-study has been conducted to evaluate various leading data interfaces. Assessment factors included link rate, flow control, signal integrity, number of wires, number of slaves, number of masters, topology, heritage, etc. Various data interfaces spanning from PC104 to Bluetooth as well as traditional serial links were evaluated. The two leading contenders were CAN and RS485. The CAN standard is compliant with ISO 11898-1:2003 and ISO 11898-2:2003 and has been down-selected as the CNGB data standard as it offers the link layer definition by providing support for error checking, bus arbitration (multi-slave/multi-master support) and builds upon an extensive heritage in the harsh automotive industry.

In addition to the underlying CAN interface protocol, the data interface has been defined to require the following to comply with the CNGB architecture:

- Interface shall be self-powered per the CAN bus standard (5V) and require 50mW to 350mW or less. Power from the interface shall not be used to power any other functionality in the sub-system.

- Interface shall implement a standby mode when not in use. Nominal standby current shall be 200uA.

- Data Interface shall implement the following:

    - Power enable control for each power source from the power busses.

    - Telemetry temperature as well as voltage and current monitoring of the module from the power busses

    - In-system programming to any programmable devices on the sub-system.

    - Up to three real-time interrupts logic for hard, real-time requests. Interrupt lines shall be active low open collector to allow all modules to receive or send an interrupt. Pulse per second (PPS) interrupt type and wake-up interrupt shall conform to the pinout if used.

- Data Interface shall use a 9-pin dual row nano-D connector (male connector on the module and female on the cable harness). Unused connectors shall have a protective cap mounted using the mounting hardware.

Table 1: CNGB Data Interface

Pin Number	Pin Name
1	Interface Power (5V)
2	CAN-
3	Interface Ground
4	Interrupt 1 (reserved for PPS if present)
5	Interrupt 2 (reserved as wake-up signal for interface standby mode requests)
6	Interface Ground
7	CAN+
8	Interrupt 3
9	Interface Power (5V)

Nano-D connectors have been selected for this architecture as various vendors currently provide compatible connectors (Omnetics Connector Corp., Glenair Inc., Axon Cable SAS, etc.). In addition their robustness has been proven in the space industry on large and small programs alike.

Table 2: CNGB Power Interface

Pin Number	Pin Name
1, 2, 3, 4, 5, 6, 7, 8	Ground
9, 10, 11, 12, 13, 14, 15	12V regulated

A 12V regulated power distribution has been selected to accommodate high power distribution while still allowing a wide selection of electronics components. Nano-D connectors support gauge 30 wires, which are rated for 1A and will not cause self-fusing in vacuum, while providing a voltage drop of less than 1% at 12V for cable length relevant to 3U CubeSats. Each CNGB power interface is rated for 6A and uses a dual row 15-pin nano-D connector (male connector on the module and female on the cable harness).

The CNGB architecture defines two data busses and two power busses. Each module implementing a connection to a data bus or a power bus must provide two connectors per corner on opposite sides of the board to allow for daisy chaining. The locations of the connectors are defined per a standard PCB drawing.

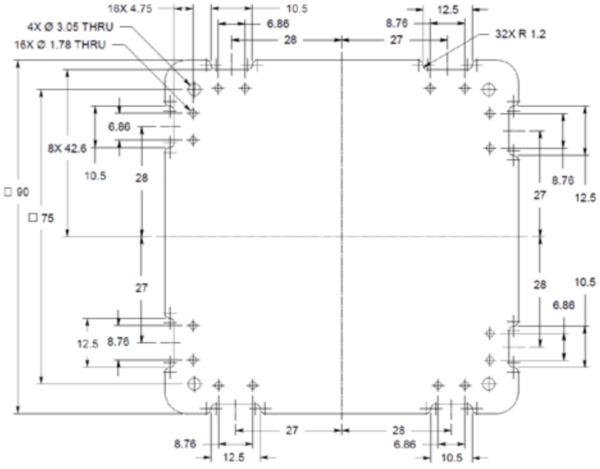

Figure 5: CNGB Standard PCB definition.

While complying with the standard PCB outline is not required, the location of the connectors in use must be followed to allow using the dedicated routing volume. It should be noted that, at the bare minimum, one data interface (Two dual row 9-pin nano-D connectors) and one power interface (Two dual-row 15-pin nano-D connectors) must be implemented.

A trade was done to assess the benefit of implementing a daisy chaining at the board level versus fabricating a custom multi-drop cable harness. The daisy chaining was selected as it allows a simple unique cable type, which lowers the cost and increases reliability. In addition, the presence of two connectors per interface on each board provides easy monitoring access to the overall interface during integration and test.

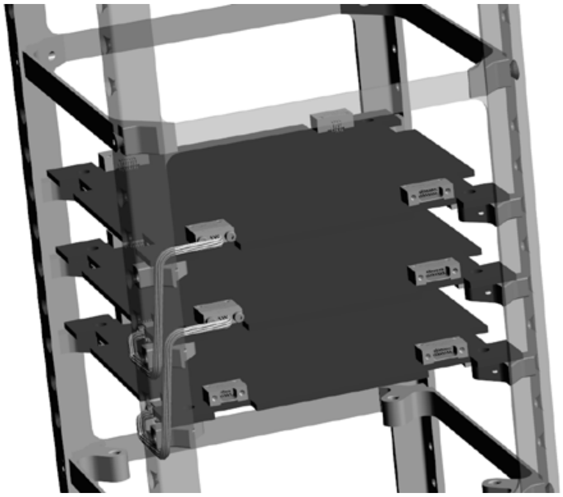

Figure 6: CNGB wire harness daisy chaining in the dedicated routing volume example.

SOFTWARE INTERFACE

The software architecture is expected to support hardware interchangeability, be compatible across a broad range of computing platforms (from small micro-controllers to multi-core large processors) and allow for an open source model. In addition, it is expected that computing resources in a CubeSat or small satellite will be distributed across various physical locations. A good match for addressing these constraints is the already existing Space Plug-and-play Avionics (SPA) architecture[6] developed as the foundation of the U.S. Department of Defense (DoD) Operationally Responsive Space (ORS) initiative. The Space Dynamics Laboratory (SDL) has developed a reference implementation[7] of the SPA Standards called the SPA Services Manager (SSM). The SPA architecture allows the creation of a distributed network and supports component discovery and registration as well as health and status reporting. The hardware interface is abstracted using the Applique Sensor Interface Module (ASIM), which can easily be implemented either in a micro-controller or in a full-fledged processor. For CNGB, ASIMs implement the CAN link layer and provide all the services required by the SPA network.

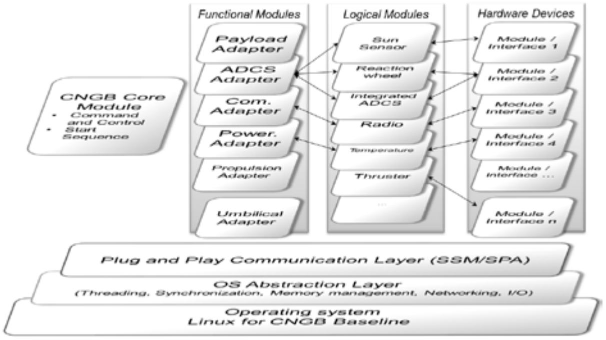

Figure 7: CNGB software architecture.

SPA provides plug-and-play capability by making use of the extensible Transducer Electronic Data Sheet (xTEDS), which is an extension of the IEEE 1451.4. Each device or module on the network must have an xTEDS (usually residing in the ASIM for physical devices) describing the capabilities and available commands to other components in the system. As shown in Figure 7, the CNGB architecture builds on the SPA network and defines various SPA component types:

- Functional Modules: these SPA modules describe high level capabilities expected to be found in a spacecraft. They are generally software only modules relying on Logical Modules for data and control. They are the modules providing high level mission commands to a user. It is expected that these modules do not know about the actual hardware available and execute high level control algorithms. Fully integrated capability such as a fully stand-alone ADCS sub-system can provide functional module capability.

- Logical Modules: these SPA modules describe logical capabilities in the network. They attempt to decouple the physical sensors, actuators or systems and provide the logical measurements or commands to Functional Modules. These modules can be software only modules or physical devices when well partitioned.
- Hardware Devices: these SPA modules are directly tied to a physical device. These devices must implement an ASIM to connect to the SPA network.

One should note that each SPA component on the network can provide one or more functionality types depending on the level of integration provided by the hardware.

One should also note that to comply with the electrical architecture, each module implementing a hardware device must provide an xTEDS providing at least the CNGB interface capability as follows.

```
<Interface name="CNGBDataInterface">
 <Variable name="Time" units="Seconds" format="FLOAT64"/>
 <Variable name="Voltage" units="Volts" format="FLOAT32"/>
 <Variable name="Current" units="Amperes" format="FLOAT32"/>
 <Variable name="Temperature" units="Celsius" format="FLOAT32"/>
 <Variable name="PowerFault" format="UINT08">
  <Drange name="ThresholdBatteryState">
   <Option name="No Fault" value="0"/>
   <Option name="Generic Fault" value="1"/>
   <Option name="Redundancy Fault" value="2"/>
   <Option name="Regulation Fault" value="3"/>
  </Drange>
 </Variable>
 <Variable name="PowerState" kind="PowerState" format="UINT08">
  <Drange name="ThresholdBatteryState">
   <Option name="On" value="0"/>
   <Option name="Off" value="1"/>
  </Drange>
 </Variable>
 <Notification>
  <DataMsg msgArrival="EVENT" name="CNGBDataInterface_Status"/>
   <VariableRef name="Time"/>
   <VariableRef name="PowerFault"/>
 </Notification>
 <Command>
  <CommandMsg name="PowerOn" description="turn on device"/>
 </Command>
 <Command>
  <CommandMsg name="PowerOff" description="turn off device"/>
 </Command>
 <Request>
  <CommandMsg name="CNGBDataInterface_GetTelemetry"\>
  <DataReplyMsg name="CNGBDataInterface_Telemetry">
   <VariableRef name="Time"/>
   <VariableRef name="Voltage"/>
   <VariableRef name="Current"/>
   <VariableRef name="Temperature"/>
   <VariableRef name="PowerFault"/>
   <VariableRef name="PowerState"/>
  </DataReplyMsg>
 </Request>
</Interface>
```

TESTING RESOURCES

Testing tools have been considered in the architecture design to facilitate integration and test by developers making use of the CNGB architecture. In particular, a simple set of commercial, off-the-shelf tools can be assembled to provide bench top testing of all CNGB compliant modules. This setup allows direct data connection from any laptop or desktop and consists of a DB9 to nano-D adapter cable, a USB-to-CAN adapter (USBmodul1 from Systec, PCAN-USB from PhyTools, etc.) and the Monarch Studio GUI provided by SDL. Note that for lower level direct CAN access, various commercial software packages are available free of charge and are compatible with most USB-to-CAN adapters (PCAN-View is the most popular). Also, some CAN controller integrated circuits such as the AT89C51CC03 from Atmel can be re-programmed directly through the CAN interface using this tool chain, allowing convenient in-system reprograming access to most modules.

The distributed capability of the SPA network also allows testing in simulated environments where some modules can reside physically outside the spacecraft. This allows distributed development teams to test interfaces with the rest of the system remotely via any network (current SSM supports Ethernet). Neighbor modules can also be simulated in software for test purposes and added to the spacecraft SPA network (through the umbilical network interface or on the test processor). This feature allows development of higher level C&DH capabilities without the need for actual hardware.

Finally the use of an open source Linux based operating system for full-fledged processors (used for C&DH and any other processing intensive module) allows software to be developed and tested on virtual machines with endless possibilities to interconnect through a SPA network. This allows software development to be conducted without the need to physically reprogram the non-volatile storage until the final version is ready.

SUMMARY

The CNGB architecture builds upon past CubeSat bus programs and provides the flexibility and transparency needed for community as well as industry endorsement. This program aims at providing the framework necessary to develop a base of interchangeable commercial products that can be selected to tailor a CubeSat or small satellite design for a specific application with a reduced need for custom development. This program also provides the foundation for a software base that can grow and be enriched as programs contribute to the available capabilities implemented. CubeSat development teams can choose to comply with the software architecture, electrical architecture, mechanical architecture or any combinations based on their needs. The CNGB architecture also provides flexible test tools for agile development, integration and testing.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-655896.

REFERENCES

1. A. Marinan, A. Nicholas, K. Cahoy, "Ad Hoc CubeSat Constellations: Secondary Launch Coverage and Distribution," accepted for publication, IEEE Aerospace Conference, 2013.

2. H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, R. Twiggs, “CubeSat: A new Generation of Picosatellite for Education and Industry Low-Cost Space Experi-mentation”, Proceedings of the Thirteenth Annual AIAA/USU Small Satellite Conference, Logan, UT, August 2000.

3. DECLAN BUTLER, "Many eyes on Earth: Swarms of small satellites set to deliver close to real-time imagery of swathes of the plan", Nature, Vol 505, 9 January 2014

4. Stephen Arnold, James Armstrong, Clark Person, Michael Tietz, "QbX - The CubeSat Experiment", 26th Annual AIAA/USU Conference on Small Satellites

5. Maj David “Dutch” Shultz, Capt Rebecca A. Unruh, David C. Williamson, Col John Anttonen, "Colony: A New Business Model for Research and Development", 24th Annual AIAA/USU conference on Small Satellites

6. American Institute of Aeronautics and Astronautics (AIAA) Standard, “Space Plug-and-Play Avionics (SPA) Satellite Data Model (SDM),” Draft Version 1.1.1_20080715 DLL, July 2008

7. Craig J. Kief and Brian K. Zufelt, "Trailblazer: Proof of Concept CubeSat Mission for SPA-1", AIAA Infotech, 29-31 March 2011, St. Louis, Missouri, USA