REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
10 March 2010	Final	09 March 2009 – 10 March 2010

4. TITLE AND SUBTITLE
A Modular Architecture for Responsive Configuration of Satellite Autonomy

5a. CONTRACT NUMBER
FA9453-09-M-0092

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
65502 F

6. AUTHOR(S)
Louis Marketos

5d. PROJECT NUMBER
3005

5e. TASK NUMBER
RV

5f. WORK UNIT NUMBER
EZ

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Design Net Engineering LLC
16080 Table Mountain Parkway
Suite 500, Golden, CO 80403

8. PERFORMING ORGANIZATION REPORT NUMBER
A Modular Architecture for Responsive Configuration of Satellite Autonomy Phase I Draft Final Scientific & Technical Report

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Space Vehicles Directorate
3550 Aberdeen Ave. SE
Kirtland AFB, NM 87117-5776

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RVSV

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RV-PS-TR-2010-1012

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution authorized to U.S. Government Agencies only, Proprietary Information, 10 March 2010. Other requests for this document shall be referred to the Air Force Research Laboratory /RVSV, 3550 Aberdeen Ave SE, Kirtland AFB, NM 87117-5776.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
A repeating topic in aerospace discussion forums is the [redacted] (b)(3):10 USC 130 to support our [redacted] (b)(3):10 USC 130 [redacted] (b)(3):10 USC 130 significant discussion about how to facilitate better situational awareness [redacted] (b)(3):10 USC 130 One strategy is to offload the ground by making satellites more capable at a platform level –allowing them [redacted] (b)(3):10 USC 130 [redacted] (b)(3):10 USC 130 satisfy their own needs and respond appropriately to space arena changes. If autonomy is to become robust and commonplace, the community needs a standardized approach [redacted] (b)(3):10 USC 130 developing an architecture specifically designed to accommodate the needs of spacecraft autonomous operations. The architecture will be naturally predisposed to scalability, rapid configuration, and reusability – all qualities that are necessary features of a framework that will be significantly built upon in the years to come to address the maintenance and protection of our military, science and commercial space infrastructure.

15. SUBJECT TERMS
[redacted] (b)(3):10 USC 130

16. SECURITY CLASSIFICATION OF:			**17. LIM. OF ABSTR.**	**18. NO. OF PGS.**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR		[redacted] (b)(6)
UN-CLASSIFIED	UN-CLASSIFIED	UN-CLASSIFIED			**19b. TELEPHONE NUMBER** *(include area code)* [redacted] (b)(6)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank

Table of Contents

* SECTIONS OF THIS REPORT MARKED WITH AN ASTERISK IN THE TABLE OF CONTENTS AND IN SECTION HEADERS, PRESENT MATERIAL THAT IS DESIGN_NET ENGINEERING PROPRIETARY, AND DISTRIBUTION IS LIMITED UNDER DFARS *252.227-7018.*

(b)(3):10 USC 130

1. SUMMARY

This document is the final report for the Phase I Small Business Innovative Research (SBIR) project effort entitled "A Modular Architecture for Responsive Configuration of Satellite Autonomy" under the contract number FA9453-09-M-0092. The Air Force Research Laboratory (AFRL) is the SBIR sponsor. This report provides an introduction to the project, reports on the task results, and provides some discussion the implications of the research on a go-forward basis.

2. INTRODUCTION

2.1. Background

Design_Net Engineering, Inc (DNet) has gained unique experience in the Responsive Space Testbed (RST) in the area of developing standards that facilitate the rapid design, configuration, assembly, and testing of tactical satellites. The Spacecraft Plug&play Avionics (SPA) standards were initiated by AFRL to specifically address the end-to-end process bottlenecks that stand as barriers to operationally responsive space. (b)(3):10 USC 130 and (b)(3):10 USC 130 have contributed significantly the SPA development effort. Each of our companies has helped steer these standards through their evolution.

(b)(3):10 USC 130 on this contract includes a (b)(3):10 USC 130 that facilitates (b)(3):10 USC 130 interface definitions that allow components to cleanly join a system without customization of code, and the beginnings of a standard to allow the satellite's mission-specific operation to be governed by modular "agents" that can be easily "dropped-in" to the system to compose desired capability.

DNet, through other SBIR vehicles, already developed the infrastructure and code constructs to configure tactically-relevant satellite capability from these modular elements, as well as robustly test the virtual satellite using environment and threat models in a component-oriented simulation framework.

(b)(3):10 USC 130 (b)(3):10 USC 130 from the SPA initiative, leveraging from existing work to take the next step in formalizing a truly robust architecture for spacecraft autonomy. In doing so, the stage will be set for dedicated efforts to develop a first generation autonomy complement for a candidate mission that fully demonstrates the ability to configure a collection of modular autonomy code elements that are platform agnostic and reusable by virtue of their service-oriented design. In the short term, this provides DNet (b)(3):10 USC 130 to contribute to this important challenge by drawing from our significant experience bases in this arena. In the longer timeframe, both companies seek to be preferred vendors of modular software elements (b)(3):10 USC 130 (b)(3):10 USC 130 that are compatible with the architecture that we initially develop. It is our feeling that the capabilities described in the solicitation, in addition to being in alignment with the objectives of the AFRL and the ORS program office, are the future of most space operations. In performing this work we hope to be well positioned to service the needs of the larger community at a time in the not-so-distant future when these strategies are commonplace.

2.2. Autonomous Mission Management Architecture

2.2.1. Introduction.

In order to respond to today's more dynamic mission [redacted] (b)(3):10 USC 130 satellites must be able to be placed into service rapidly and operated more effectively. [redacted] (b)(3):10 USC 130

[redacted] (b)(3):10 USC 130

Most of today's satellites also have virtually no ability to process Intelligence, Surveillance, and Reconnaissance (ISR) sensor data on-board and plan courses of action based on "trigger" conditions. In general ISR data is down linked, processed, and analyzed. Courses of action are then planned on the ground and uploaded to the satellites. Availability of ground contacts further delays new actions. The long lead time may result in missed opportunities to react to evolving scenarios and higher risk of mission failure. This approach also requires higher communication bandwidth since all data must be sent to the ground rather than just the relevant information being sent to ground.

To mitigate these limitations future satellites need the ability to:

- Process sensed data
- Autonomously detect tasking trigger conditions on-board
- Autonomously detect anomalies and threats on-board
- Accept tasking requests from tactical users
- Plan courses of action
- Autonomously execute planned procedures
- Operate without ground reliance

While some of these issues have been addressed with point solutions, this paper describes a model driven, loosely coupled, open architecture and an initial implementation that meets the requirements for rapid call-up and satellite autonomy.

2.2.2. Requirements

2.2.2.1. Focused Long Term Technical Challenges (FLTC) Objectives

[redacted] (b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130 [redacted]

2.2.3. Architecture

(b)(3):10 USC 130 [redacted]

The architecture must provide an initial implementation capable of meeting the current autonomy objectives and requirements while remaining open to accommodate new and evolving requirements and technologies.

Flexibility refers to the architecture's ability to adapt to different situations. The architecture must be able to meet a range of autonomy requirements for future missions. The architecture must also be able to interface with multiple satellite buses and flight software systems without being dependent on them. For example, the architecture must be capable of interacting with a Spacecraft Plug And Play Avionics (SPA) flight software system, (b)(3):10 USC 130 (b)(3):10 USC 130 [redacted]

Extensibility refers to the architecture's ability to be extended to include new components. The architecture must be able to accommodate new mission or payload unique components and must also accommodate new technologies as they become available.

In order to address the characteristics of being open, flexible, and extensible (b)(3):10 USC 130 (b)(3):10 USC 130 (b)(3):10 USC 130 [redacted] Vendor lock is eliminated and technology insertion is easily accomplished with little cost and schedule impact. Flexibility is also addressed by the use of a model driven architecture with declarative autonomy flight software components. The models express all the satellite and mission unique processing and the code of the declarative components does not change.

(b)(3):10 USC 130 (b)(3):10 USC 130 [redacted] No software is guaranteed to be able to run on all processors and all operating systems current and future without revision, (b)(3):10 USC 130 (b)(3):10 USC 130 [redacted] should at least be able to run on the most common processors and operating systems currently used for satellite systems.

(b)(3):10 USC 130 [redacted] functional model, component model, and initial implementation are described in following sub-sections.

2.2.3.1. Functional Model

The functional model of the autonomy architecture exhibits characteristics of deliberative artificial intelligence and robotics systems that combine sense-plan-execute and reactive approaches. Systems using the classical sense-plan-execute cycle were introduced in the early 1980's. They were followed by faster more robust reactive systems designed to meet response time requirements that could not be accomplished using the processors available at that time. Given the faster flight processors available today, the AMM architecture provides the ability to formulate complex plans while retaining the capability to react quickly to exogenous events. The AMM architecture also augments the sense-plan-execute cycle with the addition of assessment functionality.

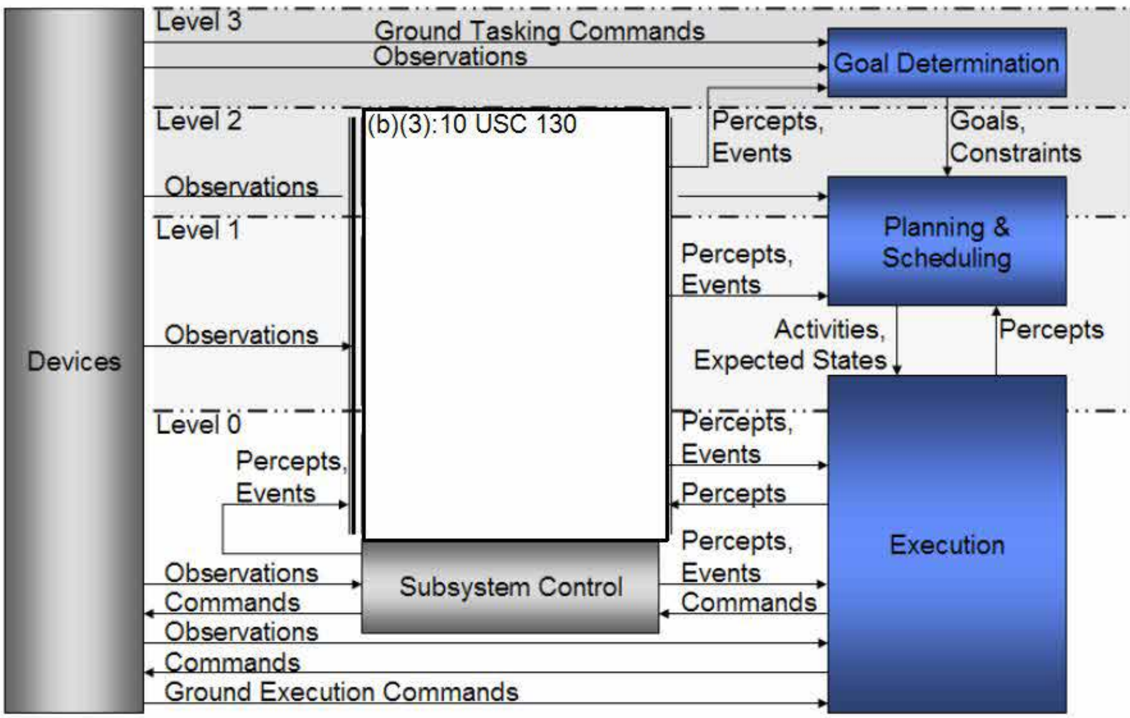

Figure 1 (b)(3):10 USC 130 **Model**

In the functional diagram (**Error! Reference source not found.**), sensing is provided by the Devices on the left. The Devices include the hardware and any device control software directly associated with the hardware. Observations flow from the Devices to Subsystem Control, Assessment, deliberation (Goal Determination and Planning), and Execution functions. The lower level closed loop control is provided by device control software and Subsystem Control.

Assessment functionality includes multiple levels of data fusion that can be applied as necessary to achieve the degree of autonomy required to perform the mission. Five levels of assessment and response processing have been defined[1] based upon the Data Fusion & Resource

[1] Andrew Gelfand, Mike Colony, Chris Smith, Chris Bowman, Richard Pei, Thien Huynh, & Clinton Brown, "A Distributed, Decentralized Architecture for Low Level Fusion and Track Adjudication of Army ISR Assets", NSSDF Conference, JHAPL, 2007.

Management (DF&RM) Dual Node Network (DNN) technical architecture. The DF&RM DNN architecture has been applied by Data Fusion & Neural Networks (DF&NN) (b)(3):10 USC 130 (b)(3):10 USC 130

(b)(3):10 USC 130 to flight systems by including the first 3 assessment levels (b)(3):10 USC 130 and event relationships) in the functional model. These assessment levels are depicted as Feature (b)(3):10 USC 130 the functional model with example processing for each level. The DF&RM DNN levels are also shown as a layered background in the diagram. The upper 2 assessment levels (b)(3):10 USC 130 (b)(3):10 USC 130 are not included in the flight autonomy architecture at this time but could be added in the future to meet increased autonomy requirements as flight processor performance and data storage improves. Planned improvement is easily integrated because of the loosely coupled publish/subscribe metaphor.

Assignment of the various assessment functions to a level is meant to provide a performance versus cost framework with partitions according to canonical decomposition & solution techniques. The intent is that assessment at a given level relies on observations and information from the same or lower levels. The assessment does not rely on information from a higher level. The information from the assessment functions flows to the Goal Determination, Planning, and Execution functions. This information is labeled as Percepts in the diagram to distinguish it from the Observations coming directly from the Devices. The name is intended to convey the meaning that the assessment information is a perception of the system as opposed to a direct observation (or measurement) of the system. This subtle distinction does not typically affect the processing of the information other than some data fusion processes may weigh the information differently.

The assignment of the responses to a level follows a similar rationale. A response at a higher level is further decomposed by lower levels until primitive actions are determined. Goal Determination and Planning provide the deliberative response. Goal Determination receives Observations, Percepts, and Ground Tasking Commands to determine the goals to be submitted to Planning given the current state of the system. Goal Determination may also receive tasking requests from other sources on-board such as an assessment process or a subsystem controller. Planning then formulates a sequence of activities that will attain the specified goals given the current state of the system.

Finally, Execution provides the reactive (or reflexive) response. Execution further decomposes the sequence of activities given by the plan into a set of atomic actions (i.e. commands) that will be performed by Subsystem Controllers and Devices. Execution also monitors the Observations, Percepts, and Events to check preconditions, verify command execution, check end conditions, and maintain its system and external state knowledge. The perceived state is used to trigger immediate event response activities or execute command failure contingencies to respond to exogenous events and system failures in a timely manner.

This functional model is not intended to be a single point solution. Rather functionality represented in this model can be chosen to meet different autonomy requirements ranging from low-level assessment with reflex response to complex higher-level assessment with highly deliberative response. This model is also not intended to imply any specific functional allocation to software components – i.e. a functional block is not necessarily implemented as a single component. The component model will be presented in the following section.

2.2.3.2. (b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130 Standardized messages allow different components to be selected to compose a specific implementation. For example, the Component Model includes the typical components for planning and execution used in most deliberative autonomous systems. However, rather than the usual tight coupling between the planner and the executive, (b)(3):10 USC 130 (b)(3):10 USC 130 provides the flexibility to use different planners and executives without modification to meet specific program needs (b)(3):10 USC 130 (b)(3):10 USC 130

(b)(3):10 USC 130 An agent is commonly defined as any component that perceives its environment through sensors and acts upon that environment through actuators. Although in this architecture the Executive Agent does not directly interface with sensors or actuators, it perceives its environment through the observations and percepts mentioned in the Functional Model section (b)(3):10 USC 130 (b)(3):10 USC 130

(b)(3):10 USC 130 planning and scheduling, and data processing functions (b)(3):10 USC 130 The components are referred to as services since they perform essential processing which may involve observations or percepts but do not act upon the environment.

(b)(3):10 USC 130

2.2.3.3. Initial Implementation

The overall approach (b)(3):10 USC 130 is to exploit proven autonomy components with flight heritage (b)(3):10 USC 130 to increase the Technology Readiness Level (TRL) and reduce risk. (b)(3):10 USC 130 (b)(3):10 USC 130 (b)(3):10 USC 130 The initial implementation also provides components for planning and scheduling but allows flexibility to substitute other planners and add assessment and data processing services.

(b)(3):10 USC 130

(b)(3):10 USC 130

The autonomy components must interoperate with the Flight Software Subsystem (FSS) components. The FSS components typically include Subsystem Controllers (e.g. ADCS, GNC, Communications, and Power) and Device control or interfacing software. FSS components can by individually integrated or interfaced to the SWBus or a FSS Bridge component can be used to translate between the SWBus and the FSS infrastructure. The bridging approach has been used successfully to add autonomy capabilities to an existing FSS.

(b)(3):10 USC 130 architecture implementation infrastructure and autonomy components will be described in more detail in the following sections. The allocation of the FLTC 7.4 objectives to the initial implementation and the Model Driven Approach (MDA) described in section **Error! Reference source not found.** is provided in **Error! Reference source not found.**.

Table 2 FLTC 7.4 Objectives Allocation

Number	Description	Primary Allocations	Secondary Allocations
7.4.1	**Rapidly Checkout Spacecraft and Autonomous Operations**		
7.4.1.1	On-Orbit Checkout		
	Place satellite in operations quickly after launch vehicle separation		
7.4.1.1.1	Deployment and on-orbit autonomous checkout of space vehicle bus	SCL	FSS Subsystem and Bus Device Controllers
7.4.1.1.2	Deployment and on-orbit autonomous checkout of payloads	SCL	Payloads and FSS Payload Controllers
7.4.1.1.3	On-orbit autonomous sensor calibration	Payloads and Data Processing Services	SCL
	Minimize time from call-up to launch readiness		
7.4.1.1.4	Automated discovery of commands and telemetry points	MDA	FSS Bus and Payload Components
7.4.1.1.5	Automated discovery of device behavior	MDA	FSS Bus and Payload Components
7.4.1.1.6	Development of knowledge base for known conditions	MDA	
7.4.1.2	On-Board Planning and Reconfiguration		
	Autonomously plan activities to remedy threats or anomalies; respond to events; or service components	ActivityPlanner or Alternate Planners	
	Provide multi-mission capable satellites through software reconfiguration	ActivityPlanner or Alternate Planners	SCL
7.4.1.3	Autonomous Mission Management		
	Detect threats and anomalous conditions	SCL and Assessment Services	
	Enable goal-based operations	ActivityPlanner or Alternate Planners	SCL
	Robust on-orbit processing of sensor data with autonomous re-queuing of satellite	Data Processing Services	ActivityPlanner or Alternate Planners

2.2.3.4. SCL Software Bus (SWBus) Infrastructure

The SCL SWBus provides the initial infrastructure implementation, but other implementations based on other standards such as (b)(3):10 USC 130 (b)(3):10 USC 130 [redacted]

The SWBus provides a high level message oriented facility for interprocess communication among SCL and other components. Components interface with the SWBus through an SCL API that provides a set of classes that are dynamically linked with the component. The API provides methods for creating and initializing a SWBus handler, connecting to the SWBus, registering for messages, sending messages, and receiving messages. The complete SWBus API is defined in the SCL Application Programming Interface Guide distributed with the SCL product.

The SCL SWBus is built on a C++ framework that allows dynamic instantiation of an underlying technology. (b)(3):10 USC 130 (b)(3):10 USC 130 [redacted] The following implementations are currently supported: sockets (Windows, Linux, and Solaris), DCE (Windows), ToolTalk (Solaris), and native message queues (VxWorks). Note that the DCE implementation is being phased out on Windows and the preferred implementation for that platform is the socket implementation.

The SWBus supports multicast and point-to-point communication. Multicast messaging follows the typical publish and subscribe pattern providing asynchronous unidirectional messaging to multiple subscribers. Using a *notify/listen* message protocol, copies of a particular notify message are only sent to components that have previously registered a listener to receive that specific message. The component sending the notify message does not need to have any knowledge about the number or identity of the components listening for the message. This protocol is shown in **Error! Reference source not found.**. Although this discussion has focused on sending and receiving messages from components, Notify messages can also be sent from SCL Scripts, Rules, Functions, and Constraints.

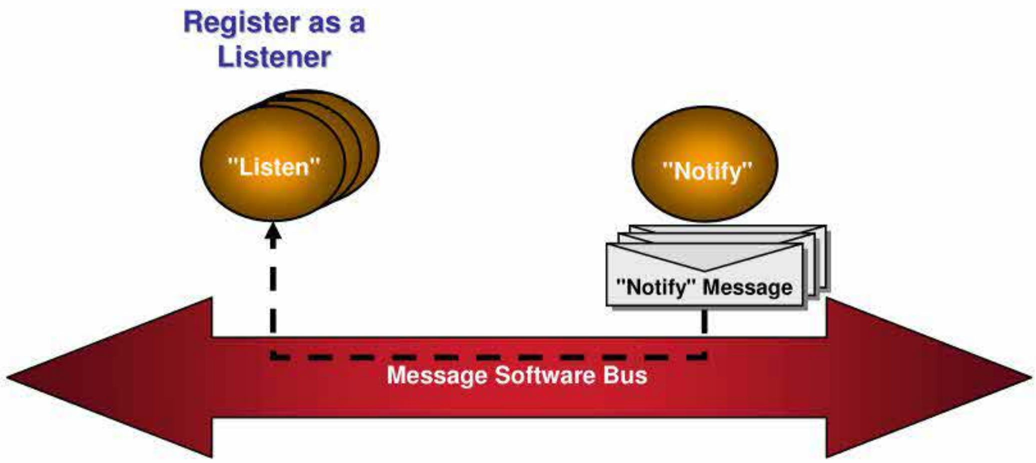

Figure 4 Notify/Listen Message Protocol

Point-to-point messaging follows a request/reply pattern providing synchronous directional messaging between two components. Using a *request/reply handler* protocol a request message is sent to the component that has previously registered a handler to receive and process that specific message. As with the *notify/listen* protocol, the component sending the request message does not need to have any knowledge about the identity of the component handling the message. The *request/reply handler* protocol is shown in **Error! Reference source not found.**.

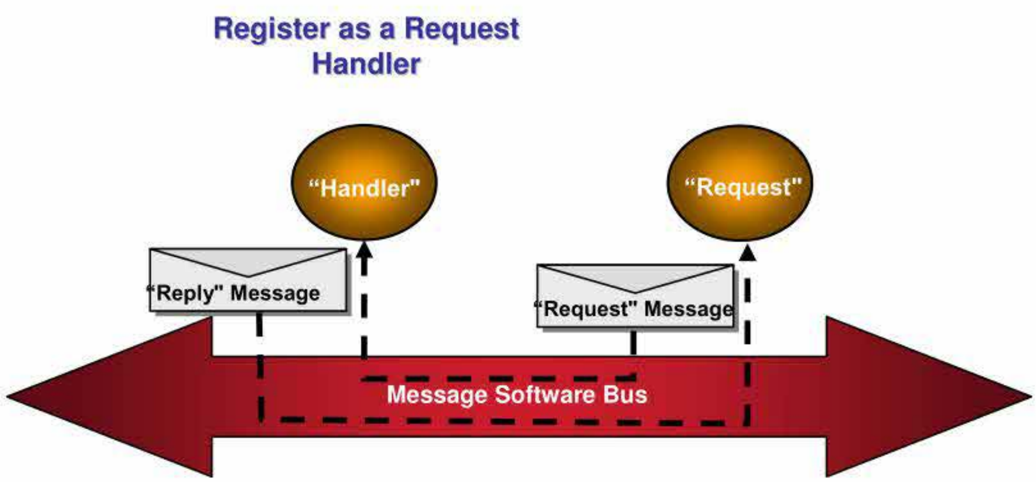

Figure 5 Request / Reply Message Protocol

The structure of a message allows the user to define message formats provides a flexible representation of data, while taking care of much of the low level data formatting operations such as byte ordering. Each message is composed of *fields* of a given *type*. The field types are *int, float, double, string,* and *mem* (where *mem* is an unsigned char* which allows the user to insert data of any format into a message – e.g. raw binary data). Each of the fields in the message has a name, so that message field data may be randomly accessed by name. Methods are also provided for inspecting the message format.

Although it is not required for using the SWBus or for accessing message fields, it is typically convenient to define the messages using the Spacecraft Markup Language (SML). SML was introduced in 1999 and is a defined set of Extensible Markup Language (XML) elements and attributes providing the Space Community with a standard means of defining SWBus messages, spacecraft command and telemetry packets, and other support data objects.

2.2.3.5. SCL Executive

SCL includes a set of components that provide a solid foundation and framework for flight autonomy and a fifth generation language used to define procedural logic (scripts and functions) and event driven logic (rules and constraints). With the components and features described below, SCL is able to perform the following autonomy functions:

- Assessment Level 0 (b)(3):10 USC 130
- Assessment Level 1 (b)(3):10 USC 130
  (b)(3):10 USC 130
- Execution
- Goal Determination

SCL is best characterized as an intelligent, model-based, goal oriented agent. An intelligent agent is an agent that, for each possible observation and percept sequence, selects the appropriate action given the evidence provided by the sequence and its built-in knowledge – i.e. its model of the system.

In order to act intelligently, a model of the system is required to provide a way for the agent to maintain state information based on the observation and percept history and thereby compensate for at least some unobservable aspects of the system. The model also includes information about how the system responds to agent actions as well as how the system evolves independently of the agent. SCL is a declarative application, i.e. the model defines all the satellite and mission unique processing and the code does not change. (b)(3):10 USC 130 (b)(3):10 USC 130 (b)(3):10 USC 130 The SCL database is often generated from some other representation of the commands and telemetry – e.g. a relational database or other xml format such as XTCE or xTEDS. (b)(3):10 USC 130 (b)(3):10 USC 130

Goals are typically presented to SCL in the form of operational procedures – i.e. scripts – to be executed at a specified time in the future. The goals are determined by additional on-board assessment software (e.g. target recognition software) or are received from ground sources such as spacecraft operators or tactical users. Goals may also be determined by SCL using event-driven rules. The goals are sent to the planner if available and dispatched to SCL at the appropriate time for execution. If a planner is not available, or if the required response does not allow time for deliberation, goals may be sent directly to SCL.

Error! Reference source not found. shows the basic SCL flight components including:

- Real-Time Engine (RTE): Inference engine, script scheduler, SCL interpreter, and Real-Time Change Order (RTCO) propagation/generation
- SCL Database: All observations, percepts, command parameters, and packet definitions
- DataIO: Data acquisition, reduction, and RTCO generation to the RTE
-

It also shows the following optional flight components:

- Pktgen: Generates command and telemetry packets from database
- RTCO Server: Provides capability to subscribe to RTCOs for specific SCL DB items
-

In addition to the flight components SCL provides a workstation development environment consisting of:

- Compatible ground install of SCL
- DB Compiler: Compiles SML into binary SCL real-time database
- DB Loader: Prepares binary database load file for flight and loads binary database into memory
- Cmdgen: Formats command definitions to support embedding command packets into script
- SCL Compiler: Compiles SCL language into (b)(3):10 USC 130
- SCL Recorder: Prepares the binary project load file for upload to flight
- Tools and Utilities: SCLViewer, DB Test, SCL Environment Setup
-

SCL also offers a complete operational ground system solution (b)(3):10 USC 130 (b)(3):10 USC 130 With this complete development cycle, multi-platform support, SCL facilitates the migration of database definitions, scripts, and rules from development, assembly, integration, and Test (AI&T) (b)(3):10 USC 130 (b)(3):10 USC 130

Some additional information on the SCL database and language is provided in sub-sections below. The SCL database, language, tools, and utilities are completely described in the SCL Users Guide distributed with the SCL product.

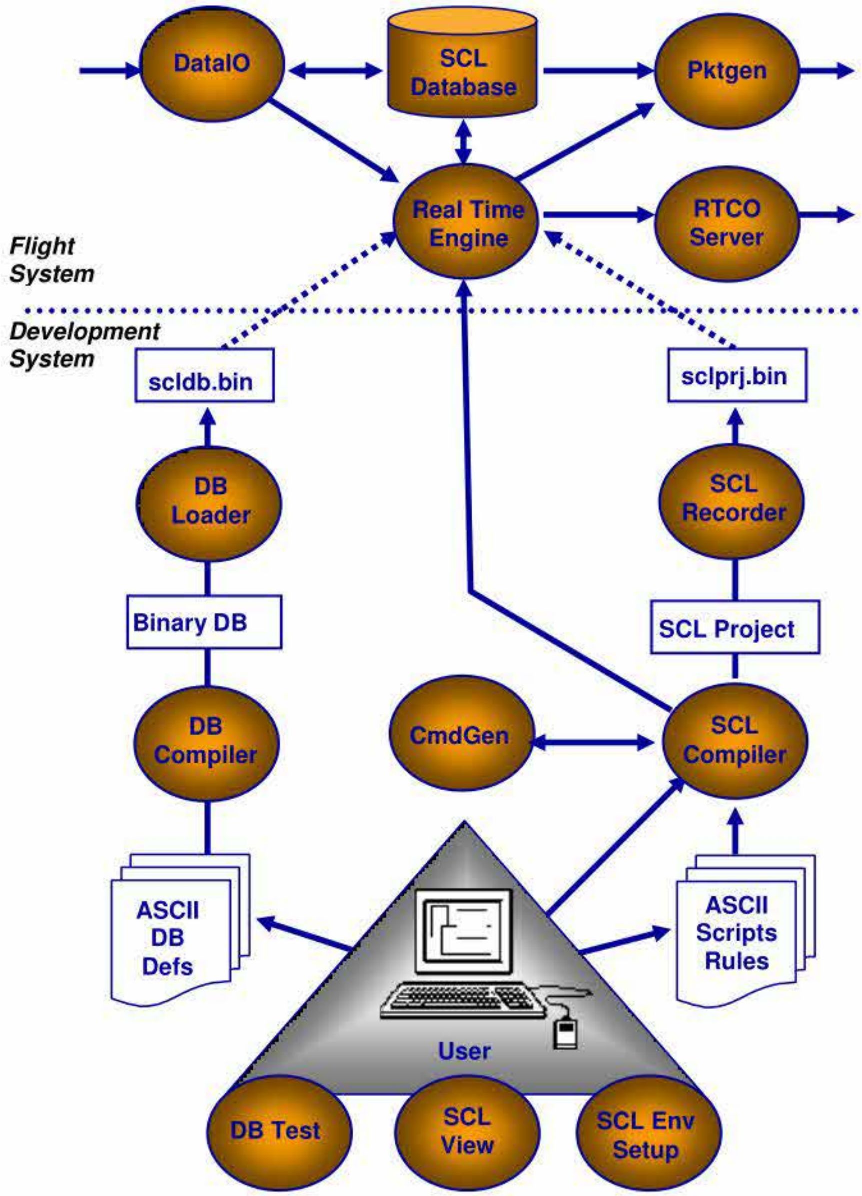

Figure 6 SCL Flight and Development System Components

2.2.3.5.1. SCL Database

The SCL object-oriented real-time database is a useful data fusion platform able to process more than 250,000 measurands from disparate data sources. In addition to being the central repository for integrated script, rule, constraint, and RTCO processing, the database provides the following Assessment Level 0 Observation Processing functionality:

- Engineering Value Conversion
- Smoothing
- Change Sensitivity
- Delta Limit Checking

- Rail Limit Checking
- Stale Value Determination
-

The SCL DB provides three basic record classes (Sensor Data Items, Command Actuator Data Items, and Derived Data Items) two basic sub-classes for each of the three basic classes (Discrete and Analog), and other specializations to provide a rich set of record classes required to represent satellite data. The record type structure is shown in **Error! Reference source not found.** with a list describing the classes in **Error! Reference source not found.**.

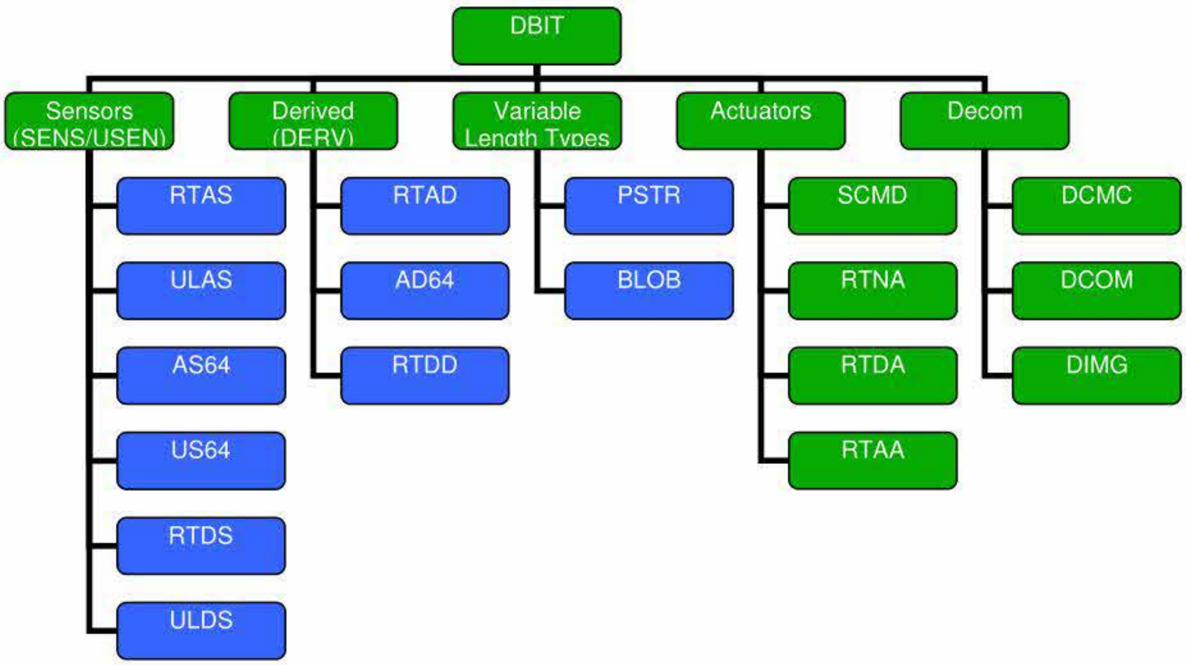

Figure 7 SCL Database Record Class Diagram

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

[redacted]

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

[redacted]

2.2.5. Related Work

2.2.5.1. Related SCL Projects

SCL is a COTS product with more than 20 years of on-orbit operation. SCL has recently been used on-board EO-1, TacSat-2, and TacSat-3. SCL will also be used on-board TacSat-4 and the NASA Orion Crew Exploration Vehicle.

(b)(3):10 USC 130

[redacted]

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130 [redacted]

2.2.5.2. Related (b)(3):10 USC 130 [redacted] Projects

(b)(3):10 USC 130 [redacted] variety of missions, mission-oriented research, and demonstrations. (b)(3):10 USC 130 [redacted]

(b)(3):10 USC 130 [redacted]

(b)(3):10 USC 130 [redacted]

(b)(3):10 USC 130

2.2.6. Summary of Autonomous Mission Management Architecture

(b)(3):10 USC 130 generalized architecture that allows the selection and integration of flight autonomy components.

(b)(3):10 USC 130

(b)(3):10 USC 130 SCL provides the Publish and Subscribe messaging infrastructure and an Intelligent Executive Agent. SCL alone provides a viable model-based reflex agent solution that meets run time autonomy requirements except for planning and autonomous on-orbit sensor calibration which requires sensor unique data processing. The ActivityPlanner and OrbitPropagator address the planning requirements.

(b)(3):10 USC 130 the initial implementation is not a point solution, but is an open architecture facilitating the use of alternate planners and the addition of assessment and data processing components as needed to meet mission requirements.

The use of a model driven approach and declarative components addresses the rapid call-up requirements. The applicability of any declarative component to a problem domain is largely dependent upon the richness of the modeling language. The SCL language has been proven to be applicable to space domain through its use on multiple satellite programs. (b)(3):10 USC 130

(b)(3):10 USC 130

With the AMM architecture and the initial implementation, the main technical challenges are associated with model creation and representation, robust on-board planning, and data processing required for calibration.

2.3. Objectives of the Phase I Effort - *Task Definitions*

The predominant theme of this SBIR is the maturation of several existing capabilities into a robust, reusable architecture for responsive spacecraft autonomy. Recognizing that much of the effort in realizing this goal involves correctly composing the underlying architecture and tools, the Phase I contract will be used to take a step back, consider the long-range needs, and specify how the existing design elements will be adapted. (b)(3):10 USC 130 already satisfy the expressed desire to demonstrate an architecture which performs autonomous onboard flight operational decision logic – that capability has been demonstrated to a limited extent (b)(3):10 USC 130 In this Phase I we proposed to assess the ability of the existing capabilities to meet the long-term needs. Some of the findings are already known and have been previously cited in section 2.1. Others are the focus of the tasks below. The end result of the Phase I activity will be a detailed specification of the work required to arrive at a highly capable implementation running on a flight-like processor against high fidelity simulation (b)(3):10 USC 130

2.3.1. Task 1 – Project Management

The project management WBS includes technical oversight throughout the period of performance. It also includes financial and status tracking as well as regularly quarterly reporting and generation of the final report and required SBIR documentation responsibilities.

2.3.2. Task 2 – Define Use Cases and Requirements for Architecture

The process will initially involve defining use cases with the technical representatives at AFRL. Since correct and accurate response of the autonomous system is one of the key objectives of the research, the scenarios and appropriate responses will be thoroughly discussed and documented prior to proceeding with further design work. (b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130 requirements for the performance of the system will be captured. These will largely be top-level requirements expressing the characteristics desired to support the goals of configurability, reuse, and multi-platform compatibility. The first three weeks of the contract will be used to develop and capture these use cases and requirements in documentation.

2.3.3. Task 3 - Develop Interface Standards for Components and Subsystems

One of the known needs for a complete and capable architecture is the refinement of the metadata descriptions of components and subsystem functionality in the space system to facilitate the configuration of autonomy for the spacecraft. Though much of the relational information required to properly specify the complex interactions that occur during the automated decision process will require the assistance of user-generated constraints and rule-based logic, much of the supporting information and system state data should be readily available from standard interfaces exposed by hardware and software components in the system. This is an area where some work has been devoted in the xTEDS development of the PnPSat program, (b)(3):10 USC 130

(b)(3):10 USC 130

Developing an architecture in which the autonomous function of the spacecraft has sufficient access to accurate knowledge of satellite state is essential to correct action. (b)(3):10 USC 130 be devoted to the completion of this task, to be initiated once the user cases and requirements of Task 1 have been captured.

2.3.4. Task 4 - Develop a Design for Subsystem Planning Interfaces

The existing implementation of response planning, a remnant of development for the PnPSat program, is highly distributed. Each subsystem controller application module currently provides a resource planning interface (b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

(b)(3):10 USC 130

5. LIST OF ACRONYMS

Table 4 List of Acronyms

AFRL	Air Force Research Laboratories
ATE	Autonomous Tasking Executive
COTR	Contracting Officer's Technical Representative
CSOW	Contractor Statement of Work
DNet	Design Net Engineering, LLC
PnPSat	Plug and Play Satellite
RST	Responsive Space Testbed
SBIR	Small Business Innovative Research
SCL	Systems Control Language
SDM	Satellite Data Model
SPA	Spacecraft Plug&play Avionics

6. DISTRIBUTION LIST

AFRL/RVSV
Attn: (b)(6) [redacted]
3550 Aberdeen Ave. SE, Bldg 472
Kirtland AFB, NM 87117-5776
Marked For: Contract FA9453-09-M-0092

AFRL/RVOP
Attn: STINFO Officer
3550 Aberdeen Ave. SE, Bldg 497
Kirtland AFB, NM 87117-5776
Marked For: Contract FA9453-09-M-0092

NOTICE AND SIGNATURE PAGE

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RV-PS-TR-2010-1012 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

(b)(6)

(b)(6)

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

AFRL-RV-PS-
TR-2010-1012

AFRL-RV-PS-
TR-2010-1012

A Modular Architecture for Responsive Configuration of Satellite Autonomy

Louis Marketos

Design Net Engineering LLC
16080 Table Mountain Parkway, Suite 500
Golden, CO 80403

4 May 2010

11/8/13

Final Report

* Distribution Statement Changed per FOIA# 2013-05797-F, dated 07 Nov 13 by AFRL/RVSVC, Paul Zetocha.

Distribution authorized to ~~U.S. Government agencies only; Proprietary Information: DFARS SBIR Data Rights; 4 May 2010.~~ Other requests for this document shall be referred to AFRL/RVSV, 3550 Aberdeen Avenue SE, Kirtland AFB, NM 87117-5776. (Distribution Statement A: Approved for public release; distribution is unlimited.)

See inside notice and signature page for additional restrictions.

WARNING - This document contains technical data whose export is restricted by the Arms Export Control Act (Title 22, U.S.C., Sec. 2751 et seq.) or The Export Administration Act of 1979, as amended, Title 50, U.S.C. App 2401 et seq. Violations of these export laws are subject to severe criminal penalties. Disseminate in accordance with the provisions of DoDD 5230.25.

DESTRUCTION NOTICE - For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND