

# Data Abstraction Architecture for Monitoring and Control of Lunar Habitats

Scott Bell and David Kortenkamp

TRAC Labs Inc

Jack Zaiantz

Soar Technologies Inc

Copyright © 2009 SAE International

## ABSTRACT

A Lunar habitat will be highly sensed and generate large amounts of data or telemetry. For this data to be useful to humans monitoring these systems and to automated algorithms controlling these systems it will need to be converted into more abstract data. This abstracted data will reflect the trends, states and characteristics of the systems and their environments. Currently this data abstraction process is manual and *ad hoc*. We are developing a Data Abstraction Architecture (DAA) that allows engineers to design software processes that iteratively convert habitat data into higher and higher levels of abstraction. The DAA is a series of mathematical or logical transformations of telemetry data to provide appropriate inputs from a hardware system to a hardware system controller, system engineer, or crew. The DAA also formalizes the relationships between data and control and the relationships between the data themselves. We have connected our Data Abstraction Architecture to a simulation of a Lunar habitat in order to test its ability to aid in the monitoring and control functions.

## INTRODUCTION

Space systems are growing more and more complicated and containing more and more sensors. The large amount of data generated is overwhelming both ground controllers and automated control systems. Coping with this sensor data will require developing software systems that can iteratively convert raw sensor data into more meaningful, derived data that can be used by controllers, both human and automated. In this paper

we describe a data abstraction architecture that allows engineers to design software processes that convert habitat data into higher levels of abstraction. First, we present the architecture including its components and representations. Then we present some use cases in which the architecture is used to monitor and control a Lunar habitat. We then describe our experimental environment including a Lunar habitat simulation and discuss the performance of the data abstraction architecture in implementing the use cases. Next we talk about related work in this area. Finally, we discuss future directions and conclusions.

## DATA ABSTRACTION ARCHITECTURE

A data abstraction architecture (DAA) formalizes the relationship between raw telemetry and derived data. A set of integrated components and representations are part of the DAA, including:

- Data events define the data (both raw and derived) upon which the architecture operates. Data events can be generated by hardware or software and happen asynchronously.
- Data abstractors, which are software programs that perform a transformation on data events. They consume certain kinds of data events and produce different data events.
- Sensor event abstraction language (SEAL), which is an eXtensible Markup Language (XML) schema that defines a grammar for connecting data abstractors, events, sources and sinks.

- Data abstraction reasoning engine (DARE), which instantiates a SEAL file into a computer program that connects to data sources and data sinks to perform abstractions.
- Development environment, which is an end-user software tool to aid in the construction, debugging and viewing of SEAL files.

Together these components provide a mechanism for creating more abstract data from raw telemetry. Each of them will be discussed in the rest of this section.

## DATA EVENTS

Data events define the data upon which the DAA operates. Events are heterogeneous, hierarchical, multi-valued messages and may occur asynchronously. They can be generated from sensors, controllers or from data abstractors. These generators are called data sources. Data events are not simply a number, but instead are complicated data structures that can contain attributes such as confidence, timing, processing history, counts and arrays. Here is an example data event for a Lunar habitat scenario

```
{
  sensors: [
    {
      name: cabin pressure sensor,
      units: kPA,
      value: 101
    }
    {
      name: airlock pressure sensor,
      units: kPA,
      value: 85
    }
  ]
  average: 93
}
```

In this case the data event contains two sensed values, their units and their average. This would be produced from an averaging data abstractor.

## DATA ABSTRACTORS

Data abstractors transform data events by performing processing on them. There is no limit to the number or kinds of data abstractors that can be built. Each data abstractor is implemented in a common programming language such as Java. The abstractor must conform to an Application Programmers Interface (API) that allows it to receive and generated data events and to be controlled. We have implemented the following kinds of data abstractors:

- Reducing
  - Average: Collects a series of values and produces their average (choice of mean or median)

- Min/Max: Collects a series of values and produces their minimum or maximum value

- Timing

- Temporal Alignment: Outputs a single new data event containing a set of events that are all temporally common.
- Sampling: Accumulates a buffer of discrete events over a defined observation period and produces a single data event.

- Symbol Processing

- Categorical Binner: Reduce real value data events into symbol categories. For example, a temperature input could be reduced to “high”, “medium” or “low”.

- Propagation

- Propagate on Change: Only generates an output data event when the input data event has a different value from the previous event
- Quiescence: Only generates a data event when the input data event has remained constant for a set period of time.

- Mathematical and Logical

- Basic mathematical and logical operations

Several of these will be described in the use cases presented later in the paper.

## SENSOR EVENT ABSTRACTION LANGUAGE

The Sensor Event Abstraction Language (SEAL) is an XML grammar that defines data abstraction networks. It encodes the data abstractors and their parameters, the data events, the data sources and sinks and the connections between all of them. It also defines the data events. The SEAL syntax and semantics are intended to support the computational requirements of NASA telemetry and telemetry management processes and align to the conceptual model of those processes held by expert NASA flight control engineers. The language is intended to support rapid visual development and inspection of data transformation by skilled engineers who are typically trained in disciplines other than software engineering.