Data Management Backbone for Embedded and PC-based Systems Using OSA-CBM and OSA-EAI

Andreas Löhr[1], Conor Haines[2], and Matthias Buderath[3]

[1,2]*Linova Software GmbH, Garching b. München, 85748, Germany*

andreas.loehr@linova.de
conor.haines@linova.de

[3] *Cassidian, Manching, 85077, Germany*

matthias.buderath@cassidian.com

ABSTRACT

Cassidian is in the process of developing a comprehensive simulation framework for integrated system health monitoring and management research and development. One significant building block is to invite 1st class technology providers, e.g. Universities and SMIs, to provide innovative technologies and support their integration into the simulation framework. This paper is a joint presentation of Cassidian and Linova Software GmbH, a Cassidian preferred software provider.

Prognostic Health Management (PHM) systems are commonly composed of disparate and distributed hard- and software components. Further, these components exchange vast amounts of data over a heterogeneous collection of communication channels. Any such system's success depends upon an open, uniform, and performance-optimized solution for data management. A solution that includes: data definition, data communication, and data storage. The Open System Architecture for Condition-based Maintenance (OSA-CBM) and Open System Architecture for Enterprise Application Integration (OSA-EAI) are complementary reference architectures and represent an emerging standard for application domain-independent asset and condition data management. Herein, we will report on our experiences while implementing a data management backbone based on OSA-CBM and OSA-EAI for a simulation environment supporting PHM systems in the aerospace domain. Our work encompasses both airborne embedded systems and ground-based PC systems. While we can generally confirm the feasibility of OSA-CBM and OSA-EAI, we found several implementation recommendations unsuited to real-time operating conditions. To address these issues, we propose work towards standardizing non-XML-based transportation formats for OSA-CBM data packets. Further, we discovered issues specific to implementing the OSA-EAI data model in the aerospace domain. These issues drove our proposal to extend the OSA-EAI database model, where we seek to optimize its usability for analytical tasks. To underline the feasibility of our solutions, we provide empirical evidence drawn from our work. The conclusion is a summary of our experience and the direction of future work in the area of PHM system design for aircraft maintenance. In total, our contribution to the community is best seen from a practitioner's perspective. We aim to establish best practices for and contribute to the evolution of OSA-CBM and OSA-EAI.

1. SIMULATION ENVIRONMENT

The aerospace industry is a core application domain and development driver for PHM systems. The paradigm shift towards predictive maintenance which PHM systems impose to maintenance and overhaul processes promises higher aircraft availability coupled with lower overall maintenance costs. As in any other domain, challenges in introducing PHM systems to the aerospace domain are twofold. On the one hand, there are individual challenges in developing sensor technology, state detection, and health assessment methodologies/models for determining the future life span of a (possibly deteriorated) component. On the other hand, there are distinct challenges when integrating heterogeneous data from disparate and distributed sources into consolidated information and dependable decision support. This applies at both the aircraft and fleet level. It has therefore been recognized in the community that standardized and open data management solutions are crucial to the success of PHM. Such a standard should introduce a commonly accepted framework for data representation, data communication, and data storage.

EADS Deutschland GmbH, Cassidian, is developing a comprehensive simulation framework for research in the areas of condition monitoring and prognostic health management. The framework includes airborne functions hosted on embedded systems, as well as ground-based functions hosted on PC-based systems. The primary objective is to interconnect both airborne and ground-based systems using a uniform data management philosophy and, as far as possible, uniform communication protocols. In this paper, we report on experience from our task to define and implement the data management backbone for such a simulation framework. The backbone is based on the Open System Architecture for Condition-based Maintenance (OSA-CBM) and the Open System Architecture for Enterprise Application Integration (OSA-EAI).

1.1. OSA-CBM

The OSA-CBM reference architecture has become the de facto standard for exchanging data in a condition monitoring system. Being an implementation of the ISO-13374 functional specification, the architecture defines six functional layers. Each layer is allocated different and unique functions of the data processing chain in a condition monitoring system.(see Figure 1).

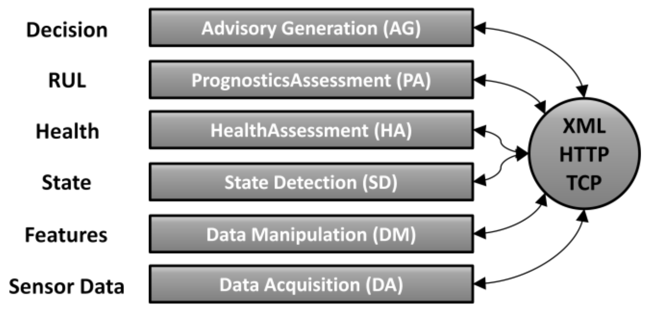

Figure 1.OSA-CBM Reference Architecture

This architecture focuses on the definition and communication of data. Specifically, on the question as to which data entities and events can be exchanged between the layers during operation and the communication interfaces used for this purpose. The format by which the data is exchanged between the layers remains unspecified; however, the usage of XML messages, which are transported over HTTP, is recommended. For this purpose, the standard provides a thorough collection of specifications for XML messages.

1.2. OSA-EAI

The reference architecture OSA-EAI is complementary to OSA-CBM. It specifies a comprehensive data storage architecture for asset management systems. This architecture consists of: a physical relational data model (Common Relational Information Schema, CRIS), a corresponding logical object model (Common Conceptual Object Model), and CRUD interfaces (Create, Retrieve, Update, Delete) for all defined entities in the data model, as depicted in Figure 2. In the course of harmonizing OSA-EAI with OSA-CBM, the data model defines entities that are capable of storing data originating from all six OSA-CBM layers. Analogously to OSA-CBM, it is recommended that clients interact with an OSA-EAI database via XML messages transported via HTTP. For this purpose, the authors of the OSA-EAI standard provide a multitude of CRUD XML message specifications. These specifications define how to manage data contained in the database and how to make the data available to any other stakeholder or application within a PHM system.

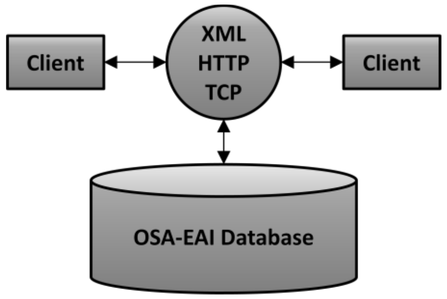

Figure 2. OSA-EAI Reference Architecture

A link to the MIMOSA organization, which maintains the reference architectures, can be found in the references section.

2. Simulation Environment

The simulation environment consists of an air segment and a ground segment, (inter-)connected by a data management backbone that relies on OSA-CBM and OSA-EAI. In the following section, we introduce the high level architecture of our simulation framework.

2.1. Air Segment

The air segment of the simulation framework models those systems and associated sensors for which we intend to develop IVHM capabilities. At the core of the framework is a central IVHM data processor. Sensors push their data to this IVHM data processor via an OSA-CBM compliant implementation. As a reflection of the working environment, the underlying message protocol is optimized for embedded systems (detailed in section 3). The IVHM data processor calculates IVHM information according to the OSA-CBM layer specifications, up to the health assessment layer (refer to Figure 3).

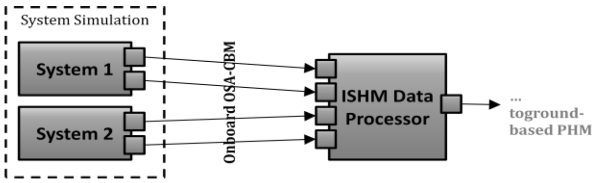

Figure 3. Air Segment of Simulation Framework

2.2. Ground Segment

The central data processor supports the downloading of data, which has been collected and calculated on board the aircraft, to the ground-based environment for further processing (e.g. during the aircraft's turnaround). Once downloaded, the data is stored in a central data management component, which we call the CBM data warehouse (refer to Figure 4).

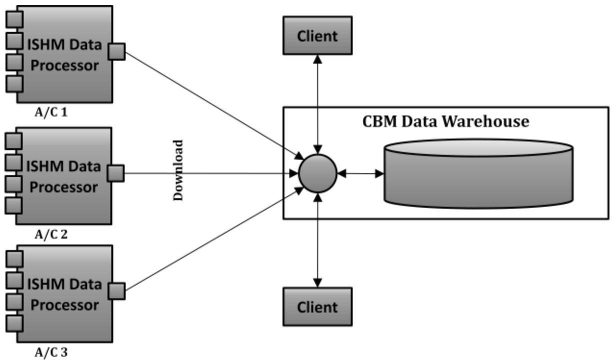

Figure 4. CBM Data Warehouse

The CBM data warehouse is based on the OSA-CBM/OSA-EAI reference architectures and it serves two major purposes: first, it hosts all current (i.e. short timeframe) and historical (i.e. long timeframe) condition data. Second, it provides services to distributed client applications that are involved in the PHM process. Such services include the CRUD interfaces as defined by OSA-EAI (e.g. for asset configuration management), high layer functions as defined by OSA-CBM (prognostic assessment and advisory generation), and other services relevant for a PHM system.

In our context, data management includes the entire data set life cycle: from initial instantiation of a sensor value, transportation to the IVHM data processor, downloading to the ground-based environment, on through to storage and further processing. In section 3 we discuss aspects of OSA-CBM-based data management in an embedded system. Section 4 derives from experience gained while realizing the CBM data warehouse.

3. OSA-CBM IN AN EMBEDDED SYSTEM

Following an initial implementation of OSA-CBM using XML messages transported via HTTP/TCP, we decided to use binary messages transported via a UDP/IP stack. This significant departure from the MIMOSA recommendations was driven by requirements that arose from our intended use of OSA-CBM in the context of embedded systems certified for in-flight usage. Our focus of interest for on-board implementation ranges from data acquisition layer up to health assessment and the following sections report about our experience in implementing these classes using the C programming language.

3.1. Environment

When fielding OSA-CBM compliant applications on embedded systems certified for in-flight usage, several issues are brought to the fore. Ultimately, two aspects defined the unique structure of our solution: resource limitation and non-dynamism. Computing hardware for avionics, due to qualification requirements, are generations behind present off the shelf computing hardware. Implementation rules for applications hosted on real-time operating systems (such as VxWorks) typically forbid dynamically allocating memory resources, as these operations are potentially non-deterministic and lead to memory leaks if not used carefully. This environment imposes further constraints on the solution space: due to qualification or certification requirements (depending on the risk class of the final system) all embedded code must be written in the C programming language. Furthermore, UDP must be used as the sole protocol for network communication.

3.2. Use Case and Design Considerations

We want to transmit a *heavy load data event set* which contains four heterogeneous OSA-CBM `DMDataSeq` events at individual sample rates of 160Hz, 360Hz and 1 kHz. Additionally, we want to transmit a *light load data event set*, containing a single `DMDataSeq` event recorded at 20Hz; both data event sets will be transmitted with a frequency of 1Hz.

Generating OSA-CBM compliant XML representing our two event sets and packaging the XML into UDP packages as ASCII code was a straight-forward implementation approach as it has been performed by others (Swearingen, Kajkowski, Bruggeman, Gilbertson &Dunsdon, 2007). Generally, it involves the following three steps:

1. Sender: assemble the XML from an internal data representation in memory
2. Sender: marshal the XML into a UDP package and send
3. Receiver: Unmarshal and parse the received XML and populate an internal data representation in memory

As we will show later on, in Table 1, using XML generates a structure in which 75% of the transmitted data is apportioned to meta-data defining the XML structure. Additionally, due to its absolute size, the heavy load data event set exceeds the maximum size of a UDP packet. While it would have been possible to split up its data into several UDP packages, we consider the ratio between meta-data and payload to be unsuited to the constrained allocation of computing resources. We acknowledge that if we assume our heavy and light load data event sets would be the only loads on the communication channel (e.g., ethernet), there is no risk that it will exceed transmission capacity; but this assumption may not hold in a real aircraft design where communication is channeled and, due to the availability of qualified or certified hardware, the transmission capacity might be drastically limited. We also researched XML parsers that are written in C, and therefore compile for embedded environments, (e.g., Mini-XML, Expat, RXP) but we found them incompatible with internal programming policies (static memory allocation). Additionally, the high risk involved in the certification or qualification of an XML parser for an embedded system finally drove our decision towards a non-XML-based binary solution for marshalling and unmarshalling OSA-CBM data.

3.3. Design and Implementation

OSA-CBM is an object-oriented specification and therefore makes use of polymorphism, which is the ability to create object attributes, object functions or even an entire object that has more than one form. Our implementation of OSA-CBM is based upon the representation of OSA-CBM classes by a set of C structures. The C programming language is procedural and does not offer native polymorphism. After analyzing data manipulation through health assessment layer communication classes of the OSA-CBM object model, we concluded that a mapping of OSA-CBM classes to C structures is possible. We will next explain our rationale in supporting this approach.

The C programming language decouples data from functionality, therefore we did not have to map polymorphism of functions (OSA-CBM does not define behavior of the classes, anyway). We also could not identify polymorphism of attributes for the classes of our interest. However, there is polymorphism of objects, i.e., specific derived classes inherit part of their structure from one or more base classes. We mapped this kind of polymorphism by initially modeling C structures for each root class (i.e., classes that do not have a base class in the OSA-CBM model). For all non-root classes we modeled a member in the derived class which is of the type of the respective super class. As an example, the structure for the data sequence event of the DM layer (`DMDataSeq`) is shown in Figure 5(c). The corresponding base class structures are shown in parts (b) and (a), respectively.

Within specific limits our approach is also able to emulate multiple inheritance by including more than one base class member; however, the part of the OSA-CBM data model that we focused on does not involve multiple inheritance. For transmission, multiple data event instances are bound together into a data event set. Regarding a single instance of an OSA-CBM base class, its actual subtype at runtime can be anything. This is critical to the C implementation as the `DataEventSet` class acts as a transportation container for any `DataEvent` instances. We solved this problem by introducing a constraint: an OSA-CBM data event set may only include data events of the same type. This allowed us to introduce a non-standard member on the `DataEventSet` class which is of enumerated type `OsacbmDataType` and which indicates the type of included events.

```
typedef struct {
   char pad[4];
   long id;
   double confid;                                   (a)
   int alertStatus;
   int isSimulationFinished;
} DataEvent;
```

```
typedef struct {
   char pad[4];
   DataStatus_ENUM dataStatus;                      (b)
   DataEvent baseClass;
} DMDataEvent;
```

```
typedef struct {
   char pad[4];
   int dataSize;
   double* values;
   double* xAxisDeltas;                             (c)
   double xAxisStart;
   DMDataEvent baseClass;
} DMDataSeq;
```

Figure 5. Exemplary Payload OSA-CBM Structures

The received byte stream can therefore be interpreted correctly on the receiver side. For the transmission itself, we copy a structure's memory image into a temporary buffer. Additionally, as required by the event type, the buffer memory is appended with a data block for each reference from a structure's pointer members (here: `values` and `xAxisDeltas`). Finally, the buffer is sent as a UDP packet to the receiver, where is reconstructed into a set of OSA-CBM compliant data. Consequently, we support both static data types (such as `DMReal`) and dynamic types (such as `DMDataSeq`). Though, as a necessary overhead, complex data sequences require recipient side remapping of pointers at run time and a maximum payload size must be defined for real time operation.

3.4. Evaluation

Quantitative evaluation will be accomplished here with a comparison between the data required for an ASCII XML data transmission versus that of our custom binary transmission protocol. We used Ubuntu 10.0.4 (32bit) as sender and VxWorks on Power PC (32bit) bit as recipient. Table **1** outlines the data characteristics of two representative communication samples.

```
typedef struct {
   long id;
   Site site;
   OsacbmTime time;
   int alertStatus;
   OsacbmDataType_ENUM osaCbmDataType;
   int noEvents;
   char* events;
} DataEventSet;
```

Figure 6. Data Event Set as C Structure

The first sample is a heavy load data event set. It contains four heterogeneous OSA-CBM `DMDataSeq` events at individual sample rates of 160Hz, 360Hz and 1 kHz. The overall data event set has a frequency of 1Hz. The resulting data push represents 2,520 individual measurements being sent across the system every second. The second sample is a light load data event set, containing a single `DMDataSeq` event recorded at 20Hz; the corresponding overall data event set has a frequency of 1Hz.

	XML	Binary	Ratio
Heavy Load	165 345 bytes	40 792 bytes	4.1
Light Load	1 827 bytes	576 bytes	3.2

Table 1. Data Transmission Size Comparison

As seen in Table 1, there is a significant reduction in the volume of data transmissions achieved by our approach, ranging up to a factor of four. An additional effect of our approach, as compared to sending XML messages via UDP instead of via HTPP/TCP (Swearingen, Kajkowski, Bruggeman, Gilbertson & Dunsdon, 2007) is a significant reduction in the processing overhead required by XML structural parsing; this reduction is beyond the scope of our present analysis.

However, there are drawbacks of our approach. As UDP is a stateless protocol, there is a cap on the amount of data that can be transmitted per event set. It is limited to the maximum allowed size of a UDP Data package (UDP specifies a maximum allowed size). Depending on platform specific settings the maximum available size can be significantly less.
We believe that this size limitation is best addressed by splitting the data set into a series of discrete packets, as opposed to introducing additional limitations and overheads on the binary transmission format. Data management within a closed on-board real-time environment a priori requires that the overall data communication is well designed regarding timing and loads. In such a closed and well controlled environment the likelihood of UDP packet loss is minimized, however, it may happen. Therefore, we propose the usage of UDP-based transmission only for functions which can cope with temporary gaps in their data input, such as our diagnostics algorithm. For functions which are not robust to data losses, a confirmation and resend protocol could be invented, but that would negate the usage of UDP and TCP would be the transmission protocol of choice.

Our current implementation is highly platform dependent as it is patched to meet the characteristics of our environment (sender 32bit Ubuntu, recipient 32bit VxWorks). To overcome platform differences we introduced artificial padding bytes (see C structure members in Figure 5) so that the internal in-memory arrangement is equal on both platforms and performed byte-swapping on the receiving platform. This allowed us to easily case the UDP package payload into the required structures (including pointer remapping).

Finally, XML messages can be read by humans more easily than binary messages. This may impose complications to the debugging cycles during software development; however, from our experience, software developers tend to develop the ability to “read” binary content over time, in particular if sophisticated Hex editor tools are being used. A steeper learning curve certainly is worth the performance gains. As for the generation of test data for certification or qualification, binary protocols do not impose significant overhead, as also with XML a generative approach will have to be used to deal with the large amount of test cases.

3.5. Outlook

Our initial implementation, transmitting the memory image of structures, is not optimal when communication must take place between heterogeneous platforms and only allows for a homogenous data event set payload. Yet, it yields significant performance gains, reduces the consumption of memory, and simplifies certification or qualification. As shown above, issues related to padding and regarding the arrangement of data in RAM may arise. While these issues can be mitigated if the characteristics of the platforms are known, the scalability in general remains limited. To address these issues, we started the development of a custom binary OSA-CBM protocol. The vision was to evolve this protocol as a generic and platform-independent means for transporting OSA-CBM events over the network in a binary fashion. In Figure 7 we provide an excerpt from our initial work to illustrate the proposed design approach. Based on preliminary low level definitions (such as big or little endian, widths of primitive data types) all OSA-CBM classes are modeled as a sequence of 16 Bit words. In our example, an ID consists of two words, i.e. it represents a 32bit integer value.

Analogously, the `OsacbmTime` class is represented as a sequence of five words (our customized implementation only required the `time_type` and `time_binary` attribute). With every class having such a specific representation, data events and entire heterogeneous data event sets can be assembled. For dynamic structures, upper bounds for the allowed amount of dynamic data must be defined (possibly implementation specific) in order to meet the requirements of real-time operating systems. To avoid sending spare data, the binary representation of such dynamic portions requires that one includes a member that defines the actually allocated amount of data (up to a maximum dictated by the data size allowed in a UDP packet). An example is the member `DMDataSeq.dataSize`, which is not part of the OSA-CBM specification but which is required for correctly interpreting the words. Checksums to detect transmission failures were foreseen as well. By standardizing the binary representation for the network format, senders as well as recipients have to translate between their platform specific representation and the network format. Although there is marshalling and un-marshalling to be done, we hypothesize that the CPU load for this process can be neglected compared to XML parsing.

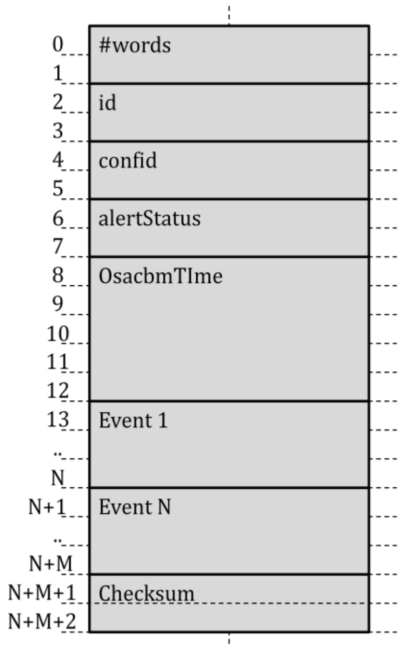

Figure 7. Exemplary binary representation of `DataEventSet`

Based upon results shown in the previous section, the size of data structures in this new network format will be in the area of 25% of a corresponding XML representation.

3.6. Binary Message Format in OSA-CBM 3.3.1

The most recent version of OSA-CBM, Version 3.3.1, includes a specification for a binary transmission format for OSA-CBM messages. We see our work confirmed by this addition to the OSA-CBM standard. Following an initial design and trade study, we decided to adopt MIMOSA's specification as the network layer format amongst our subsystems. Though this choice rendered our custom protocol design work moot, it is implementation that has been and remains the focus of our work. Furthermore, the compatibility of our systems with the rest of the community will be ensured by following a standard which is now part of that community. That is to say, our optimizations in the marshalling/un-marshalling of data within and amongst real time embedded systems and in the creation of an API/library for OSA-CBM transmission is just as critical while using the MIMOSA standard as with our custom message format. Our aim is to create a fully C coded, statically allocated implementation of the OSA-CBM Binary message specification for embedded systems.

4. CBM DATA WAREHOUSE

The ground segment of our simulation framework includes a central repository for data and information, called *the CBM data warehouse*.

4.1. High Level Requirements

Design of the CBM data warehouse was driven by the following high-level requirements.

1. The CBM data warehouse shall act as a central information system for all applications involved in the PHM process.
2. The CBM data warehouse shall provide a uniform and standardized interface for managing and querying its data.
3. The CBM data warehouse shall maintain full traceability for any in-service data item regarding origin, allocation (to assets, aircraft and flights) and changes.

Given the need to meet these requirements across a large fleet of aircraft, the design of the CBM data warehouse faces two core challenges. First, it must process a large number of transactions originating from daily maintenance tasks, such as asset installation/removal and storing newly available IVHM-data from performed flights. Second, it must process and store a large amount of historical data for performing diagnostics and prognostics, as well as their continual improvement as more in-service data becomes available.

4.2. Realization

The OSA-EAI and OSA-CBM reference architectures define a uniform data management philosophy that allows for full traceability of virtually any sensor value and its derived information. Earlier work (Gorinevsky, Smotrich, Mah, Srivastava, Keller & Felke, 2010, and others) demonstrated the feasibility of using these architectures as a reference to build a comprehensive information system and associated service interface across multiple domains, including aerospace. We consequently considered the selection of OSA-EAI and OSA-CBM as guidelines for the design of our CBM data warehouse as a promising approach to satisfy our high level requirements.

4.2.1. Scope

We have implemented a subset of the OSA-EAI standard for our initial version of the CBM data warehouse. The subset was derived with the aim of providing data management for diagnostics and prognostics on our candidate systems. Confirming reports from other researchers, we found the documentation of OSA-EAI to be rather sparse, especially when mapping its generic universe of entities to a specific application domain. We concentrated on the ability to express system breakdowns (Assets, Segments, and Parent/Child relations) and the ability to associate data from the data acquisition, data manipulation, and state detection layers. Additionally, each asset was to have an active history of health assessments and remaining useful life estimates. We expected that this would lead to an implementation of tables exclusively from the REG, DIAG, DYN and TREND groups of entities; however, with the exception of the TRACK group, we had to implement at least one table from all other entity groups in order to satisfy mandatory connections between tables. We consider this a symptom of the complexity of the OSA-EAI standard, and strongly encourage the maintainers of the standard to establish a sample or reference application for OSA-EAI (and OSA-CBM), similar to the SCOTT database example of Oracle.

4.2.2. Customization

We customized the remaining OSA-EAI tables in a way that would simplify the generation of test and reference data, but still allow for the drawing of general conclusions (congruent customization) from our experience. We made further customizations to map specific features of the aerospace domain (domain customizations). Many tables of OSA-EAI have a composite primary key (i.e. 2 or more columns) due to the fact that the database model is designed for data exchange or integration amongst different database instances. For this purpose OSA-EAI introduces the Site concept, which uniquely identifies the stakeholder of a specific dataset. In combination with the dataset ID, any dataset can thus be uniquely identified. Since our simulation framework is currently a closed system, the maintainer remains constant. Therefore, we stripped the composite primary keys of each entity down to a single dataset id, allowing us to strip down foreign keys as well. This approach was shown to be feasible by Mathew, Zhang, Zhang and Ma Lin (2006).

We further recognized that OSA-EAI does not have the specific notion of a flight, or a mission. This was not unexpected, as OSA-EAI is generic; however, analyses in the aerospace domain are often flight/mission centered. Per definition, OSA-EAI measurements can only be related to assets/agents and time. Additions were necessary to relate measurements with a specific flight/mission entity under which they occurred. These updates allow the system to couple flight/mission characteristics and degradation. While OSA-EAI foresees enough meta-data to perform a chronological mapping to an external flight/mission database, our experience from other projects shows that a direct mapping of information to a flight (or at least a power cycle) is inevitable.

In the aerospace domain, segments represent virtual “placeholders” for assets and these placeholders have unique logistic control numbers. Such features can be represented by OSA-EAI using the attributive tables for each segment (Segment Numeric Data or Segment Character Data). However, being modeled as an explicit attribute of a segment, the evaluation of logistic control numbers is more efficient. We recognize that one could come up with many such contra arguments, as OSA-EAI is a domain independent and generic standard.

4.2.3. Performance Considerations

Coping with a large number of transactions and handling large volumes of data at the same time, the CBM data warehouse has both the role of an Online Transaction Processing (OLTP) system and that of an Online Analytical Processing (OLAP) system. These two requirements seem to contradict each other at first glance.

The database model of an OLTP system is normalized, that is, it consists of many interconnected tables and each table describes a fine granular bit of the application domain. The number of tables that contain redundant information (possibly in different representations) is minimized so that the risk of a transaction leaving the database in an inconsistent state is low. Due to its appearance from a bird’s eye view, a normalized schema is referred to as a *snowflake* schema. For an OLTP system, normalization is a prerequisite, as it supports CRUD operations with optimal performance and data integrity. The downside of a snowflake schema is that information retrieval and analysis result in complex queries involving many tables, which results in bad performance.

The database model of an OLAP system is de-normalized, which means that it consists of few tables, which contain redundant information for the sake of reduced query complexity and minimal join operations. Due to its appearance from a bird’s eye view, a de-normalized OLAP schema is referred to as a *star* schema. Snowflake and star schema are depicted in Figure 8. The information of interest is marked as grey boxes. The OSA-EAI database model in its current state is heavily normalized and therefore clearly OLTP-centered. Others have confirmed this statement using formal methods (Mathew and Ma, 2007). Although we could confirm specific issues regarding modeling and documentation (Mathew et al., 2006), we still consider OSA-EAI as well defined for transactional tasks. In contrast to criticism that has been raised by industry, we consider the normalization of OSA-EAI as essential, whereas Mathew and Ma (2007) argue that the normalized character of OSA-EAI is one of its weaknesses.

Applying standard modeling techniques to selected subsets of interconnected OSA-EAI tables, they propose OLAP-centered alterations for OSA-EAI according to star schema design. These show that, at least for selected subsets of coherent CRIS tables (so called *data marts*), the OLAP-centered model holds equivalent information. Not surprisingly, Mathew and Ma (2007) acknowledged that their redesign optimizes analytics, but has significant drawbacks for transactional use. They conclude with a discussion of their motivation for further work towards a compromise.

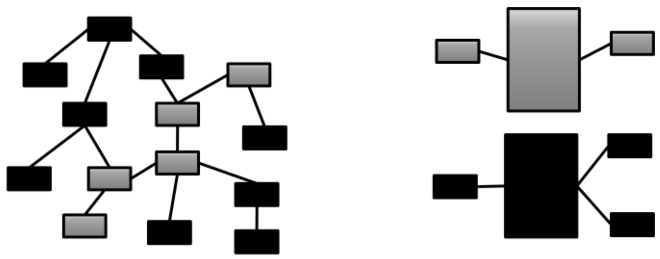

Figure 8. Snowflake (OLTP) vs. Star Schema (OLAP)

We argue that such a compromise cannot manifest as a single data model that features characteristics from both OLTP and OLAP-centered models. Such an approach would fit neither side. Instead, motivated from our findings during the realization of the CBM data warehouse and the experience from our other projects that deal with large data volumes (which go beyond the scope of this document), we propose an extension to OSA-EAI to specifically support analytical tasks on large volumes of historical data.

4.3. "Common Relational Analytics Schema"

The characteristics of OLTP and OLAP are too distinct to be merged into a single database model. The database model that is defined by OSA-EAI is called *Common Relational Information Schema* (CRIS). Instead of redesigning CRIS to include OLAP-specific features, we propose a new standardized database model named *Common Relational Analytics Schema* (CRAS). Our proposed database model lives under the umbrella of OSA-EAI and coexists with CRIS. Since an OLAP-centered database is primarily designed for reading (not writing), the CRAS portion of OSA-EAI will be populated on a regular basis from the content stored in the CRIS portion. Both portions hold an equivalent informational content – however, CRIS is optimized for transactional purposes while CRAS is optimized for analytical purposes.

4.3.1. Motivation

For a PHM system, it is necessary that prognosis be performed in a short timeframe, e.g. during the turnaround phase of an aircraft. However, this is different from actually performing analytics. At least the prognostics algorithms that we were utilizing require neither the entirety of all recorded historical data, nor any preprocessed results requiring filtering or aggregation (which are typical tasks of OLAP systems). A limited set of data, say from the last N flights, was sufficient. We found that with the standard CRIS queries these limited historical datasets could be retrieved reasonably fast. We draw this conclusion from our direct experience with the tools we created. Our sample database did not contain fleet condition data from several aircraft over several years. And with such huge amounts of data the performance will degrade. We hypothesize, however, that using table partitioning techniques, which have become available with today's relational database management system (such as Oracle's Enterprise Edition), it is possible to set an upper limit for the amount of data that has to be searched by a query to identify the prognostics raw data from the last N flights. An apparent partition key is time, but Site is also a promising candidate.

We further suggest that analysis tasks that would require an OLAP-centered database model be conducted on a regular basis, but decoupled from the daily operational (i.e. transactional) business. We claim that it is therefore suitable to populate the CRAS on demand (e.g. once a month) in order to perform retrospective analyses (e.g. for the continuous improvement of diagnosis and prognosis).

4.3.2. Architecture

A high level overview of our proposed architectural extensions of OSA-EAI is given in Figure 9. The elements drawn in grey represent the current state of the art of OSA-EAI. The OLTP-centered database model, CRIS, stores the operational data in a relational database (the corresponding object model has been omitted). Furthermore, the OSA-EAI standard defines a comprehensive service interface for accessing and modifying the operational data. We propose to extend OSA-EAI according to the following three aspects (corresponding to the black-marked items in Figure 9):

1. Database model that is optimized for analytical purposes (OLAP), which is able to store a congruent informational content as CRIS. We call this database model the Common Relational Analytics Schema (CRAS). It is organized according to the star schema approach.
2. A standardized interface for issuing multidimensional queries against CRAS.
3. Standardized *Extraction, Transformation and Loading* (ETL) process populating tables in the CRAS schema with operational data from CRIS.

4.3.3. Performance and Operational Considerations

Our work regarding CRAS suggests an a priori hybrid approach for database modeling. We are currently refining the concept and have just begun prototype implementations. Therefore, we cannot yet provide empirical results; in particular, when it comes to handling data volumes in the magnitude of terrabytes. For these volumes, the concept has yet to be proven. While the idea of CRAS as a complement to CRIS is clearly new, the methodology that it is based on, i.e., the star schema, has been available for years and is well understood. The star schema yields excellent performance results even with large data volumes. We have gained empirical knowledge from another work area which requires queries that involve both filters and aggregation. Results indicate a boost, due to the star schema approach, in the magnitude of 10 to 100 with respect to response time when handling millions of data sets.

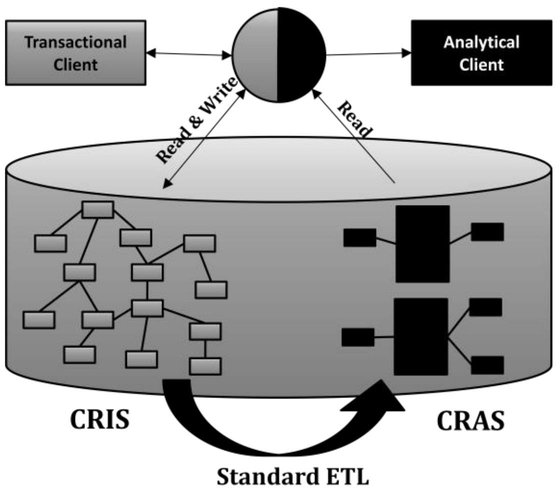

Figure 9. CRAS Extension of OSA-EAI (shown in black) with an optional data model which is optimized for analytics

To ensure scalability for the joint operation of CRIS and CRAS, we propose the following methodology. It is known that the performance of both the CRIS and CRAS schemas degrade with a growing amount of data. However, we believe the CRIS schema will degrade faster than the CRAS schema. Once a fresh system has been set up, the CRIS portion will be constantly populated with new data, and, in reasonably short intervals, the CRAS schema will be constantly recreated from the current data in CRIS by the ETL process. The CRAS schema is stateless at this phase, as it can always be recreated from CRIS. Operational tasks will be carried out in the CRIS, while analytical tasks run on the CRAS. Provided that suitable hardware segmentation is available (e.g., dedicated CPUs, dedicated RAID volumes) operations on both schemas should not influence each other. Once specific hot spots of the CRIS schema have degraded to a stage where performance is no longer acceptable, old data must be archived in the CRAS schema. We assume that one can define data as being old simply by its date of creation or other criteria. We further assume that such old data will not be altered due to operational processes; which certainly applies to sensor data. Therefore the ETL can move (instead of just transform) old data to CRAS where it will then permanently reside – just not in the CRIS form. Since there is no need to alter the old data, it can be removed from CRIS completely, mitigating the performance degradation. However, the old data is still available for analysis in CRAS. From this point on, the CRAS schema becomes stateful, as it cannot be entirely recreated from CRIS.

From a high level point of view, the CRIS schema's data volume will grow up to a specific limit and then shrink again, so there is a worst case performance for operational tasks. In contrast, the CRAS schema will constantly grow with each new archival process. However, the growth will take place in a database schema that is designed for performance and large volumes; nevertheless, without suitable measures the CRAS cannot grow indefinitely.

There are scaling measures to ensure performance of database schemas in general that can be applied to our situation. For data archived in CRAS which still needs to be considered during online analyses, so called *partitions* should be maintained. A partition influences the way a database physically stores a database table on the storage device but keeps this storage strategy transparent to the application (programmer). Partitions can be created during maintenance phases of the PHM system. Depending on specific criteria of the data set, such as the date of creation (the so called *partition key*) it will be assigned to one partition or the other. Partitions can be assigned a separate storage device, i.e., one disk for each partition. Therefore, even specific tables can be scaled independently from others. While the further discussion goes beyond the scope of this writing, the effect is that the search space for queries can be significantly reduced. Operational data that the ETL transforms from CRIS will have its own partition(s), whereas all archived data will have separate partitions. We believe therefore that the effects of a growing CRAS on the continuous ETL transformation of operational data can be mitigated. However, if the amount of data in CRAS significantly degrades the online analysis performance, one has to consider moving the oldest data from CRAS into offline storage. Here, we assume that this data no longer contributes to an operational PHM (e.g., data from assets that have been moved out of service) and can be analyzed offline (or e.g., in a separate database).

4.3.4. Challenges and Future Work

There are two core challenges involved in our work. First, the concept of joint operations between CRIS and CRAS needs to be proven. We have to derive enough sample data and set a representative database configuration and environment to prove our claim. In its current stage, this approach is merely a concept. While the methodologies and technology it is built upon have proven to be feasible in other domains, the risk of not being able to implement it as proposed is non-negligible. In the previous section we mention the introduction of offline storage for the oldest data in the system. We want to point out here a new aspect of performance research for OSA-EAI by combining it with *Hadoop*, an emerging technology for distributed storage and query of huge volumes of data. Second will be the derivation of a generic CRAS schema that fits the needs of analytical tasks for PHM in a domain-independent manner. This must be accomplished while maintaining the same level of quality as CRIS does in fitting the needs of transactional usage in a generic way. Mathew et al. (2007) have applied a formal process for attempting to derive an initial OLAP-centered database model from CRIS. They identified so called data marts (fact tables and corresponding dimensional tables) for the areas of configuration data, measurements, health and alarms, events and work management. However, they give no reason as to why no data mart for remaining useful life was identified. As such, the actual details of the generic ETL process are left open for future work.

5. Conclusion

We presented our experience from the realization of a data management backbone for a simulation framework for PHM systems in the aerospace domain. For the airborne segment OSA-CBM-based communication was chosen. We encountered issues relating to the recommended transportation protocol for OSA-CBM when implementing the standard under the conditions of a real-time operating system. From our findings, we are motivated to use a binary transportation format for OSA-CBM data events that address embedded systems. This standard is to be both binary and lean. In the process, we hope to avoid the inherent overhead in processing power and memory consumption of an XML-based transportation over HTTP. Our preliminary results are promising. They amount of raw data to represent specific OSA-CBM messages could be reduced to 25% of the XML-based size (overhead for HTTP and TCP not included). As our approach lacks platform independence we outline a path for future work towards a platform-independent binary representation for OSA-CBM messages. The ground-based part of our data management backbone is centered on an information system, which we call the CBM data warehouse. It is designed according to the OSA-EAI reference architecture. Confirming the feasibility of OSA-EAI in conjunction with OSA-CBM, we encountered minor issues in mapping aerospace domain concepts to the generic entities and could confirm issues reported by others. To answer the necessity of a PHM system to perform both transactional and analytical interaction with the CBM data warehouse, we recommend extensions to OSA-EAI. We propose an optional and complementary database model called CRAS (in analogy to CRIS) that is optimized for analytical queries and follows OLAP principles. It coexists with CRIS and is populated, on demand, by CRIS transactional data. We close by pressing for future work in this area in the form of field studies.

References

Gorinevsky, D., Smotrich, A., Mah, R., Srivastava A., Keller, K., &Felke, T. (2010). Open Architecture for Integrated Vehicle Health Management. *AAIA Infotech@Aerospace Conference*, April20-22

Mathew, A. D., &Ma, L. (2007). Multidimensional schemas for engineering asset management. *Proceedings World Congress on Engineering Asset Management*, Harrogate, England

Mathew, A. D., Zhang, L., Zhang, S., & Ma Lin (2006).A review of the MIMOSA OSA-EAI database for condition monitoring systems. *Proceedings World Congress on Engineering Asset Management*, Gold Coast, Australia

MIMOSA. Mimosa Organization Website. *http://www.mimosa.org*

Swearingen, K., Kajkowski, W., Bruggeman, B., Gilbertson, D., &Dunsdon, J. (2007). Multidimensional schemas for engineering asset management. *Proceedings IEEE Aerospace Conference*

Biographies

Matthias Buderath Aeronautical Engineer with more than 25 years of experience in structural design, system engineering and product- and service support. Main expertise and competence is related to system integrity management, service solution architecture and integrated system health monitoring and management. Today he is head of technology development at Cassidian. He is member of international Working Groups covering Through Life Cycle Management, Integrated System Health Management and Structural Health Management. He has published more than 50 papers in the field of Structural Health Management, Integrated Health Monitoring and Management, Structural Integrity Programme Management and Maintenance- and Fleet Information Management Systems.

Conor Haines received his B.Sc. degree in Aerospace Engineering from Virginia Polytechnic Institute and State University in 2003 and his M.Sc. degree in Computational Science from the Technical University of Munich in 2011. For 3 years Conor was a test engineer supporting the NASA Near Earth Network, providing simulation support used to guide system development. At his current post, he is focused on developing IVHM and Computer Vision technologies as a Software Engineer for Linova Software GmbH.

Andreas Löhr received his M.Sc. degree in Computer Science from the Technical University of Munich in 2001 (Informatics, Diplom) and earned his PhD degree in Computer Science from Technical University of Munich in 2006. For 6 years he worked as a software engineer at Inmedius Europa GmbH in the area of interactive technical publications and researched in the field of wearable computing. He founded Linova Software GmbH in 2008 and at his current post as managing director he focuses on development of maintenance information systems and data management architectures.