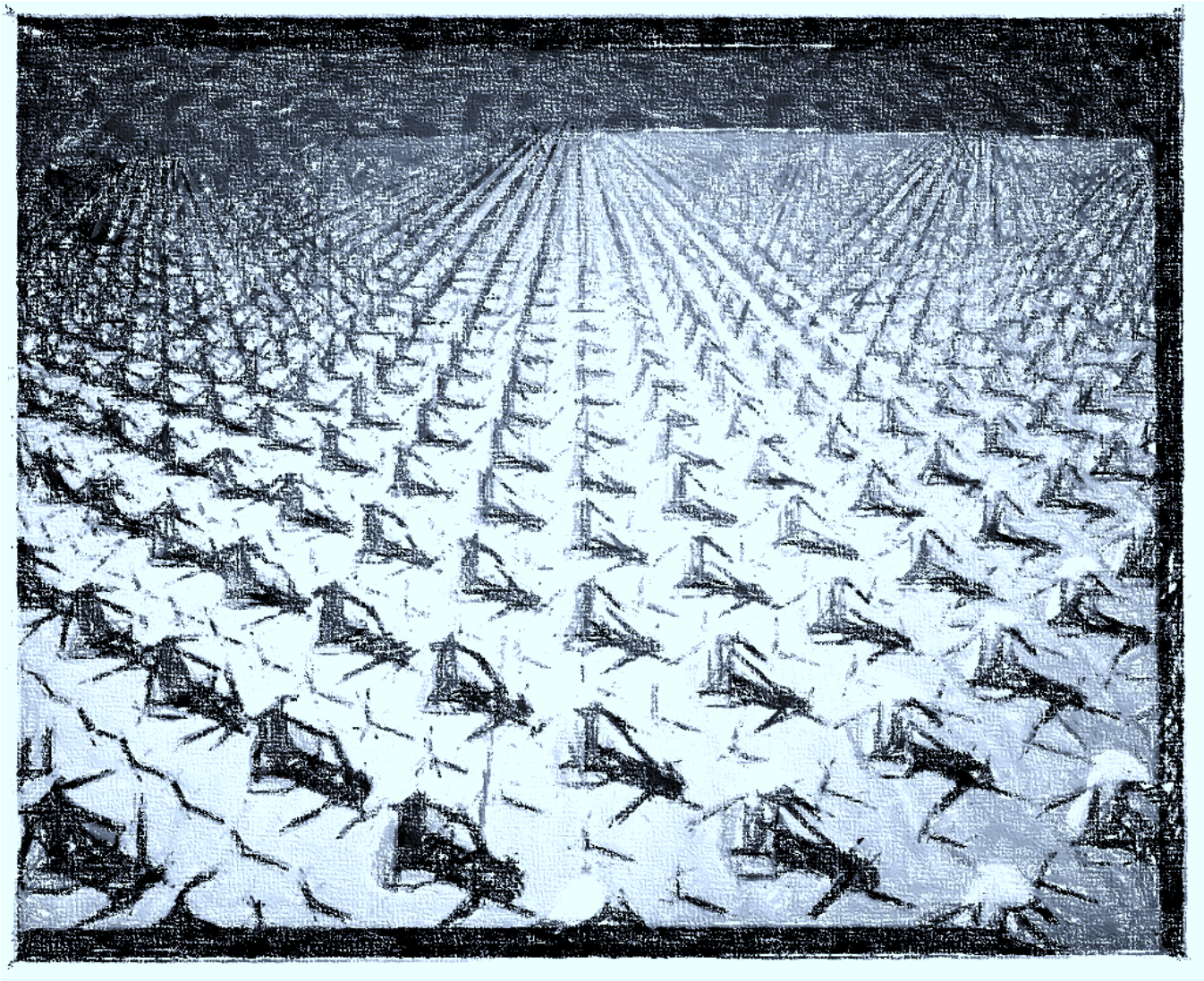

Avoiding Pitfalls to a Risky and Costly Real-Time System Lifecycle

A best practice for open and flexible data access for understanding, integrating, and testing real-time embedded systems.

TABLE OF CONTENTS

An Ecosystem Driving Real-Time Embedded System Lifecycle Costs and Risks

Throughout their lifecycle, real-time embedded systems must expose access to data artifacts to provide critical indicators of their performance. Such data artifacts are archived at pivotal points of execution to facilitate subsequent analysis, thereby avoiding interference with the system's tight timeline and limited resources. While access to real-time data is critical, it is not typically built into the system and instead evolves over the system's lifecycle. Efficient access is significant and vital to lowering lifecycle costs and risks – the Navy estimates that the "cost of analyzing alone is 40% to 60% of the development cost of large software systems."[1] Open, portable, self-describing, and flexible access to data artifacts is required to minimize the lifecycle costs and risks of real-time systems.

THE REAL-TIME EMBEDDED SYSTEM ECOSYSTEM

The ecosystem driving real-time embedded system lifecycle costs and risks refers to the complex network of technologies, processes, and stakeholders involved in developing and maintaining embedded systems that operate in real-time environments. Real-time embedded systems are computer systems that must respond to input or stimuli within a specified timeframe, often in milliseconds or microseconds, and are commonly used in applications such as aerospace, defense, automotive, and industrial control.

The ecosystem encompasses hardware and software components, development tools, testing and validation processes, supply chain management, and maintenance and support activities. It also involves various stakeholders, including hardware and software developers, system integrators, original equipment manufacturers (OEMs), and end-users.

The challenges of the ecosystem include balancing system performance, power consumption, and cost, while also addressing the risks associated with system failures, security breaches, and changing standards and regulations. Furthermore, the rapid pace of technological innovation and the increasing complexity of embedded systems present ongoing challenges that necessitate continuous improvement and adaptation.

Overall, understanding the ecosystem that drives real-time embedded system lifecycle costs and risks is critical for ensuring the reliability, security, and longevity of these systems, as well as for enabling innovation and a competitive advantage in industries that rely on them.

Real-time embedded systems are complex and costly to design, build, and test. Once operational, these systems require an elaborate ecosystem of experts to maintain and enhance them. Organizations that initially build these systems are under pressure to deliver and are not necessarily responsible for maintaining, operating, or enhancing the system. The front-line software development team, which typically decides on how data capture is implemented, does not have early incentives to invest in open and flexible

[1] Navy Automated Test & Analysis (ATA) Policy PEOIWSINST 3086.01, Dated APR 30 2010

data access strategies. Data capture and analysis is conventionally not addressed, or at best, not aligned with the development team's interests. The lack of forethought or misaligned interest triggers an entire lifecycle of substantial and growing costs and risks.

For example, in 1992, a team at TRW charged with developing a software upgrade for a US Air Force COBRA DANE[2] radar took some proactive steps in exposing real-time data. The approach included real-time capture of binary data combined with the use of a data dictionary to simplify "data reduction" access to the data[3]. The "dictionary" approach, innovative at the time, reduced the development team's time to maintain data access. However, over time, this necessitated that operations, sustainment, and enhancement stakeholders create and maintain their own custom software to analyze non-self-describing and platform– and project-specific data. This suboptimal process continues today, adding costs and unnecessary development over the system's lifecycle (22 years to date). The costs and risks of data processing and analysis compound, where, for example, the 5-year impact on US Missile Defense Agencies' costs adds up to $1.3B[4].

[2] U.S. Air Force COBRA DANE radar in Shemya, Alaska, www.mda.mil/system/sensors.html

[3] Ted Selig, Denise Brunelle Priess, and Brian D. Mack. 1992. Data analysis and reporting for real-time Ada systems. In Proceedings of the conference on TRI-Ada '92 (TRI-Ada '92), New York, NY, USA, 469-476.

[4] Ted Selig, July 27, 2025, 3 Reasons Why the Missile Defense Agency wants FishEye's Pelagic, FishEye Software, Inc.

Vital Elements of Data Access

Suboptimal capture and analysis decisions resulting from a near-term focus can impact development projects early in their lifecycle, when the system starts to operate in real-time and is initially integrated. The decisions' impact snowballs later in the system lifecycle, with a growing community of stakeholders, extended use reaching around-the-clock operation, and new teams providing system maintenance and enhancements. Alternatively, open access to real-time data provides benefits by fully supporting development, minimizing unexpected costs, and maximizing system capabilities. This is accomplished by offering all stakeholders simple, flexible, and open access to real-time system data and includes these vital elements:

High-Performance	• So data capture does not impact real-time system operation • To reduce costs of analysis over the system life cycle
Portable	• To Enable source data from specialized embedded hardware and computers to be moved to destination general purpose computers and tools that are used by downstream analysis and data fusion systems
Self-Describing Archives	• To decouple analysis and validation tools from the embedded application to allow independent development cycles. • To decouple a large community of data consumers from the embedded application development environment
Open	• To allow access without requiring project-specific or proprietary knowledge or tools
Revealing	• To allow access to internal data and data not pre-designed to be exposed
Low Total Cost of Ownership	• To minimize the long term labor and computing costs of post-processing data

LONG-TERM CHARACTERISTICS OF REAL-TIME SYSTEMS

Some characteristics of long-term of real-time embedded system maintenance and enhancement are:

1. As changes to application software are made over time, its data types also evolve (e.g., revising existing data type declarations, adding/removing data types);
2. Experience with the system can result in needed changes to data capture choices (e.g., removing capture choices, adding new capture points);
3. As more types of data are exchanged among increasing numbers of applications, the relative meanings of the exchanged data become difficult to discern (e.g., differing structure and terminology introduce semantic gaps), undermining interoperability in Systems of Systems; and
4. In contemporary computing grids, large volumes of a variety of data types are being distributed at high velocity (Big Fast Data), but the conventional point solution approach is not sufficient to support data exchange needs in a timely manner nor can it scale to expansive networks (i.e., MxN point-to-point custom converters cannot be readily built nor can they efficiently handle exchange of large volumes of data among large numbers of exchange partners).

FOUR APPROACHES TO EXPOSING REAL-TIME EMBEDDED DATA

Real-time embedded system data capture and analysis breaks into three conventional approaches and one emerging approach:

1.	**Binary**	Conventional
2.	**Text**	Conventional
3.	**Network**	Conventional
4.	**Meta-Data Injection**	Emerging

The Binary approach simply takes snapshots of real-time data, tags and records them in a file, which is later post-processed by offline, specialized tools. The Text approach converts real-time binary data into ASCII text and stores it in a Text file. The Text data can later be read into offline generalized tools and analyzed once it is converted from text format back into binary data. The Network approach eavesdrops on data exchange protocols to capture data sent between subsystems. A fourth approach, Meta-Data Injection, inverts the conventional processes by collecting metadata offline before real-time execution, thus enabling data to be recorded in a binary format file that is open, self-describing, and does not require specialized tools or costly text conversions.

NATURAL TENDENCIES IN REAL-TIME SYSTEM DEVELOPMENT

Real-time system designers and developers typically give secondary consideration to a collection of system data. Even less consideration is given to non-mission-critical data that is required to integrate, test, validate, debug, maintain, and enhance systems over their lifecycle. In the early stages of the system life cycle, it is easy to slide into a process that captures a simple piece of data using a conventional Binary to Text approach. Such approaches seem like a fruitful way forward, but they can quickly become a disadvantage and be difficult to escape while under the pressures of new system development. Resistance builds to transitioning to another approach when the development budget is no longer available, and downstream costs are accepted and funded. Once adopted, the conventional approaches lead to

- Increased program risks stemming from a lack of visibility into critical data for debugging, validation, and innovation
- Difficulty in maintaining and expanding system capabilities
- Substantially higher costs for the systems' downstream ecosystem of analysts, data consumers, system operators, and maintainers

With foresight and understanding, a system can easily adopt an emerging approach that lowers cost, risk, and time to market while avoiding the difficulties of the conventional approaches. This emerging approach requires dedication and an understanding among stakeholders that a commitment to an advanced approach, although it necessitates an initial investment in understanding and implementing the approach, will result in a significant reduction of lifecycle costs and risks, while improving sustainability.

Approaches to Real-Time Embedded Data

CONVENTIONAL APPROACH – BINARY DATA RECORDING

In the early days of computing (pre-2000s), computer hardware and storage devices were not powerful enough to handle the processing and memory required for data analysis performed simultaneously with application processing. During runtime, the captured data would be expediently recorded in a format native to the application and computing platform, saving analysis for offline processing. The typical approach was to dedicate budget and resources to developing custom converters for each individual application and platform.

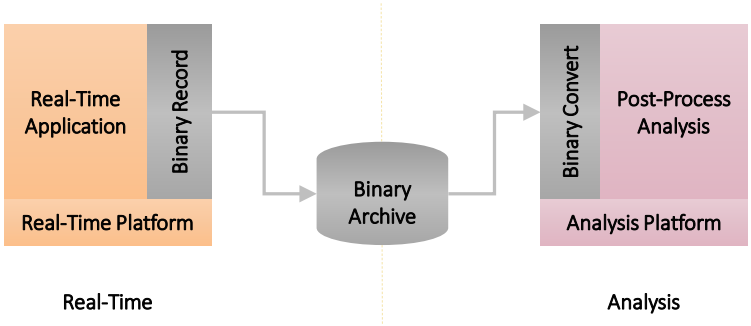

Figure 1. Conventional Binary Data Collection and Data Access

The conventional binary data capture methods offer various pros and cons:

Table 1 – Binary Data Pros and Cons

PROS	CONS
• Most efficient and highest performing method to persist captured data (i.e., saving data to a disk drive)	• Requires the development of real-time software to extract data from the real-time critical path, tag the data type, and record the data to disk. • Requires development of application specific tools to access data. • Requires data handling and analysis tools to be updated when application data or run-time platforms change. Configuration management of tools and applications is complex and expensive which can be compounded when there are many data consumers. • Requires use of the operational embedded computing resources because it is platform-specific. Also needs specialized conversion tools to convert the data artifacts from binary files. Embedded computing hardware can be specialized and expensive and is thus a limited resource.

The Binary approach offers high performance, but not the other elements required to maximize lifecycle costs & risks.

Table 2 – Binary Approach – Provides High-Performance but no other Elements

	BINARY
High Performance	✓
Portable	×
Self-Describing	×
Open	×
Revealing	✓
Low Total Cost of Ownership	×

Clearly, having data capture technology tied to a specific program results in solutions that run the risk of becoming brittle, non-scalable, non-maintainable, or obsolete. Furthermore, proprietary, non-standard data formats increase the cost of data analysis tools and hinder the ability to add future capabilities that utilize that data.

CONVENTIONAL APPROACH – TEXT DATA RECORDING

More recently, computing resources have become affordable commodity items and same-time analysis is now a viable option. However, due to the absence of enterprise data processing solutions (e.g., standard formats, platform-independent capture tools), the conventional approach has remained largely unchanged (Figure 2): point-specific tools for data capture, along with custom converters. Even with the emergence of open data exchange standards such as eXtensible Markup Language (XML)[5] and JSON, standard media converters nevertheless add processing and transport overhead that impedes the performance of real-time systems.

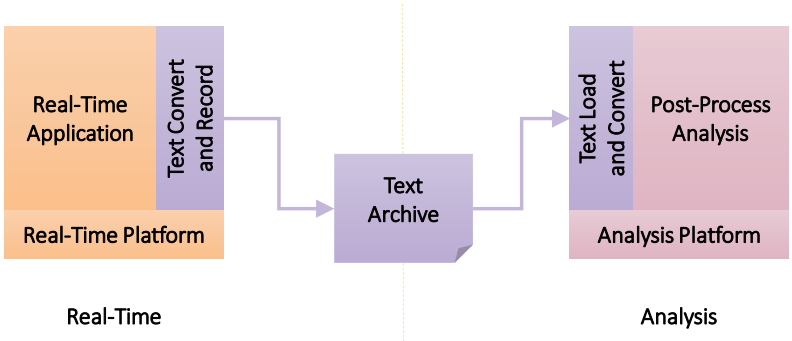

Figure 2. Conventional Text Data Collection and Data Access

[5] http://en.wikipedia.org/wiki/XML

Table 3 – Text Data Approach Pros and Cons

PROS	CONS
• Captured data is portable across platforms allowing analysis to be performed on general computers. • Captured data file is self-describing, allowing downstream data to be easily interpreted	• Requires the development of real-time software to convert data from the real-time binary to ASCII formats. • Loads real-time system with binary to ASCII conversion processing. • Leads to significant increase in storage space and network bandwidth to store and transfer data files. • Requires post processing tools to convert ASCII data back to binary before using the data, thus adding time for each analysis.

The Text approach offers flexible data that is portable, self-describing, and open, but fails to provide optimal performance and increases lifecycle costs.

Table 4 – Text Approach – Provides Flexible Data at a Cost

	TEXT
High Performance	×
Portable	✓
Self-Describing	✓
Open	✓
Revealing	✓
Low Total Cost of Ownership	×

CONVENTIONAL APPROACH – NETWORK RECORDING

Often, real-time embedded systems are made from multiple subsystems that communicate within a single processor or between computers over a network. This communication follows a selected protocol that ensures timely and accurate data transfer, which may cross different operating systems, applications, and computer platforms. Data must travel through communication protocol stacks and be converted to a machine-independent format to allow travel across different types of computing platforms. Data may be transmitted using a standard communication protocol (e.g., XML, DDS, ASCII, RPC, Java RMI, CORBA).

The advantage of this Network approach is that the development effort and processing required to expose the data are already available for the application, making it easy to use for capture and post-analysis. In such an approach, data is converted at the source using a resident Data Schema. The receiving platform converts the data back to its representation using a copy of the same Data Schema. This approach requires coordination to ensure the applications maintain common Data Schema representations. One consequence of manual coordination is that it is difficult or impossible to change a Data Schema at runtime. Another consequence is that the approach requires computer resources on the source and target platforms every time data is sent between platforms.

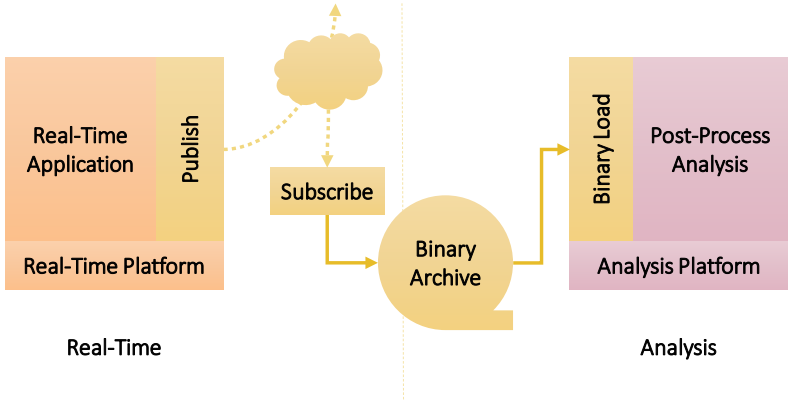

Figure 3. Conventional Network Collection and Data Access

Table 5 – Network Approach Pros and Cons

PROS	CONS
• The application is already exposing and formatting data so limited incremental processing is needed for data capture. • Data is portable across platforms allowing analysis to be performed on general purpose computers. • Captured data can be self-describing, allowing downstream data to be easily interpreted	• Data access limited to inter-subsystem communication data. • Requires the development of real-time software to convert data from the real-time binary to ASCII formats. • Loads real-time system with protocol conversion processing. • Leads to significant increase in storage space and network bandwidth to store and transfer data files. • Requires post processing tools to convert ASCII data back to binary before using the data, thus adding time for each analysis.

The Network approach offers some flexibility that depends on the network transfer protocols, but still fails to provide optimal performance while increasing lifecycle costs.

Table 6 – Network Approach – Provides Some Flexibility but at a Cost

	NETWORK
High Performance	×
Portable	✓
Self-Describing	?
Open	?
Revealing	×
Low Total Cost of Ownership	×

EMERGING APPROACH – METADATA INJECTION DATA RECORDING

An emerging approach called "MetaData Injection" flips the process by extracting an application's natural data format before run-time and injecting that MetaData to streamline real-time capture while creating an open and self-describing data access for data fusion and analysis. The approach adheres to a philosophy of working with data in its original and natural form, only converting it when necessary to optimize real-time performance. The high-performance process is made possible with software tools that extract MetaData describing the combination of platform, operating system, compiler, and application natural data binary formats. With access to MetaData upfront, an HDF5 (Hierarchical Data Format[6]) file format can be pre-populated with the data, and real-time data can be seamlessly flowed into the file. Upfront MetaData also enables real-time data transformation, integrates with other subsystems through distribution network protocols, and provides access for real-time analysis.

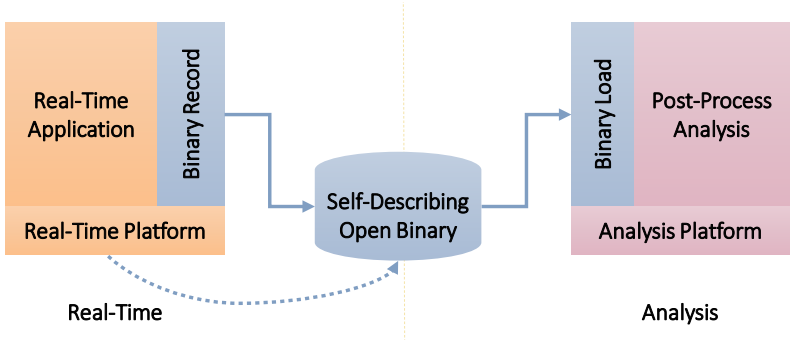

Figure 4. MetaData Injection Approach, Data Collection, and Data Access

Table 7 – MetaData Injection Approach Pros and Cons

PROS	CONS
• Recording in native binary format provides high-performance in real-time • Captured data is portable across platforms allowing analysis to be performed on general computers. • Captured data file is self-describing, thus allowing downstream data to be easily interpreted • Analysis does not require reformatting of data • Minimizes storage requirements without requiring additional processing • Open format allows anyone to access the data	• Required up-front commitment of a new-system development team to the approach • Requires investment to transition existing system to the approach.

[6] https://www.hdfgroup.org/why_hdf/

The MetaData Injection approach is able to meet all the objectives to minimize lifecycle costs & risks

Table 8 – MetaData Injection Approach – Meets Objectives to Minimize Lifecycle Costs & Risks

	METADATA INJECTION
High Performance	✓
Portable	✓
Self-Describing	✓
Open	✓
Revealing	✓
Low Total Cost of Ownership	✓

Avoiding Pitfalls to a Risky and Costly Lifecycle

Minimizing a real-time system's lifecycle costs and risks can be achieved with a little forethought and planning in the system's data capture and analysis approach. When evaluating the approach, it is also important to understand that the system's software development team has a significant influence on the approach taken, as they implement it. While it can be easy for a project and development team to marginalize this decision, it has a substantial and lasting impact on the entire system lifecycle.

A real-time system's lifecycle costs and risks can be minimized by ensuring that both planned and unplanned data artifacts are exposed to a broad audience, thereby supporting the system's operational and performance objectives. A process that combines the Binary approach, using high-performance native binaries with attributes of the Text approach, provides the opportunity to achieve the best practice approach, offering portable, open, and self-describing data. This MetaData Injection approach, when adopted, minimizes the lifecycle costs and risks of a real-time system.

Table 9 – Summary of Approaches

	BINARY	TEXT	NETWORK	METADATA INJECTION
High Performance	✓	×	×	✓
Portable	×	✓	✓	✓
Self-Describing	×	✓	?	✓
Open	×	✓	?	✓
Revealing	✓	✓	×	✓
Low Total Cost of Ownership	×	×	×	✓

When a system's full-lifecycle stakeholders collaborate and take the initiative, the MetaData Injection approach can quickly provide a method to lower the systems' lifecycle costs and risks. A balanced representation from the lifecycle stakeholders can easily highlight the importance of portable, self-describing, open, revealing and portable real-time data that will inspire more capabilities at a lower cost while reducing risk. The development team will even realize these benefits immediately after their influential decision on the direction forward. An enlightened move by system stakeholders can make a significant impact to a large agency by increasing system value and reducing risk while savings billions from the agency's lifecycle costs.

Terms

The following are some terms used in the white paper.

ASCII	American Standard Code for Information Interchange is a character-encoding scheme
CORBA	Common Object Request Broker Architecture
Data	**Data** is a set of values that is stored or transmitted, can represent similar meaning, yet differs in binary format on varying computing platforms. Data includes characteristics such as: • Data formats like integer, float, enumerations (numbers that represent different meanings like 1 = True and 0 = False) or ASCII character representations • Size and structure of data like 8-bit or 16-bit characters, 16, 32, 64-bit integers, 32 or 64-bit floats in formats like IEEE 754[7] or MiniFloats[8] , Fixed Point, Binary Coded Decimal • Data identifiers so that instructions can access and manipulate the values in memory • Data semantics like arrays of Data or memory pointers to Data • Endianness[9] convention for the byte order used to encode binary data • Bit Packing or Bit Maps[10]
Data Schema	A **Data Schema** specifies how the physical storage of Data on a Platform maps into its computational form. Note that Data Schema are NOT typically output from conventional compilation procedures.
DDS	Data Distribution Service, an Object Management Group standard for publish/subscribe middleware for distributed systems.
Java RMI	Java Remote Method Invocation performs the object-oriented equivalent of remote procedure calls (RPC)
Native Format Data	**Native Format Data** is Data consisting only of its in-memory representation for a particular Platform.
Operating System	An **Operating System** is software that manages resources and provides common services for computer programs.
Platform	The **Platform** represents a subset[11] of the components of a Computer comprising any number of the following: Processor, Operating System, and Data Storage.
RPC	Remote procedure call is an inter-process communication technique in networked computing
XML	Extensible Markup Language (XML) is syntax for encoding documents.

[7] http://en.wikipedia.org/wiki/Floating_point_arithmetic#IEEE_754:_floating_point_in_modern_computers

[8] http://en.wikipedia.org/wiki/Minifloat

[9] http://en.wikipedia.org/wiki/Endianness

[10] http://en.wikipedia.org/wiki/Bitmap

[11] It is recognized that a computer has other components such as a monitor, a mouse, keyboard, etc., but only the components listed below are relevant to this invention.

Authors

This paper was authored by Ted Selig with input and edits from Stan Nissen. It summarizes some of work and excerpts from a confidential paper titled “Ontology-Based Concurrent Information Synthesis (OCIS)”, July 9, 2014 co-authored by Mike Ackroyd and Ted Selig. Revision July 27, 2025

Mike Ackroyd
Engineering Fellow

Ted Selig
Director & COO

Company Information

FishEye, Software, Inc.
One Mill and Main, Suite 200, Maynard, MA 01754, USA
800.513.0881

http://www.FishEyeSoftware.com/Pelagic

July 2025