Assuring Software Reliability While Using Web Services and Commercial Products

Jeffrey O'Leary

En-Route and Oceanic Programs, Air Traffic Organization (ATO-E)
US Federal Aviation Administration, Washington DC 20591, USA
Jeff.Oleary@FAA.Gov

Abstract. FAA's recent large Ada based En-Route Automation Modernization (ERAM) program has reintegrated many disparate system components into a modern composite architecture. The program must now deliver on its promise to facilitate the evolution of the U.S. National Airspace System (NAS) by integrating Next Generation Air Traffic Control (ATC) capabilities starting with System Wide Information Management (SWIM), Automatic Dependent Surveillance (ADS-Broadcast) and the En Route Data Communications (Data Comm). One of the major challenges is to implement and leverage more open, flexible interfaces made possible by web service technologies and to ensure reliability and security of high performance data and communications services despite increased reliance on less trusted commercial products. The paper focuses on maturity, problems and lessons learned during the development of the initial SWIM as a Service Oriented Architecture (SOA) extension to the En Route Automation Modernization ERAM System.

Keywords: FAA, Air Traffic Control, ATC, Web Technologies, High Reliability, Safety Critical, ERAM, SWIM, DataComm, ADS-B, FUSE™, Lessons Learned, SOA, SOA Security.

1 Introduction

The U.S. Federal Aviation Administration (FAA) manages sixty percent of the world's air travel. The agency depends upon large, complex and highly available and reliable software systems to manage the vast commercial and civil aviation network and to carry out the agency's mission of ensuring high capacity, efficient and extremely safe air travel for the flying public. Ada has become a strategic technology in developing and sustaining systems that require high availability and high reliability.

ERAM, developed by the Lockheed Martin Corporation provides the automation services for the En Route domain at the twenty Continental United States Air Route Traffic Control Centers (ARTCCs). The ERAM system is comprised of three environments: 1) Operational, 2) Support, and 3) Test and Training and is the backbone of the U.S. En-route Air Traffic Control system. The current total ERAM software size is approximately 1.45 million lines of code with more than fifty percent of the operational code written in Ada.

The Next Generation Air Transportation System (NextGen) [1] is the Federal Aviation Administration's (FAA) plan to modernize the National Airspace System (NAS) through year 2025. Through NextGen, the FAA is addressing the impact of air traffic growth by increasing NAS capacity and efficiency while simultaneously improving safety, reducing environmental impacts, and increasing user access to the NAS. NextGen consists of five major elements:

SWIM will provide a single infrastructure and information management system to deliver high quality, timely data to many users and applications. By reducing the number and types of interfaces and systems, SWIM will reduce data redundancy and better facilitate multi-user information sharing. SWIM will also enable new modes of decision making as information is more easily accessed.

ADS-B will use the Global Positioning System (GPS) satellite signals to provide air traffic controllers and pilots with much more accurate information that will help to keep aircraft safely separated in the sky and on runways. Aircraft transponders receive GPS signals and use them to determine the aircraft's precise position in the sky. This and other data is then broadcast to other aircraft and air traffic control. Once fully established, both pilots and air traffic controllers will, for the first time, see the same real-time display of air traffic, substantially improving safety. The FAA will mandate the avionics necessary for implementing ADS-B.

DataComm: Current communications between aircrew and air traffic control, and between air traffic controllers, are largely realized through voice communications. Initially, the introduction of data communications will provide an additional means of two-way communication for air traffic control clearances, instructions, advisories, flight crew requests and reports. With the majority of aircraft data link equipped, the exchange of routine controller-pilot messages and clearances via data link will enable controllers to handle more traffic. This will improve air traffic controller productivity, enhancing capacity and safety.

Next Generation Network Enabled Weather (NNEW) - Seventy percent of NAS delays are attributed to weather every year. The goal of NNEW is to cut weather-related delays at least in half. Tens of thousands of global weather observations and sensor reports from ground, airborne and space-based sources will blend into a single national weather information system, updated in real time. NNEW will provide a common weather picture across the national airspace system, and enable better air transportation decision making.

NAS voice switch (NVS) - There are currently seventeen different voice switching systems in the NAS, some in use for more than twenty years. NVS will replace these systems with a single air/ground and ground/ground voice communications system.

ERAM has already been developed and currently is in the deployment phase. Several of the En-route sites have already started using the ERAM system for air traffic services. Being the backbone of the En-route ATC System, all five major NextGen elements will need to integrate with ERAM. The author is involved with the software development of the ERAM programs and will oversee development and integration of the subsystems and components to incorporate the NextGen elements and capabilities.

2 Achieving Software Reliability

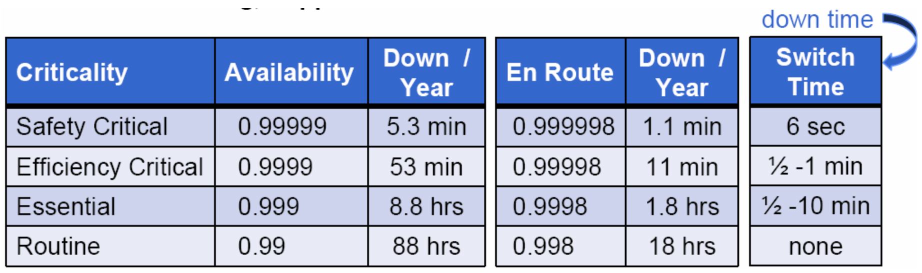
FAA's Software systems are growing in complexity and size, and new software paradigms support new forms of mix and dynamic progression of software applications. Following sections discusses the En Route systems environment and challenges inherent in software development of the web technology- based SWIM system, called ERAM SWIM.

2.1 NAS Service Criticality

The FAA defines service criticality in National Air Space Document NAS-SR-1000A [2] for each Air Traffic Control (ATC) service as

- Critical – A service that if lost would raise the risk associated with providing safe and efficient local NAS operations to an unacceptable level
- Essential – service if lost would significantly degrade ATC such as weather, General Info, Monitor & Control, System Data Recording
- Routine – service if lost would not significantly impact ATC such as Test & Training, Support

In terms of the availability, down time and switch time, numbers converts to the following –



Criticality	Availability	Down / Year	En Route	Down / Year	Switch Time
Safety Critical	0.99999	5.3 min	0.999998	1.1 min	6 sec
Efficiency Critical	0.9999	53 min	0.99998	11 min	½ -1 min
Essential	0.999	8.8 hrs	0.9998	1.8 hrs	½ -10 min
Routine	0.99	88 hrs	0.998	18 hrs	none

Fig. 1. Service Criticality, Down Time and Switch Time for En-route ATC Systems

The ERAM software implements requirements associated with each of these categories: radar data processing and display is safety critical while flight data processing is efficiency critical. The ERAM SWIM program was originally conceived as being essential, but as the SWIM program has evolved it became increasingly clear that the system or subsystem services provided would evolve as it offered opportunities to increase automation in efficiency critical flight data coordination across NAS systems. For example, the preliminary architecture for the initial En Route and Terminal portions of DataComm proposed to utilize ERAM SWIM to facilitate sharing of flight data from all twenty ERAM centers to support the logon and context management functions. This also introduced new requirements for explicit safety assurance engineering mitigation in both ERAM and ERAM SWIM. Switch time, of course, alludes to the reliability and fault tolerance strategies employed to achieve the higher availability requirements. These strategies include synchronized hot standby Address Spaces (AS), robust error detection and rapid recovery including data reconstitution redundant processor resources, and redundant network paths guaranteeing message atomicity, ordering, and integrity. Address Space (AS) is a unit of work dispatchable by the operating system. Each AS occupies its own area of memory in the processor and each AS contains one Ada main program. Reference [3] provides a good overview of the ERAM architecture; for our purposes it is sufficient to understand that reliability engineering is a prime driving consideration in the architecture and development of En Route Systems. To achieve the availability and reliability profile, the En Route program office has implemented an increasingly feature rich middleware supporting various capabilities needed to facilitate real world operations mostly supported by the FlightDeck® [4]. ERAM is the latest evolutionary step and implements a layered API-based architecture of system management and real-time monitoring capabilities. Table 1 identifies a subset of those capabilities that are vital to the reliability strategies of ERAM.

Table 1. Key ERAM Middleware FlightDeck® Features supporting RMA

Fault Tolerance features	Low-level crash and hang detection, notification to backup resources to become primary, fault data recording, and automatic restart and recovery of failed components.
Monitoring and Control features	Continuously updated, detailed status and performance data from each application, processor and network element; failure alerts and warnings; operator commands to element to restart, switch, stop, change modes (active, backup, test); monitor system security; generate detailed off-line performance and status reports; configure and control simulations.
Software Upgrade and System Maintenance features	Ability to download, load and cutover new versions of system and application software or adaptation without impacting user operations/cutover. Ability to remove and replace hardware and configure new resources
Support for data recovery, capture and debugging	Extensive reconfigurable data recording of system and application state, messages and data; off-line and on-line diagnostics, file management and state service check pointing to support application synchronization, switching and restarting even (e.g.) from complete facility power failures.

ERAM and the other major NAS systems have developed and evolved these capabilities and the software to implement them in order to achieve the Reliability, Maintainability and Availability (RMA) and system requirements. The FAA system users who monitor, control, repair and support the operational systems have developed requirements, detailed expectations, and extensive procedures for managing and certifying the operational system resources are ready and able to meet the required availability profiles and support air traffic services. The NAS systems, including ERAM, are largely custom developed, proprietary and expensive. This makes data sharing and integration across major NAS components very difficult, expensive and inefficient. To evolve the NextGen capabilities the services and data in these systems must be exposed and connected securely to the authorized systems.

3 System Wide Information Management (SWIM)

As discussed above, today's National Airspace System (NAS) comprises systems that have been developed over time for specific purposes. In general, they are connected discretely to support specific data and information exchange needs. Each of these interfaces is custom designed, developed, managed, and maintained individually at a significant cost to the FAA. The NextGen relies upon a new decision construct that will bring more data, systems, customers, and service providers into the process. Data will be needed at more places, for more purposes, in a timely manner, and in common formats and structures to ensure consistent use. The resulting decisions must be distributed to the affected parties efficiently and reliably to support timely execution.

In the past, the state of the art for connecting two systems required a fixed network connection and custom, point-to-point, application-level data interfaces. Current NAS operations depend upon these legacy information systems, but much of their data remains inaccessible to the rest of the NAS. This is an impediment to efficiency, impairs situational awareness across the NAS, and prevents optimization of ATC services. The FAA has identified a need to reduce the high degree of interdependence among systems and move away from the proliferation of unique, point-to-point application interfaces. Therefore, SWIM as envisioned will provide an open, flexible, and secure information management architecture for sharing NAS data and enabling increased common situational awareness and improved NAS agility. SWIM will implement commercial off-the-shelf hardware and software to reduce development cost and time as well as support a loosely coupled service-oriented architecture that allows for easier integration of new connections, services and systems.

The mission of the SWIM Program is to realize greater information sharing among NAS stakeholders, both FAA and non-FAA users, to support the NextGen concept of operations. This includes, but is not limited to, aeronautical information, flight data, traffic flow management data, surveillance, and weather information. To achieve this mission, SWIM's strategy is to migrate and connect NAS applications into a distributed processing environment focused on information sharing. The larger mission requires these systems to be highly scalable, robust and agile. These open architecture principles are expected to provide value by reducing costs, reducing risks, enabling new services, and extending existing services to facilitate highly coordinated NAS wide operations.

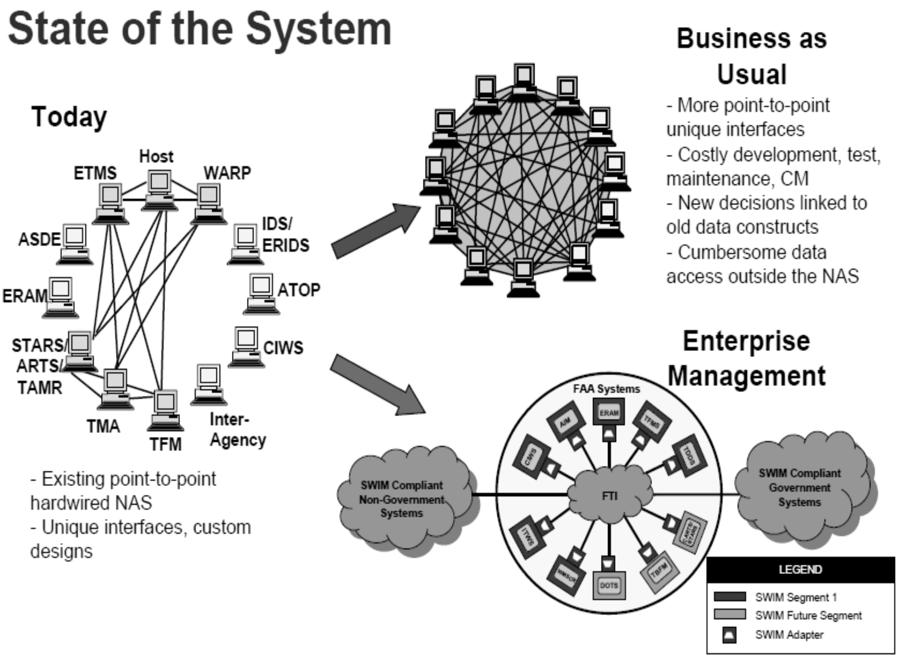

State of the System
Today
Host
ETMS
WARP
ASDE
IDS/ ERIDS
ERAM
ATOP
STARS/ ARTS/ TAMR
CIWS
TMA
TFM
Inter- Agency
- Existing point-to-point hardwired NAS
- Unique interfaces, custom designs
Business as Usual
- More point-to-point unique interfaces
- Costly development, test, maintenance, CM
- New decisions linked to old data constructs
- Cumbersome data access outside the NAS
Enterprise Management
FAA Systems
AIM
ERAM
TFMS
CIWS
TDDS
FTI
ITWS
WMSCR
DOTS
TBFM
CARTS/ STARS
SWIM Compliant Non-Government Systems
SWIM Compliant Government Systems
LEGEND
SWIM Segment 1
SWIM Future Segment
SWIM Adapter

Fig. 2. Point-to-Point Interfaces Transformed by the SWIM

As indicated in Figure 2, currently numerous systems in the NAS communicate with each other using unique, custom point to point connections.

Specifically, SWIM has been under development using Service Oriented Architecture (SOA) principles in designing and specifying NAS Air Traffic Management (ATM) services. Key functional elements of the SWIM SOA are the SWIM Core Capabilities and SWIM Business services. The initial SWIM Core Capabilities are: Interface Management (Interface Specification, Interface Discovery, Schema Management, Service Invocation, SWIM Basic Profile), Messaging (reliable messaging), Security (authentication and authorization), and Enterprise Service Management (service monitoring and configuration). The SWIM Program Office specifies the standards for all SWIM Core Capabilities; however, implementation is delegated to the domain and major program offices called SWIM Implementing Programs (SIPS).

The SWIM Service Container [5] is an infrastructure component that will provide many of the needed support (hosting) capabilities. It will relieve the SWIM Implementing Program (SIP) implementers of some of the housekeeping tasks required in service delivery. It should provide connections to data, to messaging services, and to authentication/authorization services (i.e., the SWIM Core Capabilities), as well as provide logging, error handling, and other support functions. SWIM will use the Service Container as a means for achieving consistency and interoperability among diverse NAS programs and operating elements, in the absence of a centralized service infrastructure. The Service Container will be comprised of an existing product (commercial off-the-shelf (COTS) or open source software) with components configured to support SWIM specific goals such as interoperability, extensibility, and portability. The Service Container also serves as the point of enforcement for enterprise-wide SWIM policies and accelerates service implementation by providing standard, reusable common infrastructure elements. It provides access to enterprise resources and simplifies the service design and development process.

Therefore, SWIM in other words is an IT infrastructure program that will operate in the background to provide data to authorized users. SWIM will implement a Service-Oriented Architecture (SOA) in the NAS and will allow the FAA to create new system interfaces more quickly and cheaper than is possible today. It will facilitate the data-sharing that is required for NextGen. SWIM is not a set of avionics equipment or a substitute for NAS modernization programs or to replace the FAA Telecomm Infrastructure (FTI). As a matter of fact, SWIM will enable increased common situational awareness and improved NAS agility to deliver the right information to the right place at the right time.

Progress® FUSE™ Services [6], which is a middleware providing a web service stack supporting an Enterprise Service Bus (ESB) messaging framework, was selected and is now in use in several SWIM development projects. FUSE™ is an open source SOAP and REST web services platform based on Apache CXF for use in enterprise IT organizations. It is productized and supported by the FuseSource group at Progress Software. FUSE™ Services Framework service-enables new and existing systems for use in enterprise SOA infrastructure. It is a pluggable, small-footprint engine that creates high performance, secure and robust services using front-end programming APIs like JAX-WS and JAX-RS. It supports multiple transports and bindings and is extensible so developers can add bindings for additional message formats so all systems can work together without having to communicate through a centralized server. FUSE™ Services Framework is part of a family of SOA infrastructure tools that include the FUSE™ ESB (based on Apache ServiceMix), FUSE™ Message Broker and FUSE™ Mediation Router (based on Apache Camel).

Function	**FUSE™ Component**	**Open Source Apache Project**
Web Services (WS) Stack	FUSE™ Services Framework	Apache CXF
JMS Messaging	FUSE™ Message	Apache ActiveMQ
Integration Patterns (EIP)	FUSE™ Mediation Router	Apache Camel
Service Bus (ESB), OSGi Service Container	FUSE™ ESB	Apache Service Mix

Fig. 3. FUSE™ Functions Matrix

Most importantly, the FUSE™ products can be configured in a cluster type environment to support the primary/standby configuration needed for failover to achieve higher reliabilities. Specifically, the Message Broker, of Apache ActiveMQ can coordinate Master-Slave JMS message communication between applications in redundant service containers within a processor or between two processors. For ERAM SWIM the design is to establish the redundant processors into an Application Cluster LAN with two Gigabit Ethernet cables cross-mounted between the servers. EtherChannel [7] is used to combine the two connections into a single logical channel able to withstand failure in one of the connections and support high bandwidth if required. The primary/standby state information is shared through a file system on the En Route enterprise storage servers. It required extensive learning and testing, and assistance from the Progress vendor to implement this approach, however testing so far has shown that the ActiveMQ Master-Slave broker achieves almost immediate failover after detection of the failure of a primary component. As will be seen, extensive effort went into improving detection of some failures and failure modes.

Most of the SWIM programs so far, including ERAM SWIM, have selected Eclipse, a free and open source (FOSS), web service-friendly integrated development environment (IDE). The Eclipse Project (now Eclipse Foundation) was started by IBM in November 2001 with many other vendors and established in January 2004 as an independent, not-for-profit corporation, to act as the guardian for the Eclipse community. The Eclipse platform is a universal tool platform. It can handle most any type of resource (Ada, Java, C / C++, XML, HTML, even doc or text files.). The Eclipse architecture supports extensions (developed as plug-ins) that teach the platform how to work with these different kinds of resources. Hibachi Ada Development Tools project (ADT), is the Ada plug-in for Eclipse that provides extensive functionality for Ada developers, needed for the ERAM interface. For ERAM SWIM, Eclipse is supported on the developers' Windows workstations using an existing plug-in which allows Eclipse to communicate with ERAM's AIX development machines.

Maven serves a similar purpose as the Apache Ant tool which is more or less the imake equivalent for these environments. Maven (FOSS) uses a construct known as a *Project Object Model* (POM) to describe the software project being built, its dependencies on other external modules and components, and the build order. It comes with pre-defined targets for performing certain well defined tasks such as compilation of code and its packaging. Maven is also architected to support plug-ins allowing it to use any application using standard inputs. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories. Maven provides built-in support for retrieving files from the Maven Central Repository [8] and other Maven repositories, and can upload artifacts to specific repositories after a successful build. A local cache of downloaded artifacts acts as the primary means of synchronizing the output of projects on a local system.

3.1 ERAM SWIM Architecture

The ERAM SWIM initially is developing capabilities to use this SOA web service paradigm to increase NAS wide access to flight data in support of new traffic management capabilities. As depicted in Figure 4 below, the SWIM Interface Subsystem (SIS) uses the FUSE™ ESB and infrastructure components, running on an AIX platform, to implement an OSGi service container that manifests the physical end-points and service interface. It is instantiated as a single JVM process on each of two SWIM servers which allows for load-balancing, fault-tolerance, and software maintenance. The service container also handles the inflow and outflow of management data, such as auditing, configuration, error handling, and service status. In this first phase, now in test and preparing for deployment at a single ERAM site, each SIS consists of a single service hosted in the OSGi container which will automate flight plan changes from the Traffic Flow Management System (TFMS) to mitigate severe weather or other air traffic flow initiatives [9]. TFMS will request and consume pre-departure flight plans impacted by the initiative, create a reroute amendment, and submit that back to ERAM as an update request via the SWIM Interface Service (SIS). FUSE™ Mediation router (CAMEL), Message Broker (JMS) and Services Framework (CXF) provide for SOAP/JMS messages, handle the transport mechanisms, and supports automatic failover. SIS sends the update requests (amendments) to the SWIM adapter in the appropriate ERAM using this "back-end" interface.

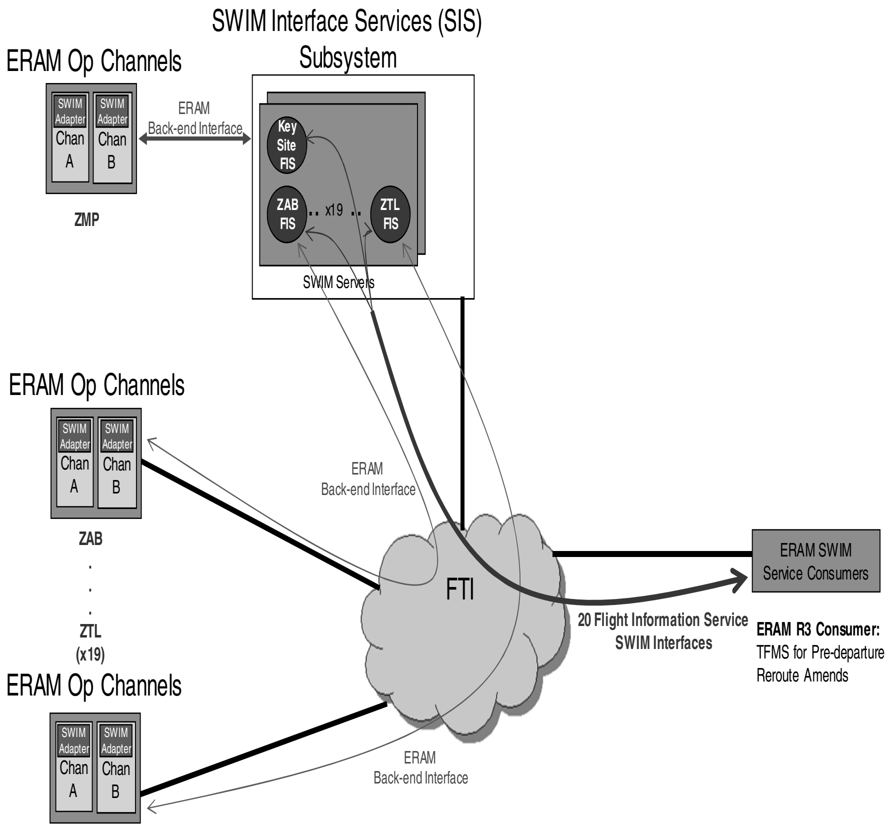


Fig. 4. High Level Physical Architecture for initial ERAM SWIM Services [9]

The SWIM adapter is a client proxy of the ERAM flight data manager (FDM) which "owns" the ERAM flight object data store. ERAM SWIM Service Consumers

(TFMS here) will communicate with the ERAM SWIM Flight Information Service (FIS) using the FAA's Telecommunications Infrastructure (FTI) routed over the existing En Route Communications Gateway (ECG). Each ERAM has an ERAM SWIM adapter component, providing the mechanism for FIS consumers to send or receive data from the ERAM core. To minimize impact on ERAM performance, the SWIM adapter - built on the ERAM Publisher FrameWork [10] - maintains a local mirror of all relevant flight data. The SWIM adapter functions as a wrapper translating the internal ERAM data representation to XML messages, publishing an XML flight object to the SWIM application database, and converting XML flight update requests back into internal ERAM binary format. Planned near-term capabilities include the ability to support terminals obtaining pre-departure flight data for local aircraft for use in uploading clearance information by DataComm in lieu of current voice procedures. Numerous other flight data "consumer" applications are in planning along with legacy replacement programs that would take advantage of the new Flight Information Service (FIS) to move away from their current custom, single-purpose interfaces. Still other services are envisioned for weather data, airspace, route and restriction status information, radar track information, etc. ERAM also becomes a consumer for other NAS system data such as Pilot Reports (PIREPs), other TFMS flow data, and ATC Tower, and Runway Visual Range (RVR) data.

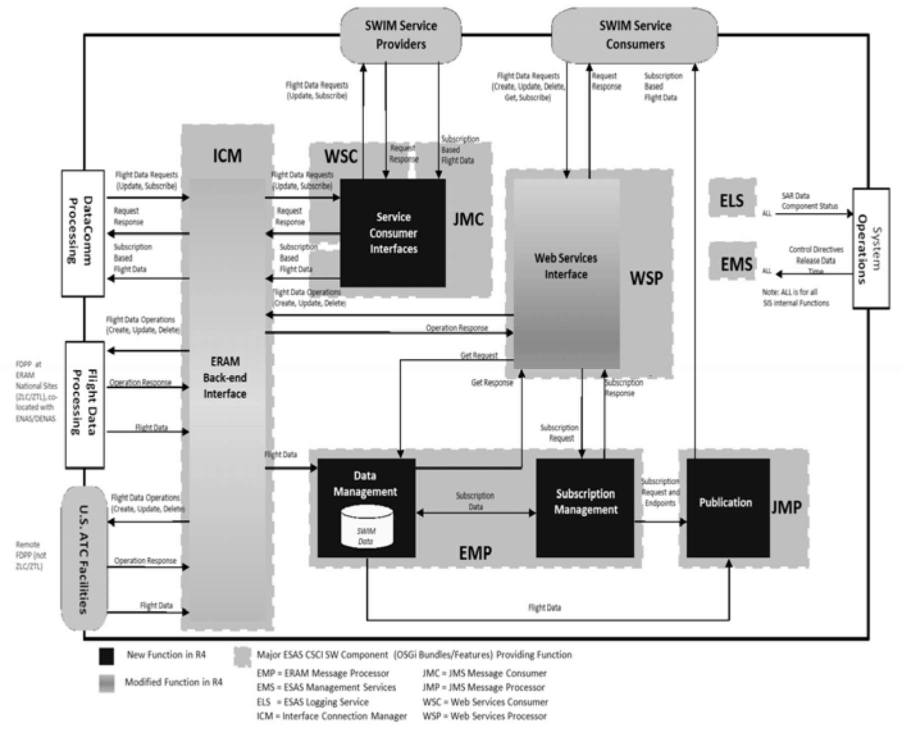

SWIM Service Providers
SWIM Service Consumers
Flight Data Requests (Update, Subscribe)
Flight Data Requests (Create, Update, Delete, Get, Subscribe)
Request Response
Subscription Based Flight Data
ICM
WSC
JMC
Service Consumer Interfaces
Web Services Interface
WSP
ELS
EMS
SAR Data Component Status
Control Directives Release Data Time
ALL
Note: ALL is for all SIS internal Functions
System Operations
DataComm Processing
Flight Data Processing
U.S. ATC Facilities
FDPP at ERAM National Sites (ZLC/ZTL), co-located with ENAS/DENAS
Remote FDPP (not ZLC/ZTL)
Flight Data Operations (Create, Update, Delete)
Operation Response
Flight Data
ERAM Back-end Interface
Get Request
Get Response
Subscription Request
Subscription Response
Data Management
SWIM Data
Subscription Data
Subscription Management
Subscription Request and Endpoints
Publication
EMP
JMP
New Function in R4
Modified Function in R4
Major ESAS CSCI SW Component (OSGi Bundles/Features) Providing Function
EMP = ERAM Message Processor
EMS = ESAS Management Services
ELS = ESAS Logging Service
ICM = Interface Connection Manager
JMC = JMS Message Consumer
JMP = JMS Message Processor
WSC = Web Services Consumer
WSP = Web Services Processor

Fig. 5. Software Architecture of the ERAM-SWIM Components

The software architecture of the ERAM SWIM Application Services (ESAS) CSCI is represented in Figure 5. In addition to the update, create, delete of the initial SIS (light blue), now depicted as the ERAM Message Processor (EMP), a subscribe and publish flight data service is provided using a WSDL-based FIS interface in the Web Services Processor (WSP) component and as a JMS Message Processor (JMP) replying to subscriber requests and endpoints. WSP exposes multiple service endpoints to support load balancing and fail-over and performs authentication and authorization of service consumers.

In addition, the future ERAM service consumer interfaces are depicted as the consumer side of the WSDL-based web service (WSC) and JMS subscription service (JMC). The context of the ERAM back end interface is extended to add DataComm processing for context management and log-on services.

3.2 ERAM SWIM RMA Design Challenges

Figure 5 also explicitly depicts the System Analysis and Recording (SAR) logging service (ELS), monitor and control and software release management interfaces (EMS) providing the types of features described earlier for the ERAM FlightDeck® middleware. ERAM SWIM service element status and performance will be aggregated on the ERAM M&C consoles using the same iconography as other ERAM elements. Control commands share the same GUI and command line semantics. However, under the covers, the ERAM middleware is not there and most of these capabilities differ not only in implementation, but also their semantics and behavior which leads to a significant operator training burden and significant user acceptability risk. Even with the more limited context of the initial single service SWIM SIS a number of difficulties arose attempting to provide for similar RMA features and providing for essential service critical performance. The next phases leading to the end state architecture must be scalable to efficiency critical if, as expected, ERAM SWIM services in the future are required to meet that level (e.g. Flight Data Input Output replacement). The balance of this paper will discuss those challenges including fault detection performance, failure recovery behavior, database fault tolerance, CAS (commercially available software) and FOSS limitations and problems that even extend to the development tools.

Certainly most critical to the FAA requirements and expectations is understanding the reliability and availability of FUSE™ and other selected CAS/FOSS. ERAM's custom developed environment is carefully designed using allocated failure thresholds on each component, however vendors and licensers of many software products generally do not have or are unwilling to make failure mode and MTBF data available to support decision making. The ERAM SWIM development team therefore had to collect whatever industry available data they could, consult with company tech support personnel, and apply industry norms based on software language and size to calculate an expected failure rates for the COTS products. Table 2 shows the results of that analysis for several key products selected for the SIS software and SWIM servers.

Table 1. Calculated Failure Rates for Key ERAM SWIM CAS Products

Product	Failure Rate
FUSE™	4.785 E-06
TECTIA [11]	4.027 E-05
AIX (Includes Power VM)	1.240 E-06
ORACLE	1.376 E-06

Clearly FUSE™ presented a risk in achieving high reliability in the long run even with robust redundancy and fail-over from the Master-Slave mechanisms. Perhaps even more important to measure is the mean time to repair (MTTR) sometimes called mean time to recover. MTTR begins at the failure and includes the time to detect the failure, restart or switch processing resources, and reach steady state on the restarted application. In ERAM developed software, particularly software supporting the highest criticality services, recovery must usually complete on average within 5 to 6 seconds. To support this performance requirement, the middleware can detect the overwhelming majority of failures in 1 to 2 ms by using an OS process state change handler, complete failure handling and notify the standby address space well below 100 to 200 ms, and determine the scope and recovery action to begin restarting the failed component under 500 ms from detection, and have the restarted application redundancy restored with a complete restart of the failed element to become the standby in 2 to 5 seconds. The very rare or unlikely software hang or loop condition won't achieve this, but is detectable by a configurable heartbeat mechanism, usually set at about 6 seconds. The low probability of occurrence allows the aggregate availability requirements to be met.

However, with the FUSE™ instances on ERAM SWIM, the middleware was not there to implement these features partly to save cost and schedule, but also because the initial SWIM service simply did not justify the need for this robustness. After all, in the final analysis, SWIM is a message broker and router; for most uses it will suffice to have the consuming client set a timer on its requests and retry on another endpoint and/or the redundant service container so that the service is provided. ERAM SWIM initially therefore specifies the client/consumer retry while failure detection not detected by the ActiveMQ broker relies on a developed polling mechanism using JMX to monitor the JVM PID. The ERAM SWIM developer had to unexpectedly port and extend an ERAM low-level function to facilitate the monitoring of the poll mechanism. The resulting time for ESAS to be able to process web service requests after a single component failure on one SWIM server was established as 30 seconds, a time that will likely need to be improved upon in the future. To detect and recover broker hang conditions would be much higher and there is concern some unknown failure modes may not detectable by the system at all.

Therefore, ERAM SWIM developers had to incorporate product changes under the FUSE™/Apache license, in several processing threads. Essentially these changes provided FUSE™ components the ability to dispatch events to the monitor – basically heartbeats. The FUSE™ documentation and normal product tech support, however, was not sufficient to understand where all these beating hearts needed to go and it became necessary to engage Progress Software (Fuse™ vendor) architect level engineers as private consultants to the developers to facilitate these and other non-fatal alert features. Another key difference is that ERAM is able to make granular recovery decisions to minimize the scope of the restart, but ERAM SWIM restarts the entire service container on a node along with all of its bundles and endpoints in the event of any ESAS or CAS failure. This increases the time to restore redundant services by a small order of magnitude.

Other important differences exist in the area of commanding application resources from the M&C position. FlightDeck® has several specifically designed directives and receives an ACK from the application targeted. The system manager on SWIM relies on simple SNMP requests and has no ACK feature, so that manual configuration, maintenance, or recovery commanded actions are less certain. Likewise with status reporting, ERAM applications collect and report detailed health information on the applications resources and performance. SWIM relies on occasional polls to the major products and, ultimately SWIM application health is derived from the interfacing ERAM application which is running on FlightDeck® by monitoring the state of its communication with ESAS. In fact, all status collection and reporting for the available SWIM software elements is sent to the ERAM state service via the interfacing ERAM application which then sends the updates to the ERAM M&C for operator display and event recording.

Perhaps the biggest and most surprising challenge faced in architecture and design of ERAM SWIM has been in providing for a robust, persistent, fault-tolerant data store. Originally, SWIM expected to implement an Oracle cluster (Oracle RAC) in particular to support a publication/subscriber service prototype. Several technical problems were encountered and largely overcome. However, the procedures for reconfiguring the cluster after a new software cutover was error-prone, not consistent with current cutover design, and required administrator skills to correct if performed improperly. In addition, after a network failure in the prototype configuration, it was learned Oracle RAC will initiate a kernel panic to protect itself from shared database corruption. SWIM servers, unlike ERAM FlightDeck®, do not have capabilities to handle and recover from such a panic. Several alternative database options were evaluated which had their own limitations that made them risky for the En Route environment. The point design is now GPFS shared filed system implemented on the En Route Enterprise storage subsystem which is a redundant pair of units. ActiveMQ depends on the shared file system for queue/topic persistence and to support fail-over. While prototyping it was learned that AIX mirroring with GPFS will lead to a file system un-mount for lack of quorum when one of the two storage units is unavailable. An unplanned application had to be written to perform the mirroring instead of using the GPFS mirroring facilities which still leaves a tiny, but non zero dual failure risk.

3.3 System Safety and Security Considerations

ERAM SWIM makes some major changes which affect or relate to security. These include moving from a single service consumer to multiple consumers, expanding from a single key site to multiple ERAM SWIM sites and adding JMS inbound and outbound traffic in addition to web traffic. The inbound JMS and web traffic is externally initiated whereas the ERAM security architecture had ERAM initiating all physical and data interfaces. Key components of the security approach include:

- Transport-level encryption is used (both inbound and outbound). This is effectively HTTP/SSL for web traffic, JMS/SSL for publication traffic. The SSL exchange is two-way.
- Certificate-based authentication using x.509 credentials will be necessary to identify and authenticate users.
- Schema validation of all incoming messages
- Proxy web traffic, proxy JMS traffic.

Additionally, the credential-based access control is implemented on a per-operation, per-data element basis.

Pre-departure DataComm is architected to leverage the FIS in phase 2 of ERAM SWIM. DataComm applications are subject to safety assurance requirements of DO278 [12]. After much agency deliberation the requirement was established as AL3, but even this modest assurance level requirement requires that all developed, commercial, and FOSS products be developed to that level or the safety risk mitigated. The key challenge here is to assure flight data and message integrity in an xml and web service environment. The DataComm architecture uses ERAM SWIM as path or transport between DataComm application functionality that resides in ERAM and DataComm applications that reside on the consumer side. Since FIS does not need to do any actual flight data processing it was determined that the integrity risk could be mitigated and thus assured to an acceptable level, by implementing high integrity encryption around certain flight data. FIS will perform selection of flights and filtering (masks) of the fields of a flight object needed by each subscriber, but is constructed using the "untrusted" (per DO278) FUSE™ products. To resolve this problem, the (trusted) ERAM SWIM adapter computes robust checksums on the key protected data fields and includes the checksum as a field attribute in the SWIM XML message. FIS passes it along in the filtered data it publishes to the authorized consumer. Any consumer that requires high-integrity (such as DataComm) is then responsible to retain the attribute through its consumer web service implementation and verify field level integrity in a trusted application on the consumer side of the interface. In reality, this data integrity issue is not unique to the SWIM environment; ERAM will also be modified similarly to protect flight data integrity from CAS in the ERAM environment including AIX and the Ada Runtime. However, this significantly impacts CPU and network utilization of both ERAM and ERAM SWIM. Early prototyping verifies the increases are a tolerable tradeoff to achieve the assurance objectives while using the SWIM CAS and FOSS products.

As mentioned earlier, there is even an RMA impact from the associated development tools. Software maintainability and supportability depend on available tools. The FAA does not wish to become its own product support organization. In selecting this large number of CAS and FOSS products for ERAM SWIM, it was necessary to consider the cost and difficulty if these products later had to be replaced, and for more critical components whether the vendor or consortium managing them was capable and committed for the medium to long term. The Hibachi ADT plug in for Eclipse IDE fortunately was not actually used in this project because Ada software in the SWIM programs represented enhancements and modification of actual ERAM "owned" source code, not new features for the SWIM service environment. Therefore, all ERAM mods and additions followed the ERAM development process and used the baselined development environment. As of January 2011, Hibachi ADT is an archived project on the Eclipse Foundation web site. There appear to be no current plans to support and promote those capabilities so that had the program incorporated it would now have to evaluate its' risk and the cost to move away from it. In this CAS/FOSS collaborative environment there clearly are significant supportability risks to be managed.

4 Conclusions

ERAM SWIM developers successfully implemented initial ERAM SWIM supporting create, update, delete using Java based web services to interface with ERAM to implement a robust Flight Information Service that will be used to automate previously manual reroute procedures. Many more opportunities exist to exploit this single interface to ERAM to support sharing across the NAS. The service container will allow these services to share the interface even while maintaining independence and allowing decoupling of the internal implementation. A number of challenges were overcome; chief among them is overcoming the dearth of product documentation and support for developing high availability web services. Some of the difficulties encountered result from the institutionalization of strategies and procedures in the FAA En Route domain. The learning curve for developing high reliability web services is steep and requires significant experimenting, consulting and expert assistance, as well as analyzing the source code of the FOSS products for failure mode and implementation details not documented to the same standards that custom developed code is expected to meet. Productivity with FOSS is very high because the key functional capabilities exist already and the developers' job is to configure and harness those capabilities to meet the requirements of the particular service to be implemented.

Acknowledgements. The author wishes to thank Dr. Alok Srivastava, Technical Fellow of TASC Inc. for the initiative and the discussions, and FAA's ERAM and ERAM-SWIM Program Offices for their support. Assistance and support from the ERAM and ERAM-SWIM development team of Lockheed Martin Corporation is also highly appreciated.

References

1. FAA Next Generation Air Transportation System (NextGen), `http://www.faa.gov/nextgen/`
2. NAS SR-1000A, National Airspace System, System Requirements Specification, `http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/atc_facilities/cm/dcc/` also available freely on the other websites

3. Keynote presentation - Use of Ada in Lockheed Martin for Air Traffic Management and Beyond by Judith Klein of Lockheed Martin. In: ACM SIGAda 2006 International Conference, Albuquerque, New Mexico, USA (2006)
4. Keynote presentation - An Ada Retrospective: Developing Large, Mature, and Reliable Systems by Richard Schmidt of Lockheed Martin Information Systems & Global Services. In: ACM SIGAda 2009 International Conference, Tampa Bay, Florida, USA (2009)
5. FAA SWIM Program Office Reference to the SWIM Container Infrastructure, `http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/atc_comms_services/swim/documentation/media/newsletters/SWIMNewsletter_EditionOneweb.pdf`
6. FUSE^{TM} Source Progress Software Company, `http://FUSESource.com/`
7. CISCO Ether Channel, `http://www.cisco.com`
8. Maven – An Apache software project management and comprehension tool, `http://maven.apache.org/`
9. Goldstein, S., Indigo Arc LLC, Rockville Maryland USA, High Level Physical Architecture for initial ERAM SWIM Services White Paper (2009)
10. Klein, J., Sotirovski, D.: The Publisher Framework. Ada User Journal 27(4) (December 2006)
11. TECTIA Information Security COTS Solutions, `http://www.tectia.com/en.iw3`
12. DO 278 A Radio Technical Commission for Aeronautics (RTCA) Inc. developed Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance, `http://www.rtca.org/`