Bringing the Aircraft System Health Management Tool to Life Using the Informatica Tool Suite

Glenn Peters[1], Mark Mosher[2], and Chris Carlson[3]

[1]*UTC Aerospace Systems, San Diego, CA, 92123, USA*
glenn.peters@hs.utc.com

[2]*Impact Technologies, Rochester, NY, 14623, USA*
mark.mosher@impact-tek.com

[3]*Informatica Corporation, Redwood City, CA, 94063, USA*
ccarlson@informatica.com

ABSTRACT

Earlier this year, UTC Aerospace Systems introduced the Aircraft System Health Management (ASHM) Tool, a web application that takes in Aircraft Condition Monitoring Function (ACMF) reports for selected subsystems and components of an aircraft platform, parses and processes the reported parameters against thresholds and computes estimated or expected values for some key parameters, and serves the report data and the processed results as part of a fleet view available to airline and maintenance users.

The ASHM application uses Informatica PowerCenter to parse and store incoming report data and Informatica RulePoint to apply alert rules and analytic processing to the report data as it is persisted to the ASHM database. This paper describes how UTC Aerospace Systems has leveraged a commercial off-the-shelf tool suite rather than continue to build custom components for the ASHM architecture, with the goals of achieving a short development cycle, robust transaction processing, and scalability to other aircraft systems and other aircraft platforms.

Instead of building tools from scratch that would need to be reworked as the application scales, a set of scalable tools that suit the task at hand and in the future were selected. The full suite of tools, beyond what has been implemented so far, appears to provide capability to address data integrity, reliability and performance as the application grows.

1. BACKGROUND

UTC Aerospace Systems is a supplier of aircraft systems and power, controls and sensing systems for platforms that include commercial and military aircraft, including helicopters, and international space programs. They are a division of UTC Propulsion and Aerospace Systems, which also includes Pratt & Whitney (engines). Their aircraft systems and services include actuation systems, aero structures, air management systems, interiors, landing gear, propeller systems, and wheels and brakes. Their power controls and sensing systems include electric systems, engine components, engine and control systems, fire protection system, intelligence, surveillance and reconnaissance systems, sensors and integrated systems, and space systems.

Historically, UTC Aerospace Systems has not had a proactive capability in place to predict when a critical or important issue might occur for systems other than the APU. Furthermore, what was previously a once-per-day sample of sensor readings for the APU has been moving rapidly to a real-time, over-the-air paradigm.

2. MOTIVATION

Pratt & Whitney AeroPower (formerly Hamilton Sundstrand Power Systems) has been monitoring the A320 and A380 Fleets of many airline customers for over 15 years. Essential requirements developed during this time were:

- The ability to easily accommodate new versions of aircraft reports.
- An alerting engine that is scalable and that allows for the addition of alert processes against any number of parameters.
- A "modern" web based user interface that allows end users to focus on aircraft within the fleet

Informatica Corporation develops and markets data integration software tools that have allowed UTC Aerospace Systems to:

a) Develop positional parsers to quickly react to different report versions
b) Develop an alerting engine for parameters, events and trends
c) Develop a user interface to display alerts generated by the alerting engine.

3. Event Driven Architecture

3.1. Events

Events represent any change(s) in state throughout an enterprise, from the lowest, narrowest level to the highest and broadest.

Events can be sensor reads, social media postings, location changes, financial transactions, database operations and file arrivals, for example. Events are effective triggers for transitions between states.

Missing events are themselves events because something that was expected to happen did not happen. These special events are nonetheless important because they signal transition to anomaly or exception states. For example, a business process step that is not executed by a particular time, an event not received an expected time after or before another event, or no readings, input or feedback received for a specific period of time.

3.2. Event Processing

Event processing is a solution approach that deals with making sense out of events from one or more sources. Events may be combined with other sources of data to define "situations of interest". This provides automatic monitoring of changes in state, reduced decision latency, consistent application of business rules, self-service, and knowledge capture. Figure 1 shows how event processing flows from input event sources to output actions.

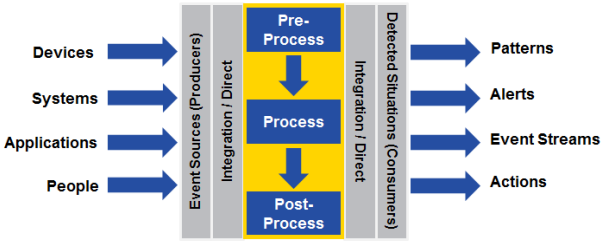

Figure 1. Event Processing Flow

3.3. Informatica RulePoint

RulePoint is Informatica's event processing solution, and is designed to be deployed as a standalone solution or as part of a broader event-based architecture. For background, RulePoint was developed by Agent Logic, which was acquired by Informatica in 2009, in a move to fill a gap in Informatica's overall data integration suite of products.

RulePoint is focused on the end user / data analyst, and provides a rule-based approach to event processing, and that includes self-service, event-condition-action and temporal/geo-spatial rule handling.

An event-driven architecture is a special type of data driven architecture in which changes in state drive the activity within an environment. Put simply, events drive the execution of logic, or perhaps more correctly, events feed the application of rules and actions based on the outcome of the applications of those rules within the architecture.

4. Application of Informatica Tools to ASHM

Figure 2 shows how the general categories of data, analytics and event processing are handled by the specific Informatica tools that are being used for the ASHM project.

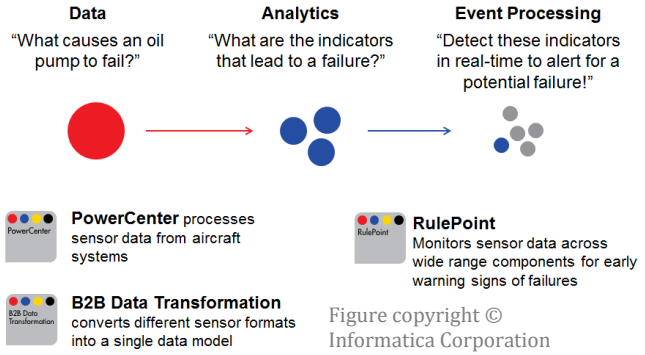

Figure 2. Data Analytics with Event Processing

4.1. High Level Solution Architecture

Figure 3 shows the ASHM architecture at a high level. Standard ATA reports come in for various subsystems of multiple aircraft, are parsed by Informatica PowerCenter using parser templates created for each report type, stored in the application database, acquired and processed by RulePoint for alerts and sent through engineering models for computation of other 'estimated' or 'corrected' parameter values.

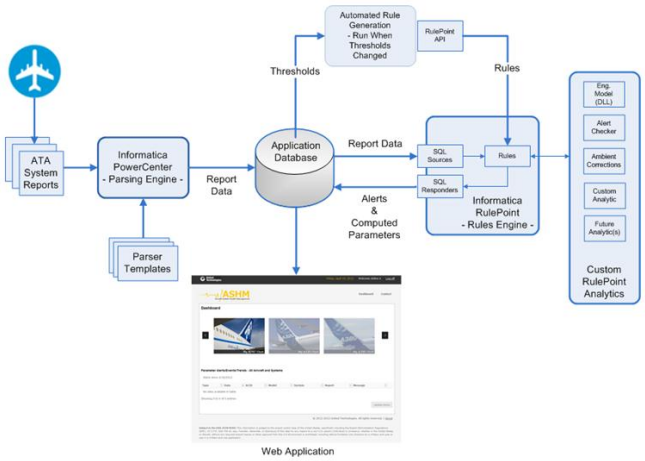

Figure 3. ASHM Architecture

4.2. Data Flow from Aircraft Reports to Database

Figure 4 shows the data flow from incoming aircraft reports to the database. At the top, different report *types* arrive and are bucketed by type into directories for further processing. Each subsystem has one or more report types, and as ASHM grows to process more systems of each aircraft platform, and adds more aircraft platforms, the number of report types will grow accordingly, as indicated by the arrows to the lower right of the diagram.

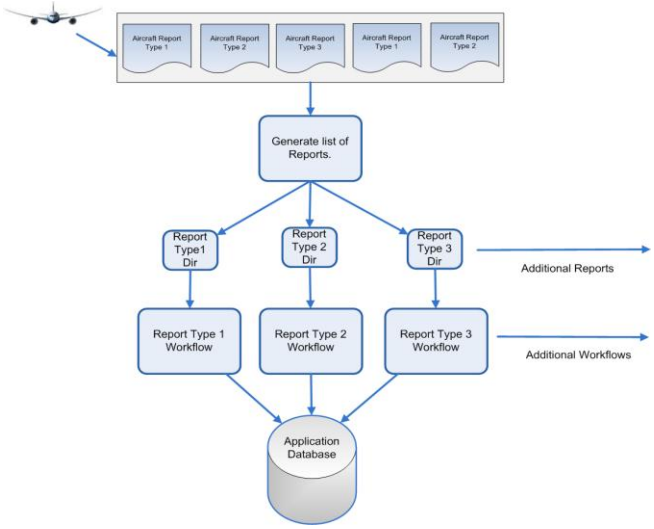

Figure 4. Data Flow – Aircraft Reports to Database

4.3. First Level Parsing to Categorize Reports

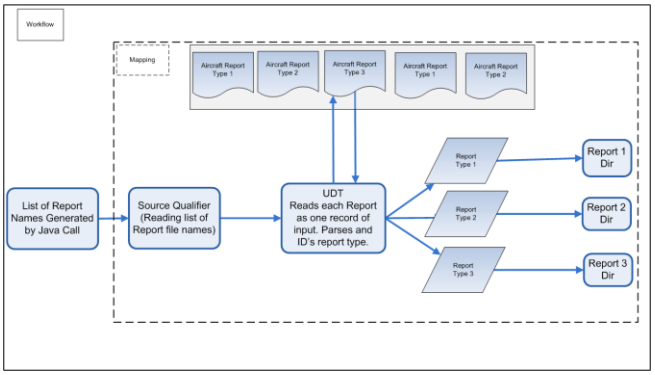

Figure 5. Report Categorization

First, as shown in Figure 5, the reports are parsed at a high level to determine the specific type of report, e.g. subsystem, report type, variant of that report type, and based on that a determination is made and action taken to move that report to the proper staging directory.

4.4. Second Level Parsing to Harvest Report Data

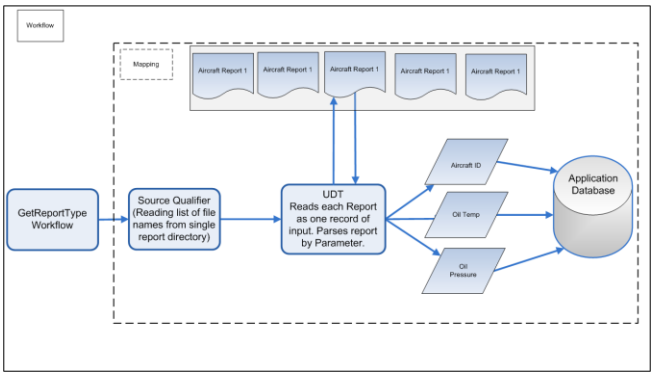

Figure 6. Report Data Transformation

Secondly, as shown in Figure 6, the Data Transformation agent reads each report as one record of input, and parses the parameters from that report. It stores each parameter as part a unique record for that report in the application database.

4.5. RulePoint Workflow in ASHM

Figure 7 shows the flow of data through RulePoint, by report type originating with a *SQL Source* that acquires parameter instance data from the ASHM database and pushes it into a RulePoint *Topic*. A *Rule* references one or more topics and may use data from those topics to 1) determine anomalous conditions, e.g. value out of range, 2) compute new values based on those parameters, 3) send those computed values or detected conditions to a *Responder* that is responsible for storing new data back to the same ASHM database. RulePoint has a wide variety of responders that can send emails, send data to other systems, write to files, et cetera.

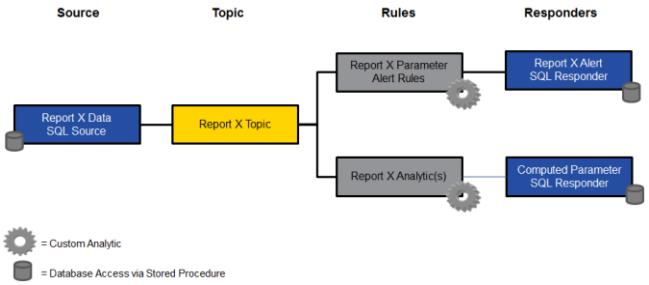

Figure 7. RulePoint Workflow

4.6. Automated Rule Generation

For the ASHM project automatic alert rule generation based on thresholds defined in the database was employed (see Figure 8). This tool uses the RulePoint Java API Adapter to 1) connect to the development RulePoint instance, 2) remove all previously generated (as opposed to hand entered) rules, and 3) generate a new set of rules based on those thresholds. Currently there are 250+ alert rules, with more being added for each new report type. These thresholds are originally defined but are inspected and assessed at the outset and as needed to reduce the occurrence of false alarms.

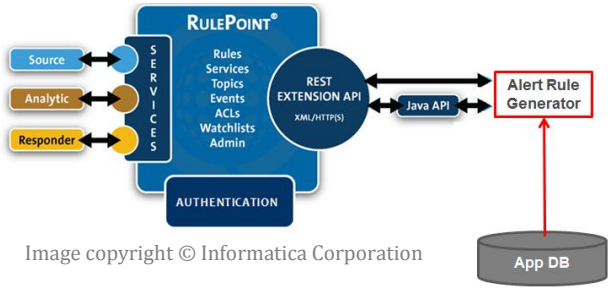

Figure 8. Alert Rule Generation

4.7. Parameter Alerts

The ASHM application checks for out of range "alert" conditions on selected incoming report parameters, looking for warning or alarm conditions that are higher or lower than expected under normal operating conditions. Each "alertable" parameter has its own set of thresholds defined in the database for low and high warning and alarms, for example 0, 1, 99, and 100, for low alarm, low warning, high warning and high alarm thresholds, which would trigger a low alarm condition if the parameter value is at or below 0, for a low warning if the parameter values is above 0 but at or below 1, and likewise a high alarm if the parameter value is at or above 100, and a high warn if the parameter values is at or above 99 but less than 100.

There are also mechanisms in place to define two additional criteria which are when the thresholds are to be ignored, say when some (the same or another) parameter's value meets a certain conditional relationship with a fixed value, e.g. <= some value, = some value, or >= some value.

The parameter alert rules store alert conditions that are detected back to the database, where they are used to display those anomalous conditions to the end user in the web application.

4.8. Analytics

In RulePoint, an *analytic* is a plugin that can be invoked from a rule, and in ASHM a *health (or diagnostic) analytic* is an engineering model of a component or corrections for ambient conditions applied to a set of input parameters. These analytics are used to compute values that would be expected from a normal running system, and these values can be compared against actual values, and also can be processed against thresholds defined in the database, if present.

4.9. Web Application

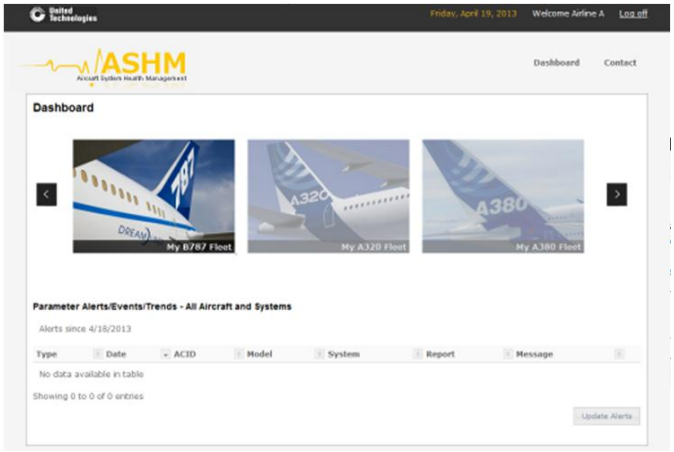

Figure 9. ASHM Dashboard

5. LESSONS LEARNED AND BEST PRACTICES

- By design, the overall application architecture and implementation is heavily dependent on the database architecture and support. This dependence makes it possible to make minor changes that support end-to-end Database to RulePoint to Database functional testing, which are critical to the integrity and proper operation of the tool.

- The integration of the general commercial off-the-shelf tools purpose-built for applications such as ASHM are viewed critical to its continued success. There are other tools in the vendor offering that can be brought in as needed when the platform requires it as it expands in systems, numbers of reports and types of aircraft that are supported.

- Monitoring and logging of the various automation components was and continues to be important for troubleshooting and debugging development and production issues and anomalies.

- As a design goal, the architecture is generic, and so can be modified to accept and process additional types of inputs, e.g. other aircraft such as helicopters, or ground vehicles, or other high value assets.

6. CONCLUSION

The use of a suite of off-the-shelf commercial tools, whose intended design was consistent with our ASHM design goals, provided the framework for the ASHM architecture. Much of the learning about the application and integration frameworks were encountered and dealt with by the vendor, and that expertise was applied to improve the general purpose set of tools. The tools also provide options for future growth as the application scales to more platforms and systems, as ASHM makes its way towards a 'big data' service.

ACKNOWLEDGEMENT

The authors would like to thank Rhonda Walthall for her leadership and support on this project, and Rocco LaPorte for his sponsorship of this key UTC Aerospace Systems initiative.

BIOGRAPHIES

Glenn Peters is an IT designer / analyst at UTC Aerospace Systems, and has been involved with the ASHM project since its inception. He played a major role in the initial Airbus report acquisition, parsing and processing, and played a key role in defining the new architecture for the ASHM Tool that resulted in the incorporation of Informatica tools into that event-driven architecture.

Mark Mosher is a project manager at Impact Technologies, a Sikorsky Innovations company that is a world's leader in the field of complex asset Condition Based Maintenance ("CBM") and Prognostics & Health Management ("PHM"). Impact provides the most comprehensive range of products and solutions for monitoring, analyzing, detecting, isolating, and predicting the performance, health and readiness of mission critical assets on the market today.

Chris Carlson is the Director of Technology Evangelism for Informatica's Complex Event Processing business. He joined Informatica with its acquisition of Agent Logic in 2009, where he ran Product Management and new solutions development for the company's flagship RulePoint complex event processing product. Prior to joining Agent Logic, Chris was Vice President of Product Management at Lucid Security, which was later acquired by Trustwave, a provider of security and PCI DSS compliance management solutions. Prior to that Mr. Carlson held Product Management and Chief Technology positions at network security and application performance management start-ups. He was a Principal Security Consultant at Science Applications International Corporation (SAIC) and held system architecture roles at Booz Allen and Hamilton.