Automated generic integration of flight logbook data into aircraft maintenance systems*

Oliver Hunte[1], Carsten Kleiner[1], Uwe Koch[1], Arne Koschel[1], Björn Koschel[2], and Stefan Nitz[1]

1 **Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik)**
Ricklinger Stadtweg 120, 30459 Hannover, Germany
carsten.kleiner@fh-hannover.de, arne.koschel@fh-hannover.de

2 **edatasystems GmbH, Am Bugapark 60, 45899 Gelsenkirchen, Germany**
bjoern.koschel@edatasystems.de

The automated transfer of flight logbook information from aircrafts into aircraft maintenance systems leads to reduced ground and maintenance time and is thus desirable from an economical point of view. Until recently, flight logbooks have not been managed electronically in aircrafts or at least the data transfer from aircraft to ground maintenance system has been executed manually. Latest aircraft types such as the Airbus A380 or the Boeing 787 do support an electronic logbook and thus make an automated transfer possible. A generic flight logbook transfer system must deal with different data formats on the input side – due to different aircraft makes and models – as well as different, distributed aircraft maintenance systems for different airlines as aircraft operators. This article contributes the concept and top level distributed system architecture of such a generic system for automated flight log data transfer. It has been developed within a joint industry and applied research project. The architecture has already been successfully evaluated in a prototypical implementation.

1998 ACM Subject Classification C.2.4 Distributed Systems, H.2.8 Database Applications, J.2 Physical Sciences and Engineering

Keywords and phrases system integration, data mapping, XML, aerospace engineering, generic interface, configurable mapping

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.201

1 Introduction and Problem Description

Ground and maintenance time is costly for aircraft carriers, thus they try to minimize them for economical reasons. Today's still mostly manual transfer of flight logbook data from an aircraft into the operator's maintenance systems should be automated to get closer to achieving this goal. This will eventually reduce information transfer time and is also likely to be less error prone. Thus in total it should result in reduced maintenance time and cost.

A generic automated flight log data transfer system needs to support the differently structured flight log information from different aircraft types and manufacturers on one side. On the other side different aircraft maintenance systems used by different operators have to be supported. Technically, all these systems are very likely distributed, even though typically in a common network within a single organization. Moreover, fault tolerance and

* In cooperation with Lufthansa Technik AG. The project is part of the Aviation Cluster Hamburg Metropolitan Region's Leading-Edge Cluster Strategy and is sponsored by the Federal Ministry of Education and Research.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 201–204

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(transactional) persistence to prevent data loss are required. Performance demands however, are not very critical due to a limited amount of log data per flight in practical applications.

To support those requirements, a *generic* transfer system for flight logbooks into maintenance systems needs to be designed and implemented. In a joint industry and research cooperation (*Verbundprojekt*) the eLog system has been designed and prototypically implemented by the University of Applied Sciences Hannover in cooperation with Lufthansa Technik AG and edatasystems GmbH. Technically this system supports different – currently XML-based, but with heterogeneous XML schemata – versions of different aircraft flight log systems on the *input* side. On the *output* side different airline systems are supported which may have different data models. As an example a relational DBMS with tables is used, that map (almost) 1:1 to the data structures of the *output* system. The *output* system is Lufthansa's core aircraft maintenance system.

The resulting eLog system offers a generic distributed system architecture for the integration and mapping of the mentioned different XML-based flight log input data formats to different output aircraft maintenance systems. The mapping itself is flexibly configurable as well. Initial tests of the prototypical implementation already validated the practical usefulness of the concept. Only very few logbook data transfer systems exist. Those that do exist are flight operator internal and/or aircraft type and maintenance system specific.

Although concepts and approaches for application/data integration in general of course do exist ([4, 7, 6, 5, 2]) their application to flight logbook data is a novelty. To the best of our knowledge a *generic* flight logbook data transfer system has not been implemented before and is thus the key contribution of our work.

The remainder of this article gives a high level eLog system overview and discusses some technical aspects in detail. Due to space constraints a discussion of related work is omitted. The paper ends with a conclusion and outlook.

2 eLog: System Overview

The designed generic flight log data transfer system is based on ideas frequently found in extract transform load (ETL) processes in data warehouse systems [5]. Note though that the implementation itself is not based on data warehouses but rather uses a transactional DBMS-based approach [3]. Next we explain the data flow within the eLog system followed by some selected eLog design decisions.

2.1 Data flow within eLog

The data flow within eLog follows several steps to map XML input data to arbitrary output aircraft maintenance systems on the conceptual level. Figure 1 shows the high level flow of input data from the flight logbook system until it is procured to the maintenance system.

An input reader component – where different implementations for different source data formats may exist – uses in step 1 polling to check whether new XML files have been delivered by aircrafts as electronic logbooks. Polling is implemented by frequently checking a predefined directory on a dedicated server for newly added files. The data is validated against the XML schema (step 2) of the particular aircraft's electronic logbook format, e.g., against different XML schema versions like those that are defined in [1]. If the data is formally valid, it is transferred into a buffer database (step 3), where it is stored in its original format in an *XML-type* attribute. Else an error handling takes place. As long as aircrafts provide source data in XML format a single database schema is sufficient for the buffer database.

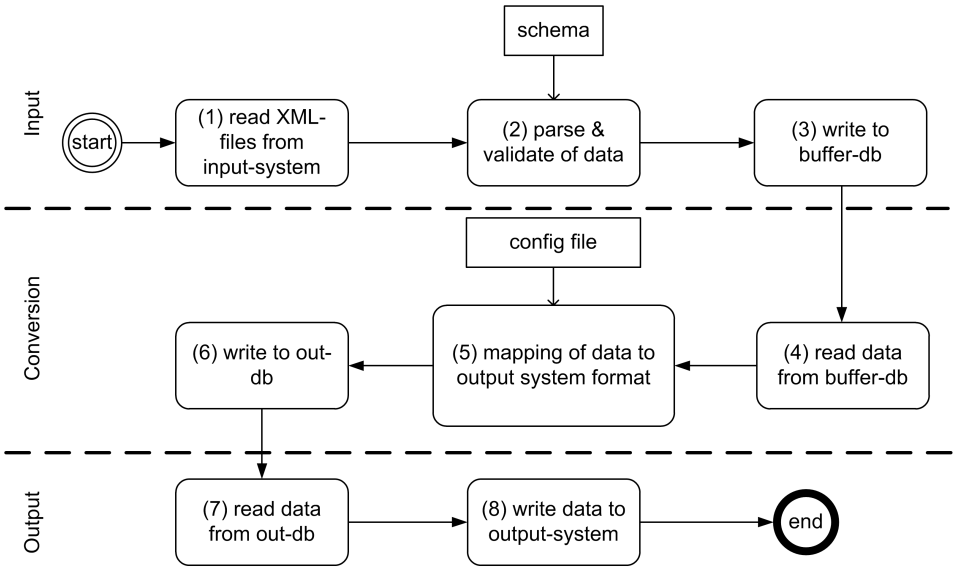

Figure 1 Generic eLog data flow

Again using polling, a mapping component checks for newly arrived source data (step 4) in the buffer database. It utilizes a flexibly configurable sequence of conversion functions to map the input data to a specific output target system (step 5). The output data is stored in a database (Out DB) again (step 6). It closely resembles the data structure of the relevant airline operator's maintenance system.

Consequently for each maintenance system there will be an individual schema in the output database. Options in the mapping configuration file include checks for already present dependent source data, flexible mapping to one or more new entities in the Out DB for every input entity as well as features to update already existing entities in the Out DB.

Eventually another upload component transfers data from the Out DB (step 7) into the airline maintenance system (step 8) whenever a meaningful entity of the maintenance system has been assembled in the Out DB from one or more input entities. Amongst other options, the airline maintenance system used here provides a Web services interface for access.

2.2 Selected eLog design decisions

To give an idea how the overall eLog architecture and design was established, we explain selected eLog design decisions:

- Error checking and handling is added to all components of the system. For example, schema errors in input documents are logged in the underlying database and an appropriate message is sent to the eLog system operator.
- Throughout the conversion steps DB transactions are utilized to ensure data consistency. In combination with operating system based fault tolerance (automated process restarts), a high degree of fault tolerance of the overall system is thus achieved.
- Technically the system is designed according to the ETL paradigm and it is implemented with different Java processes which together provide the tasks from figure 1. They are combined with a relational DBMS, that also allows for XML data storage (Oracle).

3 Conclusion and Outlook

3.1 Conclusion

So far the concept and prototypical implementation have proven to be very promising. The implementation already covers a single exemplary input and also output system. It includes entities that require almost all possible mapping options between input and output systems. The mapping configuration is defined in an XML file which is of a proprietary structure. The implementation employs several XPATH expressions to navigate in input files in order to select the information to be mapped. The output DB resembles the simple relational structure of most airline maintenance systems.

Both mapping configuration as well as input and output data formats are sufficiently generic in order for the system to be easily adjusted to specific data formats. The overall modular design of the system by decoupling input and output system leads to a highly scalable overall architecture.

3.2 Outlook

By now only a brief evaluation of the first eLog prototype took place. This evaluation mostly included general tests of the eLog system architecture (based on the prototype), some XML database performance tests based on realistic data volume assumptions, tests with different generated flight log data sets, and some error handling / restart experiments. All conducted tests with the final eLog prototype showed very promising results for further development.

Future work includes evaluation of eLog in a production environment. Since only selected representative entity types from [1] are currently with eLog processable, the remaining ones have to be added. Moreover, additional different aircraft log file source systems and target airline maintenance systems shall be connected to eLog.

The former will provide additional proof for the scalability and reliability aspects. The latter will lead to the definition of a universal XML schema for the mapping configuration in order to overcome the currently proprietary XML format used.

In summary the generic integration of flight logbooks that has been implemented in this project shows the potential for reduced transfer times of maintenance information coupled with increased correctness when compared to the current manual process. This will eventually lead to reduced maintenance times for aircrafts and thus increase profitability of the airline.

References

1 Air Transport Association of America, Inc. (ATA). *Industry Standard XML Schema for the Exchange of Electronic Logbook Data*, 1.2 edition, May 2008.

2 Jürgen Dunkel, Andreas Eberhart, Stefan Fischer, Carsten Kleiner, and Arne Koschel. *Systemarchitekturen für Verteilte Anwendungen*. Hanser, München, 2008.

3 Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems (5th Edition)*. Addison Wesley, U.S.A, 2006.

4 Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman, Boston, MA, USA, 2003.

5 W. H. Inmon. *Building the Data Warehouse*. Wiley, U.S.A, 2005.

6 Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall, Upper Saddle River, NJ, 2005.

7 David S. Linthicum. *Enterprise Application Integration*. Addison-Wesley Professional, 1999.