Use of Open Source in Integrated Modular Avionics for A380 Program

Laurent Laval, Gilles Richard

Use of Open Source

in Integrated Modular Avionics

for A380 Program

LAVAL Laurent
Flight System department
THALES AVIONICS
105, av. du G[al] Eisenhower
BP 1147
31036 Toulouse Cedex

Phone : 33 (0)5 61 19 34 49
Fax : 33 (0)5 61 19 65 10

RICHARD Gilles
Flight System department
THALES AVIONICS
105, av. du G[al] Eisenhower
BP 1147
31036 Toulouse Cedex

Phone : 33 (0)5 61 19 75 39
Fax : 33 (0)5 61 19 65 10

Keywords : SNMP, Open source, development process

1. Introduction

This paper presents the use of the open source approach on the SNMP software project developed for the Integrated Modular Avionics (IMA) for A380 Program.
The SNMP (Simple Network Management Protocol) is used to monitor the future A380 network called AFDX. This software function is widely present in standard Ethernet network, and runs on operating systems like Windows or Unix.
Open source of the SNMP function has been used as a basis for the development in order to reduce costs and to secure the development (in terms of delay and maturity).
This paper presents the context of the project, then the experiment during the development of this project by using an Open Source.

2. Project overview

2.1. IMA Concept

Traditional aircraft functions runs on a number of dedicated hardware units. Considering that most of those hardware units perform basically the same functions (input acquisitions, processing and computation, and output generation), a natural optimisation of resources has been realised to share the development efforts by identifying common parts, standardising interfaces and encapsulating services…, in other words, to adopt a modular approach. Then, the next step was to share the hardware and software resources by integrating several functions on the same execution platform, with the appropriate segregation (spatial and temporal) between them. This modular approach is known as Integrated Modular Avionics (IMA).

A full IMA system includes a number of computing modules communicating over a shared network. The standard defined for classical Ethernet doesn't meet all the requirements of the avionics world. A set of additional services has been defined in order to provide a safe, deterministic and reliable network called AFDX (Avionics Full DupleX switched ethernet). The AFDX network uses as far as possible standards protocols.

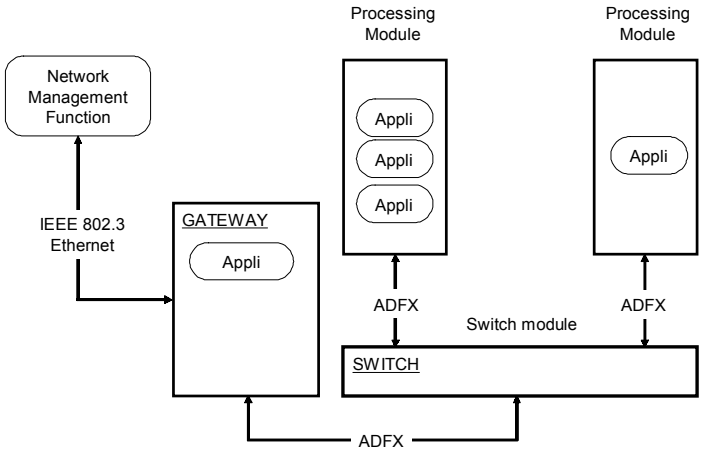


Figure 2-1: IMA communication infrastructure (simplified)

For AFDX maintenance purpose, i.e., failure detection reported by AFDX elements, the network management is made using the SNMP protocol. The SNMP is an application-level protocol part of the UDP/IP protocol suite.

Network management is designed to be as simple as possible. It is based on two elements:

- Network manager is responsible for running management application that monitors and controls the network subsystems.
- Network subsystem agents can be processing modules like workstations, intelligent hubs, routers and bridge/switches that host *SNMP agents*. These *SNMP agents* are responsible for performing the functions that are requested by the network manager (monitoring, report on failure, configuration…).

SNMP is the means by which the management station and the network subsystems communicate. SNMP is a simple protocol that allows a network administrator to inspect or alter variables on a network element from a remote management station.

The SNMP was designed in the late 1980s to facilitate the exchange of management information between networked devices. The first version of SNMP (SNMPv1) went through the standards-process quickly and became an Internet Standard in 1990. SNMP is supported by almost all network devices and is widespread used today. SNMP is the most popular protocol used to manage networked devices.

2.2. SNMP function

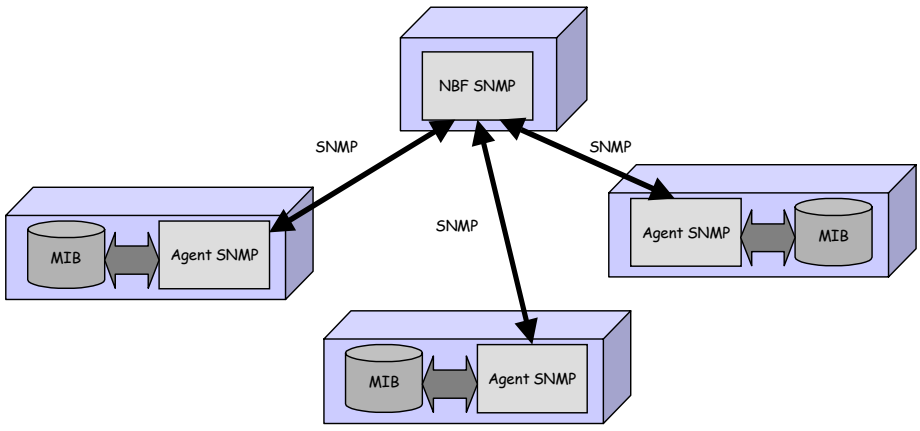
The SNMP part developed by Thales for the Integrated Modular Avionics (IMA) for A380 Program corresponds to an *SNMP agent* included on processing modules. It shares this task with the Network BITE (Built In Test Equipment) Function (NBF), which collects and centralises information on the network status and failures and performs error diagnosis and localisation.

Basically, the *SNMP agent* implements the SNMP protocol version 1 (Ref. [2]) and the MIB (Management Information Base). The *SNMP agent* interprets the SNMP messages, controls the MIB containing the information about the elements to be managed and send the response to the network manager. The MIB implemented is not the standard MIB generally used on ethernet network, but a specific MIB related to AFDX network.

The MIB is a structured database, a tree, that contains the information about the elements to be managed. Every resource to be managed is represented by an object in the MIB. The leaf objects of the tree are the actual managed objects, each of them represents some status object, counter, or related information that is to be managed. These objects allow the detection/localisation of a network failure and the analysis of the network (performance analysis, etc.) for maintenance and tests means purpose. A network management function may monitor the resource at that system by reading the values of objects in the MIB and may control the resources at that system by modifying those values.

Three general-purpose operations may be performed on objects with SNMP:
- get, to retrieve a set of objects values from an *SNMP agent*;
- set, to update a set of objects values in an *SNMP agent* and
- trap, when an *SNMP agent* sends an unsolicited objects values to a management application.

Figure 2-2: AFDX maintenance structure — general view



3. Prototyping Phase

In order to make measurement of critical resources (memory, cpu…) on an IMA platform, it has been planned to make an *SNMP agent* prototype. An open source has been used to develop this prototype with a minimal cost.

3.1. Open Source Software selection for prototype

As the SNMP version 1 is a standard defined in 1990, an important number of open source software implementation are available today. The choice of the open source must be done following specific criteria that seems to us important for our development constraints. The retained criteria was:

- A documentation available (completeness and consistency).
- Simplicity, readability and maintainability of code implementation.
- An operational open source version widely used.

Comprehensive and qualitative documentation was a key driver for the adoption of an open source in our project. A good documentation would help us to produce software specification and software design documentation during the retro-engineering phase.

The open source implementation should be simple and should have a modular design for allowing adaptations to our needs and operational requirements. The open source software found should be also reliable.

The selection of the open source is subjective and does not try to cover all important packages available. We choose an open source published in a book (Ref [1]) which was compliant to all these criteria that are dominant in the decision process concerning whether the open-source software can be reused.

3.2. Prototype results

Results produced by this prototype were very positive:

- An *SNMP agent* function that works correctly
- An open source code simply and easily to understand that allows us to fit it to our needs with few adjustements.
- Measurements on memory and timing constraints were found in line with our resources and margins.
- The use of SNMP prototype allows us to have quickly a source with few remaining bugs.
- An open source code very simple and a sufficient documentation that did not need support.

As the use of the open source in the prototype was very positive, this version was a good solution to take the concepts used in the development and we decided to reuse this prototype in our development to secure technical aspects.

As software development from scratch is very expensive, we have to find solutions or opportunities to reduce this cost by reusing previous developed software (PDS); the use of open source can help us to reduce costs. However, the development of the SNMP software shall comply with the constraints of a development for the production of an embedded software for airborne systems.

4. Constraints to use open source in aircraft embedded software development

4.1. Constraints on Aircraft embedded software

The production of software for airborne systems and equipment used on aircraft or engines, that performs its intended function with a level of confidence in safety, shall follow the DO-178 B guidelines (Ref. [3]) in order to contribute to the aircraft certification.

In order to be certified, the software production shall follows the software life cycle processes defined below:

- Software development (specification, design, coding, integration).
- The integral processes that ensure the correctness, control, and confidence of the software life cycle (software verification, software configuration management, software quality assurance, certification liaison).
- Software planning

For each process, the DO-178B describes:

- objectives for software life cycle processes
- descriptions of activities and design considerations for achieving those objectives
- descriptions of the evidence that indicate that the objectives have been satisfied.

In the IMA project, the development of the SNMP project shall apply the DO178B level C considerations. Of course the original open source did not respect all the constraints of the software life cycle.

4.2. Open Source Software definition and legal aspects

The term "Open Source software" is both an informal term that refers to software where the source code is available, and a precise definition established by the Open Source Initiative that implies some legal constraints like free redistribution, availability of a source code, and free modifications to adapt to our specific needs. For more details on this definition, refer to the Open Source Definition (http://www.opensource.org)

For these legal constraints, it is necessary to verify by lawyers that the open source chosen can be used or not in a commercial product.

5. DEVELOPMENT PROCESS

The use of open source can help us to reduce cost in some phases of the software life cycle.

The relative estimated effort in each phase for a full development of this type of software is detailed as follows :

PHASE	INITIAL TECHNICAL EFFORT	EXPECTED TECHNICAL EFFORT WITH USE OF OPEN SOURCE
Specification	20 %	20 %
Design	20 %	16 %
Coding	30 %	24 %
Test phase	30 %	30 %
TOTAL	**100 %**	**90 %**
EXPECTED GAIN	**---**	**10 %**

By using open source, we estimated at first to have a gain in design phase and coding phase.

5.1. Specification phase

Standards specification of the SNMP are the RFC documents. To improve the specification in order to produce faster a comprehensive first issue, we used:

- Internet documentation like FAQs that gives a lot of information
- Analysis of a concrete example of protocol implementation

Furthermore, specific measurements on SNMP usage domain in IMA project made with our prototype allowed us to solve quickly assumptions taken into account on SNMP response time requirements.

5.2. Design Phase

During this phase, it is interesting to understand the details of implementation in order to produce more robust code if we know how the protocol operates. The open source implementation shows data structures, source code and sometimes explains the underlying principles. This helps to answer many questions and explain many subtleties that allows product improvement. The documentation provided explained and showed how a judicious choice of data representation can make the protocol easier to implement and conversely how a poor choice of representation can make the implementation tedious and difficult.

Nevertheless, the design documentation provided with the open source was not adequate to be compliant with the DO178B considerations. So, we make a reverse engineering from SNMP protocol parts of source code. The design of the *SNMP agent* shall also take into account some specificity like AFDX MIB implementation, interfaces with Operating system and network drivers and others technical considerations linked to IMA environment.

The detailed implementation of the open source make easier to write software design documentation because we have access to some of the assumptions that went into the software. In the SNMP project, the Open Source allows to reduce the design phase duration and to get the good solutions to optimise design and performances.

5.3. Coding phase

The entire open source cannot be directly used. Importing lines of code or algorithm cannot be done due to the specific constraints for the production of software for airborne systems.

The open source chosen is a very compact source code, runs on Unix environment, uses dynamic allocation and recursive functions that are prohibited in our coding rules.

Due to the development on an embedded environment, specificities should also be taken into account like:

- Embedded software restriction for
  - The memory management (dynamic allocation prohibited, memory size limitation, …),
  - the performance (code optimisation, specific embedded OS,…)
- Drastic coding rules to have a better quality, maintenance, and a source code easily to test.
- Interfaces with a specific operating system
- Specific customer requirements

The original code has been rewritten in our project to respect these specificities. Nevertheless the data structures and concepts used for this open source implementation have been conserved and some parts of the new version are very similar from original code. Some parts as encoding/decoding SNMP protocol have been used as example to make easier the coding of the new version. For example, the number of lines of codes of the embedded source of the SNMP decoding/encoding parts, was about twice greater in order to respect the coding rules and the extra code needed for robustness.

The version of open software used, even if the code has been rewritten, allows us to reduce the number of bugs found during debug. In modified source code the number of introduced bugs was very low and the investigation/correction of them was very simple thanks to the open source code implementation.

5.4. Test phase

The test phase includes software tests and system tests.

In our quotation, we have planned to buy for tests purpose a SNMP manager COTS to validate the *SNMP agent*, but it was not possible to find a manager that can be adapted to all of our needs (manager that runs on several platforms (Unix, windows, Linux), control of SNMP messages exchanges, manager that can be launched by line command for executing tests in batch mode…). So because of all these constraints, we preferred to develop our own SNMP manager.
For software tests, we have used some parts of Open Source mixed to proprietary components and tests tools components to build a specific tool that was qualified for generation of SNMP exchange (request/response).
For system tests, we produced a specific SNMP manager by using also some parts of Open Source.

The use of open source in our project for test purpose allows us to develop easily a flexible test environment for software tests and system tests. The implementation of the open source code parts can be used directly because these tools run on host environment and are not constraints by certification.

PHASE	TECHNICAL EFFORT REALISED
Specification	18 %
Design	16 %
Coding	26 %
Test phase	20 %
TOTAL	**80 %**
REALISED GAIN	**20 %**

5.5. Estimation of the gain provided by the use of open source

In the figure presented below, we estimate the gain of the use of open source in our project in each phase.

6. CONCLUSION

The open source has been used in all the parts of the development process from specification phase to test phase. All parts of the original open source code have been rewritten due to the constraints of our implementation, but few bugs have been introduced during this task. Open source is also interesting to produce flexible test environment. In each phase, the open source made the development easier.

The open source software has been ideally suited for reuse in our project context and could be used more generally in the development of protocol standards; it avoids to start a development from scratch.

Open source process selection is very important on obtained results.

Our experiment shows that :
- The initial development delay was not significantly reduced
- The cost was reduced by 20 % versus a classical development
- The product is robust

Now we expect :
- A low number of remaining bugs
- A low cost of maintenance

REFERENCES

[1] Douglas E. Comer, David L. Stevens “Internetworking with TCP/IP Volume 2” Third Edition. Prentice Hall. 1995. Source code available at : ftp://ftp.cs.purdue.edu/pub/comer/TCPIP-vol2.dist.tar.Z

[2] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1157, SNMP Research, PSI, MIT, May 1990.

[3] RTCA, Software Considerations in Airborne systems and Equipment certification, EUROCAE - ED 12 B – RTCA DO-178 B, 1992