ATOS: Integration of Advanced Technology Software within Distributed Spacecraft Mission Operations Systems.

M Jones, J Wheadon,
W O'Mullane
European Space Operations Center
Robert-Bosch Str. 5,
64293 Darmstadt,
Germany.

D Whitgift, K Poulter
Logica UK Ltd, UK

M Niézette, R Timmermans
Space Applications Services, Belgium

Iván Rodríguez, R Romero
GMV, Spain

Abstract: The Advanced Technology Operations System (ATOS) is a programme of studies into the integration of advanced applications (including knowledge based systems (KBS)) with ground systems for the support of spacecraft mission operations.

Keywords: OO, ATOS, AAM, DARPA, KQML, KIF, Knowledgebase, Infrastructure, Operations, Interface.

1 Introduction

The aim of ATOS is to allow data to be shared by multiple knowledge based and traditional applications.

At ESOC several studies have applied knowledge based techniques to specific areas of mission operations, producing a number of independent prototype KBSs. The studies showed that a great deal of common information was used in many of the applications. However, the various prototypes used different KBS tools and knowledge representations which meant they could not be easily integrated. The initial objective of ATOS is to find a solution to this integration problem without imposing a knowledge representation on all applications.

Section 2 examines the problem domain of ATOS. Section 3 outlines the ATOS architecture and section 4 discusses a prototype of the architecture and of applications which use it.

2 SMOS Integration Problem

A Spacecraft Mission Operations System (SMOS) comprises the set of facilities needed to carry out all the mission operations. Mission operations can be split into four areas:

- *Mission Preparation*. The tasks for the preparation and configuration of the Mission Control System (MCS) prior to the start of the mission, as well as the maintenance and updating of the basic reference mission knowledge of the MCS, during the mission.
- *Mission Planning*. The planning and scheduling of mission operations activities.
- *Mission Operations*. All tasks involved in control monitoring, and reporting of the mission.
- *Training / retraining* of operations staff.

In general, independently developed software, possibly running on different platforms will support each of these areas. The areas, however, are far from independent: mission preparation produces the database for operations, mission planning produces the plan of operations to be executed by mission operations and the progress of mission operations conditions the plan. Furthermore, applications use complex data and may use knowledge based techniques.

There are a number of considerations with such systems:

- Because such systems are large and difficult to implement, re-use of many applications is necessary to avoid loss of investment.
- The required capabilities change during the system's working life, for example, for new missions, as users' needs change, or when new technology (platforms, networks...) is introduced.

- The various applications making up the whole system are frequently inflexible, with rigid and restricted interfaces. For example, replanning in the event of failures can be cumbersome because of the interface between the control and planning systems.
- The applications also make use of knowledge about the spacecraft, the ground systems and the operational procedures. Parts of this knowledge are held centrally, but a significant amount is held locally by the applications, each of which may use its own representations and conventions. This leads to potential duplication and inconsistency of knowledge and related problems in system maintenance.

These problems demand solutions: budget restrictions no longer allow the luxury of reimplementing large parts of systems for new missions.

2.1 Solution: The ATOS Approach

A combination of the following two approaches helps to address the problems outlined above:

- Implementation of generic applications from which specialisations may be built.
- Use of principles of federation to integrate heterogeneous applications into a single system.

2.1.1 Generic Systems

Individual parts of a mission operations system can take the approach of developing generic applications which can be specialised for particular missions. The new generation of ESA Spacecraft Control Operations System (SCOS-II) [6] is a good example of this. In SCOS-II, the basic functions of spacecraft control and monitoring are implemented as an object-oriented class library, from which mission systems can be built. Specialisations to mission needs can be provided using "Implementation by Difference" [3].

2.1.2 Federation

Development from generic components is fine, but a problem is usually approached from several user perspectives resulting in several specialist applications. To integrate these we need a generic communication mechanism for software components, analogous to what the SCSI protocol does for hardware components.

ATOS is a federation-enabling technology. Each application (of a federated MCS) has only to provide an interface to the ATOS infrastructure to allow it to use data (and have its data used) by other applications. In effect this extends the object-oriented philosophy to the application level by providing a consistent interface to a set of applications which a developer may use without knowing implementation details of the individual applications.

Although ATOS has been motivated by the needs of mission operations, the concepts (and possibly even the emerging tools) are not restricted to that discipline. The approach could be used in any area in which heterogeneous applications, possibly originally designed to be "stand alone" (i.e. without regard to eventual integration) must be made to work together.

3 The ATOS Architecture

This section describes the architecture which realises the approach to integration outlined above. Section 4 describes prototypes of this architecture and of applications which use it, and illustrates how the architecture works in practice.

3.1 The Mission Information Base

Each application in the ATOS environment is called an ATOS Application Module (AAM). AAMs communicate with each other via the ATOS infrastructure.

As shown in figure 1, each AAM has its own knowledge base. The Mission Information Base (MIB) is defined to be the union of the knowledge bases of all the individual AAMs. The scope of the MIB is thus very broad and encompasses:

- *Flight Operation Plans (FOP)*, including timelines (a scheme of mission operations activities for a particular mission phase or scenario) and Flight Control Procedures.
- *Documents*, including text and graphics, for example the spacecraft users manual.

- *Design information*, including the behaviour of components of the spacecraft.
- *Traditional spacecraft databases*, for example parameter characteristics and telecommand characteristics.
- *Rules and operational constraints* of the mission; for example, if the spacecraft is in eclipse then the payload is on standby.

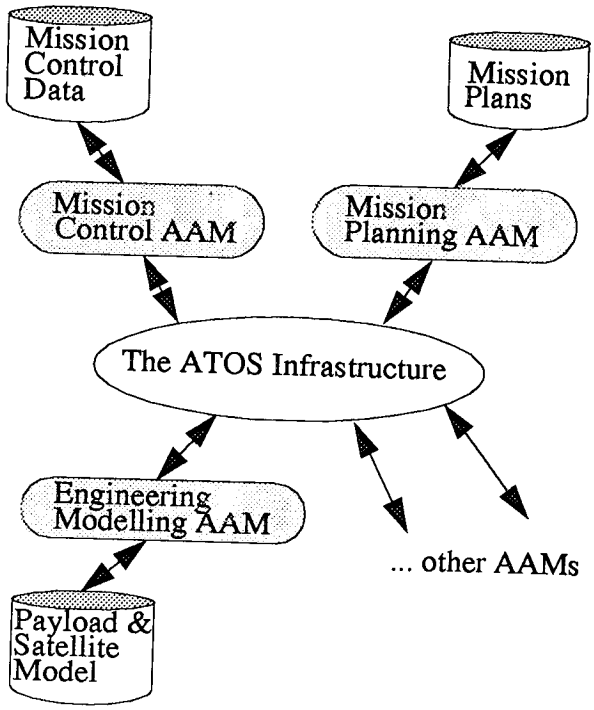

Figure 1 AAMs and the ATOS Infrastructure

The AAMs which manage components of the MIB may be physically distributed, may use different approaches to structuring knowledge (relational, object-oriented, rule based) and may use different tools for storing knowledge. The MIB is thus a federated database of loosely coupled, heterogeneous components.

3.2 The Functions of the Infrastructure

The ATOS infrastructure is the glue which integrates AAMs. The infrastructure "facilitates" (in the sense discussed in [1]) the integration of AAMs by:

- *Routing a message* to an AAM which provides the information or service required by the message.
- *Maintaining links* between information items in different components of the MIB.
- *Detecting significant changes* in the state of the MIB and informing AAMs accordingly.
- *Controlling access* to the information and services provided by AAMs.
- *Maintaining a timetable* that describes which AAMs can use which services of other AAMs and when. This timetable is updated by a mission planning AAM.
- *Logging messages*, as requested.
- *Buffering* messages before they are read.

Clearly some of these services are more innovative and interesting than others. Later sections of this paper concentrate on message routing, link management and detection of change in the MIB.

3.3 The Ontology of Shared Knowledge

AAMs must be able to share knowledge. For example, the results of mission planning are inputs to mission execution; details of a detected anomaly are the basis of fault diagnosis. [3] includes a detailed discussion of the importance of knowledge sharing in spacecraft operations.

To share knowledge AAMs must have a common understanding of concepts and terms which is provided by the ontology.

In ATOS, the ontology is written in a declarative, formally defined language called Ontolingua [2] which is:

- Expressive, so that rules and behavioural knowledge can be shared between AAMs.
- Independent of any particular approach to structuring knowledge.

Note that although Ontolingua allows terms and concepts to be defined using rules, the ATOS infrastructure does not infer knowledge from these rules - that is the responsibility of the AAMs which use the concepts.

The most basic use of the ontology is as a paper standard to which AAMs comply. Thus if there is a standard definition of the terms "resource", "schedule" and "activity" then

AAMs which comply with the standard are guaranteed to use these terms in the same way.

A second use of the ontology is to derive an AAM's knowledge structures. The ontology is written in a formal language (rather than, for example, English); it can therefore be translated into the tool-specific knowledge structures used by an AAM. This approach gives greater assurance that the AAM complies with the ontology and it can also reduce the effort of developing the AAM.

The next section discusses further uses of the ontology, including that of routing a message on the basis of its content.

3.4 The Metabase

As discussed in section 1.2, the MIB is distributed among AAMs, it is not held by the ATOS infrastructure. The infrastructure does, however, record the types of information in the MIB (as defined by the ontology) as well as information about selected objects in the MIB. The infrastructure does not actually store these objects but records their existence, some of their attributes and links between them. This view of the MIB is called the metabase.

Figure 2 depicts three example AAMs which each manage part of an MIB. One AAM uses a relational database, one uses a hierarchical database and one stores documents. The existence of certain MIB objects is recorded in the metabase; this is shown in the figure by a dashed line from the MIB object to its metabase image. The figure also shows links in the metabase between objects which are managed by different AAMs. For example, a tuple in the relational database might be described by a document to which it is linked.

The three major roles of the metabase in supporting knowledge sharing are outlined in the following three subsections. Each of these roles is based upon the global view of the MIB which the metabase provides.

3.4.1 Links between MIB components

The above discussion of touched on the first of these roles: the metabase stores links between objects in different components of the MIB which allow AAMs to navigate the MIB. Such links cannot be stored by any one AAM which only knows about objects in its own component of the MIB.

Link types are defined in the ontology and have different properties. For example, a link type might be defined to be many-to-one, or to be acyclic.

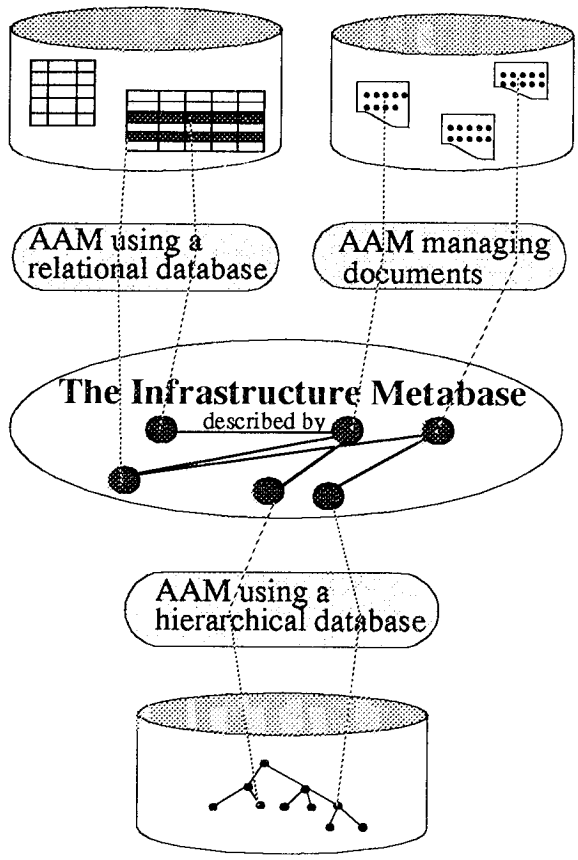

Figure 2 The Infrastructure Metabase

3.4.2 Detecting change

The second role of the metabase is in detecting significant changes in the state of the MIB. One AAM can request a second AAM to perform a specified action when a specified condition is triggered.

For example, an AAM may ask to be informed when the voltage of a battery falls below a certain level. A condition is expressed in terms of the state of the MIB; more precisely, a condition is a query over the MIB which evaluates to true or false; the condition is triggered when the value of the query changes from false to true.

Unfortunately, this approach to detecting change has two limitations. The first is that not all AAMs will be sophisticated enough to detect changes expressed as arbitrary conditions. The second is that an AAM can only monitor its own component of the MIB for change; no single AAM can effectively monitor a condition which involves two or more components of the MIB.

The two limitations are addressed by the infrastructure and its metabase: the infrastructure is able to detect changes to the state of the metabase and, because the metabase is a global view of the MIB, the scope of the condition is not limited to one component of the MIB.

This role of the metabase allows dependencies between components of the MIB to be managed. Imagine, for example, two AAMs one of which plans a mission and the other of which maintains information describing the design of the spacecraft. The metabase holds an abstraction of the plan and the spacecraft design, as well as links corresponding to dependencies of the plan upon the design. The planning AAM instructs the infrastructure to inform it when there is a change or correction to the design which requires the mission to be replanned. The planning AAM then obtains details of the change by querying the design AAM directly.

3.4.3 Content-based routing

When an AAM sends a message, it normally specifies explicitly the AAM which is to receive the message. Sometimes, however, an AAM does not know to which AAM to send the message. In this case the AAM instructs the infrastructure to send the message to the AAM which can best provide the required service or information. This is called content-based routing: AAMs first advertise their abilities to process messages, then the infrastructure routes messages on the basis of these advertisements.

As a simple example of content-based routing, an AAM might advertise its ability to provide the voltage all batteries. The infrastructure can then route a message which asks the voltage of a specified battery to this AAM.

The infrastructure can perform more sophisticated content based routing. Suppose, for example, an AAM advertises its ability to provide information about the power supply subsystem. The infrastructure can then route to this AAM a message which queries the current from the solar array if it knows from its metabase that the solar array is part of the power supply.

3.4.4 Managing the metabase

The metabase records the existence of some of the objects in the MIB and contains some of their attributes. Except for links, the metabase is a partial copy of the MIB. Objects, attributes and links should only be in the metabase if they are required for one of the roles of the metabase described above. The metabase is not expected to be large. For example, an AAM which manages documents might record in the metabase the existence of each document and the date of its most recent issue, but it would not hold the text of the document.

The accuracy of the metabase is, of course, the responsibility of the AAMs. For example, if the metabase records the voltage of a battery then the AAM which manages the corresponding part of the MIB must update the metabase when the voltage of the battery changes.

Which parts of the MIB should an AAM copy to the metabase? The simple answer is that this is a question for the designers of the system who decide how to integrate the AAMs. A more sophisticated and dynamic approach is that the AAM is sent a message which specifies the knowledge which it must copy to the metabase.

3.5 Messages and their Structure

In a spacecraft control system, AAMs and the infrastructure share knowledge by sending each other messages. The meaning of these messages is defined at three levels:

- As discussed in section 3.3, the ontology is a dictionary of the terms and concepts of spacecraft operations and is expressed in Ontolingua.
- A language called Knowledge Interchange Format (KIF) is used to express knowledge using the terms and concepts of the ontology.

- A protocol called Knowledge Query Manipulation Language (KQML) which AAMs use to communicate at run time.

This approach to knowledge sharing is based upon the work of the DARPA Knowledge Sharing Effort [5].

KIF expresses first order predicate calculus in a LISP-like syntax. It is not expected that AAMs use KIF internally; indeed, it is important that AAMs are not constrained to use a particular representation format. AAMs must therefore translate shared knowledge to and from KIF.

KQML provides performatives, i.e. message types, which define the intent of a message. Consider, for example, the following simple KIF sentence:

```
(position sample lower-most)
```

This sentence could be the content of any of the following three KQML performatives:

- `ask`, a query "Is the sample in its lower-most position?" with answer yes or no.
- `reply`, an answer to a question such as "What is the position of the sample?"
- `assert`, informing the receiving AAM that the sample is in its lower-most position.

The KQML performatives used by ATOS are adapted from those described in the draft KQML standard and include performatives in the following areas:

- Asking and replying to a question. Multiple answers to a question can be sent as one long reply or as a stream of replies each containing one answer.
- Asserting a fact to be added to the receiver's knowledge base.
- Advertising the sender's capability to perform a service.
- Instructing the infrastructure to route a message to the AAM best able to process it.
- Instructing the receiver to perform an action when a condition arises.

Possible arguments of a message include:

- *Content*. This is the body of the message, for example the actual query of an ask message.
- *Language*. The language of the content. Normally this is KIF but AAMs can communicate using other languages such as SQL and SGML. If they do so the content is not understood by the ATOS infrastructure.
- *Receiver*. The AAM to which the infrastructure should send the message.
- *Reply-with*. Whether the sender of the message expects a reply, and if so a tag for the reply.
- *Receipt*. Whether the sender requires a return receipt when the message is read.
- *Log*. Whether the message should be logged by the infrastructure.

Most messages are from one AAM to another (via the infrastructure, of course). Some messages are intended for the infrastructure alone; for example, messages which advertise an AAM's ability to perform a service, and messages which query or update the metabase.

All knowledge held by the infrastructure can be queried using KIF and KQML. There are two distinct parts of this knowledge:

- The *metabase*, the structure of which is defined by the ontology of spacecraft operations, as discussed in section 3.4.
- The *infrastructure database*, which contains the housekeeping data held by the ATOS infrastructure. For example, AAMs and their capabilities, message logs... The structure of this database is defined by the infrastructure ontology.

4 Prototypes and Example AAMs

4.1 The Infrastructure Prototype

The ATOS infrastructure has been implemented as a prototype which runs on Unix workstations.

AAMs normally run on different workstations which communicate with the infrastructure using TCP/IP. At the time of writing AAMs must also run on Unix workstations, however, it would be straightforward to port to other

platforms the software which must be linked into the client AAMs.

Two interesting aspects of the prototype are:

- *Storing persistent data.* The metabase and the infrastructure database are stored using a relational database. This requires that the ontology is translated to SQL data definition language and that KIF queries and assertions are translated to SQL data manipulation language.

  Translations must also be performed by AAMs which do not use KIF internally. An important difference is that each AAM has its own specific knowledge structures; it does not need to translate arbitrary KIF queries unrelated to these structures.
- *Content-based routing.* Advertised capabilities, and messages to be routed on the basis of their content, are both expressed in KIF. Matching a message with a capability involves conversion of the two KIF expressions to a normal form and then unifying them using the knowledge in the metabase.

4.2 AAM Prototypes

[7] describes a prototype tool called AMFESYS which maintains a model of a payload: a microprocessor controlled remotely programmable Automatic Mirror Furnace (AMF) for growing crystals in zero gravity.

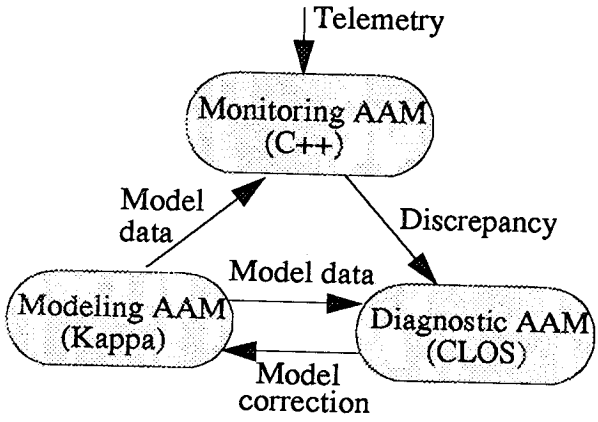

Figure 3 AAMs derived from AMFESYS

The AMFESYS tool has been decomposed into three AAMs as shown in Figure 3. The Modeling AAM maintains a model of the AMF which the Monitoring AAM compares with telemetry from the spacecraft. If a significant discrepancy is detected the Diagnostic AAM performs a rule-based diagnosis of the fault and then corrects the model.

The three AAMs use different approaches to structuring knowledge (C++, CLOS and Kappa) and interact with each other via the ATOS infrastructure. The ontology defines the structure of the AMF.

The following is a fragment of the AMF ontology which specifies a subclass of devices called `Spindles` and specifies that each spindle has a height. The fragment also identifies the instances of the class of spindles in the AMF: `SampleConvSpindle` (the sample conveyance spindle) and `LampDiskSpindle` (the lamp disk spindle).

```
(define-class Spindles (?x)
:def (devices ?x))

(define-relation Height (?x ?y)
:axiom-def (and(single-valued Height)
           (range Height number)
           (domain Height Spindles)))

(define-instance
   SampleConvSpindle (Spindles))
(define-instance
   LampDiskSpindle (Spindles))
```

With this ontology, the AAMs can converse about the height of spindles. For example, the Diagnostic AAM might send the following message to the Modeling AAM:

```
(ask-one :receiver MODEL
 :content (Height LampDiskSpindle ?h)
 :reply-with rid)
```

which might then reply

```
(reply :receiver DIAGNOSTIC
 :content (Height LampDiskSpindle 70)
 :in-reply-to rid)
```

In these messages `ask-one` and `reply` are two KQML performatives; `:receiver`, `:content`, `:reply-with` and `:in-reply-to` label the message arguments. The language of the content of each message is KIF (the default language of all messages). The meaning of the terms in the content of each message is defined by the ontology.

Following the approach discussed in section 3.3, the AMF ontology has been automatically translated into the data structures used by the Diagnostic AAM. This AAM is written in CLOS, so the translation is from Ontolingua to CLOS.

5 Conclusion

The paper has discussed the problems of implementing complex mission operations systems and has described a two-fold approach to their solution: build generic applications and adopt a standard integration framework.

We have not mentioned here the possible overlap of with other integration technologies such as CORBA [4]. They could provide a basis for the more advanced ATOS features.

Much has been achieved since the initiation of the ATOS programme in 1992. We also believe that the approach we have adopted may be effective not only for the space industry but for any industry which needs to integrate applications to build complex systems.

6 Acknowledgments

This work was funded by ESA's Advanced Systems and Technology Programme (ASTP) which is managed by ESA's Directorate of Telecommunications. ASTP promotes the development of new technologies for the space domain, and supports European industry in implementing these in the form of marketable products.

The authors also acknowledge the contributions of Howard Smith (Logica UK) and Herwig Laue (ESOC) to the ATOS programme.

7 Bibliography

[1] Genesereth M.; *An Agent Based Framework for Software Interoperability*, Software Technology Conference '92, Los Angeles, April 1992.

[2] Gruber T., Ontolingua: *A Mechanism to Support Portable Ontologies*, Knowledge System Laboratory Technical Report, Stanford University, June 1992.

[3] Laue H., Kaufeler J-F, Poulter K., Smith H.; *The Advanced Technology Operations System ATOS*; Proc. 2nd Int. Sym. Ground Data Systems for Space Mission Operations. SPACEOPS 1992 Pasadena, California, USA. JPL Publication 93-5.

[4] Object Management Group; *The Common Object Request Broker: Architecture and Specification;* OMG Document number 91.12.1 10 December 1991.

[5] Patil R. et al; *The DARPA Knowledge Sharing Effort: Progress Report*, Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation, Cambridge MA, Morgan Kaufman, 1992.

[6] Poulter K. J., Smith H. N.; *ATOS-1: Designing the Infrastructure for an Advanced Spacecraft Operations*; Proc. 2nd Int. Sym. Ground Data Systems for Space Mission Operations. SPACEOPS 1992 Pasadena, California, USA. JPL Publication 93-5.

[7] Wheadon J.;*AMFESYS: Modelling and Diagnosis Functions for Operations Support.*; Proc. 2nd Int. Sym. Ground Data Systems for Space Mission Operations. SPACEOPS 1992 Pasadena, California, USA. JPL Publication 93.