Intrinsic Interoperability Of Services

A Dream Or A Key Objective For Mission Operation Systems

Dr. Mehran Sarkarati[1], Dr. M. Merri [2], Mariella Spada[3]
European Space Agency, ESA/ESOC, Robert Bosch Strasse 5, 64293 Darmstadt, Germany

Sam Cooper[4]
SciSys UK Ltd, Bristol, BS4 5SS, UK,

Interoperability of mission operation software systems is a key requirement in space operations domain, both within an organisation and in the context of cross-organisation cooperation scenarios. This interoperability is traditionally achieved through deployment of dedicated integration adapters, for bridging discrepancies between proprietary data models, data formats, communication protocols and software implementation technology of each interacting application. The CCSDS Spacecraft Monitoring and Control (SM&C) working group has defined a service-oriented framework, which allows the specification of mission operation services in an implementation and communication technology agnostic manner. One of the main objectives of this framework is to increase the interoperability of mission operation services and reduce the required integration effort for federating services of different stakeholders. Domain specific design standards are a key factor for achieving service interoperability. They specify a reference Service Model for all services within a service inventory, addressing both functional and non-functional aspects of service contracts such as service taxonomy, service addressing, service security, Quality of Service (QoS), transaction management and reliable messaging. The CCSDS SM&C Message Abstraction Layer (MAL) specification provides a reference Service Model for mission operation services. It defines a set of service meta-data for capturing the above-mentioned architectural concerns. Our paper elaborates on what is involved in achieving true intrinsic interoperability of services at run-time, the main obstacles on the way and possible solutions. We believe true intrinsic interoperability of services at run-time (dynamic plug-and-play) is not the prime focus of mission operation services for space programmes of today and near future. The abstraction from the implementation and communication technology, which is currently provided by the CCSDS MO services framework and its Reference Service Model can reduce the effort required for integration of mission operation services significantly, paving the path towards implementation of better and more interoperable mission operation systems.

I. Introduction

A. Integration vs Interoperability

INTEROPERABILITY and integration are closely related concepts in general software engineering, which are occasionally used by mistake as exchangeable terms. They refer however to two very distinct software design

[1] Ground Data Systems Manager, HSO-GDA, CCSDS SM&C Working Group Member, mehran.sarkarati@esa.int
[2] Head of Mission Data Systems, HSO-GD, CCSDS SM&C Working Group Chair, Mario.merri@esa.int
[3] Head of Applications and Special Projects Data Systems, HSO-GDA, mariella.spada@esa.int
[2] Head of Mission Data Systems, HSO-GD, CCSDS SM&C Working Group Chair, Mario.merri@esa.int
[3] Head of Applications and Special Projects Data Systems, HSO-GDA, mariella.spada@esa.int
[4] CCSDS SM&C Working Group Member, sam.cooper@scisys.co.uk

concepts, which can be considered as the exact opposites. The ultimate goal of interoperability, the so-called intrinsic interoperability of software components, is eliminating the need for integration.

Each of these two design concepts has influenced significantly the IT landscape over the last two decades. Integration has been the focus of he Enterprise Application Integration (EAI) era, while interoperability concept has been declared one of the strategic goals of Service Oriented Architectures (SOA). The EAI era has been marked by emergence of a set of enterprise integration patterns [1] and a multitude of both open source and commercial integration middleware products.

The Enterprise Service Bus (ESB) compound pattern [2] has complemented the EA integration patterns with supplementary aspects in support of SOA. The term ESB has become an ambiguous IT buzzword in the recent years, since software product vendors have used it for branding their products. It is therefore helpful for the further discussions of this paper to contemplate the ESB purely as a compound integration pattern, as shown in figure 1.

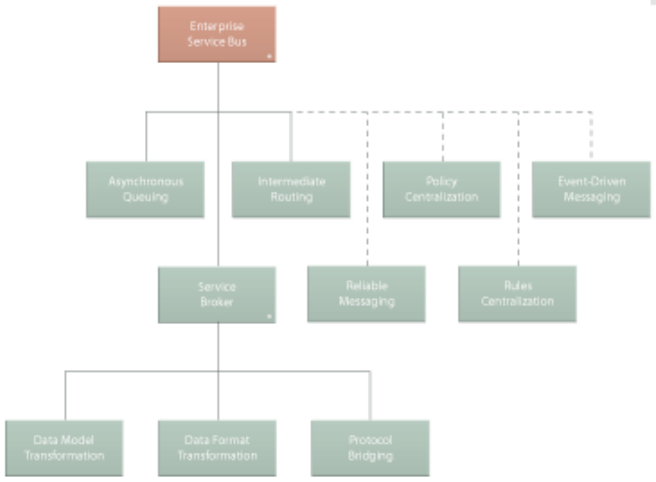

Figure 1: The Interprise Service Buse Compound Pattern [2]

Apache Camel, Appache SeviceMix, Microsoft BizTalk Server, IBM WebShpere Message Broker, IBM WebSphere ESB, TIBCO BusinessWorks, BEA System's AquaLogic Service Bus, Oracle Service Bus, SoftwareAG WebMethods Integration Server, Talend ESB, Progress Software's Sonic ESB, Fuse ESB, Mule ESB and many other middleware products are the implementations of these patterns complemented with a set of out-of-the-box adapters for specific software technologies and products. Figure 2 illustrates the ESB from a middleware platform perspective.

Although the ESB concept, the design pattern and the software products are usually associated with SOA, many of the above mentioned middleware platforms are often used for solving the traditional integration problem of overcoming disparity between interacting non-service oriented software applications. The philosophy behind all EAI patterns has been to allow disparity at different architectural levels among the interacting software applications (e.g. implementation technology, communication, data model and data format). The main objective of the EAI middleware platforms is respectively to reduce the resulting integration effort.

In today's mission operations world, integration is the primarily means of interaction between mission operation software applications both within the boundaries of an space organisation as well as in cross-organisation cooperation scenarios. This is typically achieved by agreeing on an ICD and the communication protocols to be used between any two interacting software applications and transforming to and from the agreed ICD within the boundaries of each application, with or without an integration middleware.

In contrary to the EAI paradigm, one of the strategic goals of service oriented computing is to provide at design time for abstraction from the implementation and communication technology of units of service oriented computing logic, the so called services, hence avoiding as much as possible disparity at run-time when service

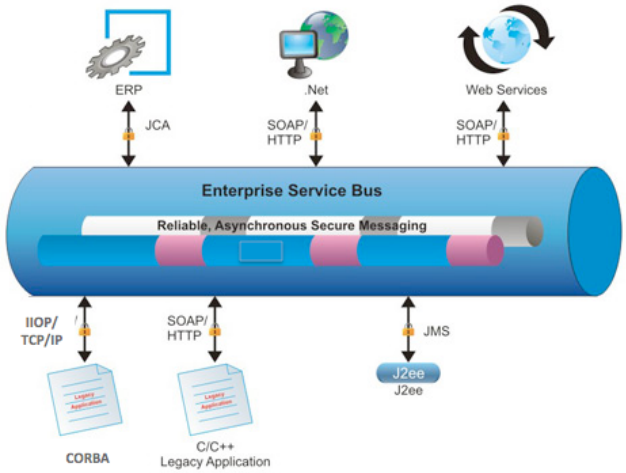

Figure 2: Enterprise Service Bus as an integration middleware

providers and consumers engage in interaction scenarios. This goal is addressed by adopting certain service design principles, such as service abstraction, discoverability, composability, autonomy, loose-coupling and statelessness [3]. Applying a set of domain specific design standards consistently to all services within a domain service inventory and agreeing on a reference service model can furthermore increase significantly the interoperability of services. This design principle is referred to as the "standardized service contract" [3] and will be elaborated on in the next chapters of this paper.

B. Sources of Disparity Among Integrating Software Applications

When discussing integration and interoperability it is worth analysing the sources of incompatibilities between two interacting software applications. The main sources of disparity are illustrated in figure 3 and can be summarised as following:

- Incompatible software implementation technologies, e.g. a Telemetry provider application is implemented in C programming language, running on an a real-time operating system on-board a spacecraft, while the TM consumer is a mobile application, developed in Java running in an Android environment;
- Incompatible data models used by each application, e.g. the specification of telemetry parameters can be different between a Telemetry data provider and consumer;
- Incompatible data format and communication protocols, e.g. the Telemetry data provider publishes data in CCSDS packet telemetry format while the consumer being a mobile devise can only understand XML encoded data received via the HTTP protocol;
- Incompatible syntax and semantics of the exposed capabilities of interacting applications, e.g. the exposed interface of the Telemetry data provider for retrieving Telemetry data provides raw Telemetry data in SI system while the consumer expects calibrated engineering data in imperial units.
- The non-functional requirements such as QoS and security policies of the exposed interfaces, e.g. The Telemetry data consumer application attempts to authenticate itself using a SAML token while the Telemetry providing system expects encrypted username and password. To give a QoS example, the satellite visibility in a particular location may be limited to few minutes, while the consumer is supposed to send requests at any given time;

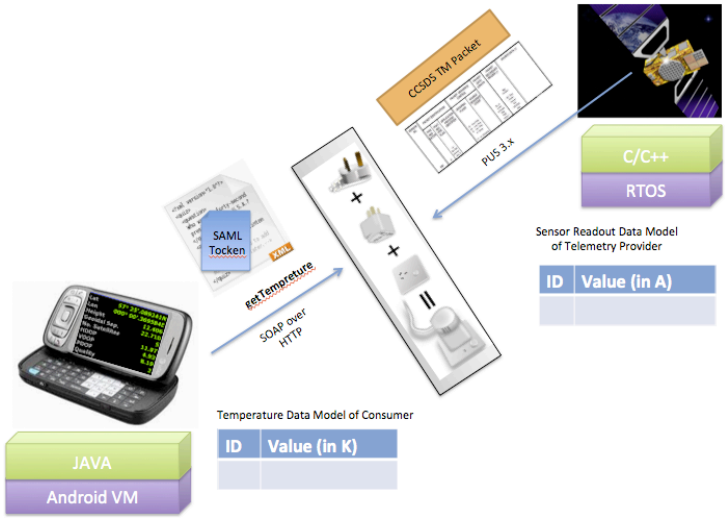

Figure 3: Sources of disparity in interacting software appliations

In order to achieve intrinsic interoperability, i.e. avoid the need for custom integration, the interacting applications (both service consumer and service provider) must comply with

- a set of technical industry standards for abstracting from software implementation and communication technologies to allow a generic middleware to establish at run-time the communication between them;
- a set of domain specific design standards to achieve a common service model which ensure that the content of the communication is interpreted the same way by all communicating parties;

Adherence to the first point does not mean that all interacting parties must use the same software implementation and communication technology. It requires instead that the capabilities of each application be exposed through an implementation and communication technology agnostic service contract. The web-services specifications framework, the so-called WS-* specifications[5], the Service Component Architecture (SCA) specifications [4] and the CCSDS Massage Abstraction Layer, MAL, Specification [10] are examples of such agnostic service specification frameworks.

The implementations of these specifications rely on a run-time environment (middleware), which can map the abstract service specifications dynamically to the concrete implementation and communication technologies on the service provider and consumer side. This mapping from the abstract service contract to the concrete implementation and compunction technology is called service binding.

Despite the importance of the fist point as a prerequisite for achieving interoperability, it is important to acknowledge that no service abstraction framework and the corresponding middleware can alone provide intrinsic interoperability between service providers and consumers, as they do not imply any assumptions on the content and quality of the communication. To fully appreciate the difference, lets consider the example of a USB keyboard for a PC. The USB standard harmonies the physical interface between the keyboard and the computer as well as the electronic communication between the two devises for successful exchange of messages. It makes however no assumptions on the content of the exchanged messages and how they should be interpreted. This becomes apparent when plugging a PC keyboard into a MAC computer and pressing the "Windows Key". The interpretation of the received messages from the keyboard and the associated capabilities are indeed very different in Windows and Mac computers. This is where standardising the semantics of the service contracts for each domain service and the role of a reference service model come into the picture, as we will explain in section 0.

C. The CCSDS SM&C MO Framework Background [12]

The Spacecraft Monitoring & Control (SM&C) Working Group of the Consultative Committee for Space Data Systems (CCSDS), which sees the active participation of 10 space agencies, has been working since 2003 on the definition of a service oriented architecture for space mission operations. The ambitious goal of the WG is to define a set of standardised, interoperable mission operation (MO) services, which allow rapid and efficient construction of co-operating space systems (Ground Segment, but also part of the Space Segment).

For this purpose the WG has defined a MO layered service framework, shown in figure 4, which allows mission operation services to be specified in an implementation and communication agnostic manner. The core of the MO service framework is its Massage Abstraction Layer, MAL, which ensures interoperability between mission operation services deployed on different framework implementations.

The MO services are defined in compliance to a reference service model using an abstract service description language specified by the MAL (similar to WSDL). For each concrete software implementation and communication technology the abstract service contracts must be bound to that particular technology. The MAL layer provides in turn standardised interfaces in form of Application Programming Interfaces (API) both towards both upper and lower layers.

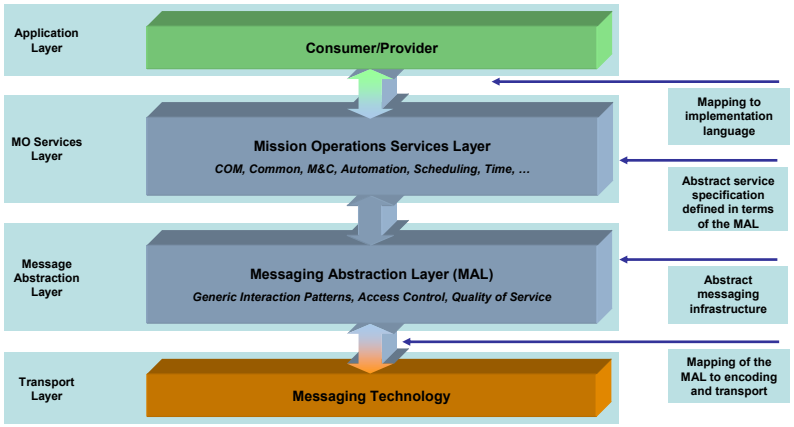

Figure 4: The Hub and Spoke Pattern vs Point-to-Point Integration

[5] WSDL [5], WS-Security[6], WS-Addressing [7], WS-Reliability [8], WS-Policy [9], WS-Transaction [10]

II. The Canonical Message Format and the ESB Role in Achieving Interoperability

The scenario shown in figure 3 depicts a point-to-point interaction between the service provider and service consumer. In this scenario the integration layer is responsible for bridging between different service provider and consumer bindings at run-time. As the number of service providers and consumers grows, the bridging can become a challenge and would require upfront agreements between any two interacting parties, hence a barrier to out-of-the-

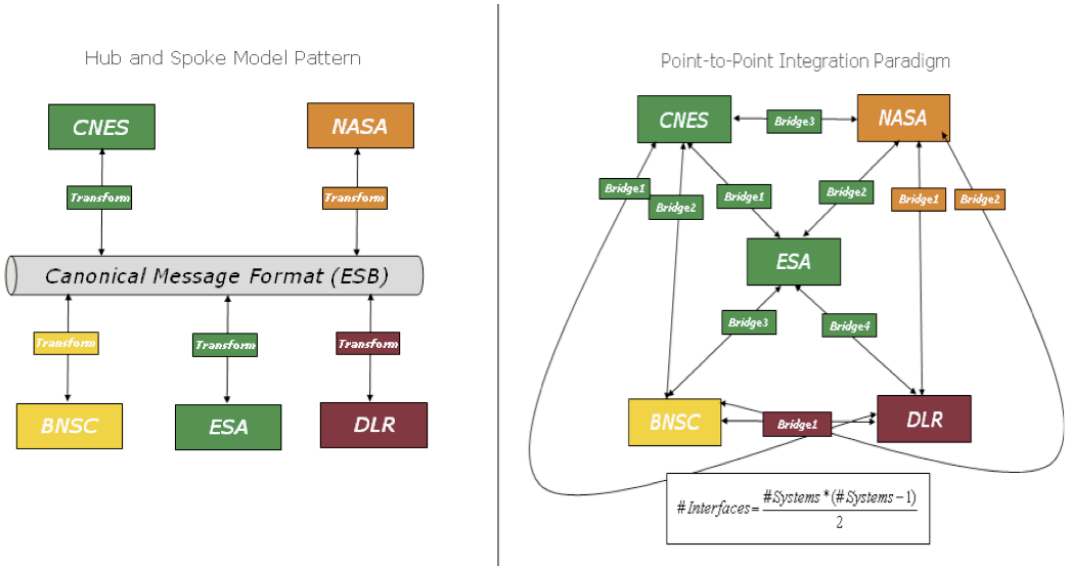

Figure 5: The Hub and Spoke Pattern vs Point-to-Point Integration

box interoperability (i.e. no plug-and-play). The Hub and Spoke pattern, shown in Figure 5, solves this issue by introducing a Canonical Message Format, which is used by an ESB to bridge from a given service binding to any other binding, hence allowing n-to-n communication among all plugged service providers and consumers.

It is again important to appreciate that the utilisation of a Canonical Message Format by the ESB does not mean that all services must adopt the same message format and communication protocol. It means instead that by implementing once a single transformation from a given binding to the Canonical Message Format, the ESB can establish the communication to any other already bridged binding. The advantages of the Hub and Spoke concept can best be explained with an example of natural languages. Lets consider the example of a medical research association with fifty member states. The objective of the association is to make the work results of each researcher as soon as possible available to other researchers in all member states. If all researchers would publish their work results in a common language, e.g. English, the association would need no translators. This would be similar to requiring all interacting applications to use the same software implementation and communication technology, which is often not a realistic scenario. In our example many renowned researches may publish their work in their native language. The association could hire a large number of translators for translating from any of the fifty member state languages to any other language. In this scenario the association would need 49 translators per language, hence 1225 translators in total. Moreover, each time a new member state joins the association, many new translators must be hired (as many as the previous number of the member states). This would be an example of a point-to-point communication pattern. A more appropriate scenario would be however to hire only 50 translators which would translate the research results from any member state language to a common language, e.g. English, for internal use in the translation department, allowing each of the other 49 translators to further translate it from English to its own native language. In this Hub and Spoke scenario when a new member state joins the organisation, only a single new translator must be hired and its researches would have automatically access to the work results of all other researches linked to the association and vice versa

III. The Role of Domain Specific Design Standards and Reference Service Models in Achieving Interoperability

In Service Oriented Architectures, the capabilities of each service are exposed through so called service contracts. This is typically done by specifying the exposed capabilities of each service as a list of operations and specifying the order and the structure of the exchanged messages for each service capability (operation) in an

Operation Identifier	List	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	ListOperationRequest
OUT	RESPONSE	ListOperationResponse

Structure Name	ListOperationRequest	
Extends	Composite	
Field	Type	Comments
folderName	String	Directory name to list the files

Structure Name	File	
Extends	Composite	
Field	Type	Comments
Name	String	Name of the file
Lastupdated	Time	Last updated time of the file

Structure Name	ListOperationResponse	
Extends	GenericOperationResponseType	
Field	Type	Comments
Files	File	Array of type file

Figure 6: The CCSDS Massage Abstraction Layer, MAL, Concept

abstract manner. Figure 6 shows an example of an abstract service contract of an imaginary file management service, specified in the CCSDS MAL service description language. The service provides an operation called "List" for retrieving the list of all files in a directory. The same abstract service contract can also be represented in a machine-readable XML format, as shown in figure 7, which is used by the runtime middleware (the MAL layer in this case) to establish a concrete binding to technology platforms of the service provider and service consumer.

As illustrated in these figures, formalised service contracts define the functional semantics of the exposed service capabilities. As it is the case with legal contracts in real life, service contracts also establish the ground for reaching

Figure 7: The CCSDS Massage Abstraction Layer, MAL, Concept

a common understanding between the service provider and consumer of the modalities of transaction, the duties of each party and its expectations on the counter side. And again as with the legal contracts of the real life the clarity and quality of the agreements formed by a service contract can impact significantly the smoothness of the transaction. To achieve intrinsic interoperability between software services at runtime, it is often not enough to specify the functional semantics of the service interface. The service contract must also cover agreements on a number of non-functional aspects related to communications and interaction management, including

- Addressing
- Security
- Quality of Service (QoS)
- Communication Reliability
- Taxonomy, description and discovery
- Transaction management

Otherwise supplementary point-to-point agreements must be reached between the service providers and consumers before the interaction can succeed. Conversations of type "where do I find your service", "Do I need to authenticate myself before consuming the service?", "What kind of authentication do you expect?", "Do I have to use a secure communication line and encrypt the all messages?", "Make sure the Account number is encrypted when you are sending the request to query operation", "How many consumers can your service support at the same time?", "Please configure your service to send the answer to my request to this endpoint", "The integrity of this operation is essential, so make sure you use a reliable communication so that all messages are received at each end" would be unavoidable if the service contracts are limited to specification of functional service capabilities.

The so-called WS-* specifications WSDL [5], WS-Security[6], WS-Addressing [7], WS-Reliability [8], WS-Policy [9] and WS-Transaction [10] provide a set of standards for addressing each of the above-mentioned architectural concerns for web-services. These specifications define a set of standardised attributes, which can be used at design time for extending the technical service contracts with none-functional requirements of the service. At run-time, the required information (e.g. the replyTo endpoint address, or the credentials of the service consumer) is exchanged between the service provider and consumer, using meta-data embedded in the message headers.

The SCA Policy Framework specification [4] is based on a similar concept. The important issue is however to realise that while these standards provide a generic framework for specifying meta-data, related to non-functional requirements of a service contract, they do not provide a reference service model for a particular domain. In other words, they do not specify which of the many possible non-functional aspects must be specified for all services of a given domain, e.g. for mission operation services. As a result adherence to these generic industry standards alone does again not guarantee interoperability of services. For instance, the WS-Security standard provides the means for specifying many different kinds of security requirements in a service contract. It does however not make any assumptions of which of these aspects shall or shall not be specified for all services of a particular domain. If a service provider would like to specify as part of its service contract that a certain authentication mechanism is required for one of its exposed operations, it could use the corresponding WS-Security policies to do so. The standard does however not require that an authentication mechanism be specified for all operations in all service contracts nor put any preference on a particular authentication mechanism (e.g. plain or encrypted username/password vs. use of certificates). The same applies for specifying if whole or portions of a service operation message shall be encrypted. The WS-Security standard provides again the means for specifying such requirements in a standardised manner; it does however not require the presence of such policy assertions in service contracts.

This is exactly where the **standardised service contract** design principle [3] steps in. Domain specific design standards specify how and to what extent the generic industry standards shall be adopted in service contracts of all services within a domain service inventory. The result of application of this design principle is a **Reference Service Model** that specifies which exact architectural concerns must be addressed by all services of a domain inventory, i.e. which meta-data must be present in all service contracts. The Reference Service Model must consider both design time and run-time architectural concerns to ensure interoperability of compliant services. The abstract service specification template and the related run-time MAL header of the CCSDS MO service framework shown in figure 8 define such a Reference Service Model for mission operation Services.

Area Identifier	Service Identifier	Area Number	Service Number	Service Version
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in replay**	**Capability Set**

Design-Time

Field	Value
URI From	
Authentication Id	
URI To	
Timestamp	Message generation timestamp
QoSlevel	The QoS level of the message
Priority	The QoS priority of the message
Domain	Domain of the message
Network Zone	Network zone of the message
Session	Session of the message
Session Name	Session Identifier for Simulation and Replay sessions
Interaction Type	
Interaction Stage	
Transaction Id	
Service Area	Area Identifier
Service	Service Identifier
Operation	Operation Identifier
Service version	Service Version number
Is Error Message	FALSE

Run-Time

Figure 8: The CCSDS Massage Abstraction Layer, MAL, Concept

IV. Conclusion

Achieving true intrinsic interoperability of services at run-time (real plug-and-play) is a challenging objective for any SOA. The traditional notion of application integration is replaced in service oriented computing with the concept of service orchestration, which aims at composing new service oriented solutions from the repository of existing services and extending them at run-time. This requires that all services be specified in an implementation and communication agnostic manner. A set of cross-domain industry standards such as WS-* and SCA specifications exist today for this purpose. The adoption of these standards alone can however not provide the envisaged intrinsic interoperability of services, as they do not specify a Reference Service Model for any particular application domain.

The CCSDS Mission Operation Massage Abstraction Layer (MAL) specification provides an abstract service specification template for design-time and a related message header model for exchange of corresponding meta-data at run-time. Together they form a Reference Service Model for mission operation services domain. This Reference Service Model is focused on the immediate needs of typical space mission operation scenarios. Dynamic plug-and-play of services at run-time is not the prime concern of these scenarios, since many agreements at organisational level are usually required before software services can engage in interactions. The MAL Reference Service Model captures a more focused set of meta-data related to the non-functional requirements of mission operations services. This metadata includes information related to service taxonomy, service addressing, QoS and transaction management. The security aspects are deliberately limited to authentication and authorisation, for which the Reference Service Model meta-data only includes an identifier, delegating the rest to the implementation of the runtime environment.

The current specification of the MO service framework does not utilise a Canonical Message Format for bridging between different technology bindings at run-time. Hence dedicated technology bridges are required for any incompatible point-to-point interaction of services. Given the scenarios of today's mission operations, this approach is sufficient for the primarily objective of enabling cross-organisation interoperability of mission operation services. It could be however a point for improvement of the MO services framework for increasing interoperability in future.

The abstraction from the implementation and communication technology, which is currently provided by the CCSDS MO services framework and its Reference Service Model can reduce the effort required for integration of mission operation services significantly, paving the path towards implementation of better and more interoperable mission operation systems.

Appendix A

Acronym List

API	Application Programming Interface
CCSDS	Consultative Committee for Space Data Systems
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
HTTP	Hypertext Transfer Protocol
ICD	Interface Control Document
IT	Information Technology
MAL	Massage Abstraction Layer
MDA	Model Driven Architecture
MO	Mission Operations
PIM	Platform Independent Model
QoS	Quality of Service
SAML	Security Assertion Markup Language
SCA	Service Component Architecture
SI	Système International d´Unitées
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	SOAP is not an acronym but a name[6]
TM	Telemetry
WS	Web Service
WSDL	Web Services Description Language
XML	Extensible Markup Language

[6] Multiple abbreviations exist for SOAP including Simple Object Access Protocol and Service Oriented Architecture Protocol. None of them does however reflect the current state of SOAP as basis for many WS-* standards specification.

References

[1]Gregor Hohpe and Boby Woolf, “Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions” Addison-Wesley Professional, ISBN-13: 978-0321200686

[z] Thomas Earl, “SOA Design Patterns”, Prentice Hall, ISBN-13: 978-0136135166

[3] Thomas Earl, “Principles of Service Design”, Prentice Hall, ISBN-13: 978-0132344821

[4] OASIS, SCA Specifications, URL: http://www.oasis-opencsa.org/sca [cited 15 May 2012]

[5] W3C, Web Services Description Language Specification, URL: http://www.w3.org/TR/wsdl”, [cited 15 May 2012]

[6] OASIS, Web Services Security Specification, URL: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss [cited 15 May 2012]

[7] W3C, Web Services Addressing Specification, URL: http://www.w3.org/Submission/ws-addressing/ [cited 15 May 2012].

[8] OASIS, Web Services Reliable Messaging specificatoin, URL: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm [cited 15 May 2012].

[9] W3C, Web Services Policy Framework specification, URL: http://www.w3.org/TR/ws-policy/ [cited 15 May 2012].

[10] OASIS, Web Services Transaction Specification, URL: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx [cited 15 May 2012].

[11] CCSDS, “Mission Operation Massage Abstraction Layer, Blue Book”, CCSDS-521.0-B-1

[12] M. Merri and CCSDS SM&C WG, “Cheaper, Faster and Better Missions with the CCSDS SM&C Mission Operations Framework”, Paper at SpaceOps 2009 .