European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 7: Approach to Safety and Security Reference Manual

BAE Ref No: IAWG-ECOA-TR-009
Dassault Ref No: DGT 144484-B

Issue: 2

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

Note: *This specification represents the output of a research programme and contains mature high-level concepts, though low-level mechanisms and interfaces remain under development and are subject to change. This standard of documentation is recommended as appropriate for limited lab-based evaluation only. Product development based on this standard of documentation is not recommended.*

1 Table of Contents

2 List of Figures

3 List of Tables

4 Abbreviations

API	Application Programming Interface
ASC	Application Software Component
ASCI	Application Software Component Interface
DAL	Development Assurance Level
EAL	Evidence Assurance Level
ECOA	European Component Oriented Architecture
ELI	ECOA Logical Interface
GPS	Global Positioning System
IAWG	Industrial Avionics Working Group
IMA	Integrated Modular Avionics
QoS	Quality of Service
RPC	Remote Procedure Call
SIL	Safety Integrity Level
SRS	Software Requirement Specification
SVD	Software Version Description
UAS	Unmanned Air Systems
XML	Extensible Mark-up Language

5 Introduction

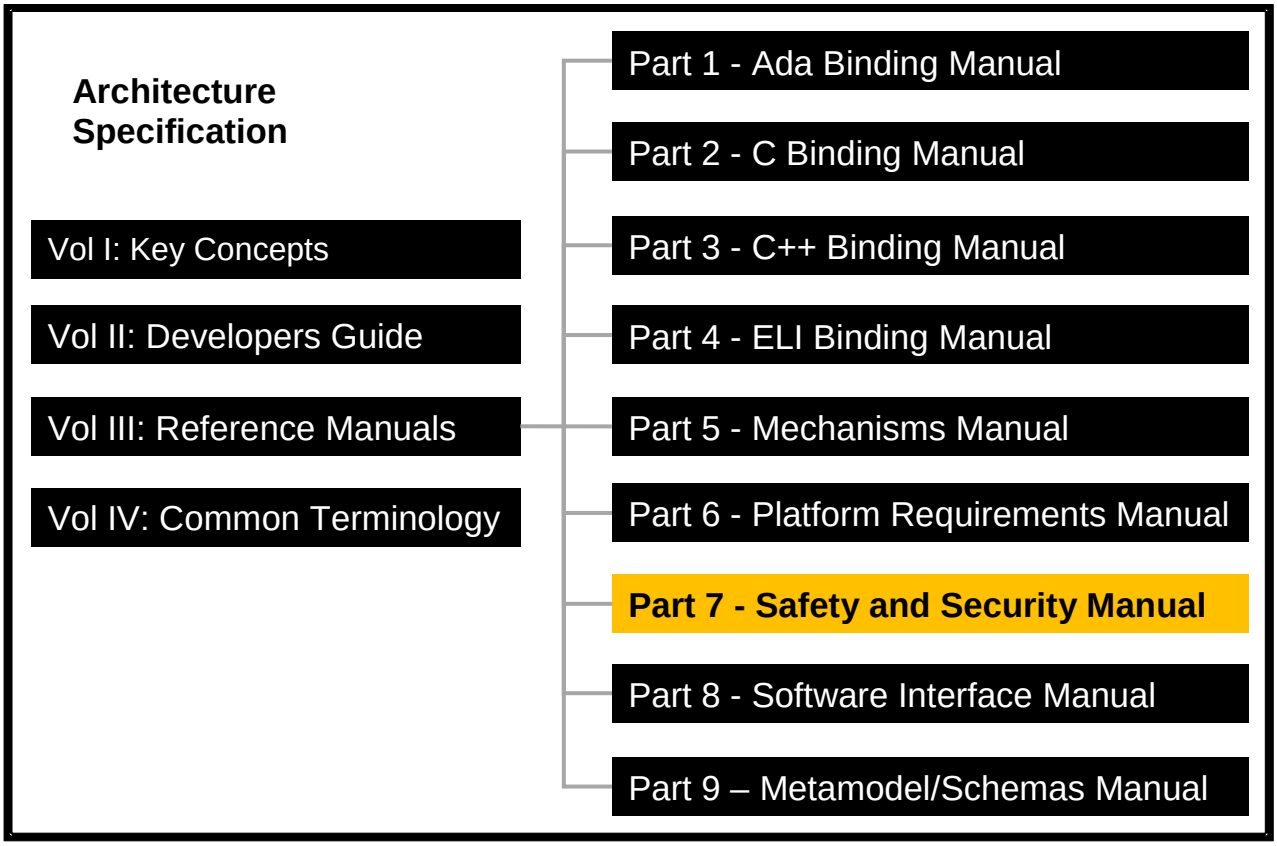

Figure 1 – ECOA Documentation

The Architecture Specification provides the definitive specification for creating ECOA-based systems. It describes the standardised programming interfaces and data-model that allow a developer to construct an ECOA-based system. It is introduced in Key Concepts (Reference AS-1) and uses terms defined in the Common Terminology (Reference AS-12). For this reason, the reader should read these documents, prior to this document. The details of the other documents comprising the rest of the Architecture Specification can be found in Section 14.

The Architecture Specification consists of four volumes, as shown in Figure 1:

- Volume I: Key Concepts
- Volume II: Developer's Guide
- Volume III: Reference Manuals
- Volume IV: Common Terminology

This document comprises Volume III Part 7 of the ECOA Architecture Specification, and contains information on how to support safe and secure reuse with an ECOA system.

The document is structured as follows:

- Section 8 describes the conceptual view of the ECOA component interface in the component catalogue entry,
- Section 9 provides some guidance about these interfaces,

- Section 10 describes the detailed view of assurance artefacts by explaining how to find safety and security-related attributes in the ECOA metamodel,
- Section 11 provides high-level guidance when considering safety and security in an ECOA system,
- Section 12 explains further work and limitations of this report,
- Section 13 provides the conclusions of this work,
- Appendix A provides a super-set of safety assurance artefacts,
- Appendix B provides a super-set of security assurance artefacts,
- Appendix C provides example XML for assurance level data.

6 Executive Summary

The European Component Oriented Architecture (ECOA) project is a collaborative research and development programme intended to improve the development costs and through-life affordability of military avionic mission systems within legacy, current and new-build platforms. The aim of the ECOA project is to define jointly, between UK and France, a real-time software architecture, which uses some aspects of service orientation, and introduces Application Software Components, (ASC), [Ref: AS-1]. These ASCs facilitate and encourage software reuse between aircraft platforms and potentially on other platforms beyond the air domain.

Success in achieving the expected benefits of cost savings from reuse will be significantly dependent upon successful safety certification and security accreditation of systems that reuse ASCs. This document proposes an approach to recording safety and security design information and assurance artefacts to support such reuse. It has resulted from analysis of the Phase 1 Stage 2 Interim standard ECOA design concepts and necessarily may need to be updated as more advanced concepts are introduced and defined, later in the programme. Limitations on use are clearly defined, predominantly an assumption that all modules within the ASCs are deployed within the same protection domain, plus mechanisms for dynamic discovery were not yet fully defined so are not considered herein, other than to currently recommend against its use for high assurance systems.

In terms of evaluating the risk for early adopters of ECOA, it is considered that this approach presents an initial simple and pragmatic approach which should be consistent with that expected by anticipated System Integrators, customers and regulators. As the research programme matures more advanced concepts and initial adopters begin to discuss certification and accreditation of deployments of the technology with their regulators, this work should be revisited and refined, as required.

7 Introduction

7.1 Purpose of the Report

This report describes what safety-related and security-related properties may need to be considered for the integration or the reuse of ECOA components. This report defines what information will be needed, and recorded, to describe safety or security related attributes to support system integration, safety certification and security accreditation. The way these properties will be defined/created or mapped/used onto/within safety or security assurance schemas is out of scope of this report, as is the process for identifying safety or security requirements for the components.

This report defines a systematic approach to providing and structuring security / safety / airworthiness assurance information, that should be used by the developers to facilitate component re-use. These elements are tracked/identified in a flexible framework with the help of the ECOA metamodel and with a versatile superset of safety and security assurance artefacts, as defined in Appendices A (Section 15) and B (Section 16).

This framework does not supersede what is requested by security/safety/airworthiness assurance processes. It proposes a method of organising the evidence data required by the regulators, rather than redefining its content. The intent of this report is to be neutral on any regulatory approach.

In addition, this report provides high level guidance to consider when deploying and re-using components from a safety and security point of view. The guidance is predominantly ECOA-specific and not intended to replicate generic safety and security guidance of which the reader should be aware. The guidance provided is as the result of an initial assessment. Some further analysis that could be undertaken to mature and refine the recommendations provided is identified.

7.2 Scope

ECOA has been initially developed for use in the air domain, particularly for unmanned air systems (UAS) and to support legacy platforms, however it is anticipated that the concept could also be used to implement functionality across different domains, (land, sea and air), and for diverse platforms and platform sub-systems.

It is a requirement to ensure that any such system is designed in a way that is acceptably safe, secure and suitably certified and accredited. It follows then that any system constructed using the ECOA software architecture should be built in a way that is acceptably safe and secure, supporting any required safety or security functions. An authority will need to be satisfied that all potential risks to a system have been appropriately treated, and that the assessed entity has balanced outstanding risks, countermeasures and risk appetite accordingly. In addition, a supplied ECOA solution may be able to provide the necessary mechanisms to support mixed safety and/or security integrity levels or to support components with differing protective markings and information exchange requirements.

The anticipated use of ECOA includes anything from non-safety or security-related to high integrity real-time applications. Safety and security controls can be realised on a platform as Technical, Physical or Procedural controls. Ultimately the controls used and their suitability for purpose must satisfy the risk appetite of the certifying or accrediting authority. The implications of the decision on use of an existing component are the responsibility of the users of the ECOA components who must justify the selection and implementation of these controls.

8 Conceptual view of an ECOA Component Catalogue Entry

Each reusable ECOA component will have an entry in the component 'catalogue' which acts as a 'data-sheet', advertising the availability of the component to purchasers who may wish to reuse existing components in their system. As such, the information contained in the catalogue entry must record information that is pertinent to safety and security, such that the purchaser can understand whether it provides the necessary functionality or behaviour needed and whether sufficient assurance information is provided to support the certification or accreditation process that will be undertaken at system level.

At an abstract level, a typical ECOA system deployment might be as shown in Figure 2.

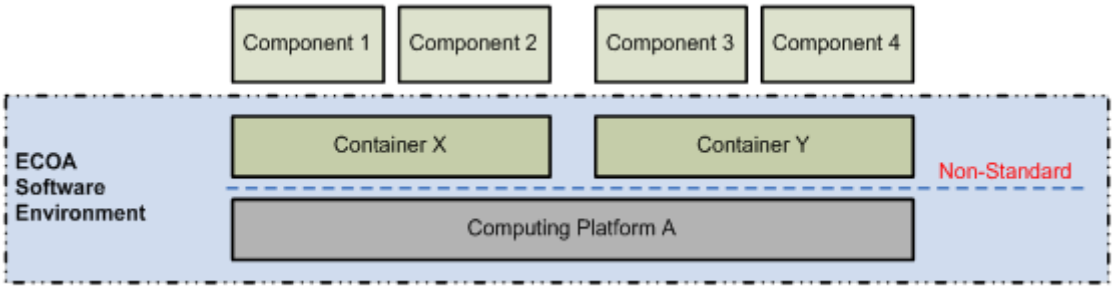

Figure 2 – Example of ECOA Deployment

That is, a number of Application Software Components, (ASCs), are deployed onto a computing platform via one or more containers. As defined in the basic ECOA concept, the interface between the container and computing platform is non-standard and so requires tailoring for each computing platform.

Where a system has safety or security requirements defined, there are expected to be properties that are relevant to safety and/or security at each interface in the architecture and these may include bi-directional interdependencies as illustrated in Figure 3.

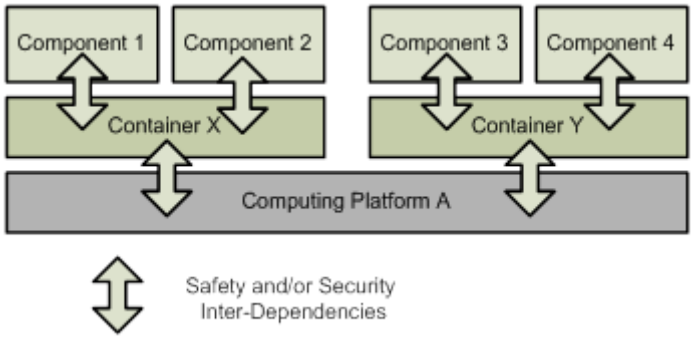

Figure 3 – Safety and Security Interdependencies

Figure 4 shows a refinement of these bi-directional dependencies. Components may have dependencies, which are standard for the component, on other components, the container or the underlying platform; these are illustrated by the (green) arrows coming down from the

components. The underlying computing platform may provide standard services whenever it is deployed; illustrated by the (green) arrow coming up from the computing platform. The container will need to 'connect' the dependencies from the components to the mechanisms which will satisfy those dependencies - either other components, within the container itself, or services within the computing platform. This matching will be unique to the specific deployment configuration and is illustrated by the (red) arrows coming from the container.

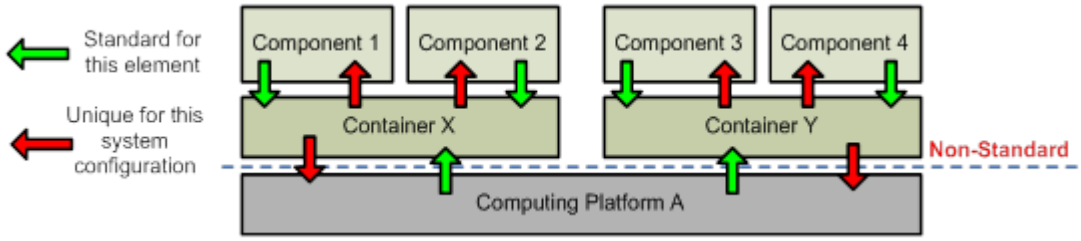

Figure 4 – Uniqueness of interdependencies

There may also be dependencies that can only be assessed at system level, E.g. resource usage, but to assess at the system level, a System Integrator may need information about each component.

Based on the above, the following conceptual view of the information necessary to facilitate Application Software Component reuse is proposed:

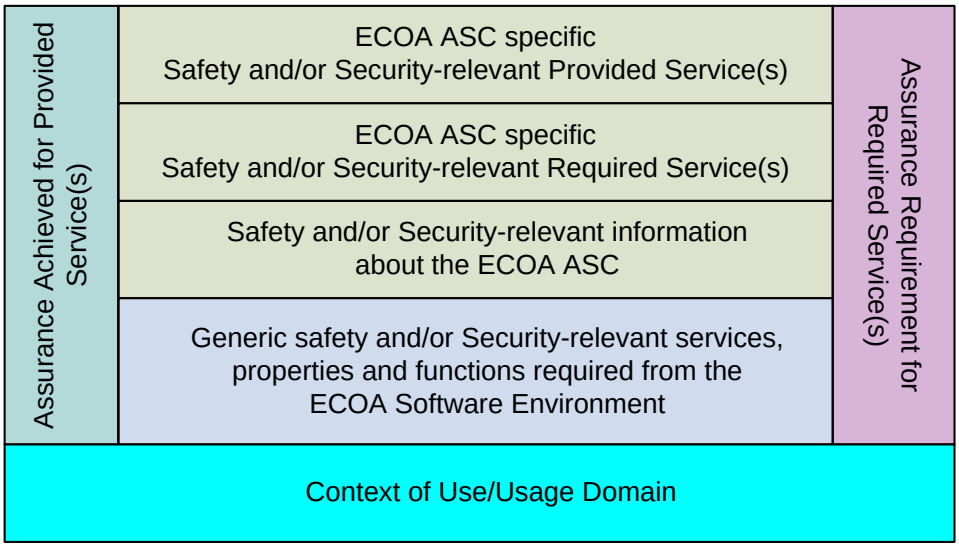

Figure 5 – Conceptual view of assurance artefacts

These are the types of information that the Component Developer will need to record for every ASC to facilitate its re-use in an ECOA system.

ECOA ASC specific safety and/or security-relevant provided service(s)

Any functional/behavioural/services that represent the safety function(s)/ property or properties *provided by* the component

ECOA ASC specific safety and/or security-relevant required service(s)

Any functional/behavioural/services that are required by the component to enable it to deliver its own safety function(s)/ property or properties

Safety and/or security-relevant information about the ECOA ASC

Any system level safety considerations, E.g. separation/independence requirements

Information about assurance evidence available for the component

Generic safety and/or security-relevant properties and functions required from the ECOA software environment

Any behaviour required from the ECOA software environment that is not explicitly covered by a required service, E.g. a particular scheduling approach, availability of memory, processor time, etc.

Assurance achieved for provided service(s)

An indication of assurance achieved for the safety relevant service(s) provided by the component – see notes on assurance measures below

Assurance required for required service(s)

An indication of assurance required for the safety relevant service(s) required by the component – see notes on assurance measures below

Context of Use / usage domain

A record of any information on the context of use of the component that is relevant to safety, or can be reasonably anticipated to be relevant to safety for future deployments.

Section 9 provides guidance on each of the safety and security-relevant interfaces defined for the ECOA Component Catalogue.

9 The Safety and Security-Relevant Interfaces of an ECOA Component Catalogue Entry

This section considers how components interact, as in Figure 6, when delivering safety and security-relevant services, behaviours, or properties, and provides additional high-level guidance on recording them in the ECOA Component Catalogue structure introduced in section 8.

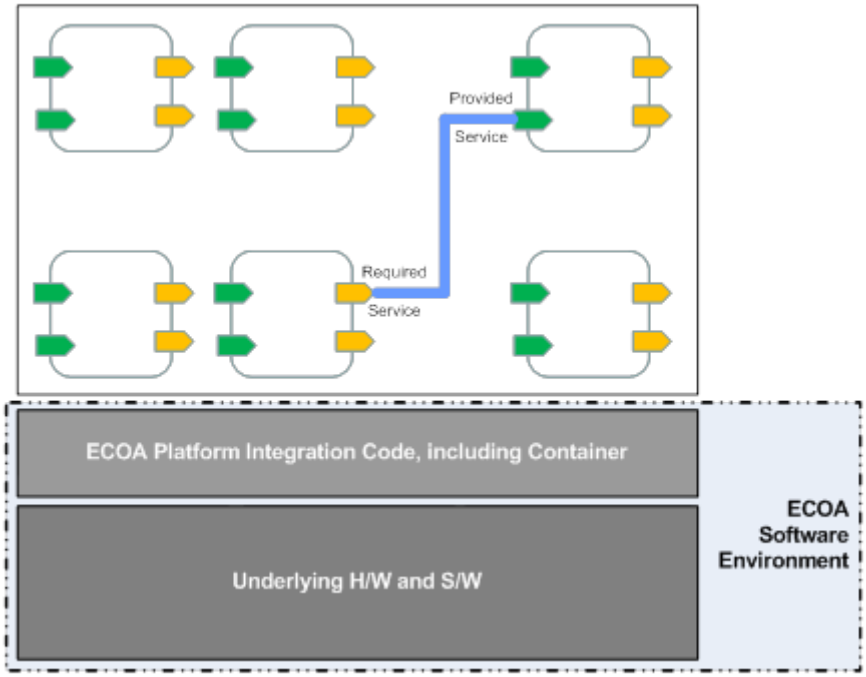

Figure 6 – Example of interactions between components

9.1 ECOA ASC Specific Safety and/or Security-Relevant Provided Service(s)

- Any function/behaviour/service that represents the safety and/or security function(s)/ property or properties *provided by* the component

During initial design, these safety and/or security properties would represent requirements placed on a component that had been derived from design-time system safety analysis or security risk assessment, such as HMG IS 1&2 [Ref: IS1&2]. Where a component is re-used, the System Integrator will need to ensure that the services declared for the component meet the needs of their system safety or security implementation strategy, as claims and assurances made on a component in one system may not be valid in a different concept of use.

> *For example, provision of positional information, to a required accuracy, and refreshed at a specified rate, may be a service that is provided by the component*; *or a service providing Encryption may need to use a different algorithm or operate to a different standard in a new environment in which is reused.*

9.2 ECOA ASC Specific Safety and/or Security-Relevant Required Service(s)

- Any function/behaviour/service that is *required by* the component to enable it to deliver its own safety or security function(s)/ property or properties

During initial design, these safety and/or security properties would represent requirements identified during the design and safety/security analysis of the component where support from outside the component is required so that the component can provide it declared services. Where a component is re-used, the System Integrator will need to ensure that the services required by the component can be provided by their implementation strategy.

> *For example, the provision of positional input data to the component from a range of different sources, such as GPS, inertial navigation, etc, with associated accuracy, update rates, etc., would be required for the component to provide consolidated positional information which complies with the declared criteria; or a Service providing Encryption functions may need a service from another Component to provide an algorithm seed.*

9.3 Safety and/or Security-Relevant Information about the ECOA ASC

- This section of the catalogue entry provides an opportunity for the Component Developer to:
    - record any additional constraints or assumptions about the safe and secure reuse of their components, such as permissible mappings of the component modules onto protection domains
    - provide a record of the design and assurance data sets available to support the component

To encourage a consistent approach to recording supporting documentation, 'supersets' of typical data items are recorded in Appendices B (Section **15**) and C (Section **16**). This structuring will be used in the component entry form that is completed when a component is registered with the ECOA Agency. It is intended that the component supplier will complete the following checklist, indicating whether a specific document, (or equivalent):

- is available as part of the baseline documentation set;
- could be generated if part of a customer contract; or
- could not be generated, (for example, some documents cannot be created retrospectively).

A very detailed list of specific documents has been generated to support safety assurance documentation, based on collating those from common standards. There is less consensus in the security domain, however, and so the list in Appendix B, (Section 16) only contains classes of document, as proposed by Common Criteria [Ref: CC]. It may be possible to refine this list in the future.

9.4 Generic Safety and/or Security-Relevant Properties And Functions Required From The ECOA Software Environment

- Generic safety relevant properties and functions required by the component from the ECOA Software Environment, i.e. container, any other platform integration code and the computing platform. Any behaviour required from the software environment that is not explicitly covered by a required service, E.g. availability of memory, processor time, etc.

These safety and/or security properties typically represent 'generic' properties of the ECOA Software Environment that are required by all services and so may be difficult to identify as a property that is specific to any individual service. To facilitate reuse, however, it is important

that the System Integrator can understand these supporting requirements for a component. It may be necessary to, for example, supply benchmarking data about timing behaviour, given a specific processor speed.

> *For example, the positional information component may require a particular approach to scheduling to be employed or a particular memory allocation or processor slot availability to deliver the necessary accuracy and timeliness of data.*

9.5 Provided and Required Assurance

- An indication of assurance achieved for the safety and/or security-relevant service(s) provided by the component and required for the safety and/or security-relevant service(s) required by the component

There are no universally agreed measures of safety or security assurance. Each standard or regulatory environment uses different approaches to achieving system safety and security which are typically either:

- process-based, i.e. they classify safety or security scenarios and mandate detailed processes and assessment measures which must be used for each scenario; or
- product/goal-based, i.e. they require a System Integrator to make a claim about the acceptability of the residual risk in the system, giving them the opportunity to describe their own processes and measurement criterion.

In reality, there is often significant overlap in the processes selected in comparable cases and the assurance evidence artefacts generated. The System Integrator may organise these evidence artefacts very differently to support their safety and security submissions to their regulators, but the detailed evidence content is likely to be reusable, with caution. This is the basis for requiring a Component Developer to record their provision of evidence artefact from the 'superset' of documents described in Appendices B (Section **15**) & C (Section **16**).

As part of the identification of safety or security requirements on a system, an assurance or integrity requirement will also be identified, indicating the level of risk that the service, function or property presents and hence the level of confidence required in the correct behaviour of the deployed component. In DO-178B/C, [Ref: DO-178], these are represented as Development Assurance Levels, DALs, or in IEC61508, [Ref: IEC], they are Safety Integrity Levels, SILs. DEF STAN 00-56, [Ref: DS-56], does not mandate a measure, but offers 'high/medium/low' as a potential option. The higher the assurance/integrity requirement, the more rigorous the design processes should be and the greater the evidence required to support certification. This is true of all the standards, however, as the approaches are not the same for each standard, making direct assumptions about equivalence is inadvisable. The System Integrator needs to assess whether a claim in one assurance environment is sufficient to address an assurance requirement in another assurance environment. A framework is proposed below which allows the assurance requirement/achievement and assurance environment to be recorded for each component, to allow the System Integrator to make such an assessment.

9.5.1 Framework for Provided and Required Assurance

This framework is intended to declare the achieved and required assurance of services provided and required by a component such that a System Integrator might assess compatibility. This is reflected in the Assurance Level Data XML in Appendix D, (Section **17**).

- Software Assurance – a statement of assurance achievement claimed / requirement
  - E.g. DAL A; SIL 3; EAL 5
- Assurance Classification Type – a definition of the measurement scheme for the above

- E.g. DO-178C; IEC61508; Common Criteria; project-specific

NOTE: System Integrators should not accept claims of assurance achievement from component development without investigating the necessary supporting evidence.

9.6 Context of Use / Usage Domain

- This section of the catalogue entry is used to record any assumptions about usage of the component that could be relevant for reuse. This will be especially important where components could be used cross-domain.

  For example, the update rate of positional information to support a safety property may be comparable between fixed and rotary winged manned aircraft but, though the functionality may be comparable, the update rate required for a naval ship may be significantly less. Or a component may have been developed on the assumption of being a single channel of a dual-redundant system implementation and be unsuitable for use in a functionally equivalent, but single channel system.

Of particular concern is where assumptions about system implementations or acceptability of residual risk have been determined by a regulator which may not be acceptable to a different regulator or regulatory environment. Similarly, some standards require testing to be undertaken by staff who are independent from the design team and this may not be readily determined from a simple test report. Recording such issues provides the best opportunity for a System Integrator to support the requirements in the new system context.

This would also be the appropriate place to indicate issues such as whether the component's behaviour is dependent upon an assumed level of protection against non-interference from other components. This gives the System Integrator the option to protect the component when deployed, perhaps through use of partitioning or by being deployed on a separate computing resource.

10 Detailed view of the Assurance Artefacts

Component meta-information covers all data required for assessments (safety, security, etc.) and technical uses (deployment, etc.) of the component.

The component meta-information is defined and refined during the development lifecycle. To explain this process, the following main stages are considered: specification, development, deployment, qualification.

In particular, the component meta-information for a single instance of a component is captured by the following artefacts:

- The component specification, described in section 10.1,
- The component implementation, i.e. the logical architecture of the component (comp.impl.xml), described in [Ref: AS-11],
- The Technical Insertion Requirements, i.e. the resource requirements, described in section 10.2,
- Information about assurances, described in section 10.3.

It is important to note that, internationally, for military Programmes, there does not exist a unique assessment process or a unique assurance schema applicable by all industry partners. As a consequence, it is not possible to provide a single description of elements for assessment process or assurance schema. For this reason, the assurance level data file has been defined to be flexible and to support several methodologies.

10.1 Component Specification

The component specification needs to address all usual concerns of a software item: the interfaces, the functional behaviour, the non-functional requirements and the insertion constraints. These concerns can then be concretely stored in many ways.

From an ECOA concepts point of view, the static part of the interfaces is described in XML with the help of Type Definitions, Service Definitions and the Component Definition. The dynamic part of the interfaces is described in XML with the help of the service QoS (abstract and concrete models in §6.1 and §6.2 of the Architecture Specification, [Ref: AS-11]), the service behaviour and the component behaviour.

Behaviours can be described with tools like statecharts, sequence diagrams, etc.

The functional behaviour and non-functional requirements can be described in several ways: text, model, requirements, etc. Formalisms are left open and best practices can be used at this level. For the sake of this report, this part is called the “SRS” (Software Requirement Specification).

Insertion constraints are provided through a partial technical insertion policy also called specifying technical insertion policy.

10.2 Technical Insertion Policy

The Technical Insertion Policy contains information or link to external information to constrain a development or to allow the technical insertion on top of an ECOA platform.

The table below is a first attempt to list all elements included in a technical insertion policy. Some are relevant at component-level while other only concern modules. The exact content and the actual location of the information will ultimately be described in the Architecture Specification [Ref: AS-11].

Item	SubItem	Level of Scope	Values	Comments / Actual location
processorType		component	text	component bin-desc.xml file - (Ref:[AS-11])
registerSize		component	32|64 bits	component bin-desc.xml file
memory				
	userContextSize	module	XXX kB	component bin-desc.xml file
	userContextAutoSave	component	yes|no	(see fault management) - Could also be a service link towards a reliable data server
	dynamicMemoryAllocation	module	yes|no (default = no)	
	dynamicMemorySize	module	XXX kB	if dynamicMemoryAllocation == yes
	stackSize	module	XXX kB	component bin-desc.xml file
	heapSize	module	XXX kB	component bin-desc.xml file
realtimeCharacteristics		module	moduleInstance in comp.impl.xml	All information required for scheduling. They can be retrieved by using the start element moduleInstance in comp.impl.xml
timePrecision		component	TBD	
transportProtocolRequirements		component	TBD	To define expectations about transport protocol reliability
deploymentConstraints		component		

	maximumSafetyCriticality	text	See Assurance Level Data The maximum safety criticality associated to that component Most of the time it is sufficient to consider one single level since one component is only part of the core mission application
	maximumSecurityCriticality	text	See AssuranceLevelData The maximum security criticality associated to that component Most of the time it is sufficient to consider one single level since one component is only part of the core mission application.
	colocatedModules	yes|no	to allow minimizing exchange latencies within the same protection domain. Can be useful to know this information for unit testing on the target
	extensive description	deployment file	If needed, a kind of required deployment file set 1: list of modules in protection domain 1 on node 1 safetyCriticality securityCriticality set 2: list of modules in protection domain 2 on node 1 set 3: list of modules in protection domain 3 on node 2 etc Normally, if modules can be all colocated, there is only one set: set 1: list of all modules in protection domain 1 on node 1
serviceAvailabilityAPIAccess	component	yes|no	To allow generation and use of service Availability API at supervision modules level
logLevel	module	See deployment file	Initial log level defined by the supplier to help deployment
internalExchangeCyphering	component	yes|no	to require cyphering exchange within the component. Outside the component, it is the System Integrator's choice that can be stored at assembly|deployment level.

languageRuntime	module	C|C++|Ada	moduleImplementation in comp.impl.xml
languageLibraries	module	math|graphics|etc	Library dependencies
libraryVersion		text	
osAPIAccess	module	yes|no (default=no)	
osAPI	module	POSIX|A653|ASAAC|FACE	if osAPIAccess == yes
compiler	component	text	textual description of the compiler
CompilerOptions	component	text. E.g. "-fpic"	Applicable to all modules
specificCompilerOptions	module	text. E.g. "-x"	Specific to one module implementation binary
extraConcerns	component	ad-hoc document	Elements that cannot be covered under a generic form

Table 1 – Preliminary Technical Insertion Policy

10.3 Assurance Level Data

The assurance level data summarises basic assurance information and provides links to external evidence that contains detailed assurance data. External evidence can be any of the artefacts identified for a particular assurance schema, (see Appendices B (Section **15**) and C (Section **16**)).This approach supports coping with multiple assurance schemas.

The abstract content is the following:

- General information on the component implementation (supplier identification, etc.)
- Safety or Security assurance level associated with the component or individual modules of the component (development standard and level)
- Safety or Security properties offered by the component with their associated evidence and insertion constraints.
- List of evidence to justify the level of assurance reached by the implementation

A concrete XML example is provided in Appendix D, (Section **17**).

10.4 Artefacts Initialisation Route

Figure 7 shows, from a component supplier point of view, the main flow of data between development steps. It defines for each step which data are created or refined.

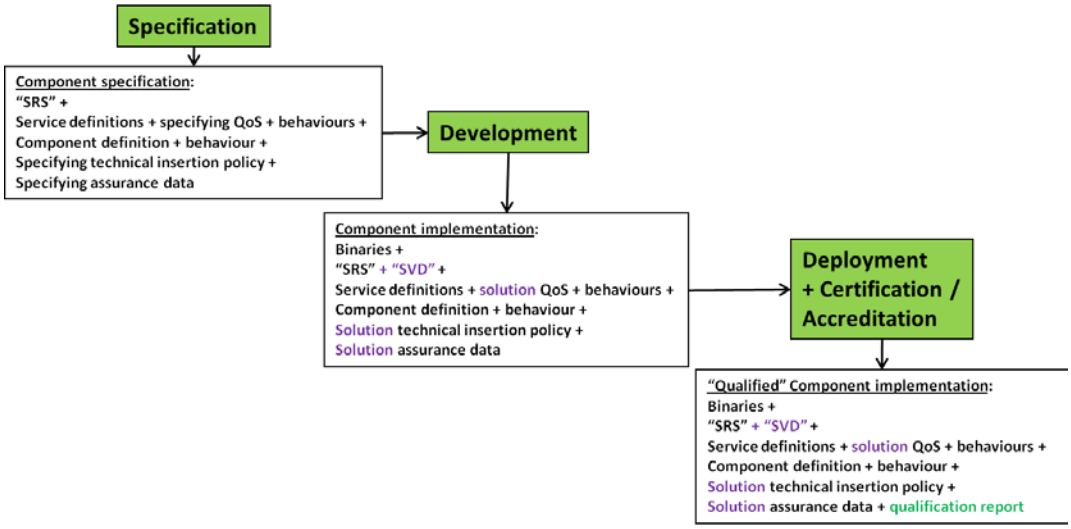

Figure 7 – Flow of Information between component development steps

In the figure above, "SRS" and "SVD" stand respectively for Software Requirement Specification and Software Version Description. The SRS describes functional and non-functional requirements while the SVD recaps all items describing a component implementation. The exact content of these documents is not prescribed by ECOA concepts.

11 High level guidance

This section provides general high-level guidance on providing evidence supporting the fulfilment of safety and security relevant services, behaviours or properties. The availability of any particular types of evidence would be indicated in the component catalogue as described in Section 9.3.

11.1 "Business as Usual"

Predominantly, the development of safety and security relevant software should follow the existing well-established and well-understood practices already used for the development of such software. It is not an objective of ECOA concepts to radically alter these practices – just to make it practical to identify and exchange available information effectively when software is re-used.

The fundamentals of these practices are:

- Effective **configuration control** – it must be precisely clear what the version of any artefact (plan, requirement, design, code, test case, test result, etc.) is when it is being referred to. Without this there can be no confidence as to what any particular piece of evidence relates to and this would undermine any assurance being sought. The configuration control process would combine robust change control management with detailed record keeping and secure storage of artefacts (to prevent unwarranted alteration).
- **Planning** – all activities should be conducted in accordance with an agreed plan. The plans might also identify particular processes that are to be followed to complete specific tasks.
- **Traceability** – the relationships between artefacts must be clear so that any external assessment of the safety and security considerations can freely navigate between top level requirements, the design, and the evidence that demonstrates compliance.

The detailed way in which these practices are undertaken is likely to be affected by the levels of safety assurance and security accreditation being sought, and the regulatory authority to be satisfied. The ECOA catalogue provides a means (see Section 9) of recording the nature and extent of what has been done in such a way that relevant safety and security information can be readily re-used in a different context.

> *Note: The assessment as to whether any particular item of information remains valid from one context to another is outside the scope of ECOA. For example, the ECOA catalogue might record that a component has a plan that considers the software aspects of certification – it does not provide details of what that plan is. If the component is also recorded as meeting a particular design assurance level this would imply that the plan was adequate (at the time and for the circumstances). However, it remains for the new user of the component to determine whether this implied adequacy is sufficient or not; even if it is not accepted at face-value the new user has the benefit of knowing that a plan existed and can consider reviewing how the component was developed.*

11.2 Component

Components have four distinguishable lifecycle stages which need to be considered:

- **Specification** – establishes the interfaces of the component (its provided and required services together with configurable properties and insertion policies). Where there is an identifiable safety and/or security consideration associated with an aspect of the interface this needs to be clearly identified.

  The behaviours expected and required of services, together with their QoS, need to be precisely defined. Where a configurable property is provided there needs to be adequate information to allow a user of the component to determine the effects of the property and to be able to accurately determine an appropriate value for it. Other information, relevant to satisfying safety and security considerations when the component is used, need to be recorded in the insertion policy.

- **Development** – establishes an implementation of a component using features of the ECOA framework and bespoke software for the component. Where features of the ECOA framework are used any safety and security considerations that it needs to satisfy need to be identified and recorded as part of the implementation detail.

- **Deployment** – needs to consider the issues of integrating a component onto a platform. This will include ensuring all relevant information is provided to the System Integrator in a clear manner. Information that is not recorded as part of the QoS should be recorded in the insertion policy.

- **Certification/Accreditation** – it is anticipated that components will not normally be certified/accredited in their own right as this is normally a system level consideration. However, evidence produced as part of the development of components would normally play a significant role in achieving system certification. In addition to demonstrating the inherent functionality of the component (in particular any functionality associated with safety and security considerations), there would be a need to demonstrate its correctness with respect to the ECOA framework features used in its implementation (typically accessed via the container API) and the ECOA component lifecycle model. The correctness of the component would also need to be established across the defined ranges of QoS and any configuration settings that can be set through the component properties.

  Where insertion policies allow flexibility there needs to be evidence that all possible deployments satisfy the safety and security considerations. Where a System Integrator needs to validate particular properties on their system this needs to be clearly identified.

Generally, where safety and security considerations are a concern it is advisable to keep matters simple. Supporting variation typically adds certification and accreditation costs and so needs to be justifiable by the benefits it offers. Similarly, using some features of the ECOA framework may introduce additional complexity into obtaining certification/accreditation. Such pitfalls can usually be avoided by having a well-defined plan for meeting the standards needed for certification/accreditation.

11.3 ECOA Software Environment

Containers are involved in the system's delivery of safety and security related services and behaviours provided by the applications as they provide:

- the mechanisms for mapping to the transport protocols offered by the computing platform,
- the scheduling of those components
- some technical functions/services that may contribute to safety and security - such as the timing service.

In addition, a platform may itself provide generic safety and security related functions and behaviours as part of the computing infrastructure the component and container code runs on (E.g. partitioning of processor memory, processor time etc.). Some of these features might be relied upon by the container software to deliver its safety and security related features; alternatively they might be directly accessed by components that are intended to be deployed on specific types of platform.

Components will be responsible for identifying where they rely on such platform/container features. The platform/container providers will be responsible for providing supporting evidence that platforms/containers adequately support the safety and security features they claim to provide (in a similar fashion to component suppliers). System Integrators will be responsible for ensuring component needs are matched by the selected platform/container features, and providing any additional evidence needed to demonstrate that components operate correctly, with respect to the safety and security considerations, on the selected platform.

11.4 Integration

The System Integrator will need to establish that safety and security considerations are satisfied where:

- one component requires services and behaviours of another component;
- a component implementation requires the platform/container to respect or implement a service or behaviour;
- a component implementation has an insertion policy that needs to be honoured by the system deployment;
- a component requires specific verification activities to be carried out on the target system.

This would include verifying that functional and QoS definitions are compatible; that configuration properties are appropriately set, with traceability of how the settings were determined and evidence that supports the claim that they have been correctly set; and evidence that component insertion policies have been properly implemented.

The System Integrator will also need to verify that all system related requirements have been fulfilled by the given deployment onto a specific platform and establish the overall safety and security credentials of the system. This will include determining what supporting information from component and platform/container suppliers needs auditing and where necessary commissioning additional safety and security assessments/evidence.

The System Integrator will need to verify that system behaviour is correct and that safety and security considerations are properly addressed across all stages of the system lifecycle – but notably during initialization, normal operation, reconfiguration, fault handling, and shutdown.

12 Further Work/Limitations

12.1 Limitations

The proposals in this document are based on Interim standard Documents from Phase 1 Stage 2 of ECOA project. As such, some concepts are yet to be matured and so this work is not able to make recommendations on those issues and will need to be revisited when the concepts are finalised. In particular:

- This report assumes that all modules that make up a deployed Application Software Component are within the same protection domain and container. The Architecture Specification, [Ref: AS-1], allows for deployment of modules in different protections domains, and on different computing nodes (belonging to the same platform). The supporting mechanisms explored to date are platform-specific, hence it is not possible to give generic guidance. Currently it is recommended that a component instance is deployed in a single protection domain and that any assurance evidence is generated from testing undertaken using the deployment of module instances to be used in the final system. It may be possible to provide additional generic guidance in the future.
- Dynamic discovery has not been fully defined or matured within the design documentation set so the general impact on safety and security cannot be adequately assessed. It is noted that certification will be significantly more challenging to achieve with systems that exhibit dynamic discovery and more so for systems with higher assurance requirements. A pragmatic solution is to avoid use of dynamic discovery in high assurance systems, but it may be possible to provide additional guidance once this concept is mature.
- Domain (E.g. UAS) specific reference architectures providing a baseline set of extended types and service definitions, together with supporting data dictionaries, have not been fully defined or matured. Nor has the interaction and re-usability of type and service definitions (with data dictionaries) developed for specific systems been examined. Consequently, these definitions will initially need to be managed as part of each system development – including any re-use between programmes. These definitions represent an important part of the safety and security context of a Component so their appropriate management is a factor in the reusability of assurance evidence. As the reference architecture concept matures it will be necessary to review the guidance given here.

12.2 Further Work

It is proposed that this work should be revisited in future phases of the ECOA programme to determine any impact from maturing the ECOA concepts on the recommendation here. In addition, some safety and security-specific work should be undertaken:

1) A case study, as close as possible to a deployable aircraft standard, to enable and support discussions with relevant regulators, providing maturity and refinement to the approach and additional practical guidance.
2) Refinement of the content and representation of safety and security behaviours and properties in the ECOA Component Catalogue

3) Consideration of a more rigorous approach to recording non-functional behaviours, including safety and security, to facilitate analysis, including automated checking for compatibility

13 Conclusions

The purpose of the report is to ensure that the adoption of the ECOA standard in developing software for a military platform would not present unnecessary or unacceptable safety or security risk or cost to the project or in-service platform and its stakeholders. The aim is to provide early visibility and confidence to the customer about any potential safety or security concerns for ECOA that might limit or constrain the scenarios for which it could be deployed.

The analysis reported in this document was undertaken against a baseline of the Interim ECOA documents [Refs: AC, CT & AS], which is the current stable document set.

Whereas traditional safety and security analyses are undertaken against a specific system design and implementation, ECOA is a software architecture definition and hence the strategy for the analysis reported here was to determine whether the introduction of ECOA would specifically support or, probably more importantly, prevent the satisfaction of typical safety and/or security requirements.

ECOA concepts introduce unique considerations, but along with the guidance detailed in this report, it is considered that ECOA is suitable for developing systems which implement safety and security properties, including those with mixed safety and security assurance/integrity requirements.

The guidance provided herein is as the result of an initial assessment. Some further analysis that could be undertaken to mature and refine the recommendations provided is proposed in Section 12.2

The framework proposed for safe and secure reuse using ECOA is also a potential starting point for facilitating transferable safety between domains E.g. rail, automobile, military & civil aerospace.

14 References

Ref.	Document Number	Version	Title
AS-1	IAWG-ECOA-TR-001	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume I Key Concepts
AS-2	IAWG-ECOA-TR-002	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume II Developers Guide
AS-3	IAWG-ECOA-TR-003	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 1: Ada Binding Reference Manual
AS-4	IAWG-ECOA-TR-004	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 2: C Binding Reference Manual
AS-5	IAWG-ECOA-TR-005	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 3: C++ Binding Reference Manual
AS-6	IAWG-ECOA-TR-006	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 4: ELI and Transport Binding Reference Manual
AS-7	IAWG-ECOA-TR-007	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 5: Mechanisms Reference Manual
AS-8	IAWG-ECOA-TR-008	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 6: Platform Requirements Reference Manual
AS-9	IAWG-ECOA-TR-009	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 7: Approach to Safety and Security Reference Manual
AS-10	IAWG-ECOA-TR-010	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 8: Software Interface Reference Manual
AS-11	IAWG-ECOA-TR-011	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 9: Metamodel/Schemas Reference Manual
AS-12	IAWG-ECOA-TR-012	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume lV Common Terminology

Table 2 - Table of ECOA references

Ref.	Document Number	Version	Title
CC	CC v3.1.	Release 4 (latest)	Common Criteria
IS 1&2	HMG IS 1&2	Issue 1, 2012	HMG IS Standard Numbers 1&2 Technical Risk Assessment and Risk Treatment Issue 1.0, 2012.
DO-178	DO-178	B and C	Software Considerations in Airborne Systems and Equipment Certification
DS-56	Defence Standard 00-56	4 and 5	Safety Management Requirements for Defence Systems
IEC	IEC61508	Edition 2	Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems

Table 3 – Table of External References

15 Appendix A – Superset of Safety Assurance Artefacts

The following list is a 'superset' of documents typically used in software development programmes and has been derived from collating documentation and evidence requirements from common software and software safety standards. It is NOT anticipated that all programmes will generate all documents. It is anticipated that the documents produced will vary in accordance with the assurance/integrity requirements that apply to the software being developed and the processes and standards that are used. These are indicative document titles; descriptions of content will be provided as guidance for those completing the component registration process.

1 System Artefact Hierarchy

- System Definition
  - System Safety
    - System Hazard Log
    - Software Integrity Requirements
  - System Management
  - System Requirement
    - System Interface Requirements Specification
    - System Requirements Specification
    - System Requirements
    - System Interface Requirements
  - System Design
  - System Implementation
  - System Verification and Validation

2 Software Artefact Hierarchy

- Software Definition
  - Software Management
    - Software Plan
      - + Software Development Plan
      - + Software Tool Qualification Plan
      - + Software Verification and Validation Plan
  - Software Process
  - Software Procedure
    - Software Specification Code of Practice
    - Software High Integrity Design Code of Practice
    - Software Automated Implementation Code of Practice
    - Software Automated Dynamic Analysis Code of Practice
    - Software Automated Static Analysis Code of Practice
    - Software Automated Reverse Engineering Code of Practice
  - Development Repository
  - Tool Selection Criteria
  - Automation Strategy

- Automated Implementation Strategy
- Automated Static Analysis Strategy
- Automated Dynamic Analysis Strategy
- Automated Reverse Engineering Strategy

o Software Safety
- Software Safety Plan
- Development Process Hazard Analysis
- Software Defect Trend Analysis
- Software Safety Records Log
  + Safety Arguments from Testing
  + Analytical Safety Arguments
  + Development Process Related Safety Arguments
  + Tool Related Safety Arguments
  + Software Safety Case

o Software Requirement
- Software Requirement Hazard Analysis
- Software Requirement Models
  + Requirement Model
  + Requirement Simulation
- Software Requirements Analysis Model
- Software Requirements Traceability
- Software Requirements Record (Reviews)
- Software Design Requirements
  + Module Software Requirements
  + Software Requirements
- System Specification Refinement

o Software Design
- Software Architecture Model
- Software Design
  + Design Model
  + Design Simulation
- Software Design Traceability
- Software Design Record (Reviews)
- Dynamic Analysis
  + Software Module Dynamic Analysis
    - Software Module Dynamic Analysis Tests
    - Software Module Dynamic Analysis Traceability
    - Software Module Dynamic Analysis Objectives
    - Software Module Dynamic Analysis Test Results
    - Software Module Dynamic Analysis Record (Reviews)
  + Software Dynamic Analysis
    - Software Dynamic Analysis Tests
    - Software Dynamic Analysis Traceability
    - Software Dynamic Analysis Objectives

- Software Dynamic Analysis Test Results
- Software Dynamic Analysis Record (Reviews)

- Static Analysis
  - + Static Analysis Checks
  - + Static Analysis Results
  - + Static Analysis Record (Reviews)
- Software Requirement Refinement

o Software Implementation
- Automated Software Code
- Manual Software Code
- Software build
- Software Design Refinement (IMPL)
- Software Implementation Traceability
- Executable Software Load
  - + Host Software Executable
  - + Target Software Executable
- Legacy Software Code

o Software Verification and Validation
- Software Hazard Log
- Software Static Analysis
  - + Software Static Analysis Checks
  - + Software Static Analysis Results
  - + Software Static Analysis Record (Reviews)
  - + Legacy Static Analysis
  - + Software Design Refinement (STATIC)
- Software Dynamic Analysis
  - + Software Automated Dynamic Analysis Tests
  - + Legacy Dynamic Analysis
  - + Software Automated Dynamic Analysis Test Results
  - + Software Design Refinement (DYNAMIC)
- Software Verification Cross Reference Matrix (VCRM)
- Software Design Revision

16 Appendix B – Superset of Security Assurance Artefacts

The following section is based upon the internationally recognised Common Criteria for Information Technology Security Evaluation. It allows suppliers and Platform Integrators to specify their security functional and assurance requirements, allowing them to make claims about the security aspects of their components.

The Target of Evaluation (TOE)

The target of evaluation can consist simply of a single component or be as complex as a set of components complete with an Operating System and hardware. It is crucial that suppliers and integrators have a clear depiction of the Target of Evaluation to ensure that evaluation is not misrepresented in situations in which only select components have undergone evaluation.

This is of particular importance in an ECOA system in which components can be deployed and reconfigured. All possible configurations must be considered for any evaluation to be valid.

The component supplier should consider the potential usage and deployment scenarios of their software, and determine the appropriate level of assurance artefacts to supply.

The Security Assurance artefacts can be broken into several main sections consisting of Development, Guidance Documentation, Lifecycle Support, Security and Target evaluation, Tests and Vulnerability Assessment. This list is intended to provide initial high level guidance and is not exhaustive. . It is NOT anticipated that all programmes will generate all documents. It is anticipated that the documents produced will vary in accordance with the assurance/integrity requirements that apply to the software being developed and the processes and standards that are used.

1. Development

Security Architecture Description

The objective is for the supplier to provide a description of the security architecture of the component. This will allow analysis of the information which, when coupled with other evidence, will support the claim that analysis of the component can be achieved by examining the security functionality.

Complete Semi-formal functional specification with additional formal specification

The supplier will produce a functional specification and a formal presentation of the functional specification of the component security functionality. This shall include all error messages contained in the implementation representation.

Complete Mapping of the implementation representation of the security functionality

The supplier shall make available the implementation representation for the entire component security functionality. This should include a mapping between the design description and the entire implementation representation. This should be in a form used by development personnel.

Minimally Complex internals

Supplier should ensure that components are well-structured and of minimal complexity. The intent is that components are designed and implemented using sound engineering principles.

Well-defined development tools should be used, and evidence should be provided in the form of a component's internals description and justification.

Formal Target of Evaluation security policy model

The supplier should provide a formal security policy model for the components. Proof of correspondence between the component and any formal functional specification should be supplied. Where possible this should be clearly demonstrated, supported by explanatory text as required. This should be modelled such that security for the component is well-defined. Formal proof should be supplied to demonstrate the component cannot reach a state that is not secure.

Complete semiformal modular design with formal high-level design presentation

The supplier shall provide the design of the components. This should include a mapping from the highest level of the functional specification to the lowest level of decomposition available in the design. This should include all the subsystems and a description of the interactions among all subsystems.

2. Guidance Documentation

Operational User Guidance

This refers to written documentation that is intended to be used by all types of users of the components. User guidance should include how to maintain and administer components in appropriate configurations for maximum security and to allow Platform Integrators to use the component interfaces. Guidance should be clear, concise and coherent, highlighting any secure procedures required for all modes of operation.

Preparative procedures

Preparative procedures are intended to ensure that components are received and installed in a secure manner as intended by the developer. This helps to prevent accidental misconfiguration and ensures that components are appropriately delivered and installed. Suppliers should provide a full set of preparative procedures documenting all the steps necessary for a secure installation.

3. Lifecycle Support

Advanced Support

Well-proven configuration management tools should be used to clearly demonstrate that configuration items are maintained in a controlled manner. This ensures that unauthorised modifications cannot be made to components and their integrity is suitably preserved.

Development Tools Configuration Management Coverage

Supporting evidence, evaluation evidence and security flaw reports should be placed under configuration management in the same way as components to ensure they are suitably controlled. This prevents unauthorised document modification and enabled developers to track security issues within components.

Delivery Procedures

Suppliers should document and provide procedures for the delivery of components to the Platform Integrator. This should describe all procedures that are necessary to maintain security when distributing versions of the software to the integrator.

Sufficiency of security measures

The supplier should produce and provide development security documentation. This shall describe all the physical, personnel and other security measures that are necessary to protect the confidentiality, integrity and availability of components through design and implementation to its deployment environment. The documentation should justify that the security measures provide the necessary level of protection.

Measurable life-cycle model

Component suppliers should establish a life-cycle model to be used in the development and maintenance of components and systems that is based on a measurable life-cycle model. This should be supported by life-cycle definition documentation.

Compliance with implementation standards

Components should be provided with documentation identifying each development tool being used for development. This should be documented and selected implementation-dependent options of each development tool provided unambiguously. All conventions, directives and statements used in the implementation should be well-defined.

4. Security Target Evaluation

Conformance Claims

Suppliers should supply a conformance statement along with a conformance claim and accompanying rationale for components. This should include an extension of a component's definition and stated security requirements.

Extension of a Component's Definition

These are requirements that are not directly based on components but are based on extended definitions for a component. Suppliers should provide a statement of security requirements and an extension of a component's definition. These extensions should consist of measurable and objective elements such that conformance or non-conformance to these elements can be clearly demonstrated.

Security Target Introduction

Suppliers should provide a security target introduction, identifying the target of evaluation and describing the physical and logical scope. This should include a full description of the usage concepts and major security features of the component/system.

Security Objectives

The security objectives should be a concise statement of the intended response to the defined security requirements. This enables a supplier to demonstrate that the component/system security objectives have been adequately met and completely addressed.

Derived Security Requirements

Derived security requirements should be supplied in the form of a statement of security requirements. This should describe all subjects, objects, operations, security attributes, external entities and other terms that are used.

Security problem definition

This involves definition of the security problem to be addressed by the component/system. Evaluation of the security problem definition is required to demonstrate that it is addressed by the component/system in its operational environment. This should include the threats to the component/system and any assumptions about the operational environment.

Target of Evaluation summary specification

The summary specification is intended to enable evaluators and potential component integrators to gain a general understanding of how the component is implemented. Suppliers should supply a summary specification to confirm that the component meets its stated security requirements and basic functional specification.

5. Tests

Rigorous analysis of coverage

The supplier should provide evidence of how testing undertaken corresponds to the components functional specifications. The objective is to confirm that the supplier has undertaken exhaustive tests of all interfaces and that all parameters have been exercised. This should include bounds testing and negative testing and be supported by an analysis of the test coverage.

Testing: implementation representation

Suppliers should provide an analysis of the depth of testing undertaken. This should demonstrate the correspondence between the tests in the test documentation and the behaviours/interactions of the subsystems and modules. This testing should demonstrate that modules behave and interact as described in the design and security architecture documentation and is in accordance with the implementation representation.

Ordered functional testing

Test documentation should accompany any components and consist of test plans, expected test results and actual test results. This should identify the tests performed, describe the scenarios for performing each test and include any ordering dependencies on the results of other tests. The expected test results shall show the anticipated outputs from a successful execution of the tests.

Independent testing – complete

Independent testing should clearly demonstrate that components operate in accordance with their design representation and guidance documents. Where necessary, evaluation testing should include repeating all of the developer tests including any test harnesses, test programs and machine-readable test documentation where necessary.

6. Vulnerability Assessment

Advanced methodical vulnerability analysis

A methodical vulnerability analysis performed by an evaluator to ascertain the presence of potential vulnerabilities. This could include penetration testing to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the component. Ideally this should be an independent methodical vulnerability analysis using the guidance documentation, functional specification, component design, security architecture description and implementation representation.

Determining the level of assurance/artefacts required

The purpose of any component/system evaluation should be to independently verify that the claims made for a given component are valid and true in the end usage context. The level of testing/evidence required will depend on the assurance level required on the end use system. This will most likely be determined by the component integrator. This may range from:

- EAL1 – Functionally tested, some confidence in correct operation is required but threats to security are not viewed as serious. Independent assurance will be of value to show that due care has been exercised with respect to the protection of information.
- EAL7 – The system is for application in extremely high risk situations where the system has a high value of assets and justifies higher costs. There will be tightly focused security functionality that is amendable to extensive formal analysis. Requires a high level of supporting security design evidence.

By supplying a large amount of evidence, a supplier can increase the number of potential applications of their component(s).

17 Appendix C - AssuranceLevelData XML

This appendix provides a preliminary example illustrating what the scope and structure of the Assurance Level Data might look like. The final definition will be described in the planned future metamodel reference manual.

```
<assuranceLevelData componentName="FireControl" configurationVersion="234">

<!-- Except when specifically mentioned, all information here below is
     defined at component specification time -->

<!-- #IF COMPONENT IMPLEMENTED -->
  <componentInformation>
    <!-- Information not really related to safety|security but it might be useful for
assessment/deployment -->
    <technology value="CrownJewels"/>
    <provider name=""/>
    <!-- ... -->
  </componentInformation>
<!-- #ENDIF -->

  <assuranceDevelopmentLevels>
    <!-- Assurance Levels required or used for this specific version -->
    <generalAssuranceDevelopmentLevel domain="safety">
      <process standard="IEC61508" version="23" name="SIL"/>
      <level value="4"/>
    </generalAssuranceDevelopmentLevel>
    <generalAssuranceDevelopmentLevel domain="security">
      <process standard="CommonCriteria" version="3.3" name="EAL"/>
      <level value="4"/>
    </generalAssuranceDevelopmentLevel>
    <!-- ... -->
<!--  #IF COMPONENT IMPLEMENTED -->
<!--
If the supplier decides to provide a multicriticality component
the supplier may provide a reply module by module
in that case, the generalAssuranceDevelopmentLevel values are
described in the lines just below.
It is not necessary to repeat the process definition !!
-->
    <moduleImplementationAssuranceDevelopmentLevels>
      <moduleImplementation name="M1" domain="safety|security" level="4"/>
    </moduleImplementationAssuranceDevelopmentLevels>
<!-- #ENDIF -->
  </assuranceDevelopmentLevels>
```

```
  <properties>
    <property name="foobar" domain="safety">
      <value>
       free text
      </value>
<!--  #IF COMPONENT IMPLEMENTED -->
      <reliesOn>
       <insertionRequirement name="@XPath in solution TIP1"/>
       <insertionRequirement name="@XPath in solution TIP1"/>
        <evidence name="@XPath in this file evidence/..."/>
       <!-- .. -->
      </reliesOn>
```

```
<!-- #ENDIF -->
    </property>
    <!-- ... -->
  </properties>

  <technicalInsertionPolicy name="specifying-TIP1"/>
  <!-- If needed, the component supplier may update the technical insertion
       policy referenced here to point to the solution Technical Insertion
       Policy -->
```

```
<!--  #IF COMPONENT IMPLEMENTED -->
  <evidence>
    <evidence name="E1" domain="safety">free text</evidence>
<!--
For the pieces of evidence, we can rely on external files for which the
format is specific for the associated regulation requirements.
By example, for a DAL C, we need to provide plans, files, etc based on the EASA
requirements from their understanding of the DO178.
In relation with the superset of data.
-->
    <externalEvidence name="E2"  domain="safety" file="foobar.txt"/>
    <!-- ... -->
  </evidence>
<!--  #ENDIF -->

</assuranceLevelData>
```

Figure 8 – Concrete XML example of the Assurance Level Data

When a component will be certified or accredited, the Catalogue Entry may register the programmes on which the configuration above has been agreed. For each agreement, the Catalogue Entry shall refer to the associated report.

```
<!-- #IF COMPONENT CERTIFIED OR ACCREDITED -->
<!--
To register the programmes on which this configuration has been agreed
(certified/accredited) with the same insertion policy, properties and
evidence
-->
    <certification program="Nieuport XVII" date="1915-07-13" authority="Aéronautique
militaire">
      <report name="" file="foobar.txt"/>
    </certification>
    <accreditation program="Sopwith Camel" date="1915-08-23" authority="Royal Flying
Corps"/>

    <!-- ... -->
<!-- #ENDIF -->
```

Figure 9 – Snippet of the Catalogue entry for registering certification/accreditation