European Component Oriented Architecture (ECOA®) Collaboration Programme: Guidance Document: Driver Components

Date: 27/11/2017

Prepared by
BAE Systems (Operations) Limited

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information..

Contents

1	Scope	1
2	Introduction	1
3	Abbreviations	2
4	Definitions	3
5	References	4
6	Driver Component Scenarios	5
6.1	General Design Considerations	6
6.1.1	Reusability within an ECOA System	6
6.1.2	Reusability within an ECOA Component	7
6.1.3	Use of the External Interface mechanism:	9
6.1.4	Integration to a Bespoke software call-back Mechanism	11
6.2	Interacting with a HMI	12
6.2.1	ECOA System Design	14
6.2.2	Example Driver Component Operation Links	16

Figures

Figure 1 - Key	5
Figure 2 - Strongly coupled Driver Component	6
Figure 3 - Decoupled Driver Component	7
Figure 4 - Strongly coupled Modules	8
Figure 5 - Decoupled Modules	9
Figure 6 - ECOA External Interface	10
Figure 7 – Bespoke Software Call-backs within a “Driver” Module	11
Figure 8 - Interaction with a GUI	12
Figure 9 - Example Graphical User Interface	13
Figure 10 - Example Driver Component	14
Figure 11 - Example Driver Component Operation Links	17

Tables

No table of figures entries found.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

0 Executive Summary

'*Driver Component'* is the term used to describe an ECOA Component which communicates with hardware and/or software using interfaces other than those defined by ECOA. Examples of Driver Components may include a Sensor Component which communicates directly with sensor pod hardware or a File Server Component which communicates directly with an underlying file system.

This document describes a number of scenarios which would require the use of a Driver Component and highlights some of the considerations which should be taken into account whilst designing and developing the Driver Component.

It is not in any way a "normative", part of ECOA, or even definitive. The discussions here are purely examples of how ECOA Driver Components can be designed and implemented.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

1 Scope

This document is intended to provide guidance for Component Designers and Implementers regarding Driver Components. This includes suggestions on possible design patterns along with specific examples of ways that the Driver Component may interact with software external to ECOA.

The document is structured as follows:

Section 2 gives a brief introduction to the Driver Component topic.

Section 3 expands abbreviations used in this report.

Section 4 provides definitions for the key terms used in this report.

Section 5 lists key documents referenced by this report.

Section 6 discusses a number of scenarios requiring Driver Components which may be relevant to an ECOA System.

2 Introduction

This document provides a number of examples of ECOA Driver Components. It highlights some of the considerations which should be taken into account during design and development and provides rationale on why these aspects are important.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

3 Abbreviations

API	Application Programming Interface
COTS	Commercial Off-The-Shelf
DGA	Direction Générale de l'Armement
Dstl	Defence Science and Technology Laboratory
ECOA	European Component Oriented Architecture
IP	Internet Protocol
MOD	Ministry of Defence
SOA	Service-oriented Architecture
TCP	Transmission Control Protocol
XML	eXtensible Markup Language
XSD	XML Schema Definition

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

4 Definitions

For the purpose of this document, the definitions given in the ECOA Architecture Specification *(ref. [AS])* Part 2 and those given below apply.

Term	*Definition*
(currently none)	

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

5 References

AS	European Component Oriented Architecture (ECOA) Collaboration Programme: Architecture Specification (Parts 1 to 11) “ECOA” is a registered mark.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6 Driver Component Scenarios

A number of scenarios can be envisaged which would require the use of a Driver Component if the functionality was to be implemented as an ECOA Component. The following sections discuss some of these scenarios and provide guidance on possible designs and issues to be considered.

Figure 1 provides a key to the symbology used througout the this document to help illustrate the concepts being discussed.

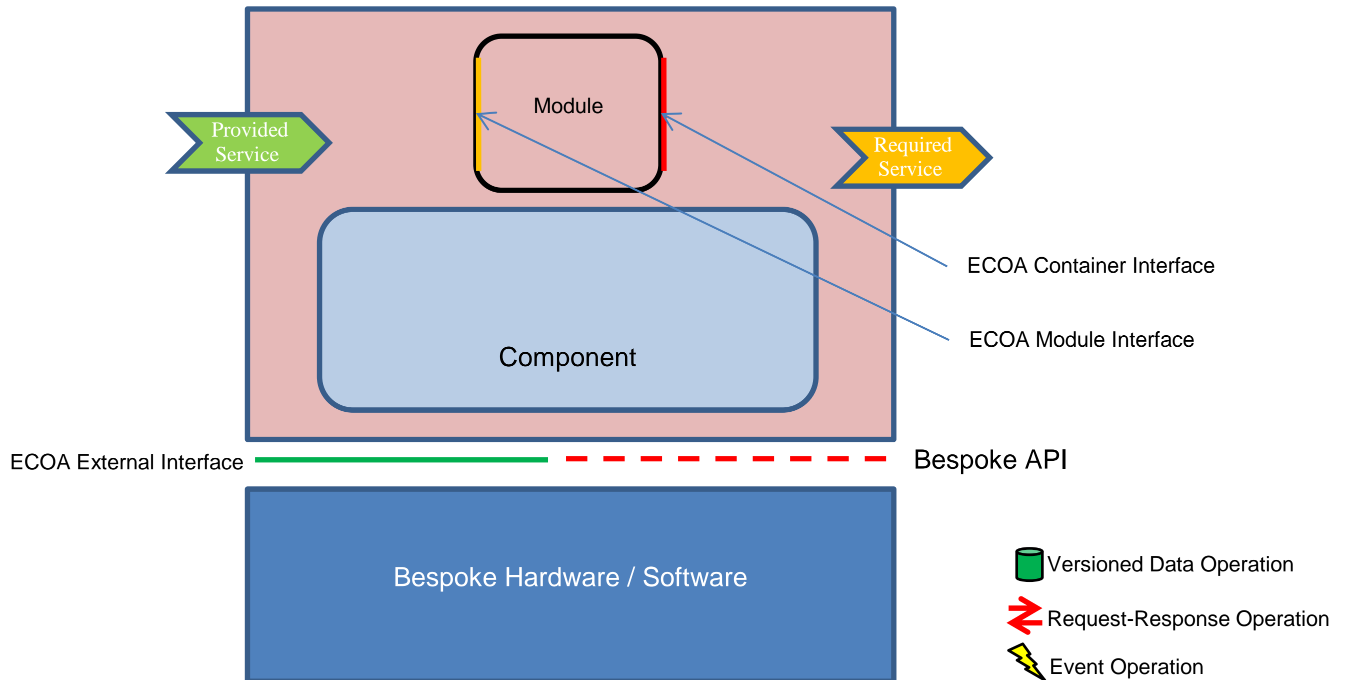

Figure 1 - Key

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.1 General Design Considerations

There are a number of general considerations which should be addressed when designing and developing ECOA Driver Components. The following sections highlight some of the key areas.

6.1.1 Reusability within an ECOA System

In order to gain the maximum benefit the use of ECOA can provide; it is important to design systems with reuse of Components in mind. In order to achieve this, it is advantageous to limit the use of Driver Components where possible. When the use of a driver component is necessary, reuse can be maximised by limiting the role of a driver component to a single task.

An example of this design methodology would be a Component which requires access to a file for configuration data. It is possible for the Component to directly access a file using a non-ECOA interface directly as shown in Figure 2.

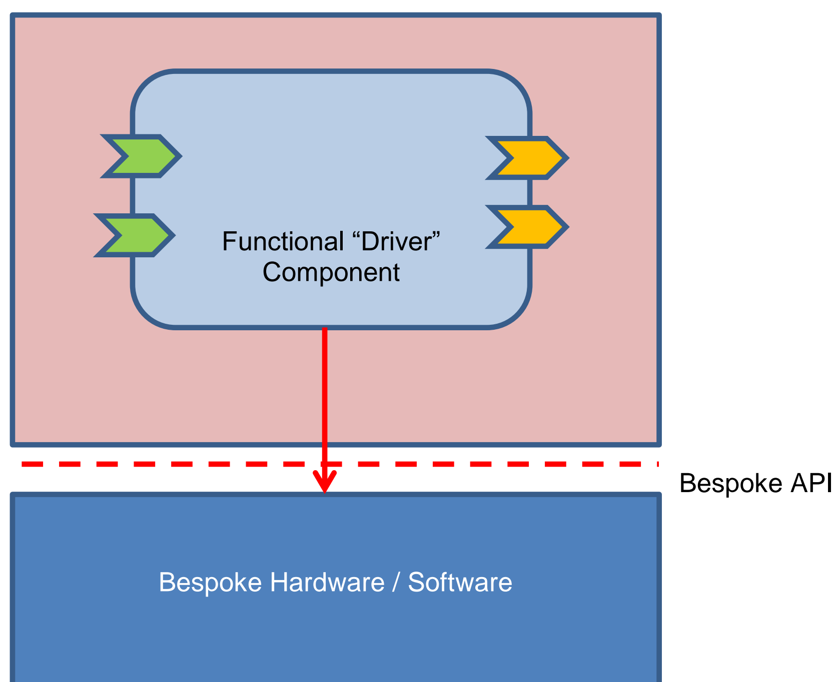

Figure 2 - Strongly coupled Driver Component

However, an option which may promote reuse is to decouple the file handling from the functional operation of the Component. This could be achieved by defining a “File Handler” Component. The “File Handler” Component could then provide a regular ECOA “File Service” to the functional Component. This would mean that the functional Component would in fact be a regular (non-Driver) Component, thus meaning its reuse properties remain unaffected. An example of this approach is shown in Figure 3.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

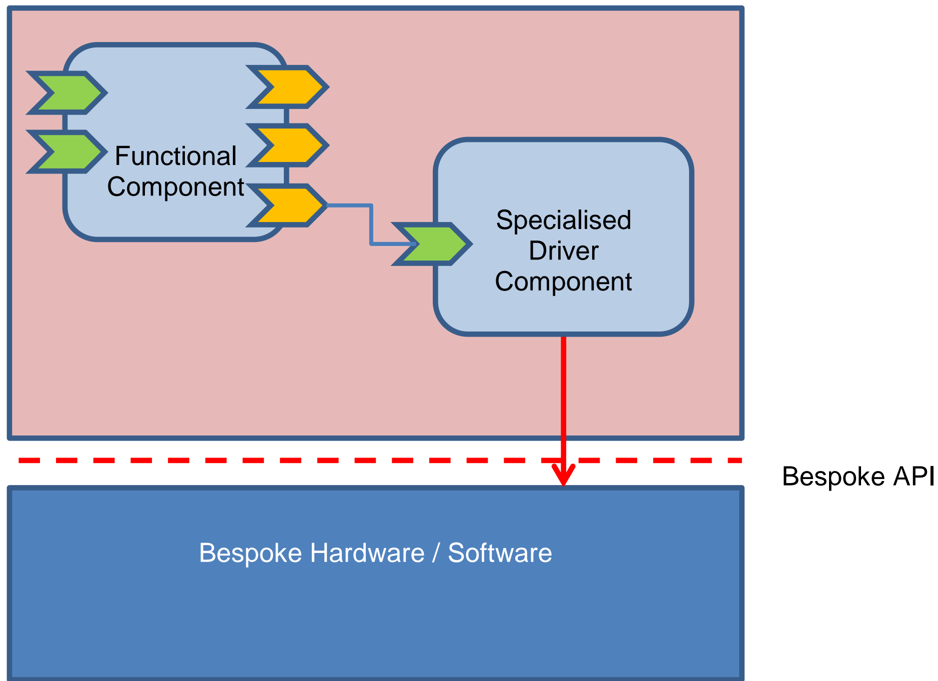

Figure 3 - Decoupled Driver Component

Note that this example could make use of the PINFO mechanism provided by ECOA; it only serves to illustrate the concept.

6.1.2 Reusability within an ECOA Component

Although the very nature of a Driver Component means that reusability of the Component is inherently restricted; it is still advantageous to maximise reuse where possible. One method of promoting reuse is from within a Driver Component itself. This can be achieved by limiting the number of modules which use a non-ECOA interface. Figure 4 shows an example of a Component whereby all its Modules interact with non-ECOA interfaces; meaning each Module has limited its reuse potential.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

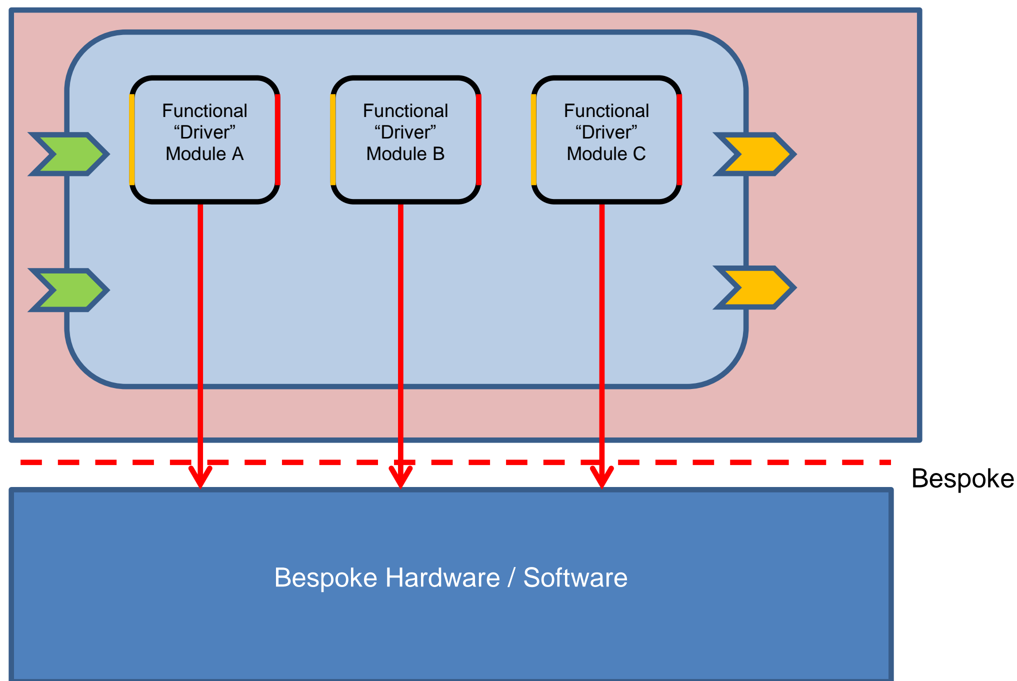

Figure 4 - Strongly coupled Modules

Figure 5 shows the same example, but this time the Component has been designed to have a single Module handling all interactions with the non-ECOA interface. This means that the majority of modules within the Component are regular ECOA Modules. The Component is therefore likely to be more reusable, as only the single (or subset) of modules which make use of the non-ECOA interface are likely to require rework in order for the Component to be reused.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

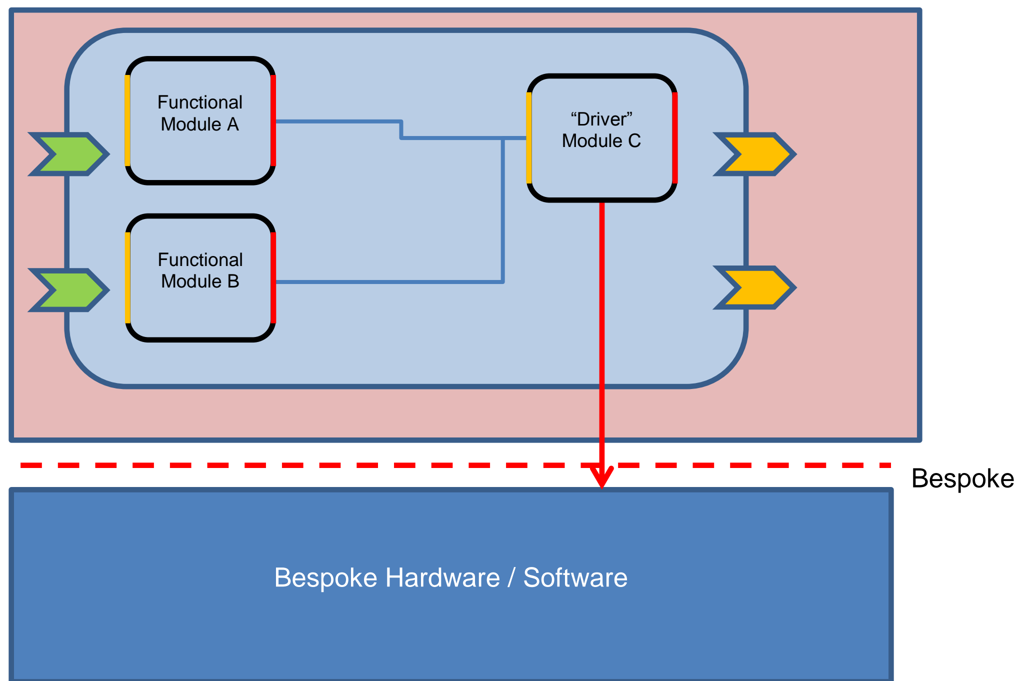

Figure 5 - Decoupled Modules

6.1.3 Use of the External Interface mechanism:

ECOA provides a mechanism for receiving input from an external source. This allows asynchronous input from non-ECOA hardware/software to be received by an ECOA module. As far as the receiving Module is concerned, the input behaves exactly the same as a normal event operation would (i.e. the message is queued and processed as per any other operation). The use of this mechanism decouples the receiving Module from the bespoke hardware/software; meaning it is possible be that the bespoke hardware/software could be changed without the Module needing updates (the only requirement is that the bespoke hardware/software complies with the same ECOA-defined external interface).

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

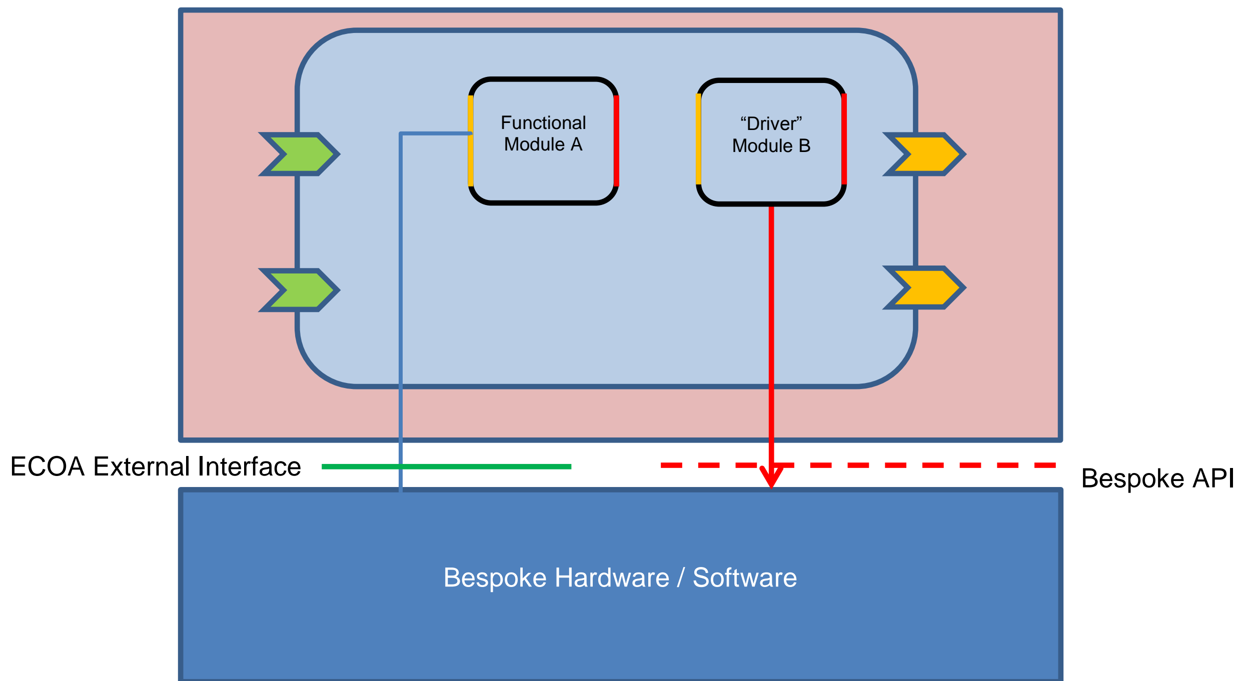

Figure 6 - ECOA External Interface

Note that the bespoke hardware/software can be shown located within the boundary of the Component or externally. This distinction is largely irrelevant; but commonly it is shown internal if the non-ECOA software is provided by the Component developer and external if the non-ECOA software is provided by a 3rd party; for example, Commercial Off-The-Shelf (COTS).

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.1.4 Integration to a Bespoke software call-back Mechanism

Some bespoke APIs may implement a call-back mechanism in accordance with the principle of inversion of control. In this case a thread from the bespoke software invokes a call-back handler that needs to exercise the container interface of a module, perhaps to send some information as an ECOA event or ECOA versioned data. This may be achieved without using the external interface described in section 6.1.3, bypassing the module queues.

In order that the ECOA module remains single threaded the bespoke software may be invoked with the module thread so that it can then be used to check for call-backs/handlers, i.e. to perform dispatching. The bespoke software dispatcher must guarantee;

- to use the module thread to make call-backs, and not use any other.
- not to block the thread preventing module operation or causing deadlock.

Under these conditions the call-back handler may use the container interface directly, rather than invoking the module interface to asynchronously queue an operation to be executed by the module thread. This approach is shown in Figure 7. As the bespoke software call-back is guaranteed to use the module thread, it may be considered a part of the “Driver” module, legitimising its use of the container interface.

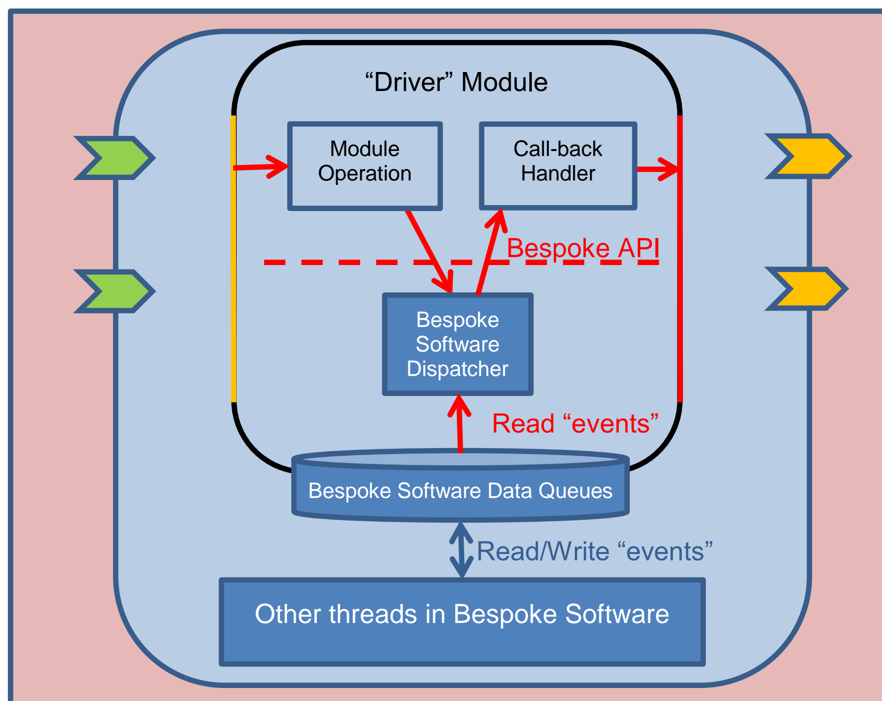

Figure 7 – Bespoke Software Call-backs within a “Driver” Module

In Figure 7 the component contains any number of threads from the bespoke software, but the “Driver” Module remains single threaded.

A potential issue with this mechanism is that in order for the Call-back handler to invoke the Container API in certain languages it would require access to the Module Context passed into the Module Operation. Two possible solutions are:

1. The Module Operation code stores the Module Context in a ‘global state’ variable such that the Call-back handler can access and use it to invoke the Container API. This approach is not recommended, as it contravenes that ECOA concept of Modules not containing their own state (other than through the Module Context). In addition this solution is not thread-safe; as if multiple Module Instances are used it may cause issues with accessing an incorrect Module Context.
2. The Module Context could be passed into the Bespoke Software Dispatcher as a parameter, which is then passed to the Call-back handler to use when invoking the Container API. This relies upon

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

the Bespoke Software Dispatcher allowing for a user defined parameter to be passed in this way, however this mechanism ensure a thread–safe multiple Module Instance solution.

These may not be the only solutions, and for certain languages the implementation and management of the Module Context may remove this issue. It is up to the Driver Component implementer along with the System Integrator to ensure a suitable solution is found.

6.2 Interacting with a HMI

It is a common requirement to provide a Human Machine Interface (HMI) to an operator in order to allow effective operation and control of a machine or system. This interface is often realised with the use of a graphical user interface (GUI).

ECOA does not provide a mechanism for interacting with a GUI; but the Driver Component concept can be employed to allow this interaction to take place within the constraints of an ECOA System. There are numerous options available in order to achieve this goal; the following section details one example.

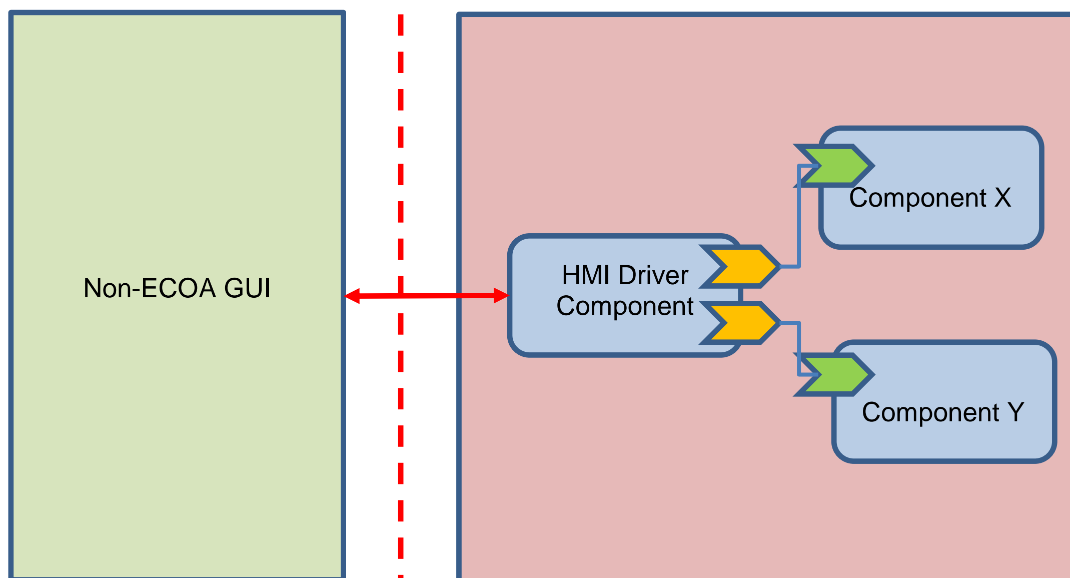

Figure 8 - Interaction with a GUI

Although the example presented is of a very simple system; its primary objective is to illustrate the concepts and guidance detailed in section 6.1 of this document, which can be applied to systems of varying complexity.

The example system consists of two main capabilities:

- The first provides a simple “addition” capability whereby a user can provide 2 values to be added, and an ECOA Component will perform the calculation and provide a result.
- The second capability allows the user to control the speed of a bouncing ball (via increase and decrease operations). An ECOA component is responsible for managing the current speed of the ball.

An example Graphical User Interface (GUI) for this application is shown in Figure 9.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an ‘as is’ basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

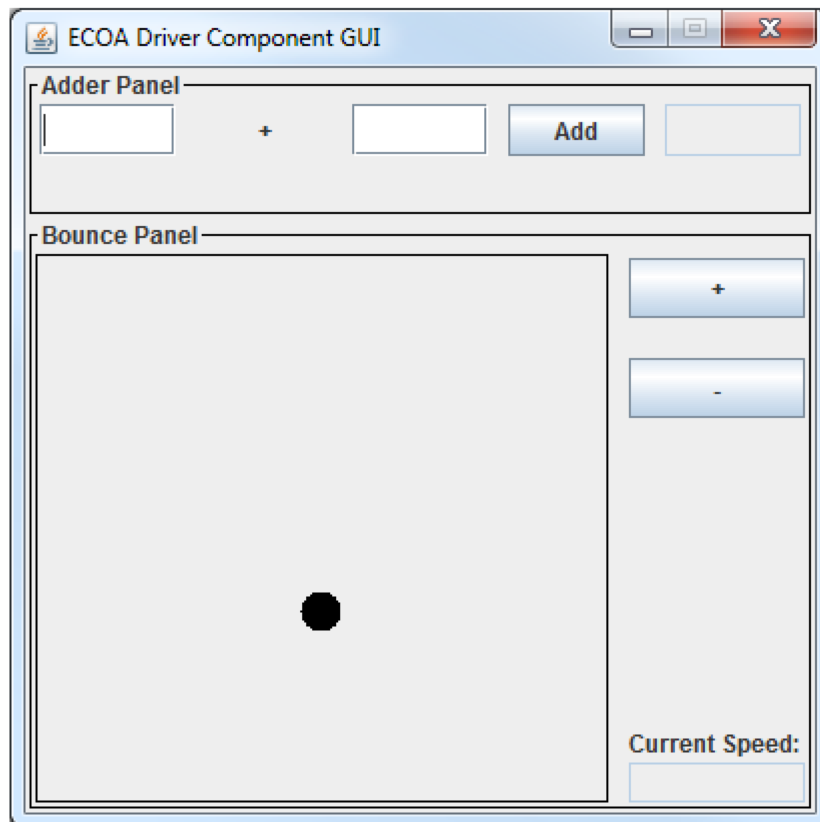

Figure 9 - Example Graphical User Interface

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.2.1 ECOA System Design

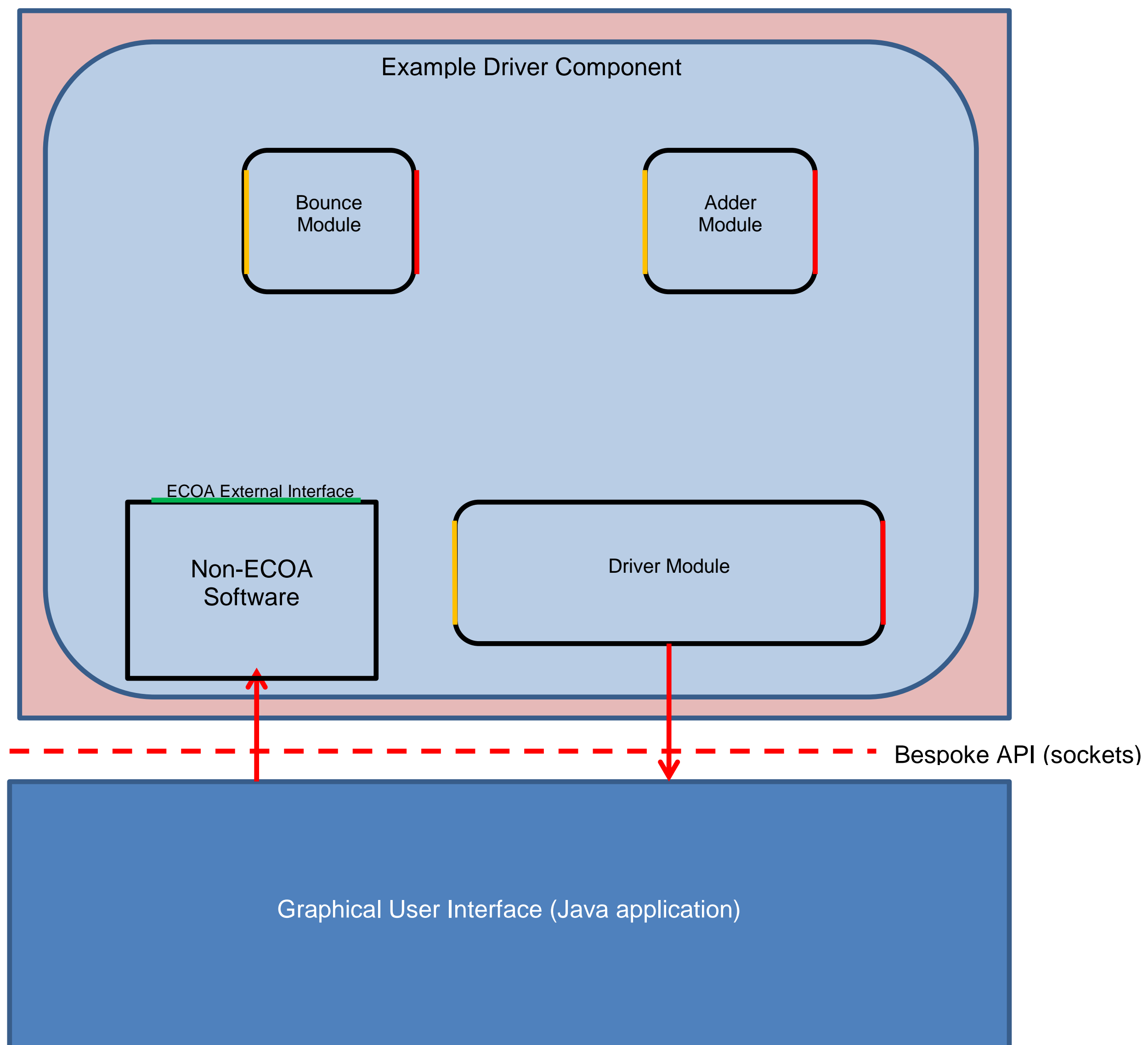

Figure 10 - Example Driver Component

Figure 10 shows a single Component which has been decomposed into 3 ECOA modules. In addition, there is a section of "non-ECOA software" which makes use of the ECOA External Interface to provide asynchronous input to the ECOA Modules. The non-ECOA software is shown within the bounds of the Component as it is to be provided by the Component developer. The Graphical User Interface (Java application) is shown external to the Component, as this could potentially be a 3rd party or COTS tool (however it could equally be shown as part of the non-ECOA software residing within the Component boundary).

The concept of execution is as follows:

- The Driver Module communicates with the Graphical User Interface application using sockets.
- The Graphical User Interface can also communicate with the ECOA component using sockets.
- The non-ECOA software located within the Component is responsible for handling the input and performing the required actions. This is done using a separate thread of control to that of the Driver Module, so as to comply with the ECOA Inversion of Control principles (i.e. the Driver Module thread of control should not block waiting for input from the GUI).

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

The following sections provide an overview of the functionality each element in Figure 10 is responsible for.

6.2.1.1 Adder Module

The Adder Module is responsible for performing the calculation. It has one request response operation which takes 2 unsigned integer values and produces a response which is the result of adding the two numbers.

Its Module Type is defined as:

```
<moduleType name="Adder_Module_Type">
     <operations>
          <requestReceived name="add">
               <input name="value1" type="uint32" />
               <input name="value2" type="uint32" />
               <output name="result" type="uint32" />
          </requestReceived>
     </operations>
</moduleType>
```

6.2.1.2 Bounce Module

The Bounce Module is responsible for managing the speed at which the ball should travel. The module can receive a request to change speed (an event operation) and it can output an event containing its current speed.

Its Module Type is defined as:

```
<moduleType name="Bounce_Module_Type">
     <operations>
          <eventReceived name="changeSpeed">
               <input name="varySpeed" type="speed:varySpeed" />
          </eventReceived>
          <eventSent name="currentSpeed">
               <input name="speed" type="uint8" />
          </eventSent>
     </operations>
</moduleType>
```

Where the varySpeed type is defined as:

```
<enum name="varySpeed" type="uint8">
     <value name="increase"/>
     <value name="decrease"/>
</enum>
```

6.2.1.3 Driver Module

The Driver Module is responsible for handling all outbound communication with the Java Graphical User Interface application. This communication is realised through the use of sockets. In order to comply with the ECOA Inversion of Control (IoC) principle, the Driver Module must not perform any blocking operations. This implies that if a blocking receive operation is to be performed; a separate thread of control would be required. The Driver Module is responsible for creating and managing this receiver thread.

Note that the decision to make the Driver Module responsible for the creation (and management) of the non-ECOA Software's thread of control is an important one; it means that the other modules of the Component do not have their reusability properties altered and only the single module is still less portable.

Its Module Type is defined as:

```
<moduleType name="Driver_Module_Type">
```

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <operations>
            <eventReceived name="currentSpeed">
                <input name="speed" type="uint8" />
            </eventReceived>
            <eventReceived name="handleAdd">
                <input name="value1" type="uint32" />
                <input name="value2" type="uint32" />
            </eventReceived>
            <requestSent name="add" isSynchronous="true" timeout="0">
                <input name="value1" type="uint32" />
                <input name="value2" type="uint32" />
                <output name="result" type="uint32" />
            </requestSent>
        </operations>
    </moduleType>
```

6.2.1.4 Non-ECOA Software

The non-ECOA software is responsible for handling all inbound communication from the Java Graphical User Interface application. The non-ECOA Software will execute in its own thread of control managed (i.e. created) by the Driver Module.

The non-ECOA Software's thread of control will enter an infinite loop and block waiting for input from the user (via the GUI). On receipt of a message from the GUI application, the non-ECOA Software will use the ECOA External Interface to inform the ECOA Modules of a requested action.

The ECOA External Interface has the following operations:

```
<external operationName="changeSpeed" language="C"/>
<external operationName="add" language="C"/>
```

6.2.1.5 Graphical User Interface (Java Application)

The Graphical User Interface is a standalone Java application which interacts with the ECOA system using sockets. It allows the user to perform two main activities:

- Provides the capability to add 2 numbers together
- To manage the speed of a bouncing ball (using increase/decrease operations)

6.2.2 Example Driver Component Operation Links

Figure 11 shows the functional operation link connectivity within Example Driver Component.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

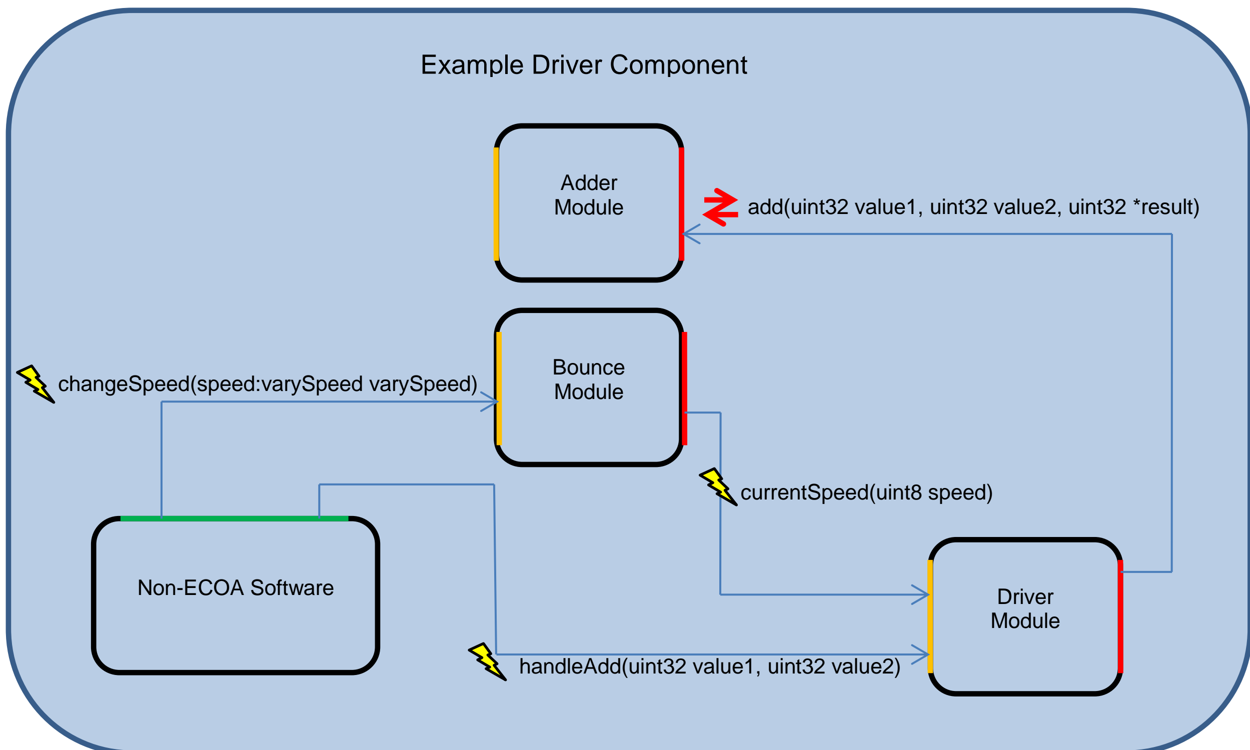

Figure 11 - Example Driver Component Operation Links

The non-ECOA Software has two operations available on the ECOA External Interface. The first is the "changeSpeed" operation which is connected to the Bounce Module's operation of the same name. The second is the "add" operation which is connected to the Driver Module's "handleAdd" operation.

The "add" operation of the External Interface could equally have been connected to the Adder Module directly (as per the "changeSpeed" operation). The example serves to highlight that multiple options are available. There is no perceived advantage to either method, as the External Interface operation will act as per any other normal event operation to the receiver; therefore the reusability properties of the receiving module will remain unaffected.

This Component has been designed with a single "driver" module in mind. It would be possible to send the current speed directly from the Bounce Module to the GUI application via sockets. However, this would then expose the Bounce Module to a non-ECOA API which would affect its reusability properties. In other use cases it may be necessary, due to latency requirements for example, to send the message directly to the GUI application. In such a case, reusability properties are sacrificed in order to gain performance.

This document is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. The information set out in this document is provided solely on an 'as is' basis and developers of this document make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.