



Zeligsoft CE™ in Action

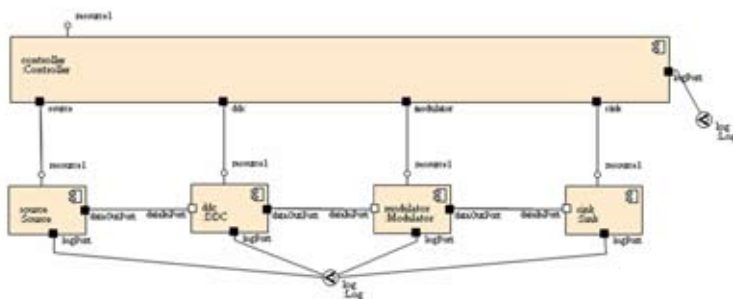
Running a waveform on multiple SCA compliant Core Frameworks

Zeligsoft is the expert in software engineering processes and tools for component-based software development. Zeligsoft Component Enabler (CE) is a model-driven development tool proven to accelerate development and improve software quality of SCA-based applications.

By using the SCA standard to develop Software Defined Radios, components and applications (waveforms) will be reusable and portable across platforms. Here, CE is used to model components, waveforms and deployments, write waveform implementations and execute components on multiple SCA Core Frameworks running in a development environment. The components and applications developed with CE are easily portable.

The Application (Waveform)

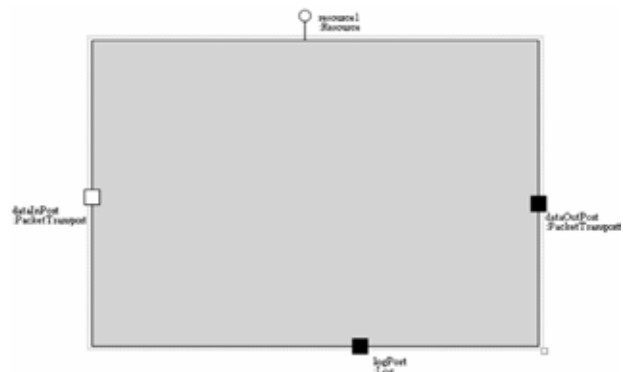
This sample waveform consists of a source, a digital down converter, a modulator and a sink. The waveform also contains a controller component that performs the role of the assembly controller.



Sample waveform

(Note that this application is a sample, used to illustrate the design and implementation of components. In live systems, the source and sink would be devices that create and consume data streams.)

In designing interfaces for individual components, “uses” and “provides” ports are defined. Ports for this component are based on CORBA interfaces defined in IDL.



Digital Down Converter (DDC) component

The source component pushes data to the digital down converter. Both the dataInPort and the dataOutPort use the PacketTransport IDL interface. This interface pushes sequences of data through the sample waveform.

Components have properties that control details and fine-tune their behavior. The SCA mandates that each component have a property PRODUCER_LOG_LEVEL used to control the verbosity of the message logging of the component. Other properties are user-definable. The DDC has a property representing the base frequency required.

baseFrequencySimple Properties:	
Attribute Info	
AdvancedSettings	
Action	external
Description	<no value>
Enumerations	Add/Remove enumerations here
Id	baseFrequency
Kind	
Mode	readWrite
Range	
Units	Mhz
XMLComment	<no value>
AttributeName	baseFrequency
Type	Long
Value	3

baseFrequency attribute for DDC

Zeligsoft Component Enabler allows properties such as this to be defined and represented in the visual model. This ensures that properties propagate to all artifacts that can be generated from the model: code, descriptor artifacts (XML) and documentation.

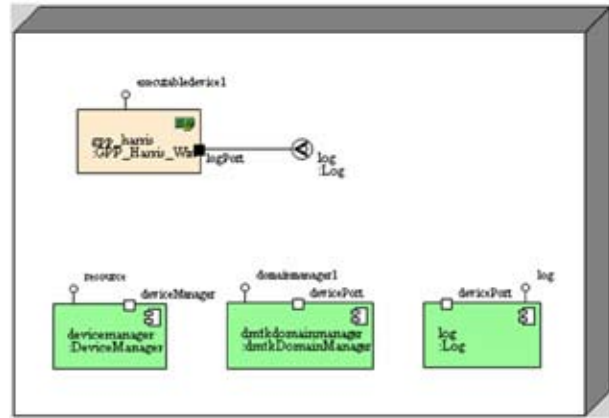
The Platforms

The sample waveform can run on multiple platforms. A platform is everything that is needed to execute the application and constitutes physical hardware, RTOS, CORBA ORB, Core Framework and devices that abstract the hardware elements. A minimal platform has an executable device (also known as General Purpose Processor (GPP)), a device manager, and a domain manager. Although not mandatory a log service is usually present also.

The sample waveform will run on:

- A platform based on the open source OSSIE Core Framework (Version C) running on a windows host, using a GPP device
- A platform based on the commercial Harris Core Framework (v.2.5) running on a windows host, using a GPP device
- The Spectrum SDR-3000 platform (Harris Core Framework v.2.5), using numerous GPPs on different Nodes – PRO3500 board, Host node

All platforms are modeled using Zeligsoft CE.



Node running the Harris Core Framework v.2.5

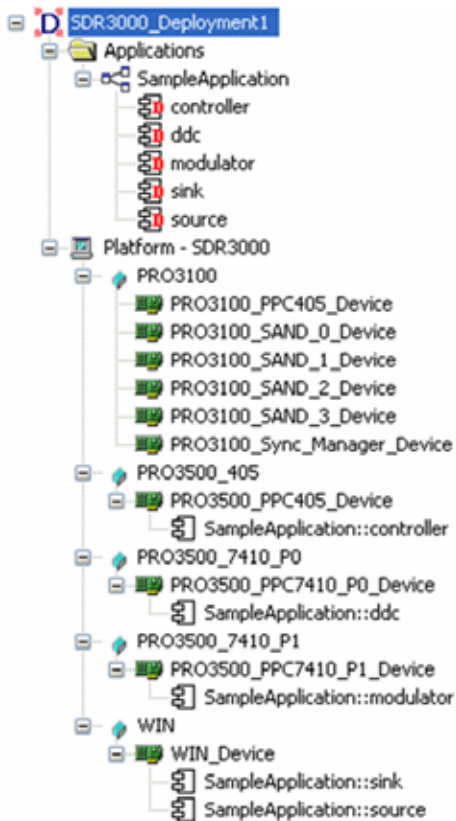
The Deployments

A component implementation according to the SCA is an executable image, kernel module or shared library that is written for a specific platform and device. A component implementation has SCA dependencies on a processor and an operating system.

A best practice in SCA development is to model the deployment of the application on the platform before generating a component implementation. This allows the designer to first understand the resources that can be used, and decide which components will run on which parts of the platform. Three deployments are modeled.

All components in each of the two windows-based platforms will run on the GPP executable device.

The SDR-3000 platform comprises different devices and so components do not have to all run on the same device. The sink and source components will run on the WIN Node (the host workstation). The controller will run on the PPC405 (the controller for the PRO3500). The DDC and modulator will run on 7410_P0 and 7410_P1 (on the PRO3500 Node).



Deployment model of the sample waveform on the SDR-3000 platform

Creating Component Implementations

The DDC (digital down converter) component has three implementations, one for each of the devices it will execute on (as per the three deployment models).



DDC component implementations

Implementations depend on the capabilities and power of the platform's operating system, the processor, and other properties. For example the NODE_TYPE string property must have the value PRO3500_7410 for the SDR 3000 deployment.



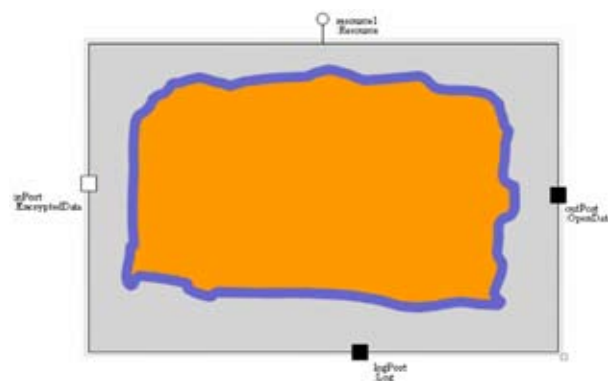
DDC component properties

Once the implementations have been created for all the components in an application, the deployment of the application on the platform can be validated against the SCA. Validating a deployment before actually coding the components that make up the application significantly reduces the time needed to debug a deployment on a real target.

All three deployments of the sample waveform on the three platforms are validated and conform to the SCA.

Implementing Components — Coding

Component code contains two aspects: the aspect that handles the SCA requirements (e.g. SCA “wrapper” code) and the aspect that handles the functional behavior of the component (e.g. control or signal processing). The code that handles the information exchange between these two aspects is dubbed “glue code”.



An SCA component

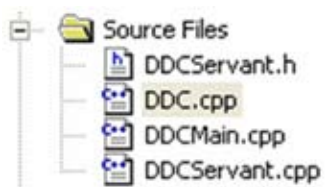
The functional code (control code or signal processing code) for a component will remain the same across platforms and Core Frameworks.

The SCA wrapper code is shaped by the SCA Core Framework the CORBA ORB and sometimes the operating system used by the component. For this reason, the SCA wrapper code for a component differs from one platform to another.

Zeligsoft's tooling environment features automated generation of SCA wrapper code. SCA wrapper code is generated for the sample application running on all three platforms (i.e. for all three deployments). The SCA wrapper code is generated with CE in a customizable fashion, to take into account the intricacies of the OSSIE and Harris Core Frameworks. CE's code is also customizable to the developers' preference and coding standard.

Code generation frees the designer's time. Developers can focus on building radios instead of deciphering the SCA. Zeligsoft CE takes care of the SCA artifacts.

The functional code implementation is done using MS Visual Studio (other popular modeling tools or IDEs such as I-Logix Rhapsody, IBM Rational Rose, Eclipse, Tornado, etc... are just as easily used). (Note that the functional code for the components making up the sample application is "empty". The DDC component will not be able to actually perform the function of a digital down converter, but it will exhibit the proper SCA behavior such as being started and connected within the Core Framework).



DDC component in Visual Studio

SCA code generated in CE is compiled in the appropriate development environment for the platform. For deployments using Windows platforms, Visual Studio is used. For the SDR-3000 deployment, the WindRiver Tornado IDE is used. The SCA wrapper code for the DDC component using the Harris Core Framework is compiled for both an x86 processor running Windows and VxWorks running on a PPC.

Compiled code is linked into executable images, combined with XML descriptor files, and transferred to the platform.

Testing the Deployments

With all the components coded, the sample waveform (application) can be deployed onto a test bed. The component executables are taken together with the XML descriptor files (also generated with CE).

The Core Framework specific tools (e.g. Visual Monitor for Harris, the testInterface for OSSIE) are used to install the waveform onto the SCA compliant platform and instantiate it. The execution results in a connected assembly of components illustrating the infrastructure of the SCA compliant application.

The sample waveform is successfully implemented using Zeligsoft CE in three configurations:

- OSSIE Core Framework (VersionC) running on a Windows host
- Harris Core Framework (v.2.5) running on a Windows host
- Harris Core Framework (v.2.5) running on the Spectrum SDR-3000

The design and implementation of an SCA compliant system can be done relatively easily with the use of proper tools.

Zeligsoft CE is an adaptable tool. It is easily used in variety of development environments, and with a variety of platforms using different SCA Core Frameworks. Components developed with CE are easily portable across multiple platforms.