

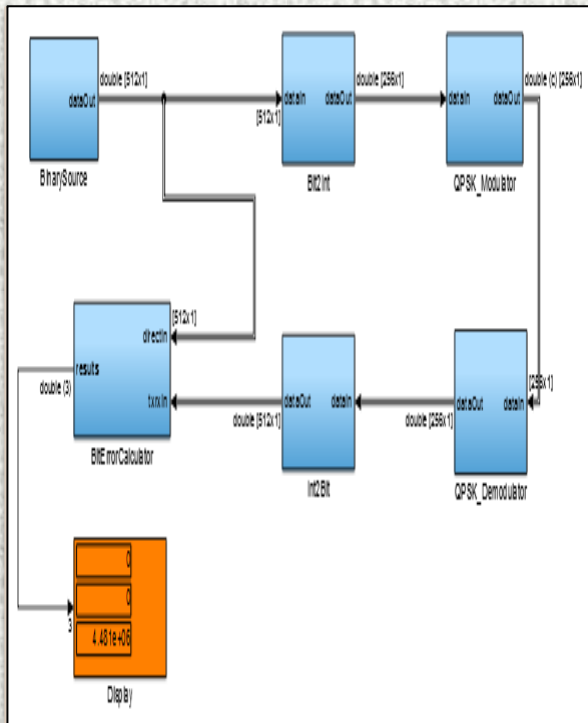


Performance Critical Middleware

A Model Driven Tool Integration Process for Rapid SDR Development

May 12, 2013

Provide an **integrated** and tool supported waveform development and deployment workflow that **reduces** time to market

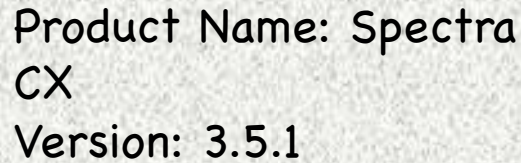


**The MathWorks™
Simulink**

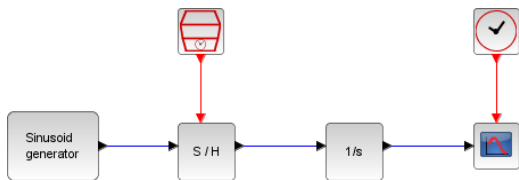
Product Name: Simulink
Version: 2012b

Key Capabilities:

- Model-based design environment
- Simulation and analysis of dynamic systems
- Comprehensive library of blocks
- Code generation



- ## Key Capabilities:
- Model-driven engineering environment
 - Code and test generation for the SCA 2.2.2
 - Remote management of platform and applications

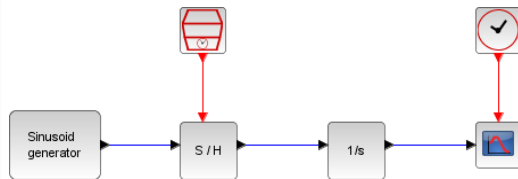


Functional WF Block Model

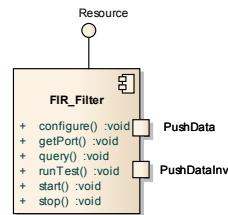
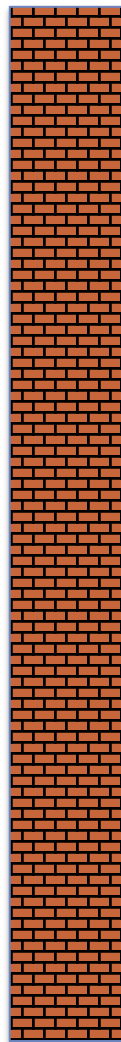
Design Waveform

Simulate & Test
Waveform

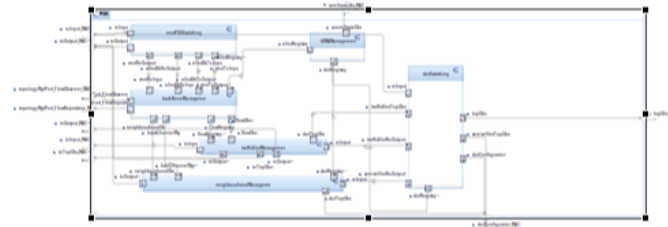
Today's Situation



Functional WF Block Model



Deployable Components

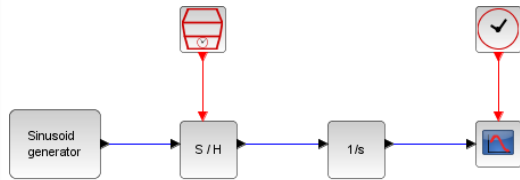


Waveform/Application

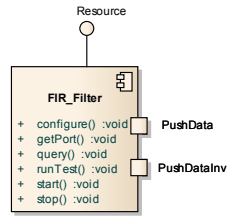
Design Waveform Components

Implement & Test Waveform Components

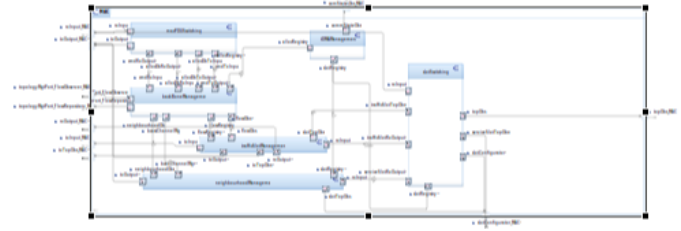
Today's Situation



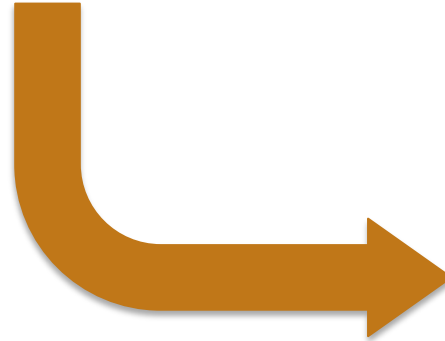
Functional WF Block Model



Deployable Components



Waveform/Application



Deploy Waveform

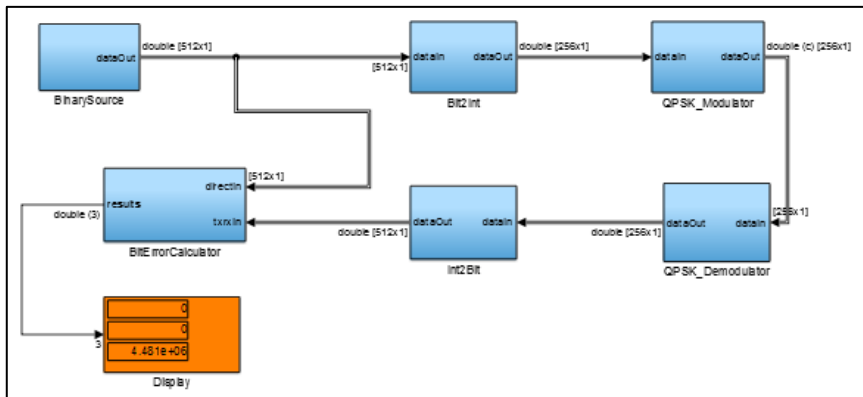


- ▶ Possibility for **errors to be introduced** during hand coding of algorithmic parts
- ▶ **Duplication of effort** as algorithm has to be implemented again
- ▶ Waveform design not always **in synch** with deployable waveform

A New Workflow...

Design and
Simulate
Waveform
Algorithm

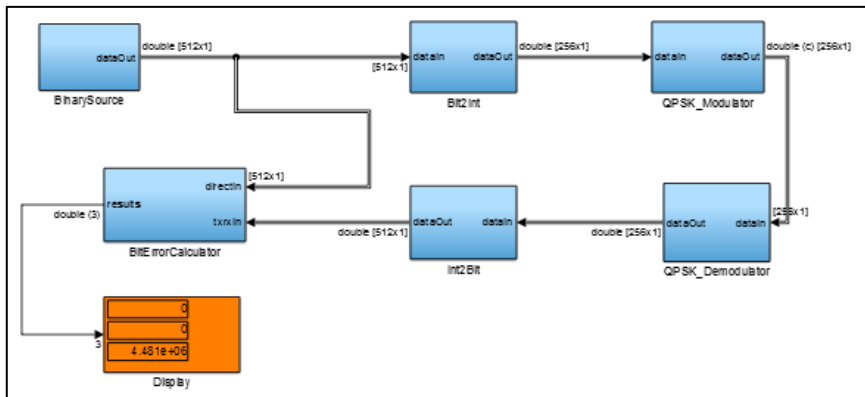
Design and Simulate WF...



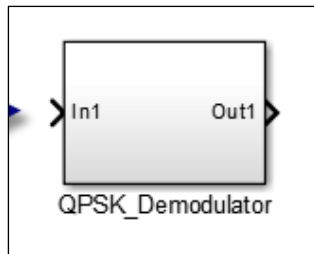
MathWorks Simulink

Design Waveform

Simulate and Test



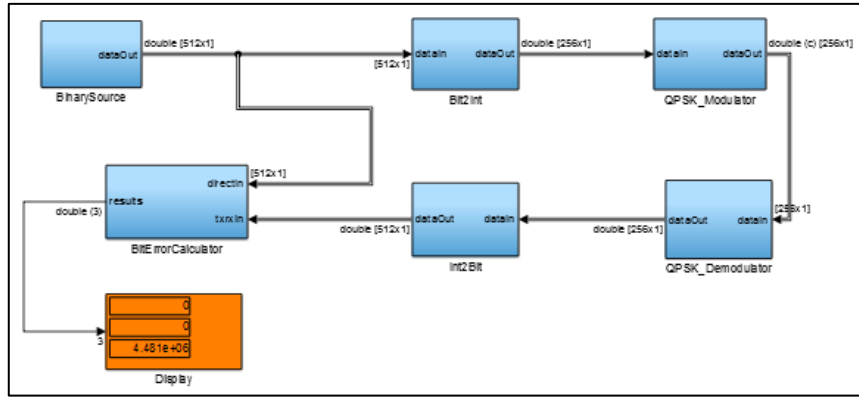
MathWorks Simulink



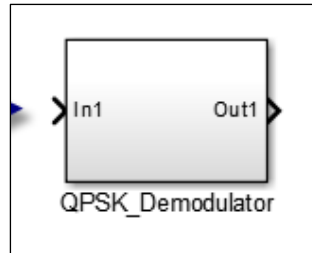
Block

Ensure all blocks
are subsystems

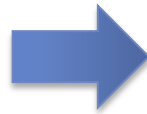
Add platform
specifics for RTW



MathWorks Simulink



Block

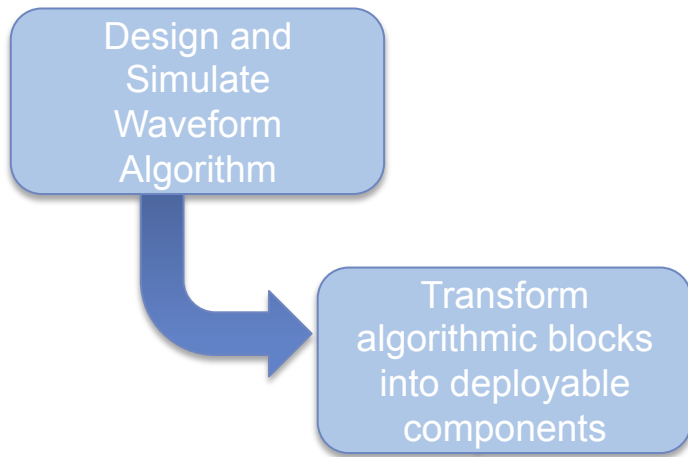


Generate C/C++ for
subsystems

- QPSK_Demodulator_private.h
- QPSK_Demodulator_ref.rsp
- QPSK_Demodulator_types.h
- QPSK_Demodulator.cpp
- QPSK_Demodulator.h
- rtmodel.h
- rtw_proj.tmw
- rtwtypes.h

Generated Source Code

A New Workflow...



Transform Blocks...

The screenshot shows the 'Simulink MDL Import' wizard window. It has a title bar with a question mark icon and standard window controls. The main content area is titled 'MDL' and contains the instruction 'Import Simulink MDL resource as model element.' with a folder icon. Below this, there is a 'Simulink MDL File' field with the path '/SimulinkSCA/qpsk.mdl' and a 'Browse...' button. A 'Simulink Subsystems' section contains a tree view for 'qpsk' with several sub-items: 'BinarySource', 'Bit2Int', 'BitErrorCalculator', 'Display', 'Int2Bit', 'QPSK_Demodulator', and 'QPSK_Modulator'. Each item has a checkbox, all of which are checked. Below the subsystems, there is an 'Into Target Model' field with the path '/SimulinkSCA/SCAModel.emx' and a 'Browse...' button. An 'Implementation' section contains fields for 'OS' (set to 'SCAModel::Linux'), 'Processor' (set to 'SCAModel::x86'), and 'Build config' (set to 'eorb16-linux-gcc-x86::eorb16-linux-gcc-x86-build'), each with 'Select' and 'Clear' buttons. A 'Diagrams' section has two checked checkboxes: 'Generate Application Diagrams' and 'Generate Component Diagrams'. A 'Controller' section has a checked checkbox 'Create SCA Application Controller' and a 'Controller Name' field containing the text 'Controller'. At the bottom, there is a help icon, a '< Back' button, a 'Next >' button, a 'Finish' button, and a 'Cancel' button.

Simulink MDL Import

MDL
Import Simulink MDL resource as model element.

Simulink MDL File: /SimulinkSCA/qpsk.mdl Browse...

Simulink Subsystems

- ☒ qpsk
 - ☒ BinarySource
 - ☒ Bit2Int
 - ☒ BitErrorCalculator
 - ☐ Display
 - ☒ Int2Bit
 - ☒ QPSK_Demodulator
 - ☒ QPSK_Modulator

Into Target Model: /SimulinkSCA/SCAModel.emx Browse...

Implementation

OS: SCAModel::Linux Select Clear

Processor: SCAModel::x86 Select Clear

Build config: eorb16-linux-gcc-x86::eorb16-linux-gcc-x86-build

Diagrams

- ☒ Generate Application Diagrams
- ☒ Generate Component Diagrams

Controller

- ☒ Create SCA Application Controller

Controller Name: Controller

? < Back Next > Finish Cancel

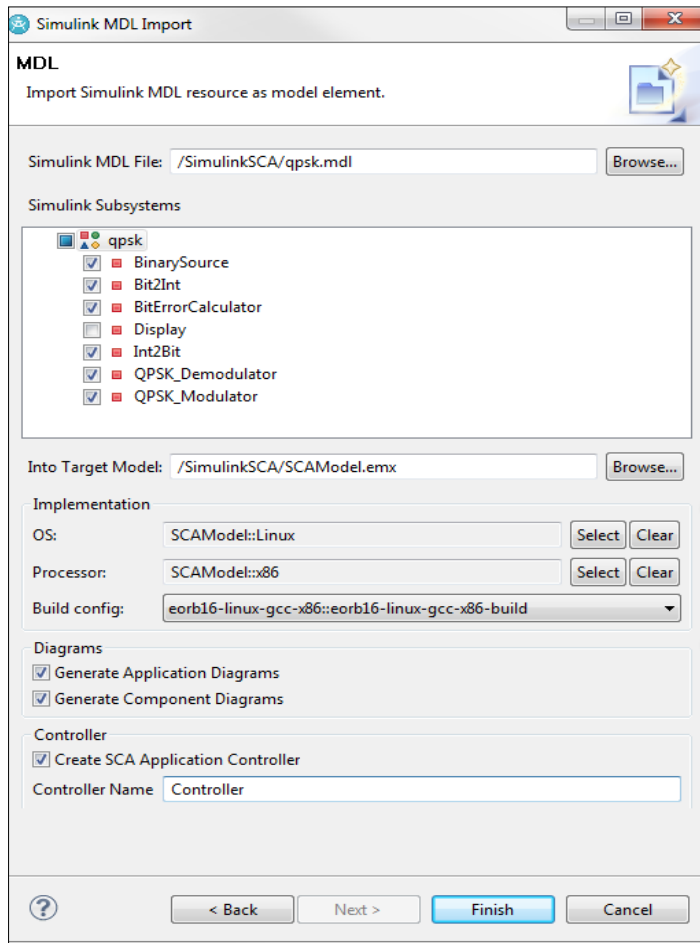
Wizard guided transformation
of design

Creates a component based
design model in CX

Select which blocks to
transform

Completely automated

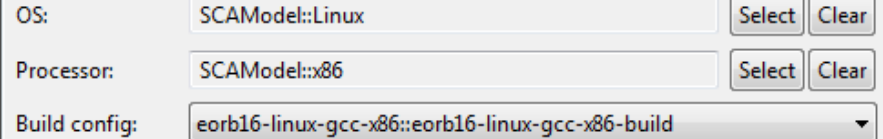
Transform Blocks...



The Simulink MDL Import dialog box is shown with the following fields and options:

- MDL**
Import Simulink MDL resource as model element.
- Simulink MDL File:** /SimulinkSCA/qpsk.mdl (with a Browse... button)
- Simulink Subsystems:** A tree view showing a 'qpsk' subsystem with the following components checked: BinarySource, Bit2Int, BitErrorCalculator, Display, Int2Bit, QPSK_Demodulator, and QPSK_Modulator.
- Into Target Model:** /SimulinkSCA/SCAModel.emx (with a Browse... button)
- Implementation:**
 - OS:** SCAModel::Linux (with Select and Clear buttons)
 - Processor:** SCAModel::x86 (with Select and Clear buttons)
 - Build config:** eorb16-linux-gcc-x86::eorb16-linux-gcc-x86-build (dropdown menu)
- Diagrams:**
 - ☒ Generate Application Diagrams
 - ☒ Generate Component Diagrams
- Controller:**
 - ☒ Create SCA Application Controller
 - Controller Name:** Controller (text field)
- Navigation:** ? (help icon), < Back, Next >, Finish, Cancel

Implementation



The Implementation configuration details are shown in a separate window:

- OS:** SCAModel::Linux (with Select and Clear buttons)
- Processor:** SCAModel::x86 (with Select and Clear buttons)
- Build config:** eorb16-linux-gcc-x86::eorb16-linux-gcc-x86-build (dropdown menu)

Select target platform for components

Enables CX to generate platform specific code

Transform Blocks...

Simulink MDL Import

MDL
Import Simulink MDL resource as model element.

Simulink MDL File:

Simulink Subsystems

- ☒ qpsk
 - ☒ BinarySource
 - ☒ Bit2Int
 - ☒ BitErrorCalculator
 - ☐ Display
 - ☒ Int2Bit
 - ☒ QPSK_Demodulator
 - ☒ QPSK_Modulator

Into Target Model:

Implementation

OS:

Processor:

Build config:

Diagrams

- ☒ Generate Application Diagrams
- ☒ Generate Component Diagrams

Controller

- ☒ Create SCA Application Controller

Controller Name

Diagrams

- ☒ Generate Application Diagrams
- ☒ Generate Component Diagrams

Options to create diagrams
as part of transformation

Transform Blocks...

Simulink MDL Import

MDL
Import Simulink MDL resource as model element.

Simulink MDL File:

Simulink Subsystems

- ☒ qpsk
 - ☒ BinarySource
 - ☒ Bit2Int
 - ☒ BitErrorCalculator
 - ☐ Display
 - ☒ Int2Bit
 - ☒ QPSK_Demodulator
 - ☒ QPSK_Modulator

Into Target Model:

Implementation

OS:

Processor:

Build config:

Diagrams

- ☒ Generate Application Diagrams
- ☒ Generate Component Diagrams

Controller

- ☒ Create SCA Application Controller

Controller Name

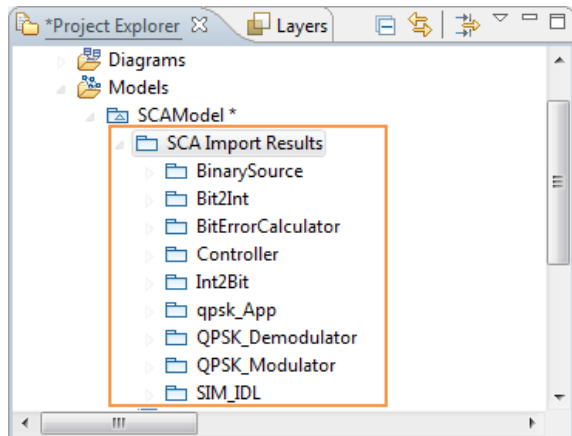
Option to create an application controller

Controller

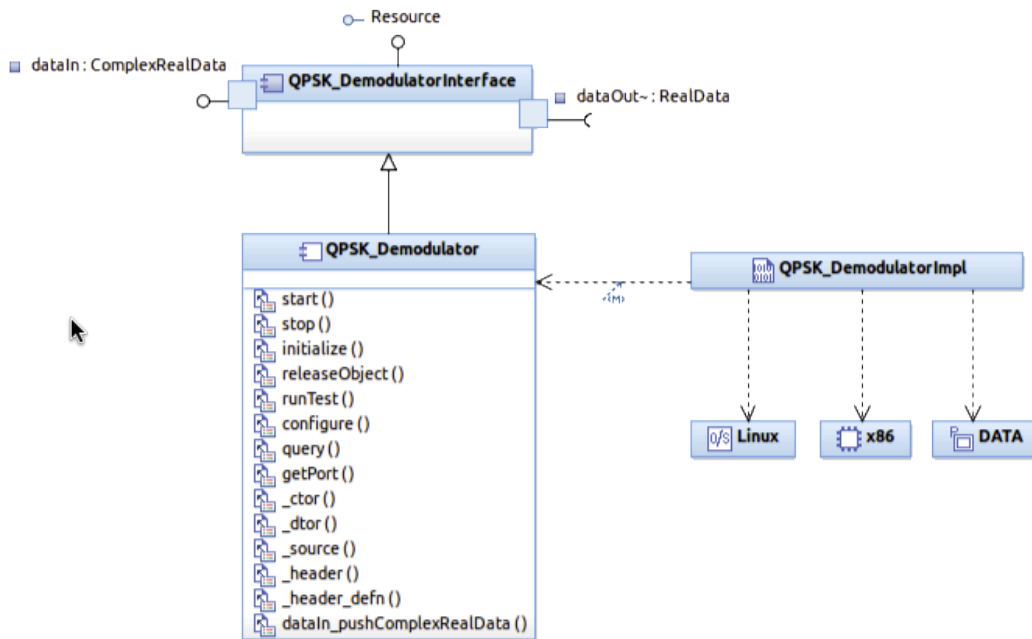
☒ Create SCA Application Controller

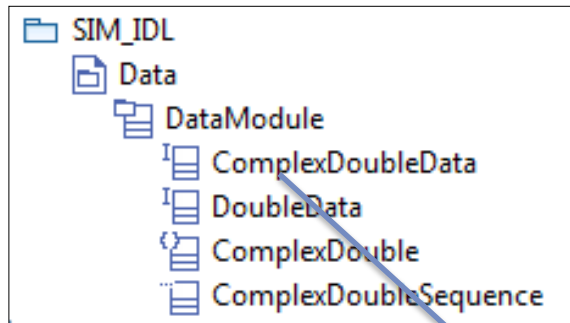
Controller Name

Components



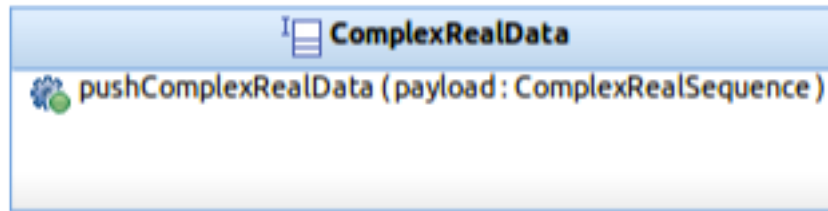
Resulting components can be compiled and deployed



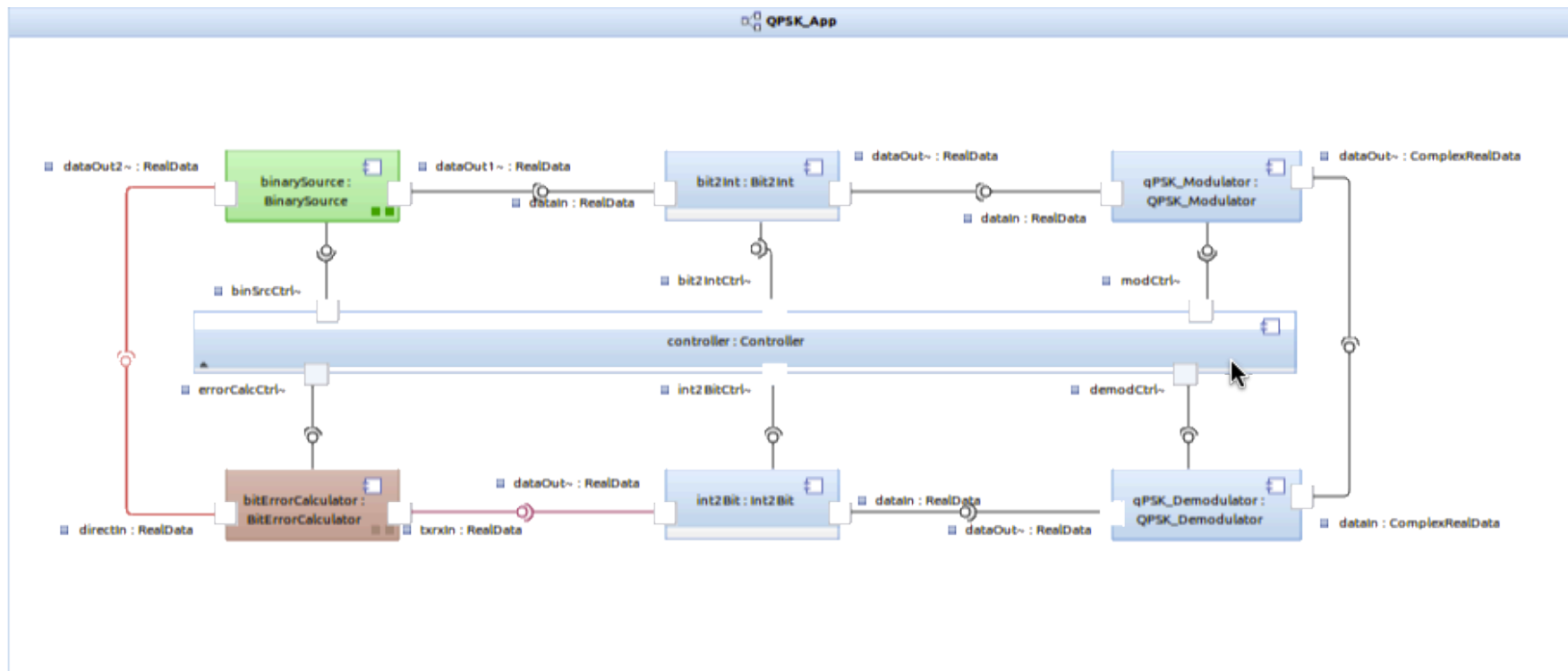


Automated creation of all
required interfaces

Interfaces define contracts
between components for
communication



Resulting application is ready to be deployed



Mapping

Simulink BlockType

Spectra CX

Subsystems

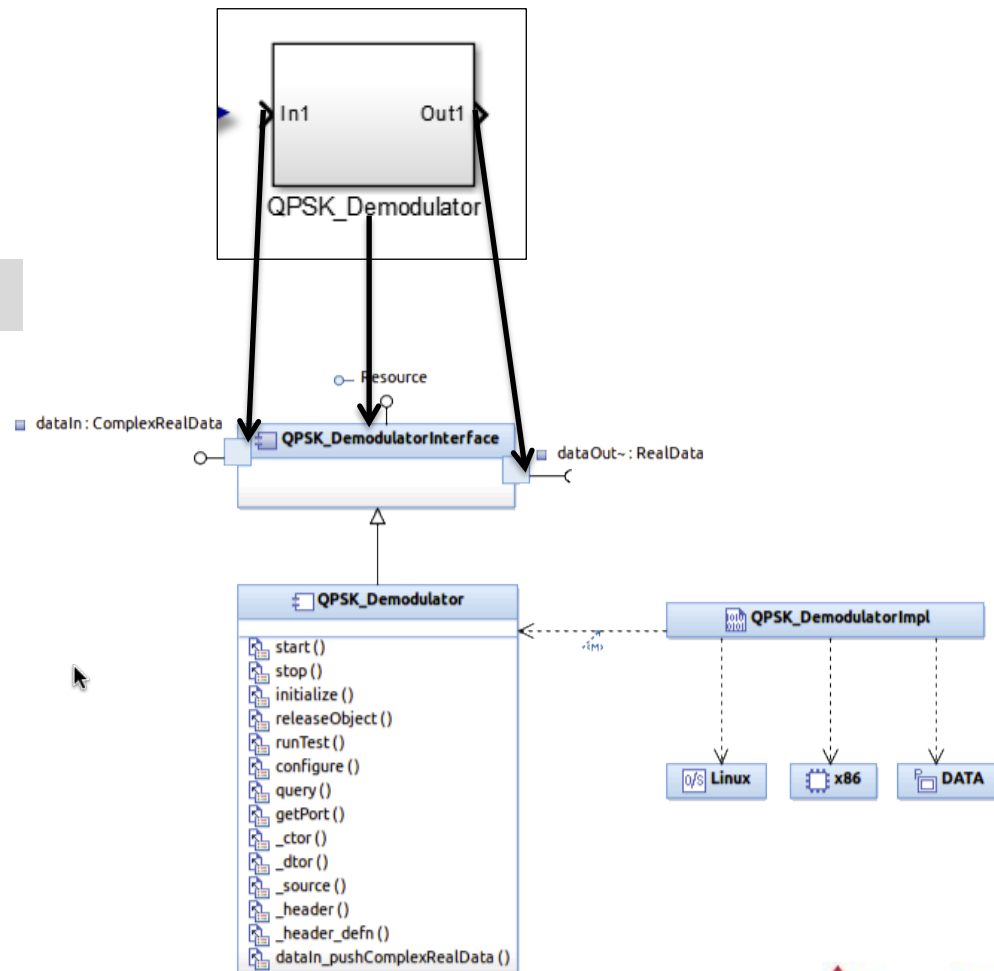
Components

Inport

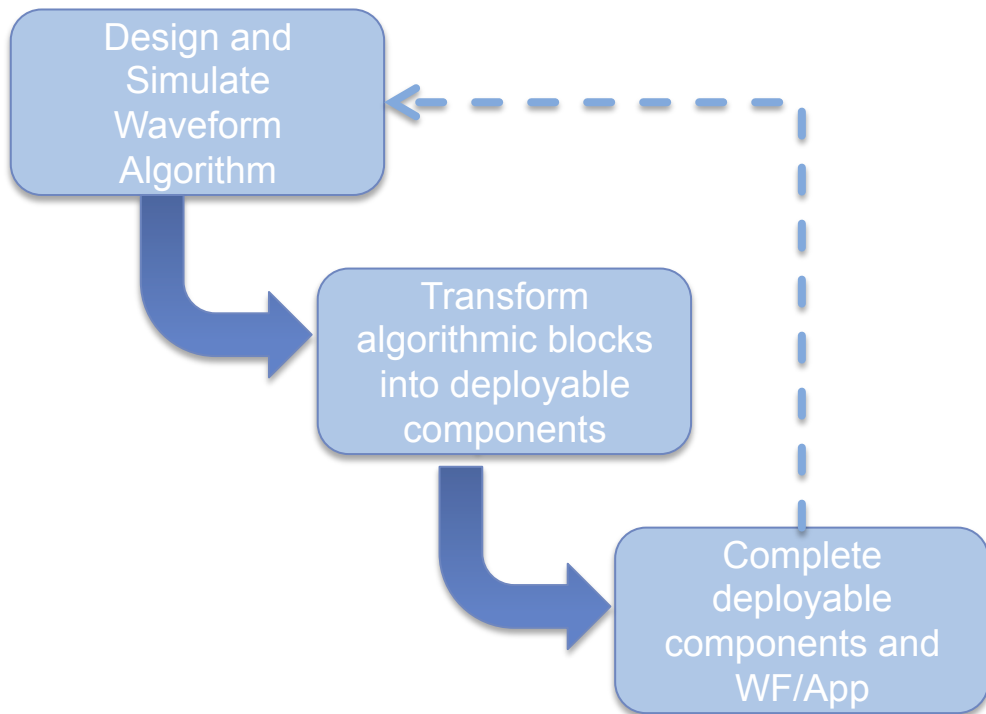
Provides Ports

Outport

Uses Ports



A New Workflow...

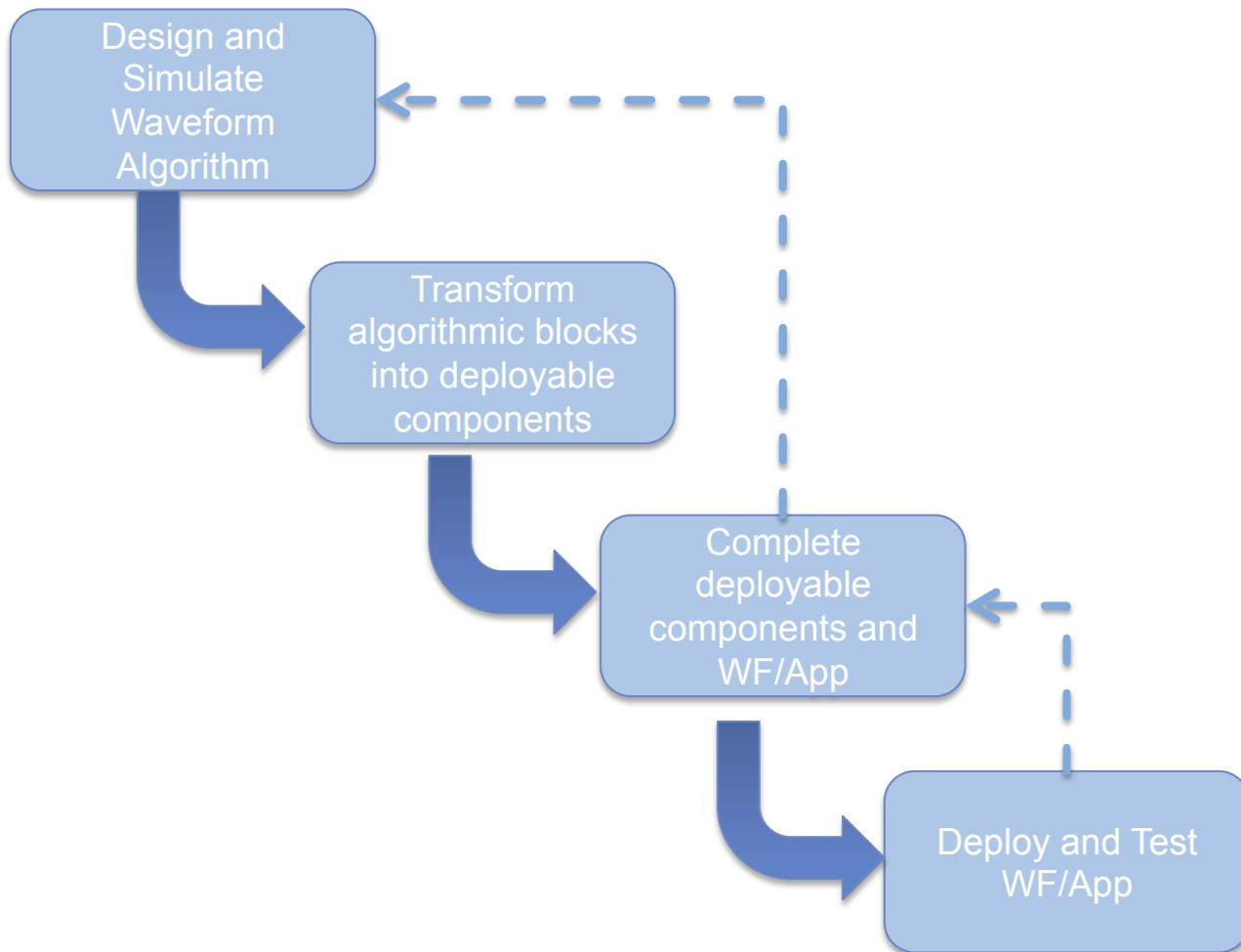


Add glue code to CX component

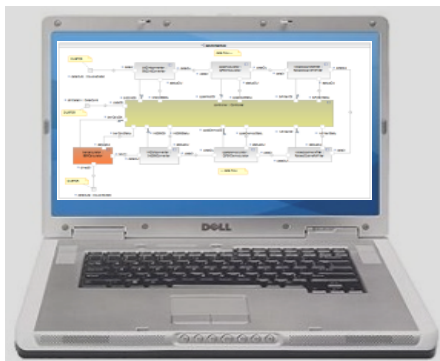
Address timing related issues

```
// fill the input buffer  
for(unsigned int x = 0; x < IN_BUFF_SIZE; x++) {  
    QPSK_Demodulator_U.dataIn[x].re = payload[x].re;  
    QPSK_Demodulator_U.dataIn[x].im = payload[x].im;  
}  
  
// step the rtw model  
QPSK_Demodulator_step();  
  
// fill the result in the complex output sequence  
for(unsigned int x = 0; x < OUT_BUFF_SIZE; x++) {  
    dataOutSeq[x] = QPSK_Demodulator_Y.dataOut[x];  
}  
  
// send the processed data  
dataOut_->pushRealData(dataOutSeq);
```

A New Workflow...



Deploy and Test



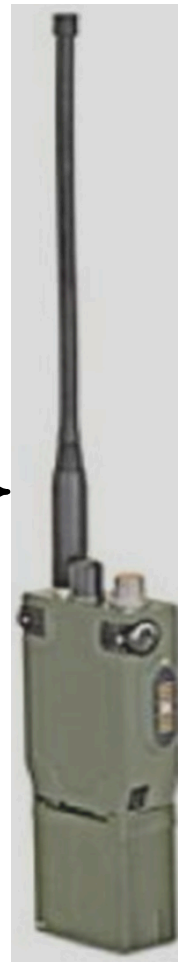
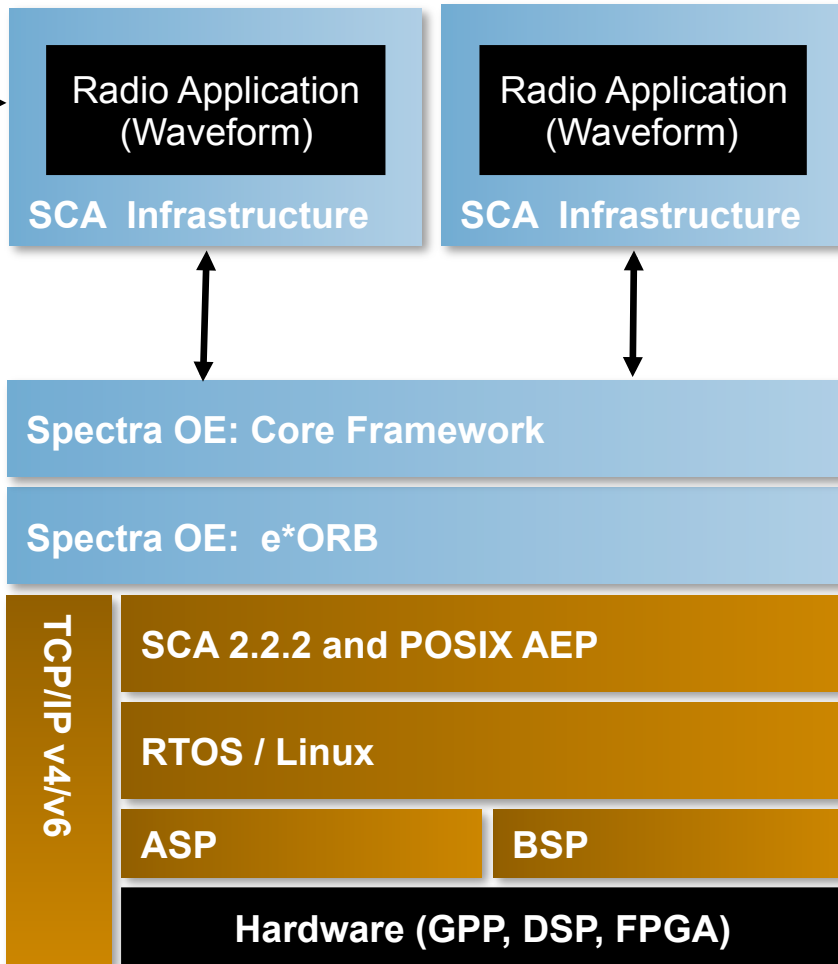
Spectra CX

Deploy WF

Instantiate WF

Control WF

Monitor WF



- ▶ Operating Environment in the loop
 - ▶ Testing against 'Golden Waveform'
 - ▶ Hand written parts of the waveform
- ▶ Automate glue
 - ▶ Simulink libraries, object files and header files
 - ▶ Behavioral code
 - ▶ Optimize glue code

- ▶ **Rapid** design, development, testing and deployment
- ▶ **Minimize** manual coding effort
- ▶ **Reduce** opportunity for human error
- ▶ **Reduce** time from requirements to deployment