SVG in Data Acquisition and Control Systems

Tao Jiang
Boston Scientific
tao.jiang@bsci.com
Test Cell Engineering

Cenco International™ covers virtually all areas of engine testing:

- Commercial Large Fan
- Small Propulsion Turbofan
- Military Jet Engines (dry and wet)
- Turboshaft & Turbopropeller Engines
- Auxiliary Power Units (APU)
- Engine Components and Accessories Benches
- Space Propulsion and Test Equipment.

Typical Projects and Services:

- Indoor and Outdoor Turnkey Test Facilities
- Facility Upgrades
- Technical Audits
- Design-Only Projects
- Customer Care.

Quality at Your Service

We continue to improve our products, processes and performances, and are working according to many quality certifications, such as ISO 14001, ISO 9001:2000, EN 9100, EASA Part 21, EASA Part 145, FAR-145, AQAP-2110, OEM qualifications.
The Complete Solution To Your Engine

As the prime contractor for your turnkey project, Cenco International™ ensures full-time supervision, complete schedule coordination, financial management and quality control of all subcontractors. Providing you with a complete test facility from a single source.

Turnkey Projects

- Facility Design
- Construction
- Design and Manufacture
  of all the engine specific test and test nacelle equipment, including thrust stand, engine adapter, and complete computerized data acquisition and command system
- Installation and Calibration
- Commissioning and Correlation Support
- Customer-Oriented Support
  from our team of specialists, including training, help desk, maintenance, and upgrades through on-site or remote support.

Facility-Upgrade Projects

Cenco International™ provides flexible Project Management of your test activities such as:
- Technical Audits
- Data Acquisition
- Control systems
- Test equipment.

Technical Audit and Acoustic, Aerodynamic, Structural Design

Definition and understanding of your unique test requirements, now and in the future, is vital for the success of a test facility project. Cenco International™ provides professional audit and design services in all areas of testing including:
- Facilities
- Aerodynamics and Acoustics
- Engine preparation
- Mechanical and Electrical requirements
- Engine transport systems
- Support equipment such as fuel, oil, air start.
Engine Test Equipment

Cenco International™ designs and manufactures the special test equipment required for all engine testing activities. Cenco International’s™ products are developed by a staff of highly-trained engineers using a wide range of state-of-the-art computer-aided design and engineering tools. This ensures that our products meet the highest quality standards, while maintaining short development times and competitive prices.

Engine Control Systems
Engine throttle control systems, Engine Control Modules (PECM), Instrumentation, Programmable Logic Controls (PLC) for facility and engine control, vibration systems

Engine Test Systems
Engine thrust stand, engine dynamometer test stand, turboshaft test stand, engine adapters, mobile engine test stands, aerodynamic devices and noise suppression systems, exhaust diffusers and collectors

Equipment Test Benches
Test systems for engine components, such as lubrication units, and APU’s

Engine Transportation and Handling Systems
Transport carts, storage carts, monorail handling systems

Services and Auxiliary Equipment
Fuel, air, electrical supply and conditioning systems, lubrication and hydraulic systems, start systems, fire protection systems, monitoring systems.
Data Acquisition and Control Systems

Cyres.mp combines Cenco US industry-leading CERES software and Cenco Belgium award-winning MCC software.
Cyres.mp gives users extremely powerful and flexible tools for any jet engine testing application.

User-Friendly Approach

- Graphical User Interface (GUI)
- Multi-platform support: VXI, PXI/SCXI, VME, PCI, PC, PLC and various LAN-based instruments
- Browser-based interface
- Real-time data display through the test cell LAN, the customer’s intranet, or over the Internet.

Easily Configured

- Create real-time displays, add or modify calculations, define limits, create functional test logic
- Easy conversion of existing CERES and MCC engine-specific configuration files.

Wide Range of Screen Widgets

- Numeric or graphical test parameters
- Vertical or horizontal bars
- Digital gauges
- Real-time plots
- Digital displays.
Inside Data Acquisition and Control Systems

- The system is used by engine test operators and engine performance engineer. No special computer knowledge is required.

- User interface and system control logic are greatly affected by the hardware selection and test cell design itself.
  - Different customer might have quite different look-and-feel for their systems even though they are designed to test the same type of engine.

- Need to be field-programmable during system installation even during engine run.
New Data Acquisition and Control System

- In late 2003, Cenco started a project to build its next generation engine test system. The first goal is to replace aging X window based UI with browser based user interface to stand out from competition.
- Remotely monitoring and controlling capability
- Live real-time test data updating at least 10 frames/second
- Improve Test Data Reporting Capability
- Easy Configuration
The New Engine Control Display

CFM56 Engine Control

- Fire Loop Status
  - Fan Loop A Normal
  - Core Loop A Normal
  - Fan Loop B Normal
  - Core Loop B Normal

- Fuel/Start Controls
  - Fuel Off
  - Starter Off
  - Master Lever
    - ON
  - Manual Start
    - ON
  - Mode Select
    - Normal
    - Ignition Power
    - ON
    - Ignition Crank

- Engine Controls
  - QMOP
  - Nac T
  - SAP
  - FSP
    - PSI
    - Deg C
    - PSI
    - PSI

- Additional Controls
  - Oil Press Normal
  - Engine Run
  - Low Hyd Pressure
  - Bid/Viv Reg Valve
  - High Press Reg Valve
  - IDG Filter Clogging
  - IDG Oil Press Low
  - IDG Disconnect
  - High Press Cowl AI Valve
    - Open
    - Close
  - Hyd Pump Press
    - Open
    - Close
    - De-Press
  - Idle Select
    - Flight
    - ON
    - Ground
  - EEC Gnd Pwr Ch A
  - EEC Gnd Pwr Ch B

8
Implementation Strategy

- Make best use of existing system (over 1M LOC) and replace those matter the most to customers
- Creating a web service layer to turn traditional data acquisition and control system into an open platform
  - Give user interface developer freedom to plug-in any presentation unit using the latest technologies.
- Create a component-based application framework for producing massive amount of user configurable displays at very low cost.
  - On average, over 20 custom displays for any single engine type.
    The development and testing effort would be enormous if coding directly in Java or C#.
- SVG is used to build UI components and glue them together.
Why SVG?

- SVG is a royalty-free vendor-neutral open standard.
- Excellent graphical results: each of UI component even the whole user interface could be developed using professional grade tool, such as: Adobe Illustrator.
- No need to compile and build applications. SVG player can directly render SVG files to have rich graphical results.
- Resolution Independent: SVG content can be viewed on monitors with different resolutions or enlarged/shrunken without losing quality.
- No need for UI layout manager: all components are absolutely positioned and proportionally resized.
- Minimal coding effort: coding effort only involves in wiring events and data to SVG elements but it can be simpler.
Architecture Overview

Client Layer
- Excel Reporting System
- Java Applet/Application for standard displays
- User Configurable Displays in SVG
- .NET Application (DVR)
- TV Monitor

Web Service Layer
- Web Service for Engine Test and Calibration
- Web Service for Test Report
- Web Service for Plot Analysis
- Real time test data Streamer

Data Processing and Control Layer
- Test Data Calculations
- Limits
- Test Control Logic
- Calibration
- Data and Event Logging

Device Driver Layer
- Fanuc PLC Driver
- External Controller Driver
- MPI Driver for Vibration Data
- PSI Driver for Pressure Data
- DTS Driver for Temperature Data

Apache HTTP Server
- Web Service
  - UNIX domain socket (aka. Named Pipe)

History
- Multi-channel DVR
  - records test process
- Continuous Data Logging
- Transient Data Logging

Data Retrieved

HTTP/HTTPS

Video Playback

Real time test data Streamer

Data Retrieved

Excel Reporting System

Java Applet/Application for standard displays

User Configurable Displays in SVG

.NET Application (DVR)

TV Monitor

Web Service for Engine Test and Calibration

Web Service for Test Report

Web Service for Plot Analysis

Real time test data Streamer

Test Data Calculations

Limits

Test Control Logic

Calibration

Data and Event Logging

Fanuc PLC Driver

External Controller Driver

MPI Driver for Vibration Data

PSI Driver for Pressure Data

DTS Driver for Temperature Data

Oracle Database

Continuous Data Logging

Transient Data Logging
Client Layer

- **Excel reporting system:** is a test report design and generation system built on the top of Microsoft Excel.

- **.NET application for DVR playback:** is used to playback test data in-sync with multi-channel video recording on the engine test process.

- **User configurable displays in SVG:** are displays specially ordered by customers and they can be changed in the field or even during the engine test in progress.
SVG based Displays Rendered by Adobe SVG Viewer

Browser Based Web Client

Adobe SVG Viewer

SVG based Displays

Update SVG DOM

Response User Interaction

JavaScript

Decoded binary stream

Make SOAP Call

Invisible Java Applet

Real time test data Streamer

Web Service for Engine Test and Calibration

SVG GUI Component library.
Different Approaches in creating SVG GUI component library

- UI components can be defined as “symbol” elements and put together inside a component library. (Jay Nick: http://www.jaynick.com/ SVG/)
  - SVG “use” element can refer to external “symbol” element as template and this is how SVG component gets reused.
  - However, event handler cannot be encapsulated into symbol.

- JavaScript library.
  - Limited functionality at the time. (Kevin Lindsey: http://www.kevlindev.com/)
  - JavaScript was hard to debug.

- Define new SVG tag (SVG 1.2) for reusable component.
Risks Analysis for SVG 1.2 Approach

- **Risks:**
  - SVG 1.2 specification was still at draft stage.
  - ASV6 was only a pre-release.
  - No SVG authoring tool for SVG 1.2.

- **Rewards:**
  - Go SVG 1.2 costs only a fraction of other alternative approaches.

- **Bottom Lines:**
  - ASV6 has by far the greatest functionalities and the best graphic quality at that time.
  - RENESIS player promised to support full SVG 1.2
  - Develop a SVG viewer with limited SVG 1.2 support.
Custom UI Components

- Create custom UI components in SVG.
- Define new SVG tags for UI components.
- There is no need to do JavaScript programming because each UI component's behavior is controlled through its parameter's value.

```xml
<ui:momentarybutton enabled="1" height="70" icon="" id="ui:momentarybutton315"
tooltip="Momentary Push Button" transform="matrix(1 0 0 1 472.52470204846236 -9)" value="0" visible="1"
width="75" x="174" y="583" buttonStyle="3">
<parameter name="IDGDisc_MB" value="0"/>
<text_style font="Arial" fontstyle="bold" font-size="14"/>
<label_config>
    <offlabel color="green" label="IDG\nDisconnect" text-color="black"/>
    <onlabel color="orange" label="IDG\nDisconnect" text-color="black"/>
</label_config>
</ui:momentarybutton>
```
Each UI component has one or multiple data objects (e.g. plot parameters), representing the component data value, attributes, etc.

The application framework takes care of issuing SOAP command to server side and distributing streaming data back to UI components.

The client side developer only needs to take care of assembling UI component and parameter mapping. No Java or JavaScript coding.

The data processing engine is configured similar to PLC’s ladder logic using VB like language without looping statement.
Display Editor
System Performance on SVG Approach

- Browser based solution (native or plug-in) gets UI elements updated by updating SVG DOM.
- Jet engine testing has lots of parameters (20+ per display) and they need to be updated at least 10 frames/second.
- Our test result showed the maximum refreshing rate can only achieve 5 frames/second with only one display page opened. The bigger screen/higher resolution, the lower the refreshing rate.
- Unfortunately, Adobe abandoned the development on SVG but it didn't open source its SVG Viewer.
- The release of RENESIS player had been delayed many times and we never got chance to use it.
- We decided to render SVG ourselves.
Using Batik SVG Toolkit

- Batik is a Java-based SVG toolkit.
- Using Batik Swing components (core) module, each custom SVG component definition is used as a resource file to create JSVGCanvas object and then inserted into JPanel.
- However, JSVGCanvas takes long time to load SVG component and uses lots of memory.
- Later, we developed a simple yet efficient SVG rendering library by simplifying Batik’s low level modules.
- JSVGCanvas is still used for some special and complex components, such as: gauges, stopwatches.
Any custom component can be created by sub-classing JComponent. Basically, do everything from scratch.

The simpler alternative is to use custom UI delegate.

- Most custom UI components could find surrogate hosts in Java Swing components for managing state and event handling.
- Creating a custom UI delegate and using SVG rendering engine to render the UI.

SVG definition in UI component library is used as resource to define look and feel.

Other custom components, such as: Plot, has to be handled differently because it is hard to implement direct in either Java or SVG.
Plots used in Jet engine testing, including: performance plot, real-time plot and real-time scrolling plot.

They are implemented using JFreeChart package directly without corresponding SVG implementation in the component library.

A new custom SVG tag for plots is created to configure the chart behavior without coding in Java for end user.
Real-Time Enhancement on JFreeChart

- JFreeChart is excellent in creating large variety of high quality chart.
  - But, It doesn’t support real-time charting.
  - Jet engine testing requires real-time charting at 10Hz.
- The idea of real-time enhancement is very simple:
  - Let JFreeChart use chart buffer.
  - Draw only two points at a time instead of all points in its data set.
  - Redraw all points only when chart is resized.
  - The updated JFreeChart can do real-time charting over 30Hz.
- For more detail, please see my post on JFreeChart Forum at:
Project Update

- **Project Time Line:**
  - The project was started in late 2003.
  - First working prototype debut on Paris Air Show 2005.
  - First sales contract (N3 engine overhaul service in Germany) was awarded at the end of 2005.
  - First system (N3) in production service and a formal grand open ceremony on Sep. 14, 2007.

- **Until now, Cenco's new data acquisition and control system has been installed and in production use in Germany, India, Saudi Arabia, Egypt, US, China and growing.**
Conclusion

- SVG helps closing gap between UI designer and developer. UI designers have total freedom to design whatever they want.
- SVG makes easy to develop custom UI components.
- Application framework makes possible to develop a full fledged application by simply assembling components and parameter mapping.
  - Cost of building and testing are low.
  - At Cento, many customers can make displays themselves.
- Future Application:
  - Current implementation in Java takes relatively long time to load initially. It is easy to port to other implementations, such as: WPF
  - Using Flash to host SVG.
  - Pure browser solution using JavaScript library for SVG.
Questions?