Learning to Love Text Again
With Measurement Studio

NIWeek 2019
Thursday, May 23
Mark Locascio
Senior Project Engineer - DMC, Inc.

Kyle Mitchell
Systems Engineer - DMC, Inc.
Who Are You?
Outline

- Motivation
- What is Measurement Studio?
- Leveraging Existing Knowledge
- Adding New Tools
- DMC’s Best Practices
Established in 1996, DMC serves customers worldwide from offices in Chicago, Boston, Dallas, Denver, Houston, New York, Seattle, and St. Louis.
Areas of Expertise

- Manufacturing Automation & Intelligence
- Embedded Development & Programming
- Test & Measurement Automation
- Application Development
- Digital Workplace Solutions

Logos and Partnerships:
- ARMmbed
- National Instruments Alliance Member
- Microsoft .NET Silver Partner
- Microsoft Partner
- SystemLink Specialty

Other Logos:
- Siemens
- Rockwell Automation Systems Integrator
- Linux
- National Instruments
- Microsoft
- C#
Expectations for this Session

- Motivate the use of .NET for test & measurement
- Provide awareness of the tools available
- Discuss why we like it at DMC
- Provide resources for learning additional skills

Fightin’ Words

```python
while (x == y) {
    something();
    somethingelse();
}
```

```python
while (x == y) {
    something();
    somethingelse();
}
```
Key Concepts
The Right Tool for the Job

- Engineers get paid to know the right tool for the job, and how to use it!

Alex Agrees

Something something something Python, .NET, something something **right tool for the job!**

Alex Davern, NI CEO
May 21, 2019
1. Use the *right tool* for the job
Why the Hesitation?
Why the Hesitation?
Strengths of Programming in LabVIEW

- Quick, easy development
- Easy hardware integration
- Standard library & toolkits with engineering focus
- UI development doesn’t get any easier
1. Use the *right tool* for the job
2. Best uses cases for .NET are *large projects*
Where Can We Do Better?

- Tools for multi-developer scenarios
- Huge library of packages for *everything*
- Really sweet dynamic UIs
- Best Practices!

Because your computer doesn’t like it as much as you do.
Key Concepts

1. Use the *right tool* for the job
2. Best uses cases for .NET are *large projects*
3. The humble yet mighty *diff*
Learning a New Language

- Syntax
- Standard lib
- Package 1
- Package 2
- Package 3

Difficulty vs. Time & experience

- Symbol representing difficulty
- Symbols representing time and experience

NATIONAL INSTRUMENTS
Learning a New Language
This Is Not as Traumatic as You Think

LabVIEW gives you most of the background you need

You don’t lose development/debugging functionality

You gain a lot of nice new tools!
Does Anyone Actually Do This?

Yep.
OK, So... What is Measurement Studio?!
Measurement Studio Is…

- A Visual Studio plug-in
- .NET UI controls (especially graphs!)
- .NET libraries for data analysis
- Additional debugging tools

Measurement Studio is what fills in the gaps that are left when you leave the pillow-y comfort of LabVIEW!
Measurement Studio Is NOT…

- LabWindows/CVI
- A specialized language
- A specialized IDE
Measurement Studio Is NOT...

- .NET hardware drivers
- Even **without** Measurement Studio, you have access to these
Measurement Studio Is What Makes The Transition Easy

<table>
<thead>
<tr>
<th>Feature</th>
<th>.NET Library Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquire data from instruments</td>
<td>DAQmx</td>
</tr>
<tr>
<td>Plot data / handle user interaction</td>
<td>Measurement Studio or third-party</td>
</tr>
<tr>
<td>Process data</td>
<td>Measurement Studio or third-party</td>
</tr>
<tr>
<td>Write data to TDMS</td>
<td>Measurement Studio or third-party</td>
</tr>
</tbody>
</table>

All standard test & measurement app features are supported in .NET

Measurement Studio will be faster to learn than third-party libraries
Learning a New Language

Difficulty

Time & experience

- Syntax
- Standard lib
- Package 1
- Package 2
- Package 3
## Direct Analogies to What You Already Know

<table>
<thead>
<tr>
<th>LabVIEW Palette</th>
<th>.NET Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure I/O</td>
<td>NationalInstruments.DAQmx</td>
</tr>
<tr>
<td>NI DAQmx</td>
<td></td>
</tr>
<tr>
<td>Sig Processing</td>
<td>NationalInstruments.Analysis.Dsp.Filters</td>
</tr>
<tr>
<td>Filters</td>
<td></td>
</tr>
<tr>
<td>File I/O</td>
<td>NationalInstruments.Tdms</td>
</tr>
<tr>
<td>TDM Streaming</td>
<td></td>
</tr>
</tbody>
</table>
Direct Analogies to What You Already Know

```csharp
using NationalInstruments.Analysis.Dsp.Filters;

var newFilter = ButterworthLowpassFilter(order, fs, fh);

filteredX = newFilter.FilterData(X);
```

Not needed for Lowpass filter type

Not needed because objects can have their own state
## Third-Party Tools – WPF Graphs

<table>
<thead>
<tr>
<th>Package</th>
<th>License</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Interactive Data Display</td>
<td>MIT</td>
<td><a href="bit.ly/dmctxt03">bit.ly/dmctxt03</a></td>
</tr>
<tr>
<td>OxyPlot</td>
<td>MIT</td>
<td><a href="bit.ly/dmctxt04">bit.ly/dmctxt04</a></td>
</tr>
<tr>
<td>LiveCharts</td>
<td>MIT</td>
<td><a href="bit.ly/dmctxt05">bit.ly/dmctxt05</a></td>
</tr>
<tr>
<td>Telerik UI for WPF</td>
<td>Commercial</td>
<td><a href="bit.ly/dmctxt06">bit.ly/dmctxt06</a></td>
</tr>
</tbody>
</table>
## Third-Party Tools – Processing/Analysis/Math

<table>
<thead>
<tr>
<th>Package</th>
<th>License</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicDSP (VB!)</td>
<td>???</td>
<td>[bit.ly/dmctxt08]</td>
</tr>
<tr>
<td>Flying Frog FFT Library</td>
<td>Commercial</td>
<td>[bit.ly/dmctxt09]</td>
</tr>
<tr>
<td>ALGLIB</td>
<td>Commercial</td>
<td>[bit.ly/dmctxt12]</td>
</tr>
</tbody>
</table>
# Third-Party Tools – TDMS Logging

<table>
<thead>
<tr>
<th>Package</th>
<th>License</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some Guy’s TDMS Library</td>
<td>MIT</td>
<td>bit.ly/dmctxtxt13</td>
</tr>
</tbody>
</table>
Familiar Content
using NationalInstruments.DAQmx;

var task = new Task("MyTask");
task.AddGlobalChannel("MyChannel");
task.Start();
task.WaitUntilDone(10);
task.Stop();
task.Dispose();
TDMS API

```
using NationalInstruments.Tdms;

var tdmsFile = new TdmsFile("C:\filePath.tdms", new TdmsFileOptions());
var channel =
    tdmsFile.AddChannelGroup("DataGroup").AddChannel("DataChannel", TdmsDataType.Double);
channel.AppendData<double>(123.45);

tdmsFile.Close();
```
This Is Not as Traumatic as You Think

LabVIEW gives you most of the background you need

You don’t lose development/debugging functionality

You gain a lot of nice new tools!
Debugging
What You Don’t Lose

- Breakpoints
  - Traditional
  - Conditional
- Stepping (highlight execution)
- Data tips (probes)

What You **Do** Gain

- Edit code and continue debugging
- Change a variable in memory
- Change the execution flow

```csharp
private void Update()
{
    this.Clear();
    try
    {
        foreach (FileInfo f in _directory.GetFiles("*.jpg"))
            Add(new Photo(f.FullName));
    }
}
```
This Is Not as Traumatic as You Think

- LabVIEW gives you most of the background you need
- You don’t lose development/debugging functionality
- You gain a lot of nice new tools!
Where Can We Do Better?

(as test & measurement engineers, not just as people in general)
Best Practices:

- Team Development
- Separation of Concerns
- UI Development
Multi-developer Workflow

Technical lead

Select merge request → Review branch → Merge branch

Developers

Select open issue → Create branch → Implement feature/fix → Submit merge request

Anybody

Test → Document issues

All enabled by text!
Why We Like GitLab

- Tools for reviewing code
  - Diffs
  - Discussions
  - User management
Why We Like GitLab

- Tools for reviewing code
  - Diffs
  - Discussions
  - User management
- Tools for managing branches
  - Issue tracking
  - Branch comparisons
  - Commit visualization
Why We Like GitLab

- Tools for reviewing code
  - Diffs
  - Discussions
  - User management
- Tools for managing branches
  - Issue tracking
  - Branch comparisons
  - Commit visualization
- Tools for deployment
  - Continuous integration/delivery/deployment pipelines
  - Build artifact hosting
Best Practices:

- Team Development
- Separation of Concerns
- UI Development
Programming Paradigms

- Flow-driven
- Object-oriented
- Event-driven

- Imperative
- Object-oriented
- Event-driven
Object-Oriented LabVIEW (LVOO)

- OO is a better architecture choice in a lot of situations
- Better encapsulation
- Powerful inheritance model
- Easier to debug
- Scales better with team size and project complexity

- Maintain dataflow paradigm
- Recognize this isn’t for every case

Why We Like “Traditional” OO

<table>
<thead>
<tr>
<th>Feature</th>
<th>LabVIEW</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Polymorphism</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inheritance</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Interfaces</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Static members</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Static methods</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Overloaded methods</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Native object references</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Interfaces: What Are They?

... to provide the following:

- An integer property called `count`
- A `run()` method
- A `stop()` method

There is no implementation
Interfaces: How Are They Used?

Just like any other type!

```java
public interface IQuackable
{
    void quack();
}

public class Animal
{
    ...
}

public class Duck : Animal, IQuackable
{
    void quack();
}

public class DuckCall : IQuackable
{
    void quack();
}
```

```java
public void makeNoise(IQuackable ducklike)
{
    ducklike.quack();
}
```
Interfaces: *Why* Are They Used?

**Hotmail**: Hierarchy

- Events
  - Personal
  - Work
- Expenses
  - Personal
  - Work
- Travel
  - Personal
  - Work

**Gmail**: Tags

- Events
- Expenses
- Personal
- Work
- Travel
Simplify Inheritance with Interfaces

- This works
- It’s awesome
- It would be great if we could simplify it
Best Practices:

- Team Development
- Separation of Concerns
- UI Development
Strengths of Programming in LabVIEW

- Quick, easy development
- Easy hardware integration
- Standard library & toolkits with engineering focus
- UI development doesn’t get any easier
Separation of Concerns / the Single-Responsibility Principle

- Break problems into small parts
- Give each part **one** job
- Keep each part independent

You’ll see a lot of stuff like this, with initialisms like “MVC,” “MVP,” and “MVVM”
Separation of Concerns / the Single-Responsibility Principle

- Break problems into small parts
- Give each part **one** job
- Keep each part independent

This is what we really care about…

WPF is How We Separate These Concerns

- Describes business logic only
- Developers don’t need design skills
- Text for easy revisioning

- Describes UI layout only
- Designers don’t need development skills
- Text for easy revisioning

- Hardware-accelerated vector graphics
- Databinding

C#  XAML

XAML Controls

Out-of-the-box WPF

- Button
- Calendar
- Check box
- Slider
- Text box
- Image
- Progress bar

Measurement Studio

- Graph/chart (!!!)
- LED
- Gauge
- Knob
- Switch
<ni:Graph DataSource="{Binding myVariable}" />

<ni:Graph.Axes>
    <ni:AxisDouble Adjuster="ContinuousChart" Orientation="Horizontal"
        Range="0,1000" />
</ni:Graph.Axes>

<ni:LED Value="True" />
<ni:LED Value="{Binding myVariable}" Width="20"
    Height="20"
    FalseBrush="Transparent"
    TrueBrush="Red"0,1000" />

Key Points:
1. This is pretty easy
2. You don’t need to know anything about the business logic
3. It’s happening in its own (text) file
Why We Like XAML

- Loose coupling
- Separation of concerns!
- Diff & merge

- High level of customization
- Easier to resize windows!

- Can be templated/styled/themed like HTML

Everything Else
## More Tools Become Available

<table>
<thead>
<tr>
<th>Feature</th>
<th>.NET Library Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquire data from instruments</td>
<td>DAQmx</td>
</tr>
<tr>
<td>Plot data / handle user interaction</td>
<td>Measurement Studio or third-party</td>
</tr>
<tr>
<td>Process data</td>
<td>Measurement Studio or third-party</td>
</tr>
<tr>
<td>Write data to TDMS</td>
<td>Measurement Studio or third-party</td>
</tr>
<tr>
<td>Sweet debugging tools</td>
<td>Visual Studio</td>
</tr>
<tr>
<td>Sweet UI tools</td>
<td>.NET XAML</td>
</tr>
<tr>
<td>Sweet DB tools</td>
<td>.NET Entity Framework</td>
</tr>
<tr>
<td>Anything else your heart desires</td>
<td>NuGet packages</td>
</tr>
</tbody>
</table>
Object-Relational Mapping (ORM)

- I don’t ever want to write another SQL query ever again
- ORM writes it for you
- Just manipulate *objects* in your code

### Language | ORM Library
--- | ---
Python | SQLalchemy
Ruby | ActiveRecord

```csharp
public class Blue
{
    public int ID { get; set; }
    public string Name { get; set; }
    public ICollection<Red> Children { get; set; }
}
```

```csharp
public class Red
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int ParentID { get; set; }
    public Blue Parent { get; set; }
}
```
T&M Example Using the Microsoft Stack

Network

Web server

.NET Core

Web application

Entity Framework

ORM

Database server

Microsoft IIS

C#
T&M Example Using the Microsoft Stack

- Scalable
- Managed infrastructure
- Protected data

Internet

Azure

- .NET Core
- Entity Framework
- SQL Server

Web application

ORM

Database server
Feels Like Too Much?
You Are Probably Around Here
Roadmap

LabVIEW OO
Measurement Studio Docs
Visual Studio
C#
TOOP
Measurement Studio
.NET Framework
WPF
LINQ
Entity Framework
Success / Life of Leisure

Questions? Contact Us...

www.dmcinfo.com
sales@dmcinfo.com
312.255.8757
Questions? Contact Us…

www.dmcinfo.com
sales@dmcinfo.com
312.255.8757
Before you go, take the survey.
Download and Login to the NIWeek Mobile App

VIEW WEEKLY SCHEDULE

FIND YOUR SESSION

BUILD YOUR SCHEDULE

More Information at ni.com/niweek
Stay Connected During and After NIWeek

ni.com/niweekcommunity
facebook.com/NationalInstruments
twitter.com/niglobal
youtube.com/nationalinstruments