QA at Aerotech

CS1632

Agenda

- Introductions
- Aerotech Overview
- Motion Controller Products - Development and QA Overview
  - What is a Motion Controller?
  - How do we test the software?
  - How do we test the electrical hardware devices?
- Enterprise Software - Development and QA Overview

Introductions

- Individual Introductions
- We're glad you all are taking this course because QA is a huge part of software and this class should be required
- New hires are routinely surprised by how much time is spent on QA and how important it is to developers
- At Aerotech, we don't just write code and give it to someone to test
  - You test it yourself, integrate it, test it again, then have it tested by QA
  - Everyone is part of QA, knowing good QA makes you a better developer

What does Aerotech do?

R&D, design, engineer, prototype, build, optimize, deploy, support

We provide automation equipment, and automation impacts everyone – everywhere.

Touch screens

Angioplasty (stent surgery)

Additive manufacturing

Computer chips

Network communications (fiber optics)

Intraocular lenses (cataract surgery)

DNA sequencing

Magnetic resonance imaging

Computer hard drives

Precision optics & lenses

We provide automation equipment, and automation impacts everyone – everywhere.

Our products and engineering contribute to scientific breakthroughs.

We design and create automation and motion control products, and we ship them worldwide.

What does Aerotech make?

Things that move back and forth, or spin around in circles...

What Does Aerotech Make?

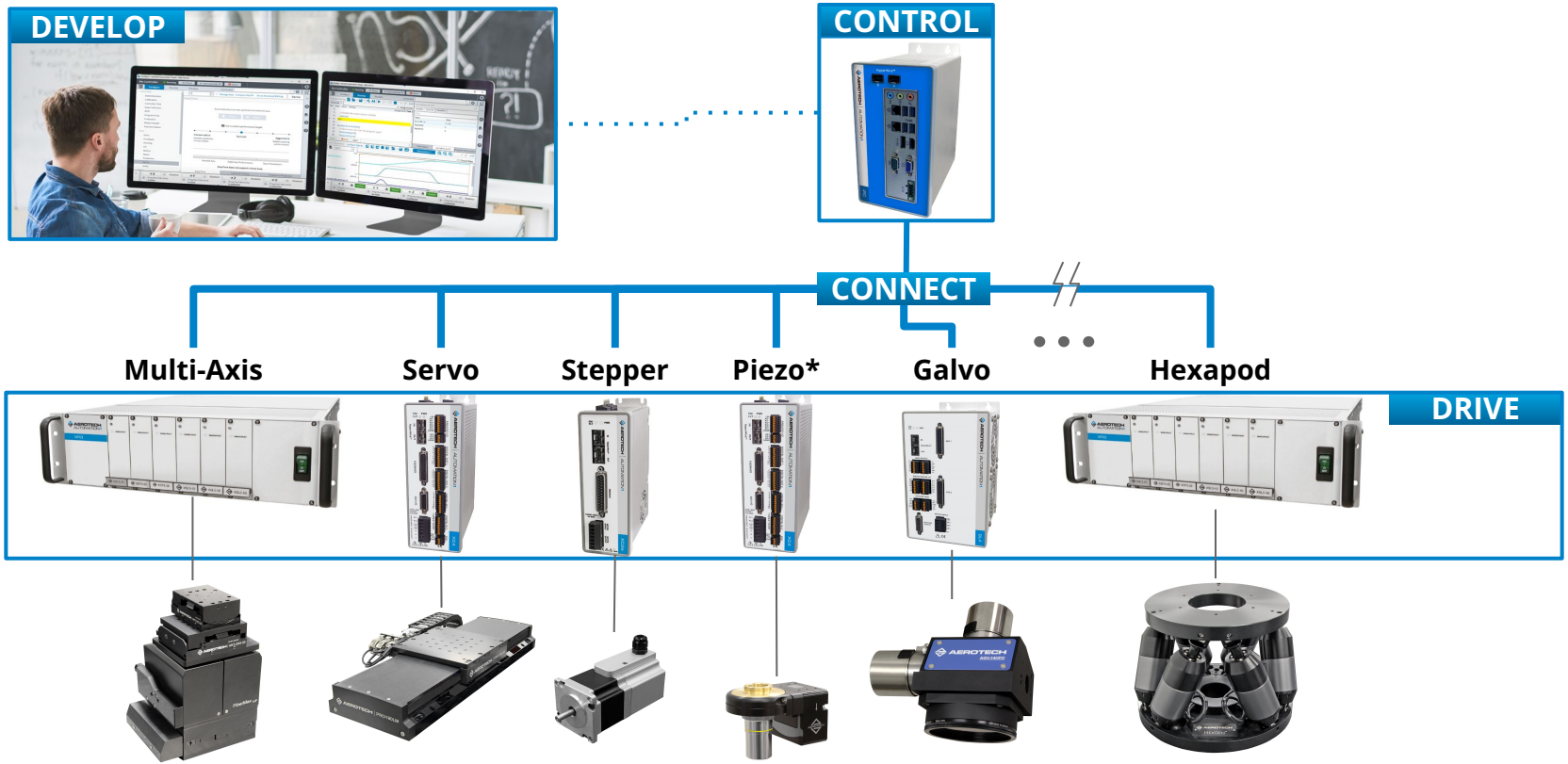
Customers want...

- Motion & positioning solutions
- Integrated automation systems
- 3D metrology systems

We make...

- Motion Control software & electronics
- Mechanical devices
  - Stages & actuators
  - Motors
  - Laser scan heads
  - Gimbal & optical mounts
  - Hexapods
  - Piezo nanopositioners

Motion Control Product Overview



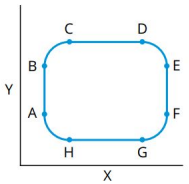
Motion Controller Basics

AeroScript Language

```
// Motion Commands
MoveLinear([X, Y], [0, 50])
MoveLinear([X, Y], [50, 0])
MoveLinear([X, Y], [0, -50])
MoveLinear([X, Y], [-50, 0])
```

MODERN • POWERFUL • REALTIME

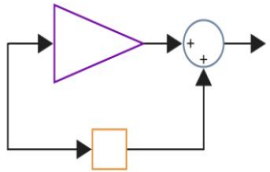
Motion Engine



CONTOUR • P2P • HOLD POSITION

Command Position, Velocity and Acceleration

Servo Loop

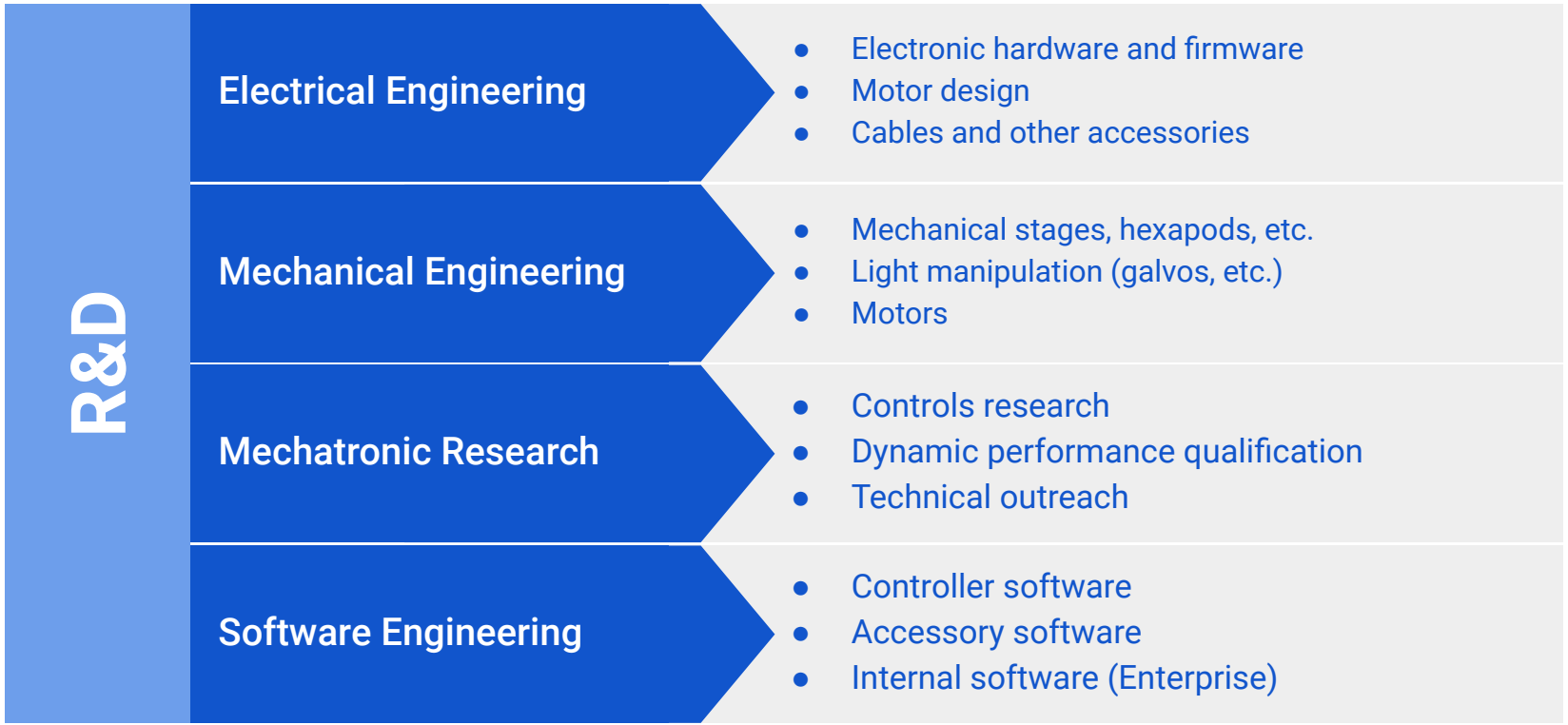


PID CONTROL • 20kHz

Motor Output (A)

Feedback Position

Four Departments of R&D



Software Engineering

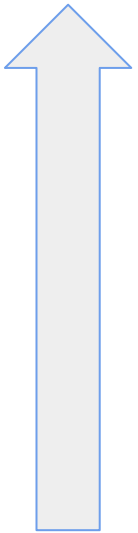
Organized by Function in the Software "Stack"

Controller Teams

Controller Architecture and Libraries
(SWE - Full Stack Team)

Help Documentation (SWE - Documentation Development Team)	• Embedded and live help • Product manuals
Application / User Interface (SWE - Application Development Team)	• Automation1 Studio • Automation1 Status Utility • CADFusion
Motion Controller (SWE - Motion Development Team)	• Real-time operating system • AeroScript language • Trajectory generation
Firmware (EE - Hardware Development Team)	• Servo and current control loops • HyperWire® communication protocol • Signal processing

User facing



Hardware facing

Software Quality Assurance at Aerotech

- What makes Aerotech unique
- The culture we foster
- What our software stack looks like
- What challenges we face for QA
- The solutions we've come up with

What makes Aerotech unique?

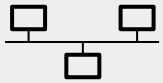
- We don’t have a dedicated QA department
  - Engineers ensure quality, “quality at the source”
- Every few weeks all engineers become QA
  - “Round robin testing”
- Verification
  - Testing to make sure a feature is correct according to the author
- Validation
  - Testing to make sure the product works the way a user would want it to
  - Validation takes weeks of time
- Tests are run automatically on a test server
  - Controlled environment, very repeatable and deterministic
  - True good QA is a cultural mindset during development

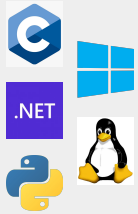
What makes Aerotech unique?

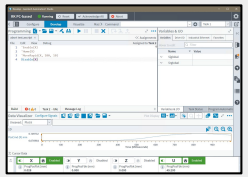
Mechanical device

Electrical device

Software on a Windows x86 PC







Electrical signals → Firmware → Motion engine → Comm libraries → APIs → Apps

C++, FPGA (DSP)

Real-time C (x86)

C++, C# (x86)

C, C#, Py

C#

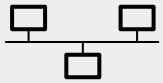
PC-Based Controller

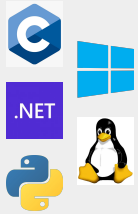
What makes Aerotech unique?

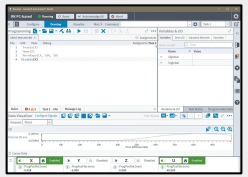
Mechanical device

Electrical device

Software on a Windows x86 PC







Electrical signals	Firmware	Motion engine	Comm library	Comm libraries	APIs	Apps
	C++, FPGA (DSP)	Real-time C (ARM)	C++ (ARM)	C++, C# (x86)	C, C#, Py	C#

Drive-Based Controller

What makes Aerotech unique?

- Large stack
  - Windows GUI → APIs → TCP/IP → C (ARM) → C++ (DSP) → FPGA → Circuits
- Multiple languages
  - C, C++, C#, Python, all at different layers
- Multiple platforms
  - Windows x86, Windows x64, Linux x64, Linux ARM, DSP
- Duplicated code paths
  - Identical C++ and C# communication libraries
- Real-time C code
  - Motion engine in C RTOS on Windows x86 and Linux ARM
- Mechanical devices in the loop

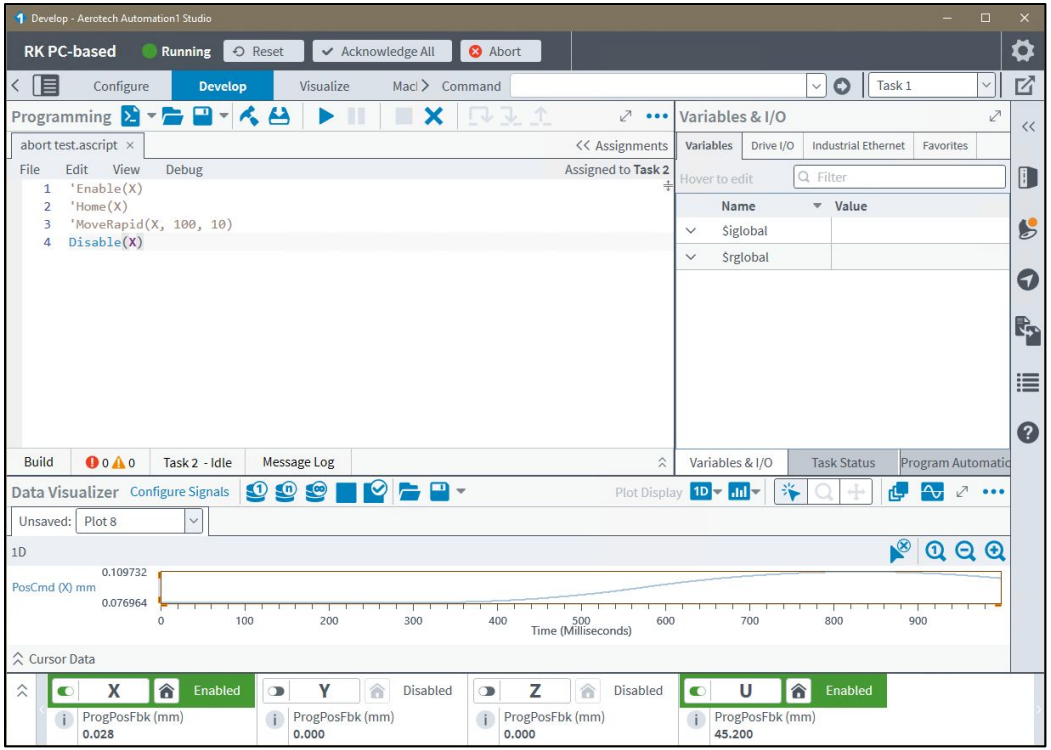
How do we test everything?

- Testing is hard
  - Can never test everything, accept the risk
- Have an excellent QA-focused culture
  - Lots and lots of manual testing, checking your own work, etc.
- Design the code to be testable
  - Good layers of abstraction mean easier to test
  - Bad abstractions mean more integration tests instead of unit tests
  - Avoid leaky abstractions
- You will have *lots* of integration tests
  - They will be slow too

How do we test everything?

Large stack

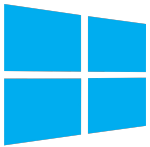
- No automated GUI testing
  - UI changes too frequently
  - Mostly manual testing
  - MVVM friendly unit tests
  - Only unit tests for the logic

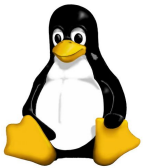


How do we test everything?

Large stack

- Lots of automated API tests
  - Mix of both unit tests and integration tests
  - Much of the APIs cannot be tested without the motion engine
  - Integration tests for behavior and logic
  - Unit tests for data structure and input validation
- Windows vs Linux
  - Tests run on Windows nightly, automatically
  - Tests run on Linux manually, periodically





How do we test everything?

Multiple languages

- Unit testing C# and Python is easy
  - The languages have features to support testing
- Unit testing C is hard
  - No way to expose internals to tests but not customers
  - No C unit tests
  - Lots of C integration tests
- C++ is somewhere in the middle
  - Not easy but not impossible

How do we test everything?

Multiple platforms

- We extensively test Windows, not so much Linux
  - The biggest hole in our automated testing
  - Manual testing is always on both
- “Trust your tools”
  - We assume the cross-platform code we write is truly cross platform
  - .NET Core, C++17, Python
- Motion Engine tested extensively for both platforms
  - Windows x86 vs Linux ARM
  - Lots of acceptance tests for both platforms

How do we test everything?

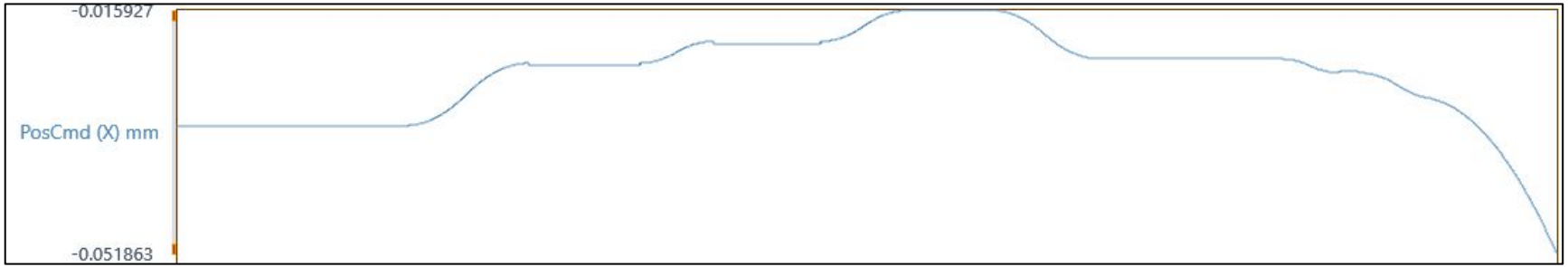
Duplicated code paths (C++ and C#)

- Duplicate all the same unit tests in C++ and C#
  - Make sure unit tests are assuming the same behavior
- Unit tests for the communication protocol
  - A protocol exists for communication library (think protobufs)
  - Unit test that each language serializes and deserializes into same data
- Rely on integration tests
  - These confirm C++ and C# can directly communicate

How do we test everything?

Real-time C code

- Can’t be unit tested, do acceptance testing instead
- “Profile” tests
  - Execute motion, compare trajectory to a known good golden copy
  - Identical profile tests run for Windows x86 and Linux ARM
  - Verifies repeatability of motion
  - Very slow



How do we test everything?

Real-time C code

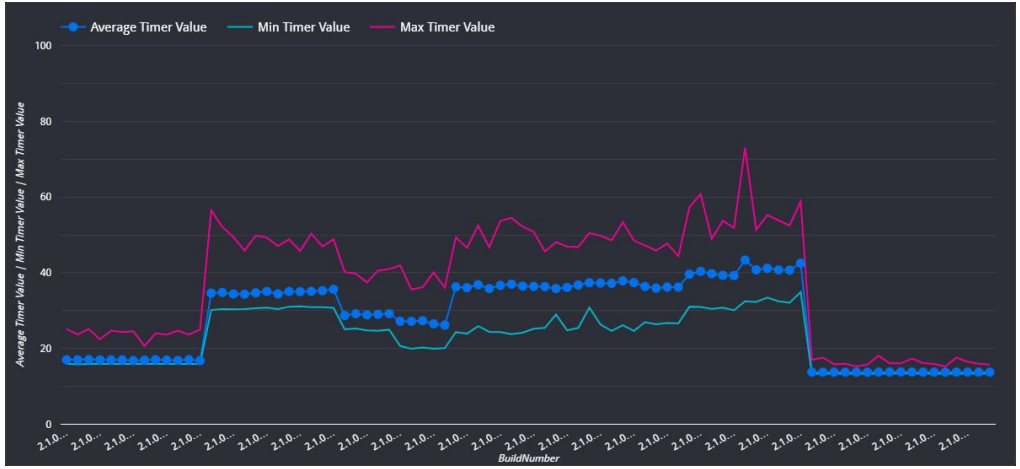
- Can’t be unit tested, do acceptance testing instead
- “Program” tests
  - Run an AeroScript program, verify the motion engine’s ending state
  - Some support for asserts
  - Mainly testing flow control
  - Doesn’t test how long it took

```
// End 50 of the critical sections
for $index = 1 to 50
    CriticalSectionEnd()
end

// Verify the correct nesting depth
$depth = StatusGetTaskItem(TaskGetIndex(), TaskStatusItem.CriticalSectionsActive)
ASSERT($depth == 50)

// Verify the correct timeout
$timeout = StatusGetTaskItem(TaskGetIndex(), TaskStatusItem.CriticalSectionTimeout)
ASSERT_WITH_TEXT($timeout == 450, IntegerToString($timeout))

// EXPECTED: Will error on the last start, because it will exceed the limit
for $index = 1 to 51
    CriticalSectionStart()
end

// If something goes wrong and we don't error, still finalize the test, but assert so we know we got here.
ASSERT_WITH_TEXT(0, "The task was supposed to error.")
FinalizeTest()
```

How do we test everything?

Real-time C code

- How do we test that the code is actually real-time?
  - Add instrumentation to code under test (timers)
  - Measure timers during all profile tests
  - Plot timers to see trends over many builds



How do we test everything?

Electrical devices

- We test hardware without software
  - We have hardware that tests our hardware…
  - In other words, make sure the circuit boards work, ignoring the software as much as possible
  - Also FPGA test benches

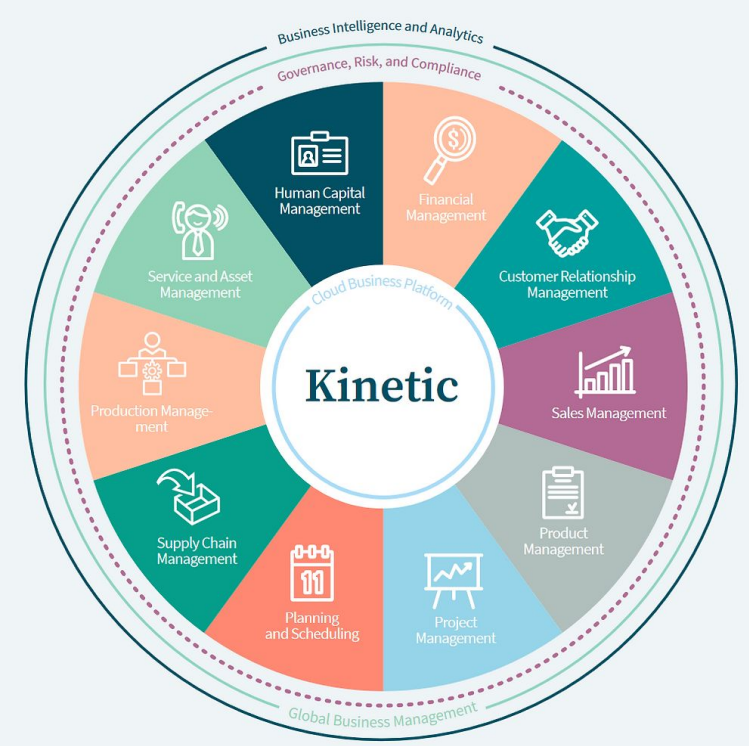
Beyond R&D

Aerotech utilizes software engineering best practices to support business operations.

Enterprise Development

Focus on building/improving business processes and integrations

- Enterprise Resource Planning (ERP) system is our key information hub
- Employees, customers, and other stakeholders depend on our enterprise systems every day across the world
- Enables us to:
  - Record business transactions
  - Schedule and prioritize work
  - Improve efficiency
  - React to changing business conditions
  - Capitalize on new technologies

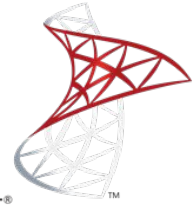


EPICOR

Enterprise Development

Development Tools and Practices

- Same IDE, source control, primary language (.NET C#), and issue tracker
- Similar approach to testing, verification, and validation
  - Added bonus of direct access to users for testing
- Integrations branch into many other technologies
  - E.g., Web APIs, relational databases, and cloud-hosted services



Questions?