

Automate Your Revit Add-In Testing with Unit Testing

Patrick Fernbach

Software Engineer

Corey Smith

Software Business Analyst





Patrick Fernbach

Lead Associate | Mechanical Engineering
Software Engineer
KLH Engineers, PSC

Patrick Fernbach specializes in HVAC and plumbing system design at KLH Engineers, PSC, and serves on the software development team. He translates the MEP engineers' needs to software engineers' in order to develop process improvements. He also assists in the creation of Revit add-ins and leads quality assurance and testing efforts to ensure the custom tools are of high quality. Patrick holds a Bachelor of Science in mechanical engineering from the University of Cincinnati.



Corey Smith

Mechanical Designer
Software Business Analyst
KLH Engineers, PSC

Corey Smith is a lead mechanical designer at KLH Engineers, PSC with over 10 years of experience designing commercial, retail and hospitality buildings. As an expert in CAD and Revit, Corey leads a team of in-house software programmers that develop custom tools and workflows that enhance construction document production. He holds a Bachelor of Science in industrial technology with a focus in computer aided drafting from Morehead State University.

Firm Overview

Offices:

Ft. Thomas, KY
Lexington, KY
Louisville, KY
Columbus, OH
New York, NY

Studios:

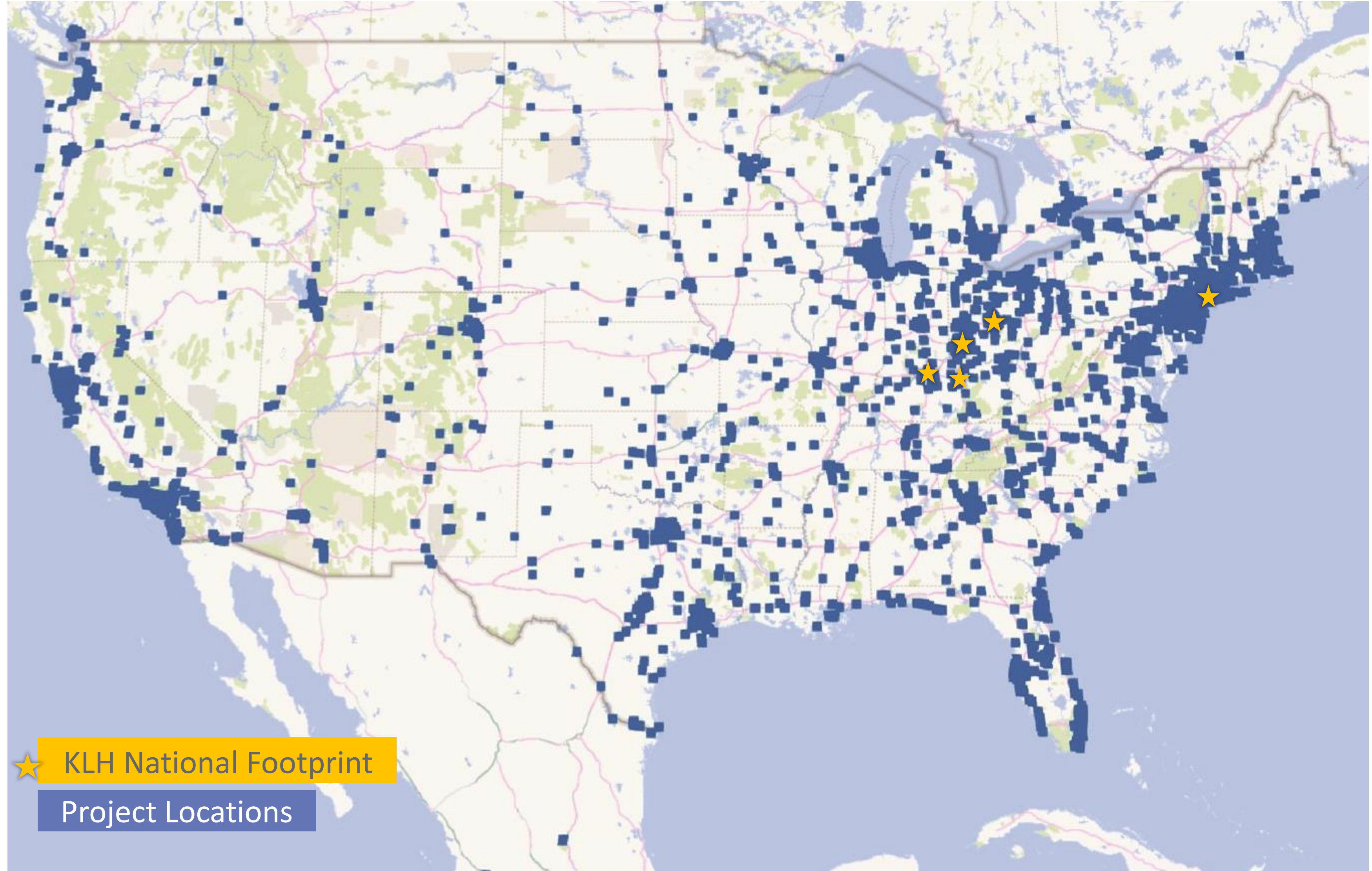
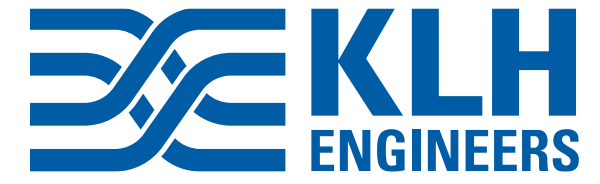
Healthcare
Education
Commercial
Civic
Retail

Services:

Mechanical
Electrical
Plumbing
Fire Protection
Technology

21,800 projects in 6 years

Licensed in all 50 states



KLH Engineers Background

KLH Engineers is a national MEP firm with a newly formed software department.

Implemented Revit in 2006; went 100% Revit on all projects in 2016.

Developed 1st custom tool in 2008; formed a software department in 2018.

- 2 Software Teams formed; Revit Development Team and a Enterprise Resource Planning (ERP) Development Team

A team of superusers was formed to test tools then roll out to the company. They work with the developers to ensure the software is working correctly.

As a newly formed department, our initial focus wasn't with testing. To refocus on QA and build and maintain internal relationships, the team had to develop an agile mentality.

Today, we will highlight these obstacles and share solutions, including a step-by-step process to setup unit testing in Visual Studio.



Who are you?

Architects/Engineers/BIM Manager?

Contractors?

Software Engineers? QA?

Learning Objectives

1. EXPLORE END-TO-END TESTS, INTEGRATION TESTS, COMPATIBILITY TESTS, AND UNIT TESTS TO CREATE BETTER REVIT ADD-INS
2. LEARN HOW TO CREATE A UNIT TEST PROJECT IN MICROSOFT VISUAL STUDIO
3. LEARN HOW TO RUN A UNIT TEST ON REVIT
4. LEARN HOW TO APPLY TESTING PHILOSOPHIES TO YOUR TEAM/COMPANY TO CREATE BETTER PRODUCTS AND INCREASE DEVELOPMENT SPEED

Problems to Overcome



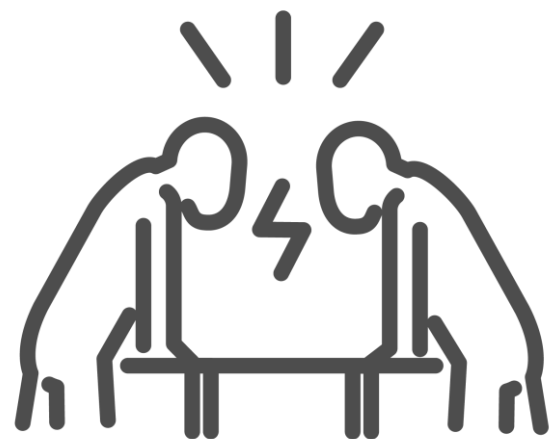
REVIT ADD-INS HAVE BUGS UPON RELEASE

Bugs are present on releases. New features don't work, or new features do work, and old features are broken due to lack of testing. Superusers only testing on their project types, Revit version they used, and scenarios in which were familiar.



TESTING IS NOT CONSISTENT AND IS TAKING TOO LONG

The initial testing process was not consistent. "Hey, can you test this tool?" was the basis of testing and hoping that the "tester" tried to break the tool. Testing was also taking too long; feedback was not getting back to the developers.



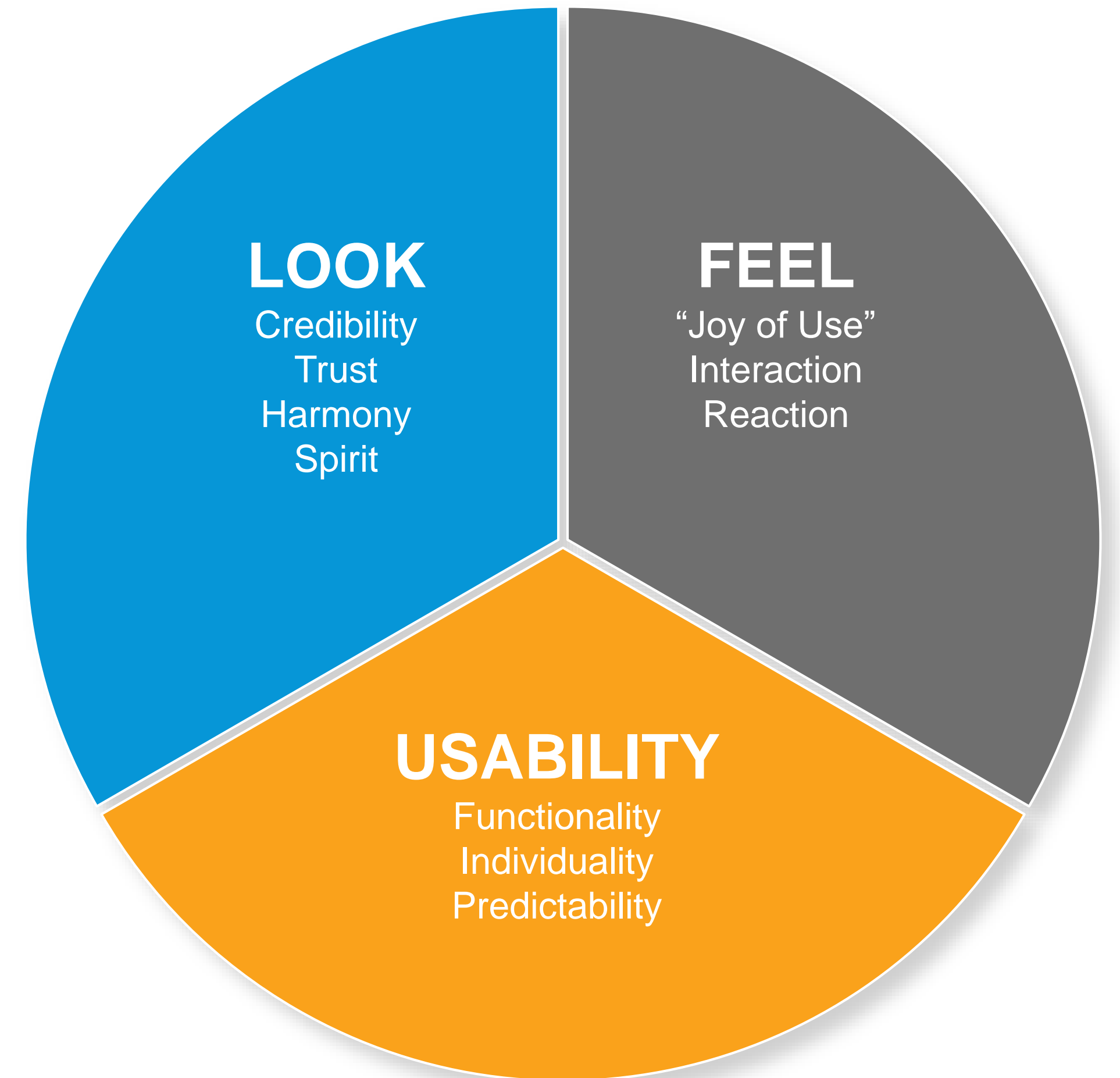
RELATIONSHIPS DETERIORATING BETWEEN SOFTWARE DEVELOPERS AND USERS

Due to bugs and FATAL ERRORS, relationships between developers were strained.

Testing Method 1: End-to-End Testing

SIMULATING THE USER EXPERIENCE

- E2E testing should focus on the user experience.
 - Does the software behave as expected?
 - What reaction do you have when you use the software?
 - How does the software interact with other applications that was developed by another team?



User Experience

Testing Method 1: End-to-End Testing

END TO END TESTING EXAMPLE

- Feature was requested to have the ability to select an exterior wall in Revit and get the correct square footage of the wall in the correct orientation.
 - **Step 1 – Review the User Stories for the Features**

The screenshot shows a Jira issue page for '5992 Calculate Wall Area within Revit - Example'. The issue is assigned to Patrick Fernbach and is in the 'Resolved' state. The reason for resolution is 'Stories complete'. The issue is associated with the 'R&D' team and the 'KLH\KLH Developers' area. The iteration is 'KLH\Biweekly Sprints\Sprint 35'. The issue was updated 18 minutes ago. Below the issue details, there is a 'Links' section with a table of related user stories.

Link ↑	State	Latest Update	Comment
6014 As a user, I need the tool to open up a view with all exteri...	● New	Updated 17 minutes ago	
6601 As a user, I need the wall square footage output populate...	● New	Updated 16 minutes ago	
6015 As a user, I need to set an exposure direction to each exte...	● New	Updated 17 minutes ago	

Testing Method 1: End-to-End Testing

END TO END TESTING EXAMPLE

- Step 1 – Review the User Stories for the Features
- **Step 2 – Open the user interface**
- **Step 3 – Select the button to launch the application. New view should appear with outline of spaces.**

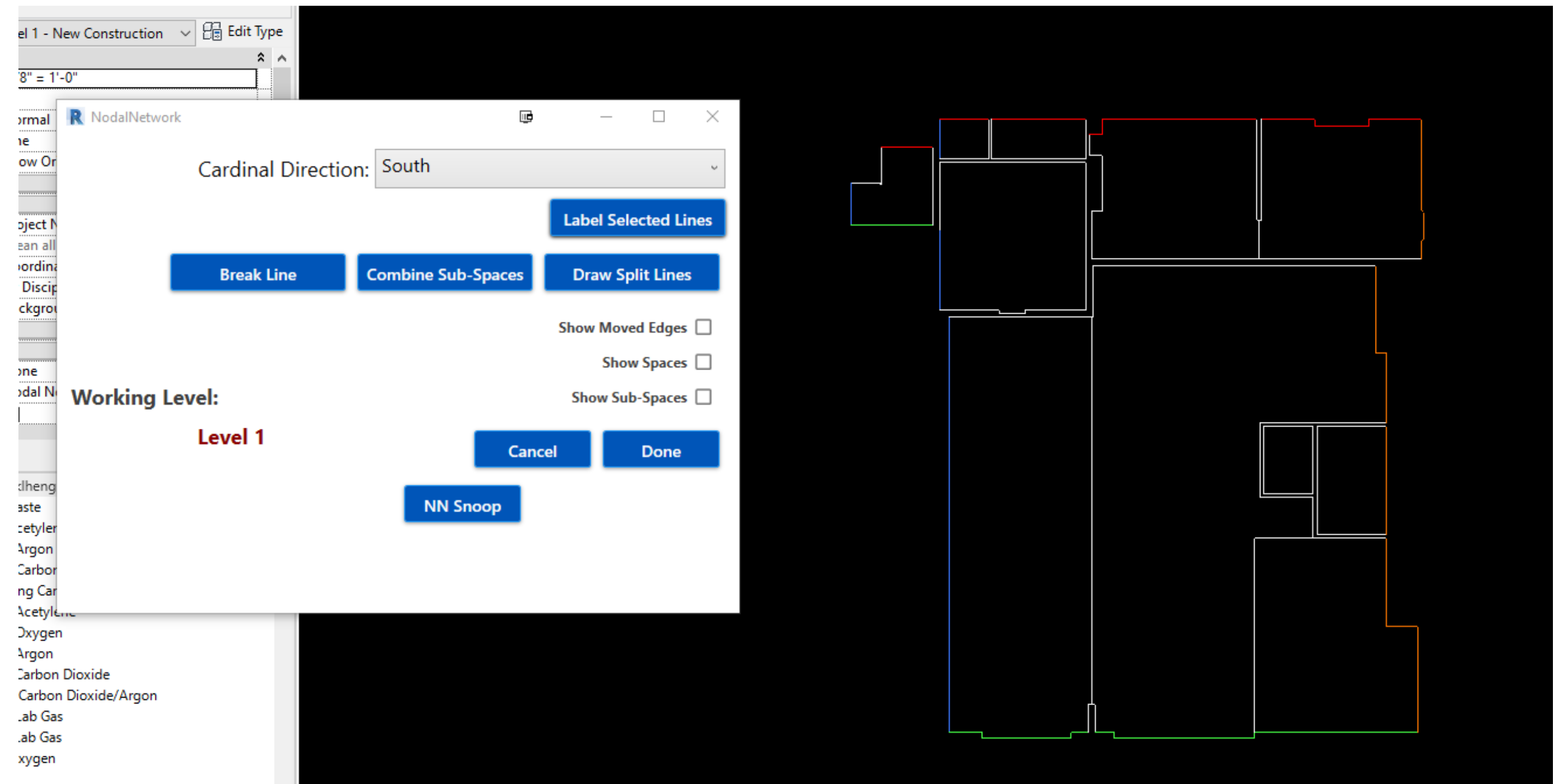
The screenshot displays the SyncLoadsV2 application interface. At the top, there are tabs for 'Project Information' and 'Space Loads'. Below the tabs, there are several control elements: 'All Inputs Visible' (checked), 'Roof/Slab' (selected), 'Exterior U Values', 'Exterior Walls', 'Ventilation', 'Misc. Loads', 'Show Only Included Loads' (unchecked), 'Outputs' (unchecked), 'Nodal Network' (highlighted in a red box), and 'Reset Nodal Network' (unchecked). There is also a search bar for spaces and a dropdown for 'System' set to 'IMC (2015)'. The main part of the interface is a table with columns: Include, NumberName, Level, System, SubspacesArea, Occupiable Area, Roof Area, Skylight Area, Slab Area, Slab Perimeter, North Wall, North Window, East Wall, East Window, South Wall, South Window, and W. The table contains 20 rows of room data. To the right of the table is a 'Space Quality Control' panel with the following information: Spaces... Not Zoned: 0, Not Given an Occupancy Class: 0, With Zero WPSF: 13, Roof Areas Not Matching Space Area: 12, Slab Areas Not Matching Space Area: 12, Total Electrical Wattage: 0, Model is CAD Converted: False. At the bottom right, there are buttons for 'View Log', 'Sync Loads', 'Sync Selected Loads', and a checkbox for 'Delete Unused Rooms'.

Include	NumberName	Level	System	SubspacesArea	Occupiable Area	Roof Area	Skylight Area	Slab Area	Slab Perimeter	North Wall	North Window	East Wall	East Window	South Wall	South Window	W
<input checked="" type="checkbox"/>	32 VISITORS LOCKER ROOM B	Level 1	1	0	345	345	365	0	365	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	17 WOMENS	Level 1	2	0	705	705	699	0	699	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	23 UNISEX	Level 1	3	0	75	75	75	0	75	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	22 MENS	Level 1	4	0	335	335	332	0	332	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	26 HOME LOCKER ROOM	Level 1	5	0	746	746	743	0	743	0	0	0	0	0	0	0
<input type="checkbox"/>	13 Room	Level 1	0	0	171	171	170	0	170	0	0	0	0	0	0	0
<input type="checkbox"/>	14 Room	Level 1	0	0	557	557	555	0	555	0	0	0	0	0	0	0
<input type="checkbox"/>	15 Room	Level 1	0	0	78	78	78	0	78	0	0	0	0	0	0	0
<input type="checkbox"/>	16 Room	Level 1	0	0	149	149	148	0	148	0	0	0	0	0	0	0
<input type="checkbox"/>	24 CHASE	Level 1	0	0	112	112	111	0	111	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	31 VISITORS LOCKER ROOM A	Level 1	6	0	389	389	365	0	365	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	34 OFFICIALS	Level 1	7	0	193	193	192	0	192	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	25 TRAINING/ MEETING	Level 1	8	0	272	272	271	0	271	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	18 STORAGE	Level 1	9	0	269	269	268	0	268	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	21 MECHANICAL	Level 1	10	0	162	162	161	0	161	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	28 VISITOR RR (WOMENS)	Level 1	11	0	103	103	102	0	102	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	33 HOME RR (MEN)	Level 1	12	0	103	103	102	0	102	0	0	0	0	0	0	0
<input checked="" type="checkbox"/>	35 OFFICIALS RR	Level 1	13	0	64	64	63	0	63	0	0	0	0	0	0	0

Testing Method 1: End-to-End Testing

END TO END TESTING EXAMPLE

- Step 1 – Review the User Stories for the Features
- Step 2 – Open the user interface
- Step 3 – Select the button to launch the application. New view should appear with outline of spaces.
- **Step 4 – Highlight the wall and set a cardinal direction. The tool should alert the user that there has been a change.**



Testing Method 1: End-to-End Testing

END TO END TESTING EXAMPLE

- Step 1 – Review the User Stories for the Features
- Step 2 – Open the user interface
- Step 3 – Select the button to launch the application. New view should appear with outline of spaces.
- Step 4 – Highlight the wall and set a cardinal direction. The tool should alert the user that there has been a change.
- **Step 5 – User finishes setting wall types and executes a Finish function to push calculated data to the correct location.**

The screenshot shows the SyncLoadsV2 software interface. The main window displays a table of space loads with columns for Inclusion, Name, Level, System, Subspaces, and various wall/window types. The 'System' column is highlighted in green. Below the table, there is a 'Space Quality Control' panel with the following information:

Space Quality Control
 Spaces...
 Not Zoned: 13
 Not Given an Occupancy Class: 0
 With Zero WPSF: 0
 Roof Areas Not Matching Space Area: 1
 Slab Areas Not Matching Space Area: 1
 Total Electrical Wattage: 800
 Model is CAD Converted: True

Buttons: View Log, Sync Loads, Sync Selected Loads, Delete Unused Rooms

Incl	Number	Name	Level	System	Subspaces	North Wall	North Window	East Wall	East Window	South Wall	South Window	West Wall	West Window	Deck Height	WINH (ft)	OVHT (in)	OVEX (in)	Partition Area
<input checked="" type="checkbox"/>	100	STAIR ENCLOSURE	Level 1	0	0	0	0	0	0	0	0	48	0	12	0	0	0	0
<input checked="" type="checkbox"/>	101	CLOSET	Level 1	0	0	55	0	0	0	0	0	46	0	12	0	0	0	0
<input checked="" type="checkbox"/>	102	EXIST. R.R.	Level 1	0	0	106	0	0	0	0	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	103	EXIST. KITCHEN	Level 1	0	0	0	0	0	0	0	0	169	0	12	0	0	0	0
<input checked="" type="checkbox"/>	104	OFFICE	Level 1	0	0	207	0	0	0	0	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	105	OFFICE	Level 1	0	0	198	0	163	0	0	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	106	PALETTE AREA	Level 1	0	0	0	0	0	0	172	0	472	0	12	0	0	0	0
<input checked="" type="checkbox"/>	107	ADDITION FLOOR	Level 1	0	0	0	0	228	0	196	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	107B	ADDITION FLOOR	Level 1	0	0	0	0	258	0	184	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	108	JAN	Level 1	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	109	RESTROOM	Level 1	0	0	0	0	105	0	0	0	0	0	12	0	0	0	0
<input checked="" type="checkbox"/>	B-1	Basement	Basement	0	0	509	0	0	0	597	0	399	0	22	0	0	0	0
<input checked="" type="checkbox"/>	B-2	Basement	Basement	0	0	494	0	292	0	497	0	0	0	22	0	0	0	0

Testing Method 1: End-to-End Testing

The screenshot displays a software interface for a building information model (BIM) project. The main window shows a floor plan of a building with several rooms and areas labeled, including:

- STAIR ENCLOSURE 100
- CLOSET 101
- EXIST. KITCHEN 103
- OFFICE 104
- OFFICE 106
- PALETTE AREA 108
- ADDITION FLOOR 107
- JAN 109
- RESTROOM 107

The interface includes a top toolbar with the following options:

- New Sync Loads
- Sync Equipment
- Match Elevation
- Analyze HVAC Piping System
- Fabrication Optimizer

The left sidebar contains a Properties panel for the selected 'Floor Plan' and a Project Browser. The Project Browser lists various plumbing and structural elements:

- (Plumbing) V - Vent
- (Plumbing) VV - Vacuum Vent
- (Plumbing) VW - Vacuum Waste
- (Plumbing) WA - Welding Acetylene
- (Plumbing) WAR - Welding Argon
- (Plumbing) WCO - Welding Carbon Dioxide
- (Plumbing) WCOAR - Welding Carbon Dioxide/Argon
- (Plumbing) WG1 - Welding Acetylene
- (Plumbing) WG2 - Welding Oxygen
- (Plumbing) WG3 - Welding Argon
- (Plumbing) WG4 - Welding Carbon Dioxide
- (Plumbing) WG5 - Welding Carbon Dioxide/Argon
- (Plumbing) WG6 - Welding Lab Gas
- (Plumbing) WLG - Welding Lab Gas
- (Plumbing) WO - Welding Oxygen
- Plumbing Fixtures
- Railings
- Ramps
- Roofs
- Site
- Stairs
- Structural Beam Systems
- Structural Foundations
- Walls
- Groups

The bottom status bar shows the current view scale as 1/8" = 1'-0".

Testing Method 1: End-to-End Testing

SUPERUSERS COLLABORATING WITH SOFTWARE TEAM

Superusers have the knowledge of how the tool should work as a whole.

- They understand the technology and the process
- They provide valuable feedback to Software Team.
 - Feedback comes in several forms:
 - Screen recording of tool in action and any accompanying notes.
 - Working session with the developer to discuss workflow issues.



Testing Method 2: Integration Testing

Integration testing involves taking the individual software modules and testing them as a group.

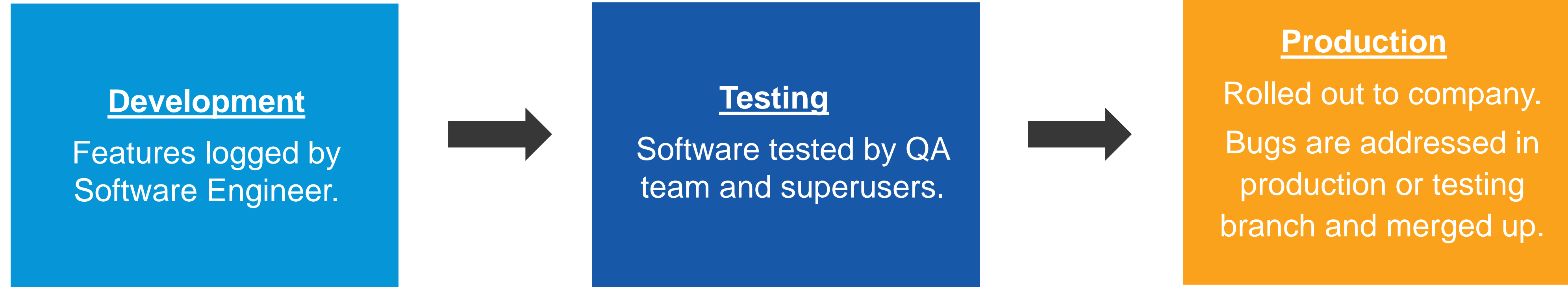


- KLH has a Revit development team along with a software team developing an ERP service.
- The Revit Team leverages the ERP teams' method and classes
- When the ERP team makes updates it's critical to perform Integration testing to ensure the tools are working correctly before being rolled out to the clients.

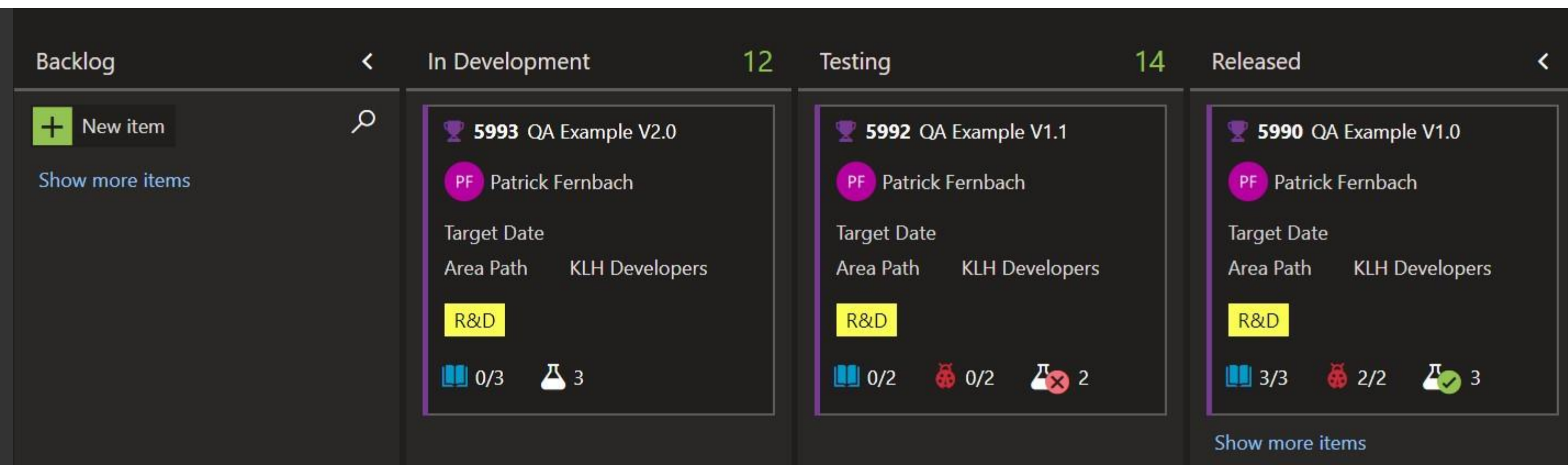
Testing Method 2: Integration Testing

Structured Merged System

KLH utilizes 3 Branches with DevOps to perform Integration Testing



DevOps Development Board



Testing Method 3: Compatibility Testing

- Compatibility Testing is a type of software testing to check whether your software is proficient enough to run in different environments.
- Your application should be checked against:
 - **Versions**
 - **Network**
 - Hardware
 - Operating Systems
 - Browsers
 - Mobile Devices



Testing Method 3: Compatibility Testing

Testing in all versions of Revit

- As a MEP firm the Revit model version is often dictated by the architect.
 - KLH maintains (4) version of Revit to ensure flexibility.
 - Software engineers develop in 2020. It is the responsibility of the engineers and developers to perform **Backward Compability Testing** to verify if the software will work with older versions of Revit.



Testing Method 3: Compatibility Testing

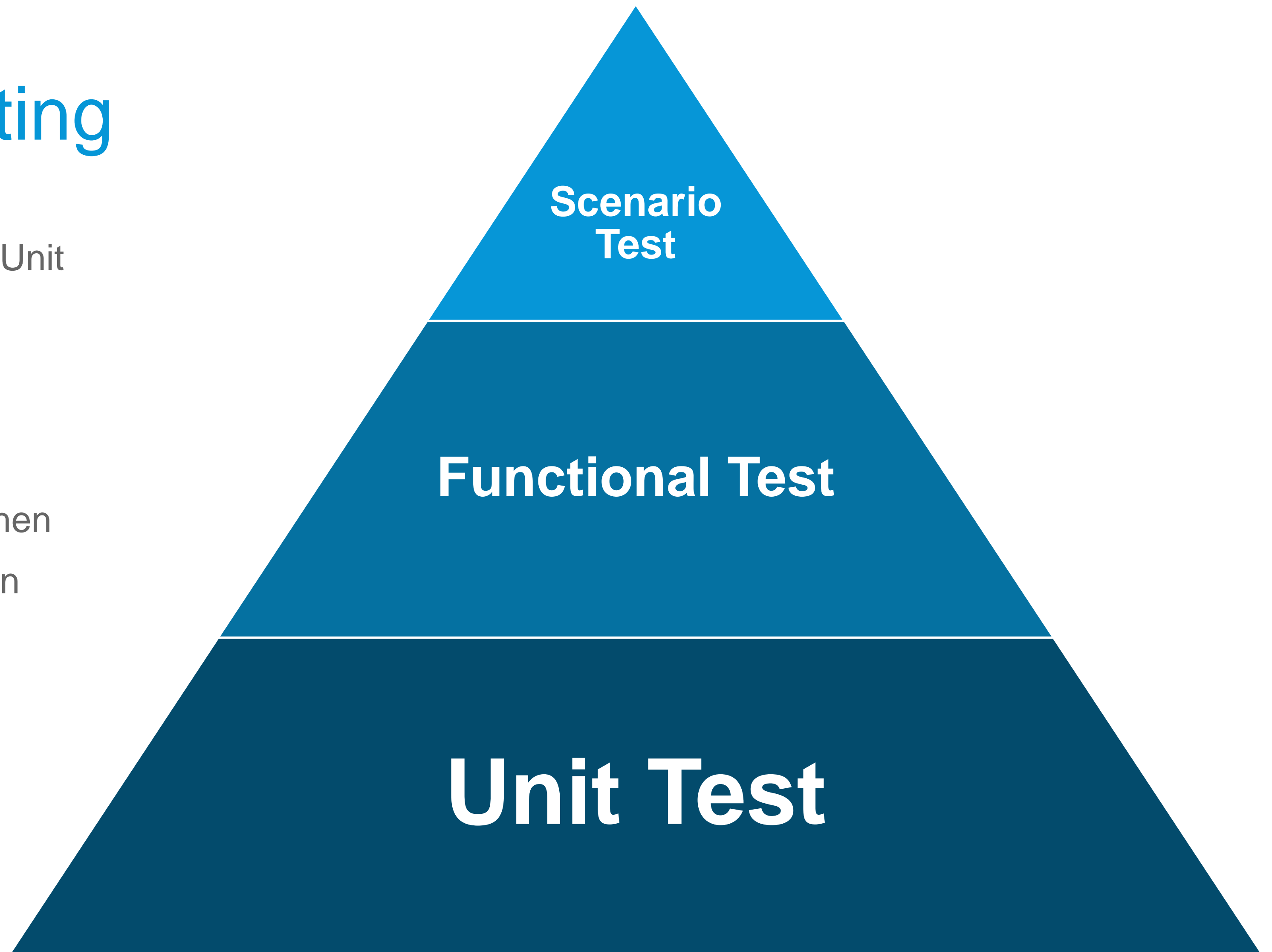
Connection Speed Tests

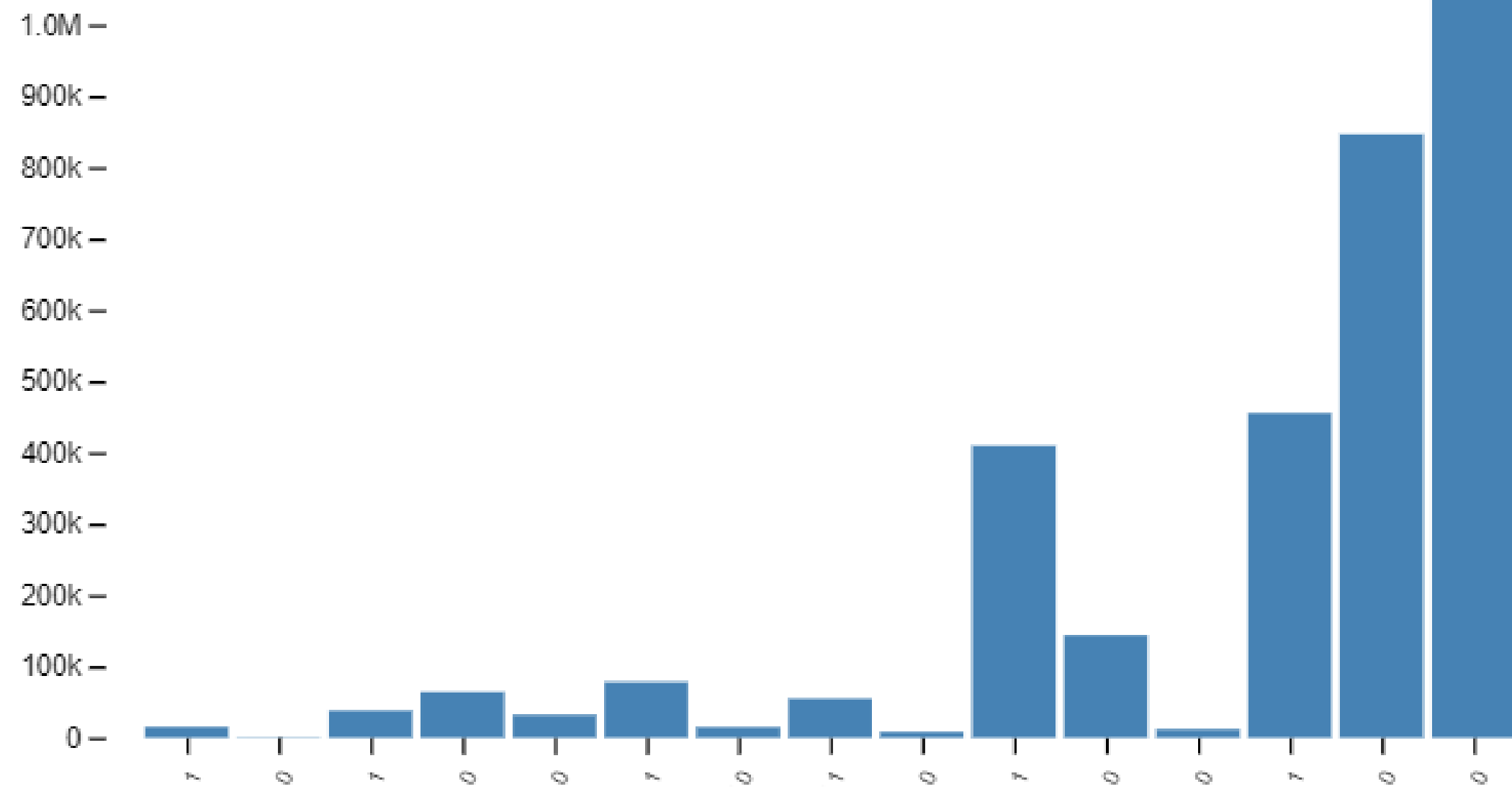
- KLH has (4) regional office all of which have different connection speeds.
 - Software Engineers perform tests that throttle connection speeds to simulate the user experience from the regional offices.
 - Software engineers explore optimization solution if load times are not acceptable.



Testing Method 4: *Automated* Unit Testing

- Unit testing is the foundation of testing. Unit tests are typically ran on individual units/functions of a large application.
- Unit tests are typically more effective when the person writing them is not the person who is or did write the code.





Downloads for 15 Latest Package Versions (Last 6 weeks)

NUnit

NUnit is a unit-testing framework that is compatible with all .NET languages. NUnit is Open Source software, with version 3 being under the MIT license.



Contributors of RTF

RTF

The Revit Testing Framework is a framework that should be referenced into the test project to use. It also has an executable file that facilitates running all the written tests.

Download here: <https://github.com/DynamoDS/RevitTestFramework>

KLH'S BATCH FILE

```
xcopy "...File Path to addin file to copy..." /D /Y /K /R /H /C /F  
xcopy "... File Path to test model to copy..." /D /Y /K /R /H /C /F
```

```
...\RevitTestFrameworkConsole.exe --dir . -a ...\KLH.Revit.Testing.dll -r results.xml -revit:"C:\Program Files\Autodesk\Revit  
2019\Revit.exe" --continuous --groupByModel --clean
```

```
del ./*.log  
del KLH_Ribbon.addin  
del KLH2019TestModel.rvt
```

Visual Studio & Batch File

Visual studio is a standard IDE from Microsoft. The batch file allows KLH to have a standard batch file that can be copied to multiple developers to run the same tests without needing to know the syntax. **NUnit and RTF need to be marked as a reference to the test project.**

Revit Test File

Create a Revit test model to run the tests. It should be setup the same way that a user would set it up, or as close as possible. KLH uses our standard setup to ensure we are running test in the environment of most users.

Console Version Rules

If the developer types "RevitTestFrameworkConsole -h" in the command line, the below options will be populated:

- `--dir=[VALUE]` The full path to the working directory. The working directory is the directory in which RTF will generate the journal and the addin to Run Revit. Revit's run-by-journal capability requires that all addins which need to be loaded are in the same directory as the journal file. So, if you're testing other addins on top of Revit using RTF, you'll need to put those addins in whatever directory you specify as the working directory.
- `-a, --assembly=[VALUE]` The full path to the assembly containing your tests.
- `-r, --results=[VALUE]` This is the full path to an .xml file that will contain the results.
- `-f, --fixture=[VALUE]` The full name (with namespace) of a test fixture to run. If no fixture, no category and no test names are specified, RTF will run all tests in the assembly.(OPTIONAL)
- `-t, --testName=[VALUE]` The name of a test to run. If no fixture, no category and no test names are specified, RTF will run all tests in the assembly. (OPTIONAL)
- `--category=[VALUE]` The name of a test category to run. If no fixture, no category and no test names are specified, RTF will run all tests in the assembly. (OPTIONAL)
- `--exclude=[VALUE]` The name of a test category to exclude. This has a higher priority than other settings. If a specified category is set here, any test cases that belongs to that category will not be run. (OPTIONAL)
- `-c, --concatenate` Concatenate the results from this run of RTF with an existing results file if one exists at the path specified. The default behavior is to replace the existing results file. (OPTIONAL)
- `--revit=[VALUE]` The Revit executable to be used for testing. If no executable is specified, RTF will use the first version of Revit that is found on the machine using the RevitAddinUtility. (OPTIONAL)
- `--copyAddins` Specify whether to copy the addins from the Revit folder to the current working directory. Copying the addins from the Revit folder will cause the test process to simulate the typical setup on your machine. (OPTIONAL)
- `--dry` Conduct a dry run. (OPTIONAL)
- `-x, --clean` Cleanup journal files after test completion. (OPTIONAL)
- `--continuous` Run all selected tests in one Revit session. (OPTIONAL)
- `--groupByModel` Run tests with same model without reopening the model for faster execution, requires --continuous. (OPTIONAL)
- `--time` The time, in milliseconds, after which RTF will close the testing process automatically. (OPTIONAL)
- `-d, --debug` Should RTF attempt to attach to a debugger?. (OPTIONAL)
- `-h, --help` Show this message and exit. (OPTIONAL)

What project??

The screenshot shows the 'Application' properties page for a project named 'KLH.Revit.Testing'. The left sidebar contains a list of settings categories: Application, Build, Build Events, Debug, Resources, Services, Settings, Reference Paths, Signing, and Code Analysis. The main area is divided into several sections:

- Configuration:** N/A (dropdown)
- Platform:** N/A (dropdown)
- Assembly name:** KLH.Revit.Testing (text box)
- Default namespace:** KLH.Revit.Testing (text box)
- Target framework:** .NET Framework 4.8 (dropdown)
- Output type:** Class Library (dropdown)
- Auto-generate binding redirects
- Startup object:** (Not set) (dropdown)
- Assembly Information...** (button)
- Resources** section:
 - Specify how application resources will be managed:
 - Icon and manifest**
 - A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
 - Icon:** (Default Icon) (dropdown) **Browse...** (button)
 - Manifest:** Embed manifest with default settings (dropdown)
 - Resource file:** (text box) **Browse...** (button)

Write A Test!

[TestFixture]

0 references | Jacob Reiter, 20 days ago | 2 authors, 3 changes | 2 reviews

```
public class IsPointOnUnboundLine_Tests {
```

```
    public XYZ _point;  
    public IEnumerable<XYZ> _points;  
    public int _precision;
```

[SetUp]

0 references | Jacob Reiter, 73 days ago | 2 authors, 2 changes | 1 review

```
public void SetupPoints()
```

```
{  
    _points = new XYZ[]  
    {  
        new XYZ(0,0,0),  
        new XYZ(0,0,1)  
    };  
  
    _precision = 7;  
}
```

[Test]

[TestModel(Variables.TestModel)]

0 references | Jacob Reiter, 20 days ago | 2 authors, 3 changes | 2 reviews

```
public void TestOnLine()
```

```
{  
    // arrange  
    _point = new XYZ(0, 0, 0.5);  
    bool result;  
  
    // act  
    result = IsPointOnUnBoundLine(_point, _points, _precision);  
  
    // assert  
  
    Assert.IsTrue(result);  
}
```

.xml Results File

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!--This file represents the results of running a test suite-->
```

```
<test-results name="C:\Users\KLH\Revit_Testing\obj\KLH.Revit_Testing.dll" total="38" failures="0" not-run="0" date="2019-11-04" time="09:14:05" errors="0" inconclusive="0" ignored="0" skipped="0" invalid="0">
```

```
<test-suite name="DynamoTestFrameworkTests" description="Unit tests in Revit." time="13.0229419" asserts="0" type="TestFixture" result="Success" executed="True">
```

```
<results>
```

```
<test-suite name="BoundingBoxOfPoints_Tests" description="Unit tests in Revit." time="0.0285923" asserts="0" type="TestFixture" result="Success" executed="True">
```

```
<results>
```

```
<test-case name="TestCorrectlyRounded" success="True" time="0.0285923" executed="True" asserts="0" result="Success" />
```

```
</results>
```

```
</test-suite>
```

```
<test-suite name="BoundingBoxXYZContains_Tests" description="Unit tests in Revit." time="0.0026858" asserts="0" type="TestFixture" result="Success" executed="True">
```

```
<results>
```

```
<test-case name="CorrectlyIDPointIn" success="True" time="0.0007379" executed="True" asserts="0" result="Success" />
```

```
<test-case name="CorrectlyIDPointOnBox" success="True" time="0.0002673" executed="True" asserts="0" result="Success" />
```

```
<test-case name="CorrectlyIDPointOut" success="True" time="0.0016806" executed="True" asserts="0" result="Success" />
```

```
</results>
```

```
</results>
```

```
</test-suite>
```

```
</test-results>
```

.xml Results File

```
<?xml version="1.0" encoding="utf-8"?> -
```

```
<!--This file represents the results of running a test suite-->
```

```
<test-results name = "...KLH.Revit.Testing\obj\KLH.Revit.Testing.dll" total="38" failures="0" not-run="0" date="2019-11-04" time="09:14:05" errors="0" inconclusive="0" ignored="0" skipped="0" invalid="0">
```

This line gives a summary of all tests, 38 total test with zero fails and zero not run. It gives the date initialized and the total time ran. There were not any tests that had errors or were inconclusive, ignored, skipped or invalid.

.xml Results File

```
<test-suite name="DynamoTestFrameworkTests" description="Unit tests in Revit." time="13.0229419" asserts="0"
type="TestFixture" result="Success" executed="True">
  <results>
    <test-suite name="BoundingBoxOfPoints_Tests" description="Unit tests in Revit." time="0.0285923" asserts="0"
type="TestFixture" result="Success" executed="True">
      <results>
        <test-case name="TestCorrectlyRounded" success="True" time="0.0285923" executed="True" asserts="0"
result="Success" />
      </results>
    </test-suite>
```

The body of the xml file is broken up into all of the **test suites** with each **test case** under each test suite. Each test case shows the name, description, time, asserts, type, result, and if it was executed.

Automate Testing with Automated Builds

Task Group: Command Line

The screenshot displays the Azure DevOps web interface for configuring a task group. The breadcrumb navigation shows 'Task groups > KLH.Revit Test'. The task group is named 'KLH.Revit Test' (Version 1.*). Two tasks are listed: 'Command Line Script' (Command line) and 'Publish Test Results **/TEST-\$(Build.BuildId).xml' (Publish Test Results). The 'Command Line Script' task is selected, and its configuration is shown on the right. The task version is set to '2.*'. The display name is 'Command Line Script'. The script field contains the text 'BATCH FILE SYNTAX'. Under 'Control Options', 'Enabled' and 'Continue on error' are checked. The timeout is set to '0'. The 'Run this task' dropdown is set to 'Only when all previous tasks have succeeded'. The 'Environment Variables' section is currently collapsed.

Automate Testing with Automated Builds

Task Group: Publish Test Results

The screenshot shows the Azure DevOps interface for configuring a task group. The left sidebar contains navigation options: Overview, Boards, Repos, Pipelines, Builds, Releases, Library, Task groups, Deployment groups, Build Tags, Test Plans, and Artifacts. The main area displays the 'Task groups > KLH.Revit Test' configuration. The selected task is 'Publish Test Results' (Version 1.*). The task configuration panel includes the following fields:

- Task version: 2.*
- Display name: Publish Test Results **/TEST-\$(Build.BuildId).xml
- Test result format: NUnit
- Test results files: **/TEST-\$(Build.BuildId).xml
- Search folder: \$(System.DefaultWorkingDirectory)
- Options: Merge test results, Fail if there are test failures
- Test run title: (empty field)
- Advanced and Control Options: (expandable sections)

The 'Copy to clipboard' dialog box displays the following YAML configuration for the task:

```
steps:  
- task: PublishTestResults@2  
  displayName: 'Publish Test Results **/TEST-$(Build.BuildId).xml'  
  inputs:  
    testResultsFormat: NUnit  
    testResultsFiles: '**/TEST-$(Build.BuildId).xml'
```

A 'Copy to clipboard' button is located at the bottom right of the dialog.

Applying Testing Philosophies

ONE TEAM

It is challenging to send testing to multiple users, especially when the users vary. One QA team is critical to manage the tests to ensure consistency and quality of testing.

ONE LOCATION

The QA members don't have to be in the same geographic proximity, but the logging and reporting of the tests needs to live in one spot.

ONE CULTURE

Testing needs to be embedded in the culture, especially when the same company is doing the development and the testing. There needs to be a healthy culture between the developers and the QA team, in addition to the relationship between the developers and the end users.

The Process

Feedback loop



KLH Training [Follow](#) – October 1 at 2:14 PM from Microsoft PowerApps and Flow

Feedback Form: Software Feedback Revit Electrical Tools J-R

Tool: Multi-Circuiter

Feedback Type: Bug

From: lheil@KLHENGRS.COM

Comments: Conn's 21570 - After placing all the electrical devices using the group inserter, I ran multi-circuiter using the connection parameter. There are groups that should have been circuited together, however they were all circuited as one circuit per connection.

[LIKE](#) [REPLY](#) [SHARE](#) ...

Seen by 52

[#Bug](#) [#Multi Circuiter](#) [#Tools](#) [#Electrical](#) [#Revit](#)



Patrick Fernbach – October 1 at 4:44 PM

Luke Heil, this is logged 5587. Thanks for the feedback.

cc: Luke Heil

[LIKE](#) [REPLY](#) [SHARE](#) ...



Jeff Mills – November 4 at 12:51 PM

This is fixed

[LIKE](#) [REPLY](#) [SHARE](#) ...

Luke Heil likes this



Write a reply



Additional Classes taught by KLH Engineers

Thursday 8:00 – 9:00

A Practical Use of Machine Learning in the AEC Industry



AUTOCAD TO REVIT LAYER NAME TRANSLATOR



AUTODESK®

Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.

