audio.dev
CONFERENCE
BRISTOL & ONLINE 2026

pluginval

How I Accidentally Built an Industry Standard(ish) Tool

Dave Rowland

audio.dev
CONFERENCE
BRISTOL & ONLINE 2026

pluginval

How I Accidentally Built an Industry Standard(ish) Tool

Dave Rowland

The Problem…

```
juce::StringArray getAllParameterText (juce::AudioProcessor& processor)
{
    juce::StringArray parameterText;

    for (int i = 0; i < processor.getNumParameters(); ++i)
        parameterText.add (processor.getParameterText (i));

    return parameterText;
}
```

- Can we make hosting more transparent?
- Could manufacturers test host behaviour without the host?
- Could users report issues directly?

```
juce::StringArray getAllParameterText (juce::AudioProcessor& processor)
{
    juce::StringArray parameterText;

    for (int i = 0; i < processor.getNumParameters(); ++i)
        parameterText.add (processor.getParameterText (i));

    return parameterText;
}
```

The Beginning



DAVID ROWLAND - TRACKTION CORPORATION

pluginval

YEAH, BUT WHY VALIDATE PLUGINS?

LANDSCAPE OF PLUGIN DEVELOPMENT

- 3 OSes (Windows, macOS, Linux)
- 4 Plugin formats (VST3, VST, AU, AAX)
- 2 architectures 32/64-bit
- 12 major hosts [Wikipedia]
  - plus 12 smaller hosts
- iOS/AUv3?




“288 Environments to Test”

PER RELEASE

- ~100 features/calls
- ~28,800 things to test
  - “Complexity factor”



LANDSCAPE OF PLUGIN DEVELOPMENT

THEN (288)

- 3 OSes (Windows, macOS, Linux)
- 4 Plugin formats (VST3, VST, AU, AAX)
- 2 architectures 32/64-bit
- 12 major hosts [Wikipedia]
  - plus 12 smaller hosts
- iOS/AUv3?

NOW (600?)

- 3 OSes (Windows, macOS, Linux*)
- 5 Plugin formats (VST3, AU, AAX, Clap, LV2)
- 2 architectures x64/ARM
- 20 major hosts [Wikipedia]
  - plus 20 smaller hosts?
- iOS/AUv3?

FEATURES

- Parameter values, names, labels, text readouts, defaults, suffixes, steps, categories
- Processing, single/double precision, bypassed processing, suspension, real-time/rendering
- Playhead information (tempo, key, position, playing, recording etc.)
- Channel layout, bus enablement
- Latency, state restoration, program lists
- MPE compatibility
- Listeners, extra functions such as automation state, editors, scale factors, resizing, Hi-DPI, parameter control highlight info, host MIDI controller presence…

PER RELEASE

THEN

- ~100 features/calls
- ~28,800 things to test
  - “Complexity factor”

400% increase

NOW

- ~200 features/calls
- ~120,000 things to test
  - “Complexity factor”


REAL QUESTIONS:
• How much should we trust it?
• Code quality?
• Security / data privacy?
• Maintainability?
• Who reviews the code?
• Replaces jobs?
• What's the real ROI?
STILL UNCLEAR:
• Best practices?
• Risks?
• Long term impa
OPPORTUNITIES:
• Faster development
• Higher productivity
• Better tooling
• New workflows
• Focus on higher value work
AI CODING LLM AGENT
PLAN
PROMPT
CODE
REVIEW
TEST
DEPLOY
REPEAT
BUILD
DEPLOY
ITERATE
Clean Code
System Design
AI Engineering

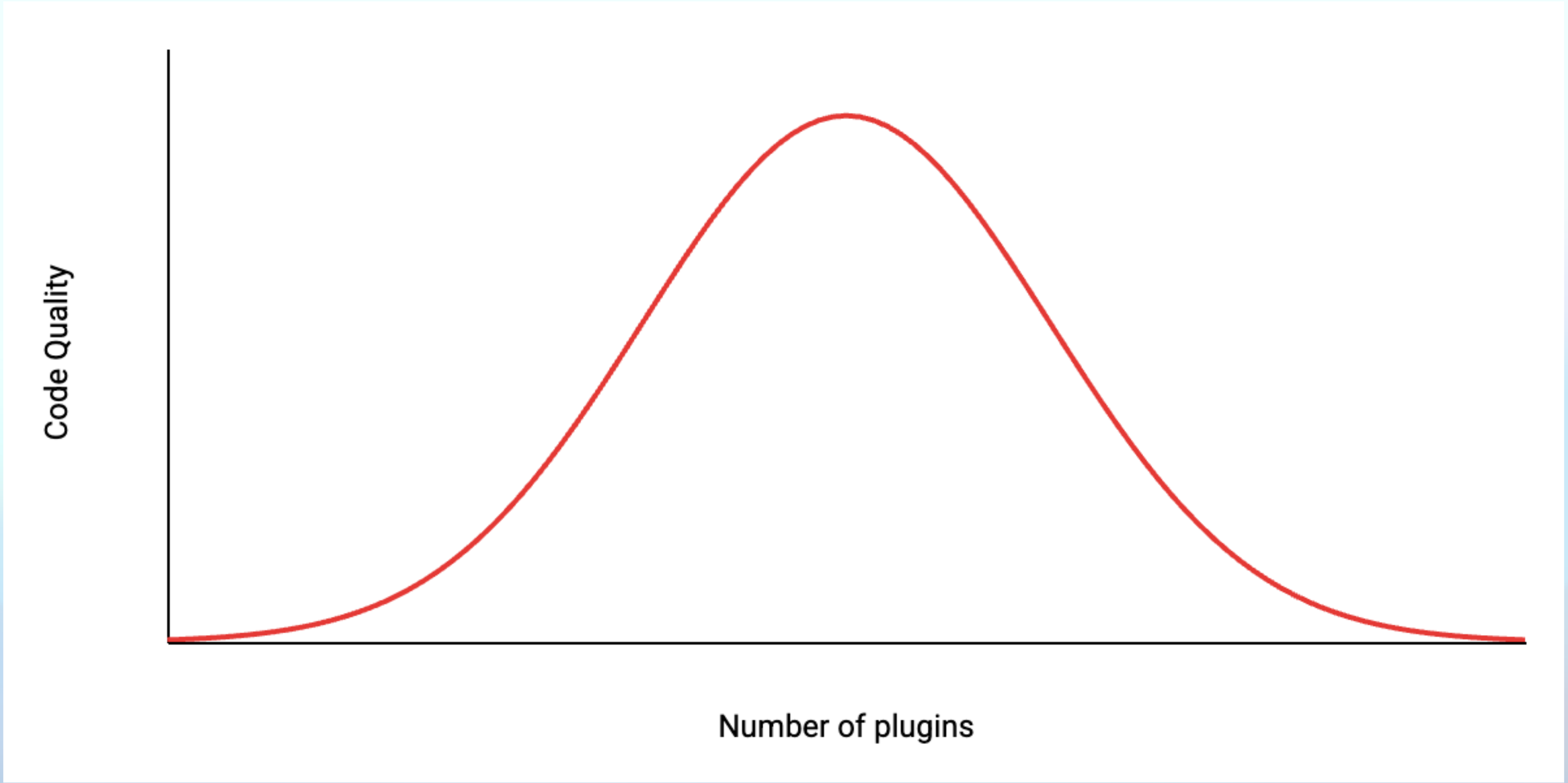
✋ Hands Up

- I read all the code changes before committing
- I commit changes manually
- I review every PR
  - Including the code changes
- I don’t read the code anymore

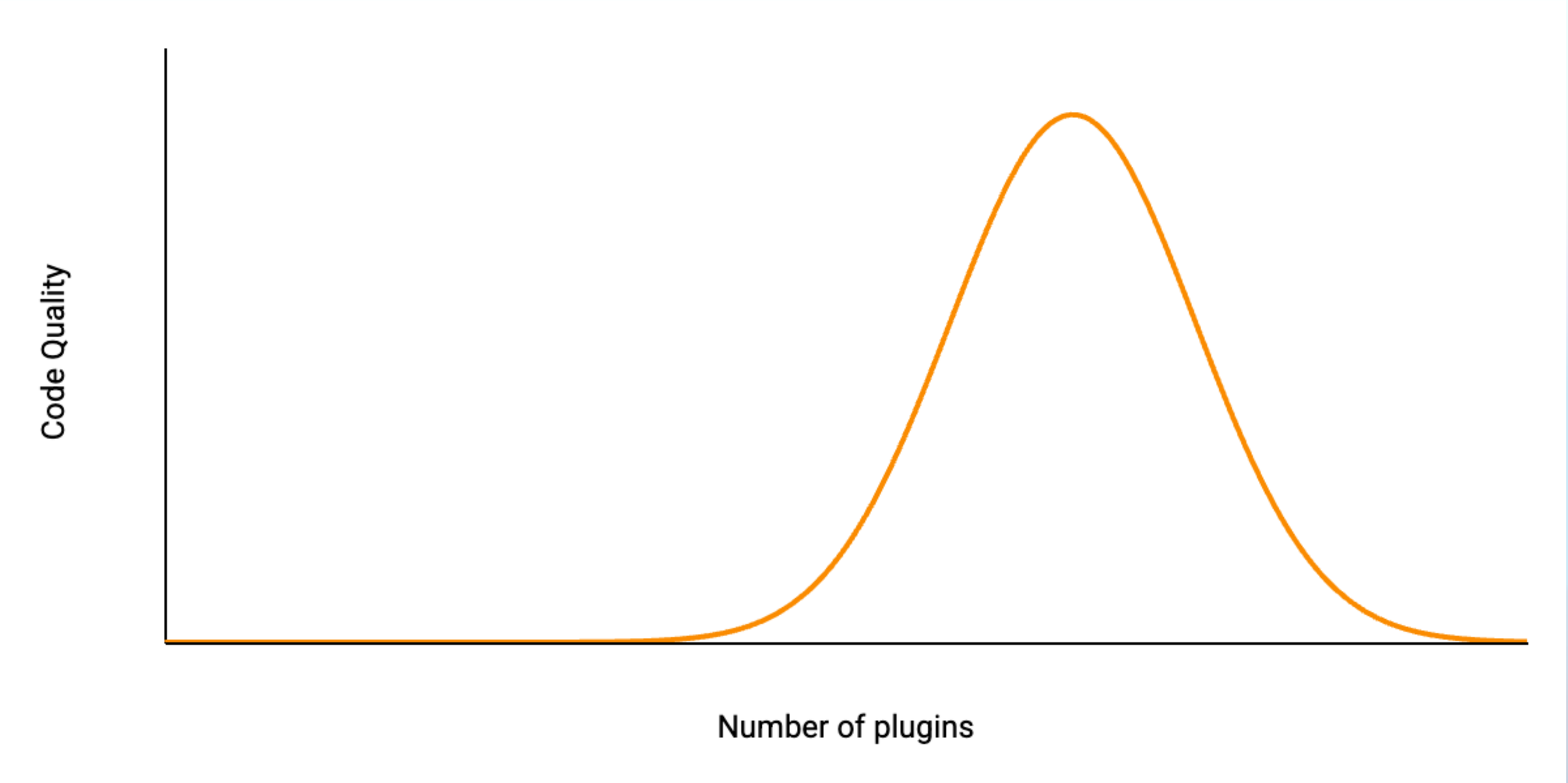
```
void processBlock (juce::AudioBuffer<float>& buffer, juce::MidiBuffer&) override
{
    dryBuffer.makeCopyOf (buffer);                          // keep dry signal for the mix
    buffer.applyGain (gainParam->load());

    for (int ch = 0; ch < buffer.getNumChannels(); ++ch)
        buffer.addFrom (ch, 0, dryBuffer, ch, 0, buffer.getNumSamples(), 1.0f - mix);

    if (auto* editor = dynamic_cast<GainEditor*> (getActiveEditor()))
        editor->meter.setLevel (buffer.getRMSLevel (0, 0, buffer.getNumSamples()));
}
```

- `makeCopyOf` allocates on the audio thread
- `getActiveEditor()` from the audio thread
- `meter.setLevel` probably calls `repaint()`
- `getRMSLevel (0, ...)` assumes channel 0 exists
- `mix` is read without being atomic

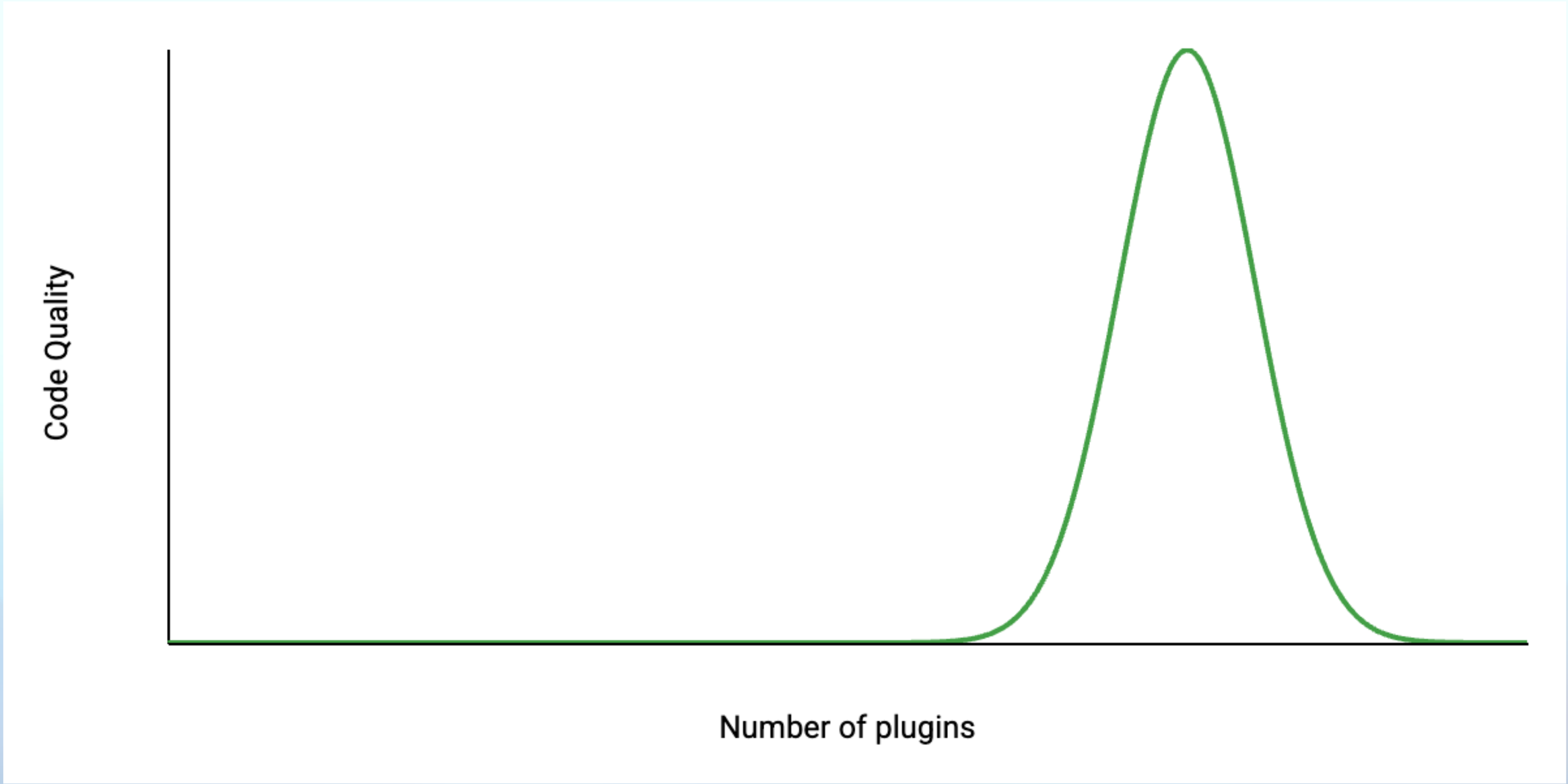
People are Bad at Writing Code



AI is Good at Writing Code



AI is Good at Writing Code



Plugin correctness lives in the host *contract*

AI is Getting Better at Writing Code

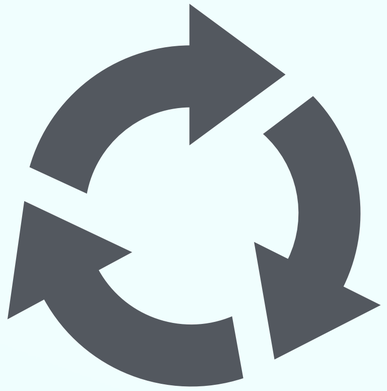
- Wait
- Improve the AI harness
- Train ourselves to use AI better
- Give AI better tools

So Why pluginval?

- It still does the job it was designed to do
  - Simulate and stress test plugin hosting and API coverage
- It's small and easy to use (by AI)
  - Well documented, simple CLI
- Well suited to CI
- It's free and open source
  - AI can *see* what pluginval was doing to cause an issue

Write

Fix



Build

Test

Generate

Fix



Build

Validate

…

`pluginval`

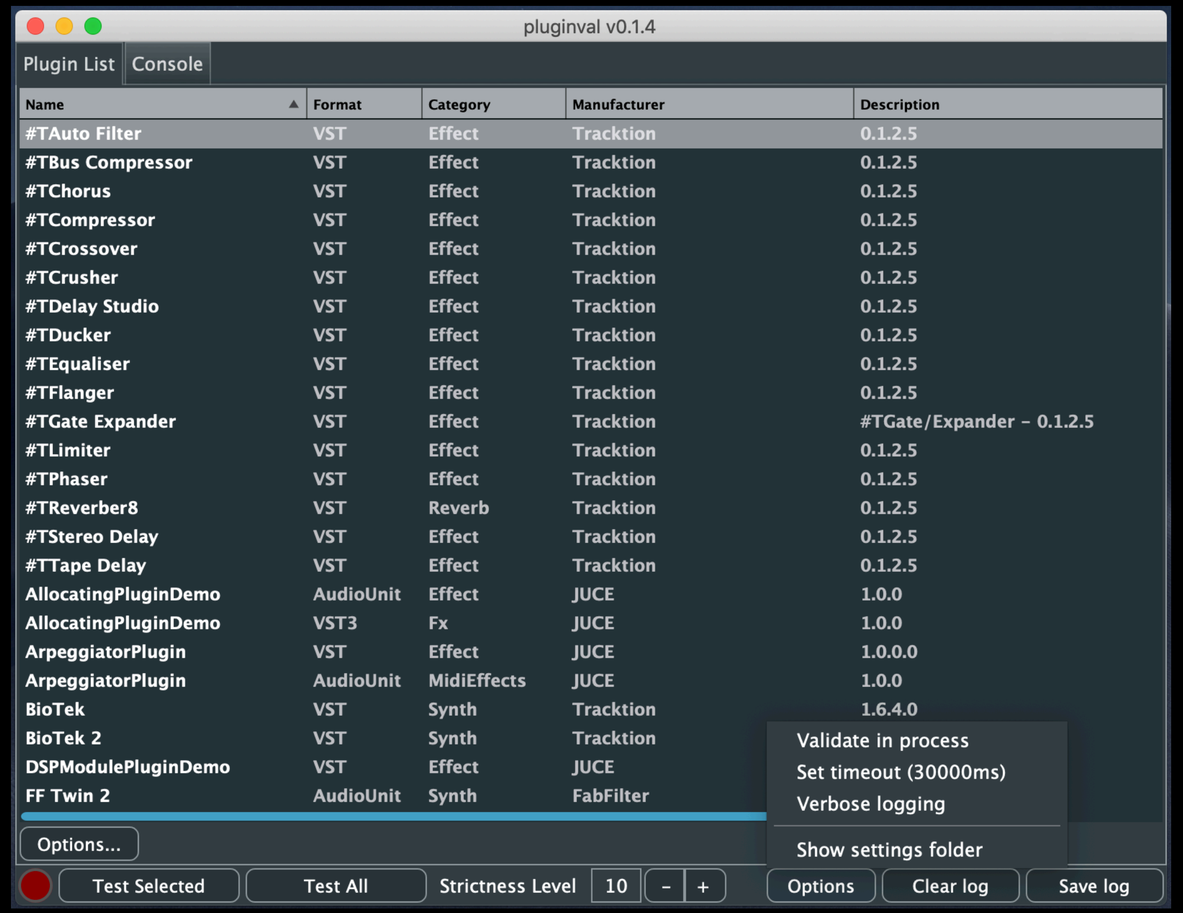
…

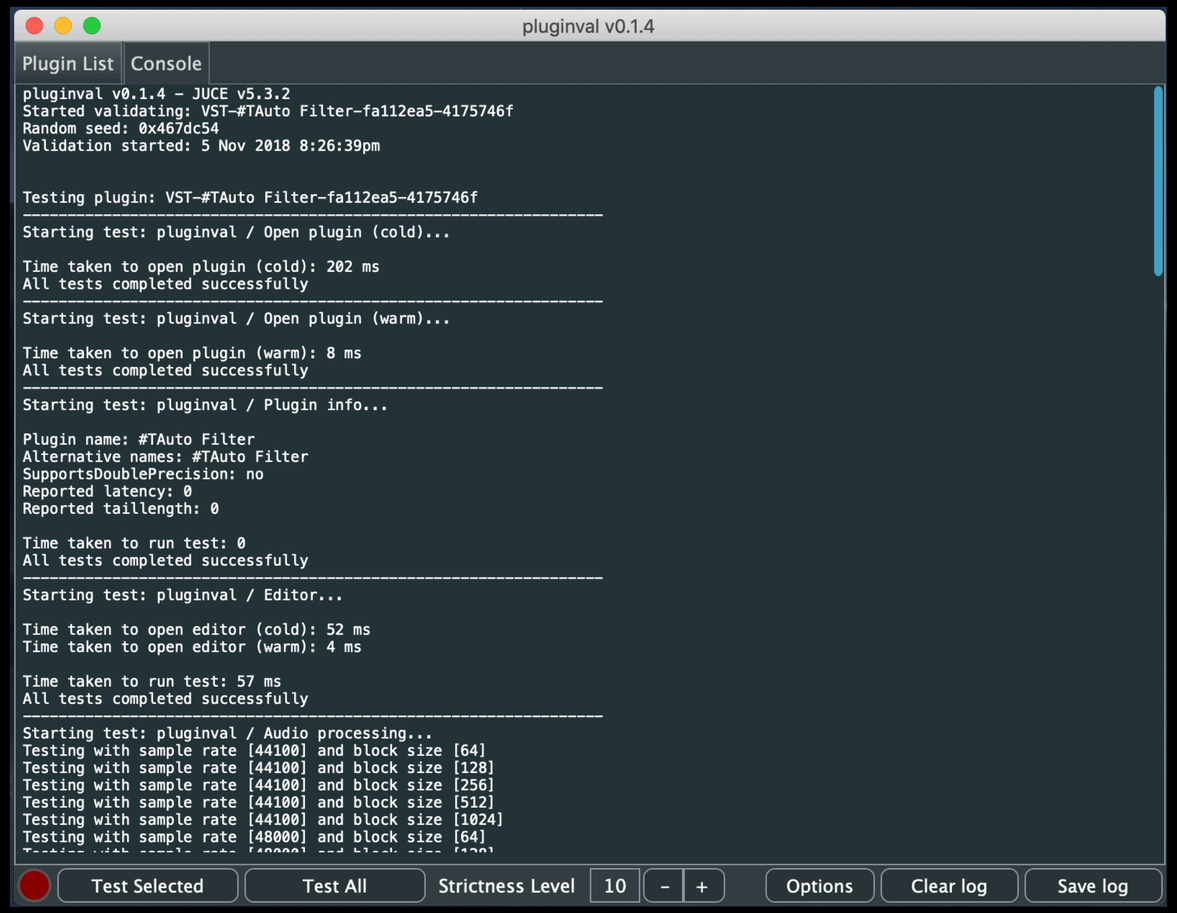
Towards a Solution?

“But it works in…”

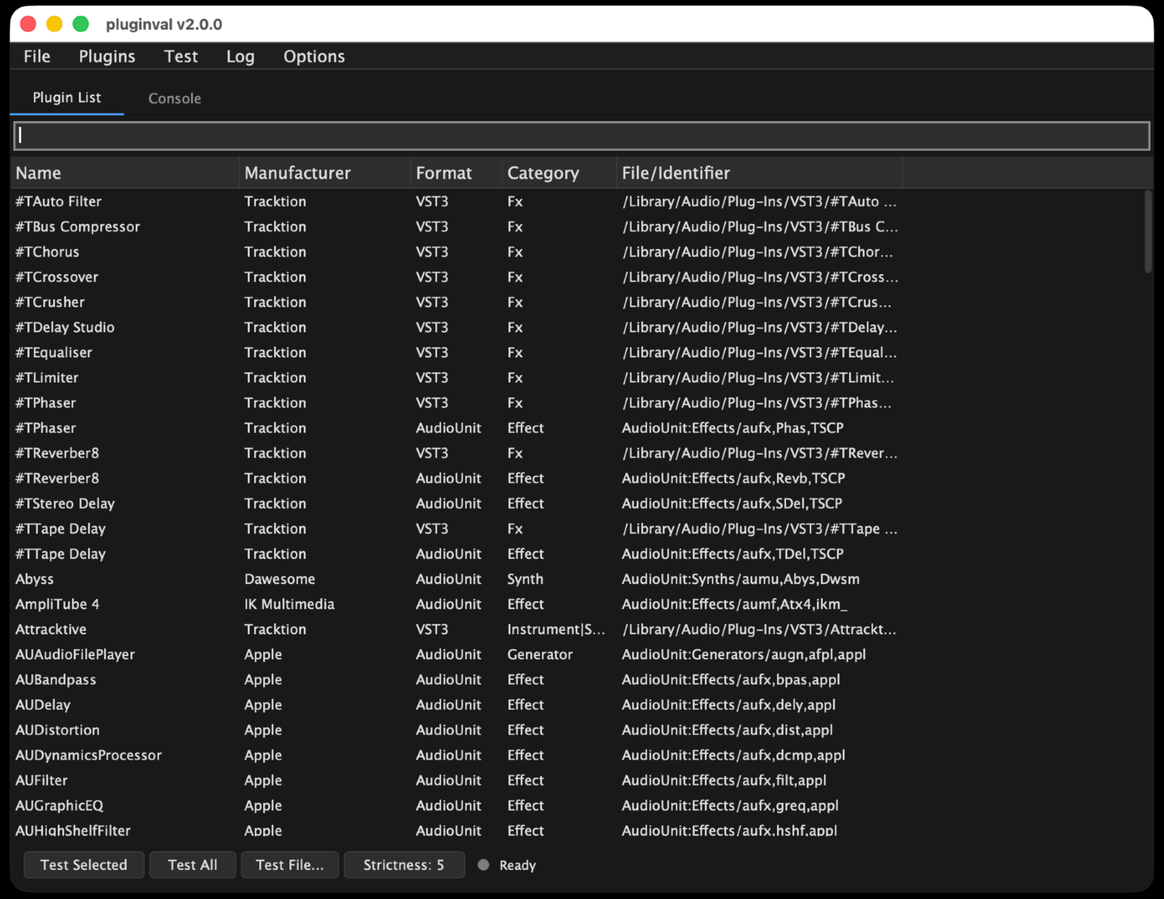
Every user

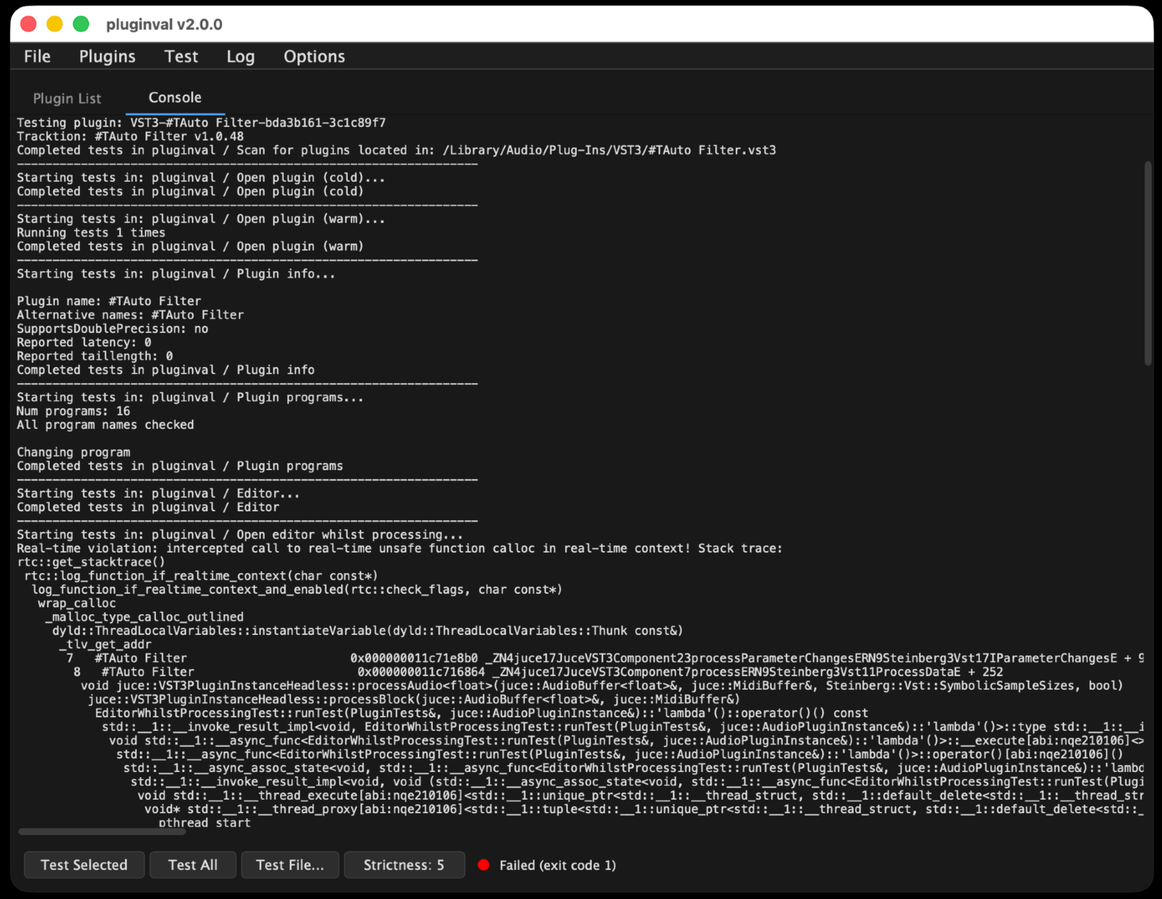
v0.1


pluginval v0.1.4
Plugin List
Console
Name
Format
Category
Manufacturer
Description
#TAuto Filter VST Effect Tracktion 0.1.2.5
#TBus Compressor VST Effect Tracktion 0.1.2.5
#TChorus VST Effect Tracktion 0.1.2.5
#TCompressor VST Effect Tracktion 0.1.2.5
#TCrossover VST Effect Tracktion 0.1.2.5
#TCrusher VST Effect Tracktion 0.1.2.5
#TDelay Studio VST Effect Tracktion 0.1.2.5
#TDucker VST Effect Tracktion 0.1.2.5
#TEqualiser VST Effect Tracktion 0.1.2.5
#TFlanger VST Effect Tracktion 0.1.2.5
#TGate Expander VST Effect Tracktion #TGate/Expander - 0.1.2.5
#TLimiter VST Effect Tracktion 0.1.2.5
#TPhaser VST Effect Tracktion 0.1.2.5
#TReverber8 VST Reverb Tracktion 0.1.2.5
#TStereo Delay VST Effect Tracktion 0.1.2.5
#TTape Delay VST Effect Tracktion 0.1.2.5
AllocatingPluginDemo AudioUnit Effect JUCE 1.0.0
AllocatingPluginDemo VST3 Fx JUCE 1.0.0
ArpeggiatorPlugin VST Effect JUCE 1.0.0.0
ArpeggiatorPlugin AudioUnit MidiEffects JUCE 1.0.0
BioTek VST Synth Tracktion 1.6.4.0
BioTek 2 VST Synth Tracktion
DSPModulePluginDemo VST Effect JUCE
FF Twin 2 AudioUnit Synth FabFilter
Validate in process
Set timeout (30000ms)
Verbose logging
Show settings folder
Options...
Test Selected
Test All
Strictness Level
10
-
+
Options
Clear log
Save log


pluginval v0.1.4
Plugin List
Console
pluginval v0.1.4 - JUCE v5.3.2
Started validating: VST-#TAuto Filter-fa112ea5-4175746f
Random seed: 0x467dc54
Validation started: 5 Nov 2018 8:26:39pm
Testing plugin: VST-#TAuto Filter-fa112ea5-4175746f
Starting test: pluginval / Open plugin (cold)...
Time taken to open plugin (cold): 202 ms
All tests completed successfully
Starting test: pluginval / Open plugin (warm)...
Time taken to open plugin (warm): 8 ms
All tests completed successfully
Starting test: pluginval / Plugin info...
Plugin name: #TAuto Filter
Alternative names: #TAuto Filter
SupportsDoublePrecision: no
Reported latency: 0
Reported taillength: 0
Time taken to run test: 0
All tests completed successfully
Starting test: pluginval / Editor...
Time taken to open editor (cold): 52 ms
Time taken to open editor (warm): 4 ms
Time taken to run test: 57 ms
All tests completed successfully
Starting test: pluginval / Audio processing...
Testing with sample rate [44100] and block size [64]
Testing with sample rate [44100] and block size [128]
Testing with sample rate [44100] and block size [256]
Testing with sample rate [44100] and block size [512]
Testing with sample rate [44100] and block size [1024]
Testing with sample rate [48000] and block size [64]
Test Selected
Test All
Strictness Level
10
-
+
Options
Clear log
Save log

v2.0


pluginval v2.0.0
File Plugins Test Log Options
Plugin List Console
Name Manufacturer Format Category File/Identifier
#TAuto Filter Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TAuto ...
#TBus Compressor Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TBus C...
#TChorus Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TChor...
#TCrossover Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TCross...
#TCrusher Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TCrus...
#TDelay Studio Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TDelay...
#TEqualiser Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TEqual...
#TLimiter Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TLimit...
#TPhaser Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TPhas...
#TPhaser Tracktion AudioUnit Effect AudioUnit:Effects/aufx,Phas,TSCP
#TReverber8 Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TRever...
#TReverber8 Tracktion AudioUnit Effect AudioUnit:Effects/aufx,Revb,TSCP
#TStereo Delay Tracktion AudioUnit Effect AudioUnit:Effects/aufx,SDel,TSCP
#TTape Delay Tracktion VST3 Fx /Library/Audio/Plug-Ins/VST3/#TTape ...
#TTape Delay Tracktion AudioUnit Effect AudioUnit:Effects/aufx,TDel,TSCP
Abyss Dawesome AudioUnit Synth AudioUnit:Synths/aumu,Abys,Dwsm
AmpliTube 4 IK Multimedia AudioUnit Effect AudioUnit:Effects/aumf,Atx4,ikm_
Attracktive Tracktion VST3 Instrument|S... /Library/Audio/Plug-Ins/VST3/Attrackt...
AUAudioFilePlayer Apple AudioUnit Generator AudioUnit:Generators/augn,afpl,appl
AUBandpass Apple AudioUnit Effect AudioUnit:Effects/aufx,bpas,appl
AUDelay Apple AudioUnit Effect AudioUnit:Effects/aufx,dely,appl
AUDistortion Apple AudioUnit Effect AudioUnit:Effects/aufx,dist,appl
AUDynamicsProcessor Apple AudioUnit Effect AudioUnit:Effects/aufx,dcmp,appl
AUFilter Apple AudioUnit Effect AudioUnit:Effects/aufx,filt,appl
AUGraphicEQ Apple AudioUnit Effect AudioUnit:Effects/aufx,greq,appl
AUHighShelfFilter Apple AudioUnit Effect AudioUnit:Effects/aufx,hshf,appl
Test Selected Test All Test File... Strictness: 5 Ready


pluginval v2.0.0
File Plugins Test Log Options
Plugin List Console
Testing plugin: VST3-#TAuto Filter-bda3b161-3c1c89f7
Tracktion: #TAuto Filter v1.0.48
Completed tests in pluginval / Scan for plugins located in: /Library/Audio/Plug-Ins/VST3/#TAuto Filter.vst3
Starting tests in: pluginval / Open plugin (cold)...
Completed tests in pluginval / Open plugin (cold)
Starting tests in: pluginval / Open plugin (warm)...
Running tests 1 times
Completed tests in pluginval / Open plugin (warm)
Starting tests in: pluginval / Plugin info...
Plugin name: #TAuto Filter
Alternative names: #TAuto Filter
SupportsDoublePrecision: no
Reported latency: 0
Reported taillength: 0
Completed tests in pluginval / Plugin info
Starting tests in: pluginval / Plugin programs...
Num programs: 16
All program names checked
Changing program
Completed tests in pluginval / Plugin programs
Starting tests in: pluginval / Editor...
Completed tests in pluginval / Editor
Starting tests in: pluginval / Open editor whilst processing...
Real-time violation: intercepted call to real-time unsafe function calloc in real-time context! Stack trace:
rtc::get_stacktrace()
rtc::log_function_if_realtime_context(char const*)
log_function_if_realtime_context_and_enabled(rtc::check_flags, char const*)
wrap_calloc
_malloc_type_calloc_outlined
dyld::ThreadLocalVariables::instantiateVariable(dyld::ThreadLocalVariables::Thunk const&)
_tlv_get_addr
7 #TAuto Filter 0x000000011c71e8b0 _ZN4juce17JuceVST3Component23processParameterChangesERN9Steinberg3Vst17IParameterChangesE + 9
8 #TAuto Filter 0x000000011c716864 _ZN4juce17JuceVST3Component7processERN9Steinberg3Vst11ProcessDataE + 252
void juce::VST3PluginInstanceHeadless::processAudio<float>(juce::AudioBuffer<float>&, juce::MidiBuffer&, Steinberg::Vst::SymbolicSampleSizes, bool)
juce::VST3PluginInstanceHeadless::processBlock(juce::AudioBuffer<float>&, juce::MidiBuffer&)
EditorWhilstProcessingTest::runTest(PluginTests&, juce::AudioPluginInstance&)::'lambda'()::operator()() const
std::__1::__invoke_result_impl<void, EditorWhilstProcessingTest::runTest(PluginTests&, juce::AudioPluginInstance&)::'lambda'()>::type std::__1::__i
void std::__1::__async_func<EditorWhilstProcessingTest::runTest(PluginTests&, juce::AudioPluginInstance&)::'lambda'()>::__execute[abi:nqe210106]<>
std::__1::__async_func<EditorWhilstProcessingTest::runTest(PluginTests&, juce::AudioPluginInstance&)::'lambda'()>::operator()[abi:nqe210106]()
std::__1::__async_assoc_state<void, std::__1::__async_func<EditorWhilstProcessingTest::runTest(PluginTests&, juce::AudioPluginInstance&)::'lambd
std::__1::__invoke_result_impl<void, void (std::__1::__async_assoc_state<void, std::__1::__async_func<EditorWhilstProcessingTest::runTest(Plugi
void std::__1::__thread_execute[abi:nqe210106]<std::__1::unique_ptr<std::__1::__thread_struct, std::__1::default_delete<std::__1::__thread_str
void* std::__1::__thread_proxy[abi:nqe210106]<std::__1::tuple<std::__1::unique_ptr<std::__1::__thread_struct, std::__1::default_delete<std::_
thread start
Test Selected Test All Test File... Strictness: 5 Failed (exit code 1)

Original Aims

- Ensure stability and compatibility between plugins and hosts
- Catch non-obvious bugs
- Be runnable by anyone
- Work across formats
- Be automatable

WHAT DOES IT TEST?

- Plugin info
- Parameters
- Editor
- Plugin state saving and restoration
- Audio processing - multiple block sizes and sample rates
- Audio processing - sub blocks
- Automation
- Multi-threaded state & value setting
- Bus/channel layout info
- Memory allocations on the audio thread (`operator new`)
- Parameter fuzzing

Architecture

- UI validates plugins in a separate process
- Runs tests reporting output and status
- Random but reproducible seeds
- Configurable tests grouped by “level” 1-10
  - 1-5 for conformance
  - 5-10 for stress testing and hardening

File Plugins Test Log Options

Plugin List Console

Test Selected Test All Test File... Strictness: 10 Ready

What Does a Test Look Like?

```
struct PluginStateTestRestoration : public PluginTest
{
    PluginStateTestRestoration()
        : PluginTest ("Plugin state restoration", 6)   // name, strictness level
    {
    }

    void runTest (PluginTests& ut, juce::AudioPluginInstance& instance) override
    {
        auto r = ut.getRandom();

        // Read state
        auto originalState = callGetStateInformationOnMessageThreadIfVST3 (instance);
        callSetStateInformationOnMessageThreadIfVST3 (instance, originalState);

        for (auto parameter : getNonBypassAutomatableParameters (instance))
        {
            const auto originalValue = parameter->getValue();
            parameter->setValue (getRandomParameterValue (r, *parameter, ut.getOptions().strictnessLevel));

            // Restore original state
            callSetStateInformationOnMessageThreadIfVST3 (instance, originalState);

            // Check parameter values return to original
            ut.expectWithinAbsoluteError (parameter->getValue(), originalValue, 0.1f,
                parameter->getName (1024) + " not restored on setStateInformation");
        }
    }
};

static PluginStateTestRestoration pluginStateTestRestoration;   // registers the test
```

Has it Met its Goals?

- Can we make hosting more transparent?
- Could manufacturers test host behaviour without the host?
- Could users report issues directly?
- Have community contributed?

What has Fallen Short

- Host-specific behaviour can't always be simulated
- Format coverage gaps
  - JUCE based hosting*
  - No Clap
- High maintenance load
- Flaky tests...

The Most Vexing Bug

State Restoration Test

1. Set a parameter value
2. Save the plugin state
3. Randomise the parameter values
4. Restore the state
5. Check the parameter value is the same from 1.

```cpp
juce::AudioParameterBool bypass { "bypass", "Bypass", false };
juce::AudioProcessorParameter& p = bypass;

p.setValue (0.37f);   // host: any normalised float is legal
p.getValue();         // [1] 0.37  - not a value a bool can have
bypass.get();         // false*

                      // getStateInformation() called
p.setValue (0.8f);    // host moves it: true
bypass.get();         // true

bypass = false;       // setStateInformation() restores "false"
p.getValue();         // 0.0   - same state, different value
```

[1] What should `getValue()` return?

- `0.37`? `0.0`?

[2] After a state store/load what `getValue()` return?

[3] What is “false”?

	0.0	0.37	0.5	0.8	1.0
cast	false	false	false	false?	true
!0	false	true	true	true	true
round	false	false	t/f	true	true

juce::AudioParameterChoice

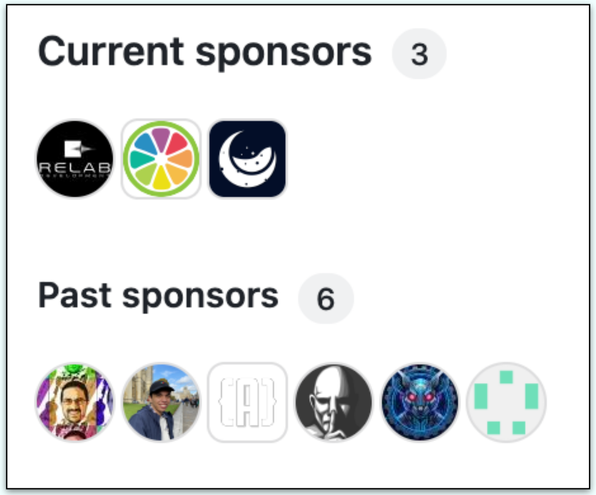
Normalised	JUCE round (v * 3)	VST3 min (3, int (v * 4))
0.20	1	0
0.45	1	1
0.55	2	2
0.80	2	3

- VST3:
  - `min (stepCount, normalized * (stepCount + 1))`
  - `normalizedParamToPlain/plainParamToNormalized`
  - `getParamNormalized/setNormalized` are raw values
- AU:
  - Problem doesn’t exist - no normalised domain
    - All parameters are plain and can be bool/integer
- JUCE:
  - Only uses the normalised domain for compatibility
  - For AU, caches normalised value, sends scaled value to plugin (state via plugin)
  - `round (v * stepCount)`

Have Community Contributed?

- 589 ⭐
- 79 ⑂
- 20 contributors
  - *Sudara*
  - *Jim Credland*
  - *Nick Dowell*

Sponsorship



- https://github.com/sponsors/drowaudio
- Monthly
  - $5 🥉
  - $10 🥈
  - $25 🥇
  - $50 💎
- One-time
  - $50 🥉
  - $100 🥈
  - $500 🥇
  - $1000 💎

Diamond Sponsors 💎

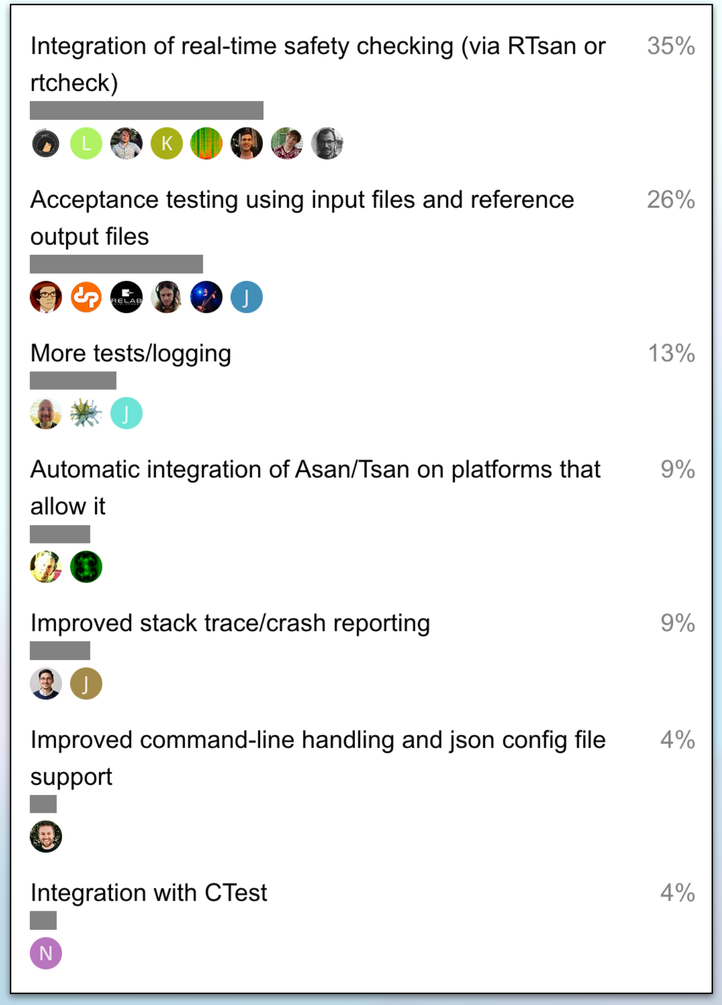
- The Audio Programmer - Real-time safety checking



- Moonbase - Modernising for v2



- Anonymous - Acceptance testing



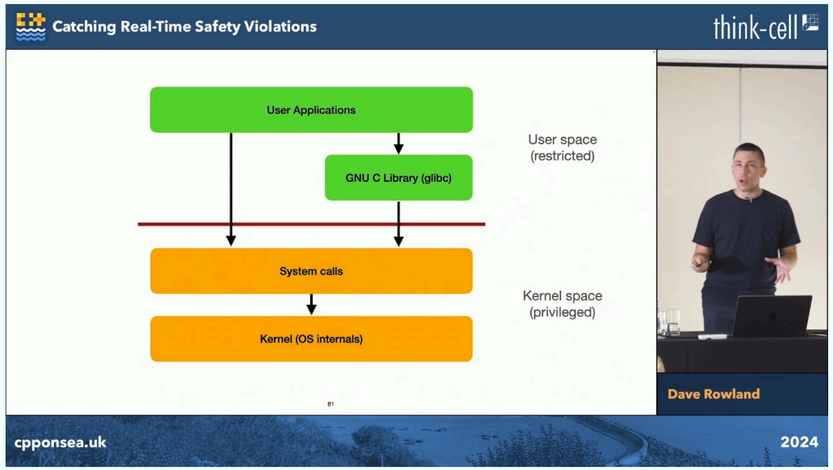
The Present

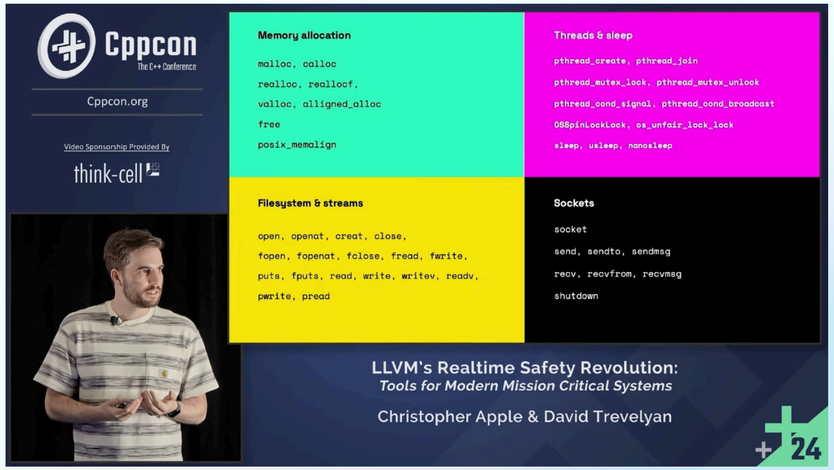
How Has it Changed? v1.x

- `auval` run
- `vstvalidator` run
- LV2 support
- CMake dependancy
- `rtcheck` real-time checking (macOS/Linux)

rtcheck

- Spoken about in 2024
- Heavily inspired by LLVM's RTSan
  - David Trevelyan, Ali Barker, Chris Apple
- Interposes system library function calls
- Works on linked libraries
- Enable via `--rtcheck`
  - Active on all `processBlock` tests
- https://github.com/Tracktion/rtcheck





rtcheck

```
void processBlock (juce::AudioBuffer<float>& buffer, juce::MidiBuffer&) override
{
    // The only line here the old new/delete overloads caught.
    std::vector<float> temp (buffer.getNumSamples());                                   // operator new









}
```

pluginval v2

- JUCE 9
- Refactored CLI internals
  - `pluginval validate [options] <path>`
  - Based on CLI11
  - JSON config file support
- junit/json output
  `--junit-xml <file>`
  `--json-results <file>`
- GitHub Action

GitHub Action

```yaml
- uses: Tracktion/pluginval/validate@v2
  with:
    plugins: |
      build/MyPlugin_artefacts/Release/VST3/MyPlugin.vst3
      build/MyPlugin_artefacts/Release/AU/MyPlugin.component
    strictness-level: 5
    args: --skip-gui-tests
```

- No need to clone or download pluginval
- Add as a step in your GitHub Action CI
- Downloads tagged pluginval for the OS
- Runs `pluginval validate` on `plugins` list
- Optional strictness level
- Optional args
- Optional `env`

pluginval validate

Plugin	Result	Tests
`build/tests/test_plugins/gain/pluginval_gain_artefacts/Release/VST3/pluginval Gain.vst3`	✅ passed	17
`build/Missing.vst3`	❌ failed	1 of 1 failed

Failed tests

- `build/Missing.vst3`
  - **Scan for plugins located in: /Users/runner/work/pluginval/pluginval/build/Missing.vst3**: !!! Test 1 failed: No types found. This usually means the plugin binary is missing or damaged, an incompatible format or that it is an AU that isn't found by macOS so can't be created.

```
pluginval validate <path>
```

```
pluginval  test <config.json>
           validate <path>
           help
```

Acceptance Testing

- Record a known output and save it:
  `pluginval test tests/acceptance/myFilter-sweep.json --record-missing`
- Check that output is the same in CI:
  `pluginval test tests/acceptance/myFilter-sweep.json`
- config.json:
- `state`
  - `parameters`
  - `file` (base64 encoded `getStateInformation`)
- `comparison`
  - `sample: tolerance`

```
{
  "name": "myReverb-default",
  "plugin": "/path/to/MyReverb.vst3",
  "input": { "audio": "inputs/drums.wav" },
  "reference": "refs/myReverb-default.wav",
  "state": { "parameters": { "Mix": 0.5, "Decay": 0.8 } },
  "sample_rate": 48000,
  "block_size": 512,
  "render_duration": 2.0,
  "playhead": { "bpm": 120,
                "time_signature": { "numerator": 4, "denominator": 4 } },
  "comparison": { "sample": 1e-6 }
}
```

pluginval acceptance tests

Config	Result	Tests
`nested/square-220.json`	✅ passed	1
`sine-440.json`	✅ passed	1

pluginval acceptance tests

Config	Result	Tests
`array.json`	❌ failed	
`wrong-ref.json`	❌ failed	1 of 1 failed

Failed tests

- `array.json`
  - Array configs aren't supported: use one config per file.
- `wrong-ref.json`
  - **sine-440**: [sample] max abs diff 1 > tolerance 3.05176e-05 (first at channel 0, sample 0)

The Standard?

If I'm building an audio plugin, are there any testing or validation tools I should use?

Yes. There are several excellent tools specifically designed for **testing audio plugins (VST3, Audio Units and AAX)**. I'd strongly recommend using a combination of automated plugin validation, real-time safety checks and DSP correctness testing.

The most important one to know about is **pluginval**.

1. Plugin validation tools

Tool	Formats	Purpose	Recommendation
pluginval	VST3, AU, others	Automated compatibility, stability and lifecycle testing	⭐ Essential
Steinberg VST3 Validator	VST3	Checks VST3 specification compliance	Essential for VST3
Apple auval	AU	Apple's official Audio Unit validator	Essential for AU
VST3 Plugin Test Host	VST3	Interactive host with built-in tests	Highly recommended
JUCE AudioPluginHost	VST3, AU	Lightweight test host for debugging	Highly recommended

pluginval

Developed by Tracktion, pluginval is probably the best starting point.

It automatically exercises a plugin through operations such as:

- Creating and destroying instances.
- Initialising and processing audio.
- Changing sample rates and buffer sizes.
- Testing different channel configurations.
- Changing parameters and testing automation.
- Saving and restoring plugin state.
- Opening and closing the editor.
- Fuzzing parameters and testing unusual conditions.

It runs on macOS, Windows and Linux and supports automated execution.

For example:

```bash
pluginval --strictness-level 5 /path/to/MyPlugin.vst3
```

Strictness ranges from 1 to 10. Level 5 is the recommended minimum for host compatibility, while higher levels include more demanding tests. It returns a nonzero exit code on failure, making it particularly suitable for CI/CD. GitHub

I'd make passing pluginval a requirement for every release build.

2. Real-time audio safety

This is particularly important for audio plugin development.

A plugin can pass normal validation and still produce glitches because it performs operations that are unsafe on the audio thread.

Useful tools

Tracktion rtcheck

A runtime checker designed to detect real-time safety violations.

Recent pluginval versions support rtcheck integration on macOS, including at higher strictness levels. GitHub

You should also run C++ sanitizers:

Tool	Detects
AddressSanitizer (ASan)	Buffer overflows, use-after-free, memory corruption
UndefinedBehaviorSanitizer (UBSan)	Undefined C++ behaviour
ThreadSanitizer (TSan)	Data races and concurrency errors
rtcheck	Operations that violate real-time execution constraints

In particular, look for memory allocations, mutex locking, blocking operations, file I/O and other unpredictable operations inside the audio callback.

Sanitizers generally need separate instrumented builds and test runs.

The most important one to know about is **pluginval**.

1. Plugin validation tools

Tool	Formats	Purpose	Recommendation
pluginval	VST3, AU, others	Automated compatibility, stability and lifecycle testing	⭐ Essential
Steinberg VST3 Validator	VST3	Checks VST3 specification compliance	Essential for VST3
Apple auval	AU	Apple's official Audio Unit validator	Essential for AU
VST3 Plugin Test Host	VST3	Interactive host with built-in tests	Highly recommended
JUCE AudioPluginHost	VST3, AU	Lightweight test host for debugging	Highly recommended

pluginval

Developed by Tracktion, pluginval is probably the best starting point.

It automatically exercises a plugin through operations such as:

- Creating and destroying instances.
- Initialising and processing audio.
- Changing sample rates and buffer sizes.
- Testing different channel configurations.
- Changing parameters and testing automation.
- Saving and restoring plugin state.
- Opening and closing the editor.
- Fuzzing parameters and testing unusual conditions.

It runs on macOS, Windows and Linux and supports automated execution.

Yes. There's a fairly standard toolkit, and most of it is free. What you need depends on which plugin formats you ship.

Format validators (run these before every release)

- **pluginval** (by Tracktion, open source). This is the closest thing to an industry-standard stress tester. It loads VST3 and AU plugins and works through a battery of tests: opening and closing editors, changing sample rates and block sizes, fuzzing parameters, and saving and restoring state. Strictness levels run from 1 to 10. It also runs headless, which makes it easy to add to CI. Aim to pass at level 5 or higher, and push toward 10.
- **auval** (Apple, built into macOS). Logic Pro won't load an Audio Unit that fails it, so this one is mandatory if you ship AU. Run `auval -a` to list installed units and `auval -v <type> <subtype> <manufacturer>` to validate yours.
- **Steinberg VST3 validator**. It ships with the VST3 SDK and checks conformance to the VST3 spec. The SDK also includes the VST3PluginTestHost, a test host for poking at the editor, automation, and bus layouts.
- **clap-validator**. Use this if you ship CLAP.
- **lv2lint**. Use this for LV2 on Linux.
- **AAX Validator**. If you're targeting Pro Tools, it comes with Avid's AAX SDK. AAX also requires PACE signing before release builds will load.

Building an audio plugin requires a mix of standard software testing approaches and specialized tools built specifically for the audio engineering ecosystem. Because plugins run inside various Digital Audio Workstations (DAWs) with unique host behaviors, standard QA isn't quite enough.

Essential testing and validation tools for audio plugin development include:

1. Dedicated Plugin Validation Tools

- **pluginval (Tracktion):** The industry-standard open-source cross-platform validator. It runs rigorous stress tests, memory checks, parameter fuzzing, and host-compliance tests (supporting VST, VST3, and AU). You can run it via GUI or in a headless CI/CD pipeline. GitHub + 1
- **Official SDK Validators:**
  - **Steinberg VST3Validator:** Comes with the VST3 SDK to ensure your plugin strictly conforms to Steinberg's VST3 specifications.
  - **Apple auval:** A command-line tool built into macOS used to validate Audio Units (`auval -v aufx yourplugin`). Passing `auval` is mandatory if you plan to distribute AU plugins on macOS.
  - **Avid DSH / DigiTest:** Used for validating AAX plugins for Pro Tools.

Is it a Standard?

Does it matter?

- Technically no
- Practically yes - people rely on it
- What matters?
  - Plugins get better
  - Users have a better experience

pluginval

`github.com/Tracktion/pluginval`

- Run it
- CI it
- Contribute

David Rowland
@drowaudio

Slides/video:
drowaudio.github.io/presentations

Questions?