When Homegrown Test Software Slows Product Development
Introduction

Does your company develop its own hardware test software? Many do. Just about all hardware testing software starts out this way. This approach is called “homegrown” testing, because the software development occurs internally using whatever software tools are available.

It can seem comforting to know that you have total control and intimate knowledge of your test software. You can develop just the right features to meet your specific test requirements. However, what happens when your team is unable to keep up? Troubles arise when product test requirements change faster than your test software can evolve. This can quickly spiral into long product development delays. Homegrown software tools are often unable to keep up with the new demands of multi-function products, and they cannot integrate with your company’s design and simulation software.

Within your organizations, different teams may have developed different test software, hindering collaboration and data exchange. And homegrown falls apart when key engineers leave the company. Finding and training people to develop and support your homegrown environment drain R&D resources that could be invested in accelerating innovation.

In this paper we explore the issues associated with homegrown test software, and discuss how modern, commercially available test software environments help.

How homegrown test software slows you down

- Unable keep pace with rapid product development and test cycles
- Difficult to integrate with design and simulation software
- Software does not facilitate collaboration
- Wastes valuable software development and support talent
Keeping Pace with Rapid Product Development and Test Cycles

Product development cycles continue to accelerate, as companies strive to keep up with competitive pressures and customer demands. Improvements in modern design and simulation software are major contributors to faster product development. These shorter design cycles increase the pressure on validation and manufacturing test teams, who may struggle to keep up and risk becoming the bottleneck in the overall release schedule. Homegrown test software might have kept up in the past, but today’s rapid design turns require a faster and more agile approach.

New product designs often require more tests, but test deadlines remain the same. And when designs take longer, there is less time in the test schedule. Homegrown software cannot keep pace with demands for faster test times. It is difficult to speed up software that was not designed to facilitate faster tests. It is equally challenging optimize test step flow and execution speed in a legacy system. Modern test software platforms provide built-in diagnostic tools, such as timing analyzers, to help engineers analyze which tests are consuming the most time, and to optimize overall test execution (Figure 1).

Design cycles are accelerating while our devices are getting more complex, generating more test data than we can analyze and manage.

Test Software Engineer, Wireless Component Manufacturer

---

1 “Modeling, Simulation and Analysis for Sustainable Product Development,” National Institute of Standards and Technology, NISTIR 7745, November 2010
2 “System-Level Modelers Race the Design Cycle,” Jean-Jacques DeLisle, Microwaves & RF, April 2, 2014
Integrating with Product Design and Simulation Software

Many companies are adopting increasingly complex product design and simulation software to help accelerate their development schedules. While these design tools help R&D teams speed up their processes, they place added responsibilities on the validation and testing teams. Modern design and simulation software can produce enormous amounts of data that the test team must validate. Integrating that design data with homegrown test software can be a formidable challenge, forcing the test developer to duplicate much of what already exists in the design tools.

Another challenge is ensuring that design simulation results match the test validation results. Unless your simulation and test software utilize the same measurement algorithms, it is impossible to guarantee consistent results. These inconsistencies often lead to confusion between the design and test teams when the results do not match, leading to further delays while each team strives to achieve consistency before the product can move into manufacturing.

Modern test environments integrate seamlessly with design and simulation software, ensuring that physical tests use the same measurement science as simulations. Any discrepancies point to actual issues with the product itself, not issues in the design or test software or methodology.

My R&D team can design a new product in 2 weeks and get samples by the end of the month…but it takes our DVT team a full year and a lot of struggle to develop the test system and procedures to begin data collection for validation and spec setting for manufacturing. We need to cut that time in half!

Product Development Manager, Semiconductor Manufacture
Collaborating Across Distributed Teams

With the rapid pace and increasing global nature of product design and development, it is vital that distributed test teams work as efficiently as possible. Collaboration is a challenge when each team has developed their test software with different algorithms and different user interfaces. Inconsistent test results lead to misunderstandings across teams, slowing down the development process. Using email to send results back and forth causes even more delays.

Homegrown test software typically cannot facilitate sharing information to other teams, such as R&D, or with design software or test software in other departments. The inability to share test software and test results across the product development lifecycle leads to costly, time-consuming duplicative work, and project rework (Figure 2).

![Figure 2. The inability to share test software, specifications, and results leads to duplicate work](image)

Homegrown software often provides only a small subset of the overall product testing workflow, typically developed to support one specific task or test. Modern test software environments provide an integrated suite of test automation, test project management, test station management, test monitoring, and analytics tools. With a modern architecture, users can exchange data from a variety of stations and locations, enhancing collaboration across (Figure 3). Modern tools provide common interfaces, common databases, and open APIs to integrate with existing hardware and software, optimizing your test operations.
What is TestOps?

Agile software development profoundly transformed software development. Few process innovations have yielded greater productivity improvements than Agile development and DevOps in the post-digital era. Ushered in by Agile methodology, DevOps teams and scrum workflows modernized software development, moving organizations from monolithic once-a-year software releases to an environment that delivers multiple software releases per day or week.

Today, a similar transformation is happening in test and measurement. Homegrown, single function tests are being replacing by modular, reusable components that can be highly parallelized to increase test efficiency and throughput. Combined with a modern collaborative work environment, a revolutionary new approach emerges — TestOps.

It’s hard to collaborate and stay in sync with my colleagues since we’re using different software and working in different cities.

DVT Engineer, Automotive Component Manufacturer
I struggle to handle data from our contract manufacturing partner even as I am in charge of optimizing manufacturing yield and turn-around time. The CM provides data in the form of CSV files and my team struggles to develop and maintain tools for analysis that enable sharing/collaboration. The tools also require maintenance and evolution as our processes change – and this is very time consuming. I also lack methods for analysis that can help me identify trends, issues and predictive analysis that would benefit our business.

Manufacturing/NPI Team Manager, Semiconductor Manufacturer

Optimizing engineering talent and supporting homegrown software

Homegrown tools are often adapted from existing tools and code, which may be outdated and difficult to support. During the development cycle, internal test software is created, but after release, the test improvement cycle can become extremely slow. Software that meets today’s requirements may not meet tomorrow’s needs.

Internal solutions and homegrown tools are often based on the unique skills of an individual developer. If that individual departs, replacing those skills can be hard to find. This is especially true when seeking software programmers who understand device testing.

Similar to data acquisition tools, test engineering teams often create their own software tools to analyze data. Sometimes this is done with spreadsheets or another standard tool, but other times, proprietary software systems are developed in-house.

These systems may be good enough, but compete for resources that could be deployed on more important tasks. The sheer volume of data being created also makes legacy tools inadequate for the task. Modern test environments provide integrated data management and analytics features to help visualize and interpret complex test results, and provide easy access for all team members (Figure 4).
We’re wasting too much valuable engineering time developing the same exact tests across multiple projects.

DVT Engineer Manager, Wireless Chipset Manufacturer

Internally developed test software tools can be challenging to maintain and expensive to manage. Often only one or a few key engineers know how to update this software. If the expert leaves the company or moves to another department, the rest of the team is forced to spend time re-learning the tools. Many times documentation is limited, and the tool may fall out of alignment with other systems that evolve.

The additional work drains resources that need to focus on actual testing instead of software development.

A modern test software environment frees engineers from developing and supporting homegrown test software, so they can focus on improving the overall test process. Rather than discard your existing test software, open application programming interfaces (APIs) make it easy to migrate critical elements of your homegrown software and link to an enterprise test software suite.

Modern environments require little or no programming expertise thanks to graphical user interfaces explicitly geared for optimizing test development (Figure 5). Moreover, quality vendors provide frequent updates to help you keep pace with rapidly changing test standards, along with a variety of support, training and consulting services.
We have one person in our team who spends nearly all his time updating our test software and producing the reports that we use to drive decisions. He should be doing engineering. And if he is not on site, we can’t hold our staff meetings because he’s the only person who knows how to use the homegrown tools. I don’t know what we’ll do if he ever leaves.

Validation Engineering VP, Semiconductor Manufacturer

**Optimize Test Operations with a Modern Test Software Environment**

Modern test software environments overcome many homegrown software limitations. A Carnegie Mellon SEI study on standardizing test software development processes shows standardizing the test software development process:

- Reduces time-to-market by 38 percent
- Eases workload by 76 percent
- Improves product quality by 80 percent; reduces defects shipped

[1](http://www.sei.cmu.edu)
Homegrown test software issues and the benefits of a modern test environment are summarized below:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Homegrown test software</th>
<th>Modern test environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep pace with rapid product development and test cycles</td>
<td>• Difficult to evolve rapidly as test needs expand</td>
<td>• Frequent vendor enhancements keep pace with evolving test standards</td>
</tr>
<tr>
<td>Integrate test with design and simulation</td>
<td>• Design software uses different algorithms than test software, yielding different results</td>
<td>• Common algorithms with integrated design &amp; test suites ensure consistent results</td>
</tr>
<tr>
<td></td>
<td>• Incompatible data formats hinder analysis across design, simulation, and test</td>
<td>• Standard data formats simplify analytics across design, simulation, and test</td>
</tr>
<tr>
<td>Collaborate across distributed teams</td>
<td>• Designed for a single-user, software not easily shared across teams</td>
<td>• Easily scales from single user to globally distributed teams</td>
</tr>
<tr>
<td></td>
<td>• Test results not readily available by all teams at all sites</td>
<td>• Cloud-based storage provides test plans and results with access by any user at any time</td>
</tr>
<tr>
<td>Optimize software development and support talent</td>
<td>• Knowledge lost if key engineers leave</td>
<td>• Requires minimal programming expertise</td>
</tr>
<tr>
<td></td>
<td>• Enhancements and support depend on the availability of key engineers</td>
<td>• Open APIs simplify integration with existing software</td>
</tr>
<tr>
<td></td>
<td>• Poorly documented APIs hinder code sharing and integration with other software</td>
<td>• Vendors provide support, software updates, training and consulting services</td>
</tr>
</tbody>
</table>
Conclusion

Teams using homegrown test software face significant challenges as product development cycles shrink, and pressure increases to deliver on time and meet customer and business demands. While homegrown test software kept pace in the past, it is increasingly difficult to maintain and support these legacy systems, and it is equally difficult to integrate yesterday’s homegrown solutions with today’s modern design systems.

If you are facing pressures to accelerate your test development processes, consider migrating your homegrown test software to a modern, distributed environment that connects your teams and test stations, and can scale from a single user to a global enterprise.

Keysight’s PathWave Test environment is your platform for modern test operations, giving you the power to collaborate across teams and manage your test workflows in any size organization. PathWave Test provides a variety of modular components with open APIs so you can integrate your existing homegrown software while migrating to a modern system at your own pace.

Learn more about Keysight’s modern test software environment at www.keysight.com/find/pathwavetest