Increased Complexity in Embedded Software Systems

Publish Date: Jun 24, 2014

Maintaining embedded software quality becomes more difficult as the complexity of intelligent devices increases. Mechanical components are combined or replaced with electrical and software-based systems, which provide higher levels of functionality. As companies shift focus, many have failed to efficiently test and deliver reliable products due to the rising cost of verifying additional software. This has led to highly publicized consumer safety issues and recalls. The number of defects rises proportionally with more lines of code. Engineering managers are looking for ways to mitigate the time and money spent on maintaining the quality of these complex systems so they can reduce time to market and retain budget for competitive features. Any company that has had to recall a product can attest that catching software defects earlier in the design cycle saves money, but many companies still need to find additional methods of identifying problems sooner.

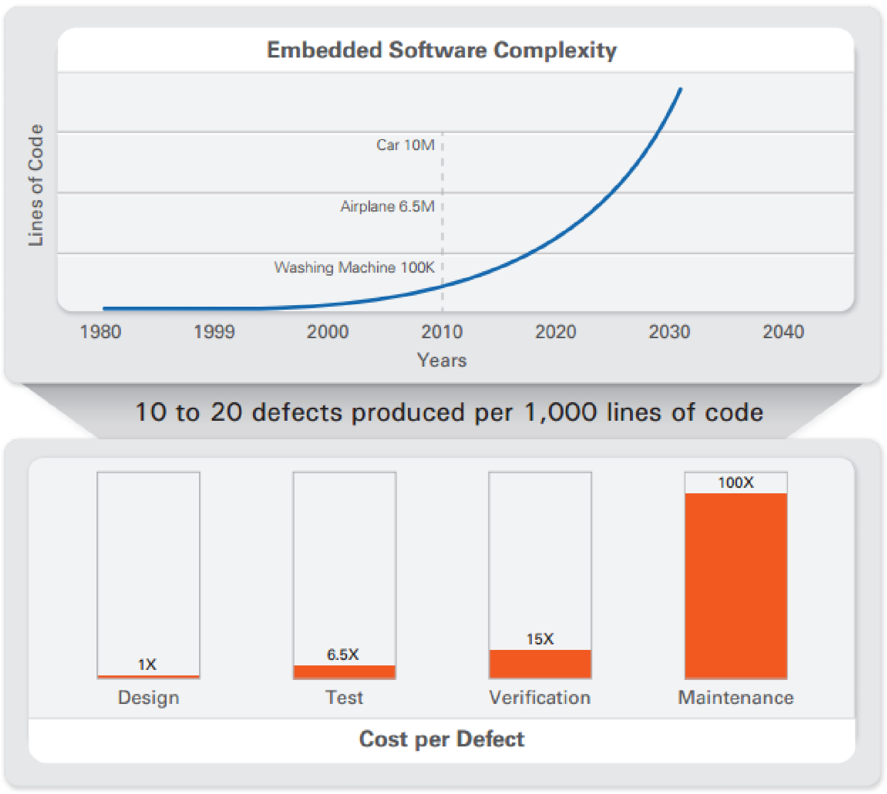

Figure 1. The recent growth in complexity necessitates increased attention to identifying defects.

Various models of the embedded software development process propose the most efficient methods of verification. Some emphasize continuous testing such as agile software development in which failed test results drive software changes in design and can even affect requirements as part of the plan. Over an iterative process, the product attains better quality. Many communication barriers still separate test, and organizations are not realizing the benefits of a complete solution. They should set a goal of total quality management through a single overarching system that traces all plan, design, and test resources together for organized informational analysis. But how well do teams handle data communication to and from test engineering? Between test and other system engineering teams (requirements, design, analysis, and more), some are geographically separated, some use incompatible IT tools, and some face language barriers. The IT software is not connected to the test machines and test software, so most teams rely on ineffective manual processes to deliver information about a product's ability to meet requirements. This includes spreadsheet data entry to trace results back to requirements, test results attached through email, and point-to-point communication for information about defects.

Much can be learned about improving this process through quality management tools that tie plan and design together. In system engineering, the requirements document acts as a central planning point to help guide development. The user can inspect related embedded software subcomponents and ensure that every feature is being developed for by linking individual requirements to a quality management tool. Users can immediately see all the affected code if there is a change in requirements. Defects are also traced to both the requirements and the code, so that all of the relevant information is accessible. To connect test, these tools include the same type of connections to requirements for scripts, results, and test code. Some engineers manually upload the information from their test machine, and others lack the time to perform this error-prone task. Very few use the proper tools to link test resources directly into this process. Today's quality management solution requires automation or the cycle is incomplete.

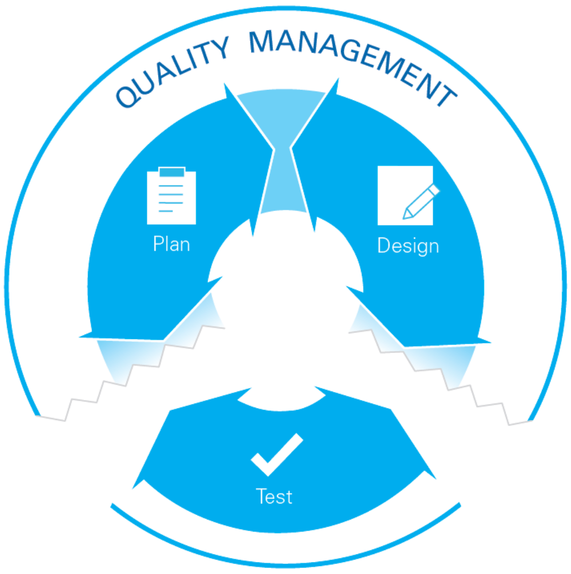

Figure 2. Most quality management solutions today do not fully incorporate test.

All system engineers need access to the same enterprise tools so that any team member can find information directly, without having to request it from somebody else. For example, the test software modules needed for automotive dynamometer testing might be the same as what were used in hardware-in-the-loop (HIL) testing. If the dynamometer test engineer never knew the HIL colleague already wrote the required test component, he would not know to ask for it and would double the test development time for that software. These resources should link to the requirements document as it is the one central piece of information that spans the entire life cycle. Both engineers would reference this document during their work and would see updates to the test code they are collaborating on. Similarly, the developers would not know what to fix until they know what is broken. An automated process could notify them of failures that affect their code so that they do not have to wait on an email report. This automated traceability goes along with automated testing. These steps tighten the iteration cycle between plan, design, and test by providing better collaboration.

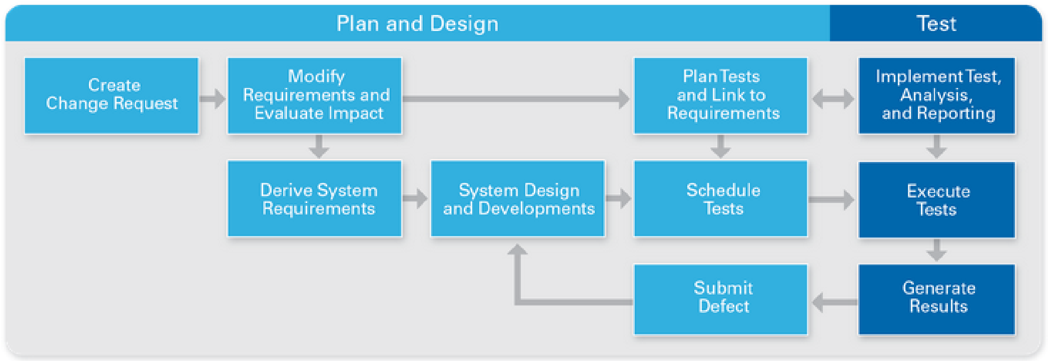
Plan and Design
Test
Create Change Request
Modify Requirements and Evaluate Impact
Plan Tests and Link to Requirements
Implement Test, Analysis, and Reporting
Derive System Requirements
System Design and Developments
Schedule Tests
Execute Tests
Submit Defect
Generate Results

Figure 3. A collaborative process must manage every point of communication.

When communication barriers are removed, each team can focus on its primary mission. The first way to achieve this is through a trusted automation process that brings a higher degree of confidence than manual data entry. Trust in data eliminates the costly and time-consuming efforts of over testing or under testing, so a test engineer can improve the efficiency of resource availability. Second, if the process is designed to automate result uploading, teams and individual members no longer rely on point-to-point communication to ensure test report accuracy. Design engineers are guaranteed to receive detailed analyses and make decisions quicker. Lastly, a manager can draw meaning from the powerful new data that can be collected. Delays and problem areas can be identified by analyzing pass/fail trends as related to requirements, software builds, testing phases, and all the other complexities that embedded products contain now that 100 percent of results are collected. With total quality management leading to end-to-end collaboration, any disparate information becomes a first-class citizen of a singular system and all engineers can collaborate.

The NI embedded software test platform completes the total quality management solution by enabling automated traceability for verification applications. Test architectures using the NI platform can connect to the industry-leading IBM Rational platform for minimal change or investment. By integrating these two portfolios through software, the embedded software complexity can be mitigated to yield lower cost of defects and lower risk of a recall.

Continue learning about NI's solution.