



# Software Verification and Validation Technologies and Tools

Moisés Rodríguez, Mario Piattini, and Christof Ebert

## From the Editor

Software quality matters—more than ever. Product liability means that we can only release what we trust. A single critical failure can cease the existence of a company or freelancer. But how to build quality and ensure necessary trust? Authors Moisés Rodríguez, Mario Piattini, and I evaluate current technologies and tools for verification and validation. Based on our industry and research experience, we cover the entire range from static analysis to test tools. I look forward to hearing from both readers and prospective authors about this department and the technologies you want to know more about. —Christof Ebert

**SOFTWARE QUALITY MATTERS**—more than ever. Software has become the most crucial infrastructure in this century. All businesses are software businesses because they based their operations and services in the Internet of Things, business intelligence (BI), artificial intelligence, cloud computing, social networks, and so forth. Classic IT and embedded systems are converging toward ubiquitous software. Systems will be connected and thus increasingly missing critical. The

digital transformation across industries depends on systems performing according to their requirements and needs. This importance of software quality has grown considerably over recent years.

Today, organizations invest around 30% of their IT budget in quality assurance (QA) and testing tasks in IT and more than 50% in safety-critical systems.<sup>1,5</sup> Moreover, organizations need to react in a very agile way, and they need to have institutionalized software processes and techniques producing software in a controlled and secure way. In fact, several companies are introducing over-the-air

seamless upgrades, DevOps, and continuous software engineering.<sup>2</sup> This implies continuous verification and continuous validation—way beyond classic smoke tests.

All these agile and continuous software engineering techniques, which seek the continuous improvement of the quality and the fulfilment of the expectations of the users, need to be supported by adequate tools to be efficient, effective, and really applicable. These tools could be used either by software producers or by software acquirers, who need to verify and validate the software products that are being outsourced. The software

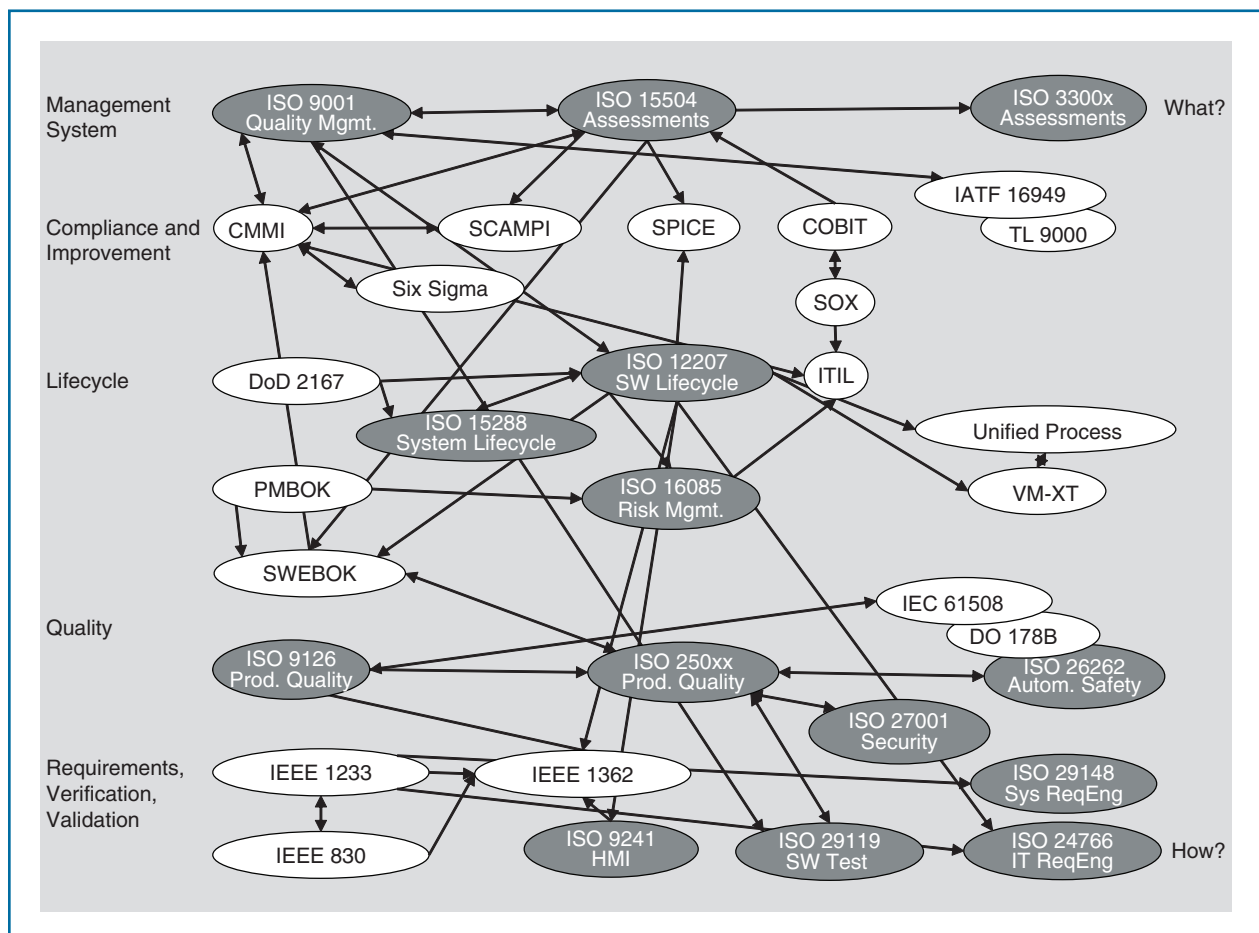
## Software quality matters— more than ever.

quality assessment could start as a self-assessment (by the own software developer organization), as a second-part assessment (by the acquirer of the software product), or as a third party (an accredited laboratory and/or a certification organization).<sup>3</sup>

Software verification and validation is considered in different models (such as the V-model, waterfall, spiral, and so on) and standards (namely, ISO/IEC 25000 SQUARE series,<sup>4</sup> or in the new ISO/IEC 12207:2017). Figure 1 shows the current software

standards as a standard quagmire with intrinsic relationships, references, and dependencies.<sup>5</sup>

On the one hand, if we focus on the software development process quality, we can highlight the ISO/IEC 12207 standard with its software verification and validation processes, as well as the V-model, where for each process on the left (development) there is the corresponding verification and validation process to the right (testing). Thus, the unit tests verify if the source code complies with the low-level design. The



**FIGURE 1.** The standards quagmire for the quality of the software process and product.

integration tests verify if the previously tested components fit; that is, they work in an integrated manner. System tests check whether the fully integrated product meets the specifications. And, finally, the acceptance tests verify if the product meets the expectations of the user or client.

On the other hand, focusing on the software product quality and following the ISO 25000 series, some of the main product quality characteristics are as follows:

- *Functional suitability*: the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
- *Maintainability*: the degree of effectiveness and efficiency with which a product or system can

be modified to improve it, correct it, or adapt it to changes in the environment and in the requirements.

- *Usability*: the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.
- *Security*: the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.
- *Performance efficiency*: this represents the performance relative to the amount of resources used under stated conditions.

Figure 2 presents the union of the two previous approaches (process and product quality) and relates them to the types of tools that can be used. So that, in a simple way, you can identify verification and validation tools that can be used for each of the phases of the V-model and each of the software product quality characteristics.

Static analysis tools are mainly used in the coding phase, so that developers can control from their IDEs if their work is compliant with coding standards. However, static analysis also serves to check aspects related to the structure of the modules or even the architecture of the software. Therefore, as can be seen in Figure 2, the static analysis tools box extends from the lowest coding level to the tests related to the verification of the architecture. In addition,

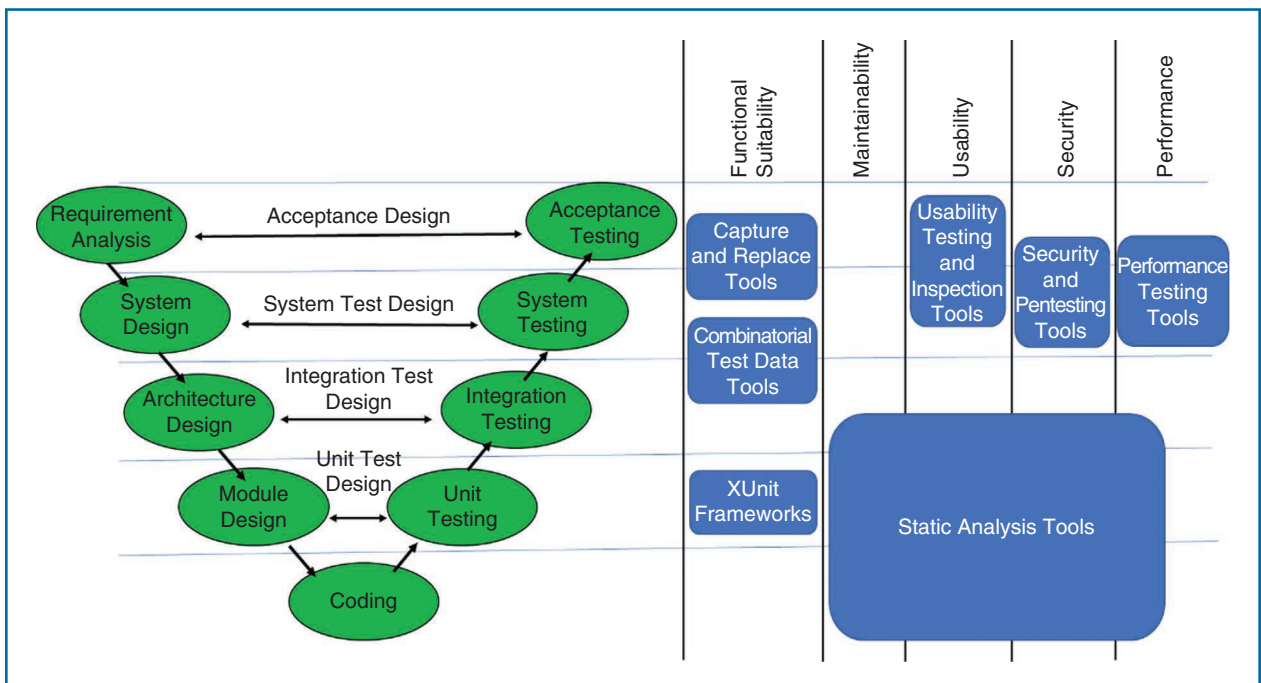


FIGURE 2. Tools for the main software quality characteristics.

static analysis tools are related to aspects of the four quality characteristics, as seen in Figure 2.

### Technologies and Tools

XUnit tools are used to verify the correct functioning of each module developed, seeking to reach the greatest coverage of lines of code. XUnit frameworks are probably the most used technology to automate tests. In

collected some of the most important for the top 10 languages according to TIOBE index.

### Maintainability Tools

Maintainability tools allow for analyzing the source code and verifying if it fulfills principles or rules about software modularity, complexity, readability, and so forth (Table 2).

Test-driven development  
is the role model.

such frameworks, test cases are written in an executable language and can be executed automatically. They also provide specific operations to implement the oracles of the test cases.

Combinatorial test data tools obtain test cases that calculate the Cartesian product of all the proposed input values, verifying the correct functioning of the different integrated modules. Thus, the pairwise strategies produce test cases that visit all the pairs of values for each two parameters, or, for example, the  $n$ -wise strategy is a generalization of the pairs, which visits all the tuples of  $n$  values.

Capture and replace tools are used to validate the correct and complete functioning of the system as well as during the acceptance tests. These tools allow registering the interactions of the testers with the application, storing them in a test script, and then being able to execute them automatically.

There are many tools for testing, practically as many as programming languages. In Table 1, we have

### Usability Tools

Usability testing tools are used to validate a software product by testing it on real users. This provides direct information on how real users will use the system. On the other hand, usability inspection tools serve experts to use different methods with which to validate the user interface without involving users (Table 3).

### Security Tools

Security testing tools allow for discovering system vulnerabilities and determining if the system protects data and maintains functionality as intended. Within these tools, the pentesting tools allow an authorized simulated attack on a software system or network through scanning and attack actions, created to look for weaknesses and then exploit them. They are also commonly known as *ethical piracy techniques* and *white hat piracy* (Table 4).

### Performance Tools

Performance testing tools allow establishing the system's reference behavior.

Unlike the previous ones, these tools do not intend to find defects in the application but instead focus on the measurement characteristics, such as response times, average time, or throughput (Table 5).

### Continuous Verification and Validation Tools

Continuous quality engineering means that verification and validation are deeply engrained to each single development step. Test-driven development is the role model: first think about the quality need, then develop the related software. Considering the benefits of continuous integration and the current boom in methodologies that implement it and take it to the extreme, such as DevOps, today it is increasingly important to have tools that allow verification and validation tasks to be carried out and integrated in the development cycle itself and automatically.

Therefore, a set of tools is presented here that serves to verify the quality characteristics previously presented in this article but which can also be used from a continuous integration environment based on Jenkins, Travis CI, Bamboo, GoCD, Ansible, or those similar. In this way, the validation and verification of quality is fully integrated into the lifecycle, carried out in an almost automatic manner, and totally transparent for the development teams.

Table 6 presents tools that have a cross scope to several quality characteristics and also allow for obtaining results for a global evaluation and high-level visualization. However, we must consider that the tools previously presented in this article for unit testing, capture and replace testing, and static analysis are also used in these verification and continuous validation environments, although they have not been included in Table 6.

Table 1. Functional testing tools.

Type of tools	Tool	Technology	Description	License	Degree of use
Functional Suitability	XUnit tools				
	Atum	PHP	A modern, simple, and intuitive PHP 5.3+ unit testing framework	Open source	1
	C++ test	C/C++	Framework that includes unit test generation and code coverage reporting	Commercial	2
	Cplusplus	C/C++	The premier unit testing framework for C++ and also for C code	LGPL	2
	Cplusplus	C/C++	Library and MS Visual Studio add-in to create and run unit tests	Open source	1
	Enhance PHP	PHP	An XUnit framework that includes mock and stub features	Commercial	1
	Go2xunit	Go	Converts go test output to XUnit or xunit.net-compatible XML output	Open source	1
	Jasmine	JavaScript	Behavior-driven development (BDD) framework for testing JavaScript code	Open source	2
	Jest	Java	Automated unit/component test generation and execution with code coverage and runtime error detection	Commercial	2
	JMock	Java	An extension of the JUnit framework to create mock objects	Free	2
	JUnit	Java	The most famous XUnit framework for Java	Open source	3
	JUnit	.NET	A model-based XUnit framework	Free	2
	Mocha	JavaScript	JavaScript test framework running on node.js	Open source	2
	NUnit	.NET	The most famous XUnit framework for all .Net languages	Open source	3
	Parasoft C/C++ test	C/C++	Automated unit/component test generation and execution on host or embedded systems with code coverage and runtime error detection	Commercial	2
	PHPUnit	PHP	An XUnit framework that reports results in XML and HTML, including coverage information	Open source	3
	pytest	Python	Framework makes it easy to write small tests, yet scales to support complex functional testing	MIT	1
	PyUnit	Python	Python language version of JUnit	Open source	1
	QA Systems Cantata	C/C++	Framework designed for testing embedded systems	Commercial	2
	QUnit	JavaScript	Tests any generic JavaScript code and is very useful for regression testing	Free	2
	SQLUnit	SQL	SQLUnit is a regression and unit testing harness for testing database-stored procedures	GPLv2	2
	utMySQL	SQL	Framework for MySQL V 5 using stored procedures	GPL	1
	Xunit.net	.NET	Written by the original inventor of NUnit v2 for unit testing C#, F#, VB.NET, and other .NET languages	Open source	2

Degree of use: 1 = low, 2 = medium, 3 = high. API: application programming interface. BDD: behavior-driven development. (Continued)

Table 1. Functional testing tools. (Continued)

Type of tools	Tool	Technology	Description	License	Degree of use
Functional Suitability	AETG (Automatic Efficient Test Generator)	Web	A web-based application that combines test data	Commercial	1
	AlIPairs	Perl	A command-line application	GPL	2
	CombTestWeb	Web	A web-based application that implements several combination algorithms, as well as tests from state machine specifications	Free	2
	IBM Intelligent Test Case Handler	Java	An Eclipse-based application for generating test value combinations	Commercial	2
	TestCover	Web	An online service of combinatorial testing	Commercial	1
	IBM Rational Robot	Multitechnology	Designed for e-commerce, enterprise resource planning, and client/server applications	Commercial	2
	Katalon Studio	Web	Functional automation test for API, web, and mobile	Free	1
	PesterCat	Web	Generates functional automation scripts in XML	Commercial	1
	Selenium	Web	Allows creating automated tests for web applications; tests can be recorded or scripted in multiple languages	Open source	3
	SilkTest	Multilanguage/multitechnology	Functional regression testing for web, mobile, desktop, and enterprise and packaged applications	Commercial	2
Degree of use: 1 = low, 2 = medium, 3 = high. API: application programming interface. BDD: behavior-driven development.	Squish GUI Testing	Java	GUI test automation tool for all kinds of cross-platform desktop, mobile, embedded, and web applications	Commercial	2
	TestComplete	Multilanguage/multitechnology	Allows creating automated tests for desktop, web, and mobile; tests can be recorded, scripted, or manually created with keyword-driven operations	Commercial	3
	Tricentis Tosca	Multilanguage/multitechnology	Automate API, web, mobile, BI, big data, and SAP tests	Commercial	2

Degree of use: 1 = low, 2 = medium, 3 = high. API: application programming interface. BDD: behavior-driven development.

Table 2. The maintainability testing tools.

Type of tools	Tool	Technology	Description	License	Degree of use
Maintainability	AndroidLint	Android	Scans Android project sources for potential bugs like missing translations, layout performance problems, unused resources, and so on	Free	2
	Checkstyle	Java	Help programmers write Java code that adheres to a coding standard and integrate with IDEs, maven, and so forth	LGPL 2.1	3
	CodeNarc	Groovy	Analyzes code for defects, bad practices, inconsistencies, style issues, and more	Open source	2
	Complexity Report	JavaScript	Software complexity analysis for JavaScript projects	Open source	2
	CPPCheck	C/C++	Code analysis to detect bugs and focuses on detecting undefined behavior and dangerous coding constructs	Open source	2
	CPPDepend	C/C++	Offers metrics related to the quality of the source code and coding standards	Commercial	2
	FxCop	.NET	Analyzes managed code assemblies and reports information about flaws	Free	3
	GMetrics	Groovy	Provides calculation and reporting of size and complexity metrics for Groovy source code	Open source	1
	JavaNCSS	Java	Offers metrics related to the size and complexity of the source code	GPL	2
	JDepend	Java	Generates quality indicators related to package design	Open source	3
	JSHint	JavaScript	Helps to detect errors and potential problems in your JavaScript code	Open source	1
	NDepend	.NET	Offers metrics related to the quality of the source code	Commercial	3
	OCLint	ObjectiveC	Helps to improve quality looking for potential problems like unused code, code smells, bad practices	Open source	2
	PHPCodeSniffer	PHP	Set of two PHP scripts to detect violations of a defined coding standard and to automatically correct coding standard violations	Open source	2
	PHPCPD	PHP	Copy/Paste Detector (CPD) for PHP code	Open source	2
	PHPDepend	PHP	Can be used to measure the quality of a software project and identify that parts of an application where a refactoring should be applied	GPLv3	2
	PHPMD	PHP	PHP equivalent of the well-known Java tool PMD	GPLv3	2
	PMD	Java, JavaScript, and Apex	Finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth	Open source	3
	Polyspace	C/C++	Used to check all MISRA (2004, 2012) rules (directives, nondirectives)	Commercial	2
	Simian	Multilanguage	Is a duplicate source code detector for Java, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic, and Groovy	Free	2
	StyleCop	.NET	Checks C# code for conformance to StyleCop's recommended coding styles and a subset of Microsoft's .NET Framework Design Guidelines	Open source	2
	XClarity	ObjectiveC	Helps to analyze code structure, specify design rules, do effective code reviews, and master evolution by comparing different versions of the code	Commercial	2

Degree of use: 1 = low, 2 = medium, 3 = high.



Table 3. The usability testing tools.					
Type of tools	Tool	Technology	Description	License	Degree of use
Static analysis tools	AccessLint	Web	Integrates automated web accessibility testing into your development workflow	Free and commercial	2
	SiteImprove Content & Accessibility	Web	Analyzes a website regarding the accessibility guidelines (WCAG) in search of errors and accessibility recommendations	Commercial	2
	TAW	Web	Web accessibility test with WCAG 2.0 guidelines (A, AA)	Free	3
	Tenon	Web	Can identify up to 508 WCAG 2.0 accessibility issues in any environment	Commercial	1
	W3C CSS Validation Service	Web	Check Cascading Style Sheets (CSS) and HTML documents (X) with style sheets	Free	3
	W3C Link Checker	Web	Detects broken links in a website	Free	3
	W3C Markup Validation Service	Web	Validate that the (X) HTML is well formed according to the doctype declared in the document; supports HTML 5 validation	Free	3
	Website Grader	Web	Grades a website's User Experience UX in four areas (mobile, SEO, performance, and security)	Free	2
	Usability testing and inspection tools	CrazyEgg	Web	Heat-mapping software that enables testers to gain a better understanding of how visitors are interacting with their website	Commercial
Hotjar		Web	Complete analytics and feedback tool that helps users identify opportunities for improvement (heat maps, visitor recordings, surveys, and more)	Commercial	3
Loop 11		Web	Creates usability tests and allows customizing your test template, assign tasks and objectives, and ask usability questions	Commercial	2
Morae		Web	Sophisticated suite of tools to help users collect data that can be used to inform their optimization strategy	Commercial	1
Responsinator		Web	Allows testing web pages to see if they comply with "responsive design"	Free	2
Userfeel		Web	Online usability testing service that provides users with a way to understand how visitors are responding to a website	Commercial	2
UserTesting		Web	Service to test the customer experience on real users and find problems early in development	Free	1

Degree of use: 1 = low, 2 = medium, 3 = high.



Table 4. The security testing tools.

Type of tools	Tool	Technology	Description	License	Degree of use
Static analysis tools	Bandit	Python	Finds common security issues in Python code	Free	1
	Brakeman	Ruby	Vulnerability scanner specifically designed for Ruby on Rails applications	Free	1
	BugScout	Java, .NET, and PHP	Detects potential security risks in applications before they reach production and can be hacked	Commercial	3
	Flaw Finder	C/C++	Simple program that examines C/C++ source code and reports possible security weaknesses sorted by risk level	Free	2
	Google SearchDiggity	Multitech	Uses Google Code Search to identify vulnerabilities in open source code projects hosted by Google Code	Free	2
	Klocwork	C, C++, C#, and Java	Integrates security analysis in IDEs, build systems, continuous integration tools, and any team's natural workflow and checks secure coding standards like OWASP, MISRA, CWE, and more	Commercial	3
	Phpcs-security-audit	PHP	PHP CodeSniffer rules that find vulnerabilities and weaknesses related to security in PHP code	Free	1
	Puma Scan	.NET	Software security analyzer that provides real-time, continuous source code analysis for C# applications	Commercial	1
	Roslyn Security Guard	.NET	Detects 29 vulnerability patterns with 69 different signatures	Free	2
	Visual Code Grepper	C++, C#, VB, PHP, Java, and PL/SQL	Code review tool for identifying bad/insecure code	Free	1
	Microfocus Fortify	C/C++, COBOL, Java (including Android), .NET, JavaScript, Objective-C, PHP, PL/SQL, Swift, among others	DAST and SAST tool to verify security from development to implementation phase	Commercial	3
	OWASP ZAP	Web	Penetration testing tool to find a variety of security vulnerabilities in web apps, even during the development and testing phases	Open source	3
	Security AppScan	Java, JavaScript, PHP, JSP, C/C++, Perl, .NET, Python, ASP, SQL, among others	DAST and SAST tool that is intended to test web applications for security vulnerabilities during the development process	Commercial	3
	Vega	Web	Web security scanner and web security testing platform to test the security of web applications	Free	2
Security pentesting tools	Veracode	JAVA/JSP, PHP or any other engine-driven web application	Allows DAST and SAST too, simulates a malicious user by attacking and probing, and seeing what results are not part of the expected result set	Commercial	3
	W3af	Web	Web application attack and audit framework with more than 200 vulnerabilities	GPLv2	2
	Wapiti	Web	Command-line application that allows assessing the security of your web applications	GPLv2	2

Degree of use: 1 = low, 2 = medium, 3 = high. DAST: dynamic application security testing; SAST: static application security testing.

Graceful degradation and fail-operational scenarios need to be designed and deployed.

To manage the activities carried out with all the previous types of tools, there are also test case management tools that allow taking

overall control of the verification and validation process. These include, for example, Test Link, Test Rail, Microfocus Quality Center (previously

HP), VSTS, IBM Rational Quality Manager, XStudio, and more.

### Where Do We Go From Here?

Our society depends on software. While humans might survive without software, it is certainly not the society as we know it. With increasing dependencies on software across all infrastructures, software quality matters more than ever. So, design and quality processes, methods, and technologies must be continuously used to safeguard our infrastructures.

Table 5. The performance testing tools.

	Type of tools	Tool	Technology	Description	License	Degree of use
Performance	Static analysis tools	Rules of Findbugs	Java	Provides a set of performance rules	Open source	2
		Rules of Gendarme	.NET	Provides a set of specific performance rules	Open source	1
		Rules of PMD	Java	Provides a set of rules for suboptimal code that can be checked against the source code	Open source	2
		SQL Monitor	SQL	Provides a set of rules regarding best practices for SQL performance	Commercial	2
	Performance testing tools	Apache JMeter	Web	Provides load testing capabilities and performance measurement for functional behavior	Open source	3
		Google PageSpeed Insights	Web	Analysis of website load performance and compliance of best practices regarding optimization	Free	2
		GTMetric	Web	Provides web load performance analysis and recommendations on how to optimize it	Free	2
		Jetbrains dotTrace	.NET	Performance profiler; provides analysis of calls execution time	Commercial	3
		LoadUI NG Pro	Web/API	Provides performance and load testing capabilities through automation and scripting	Commercial	2
		Microfocus LoadRunner	Web	Provides measuring of system behavior and performance under load conditions	Commercial	3
		Open STA	Web/Windows desktop	Provides web load testing, HTTP stress testing, and performance testing capabilities	Open source	2
		Rational Performance Tester	Web/server	Provides automated performance testing of web- and server-based applications	Commercial	3
		SmartMeter.io	Web	Provides automation for load and performance testing	Commercial	2
		WebLOAD	Web/mobile	Provides load testing, performance testing, and stress testing capabilities for web and mobile applications	Commercial	2

Degree of use: 1 = low, 2 = medium, 3 = high.

Risks steadily increase, as we see with the current fast growth of cybersecurity threats and insufficient usability. Thus, our verification and validation methods have to evolve even faster. At the same time, we need to prepare on a dual track for failure. Quality strategies thus not only mean finding defects but also hardening systems to make them robust. Graceful degradation and

fail-operational scenarios need to be designed and deployed.

With critical defects and attack schemes evolving after release, a core of any quality strategy is continuous verification and validation. Corrections and changes must be deployed in a fluid and continuous scheme, reliably over the air. We will face future scenarios where software-driven systems and maybe whole infrastructures must not be

started if they do not include all the latest software upgrades. Automotive vehicles and manufacturing processes that, by definition, are safety critical fall into that category. Even more demanding are medical devices that must provide a hierarchical software assurance because there is no room for failure.

Verification and validation depend on many factors. Every organization implements its own methodology and

**Table 6. Continuous verification and validation tools.**

	Tool	Technology	Description	License	Degree of use
Continuous Verification and Validation	Cast Highlight	Analyzes 18 different languages	Cast is one of the first companies that has started with quality measurements, and Highlight is one of its tools that is integrated with the rest of the lifecycle tools, allowing assess and track application portfolio health, identifying security vulnerabilities, and making application benchmarks.	Commercial	3
	Checkmarx Software Exposure Platform	Supports more than 20 coding and scripting languages	It is a suite of tools that allows aligning software security with DevOps culture, detecting, prioritizing, and remediating exposure across the software development lifecycle.	Commercial	2
	Codacy	Analyzes more than 20 different languages	It allows controlling both the quality and the security of the code and integrates the analyses in the development workflow. It offers the possibility to monitor the changes in the different versions of the product, and it also allows the user to customize the quality standards and the measurement thresholds.	Free and commercial	1
	Coverity Scan	Analyzes Java, C/C++, C#, JavaScript, Ruby, or Python	It is a static-analysis cloud-based service for the open source community that is integrated with GitHub and Travis CI for the continuous verification and validation.	Free and commercial	2
	Kiuwan	Analyzes more than 30 different languages	It allows evaluating the maintainability, efficiency, security, portability, and reliability of a software product. It is integrated with the rest of the lifecycle tools to allow continuous verification and validation. In addition, it allows making plans for the correction of defects and also estimating the necessary times.	commercial	3
	SonarQube	Analyzes more than 20 different languages	It is one of the references and most used tools in the continuous validation of software quality. It obtains an evaluation related to the maintainability, security, and reliability of the product, detecting code smells, vulnerabilities, and bugs. It is integrated into the development cycle and allows validating the quality of any change that is made throughout the life of the product.	Free for some languages and commercial for others	3
	Square	Analyzes more than 20 different languages	It collects and aggregates with its own results the measures from various external tools, like rule checking, test coverage, change requests, configuration management, and requirement management tools. It allows showing dashboards and reports and making optimized action plans.	Commercial	1

Degree of use: 1 = low, 2 = medium, 3 = high.



**MOISÉS RODRÍGUEZ** is the managing director of AQCLab and an associate professor at the University of Castilla-La Mancha, Spain. His research interests include software and data quality assessment, improvement, and certification. Rodríguez received a Ph.D. in computer science from the University of Castilla-La Mancha, Spain. Contact him at [mrodriguez@aqclab.es](mailto:mrodriguez@aqclab.es).



**MARIO PIATTINI** is the managing director of Alarcos Research Group and a full professor at the University of Castilla-La Mancha, Spain. His research interests include software and data quality, information systems audit and security, and IT governance. Piattini received a Ph.D. in computer science from Madrid Technical University, Spain. Contact him at [mario.piattini@uclm.es](mailto:mario.piattini@uclm.es).




**CHRISTOF EBERT** is the managing director of Vector Consulting Services. He is on the *IEEE Software* editorial board and teaches at the University of Stuttgart and the Sorbonne in Paris. Contact him at [christof.ebert@vector.com](mailto:christof.ebert@vector.com).

development environment, based on a combination of several of the tools presented in this article. It is, however, relevant to not only deploy tools but also build the necessary verification and validation competences. Too often, we see solid tool chains but no tangible test strategy. To mitigate these pure human risks, software must increasingly be capable to automatically detect its own defects and failure points.

With artificial intelligence and machine learning, we need to satisfy algorithmic transparency. For

instance, what are the rules in an obviously not any more algorithmically tangible neural network to determine who gets a credit or how an autonomous vehicle might react with several hazards at the same time? Classic traceability and regression testing will certainly not work. Rather, future verification and validation tools will include more intelligence based on big data exploits, business intelligence, and their own learning to learn about and improve software quality in a dynamic way.

The famous philosopher Aristotle observed that “Quality is not an act. It is a habit.” Let us thus focus not primarily on a wealth of tools but on a strong software quality culture and competences. 

## References

1. Capgemini, MicroFocus, and Sogeti. “World quality report,” 2018. [Online]. Available: <https://www.capgemini.com/service/world-quality-report-2018-19/>
2. B. Fitzgerald and K. Stol, “Continuous software engineering: A roadmap and agenda,” *J. Syst. Softw.*, vol. 123, pp. 176–189, Jan. 2017.
3. M. Rodríguez, M. Piattini, and C. M. Fernández, “A hard look at software quality,” *Quality Progress*, vol. 48, pp. 30–36, Sept. 2015.
4. *Software Engineering—Software Product Quality Requirements and Evaluation (SQuARE)—Quality Model*, ISO/IEC Standard 25010, 2011.
5. C. Ebert, *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*. Hoboken, NJ: Wiley—IEEE Comput. Soc. Press, 2011.



IEEE COMPUTER SOCIETY  
**DIGITAL LIBRARY**

Access all your IEEE Computer Society subscriptions at

**computer.org**  
**/mysubscriptions**