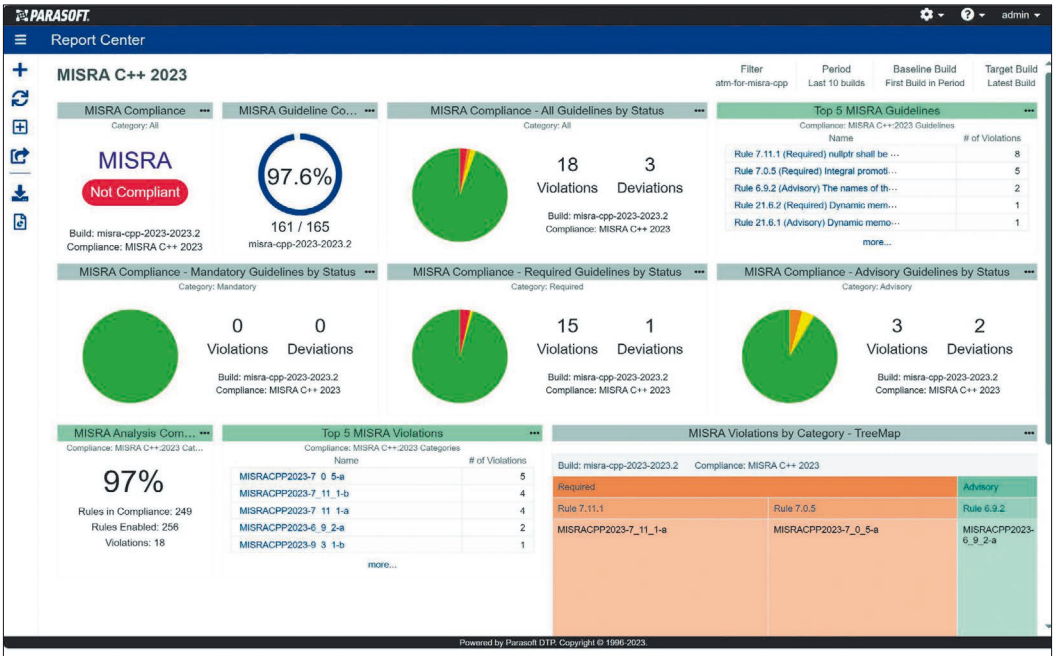
Software-Entwicklungsprozess für sicherheitskritische Systeme



Complete MISRA C++ 2023 Support Empowers C++ Developers

Bei sicherheitskritischen Systemen muss die gesamte Software während der Entwicklung strenge Sicherheitsstandards erfüllen. Diese werden regelmäßig aktualisiert, um Probleme und Schwachstellen zu beheben, sobald sie auftreten. Darum müssen Softwareentwickler für solche Systeme immer auf dem neuesten Stand sein und bei der Entwicklung agil bleiben. Sie müssen während des gesamten Lebenszyklus so viele reale Fehlerszenarien wie möglich berücksichtigen und die Sicherheitsstandards ihrer Branche einhalten. Dies erfordert auch robuste Tests, Validierung und Verifizierung. Denn ein Versagen könnte den Verlust von Menschenleben, Sachschäden oder Umweltschäden bedeuten – selbst in kleinem Maßstab. Dies ist für die Softwareentwicklung in diesen Branchen inakzeptabel.

Autor:
Ricardo Camacho
Director of Safety & Security Compliance
Parasoft Deutschland GmbH
www.parasoft.com

Normen und Vorschriften

Die 1947 gegründete Internationale Organisation für Normung (ISO) legt mit über 160 Mitgliedern aus vielen Ländern Sicherheitsstandards für zahlreiche Branchen fest. Zusammen mit anderen Gremien arbeitet die ISO ständig an der Entwicklung und Neubewertung ihrer Sicherheitsstandards für Tausende von Systemen in vielen Branchen.

Für jede Branche gibt es eigene Gremien, die sich mit der Überwachung und Regulierung sicherheitskritischer Geräte und Software befassen. So gibt es etwa in der Automobilindustrie die internationale Norm ISO 26262 für funktionale Sicherheit und ISO/SAE 21434 für Fahrzeugsicherheit. Die Sicherheit für medizinische Geräte regeln die Normen IEC 62304 und IEC 62443 für. Darüber hinaus gibt es Programmierstandards wie MISRA und AUTOSAR C++ 14, die bekanntermaßen in einer Vielzahl anderer Branchen wie Luft- und Raumfahrt, Militär, Schienenverkehr und Medizintechnik Anwendung finden. Weitere Sicherheitsstandards für verschiedene Arten von Anwendungen sind CERT, CWE und OWASP.

Im Wesentlichen existieren bestimmte Goldstandards, die Entwickler in jeder Branche einhalten müssen, und einige, die branchenübergreifend gelten. Diese Standards werden ständig auf der Grundlage der neuesten Technologien und der Möglichkeiten, diese Technologien zu nutzen, aktualisiert.

Anforderungen

Es gibt fünf wesentliche Säulen für Anforderungen in der sicherheitskritischen Softwareentwicklung:

1. **Eindeutig**. Sowohl in der Dokumentation als auch im Code selbst muss Einigkeit darüber herrschen, was die Anforderungen bedeuten, ohne Interpretationen oder Vermutungen.

2. **Verständlich**. Programmierer und Manager sollten in der Lage sein, die Anforderungen während des Code Reviews oder des Testprozesses aufwärts und abwärts zu verfolgen.

3. **Reproduzierbar**. Das Programm und die Software müssen die erwarteten Ergebnisse liefern. Sie sollten wie gefordert funktionieren und auch im Fehlerfall die erwarteten Daten und Ausgaben liefern.

4. **Konsistent**. Es sollte keine widersprüchlichen Anforderungsmerkmale hinsichtlich Zeitplanung, Leistung, Priorisierung etc. geben. Auch die Terminologie in der Dokumentation sollte einheitlich und konsistent sein, um Verwirrung zu vermeiden.

5. **Testbar**. Gut entwickelte sicherheitskritische Programme erfüllen die Anforderungen in allen Testarten.

Best Practices

Die Folgen sicherheitskritischer Fehler können von einer Batterie, die während des Betriebs zu heiß wird, bis zum katastrophalen Ausfall eines Flugzeugtriebwerks reichen. Sie können auch Menschenleben kosten, Sach- und Umweltschäden verursachen, finanzielle Verluste nach sich ziehen oder sogar die nationale Sicherheit gefährden. Das macht das Einhalten von Best Practices nicht nur ratsam, sondern zwingend erforderlich:

- Genaue Aufzeichnungen und dokumentierte Historie oder Rückverfolgbarkeit.
- Schulung der Programmierer in Bezug auf bewährte Verfahren, Methodik und die Bedeutung der Dokumentation.
- Nutzung der für das Projekt am besten geeigneten Programmierstandards.
- Komplexität des Codes niedrig halten, denn komplexer Code in komplexen Systemen ist schwer zu testen und kann zu mehr Fehlern führen.
- Beim Entwickeln der Software den Endbenutzer im Hinterkopf behalten.

Zu beachtende Faktoren

Das Ziel bei der Entwicklung sicherheitskritischer Software ist nicht Geschwindigkeit oder Funktionalität, sondern Sicherheit. Dies ist der wichtigste Aspekt, der bei der Programmierung für sicherheitskritische Systeme berücksichtigt werden muss. Neben der Einhaltung

von Sicherheitsanforderungen und formalen Methoden sind auch diese Punkte zu beachten:

- Berücksichtigung der Software-Architektur, denn ein gutes Design kann Upgrades erleichtern.
- Sicherheitsbelange, Risikominderung und die Einhaltung von Vorschriften machen das Projekt komplexer.
- Ein ausfallsicheres Design bedeutet, dass das Gerät und/oder die Software auf sichere oder erwartete und vorhersehbare Weise ausfallen kann.
- Nur ein bestimmtes Maß an Gefährdung und Ausfall ist zulässig.
- Bei der Entwicklung sicherheitskritischer Software können verschiedene Methoden des Entwicklungslebenszyklus wie Spirale, Wasserfall, Agile und Tandem zum Einsatz kommen.
- Man sollte beim Testen der Software weiter gehen, als man es für notwendig erachtet, und dann weitermachen.
- Für die Verbesserung sicherheitskritischer Systemsoftware sind Daten aus der Praxis erforderlich.

Support durch moderne Tools und Technologien

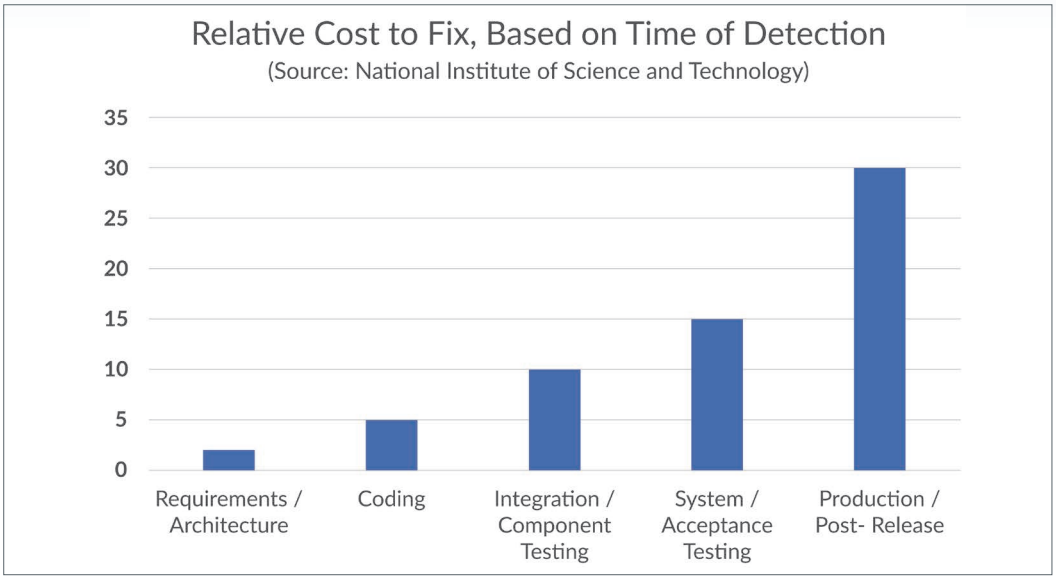
Die Verwendung eines zertifizierten Werkzeugs kann Sicherheit bieten, denn damit lässt sich die Konformität zuverlässig erzielen. Allerdings erfordert konforme Software die richtigen Entwicklungsmethoden, um lange Entwicklungszeiten und -kosten zu vermeiden. Und genau hier kommt der Shift-Left-Ansatz (zeitliche Vorverlegung) ins Spiel.

Koexistenz

Ein gutes Beispiel für agile Methoden ist die Koexistenz von DevOps CI/CD und Scrum, die Silos auflöst, Kommunikation fördert, Produktivität ermöglicht und Verifikation und Validierung automatisiert.

Automatisierte Tests

Die Aufrechterhaltung von Qualität und Erschwinglichkeit ist für viele Entwicklungsteams zu einer Priorität und zu einer einzigartigen Herausforderung geworden. Ein Beispiel ist die Automatisierung von Softwaretests, die in eine CI/CD-DevOps-Bereitstellung für die Automobilindustrie integriert ist. CI/CD-Pipelines bieten kontinuierliche Tests, die Projektkosten senken und Projektlaufzeiten verkürzen können. Selbst wenn strenge Sicherheitsstandards wie ISO 26262 eingehalten werden, bieten automatisierte Tests zusätzliche Codeabdeckung, Sicherheit und verwertbare Daten.



Stattet man die Entwickler mit den besten Werkzeugen für diese Aufgabe aus, erleichtert das deren Arbeit. Dies wiederum kann zu einer strafferen Dokumentation, Konsistenz und einer pünktlichen und budgetgerechten Auslieferung des Produkts führen, wenn nicht sogar den Termin verkürzen und das angesetzte Budget unterbieten.

Anforderungen

Sicherheitskritische Software bedeutet komplexe Systeme, die zur Unterstützung der Entwickler eine konsistente und eindeutige Dokumentation, verwertbare Testdaten und Werkzeuge erfordern. Unit-Tests, anforderungsbasierte Tests, Regressionstests, Sicherheitstests und Integrationstests – alle müssen die Rückverfolgbarkeit und Wiederholbarkeit von Anforderungen beinhalten. Automatisierte und TÜV SÜD-zertifizierte Lösungen (beispielsweise C/C++test) können die Arbeitsbelastung von Entwicklern und Testern reduzieren und gleichzeitig die Codequalität, den Abdeckungsgrad und die allgemeine Produktqualität verbessern. Idealerweise unterstützen die Werkzeuge viele Sicherheitsstandards wie ISO 26262, IEC 62304, DO-178C, IEC 61508, UL 2900, DISA ASD STIG, OWASP, MISRA, AUTOSAR, CERT, CWE, PCI DSS. Mit solchen modernen Tools können Teams Fehler frühzeitig erkennen, die Einhaltung von Industriestandards automatisieren und intelligent testen.

Reaktion auf Industrietrends

Weil auch bei der Entwicklung sicherheitskritischer Systeme die Kosten steigen und zugleich die Komplexität zunimmt, müssen Entwickler größere Sorgfalt auf die Dokumentation legen, und Manager müssen bereit sein, ihre Teams entsprechend den Sicherheitsstandards zu schulen - und zwar regelmäßig, um den aktuellen Wissensstand zu gewährleisten.

Wenn es jedoch darum geht, wie die Software selbst auf Branchentrends reagieren soll, wird es etwas undurchsichtiger. Dies ist dem Internet der Dinge, dem maschinellen Lernen und der künstlichen Intelligenz zu verdanken - sowohl in nützlicher als auch in hinderlicher Weise. Während es vorteilhaft ist, dass immer mehr Dinge miteinander kommunizieren oder sich bei Bedarf sogar an verschiedene Szenarien anpassen, kann dies auch zu zusätzlichen Sicherheitslücken führen. Die Planung für die unvermeidliche Ausnutzung dieser neuen Bedrohungsvektoren macht bereits komplexe Systeme in der Softwareentwicklung noch komplexer. Die Testautomatisierung hat begonnen, sich in Richtung KI und maschinelles Lernen zu entwickeln, um diese Lücke zu schließen, bevor sie zu groß wird.

Fazit

Probleme von vornherein zu vermeiden, ist einfacher als sie zu bekämpfen, wenn sie auftreten. Dieser Aspekt der agilen Methodik ist bei der Entwicklung sicherheitskritischer Software von entscheidender Bedeutung. Mit den richtigen Werkzeugen lässt sich der Prozess noch effizienter gestalten. Der Aufbau einer soliden Code-Architektur, die Beibehaltung einer konsistenten und systematischen Dokumentationssprache sowie kontinuierliche und angemessene Tests können dem Team helfen, die Konformitäts-Standards zu erfüllen. ◄