Elektronik medical

Innovative Produkte und Lösungen für die Medizintechnik

Code-Testing für Medizingeräte

Software über die IEC 62304 hinaus prüfen

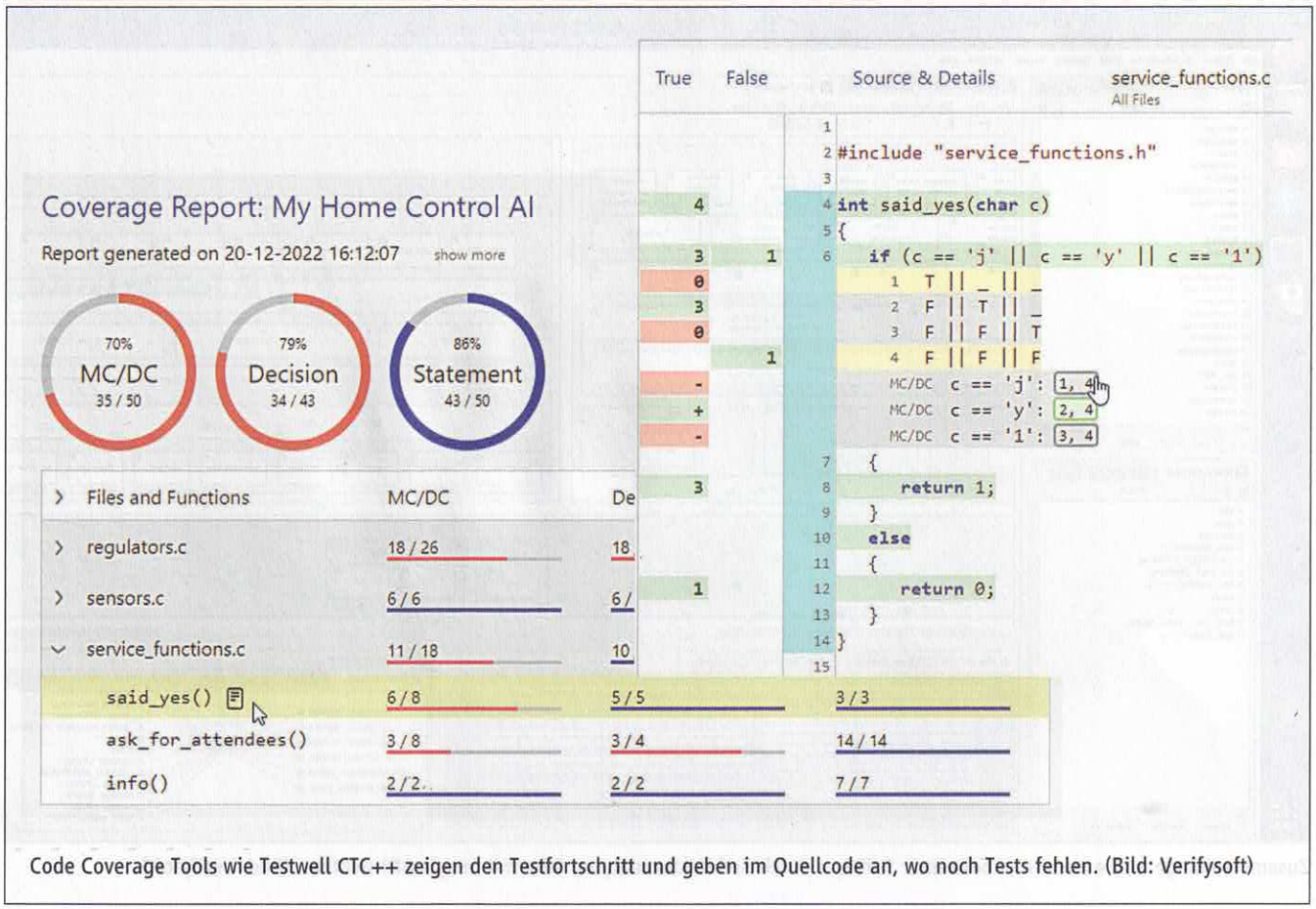
Seite 10

Code-Testing für Medizingeräte

Software über die IEC 62304 hinaus prüfen

Die Bedeutung von medizinischer Software in Diagnostik und Therapie steigt stetig. Die Digitalisierung birgt jedoch neue Risiken wie Fehlfunktionen oder Cyberattacken. Mittels Software-Testing können MedTech-Hersteller die Gefahren minimieren und Medizinprodukte mit sehr hoher Qualität, MDR- und FDA-konform sowie kostengünstig entwickeln.

Von Klaus Lambertz
Verifysoft Technology



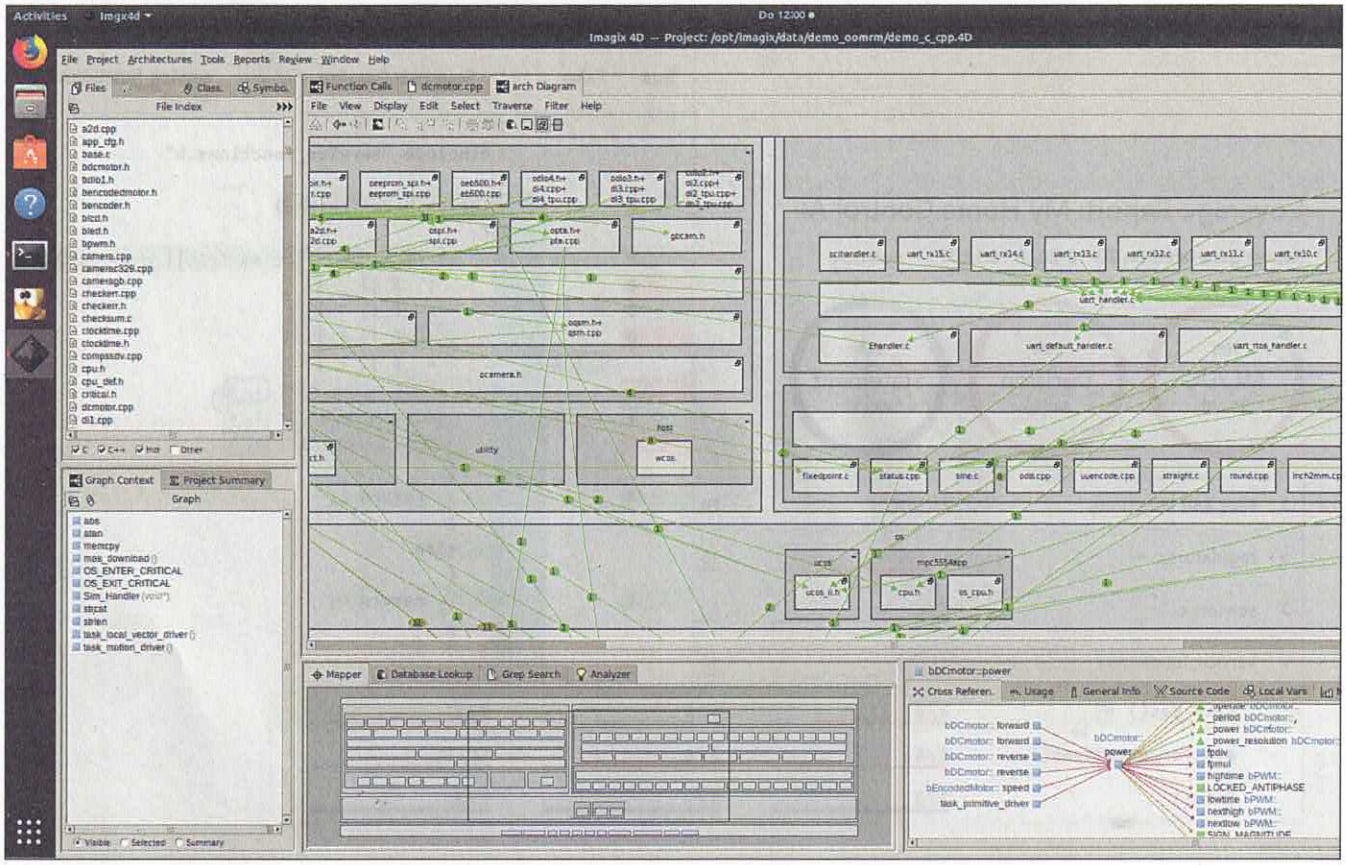
Code Coverage Tools wie Testwell CTC++ zeigen den Testfortschritt und geben im Quellcode an, wo noch Tests fehlen. (Bild: Verifysoft)

Die Medical Device Regulation (MDR), die EU-Verordnung für Hersteller von Medizinprodukten, schreibt vor, dass Hersteller die Sicherheit ihrer Produkte gewährleisten und die Qualitätssicherung über den gesamten Lebenszyklus des Produkts sicherstellen müssen. Auch die Norm IEC/ISO 62304, die die Anforderungen an den Lebenszyklus von Software in Medizinprodukten oder für Software als eigenständiges Medizinprodukt beschreibt, gibt bezogen auf die Entwicklung medizinischer Geräte einen risiko- und qualitätsgesteuerten Softwareentwicklungsprozess vor. Die Norm zeigt die Notwendigkeit von strengen Tests, Abnahmekriterien und Nachvollziehbarkeit auf. Hierzu müssen sowohl die funktionale Sicherheit (das richtige Funktionieren, Safety) als auch die Angriffssicherheit (der Schutz vor Angriffen von außen, Security) berücksichtigt werden.

Zwei Methoden für Softwaretests

Um beste Qualität zu gewährleisten, kommen in der Softwareentwicklung idealerweise zwei unterschiedliche Verfahren zum Einsatz: die statische Codeanalyse und das Testen zur Laufzeit, auch dynamische Analyse genannt. Da beide Methoden verfahrensbedingt jeweils nur einen Teil der vorhandenen Probleme aufdecken, müssen sie komplementär eingesetzt werden. Erst wenn sie gemeinsam genutzt werden, kann die Qualität der Software entscheidend gesteigert werden.

Durch richtige Planung und den Einsatz ausgereifter Test- und Analysewerkzeuge können nicht nur die geforderten Funktionalitäten korrekt umgesetzt werden, sondern auch viele typische Probleme vermieden werden. Dazu zählen etwa verzögerte Auslieferungen, ein hoher Zeit- und Kostenaufwand



Zusammenhänge werde auch in unbekannter Software durch Architekturanalysen (hier mit Imagix 4D) sichtbar. (Bild: Verifysoft)

für Fehlerkorrekturen sowie die damit verbundenen Budgetüberschreitungen.

Statische Codeanalyse findet Fehler früh

Software für Medizingeräte wird in der Regel hardwarenah programmiert. Daher kommen oft Sprachen wie C und C++ zum Einsatz, die hochperformanten und kompakten Code ermöglichen. Neben vielen Freiheiten bei der Programmierung lassen C und C++ jedoch leider auch viel Raum für das Schreiben von fehlerhaftem Quellcode.

Beispielsweise überprüft ein C/C++-Compiler nicht, ob angeforderter Speicher auch zugeteilt wurde oder zu kopierende Zeichenketten in die Zielarrays passen. Hierdurch besteht die Gefahr, dass Fehler wie Null-Pointer-Exceptions oder Buffer-Overflows auftreten. Nicht initialisierte Variablen können unbestimmtes und gefährliches Programmverhalten verursachen. Variablentypen, die sich in C und C++ relativ leicht verändern lassen, können zu impliziten Wertveränderungen führen.

Zudem wird die Überprüfung der korrekten Entwicklung schwieriger, wenn viele Entwickler gleichzeitig an unterschiedlichen, aber voneinander abhängigen Modulen arbeiten und niemand das »große Ganze« der Softwareentwicklung im Auge behält.

Tools für die statische Codeanalyse wie CodeSecure CodeSonar können hier eine entscheidende Hilfe sein. Ein Analyselauf erstellt Abbilder aller Abläufe und Datenzugriffe, die umfangreich ausgewertet werden. Basierend auf dem Quellcode werden Modelle der Datenströme und Zugriffe erzeugt und analysiert. Potenzielle Schwachstellen werden aufgezeigt und bei guten Analysetools mit einer ausführlichen Fehlerbeschreibung gemeldet, sodass diese schnell und früh im Entwicklungsprozess behoben werden können.

Rrichtlinien für saubere Programmierung

Aufgrund der Schwachstellen hardwarenaher Sprachen wie C und C++ wurden Programmierrichtlinien wie MISRA C und MISRA C++ aufgestellt. Diese beschränken die Flexibilität dieser Sprachen zugunsten größerer Sicherheit und beinhalten Vorgaben für zu nutzende Sprachelemente. Werkzeuge für die statische Codeanalyse überprüfen die Einhaltung solcher Regeln.

Statische Codeanalysetools tragen also einerseits zur Entwicklung von regelkonformer Software bei und decken andererseits zusätzlich Programmierfehler auf, die sich trotz der Beachtung von Coding-Standards in den Code »eingeschlichen« haben.

Durch den Einsatz statischer Codeanalysetools können viele Abnahmekriterien der ISO 62304 wie Daten- und Kontrollflussanalyse, Initialisierung von Variablen, Speichermanagement und Speicherüberläufe abgedeckt werden.

So empfiehlt etwa auch die US-amerikanische Arzneimittelbehörde FDA allen Unternehmen, die Software für medizinische Geräte entwickeln, den Einsatz von statischer Codeanalyse im Software Development Life Cycle (SDLC), um sichere und zuverlässige Software zu gewährleisten.

Dynamische Tests plus Code Coverage

Zusätzlich zur statischen Codeanalyse müssen dynamische Tests während des Ablaufs des Programms ausgeführt werden. Hiermit wird die funktionale Korrektheit des Systems nachgewiesen. Auch hierbei gilt: je früher, desto besser. Mit dem Testen zur Laufzeit sollte begonnen werden, sobald erste Codebestandteile lauffähig vorliegen. Je schwerwiegender die Folgen von Fehlern sein können, desto umfassender muss getestet werden. Zur Feststellung und

zum Nachweis des Testfortschritts werden Code Coverage Analyzer eingesetzt. Hierzu platzieren diese Tools an allen relevanten Stellen des Quellcodes Zähler, die protokollieren, ob bzw. wie oft der entsprechende Codeteil getestet wurde. Für den Test von Embedded-Software ist es wichtig, dass diese Zähler möglichst wenig Speicherplatz benötigen und die Performance nur minimal beeinflussen, um in zeitkritischer Software keine Fehlfunktionen zu generieren. Nach den Testläufen erzeugt das Code Coverage Tool eine Übersicht der Testabdeckung, die im Detail zeigt, welche Teile des Codes ausgeführt wurden und wo noch weiter getestet werden muss.

Bei sehr kritischer Software sollten alle Ausprägungen von Mehrfachbedingungen (Multiple Conditions) abgetestet werden. Obwohl Code Coverage nach IEC 62304 (noch) nicht verpflichtend ist, empfiehlt die Norm dennoch, die Code Coverage zu erhöhen. Etwas deutlicher wird das »Guidance Document« der FDA, die vorgibt, dass *»Maßnahmen wie [...] Test Coverage [...] genutzt werden sollen, um ein akzeptables Vertrauensniveau herzustellen, bevor das Produkt ausgeliefert wird. [...] Hierbei wird die Decision Coverage als minimale Coverage für die meisten Softwareprodukte angesehen, jedoch wird die Decision Coverage allein als unzureichend für High-Integrity-Anwendungen betrachtet.«*

Für hochkritische Software sollte daher die Modified Condition/Decision Coverage (MC/DC) erreicht werden, die beispielsweise durch die Norm ISO 26262 im Automotive-Bereich oder die DO178-C in der Luftfahrt vorgeschrieben ist.

Gute Coverage-Tools wie Testwell CTC++ unterstützen neben anderen Code-Coverage-Stufen auch dieses Coverage-Niveau und geben gezielte Hinweise darauf, warum bestimmte Funktionen nicht aufgerufen worden sind. Da das Tool während der Testausführung »passiv mitläuft«, macht das Werkzeug die Analyse der Code Coverage zu einer einfachen Tätigkeit. Die Analysen können im Rahmen von automatisierten Prozessen weiterverarbeitet werden und sind auch als verständliche HTML-Berichte abrufbar.

Statische Analysen und dynamische Tests

Beide Verfahren, die statische Analyse des Quellcodes und die dynamischen Tests zur Laufzeit mit Messung der Code-Coverage, ergänzen sich ideal und stellen sicher, dass nahezu alle Fehlerquellen aufgedeckt werden. Eingebunden in gängige Softwareentwicklungsumgebungen sind sie integraler Bestandteil des Softwareentwicklungsprozesses und unterstützen Programmierer, Qualitätsverantwortliche und Manager in allen Phasen der Softwareentwicklung.

Alter Code mit Problemen

Werden Prozesse implementiert, die einen frühzeitigen Einsatz von statischer Analyse und Tests zur Laufzeit vorsehen, sollten neue Produkte hinsichtlich der Qualitätssicherung in der Regel unproblematisch sein.

Schwieriger ist es meist bei altem Code, bei dem oft die Dokumentation fehlt und keiner mehr so richtig weiß, wie dieser aufgebaut ist. Oft sind über die Jahre immer wieder Änderungen und Anpassungen vorgenommen werden, die den Wartungs- und Modernisierungsaufwand extrem erhöhen. Da solche Software trotzdem häufig um neue, aktuelle Funktionalitäten erweitert werden muss, sind dabei meist bisher unentdeckte Fehler zu beseitigen. Oft beginnt dann eine detektivische Spurensuche in der alten Software, die Entwickler ohne geeignete Werkzeuge zur Verzweiflung bringen kann.

Hier können Tools wie beispielsweise Imagix 4D dabei helfen, die Beziehungen zwischen den unterschiedlichen Projektdateien aufzuzeigen und alle relevanten Informationen grafisch aufzubereiten. Somit steht eine Repräsentation des gesamten Projekts zur Verfügung, die alle Beziehungen in der jeweils benötigten Detailtiefe zeigt und es ermöglicht, den vorhandenen Code zu verstehen und seine Funktionalität nachzuvollziehen. Im nächsten Schritt kann der Code dann im Rahmen eines Refactorings bereinigt und in eine schlüssige Form gebracht werden. Zudem erlaubt das Tool, im Hinblick auf Zertifizierungen mit vertretbarem Aufwand eine umfassende Dokumentation zu erstellen.

Auf dieser Basis lässt sich die Anwendung dann mit neuen Funktionen versehen und das betagte Projekt kann mit aktuellen Ansätzen wie der agilen Entwicklung weiterentwickelt werden.

Geeignete MedTech-Werkzeuge

Die Qualitätssicherung ist in der Medizintechnik nicht neu, gewinnt aber durch aktuelle Technologien weiter an Gewicht. Ohne geeignete Werkzeuge sind die Analyse und der Test komplexer Anwendungen allerdings kaum zu leisten. Statische Codeanalyse und Laufzeittests mit Nachweis der Testabdeckung durch Coverage Tools unterstützen Softwareentwickler beim Schreiben qualitativ guter Software. Und auch für bestehenden, alten Code bieten Softwareanalysetools Hilfe. Von der guten Codequalität profitieren dann nicht nur die Patienten, sondern langfristig vor allem auch die Hersteller. uh