CogniMed

Gesellschaft für medizinische
und industrielle Meß- und
Automatisierungstechnik mbH

Fühl‘ die EN 62304

Eine Interpretation

1 Einleitung

Wer neu in der Entwicklung von Medizinprodukten angekommen ist, muss sich oft erst einmal abfinden mit der Menge an zu erfüllenden Normen und regulatorischen Anforderungen (obwohl sie im Vergleich zu anderen Branchen wie der Luftfahrt tatsächlich noch überschaubar ist). Manche Normenforderungen mögen als unnötig, überzogen oder auch als nicht ganz ernst zu nehmen wahrgenommen werden.

Als jemand, die schon seit einigen Jahrzehnten in der Softwareentwicklung für Medizinprodukte tätig ist, möchte ich am Beispiel der EN 62304 eine Lanze für das Verständnis dieser Norm brechen und meinen Zugang dazu mit Euch teilen. Im Idealfall ergibt sich daraus für Euch auch ein besseres Verständnis für Anforderungen anderer Normen. Ich bin keine Mitautorin der Norm, sondern Anwenderin, die nachfolgenden Ausführungen stellen meine persönliche Interpretation der EN 62304 dar.

2 Ein Negativbeispiel

Eines der berüchtigtsten Beispiele für Softwarefehler mit katastrophalen Konsequenzen ist der Fall des Therac-25, einem medizinischen Linearbeschleuniger für die Bestrahlungstherapie in den 90'er Jahren. Mehrere Patienten erlitten bei der Behandlung mit dem Therac-25 starke Verbrennungen und verstarben teilweise infolge einer Strahlungs-Überdosierung.

Damals befand sich die methodische Softwareentwicklung in den Kinderschuhen, die Vorgehensweise bei der Entwicklung der Software, die auf einem Vorgängermodell basierte, würde man aus heutiger Sicht wohl als Cowboy-Coding bezeichnen, da viele heute selbstverständliche Elemente eines systematischen Vorgehens nicht umgesetzt wurden.

Nancy G. Leveson erlangte in Fachkreisen Berühmtheit durch ihre akribische Untersuchung der Vorfälle und ihrer Analyse der Kernursachen. Sie publizierte den immer noch sehr lesenswerten Artikel: „Medical Devices: The Therac-25". Allen Einsteigern in dies Thema, die auch nur ein bisschen Interesse an Softwarequalität haben, empfehle ich den Artikel zur Nachtlektüre.

Es konnten im Verlauf der Untersuchung zwei Softwarefehler identifiziert werden [1]. Beide waren relativ trivial (eine Racing-Condition und ein Variablenüberlauf). Beide wären mit wenig Aufwand zu vermeiden gewesen und die Kernfrage, die sich sofort stellt, ist natürlich: *Warum konnten diese Fehler ihren Weg ins Feld finden?* Aufschluss gibt wiederum Frau Levesons Analyse, die u.a. folgende Aspekte aufzeigte [1]:

1. Die Software war ursprünglich für die Vorgängerversionen (Therac-6 und Therac-20) des Geräts entwickelt worden und wurde für den Therac-25 weiterentwickelt.
2. Zwischen den beiden Produktversionen wurde das Sicherheitskonzept umgestellt. Einige Risikominderungsmaßnahmen, die im Therac-20 in Hardware implementiert waren, wurden nun in Software umgesetzt
3. Die Software wurde von **einem** Softwareentwickler geschrieben und auch von ihm getestet. Dieser Softwareentwickler war nicht besonders gut vertraut mit der Software des Therac-6 und Therac-20.
4. Die Software war teilweise unnötig komplex.
5. Es gab keine systematisch dokumentierten Anforderungen, somit konnten auch keine Tests gegen die Anforderungen spezifiziert werden um zu sehen, ob die Maschine die Anforderungen komplett erfüllt.
6. Es fanden keine systematischen und dokumentierten Tests statt, sondern lediglich „Engineering-Tests" des Entwicklers
7. Die ursprüngliche Risikoanalyse enthielt keine wesentlichen Softwareaspekte (aus mehreren Gründen, einer davon, dass Software nicht verschleißt)
8. Eine Fehlermeldung, deren Beachtung Patientenschädigungen hätte verhindern können, lautete: "MALFUNCTION <n>", wobei <n> eine Zahl zwischen 1 und 64 war. Solche Fehlermeldungen traten häufiger auf und waren offenbar in der Gebrauchsanweisung nicht

erläutert. Für die Anwender war nicht ersichtlich, dass eine Patientenschädigung imminent war und sie konnten die wegen der „malfunction" ausgesetzte Anwendung bis zu fünf Mal neu starten.

Es gab weitere, wichtige Punkte, auf diese soll hier aber nicht eingegangen werden.

3 PDCA

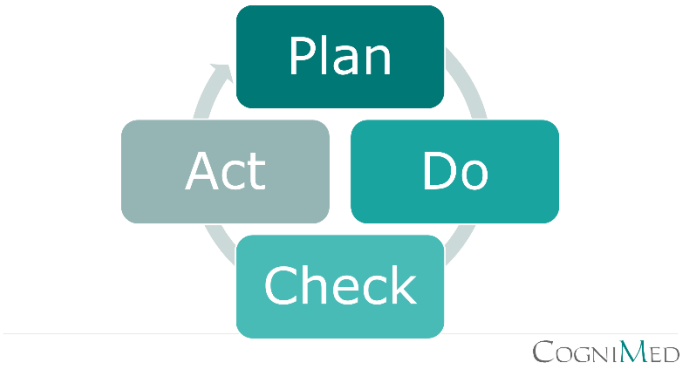

Abbildung 1 PDCA-Zyklus

Dieses gute alte Qualitäts-Credo ist so totgeritten, dass ich es schon fast gar nicht mehr erwähnen mag. Das macht es aber nicht weniger wichtig oder korrekt und es stellt auch eine der Säulen der EN 62304 dar. Der PDCA-Zyklus ist ein geschlossener Prozess-Kreislauf mit den folgenden Aktivitäten:

- (P)lan: Plane gut, was Du in diesem Prozess machen möchtest (wer, was, wann, womit…)
- (D)o: Setze Deinen Prozess wie geplant um
- (C)heck: Prüfe, ob Deine Prozessergebnisse ok sind. Wenn nicht,
- (A)ct: Ergreife Gegenmaßnahmen. Dazu kann auch gehören, den Prozess zu verbessern, damit der nächste Durchlauf besser wird (Vorbeugung).

Schon am Inhaltsverzeichnis der EN 62304 kann man erkennen, dass die Prozessvorgaben der Norm strikt dem PDCA-Schema folgen [2].

4 Spezielle Prozesse

Zunächst einmal scheint der folgende Abschnitt ein gedanklicher Bruch zu sein, doch bleibt bei mir, der Sinn erschließt sich gleich.

Die DIN EN ISO 9000:2015-11 definiert einen speziellen Prozess als *„Prozess, bei dem die Konformität […] des dabei erzeugten Ergebnisses nicht ohne weiteres oder in wirtschaftlicher Weise validiert werden kann."* [3]

Ein naheliegendes Beispiel für solch einen Prozess ist der Sterilisationsprozess für Operationsbesteck im Krankenhaus. Zwar kann nach erfolgter Sterilisation überprüft werden, ob das Besteck die Sterilitätsanforderungen erfüllt, jedoch würde diese Überprüfung die Sterilität zerstören.

Wie also kann man die Qualität der Prozessergebnisse für solche Prozesse sicherstellen? Das Standard-Vorgehen lautet:

1. Finde und beschreibe einen Prozess, der sicher das gewünschte Ergebnis erzielt. Sterilisiere 100.000 OP-Bestecke, Prüf' sie und weise so nach, dass sie nach dem Sterilisationsprozess mit vorgegebenen Temperaturen, Chemikalien, Zeiten, … tatsächlich steril sind.
2. Für jeden weiteren Durchlauf des Prozesses: Befolge sorgfältig die unter 1. etablierte Prozessvorschrift

Jede Softwareentwicklerin (m/w/d) weiß aus Erfahrung, dass das erfolgreiche Testen von Software – im Sinne von: keine Fehler gefunden – keine Fehlerfreiheit der Software vermuten lassen kann. (Der Umkehrschluss, dass eine Software, deren Test Fehler offenbart, vermutlich noch viele weitere Fehler enthält ist hingegen zulässig.) Software verschleißt nicht, das haben die Therac-Entwickler korrekt erkannt. Insofern sind Softwarefehler immer vorhanden – oder nicht, aber sie offenbaren sich leider nur unter bestimmten Umständen. Das Verhalten von Software ist abhängig von der Summe aller internen und externen Zustände und Ereignisse, und damit auch vom Faktor Zeit. Ob ein Ereignis jetzt oder eine Millisekunde später auftritt, kann ggf. ein komplett unterschiedliches (Fehler-)verhalten auslösen. Diese Erkenntnis steckt auch in der alten Weisheit, dass Software nicht „gut getestet" werden kann[1]. Daher kann die Softwareentwicklung durchaus als spezieller Prozess verstanden werden: Das Ergebnis des Softwareentwicklungsprozesses kann nicht ohne weiteres verifiziert werden.

Folgen wir dem Vorgehen für spezielle Prozesse, müssen wir also

1. einen Prozess etablieren und beschreiben, der eine gewünschte Softwarequalität sicherstellt.
2. Für jeden weiteren Durchlauf des Prozesses: sorgfältig den unter 1. etablierten Prozess befolgen

Und genau das ist der Grundgedanke der EN 62304 und anderer prozeduraler Normen.

5 EN 62304

Die EN 62304 verfolgt sowohl das PDCA-Prinzip, als auch den Ansatz spezieller Prozesse. Weitere Anforderungen können fast als eine direkte Antwort auf den Therac-Fall verstanden werden.

Einige Beispiele [2]:

1. Die Fähigkeit, Software zu erstellen, die die Anforderungen des Kunden und der anwendbaren Regularien erfüllt, muss nachgewiesen werden. Dies kann so verstanden werden, dass der Medizinproduktehersteller in der Lage sein soll, einen geeigneten Prozess zu definieren und diesen auch einzuhalten (Vgl. Spezielle Prozesse)
2. Zu Beginn der Entwicklung muss ein Softwareentwicklungsplan erstellt werden Der verwendete Entwicklungsprozess, Aktivitäten, Materialien, Methoden, … müssen festgelegt werden. (Vgl. Spezielle Prozesse und PDCA)
3. Anforderungen an die Medizinproduktesoftware müssen festgelegt werden (Vgl. Negativbeispiel, Punkt 5)
4. Diese Anforderungen müssen auch Anforderungen an die Benutzerschnittstelle enthalten (Vgl. Negativbeispiel, Punkt 8). Anforderungen an die Usability-Betrachtung von Medizinprodukten werden allerdings vorrangig durch die EN 62366 geregelt.
5. Eine Softwarearchitektur, ggf. mit detailliertem Design, muss aufgestellt werden (Vgl. Negativbeispiel Punkt 4)
6. Die Softwareentwicklungsergebnisse, auch Phasen- und Teilergebnisse müssen verifiziert werden (Vgl. Spezielle Prozesse, PDCA)
7. Die Software muss gegen die Anforderungen geprüft werden. Dabei muss nachgewiesen werden, dass die Umsetzung aller Anforderungen geprüft wurde (Coverage / Rückverfolgbarkeit). (Vgl. Negativbeispiel Punkt 5 und 6, PDCA)
8. Werden schon bestehende Softwareelemente verwendet, die vereinfacht gesagt nicht für den vorgesehenen Einsatzzweck entwickelt wurden bzw. für die die notwendige Dokumentation nicht vorliegt (sogenannte SOUP – Software of unknown Provenance), so muss dies sorgfältig analysiert werden: Welche Hardware und Umgebungsbedingungen setzt die SOUP-

[1] Selbst bei 100%-iger high-level Code Coverage kann ein Interrupt doch zu einem nicht im Test aufgetretenen Worst-Case-Moment eintreten.

Komponente voraus? Wie kann mit Fehlern in der SOUP-Komponente umgegangen werden? (Vgl. Negativbeispiel Punkt 1 und 2)

9. Der mögliche Beitrag von Software zu Gefährdungssituationen muss in der Risikoanalyse des Medizinprodukts betrachtet werden, Softwareentwicklungsprozess und Risikomanagementprozess sollen eng verzahnt sein (Vgl. Negativbeispiel Punkt 7)
10. Die personelle Unabhängigkeit wurde nicht vergessen, sie wird bereits in anderen Normen wie der ISO 13485 und IEC 60601 adressiert. (Vgl. Negativbeispiel Punkt 3)

Die EN 62304 ist zudem eine der wenigen Normen, bei denen der gesunde Menschenverstand auch in Bezug auf den Dokumentationsaufwand deutlich erkennbar ist. Im Detail sind die Aktivitäts- und Dokumentationsanforderungen an „risikoreiche" Software deutlich höher, als an „risikoarme" Software. Der hier salopp genannte „Risikoreichtum" wird anhand von Softwaresicherheitsklassen abgebildet, die auf dem Schadensrisiko für Personen fußen [3].

Mittlerweile haben die Autoren der EN 62304 auch erkannt, dass Software nicht immer im luftleeren Raum erstellt wird, sondern Medizinproduktesoftware schon seit vielen Jahren gut und unverändert im Einsatz sein kann, seit vor dem Gültigkeitsbeginn der EN 62304 und ihren Dokumentationsanforderungen. Um diese Software auch heute noch „legal" erweitern und verändern zu können, gibt es in der jüngsten Ausgabe DIN EN 62304:2016-10 einen Abschnitt, wie mit „älterer Software", das ist Medizinproduktesoftware, die schon länger legal im Markt ist, verfahren werden kann. Risikobasiert müssen dabei ggf. einige Dokumente nachträglich erzeugt werden [3].

6 Fazit

Die EN 62304 ist eine Norm, deren Anforderungen gut nachvollziehbar sind und die mit angemessenem Aufwand umsetzbar sind. Im Vergleich zum „Cowboy-Coding" ist der Dokumentationsaufwand höher und man könnte vermuten, dass Entwicklungen durch die über das reine Codieren hinaus geforderten Aktivitäten teurer werden. Allerdings hilft die strukturierte Vorgehensweise, Fehler in frühen Entwicklungsphasen zu erkennen, zu denen sie noch vergleichsweise kostengünstig behebbar sind. Insofern ist bestreitbar, ob der Entwicklungsaufwand am Ende wirklich höher ausfällt. Auf andere Vorteile wie das Einarbeiten neuer Mitarbeiter möchte ich hier nicht weiter eingehen.

Das Vorgehen nach EN 62304 ist jedenfalls ein wichtiger Beitrag dazu, dass ich als Medizinprodukte-Softwareentwicklerin nachts gut schlafen kann.

P. Margaritoff
QMB und Softwareentwicklerin bei der CogniMed GmbH

Literatur

[1] Leveson N.G. „Medical Devices: The Therac-25". The Therac-25 Accidents – Annex A aus Safeware: System Safety and Computers, Addison-Wesley, 1995

[2] DIN EN 62304:2016-10, Medizingeräte-Software - Software-Lebenszyklus-Prozesse (IEC 62304:2006 + A1:2015); Deutsche Fassung EN 62304:2006 + Cor.:2008 + A1:2015

[3] DIN EN ISO 9000:2015-11, Qualitätsmanagementsysteme - Grundlagen und Begriffe (ISO 9000:2015); Deutsche und Englische Fassung EN ISO 9000:2015