Software-Lebenszyklus für Medizinprodukte

Anforderungen an den Entwicklungsprozess

Die IEC 62304 kann als Sammlung bewährter Vorgehensweisen bei der Entwicklung medizinischer Software bezeichnet werden. Sie ist eine gute Grundlage für die Überprüfung des eigenen Software-Entwicklungsprozesses. Allerdings bleibt die Definition eines normenkonformen und effizienten Entwicklungsprozesses für jeden Hersteller von Software für Medizinprodukte eine wichtige und komplexe Aufgabe.

Von Matthias Hölzer-Klüpfel

Die Norm IEC 62304 steht kurz vor der Harmonisierung für die Medizinprodukte-Richtlinie der Europäischen Union und wird damit verpflichtend für die Hersteller von Medizinprodukten mit Software-Anteilen. Sie stellt detaillierte Forderungen an die Prozesse und Aktivitäten, die bei der Entwicklung dieser Software durchgeführt werden. Die IEC 62304:2006 [1] der International Electrotechnical Commission wurde als Nachfolgerin für die amerikanische Norm ANSI/AAMI SW68 [2] mit gleichlautendem Titel („Medical Device Software – Software Life Cycle Processes“) geschaffen, um der zunehmenden Bedeutung von Software in Medizinprodukten und den damit verbundenen Risiken Rechnung zu tragen. Die Norm folgt der Erkenntnis, dass Testen allein nicht ausreicht, um die Qualität medizinischer Software sicherzustellen. Dies ist in einer Empfehlung [3] schon seit längerem dokumentiert: „Die Konformitätsbewertung erfordert, dass für den Entwurf der Software ein Prozess befolgt wird, der auf dem Risikomanagement basiert und eine Entwicklungsmethode nutzt, die das Konzept des Software-Lebenszyklus beinhaltet.“ Während das Risikomanagement durch die Norm ISO 14971 [4] für Medizinprodukte geregelt ist, steht mit der IEC 62304 jetzt auch eine Regelung für den Software-Lebenszyklus zur Verfügung.

Als internationaler Standard ist die Norm IEC 62304 daher für alle Hersteller relevant, die Software in ihren Medizinprodukten einsetzen – einerlei, ob die Software eingebettet ist oder ob das Medizinprodukt aus einer eigenständigen Software besteht. Und gerade der letztgenannte Bereich wird sich in Zukunft stark ausweiten: Durch die Umsetzung der Novellierung der Medizinprodukte-Richtlinie der EU 2010 wird auch Software, die bisher nicht als Medizinprodukt klassifiziert wurde, unter die Richtlinie fallen, zum Beispiel Teile von Krankenhaus-Informationssystemen. Die IEC 62304 hat eine besondere Bedeutung in der EU: Nach der Harmonisierung wird im Zulassungsverfahren vermutet, dass die Hersteller die Anforderungen an die Software-Entwicklung aus der Medizinprodukte-Richtlinie einhalten, wenn sie die IEC 62304 umsetzen. Aber auch in den USA ist die Norm relevant: Ausgehend von der amerikanischen Norm SW68 hat sich die FDA an der Entwicklung der IEC 62304 beteiligt und empfiehlt sie den Herstellern. In vielen aktuellen Einreichungen an die FDA wird sie entsprechend referenziert, wenn auch die endgültige Stellungnahme der FDA zur Anwendung der IEC 62304 noch aussteht.

Für Hersteller, die Software in ihren Medizinprodukten einsetzen, ist daher die Umsetzung der IEC 62304 so gut wie unabdingbar. Im Folgenden wird betrachtet, welche Anforderungen die Norm an die Software-Entwicklungsprozesse spezifiziert. Dabei sind diese Anforderungen als Mindeststandards für einen Entwicklungsprozess zu verstehen, nicht als definitive Vorgabe einer Vorgehensweise.

Anwendungsbereiche der Norm

Die Norm beschreibt nicht alle Aktivitäten, die bei der Entwicklung medizinischer Software durchgeführt werden; sie beschränkt sich auf Kernbereiche des Software-Lebenszyklus, um einen Rahmen für die Prozesse bei den Herstellern vorzugeben. Konkret werden fünf Themenbereiche adressiert, im Sprachgebrauch der Norm als „Prozesse“ bezeichnet:

- Software-Entwicklung,
- Software-Wartung,
- Software-Risikomanagement,
- Software-Konfigurationsmanagement,
- Problemlösung für Software.

Wesentliche Bereiche, beispielsweise das Risikomanagement des Gesamtsystems oder das Qualitätsmanagement bei der Entwicklung, werden von der Norm durch normative Verweise geregelt. Die IEC 62304 fordert explizit, dass bei der Entwicklung ein Risikomanagement nach den Vorgaben der ISO 14971 durchgeführt wird, in das sich auch die Forderungen des Software-Risikomanagements einbinden. Ebenso wird ein Qualitätsmanagement gefordert, welches sicherstellt, dass alle Kundenanforderungen und alle regulatorischen Auflagen erfüllt werden. Dies kann, wie die Norm empfiehlt, durch die Einrichtung eines Qualitätsmanagement-Systems nach der ISO 13485 [5] sichergestellt werden.

Vorgaben der Norm

Die Vorgaben, die die Norm an die durchzuführenden Entwicklungsprozesse stellt, werden in einer Hierarchie von Prozessen, Aktivitäten und Aufgaben („Tasks“) dargestellt. Unter einer Aktivität wird eine Ansammlung von Aufgaben verstanden. In **Bild 1** sind die Aktivitäten der Software-Entwicklung dargestellt. Die Aufgaben schließlich beschreiben, wie eine Aktivität auszuführen ist.

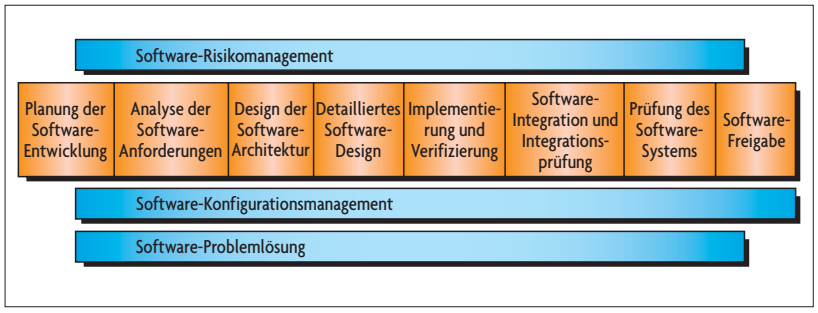

Bild 1. Aktivitäten und Prozesse im Software-Lebenszyklus.

Die Norm umfasst eine Vielzahl von Prozessen, Aktivitäten und Aufgaben. Nicht alle davon sind bei jeder Software-Entwicklung vorgeschrieben. Die Norm nennt zwei Gründe, die es rechtfertigen, bestimmte Aufgaben und Aktivitäten nicht durchzuführen: Zunächst gibt es Anforderungen, die als „soweit erforderlich“ markiert sind.

Sicherheitsklasse	Schweregrad eines möglichen Schadens
Klasse A	Keine Verletzung oder Schädigung der Gesundheit möglich
Klasse B	Keine schwere Verletzung möglich
Klasse C	Tod oder schwere Verletzung möglich

Sicherheitsklassen der IEC 62304

Diese können entfallen, wenn dokumentiert wird, warum sie nicht erforderlich sind. Dann sind bestimmte Anforderungen nur für einige Sicherheitsklassen relevant. Die Sicherheitsklassen sind ein neues Konzept, das die IEC 62304 einführt, um angemessene und angepasste Prozesse für ein breites Spektrum an Software in Medizinprodukten abzudecken.

Sicherheitsklassen

Die Definition der Sicherheitsklasse leitet sich aus der Definition des Risikos ab. In der ISO 14971 wird ein Risiko wie folgt definiert:

Risiko = Schweregrad × Auftretenswahrscheinlichkeit eines Schadens

Nun ist es bei Software nicht möglich, die Auftretenswahrscheinlichkeit eines Fehlers zu bestimmen. Software-Fehler treten nicht statistisch auf. Daher ist es eine vernünftige Annahme, die Wahrscheinlichkeit für das Auftreten eines Software-Fehlers mit 100 % zu bewerten. Das Risiko ist folglich nur noch von der Schwere des potentiellen Schadens abhängig. Deshalb definiert sich die Sicherheitsklasse in der IEC 62304 anhand des potentiellen Schadens, den eine Software bei Versagen anrichten kann (**Tabelle**).

In der Regel besteht eine Software aus mehreren Komponenten; die Sicherheitsklasse verschiedener Komponenten kann dabei durchaus unterschiedlich eingeschätzt werden.

Grundsätzlich geht die Norm davon aus, dass die Komponenten einer Software die Sicherheitsklasse der übergeordneten Komponenten erben.

Allerdings lässt sich im Rahmen einer definierten Software-Architektur begründen, warum einzelne Komponenten eine niedrigere Sicherheitsklasse aufweisen. Da der geforderte Aufwand bei der Entwicklung mit der Sicherheitsklasse ansteigt, ist es vorteilhaft, die Architektur des Software-Systems auch an den Sicherheitsklassen zu orientieren und beispielsweise Funktionen mit hoher und Funktionen mit niedriger Sicherheitsklasse in eigenen Komponenten zu kapseln. Allerdings muss dabei sichergestellt sein, dass die Risiken sich auch tatsächlich auf die entsprechenden Komponenten beschränken. In verteilten Systemen ist dies eventuell leichter, wenn beispielsweise die hoch sicherheitskritischen Funktionen auf einem eigenen Prozessor laufen.

In integrierten Systemen müssen andere Mechanismen greifen, um die Abschottung der Komponenten zu gewährleisten. **Bild 2** zeigt, wie sich die Sicherheitsklassen in einer theoretischen Architektur verteilen könnten: Einige Aktivitäten und Aufgaben sind nur in den Sicherheitsklassen B und/oder C notwendig. Die korrekte Bestimmung der Sicherheitsklasse erlaubt also, einige Arbeitsschritte auszulassen. Aber Achtung: Die Norm fordert nur einen Mindeststandard. Es gibt daneben noch andere Beweggründe, die es ratsam erscheinen lassen, beispielsweise eine Software-Architektur auch für Komponenten der Klasse A zu erstellen.

Software-Entwicklung

Der Prozess der Software-Entwicklung stellt den zentralen Prozess einer Norm dar, die sich mit dem Software-Lebenszyklus befasst. Die Software-Entwicklung macht hier mit acht Aktivitäten und insgesamt 52 Aufgaben den umfangreichsten Teil aus (Bild 1):

- Aktivität „Planung der Software-Entwicklung“: Hier wird festgelegt, wie die Entwicklung durchgeführt werden soll. Dies kann zum einen dadurch geschehen, dass ein Standardvorgehen referenziert und an die Bedürfnisse des Entwicklungsprojektes angepasst wird, oder aber durch Erstellung eines spezifischen Plans für das Projekt. Wichtig ist, dass der Detaillierungsgrad des Plans der Sicherheitsklasse der Software angemessen ist.
- Aktivität „Analyse der Software-Anforderungen“: Diese Aktivität beschreibt, wie die Software-Anforderungen aus den Anforderungen des Gesamtsystems abgeleitet werden.
- Aktivität „Design der Software-Architektur“: In dieser Aktivität, die nur für Software der Klassen B und C gefordert ist, werden die Anforderungen in eine Architektur überführt, die die Schnittstellen und besonderen Eigenschaften der Software-Komponenten festlegt. Dies schließt auch externe Software ein.
- Aktivität „Detailliertes Software-Design“: Hier wird, bei Komponenten der Klasse C, eine ausführliche Beschreibung der Software-Struktur und der Interfaces verlangt, die auf Übereinstimmung mit der Software-Architektur überprüft werden muss.
- Aktivität „Implementierung und Verifizierung der Software“: Die IEC 62304 schlägt vor, die Software entsprechend der Architektur und des eventuell erstellten Designs zu implementieren und zu verifizieren. Bei den Sicherheitsklassen B und C wird vorab definiert, welche Kriterien erfüllt sein müssen, damit die Implementierung einer Komponente akzeptiert werden kann.
- Aktivität „Software-Integration und Integrationsprüfung“: In den Klassen B und C ist es notwendig, die Software explizit zu integrieren und die Integration durch einen Test abzusichern. Eventuelle Fehler müssen mithilfe des Prozesses „Problemlösung für Software“ kontrolliert beseitigt werden, da sie unter Umständen Auswirkungen auf mehrere Bereiche des Software-Systems haben.
- Aktivität „Prüfung des Software-Systems“: Der Test der Gesamt-Software ist in den beiden höheren Sicherheitsklassen gefordert. Auch hier gilt, dass gefundene Fehler mit dem Problemlösungs-Prozess behandelt werden müssen.
- Aktivität „Software-Freigabe“: Ziel dieser abschließenden Aktivität ist es, die freigegebene Version zu dokumentieren und sicherzustellen, dass der Inhalt der Freigabe jederzeit reproduziert werden kann, beispielsweise wenn später Fehlermeldungen aus dem Betrieb des Medizinproduktes eingehen.

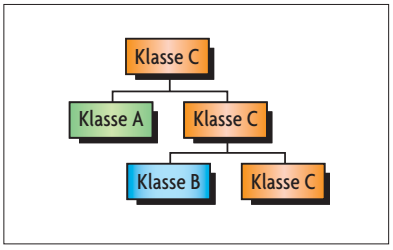

Bild 2. Sicherheitsklassen in einer definierten Software-Architektur.

Obwohl diese Aktivitäten die Vermutung nahelegen, dass das klassische V-Modell bei der Definition Pate gestanden hat, so ist doch festzuhalten, dass die IEC 62304 keine Vorgaben hinsichtlich des Vorgehensmodells macht. Der Anhang nennt explizit drei beispielhafte Modelle, mit denen die Anforderungen der Norm erfüllt werden können (**Bild 3**):

- Wasserfallmodell: Alle Aktivitäten werden in der angegebenen Reihenfolge nacheinander ausgeführt.
- Inkrementelles Modell: Nach der vollständigen Anforderungsanalyse wird das System in einer Reihe von Iterationen schrittweise aufgebaut.
- Evolutionäres Modell: Das System wird ebenfalls schrittweise aufgebaut, allerdings werden in jeder Iteration die Anforderungen überarbeitet und erweitert.

Die Norm lässt hier viele Freiheiten. Allerdings muss stets sichergestellt sein, dass die (impliziten) Abhängigkeiten zwischen den einzelnen Aktivitäten berücksichtigt und die Aufgaben in der korrekten Reihenfolge abgearbeitet werden. Beispielsweise darf die endgültige Klassifikation der Sicherheitsklasse erst nach der Fertigstellung der Risiko-Analyse erfolgen.

Eine Betrachtung der Aufgaben getrennt nach Sicherheitsklassen zeigt, dass für alle Klassen die grundlegenden Aspekte des Software-Engineering abgedeckt werden. In der Sicherheitsklasse B kommt die Forderung hinzu, die Risiko-Gegenmaßnahmen

in die Anforderungsliste aufzunehmen. Überraschenderweise werden sowohl eine Systemarchitektur als auch die Integration und der Integrationstest erst ab Klasse B verlangt. Auf den Software-Systemtest kann in Klasse A verzichtet werden. Allerdings sollte erwogen werden, ob hier die kurzfristige Zeitersparnis langfristig nicht doch durch eine ineffizientere Entwicklung negative Auswirkungen hat.

In der Sicherheitsklasse C kommt eine sehr komplexe Forderung ins Spiel: Im Entwicklungsplan müssen die Standards, Methoden und Werkzeuge, die bei der Software-Entwicklung eingesetzt werden, festgelegt und dokumentiert werden. Speziell bei den Methoden hat es sich bewährt, dabei auf die Norm ISO 61508 (besonders Teil 3) zurückzugreifen. Diese Norm enthält eine ganze Reihe von Empfehlungen über den Einsatz einzelner Verfahren bei Vorliegen einer bestimmten Sicherheitsstufe.

Software-Wartung

Die IEC 62304 beschreibt einen Prozess, der die Aktivitäten behandelt, die notwendig sind, um auf Fehler im Betrieb freigegebener Software angemessen zu reagieren. Treten Fehler auf, dann wird ein Hersteller möglichst schnell ein Update der Software entwickeln. Der Software-Wartungsprozess ähnelt daher sehr dem Software-Entwicklungsprozess, ist allerdings kleiner gehalten, um schnelle Reaktionen zu ermöglichen. Die Software-Wartung besteht aus drei Aktivitäten:

- Aktivität „Planung der Software-Pflege“: Diese Aktivität stellt verbindliche Regeln auf, wie Fehlermeldungen behandelt werden. Dazu gehört es, Fehlermeldungen entgegenzunehmen, auf neue Risiken zu prüfen, zu bewerten und zu beheben.
- Aktivität „Analyse von Problemen und Änderungen“: Dieser Abschnitt der Norm legt fest, wie Probleme festzustellen sind, wie diese dokumentiert werden und wie nach der Analyse bei Bedarf der Problemlösungs-Prozess genutzt wird, um eine fehlerbereinigte Version der Software zu erzeugen. Bei Bedarf müssen in dieser Aktivität auch Aufsichtsbehörden und Kunden informiert werden.

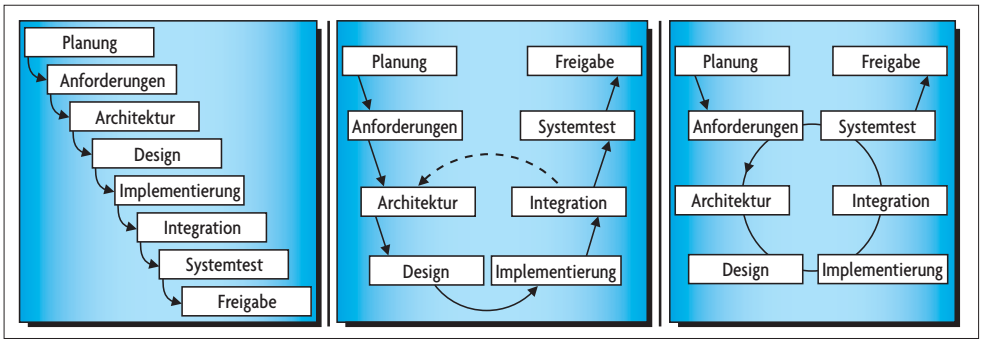

Bild 3. Exemplarische Vorgehensmodelle: Wasserfall, inkrementell, evolutionär.

- Aktivität „Implementierung der Änderungen“: Diese Aktivität ist ein Verweis auf den Software-Entwicklungsprozess, der befolgt werden muss, um den Fehler zu beseitigen.

Da Fehler in jeder Art von Software auftreten können, sind diese drei Aktivitäten für alle Sicherheitsklassen durchzuführen. Da im Notfall sehr schnell auf Fehler reagiert werden muss, ist darauf zu achten, dass die Entwicklungsaktivitäten zur Fehlerbeseitigung zügig ausgeführt werden können. Neben der notwendigen personellen Ausstattung empfiehlt es sich, möglichst viele Regressionstests so zu automatisieren, dass sie bei der Fehlerbehebung keine neuen Fehler einbauen. Regressionstests werden im Software-Entwicklungsprozess für die Klassen B und C gefordert, allerdings sind dort nicht notwendigerweise automatisierte Tests verlangt. Die Praxis zeigt jedoch, dass diese Art von Test gerade bei der Wartung sehr effektiv ist.

Software-Risikomanagement

Für das Software-Risikomanagement wird kein eigenständiger Prozess beschrieben, sondern es wird spezifiziert, wie das Risikomanagement für Medizinprodukte um Aspekte der Software erweitert wird. Das grundlegende Risikomanagement ist nach der ISO 14971 abzuwickeln, wie der normative Verweis in der 62304 verlangt. Hinzu kommt eine Reihe weiterer Aktivitäten:

- Aktivität „Analyse von Software, die zu Gefährdungssituationen beiträgt“: Hier werden die Gefährdungssituationen untersucht, die in der System-Risikoanalyse identifiziert wurden. Ziel ist es, festzustellen, welche Software-Bestandteile (selbst entwickelt oder von Drittherstellern) zu diesen Gefährdungen beitragen können. Für diese Bestandteile sind dann Risikokontroll-Maßnahmen erforderlich. In der Praxis sorgt die Forderung, diese Analyse auch auf Software von Drittherstellern auszudehnen, oft für Schwierigkeiten. Die Minimalforderung lautet hier: Die Liste der bekannten Fehler der Fremd-Software muss sorgfältig analysiert werden. Ratsam ist es, diese Fehlerlisten erstens vom Hersteller vertraglich anzufordern und zweitens mit der Software zu archivieren, zum Nachweis, welcher Stand der Fehlerliste Gegenstand der Risiko-Analyse war.
- Aktivität „Risikokontroll-Maßnahmen“: Hier wird ausdrücklich gefordert, Maßnahmen gegen das Eintreten der zuvor identifizierten Risiken zu ergreifen. Diese Gegenmaßnahmen können, wie in der ISO 14971 festgelegt, aus einem inhärent sicheren Design, einer Gegenmaßnahme oder einem Benutzerhinweis bestehen. Gegenmaß-

nahmen gegen ein Software-Risiko können durchaus auch durch eine Hardware-Komponente realisiert werden. Die Aktivitäten der Risiko-Analyse des Gesamtsystems und der Software lassen sich daher nicht wirklich trennen und sollten gemeinsam und iterativ durchgeführt werden. Iterativ deshalb, weil einige Risiken erst nach dem Architekturentwurf erkennbar werden und manche Gegenmaßnahme sich auf die Architektur auswirkt.

- Aktivität „Verifizierung der Risikokontroll-Maßnahmen“: Da die Risikokontroll-Maßnahmen letztendlich Teil der Rechtfertigung sind, warum ein Medizinprodukt eingesetzt werden darf, ist besondere Sorgfalt auf den Nachweis der Wirksamkeit dieser Maßnahmen zu verwenden. Dazu gehört auch die vollständige Nachverfolgbarkeit von der Gefährdung über die Software-Komponente bis hin zur Gegenmaßnahme und deren Test.
- Aktivität „Risikomanagement von Software-Änderungen“: Wenn die Software geändert wird, dann muss auch die Änderung dem Risikomanagement unterzogen werden. Damit soll sichergestellt werden, dass die Änderung keine neuen Risiken einbringt oder bekannte Risiken verändert. Diese Aktivität ist übrigens die einzige im Prozess „Software-Risikomanagement“, die für alle Sicherheitsklassen auszuführen ist; die anderen gelten nur für die Klassen B und C. Dadurch ist sichergestellt, dass eine Änderung auch daraufhin analysiert wird, ob sie die Einschätzung der Sicherheitsklasse verändert.

Software-Konfigurationsmanagement

Für das Software-Konfigurationsmanagement wird gefordert, dass alle Bestandteile der Software unter Versionskontrolle stehen, also jederzeit reproduzierbar sind. Die Pflicht zur Versionskontrolle erstreckt sich dabei auch auf die Software, die von Drittherstellern geliefert wird. In der Praxis wird diese oftmals nur als Binärcode verfügbar sein, der aber dennoch unter Versionskontrolle gestellt werden sollte.

Weiterhin wird geregelt, wie Änderungen an der Software durchgeführt werden. Grundlage für eine Änderung muss immer ein genehmigter Änderungsantrag sein. Dieser kann zum Beispiel aufgrund einer Fehlermeldung erstellt werden. Ist der Antrag genehmigt, dann wird die Änderung durchgeführt und verifiziert. Dieser Prozess muss nachvollziehbar dokumentiert sein.

Die Reproduktion einer Software-Version ist durch eine Versionskontrolle des Source-Code alleine kaum zu erreichen. Es existieren zu viele Abhängigkeiten beispielsweise zur Entwicklungsumgebung, etwa den eingesetzten Compilern und eventuell sogar zum Betriebssystem. Es hat sich bewährt, zusätzlich zum Source-Code auch die komplette Entwicklungsumgebung zu archivieren. Besonders einfach lässt sich das erreichen, indem die Entwicklungsumgebung in einer virtuellen Maschine installiert und das Image der virtuellen Maschine archiviert wird. Damit ist sichergestellt, dass sich ein Software Build auch in Zukunft wiederholen lässt.

Software-Problemlösung

Bei der Software-Problemlösung geht es um den Umgang mit Fehlern in der Software. In acht Aktivitäten wird darauf eingegangen, wie

- Fehlerberichte erstellt,
- die Probleme in der Software untersucht,
- Änderungen dokumentiert und verwaltet,
- Aufzeichnungen geführt und auf Trends untersucht werden sowie
- die Lösung verifiziert und dokumentiert wird.

Zu diesen Aktivitäten, die wie der gesamte Prozess auf alle Sicherheitsklassen anzuwenden sind, kommt noch eine dazu, die für Medizinprodukte besondere Bedeutung hat: Die Benachrichtigung der relevanten Stellen (Kunden, Aufsichtsbehörden, Fachhandel etc.) im Falle eines Fehlers.

Ausgangspunkt der Software-Problemlösung ist ein Fehlerbericht, der analysiert und über den entschieden werden muss. Dies ähnelt dem Änderungsantrag, der im Software-Wartungsprozess gefordert ist. Ratsamerweise behandelt man beide Dokumente in einem gemeinsamen Prozess. Dazu ist es allerdings notwendig, regelmäßige Bewertungsrunden abzuhalten, um kurzfristig reagieren zu können. *jw*

Literatur

[1] IEC 62304:2006: Medical Device Software – Software Life Cycle Processes; Deutsche Fassung EN 62304:2006: Medizingeräte-Software – Software-Lebenszyklus-Prozesse.

[2] ANSI/AAMI SW68: Medical device software – Software life cycle processes. Association for the Advancement of Medical Instrumentation / 05.06.2001, ISBN: 1-570-20161-7.

[3] NB-MED/2.2/Rec4: www.meddev.info/_documents/R2_2-4_rev5.pdf.

[4] Medizinprodukte – Anwendung des Risikomanagements auf Medizinprodukte (ISO 14971:2007); Deutsche Fassung EN ISO 14971:2007, Berichtigungen zu DIN EN ISO 14971:2007-07.

[5] Medizinprodukte – Qualitätsmanagementsysteme – Anforderungen für regulatorische Zwecke (ISO 13485:2003); Deutsche Fassung EN ISO 13485:2003+AC:2007.

[6] *Hölzer-Klüpfel, M.:* Software-Entwicklung für Medizinprodukte. *Elektronik* 2008, H. 12, S. 68ff.

Dipl.-Phys. Matthias Hölzer-Klüpfel

studierte Physik an der Universität Würzburg. Nach seinem Abschluss 1997 arbeitete er zunächst einige Jahre bei einem amerikanischen Linux-Distributor. Seit 2002 ist er bei der Method Park Software AG als Entwickler, Berater und Projektleiter tätig. Für Method Park führte er mehrere Medizintechnik-Projekte durch und war dabei sowohl bei Mittelständlern als auch in Großunternehmen eingebunden. Mit den Erfahrungen aus diesen Projekten übernahm er Anfang 2008 als Principal Consultant die Verantwortung für den Themenbereich Medizintechnik. Neben seinen beruflichen Tätigkeiten bereitet er sich derzeit auf seinen Abschluss im Master-Studiengang „IT im Gesundheitswesen“ vor.

Matthias.Hoelzer-Kluepfel@methodpark.de