Vom funktionierenden zum sicheren Produkt

ein Erfahrungsbericht zur Einführung von funktionaler Sicherheit

Andreas Stucki
Solcept AG
Wetzikon ZH, Schweiz
a.stucki@solcept.ch

***Abstract*—Dieser Beitrag zeigt den Weg von Solcept AG von der Entwicklung «normaler» embedded Systeme zu solchen mit funktionaler Sicherheit. Er zeigt unsere wichtigsten Erfahrungen auf.**

Keywords—funktionale Sicherheit, Prozesse, CMMI, DO-178, DO-254, ISO 26262, IEC 62304

I. EINFÜHRUNG

Funktionale Sicherheit wird immer häufiger eine Anforderung für embedded Systeme. Da diese Anforderung tiefgreifende Auswirkungen auf die Entwicklungsprojekte und die Entwicklungsorganisation hat, möchten wir unsere wichtigsten Erfahrungen als Entwicklungsdienstleister (17 Mitarbeiter) mit Ihnen teilen.

II. WIE SIND WIR ZU SAFETY GEKOMMEN?

A. Ausgangslage: Steuerungen (2003..jetzt)

Das einzige Ziel solcher Entwicklungen ist es, dass die Systemanforderungen erfüllt werden. Es geht in erster Linie um die Funktion und dann noch um nicht-funktionale Aspekte wie z.B. elektromagnetische Verträglichkeit, Betriebstemperaturbereiche.

B. CMMI-DEV Level 2 (2011)

Im Hinblick auf Effizienz und Qualität haben wir uns sieben Jahre nach Firmengründung einem Audit nach CMMI-DEV [1] unterzogen. Die Schlüsselthemen, an denen wir nach dem Audit gearbeitet haben, waren eine adäquate Umsetzung der Vorschläge, ein strukturiertes Projektmanagement und die Änderung des Fokus vom Produkt auf die Prozesse.

Die wichtigsten Änderungen waren die Einführung von Feedback-Schlaufen in Form von kontinuierlicher Prozessverbesserung und einer Peer-Review basierter Qualitätssicherung. Zusätzlich haben wir unsere Prozesse systematisiert.

Der wichtigste Entscheid war es, die Änderungen wirklich umzusetzen, d.h. die Zeit zur Verfügung zu stellen und die kontinuierliche Verbesserung Richtung Level 3 voranzutreiben.

C. DO-178/ DO-254 Avionik (2013)

In unserem ersten Avionik-Projekt, einem Füllstandssensor, kamen wir dann das erste Mal intensiv mit funktionaler Sicherheit in Kontakt. Der Fokus zeigte auf die Ingenieursarbeit, vor allem das Software Design («High Level Requirements») und die gesamte Hardware.

Geändert haben wir das Anforderungsmamagement, damit es die drei Ebenen für Avionik beinhaltet (System, High Level, Low Level) inkl. deren Traceability. Neu war auch, dass die Luftfahrbehörde in der Entwicklung mitredete.

Der wichtigste Entscheid war es, die Prozesse von Anfang an so zu gestalten, dass wir medizinische, industrielle und automotive Projekte mit funktionalen Sicherheit durchführen können.

D. IEC 62304 Medizintechnik (2014)

Dieser Entscheid fiel uns umso leichter, als wir fast gleichzeitig mit einer Software für die Signalverarbeitung im Medizinalbereich konfrontiert waren. Die Schlüsselthemen des Projektes, welches auf bestehenden Softwaren basierte, waren Design und statische Codeanalyse.

Änderungen an den Prozessen gab es fast keine, da diese parallel mit der Luftfahrt angepasst wurden. Wegen der Natur des Projektes (Re-Engineering) wurde entschieden, die Prozesse ad-hoc im Projekt anzupassen.

E. CMMI-DEV Level 3 (2016)

Nun wagten wir uns an Level 3, wo der Fokus mehr auf der technischen Umsetzung und der Organisation, d.h. den Prozessen der gesamten Entwicklung lag.

Die resultierenden Änderungen waren eher klein, da wir ja seit 2011 an Level 3 arbeiteten. Verbessert haben wir hauptsächlich die Qualitätssicherung und die Schulung der Mitarbeiter.

F. ISO 26262 Automotive (2016)

Ein Systemprojekt für ein Sensor-/ Kommunikationssystem lenkte den Fokus auf die Low-Level Softwareentwicklung und die Sicherheitsanalysen aller Aspekte.

Änderungen betrafen vor allem die Pläne, die bestehenden Vorlagen mussten zu einem Safety Plan/ Safety Case angepasst werden und wir mussten die Zusammenarbeit im Projekt mit einem Development Interface Agreement regeln (s.u.). Zusätzlich entstanden neue Rollen, wie z.B. der Safetymanager.

Der wichtigste Entscheid war der Einbezug eines externen Partners in die Qualitätssicherung (die «Confirmation Measures»). Einerseits ist so seine Unabhängigkeit sichergestellt und im Konfliktfall ein qualifizierter Ansprechpartner vorhanden.

III. WAS HABEN WIR GELERNT?

In den sieben Jahren haben wir einige Anpassungen an unserer Arbeitsweise vorgenommen, gesehen was funktioniert und was für funktionale Sicherheit wichtig ist.

A. Was haben wir generell gelernt?

Wir haben gesehen, dass es extrem wichtig ist, dass alle Entscheidungsträger in einem Projekt der funktionalen Sicherheit sich darüber klar sein müssen, dass funktionale Sicherheit nicht einfach ein Feature ist wie eine weitere Schnittstelle. Obschon es meist nur um eine Zeile in den Spezifikationen geht, so ändert diese Zeile das ganze Vorgehen und die ganze Entwicklung. Und zwar so, dass der Aufwand und die Entwicklungsdauer ansteigen.

Um wieviel diese Kosten ansteigen, dazu werden manchmal Zahlen angegeben. Diese Angaben beruhen meist auf «normalen» Projekten nach CMMI Level 3 (z.B. «QM» oder «DAL-E» für Automobil bzw. Avionik). Die gelebten Qualitätsstandards in vielen Industrien sind tiefer, das macht den Unterschied von «normal» zu funktionaler Sicherheit noch grösser.

Und das Gemeine ist, in der funktionalen Sicherheit gibt es keine Abkürzungen. Auch wenn der Produktmanager findet, es sei nun Messe, er wolle das Produkt verkaufen, ist das Produkt erst fertig, wenn der Safety Case respektive das Accomplishment Summary unterschrieben ist.

B. Was haben wir bezüglich Fähigkeiten gelernt?

Eine wichtige neue Fähigkeit ist das Anforderungsmanagement. Alle am Projekte beteiligten Ingenieure müssen Anforderungen und Designs so beschreiben lernen, dass diese genau und verständlich sind. Vor allem aber müssen sie die richtige Abstraktionsebene finden. Eine Anforderungs-Architektur wird wichtig, d.h. wie die Anforderungen für die verschiedenen Teile des Systems aufgebaut und für die Traceability verlinkt werden. Es muss definiert werden, wie mit Schnittstellen, Hardware-Description-Language (HDL) Code etc. umgegangen wird, vor allem wie effizient damit umgegangen wird.

Änderungsmanagement wird auch sehr viel prominenter. Alle wichtigen Artefakte unterstehen nach der ersten Freigabe einem rigorosen Prozess, der mittels z.B. eines genau definierten Workflow und eines Gremiums (Change Control Board) verhindert, dass unsichere Änderungen stattfinden. Hier hat sich eine Dokumentation des Workflows basierend auf einem Issue-Tracker (Jira) bewährt, da so der Aufwand für die Verwaltung minimiert werden konnte. Die Basis für die Freigaben ist der Dokumentations-, Review- und Releaseplan, hier hat sich eine frühe, detaillierte Planung bewährt. Da diese Planung stark von kommerziellen Abmachungen und dem Plan der Gesamtintegration abhängt, ist es wichtig das Produktmanagement früh in die Pflicht zu nehmen.

Und natürlich muss die Fähigkeit zu Sicherheitsanalysen aufgebaut werden. Am Anfang macht es Sinn, hier externe Moderatoren beizuziehen. Effizienter war es dann für uns, mithilfe guter Werkzeuge FMEA, FTA etc. für Systeme, Soft- und Hardware selbst erzeugen zu können. Für Elektronik ist die Produkte-FMEA meist ziemlich offensichtlich, sinnvolle Design-FMEA für Software, um systematische Fehler zu finden braucht etwas mehr Erfahrung.

Last but not least werden die Designfähigkeiten der Ingenieure sehr wichtig. Da wir keine dedizierten Codierer und Tester haben, ist fast jeder Ingenieur auf seiner Ebene mit Design konfrontiert. Die dazugehörige Mentalität und der richtige Abstraktionsgrad liessen sich erst über mehrere Review-Zyklen und viele Diskussionen in der Organisation erreichen.

C. Was haben wir bezüglich Werkzeugen gelernt?

A fool with a tool is still a fool... (R. Weinstein): Die Werkzeuge sind nicht immer von den Fähigkeiten zu trennen.

Werkzeuge sind für die Entwicklung funktionaler Sicherheit wichtig, da sich immerhin einige Qualitätssicherungsaufgaben automatisieren lassen. Wer auf Open-Source gesetzt hat, wird umdenken müssen, da meist nur kommerzielle Werkzeuge mit sinnvollem Aufwand qualifizierbar sind. Und qualifizieren muss man alle Werkzeuge, welche Fehler im fertigen Produkt erzeugen könnten, die nicht detektiert werden.

Da Anforderungen so wichtig sind, sind die Werkzeuge für Anforderungsmanagement zentral. Hier haben wir gesehen, dass die Industriestandards nicht immer die beste Lösung sind. Die Werkzeuge sollten den ganzen Lebenszyklus der Anforderungen effizient abdecken: Schreiben, Kommunikation mit Kunden, Review, Änderung, Coverage und Traceability. Viele Werkzeuge können zwar Coverage und Traceability sehr gut, der Reviewer sieht sich aber einem ungeordneten Anforderungshaufen gegenüber, der nur schwer zu erfassen ist. Eine Dokument-Sicht mit einem roten Faden vereinfacht das Verständnis für Autor und Leser und senkt die Fehlerrate.

Konfigurationsmanagement ist ein weiterer Pfeiler, der eines guten Werkzeuges bedarf. Gemäss unserer Erfahrung aber vor allem guter Prozesse, welche zu einer effizienten Nutzung des Tools führen. Bewährt hat sich, alle unter Konfigurationsmanagement stehenden Artefakte (Dokumente, Code, Schemas…) im selben Programm zu verwalten, man spart sich dann die Synchronisation mehrere Repositories.

D. Was haben wir bezüglich Management gelernt?

Ohne dass das Management hinter dem Projekt steht und bereit ist, die nötigen Ressourcen bereitzustellen, ist die Einführung von Entwicklung für funktionale Sicherheit zum scheitern verurteilt. Prozesse müssen entwickelt und gewartet werden, in Schulung muss investiert werden und die Entwicklungen kosten mehr. Wichtig, vor allem wenn es mal im Projekt nicht so gut läuft, ist das Mindset: funktionale Sicherheit ist getrieben von Haftung und Qualität, nicht davon, dass «es ja schon funktioniert, was wollt ihr noch…»

Implizit oder explizit fordern die Standards alle eine Sicherheitskultur. Auch diese wird von der Geschäftsleitung getrieben. Sobald der Entscheid getroffen wurde, in das Safety-Projekt zu investieren, muss es klar sein, dass Sicherheit vor

Kosten geht. Dennoch muss Effizienz auch ein Thema bleiben: es gilt also, die Gratwanderung zwischen Sicherheit und Aufwand erfolgreich zu moderieren und vorzuleben.

Der Beizug von externen Experten hat sich auch sehr bewährt, einerseits für Gap-Analysen, andererseits als Qualitätssicherung. Die Gap-Analysen der bestehenden Prozesse zur Norm sind nötig, weil die Normen nicht immer klar sind und auch um einen sinnvollen Erfüllungsgrad der Normen zu erreichen, ohne unnötiges Over-Engineering. Einen Teil der Qualitätssicherung auszulagern (wo nicht schon ein Notified Body verlangt ist), macht Sinn, da so die Unabhängigkeit gewährleistet ist und auch falls einmal Haftungsfragen auftauchen.

Ein Management-Werkzeug, welches wir nicht nur in Sicherheitsprojekten brauchen, ist das Development Interface Agreement, siehe Tabelle 1 unten für den Ausschnitt aus einem Beispiel. Eine solche Vereinbarung zeigt auf, wer im Projekt welche Verantwortung hat und kann vielen Missverständnisse und Kommunikationsprobleme vorbeugen.

Ich hoffe, dieser Beitrag konnte einige Fragen beantworten, wenn sie weitere haben, bitte kontaktieren Sie mich an den oben angegebenen Koordinaten.

[1] CMMI Product Team, CMMI for Development, Version 1.3, SEI, Caregit Mellon, 2010

TABELLE 1 DEVELOPMENT INTERFACE AGREEMENT

Aufgabe	**Ergebnis**	**Format/ Tool**	**Verantwortlichkeiten Partei 1**	**Verantwortlichkeiten Partei 1**	**Verantwortlichkeiten Partei 1**
System Definition	System Requirement Specification (“Lastenheft”)	PDF/ ODF	Consult	Review & Release	Responsible
System Design	System Specification (“Pflichtenheft”) System Architecture	PDF/ ODF	Uninvolved	Audit/ Inform	Responsible