| Datum: Stand 11.01.2019 | Autor: Siro |
|---|---|
| Thema: *Gerätename* Software Entwicklung<br>Risikoerfassung: RE07 Integrations/UnitTest | |

## Übersicht

| | |
|---|---|
| **Projektbezeichnung** | *Gerätename* |
| **Projekt ID** | 20 |
| **Produktname** | *Gerätename* |
| **Softwarename** | |
| **Software Versionsnummer** | ALLE Versionen |

## Prüfung & Freigabe

| | **Vor- und Nachname** | **Datum** | **Unterschrift** |
|---|---|---|---|
| **Autor:** | Siro | *11.01.2019* | |
| **Prüfer:** | Siro | *11.01.2019* | |
| **Freigabe:** | Siro | *11.01.2019* | |

## Änderungshistorie

| **Revision:** | **Autor:** | **Datum:** | **Änderungsinhalt:** |
|---|---|---|---|
| **01** | Siro | *11.01.2019* | Ersterstellung |

## Inhalt



## Einleitung:

Sinn und Zweck dieses Dokuments ist es, das Verhalten bzw. die Risiken aufzuzeigen welche durch Integrations und Unittest enstehen können, obwohl diese eigentlich das Gegenteil bewirken sollen. Erstaunlicherweise ist nirgends über derartige Problematiken etwas zu lesen.

## Risikoerfassung

Integrationstest sowie Unittest sind genereller Bestandteil einer Softwaredokumentation. Während dieser Tests sind jedoch Problematiken aufgetaucht, die ein Nichtfunktionieren der Software zur Folge hatten, bzw. Fehlverhalten sogar richtig stellen konnte, welches ohne diesen Test nicht gegeben war. Dieses recht komplizierte und problematische Thema soll hier kurz erläutert werden.

Wenn eine Software oder eine Softwareeinheit fertiggestellt wird, sollten entsprechende Tests ausgeführt werden, welche diesen Teil oder auch das gesamte System validieren bzw. verifizieren. Die meisten Tests können jedoch nur durchgeführt werden, wenn an dem bestehenden Source Code **Änderungen vorgenommen** werden und genau hier treten die unvorhersehbaren Probleme auf. Man testet nicht mehr die "originale" Software, sondern die durch Tests veränderte Software. Es reicht jedoch oftmals ein einzelner Befehl, ohne jegliche Bedeutung, zum Beispiel ein "NOP" No Operation um eine nicht ordnungsgemäß funktionierende Software plötzlich in eine Funktionierende zu verwandeln und umgekehrt. Damit können jegliche, auch nur die kleinste Änderung an der Sofware unvorhersehbare Folgen haben.

Dazu gibt es folgende, aufgetretene Beispiele:

Eine Timer-Inerruptfunktion schien nicht richtig zu funktionieren. Dazu wurde für den Integrationstest ein Testbit gesetzt um das Verhalten mit einem Oszilloskop zu überprüfen. Das Oszilloskop zeigte aber eine einwandfreie Funktion an. Der Integrationstest zeigte, dass die Timerfunktion einwandfrei arbeitet. Nun wurde die Testzeile wieder entfernt und plötzlich ging die Software nicht mehr richtig. Die einzelne Zeile für das Erzeugen des Testbits entschied tatsächlich darüber ob die Software richtig funktioniert oder nicht. Ohne diese Testzeile lief das Programm falsch. Genau hier lässt sich erkennen, das ein Integrationstest Fehler beseitigten kann, welche später jedoch in der Software bestehen bleiben. Dies stellt ein SEHR" hohes Risiko dar, zumal der Intergationstest in diesem speziellen Fall eine fatale Falschaussage macht.
Nach längerer Ursachenforschung stellte sich heraus, dass dieses Problem sogar abhängig von den Compiler Optimierungen war. Die wirkliche Ursache lag aber an der internen Struktur des Controllers Kerns Cortex M3. Es muss lediglich eine Zeile vorher das Interrupt Bit gelöscht werden, niemals direkt am Ende der Interrupt Funktion. Dazu reicht ein simples NOP. Sinnvollerweise sollte man dennoch einen Befehl "DSB" Data Synchronisation Barier einfügen, denn durch die Schreibe Lesepuffer kommt

das Bit nicht sofort dort an wo es soll. Es wurde daraufhin beschlossen die Software generell ohne sowie mit höchsten Optimierungsstufen zu testen. Dadurch konnten im Voraus oft schon nicht erkannte Probleme beseitigt werden.

Ein ähnliches Problem trat auf, duch einen Unittest. Um die Funktionalität zu verifizieren wurde wieder ein Testcode eingefügt und die Software verhielt sich einwandfrei. Nach dem Entfernen des Testcodes traten jedoch unerklärliche Fehler auf. Es stelle sich heraus, dass durch die Verschiebung des Programmcodes innerhalb des Speichers (bedingt durch das NOP) ein unterschiedliches Verhalten auftrat. Auch hier hat der Unittest einen negativen bzw. nicht entdeckten Fehler korrigiert, zumindest das Verhalten geändert.

Das nächste Problem besteht darin, das jegliche Änderungen nach dem Unittest wieder rückgängig gemacht werden müssen. Das Risiko ist sehr hoch, dass dadurch nicht alle zusätzlichen Codezeilen bzw. versehentlich sogar falsche Codezeilen mit gelöscht werden. So wird durch rückgängig machen der speziellen Codeänderungen die Software plötzlich unbrauchbar.

Das Risiko von Falschaussagen der Integrations/Unittest scheint recht hoch,
man darf sich leider NICHT auf die Testresultate verlassen.

## Gesamtbewertung:

Der Integrations bzw. Unittest kann Fehler aufdecken jedoch auch Fehler als PASS ausweisen oder sogar neue Fehler produzieren.

Ein Fehlverhalten aufgrund von Softwareproblemen kann fatale Auswirkungen auf die Patientensicherheit haben.

**Eine zusätzliche, zweite Sicherheit ist zwingend erforderlich.**