MICRO CONSULT

Software-Entwicklung neu gedacht

Teil 2: Durch umfassende Software Requirements typische Fehler vermeiden

In den Software-Anforderungen beschreiben Entwickler den Zweck und die Absicht eines Softwaresystems sowie dessen (externes) Verhalten. Welche Erwartungen haben Nutzer an das Softwareprodukt, und wie benutzerfreundlich ist es? Wie übersichtlich ist der Programmaufbau, wie strukturiert die Programmierung und wie verständlich die Dokumentation? Mithilfe dieser und anderer Fragen lassen sich viele Fehler vermeiden.

Etwa 50% der typischen Fehler in der Software eines Systems basieren auf fehlenden oder fehlerhaften Anforderungen (Requirements). Am Ende des "Requirements Engineering" stehen die Entwicklung und Pflege eines aussagekräftigen System Requirements Specification Dokuments, das die folgenden Anforderungen ausführt:

- **Geschäftsanforderungen:** Hier definieren das Management und das Marketing die Ziele des Produkts.
- **Benutzeranforderungen:** Anforderungen an das Softwaresystem aus der Sicht der Benutzer
- **Funktionale Anforderungen** definieren das Verhalten des Softwaresystems so, dass die Benutzeranforderungen erfüllt werden können.
- **Projektanforderungen:** Was muss alles erfüllt sein, damit das Softwareprojekt die funktionalen Anforderungen erfolgreich umsetzen kann? Dieser Aspekt zielt auf das Lifecycle Management ab und beinhaltet die Dokumentation des Softwaresystems sowie notwendige rechtliche Anforderungen. Darunter fallen unter anderem Lizenzen der eingesetzten Softwaretools und Hardwarekomponenten, die Berücksichtigung von Urheber-/ Markenrechten und Patenten sowie Qualitätsanforderungen an den Service.

Insbesondere bei den funktionalen Anforderungen kommen die Merkmale der Softwarequalität ins Spiel. An dieser Stelle lohnen sich umfangreiche Analysen zu den Aspekten des Einsatzes von Multicore, Safety und Security im Projekt.

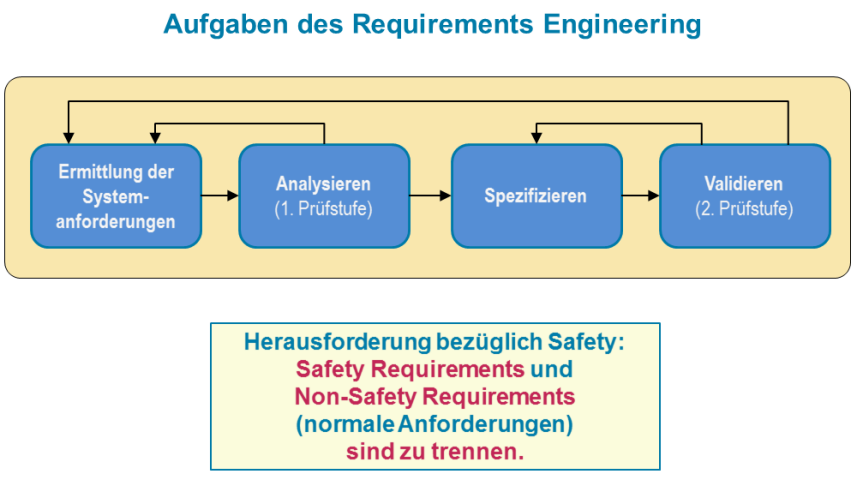

Bild 1: Aufgaben des Requirements Engineering

Das Requirements Engineering betrachtet außerdem folgende Anforderungen an die Software umfänglich:

- **Standards** und Normen, zum Beispiel ISO 26262 (ISO-Norm für sicherheitsrelevante elektrische/ elektronische Systeme in KFZ) und IEC 61508 (Funktionale Sicherheit von elektrischen/ elektronischen/ programmierbaren Steuerungssystemen)
- Welche **Software-Module** bzw. Software-Stacks werden eingesetzt?
- Welche **Programmierschnittstellen** werden verwendet?
- Welche **Qualitätssicherung** soll zum Einsatz kommen?
- Anlegen eines **Regelwerks zur Programmentwicklung** (Clean Code, Solid-Prinzip etc.), statische Codeanalyse-Methoden, Unit-Tests, Systemtests

Sind alle diese Requirements erfüllt, analysiert man die verwendeten Datentypen/ -strukturen und Zugriffsmethoden (wie Pointer), die Lebensdauer, eingesetzte Speicher sowie konkurrierende Speicherzugriffe auf diese Anforderungen hin. Eine Prüfung der Integrität der eingesetzten Tools, Dateien und Bibliotheken (insbesondere Versionskontrolle wie CVS) gehört ebenso dazu.

Weiterhin stellt man sich die Frage: Welche Daten-Sicherungsmethoden und Fehler-Reaktionen sollen zum Einsatz kommen? Hier kommt insbesondere aus der Sicht der Safety die **FMEA** (**F**ailure **M**ode and **E**ffects **A**nalysis) ins Spiel:

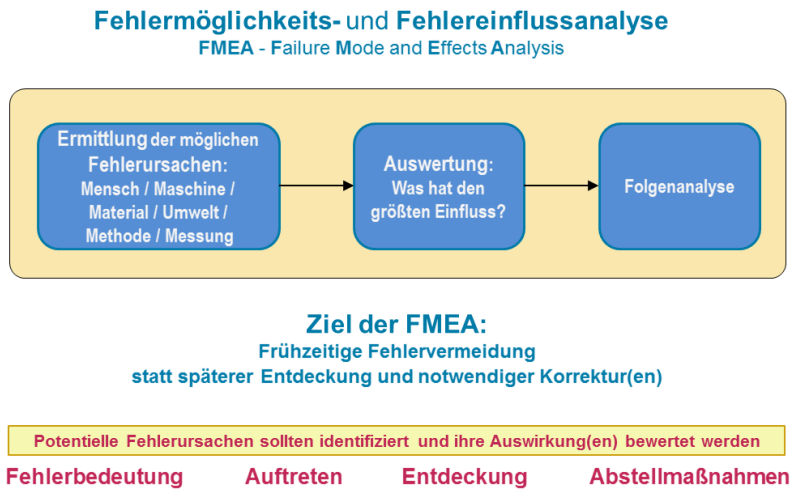

Bild 2: FMEA – Failure Mode and Effects Analysis

Werden alle Realzeit-Anforderungen erfüllt?

In diesem Kontext spielen insbesondere die Anforderungen an die System-Perfomance eine Rolle. Werden alle Aufgaben im erwarteten und auch tolerierbaren Zeitfenster erfüllt? Das gilt insbesondere dann, wenn es um Antwortzeiten von Tasks und Funktionen, Datendurchsatz und die rechtzeitige Verfügbarkeit von Ressourcen geht. Klare Anforderungen sind aus der Sicht des statischen und dynamischen Verhaltens zu definieren.

Wie steht es um das Thema System-Flexibilität?

Skalierbarkeit, Wartbarkeit, Erweiterbarkeit, Wiederverwendbarkeit sowie der Einsatz im internationalen Umfeld des Systems stehen ebenfalls im Rampenlicht der Anforderungsanalyse.

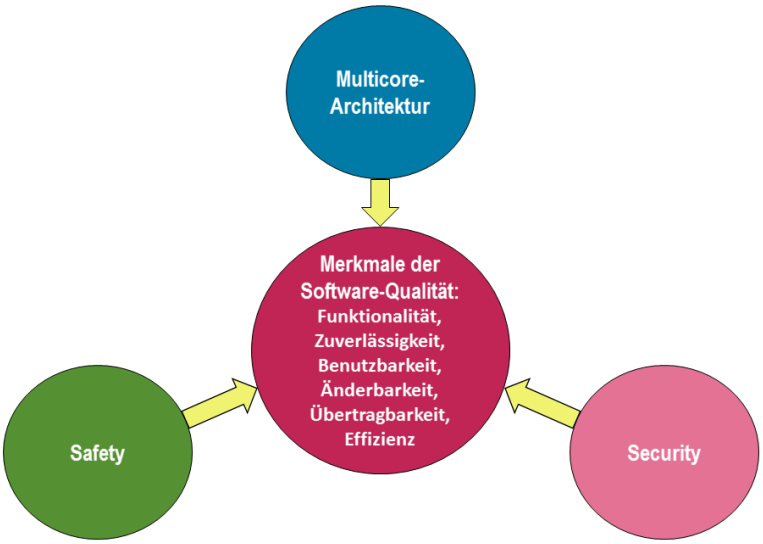

Bild 3: Multicore, Safety und Security und neue Herausforderungen

Was ist beim Einsatz von Multicore im Kontext von Safety und Security speziell an zusätzlichen Anforderungen zu betrachten und spezifizieren?

- Art und Zugriffsrechte bei der Verwendung Core-privater und globaler Speicher (RAM, ROM, Flash etc.)
- In Multicore-Architekturen betrachtet man die Spezifikation der Steuerung und der Synchronisation paralleler Abarbeitung von Funktionen und Prozessen sowie die Spezifikation der Anforderungen für das Reset- und Boot-Verhalten (insbesondere bei Multicore).
- Adäquater und individueller Schutz und Zuteilungsmechanismen von Speicher-Ressourcen mit genauem Blick auf Safety-relevante Funktionen, Prozesse und verwendete Peripherien aus Sicht der einzelnen zugreifenden Cores
- Inter-Core-Kommunikation und -Synchronisation
- Spezifikation von globalen Zeitquellen und Synchronisationsmechanismen (Time Synchronization Mechanism)
- Anforderungen an Betriebssysteme wie Multicore RTOS (Real-Time Operating System), insbesondere bei Anwendung von verschiedenen Safety-Klassen und Security
- Betrachtung und Spezifikation von dynamischen Abläufen, Funktionen und Prozessen
- Für die Anwendung von Stacks und Bibliotheken müssen der Aufbau sowie die Anwendungs- und Testvorgaben spezifiziert werden, insbesondere mit Blick auf Multicore, Safety und Security.
- Spezifikation der Schutzmechanismen von Security-Funktionen, Prozessen und Abläufen (z.B. Kommunikation), Ressourcen (z.B. Speicher) durch Firewalls, kryptografische Sicherung von Speicherinhalten und der Datenkommunikation beim Einsatz von High-Security-Modulen (HSM), außerdem Spezifikation der benötigten bzw. geforderten Kryptografie-Methoden
- Spezifikation des Kommunikationsmanagements – wer darf mit wem und wer muss mit wem kommunizieren? Z.B. beim Einsatz verschiedener Bussysteme: transparente Kommunikationsabläufe und Synchronisationsmechanismen (wie Spezifikation von Gateways)
- Definition der Anforderungen für dynamische Methoden bei Verwendung von Multicore, Safety und Security
- Für die Fehlererkennung und Fehlerbehandlung, insbesondere im Falle der Safety-Anforderungen, muss eine Spezifikation der Methoden, des Umfangs und der Abläufe erfolgen.
- Spezifikation von Diagnose-Methoden und Diagnose-Interfaces

- Spezifikation der statischen und dynamischen Testmethoden, insbesondere Datalogging-, Debugging- und Trace-Methoden unter Berücksichtigung von Multicore, Safety und Security
- Spezifikation der zu verwendenden Art der Symboldefinition im Hinblick auf Multicore
- Spezifikation der Software-Updatemethoden und deren Absicherungsmechanismen, aus statischer Sicht und bei Bedarf auch für dynamische Software-Updates
- Spezifikation der Dokumentationsvorgaben

Fazit

Beim Einsatz von Multicore-Systemen wird eine umfassende und detaillierte Anforderungsanalyse mit Safety- und Security-Anforderungen für die Systemsicherheit zum Schlüssel des Projekterfolgs.

Die weiteren Teile dieser Beitragsreihe zum Thema **Embedded-Software-Entwicklung unter den Aspekten Multicore, Safety und Security** beschäftigen sich näher mit aktuellen Einflüssen und Herausforderungen, Software-Architektur und Software-Tests. Teil 1 beleuchtet, wie Multicore, Safety- und Security-Aspekte die Softwareprojekte von heute verändern. In Teil 3 geht es um das Thema Software-Architektur.

Holen Sie sich das richtige Wissen zu Embedded Software-Entwicklung, Multicore und Safety & Security. MicroConsult bietet Ihnen professionelle Trainings und Coachings rund um diese Themen an – im Live-Online- und im Präsenz-Format.

Weiterführende Infos

MicroConsult Fachwissen: Multicore & Mikrocontroller

MicroConsult Training & Coaching: Multicore & Mikrocontroller

MicroConsult Fachwissen: Embedded SW-Entwicklung

MicroConsult Training & Coaching: Embedded SW-Entwicklung

MicroConsult Fachwissen: Safety & Security

MicroConsult Training & Coaching: Safety & Security

Alle MicroConsult Trainings & Coachings

Autor

Ingo Pohle ist Mitgründer und Geschäftsführer der MicroConsult GmbH und international anerkannter Spezialist für Embedded-Lösungen, mit einem reichen Erfahrungsschatz rund um den Einsatz von Embedded-Mikrocontrollern, Bussystemen und RTOS.