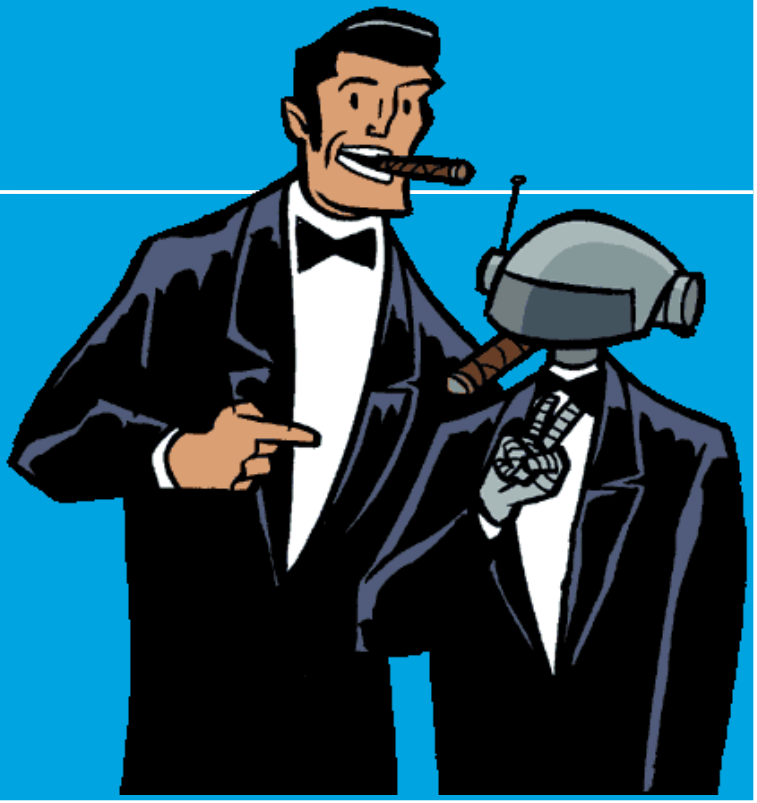
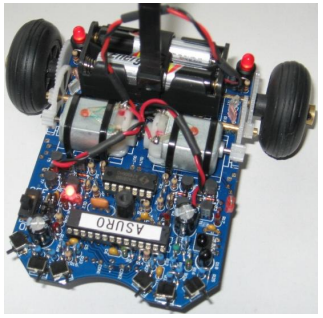
Software-Sicherheit
Risiken und Lösungen

7+1 Heiße Tipps



Was ist der Unterschied zwischen Sicherheit und Zuverlässigkeit?

Sicherheit



Zuverlässigkeit

erreicht durch

Fehlerprävention
und
richtige Fehlerreaktion
im Fehlerfall

erreicht durch

Redundanz
und / oder
Diversität

Ein System ist **sicher**, wenn es im Fehlerfall weder Mensch, Umwelt noch sich selbst gefährdet!

Ein System ist **zuverlässig**, wenn es nach dem Einschalten im Idealfall unendlich lange fehlerfrei arbeitet!

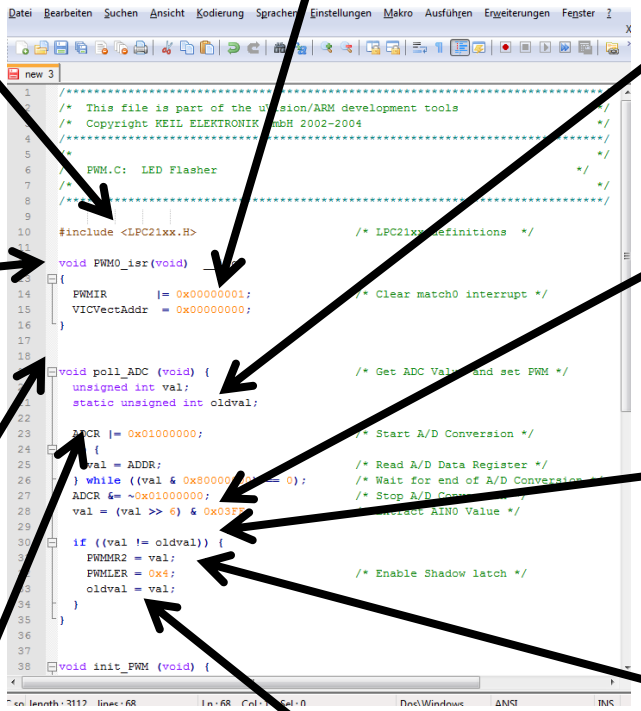
Was sind die am häufigsten in der Software auftretenden Fehler?

Speichergrenzen überschrieben

Daten im Speicher nicht initialisiert

Priorisierung vor Synchronisation anstatt umgekehrt

Arraygrenzen überschritten



Race Conditions nicht beachtet

Zeiger De-Referenzierung falsch verwendet

Deadlocks nicht beachtet

Datenmodifizierer `volatile` nicht beachtet

Floating Point Daten falsch verarbeitet

Reentrace bzw. Thread-Safe-Programmierung nicht beachtet

MICRO CONSULT

Was sind Beispiele für sicherheitssteigernde Mittel im **Entwicklungsprozess**?

Top-Down-Ansatz in der Entwicklung
(Analyse ➔ Design ➔ Implementierung ➔ Test)

Sicherheitsrelevante Anforderungen gesondert analysieren,
verschiedene Umsetzungsvarianten ausarbeiten, bewerten und auswählen

Umsetzung sicherheitsrelevanter Anforderungen mit besonders vielen
Abnahmekriterien und Testszenarien überprüfen

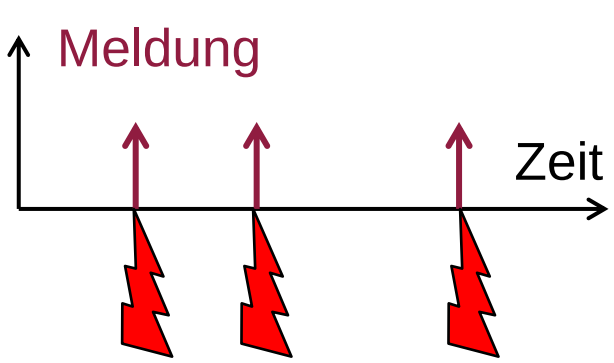
Durchgängige grafische Modellierung von der Anforderungsanalyse bis zum
Abnahmetest, basierend auf Modellierungsrichtlinien

Implementierung basiert auf Programmiersprachen-spezifischen
Codierungsrichtlinien

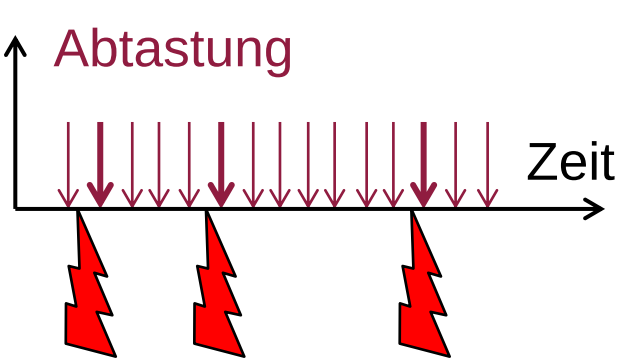
Etablierung von Reviews für Artefakte,
Modell-, Architekturverifikation, statische Codeanalyse, formale Verifikation

Was sind Beispiele für sicherheitssteigernde Mittel
in der **Software-Architektur**?

Asynchron:
direkte Meldung

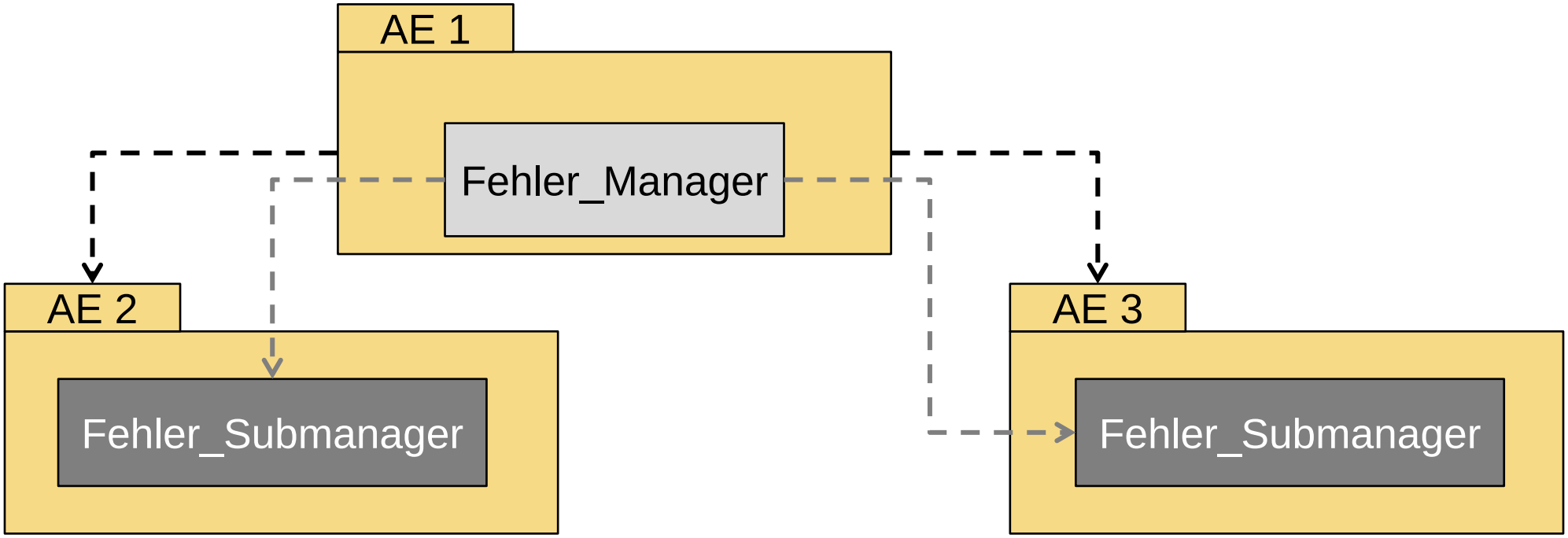


Synchron:
periodische Abtastung



Synchrone Software-Architekturen sind **deterministischer** als asynchrone.

Was sind Beispiele für sicherheitssteigernde Mittel
in der **Software-Architektur**?



- Zentrale Fehlerbehandlung: **Fehler_Manager**
- Dezentrale Fehlerbehandlung: **Fehler_Manager** und **n Fehler_Submanager**

Eine generelle Aussage, welche der beiden Fehlermanagement-Architekturen in sicherheitskritischen Systemen zu bevorzugen ist, lässt sich nicht treffen.

Was sind Beispiele für sicherheitssteigernde Mittel bei der **Implementierung** der **Applikation**?

Automatische Dateninitialisierung

CCounter
- mCount: uint32_t - mLimit: uint32_t {readOnly}
+ **CCounter(parCount, parLimit)** + setCount(parCount):void + getCount():uint32_t + count():void

➔ Konstruktor in C++

Sichere Datentypen

Cuint32_t
- **mValue1: uint32_t** - **mValue2: uint32_t** - **mStatus: bool**
+ Cuint32_t() + setValue(parValue):Cuint32_t + getValue():Cuint32_t

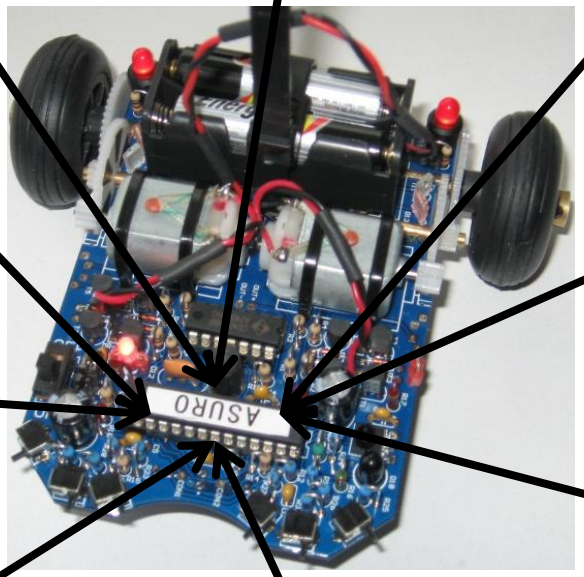
➔ Eigene Datenklassen in C++

Was sind Beispiele für sicherheitssteigernde Mittel bei der Implementierung von **hardwarenaher** Software?

Internes und externes Peripherieregister zyklisch re-initialisieren

Prozessor- und Speichertests zyklisch durchführen

Interne und externe Watchdog-Timer verwenden



ROM-Inhalte dynamisch überwachen

Systemkritische Größen (Takt, Spannungen, Strom, Temperatur, …) überwachen

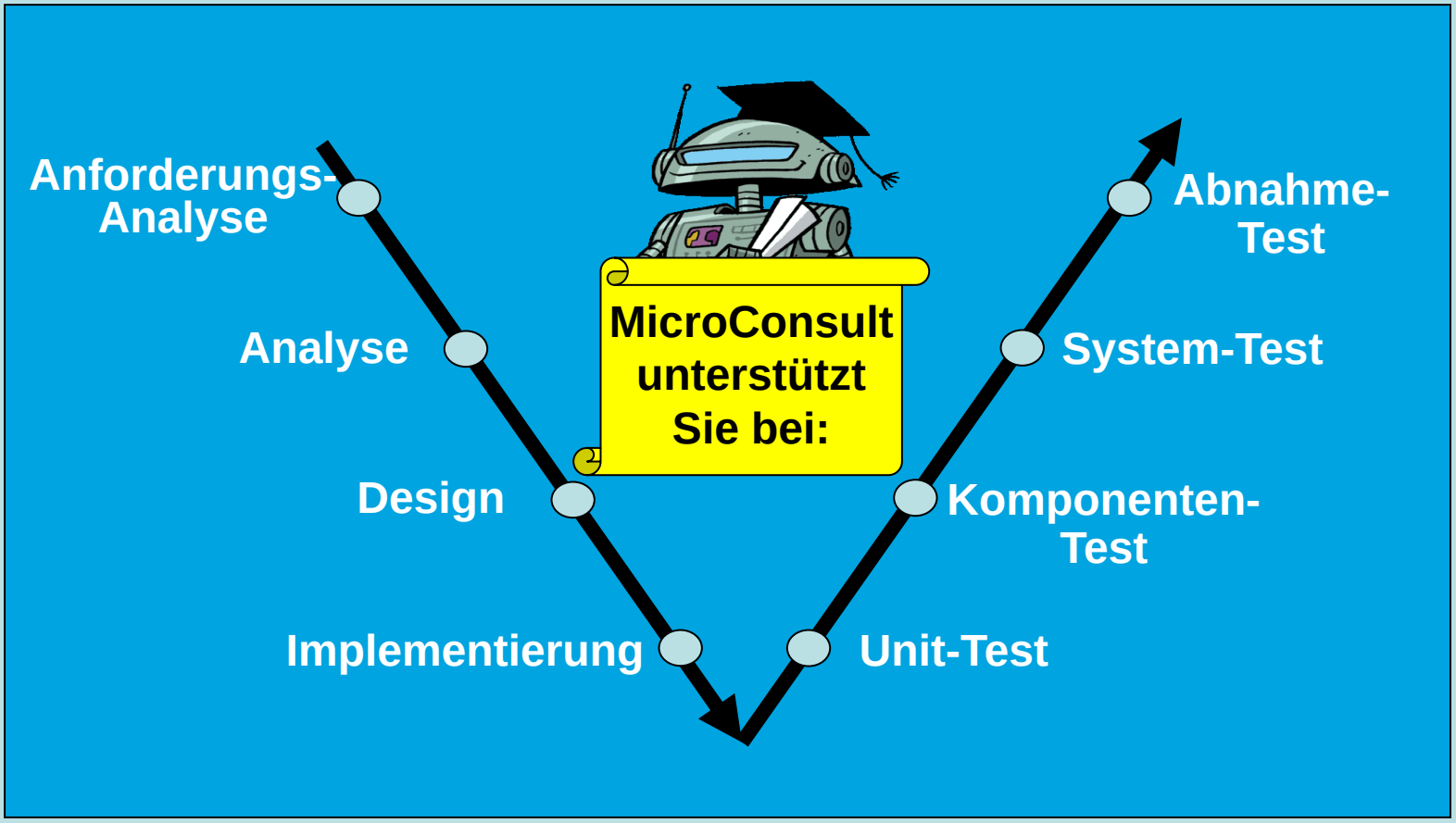
Speichergrenzen dynamisch überwachen

Schnittstellen-Daten auf Gültigkeit bzw. Plausibilität überprüfen

Datenmodifizierer `volatile` wo notwendig verwenden

Ungenutzte Interruptvektoren mit einer definierten Serviceroutine belegen

Beratung, Training, Workshops, Coaching, Consulting



HW-/SW-Technologien, Tools, Methoden, Prozess, Team

Vortrag per E-Mail anfordern: info@microconsult.de
Betreff: „Vortrag embedded world 2012“