BSSE System and Software Engineering

Automatische Software Produktion
Was ist realistisch?

MicroConsult Praxisforum „Embedded Quality“
Neue Dimensionen der Qualitätssicherung

Version 2

München 15.10.2002

Dr. Rainer Gerlich
Auf dem Ruhbühl 181
88090 Immenstaad
Germany

Tel. +49/7545/91.12.58
Fax +49/7545/91.12.40
Mobil +49/171/80.20.659
e-mail gerlich@t-online.de

“Automation liegt in der Luft”

- wichtige Punkte der heute gehaltenen Vorträge
  - Entwicklungsprozesse definieren
  - iterativ und inkrementell entwickeln
  - so früh wie möglich testen
  - Tests ableiten aus Spezifikation
  - ausführbare Spezifikationen
  - Wissen fixieren und weitergeben
- dieses Wissen können wir automatisieren
  - die Automation impliziert all diese Punkte, und noch mehr ...
  - ... Automation impliziert CMM level 5

Wie kann ich mir Automation vorstellen?

- Automation setzt manuelle Prozesse um
  - was der Entwickler täglich macht, ...
  - ... setzt die Automatische Software Produktion um
  - Konstruktionsregeln
  - Modelle für den Ablauf
- durch Automation werden manuelle Abläufe reproduzierbar
- ... und die Qualität wird beherrschbar
  - wenn manueller Anteil von 100% auf 1% sinkt ...
  - dann verringert sich die Fehlerrate sofort um 100

Beispiel: Spreadsheet

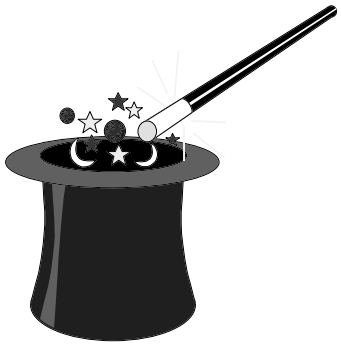
- Kopfrechnen
  - Zahlenkolonnen addieren, subtrahieren ...
- und im Spreadsheet ...
  - sum(a_1 .. a_N)
  - das sind die Regeln
- Vorteile
  - bei Änderung Ergebnis sofort verfügbar

    Spalten + Zeilen einfügen, Werte verändern, Grafik
  - Softwareänderungen

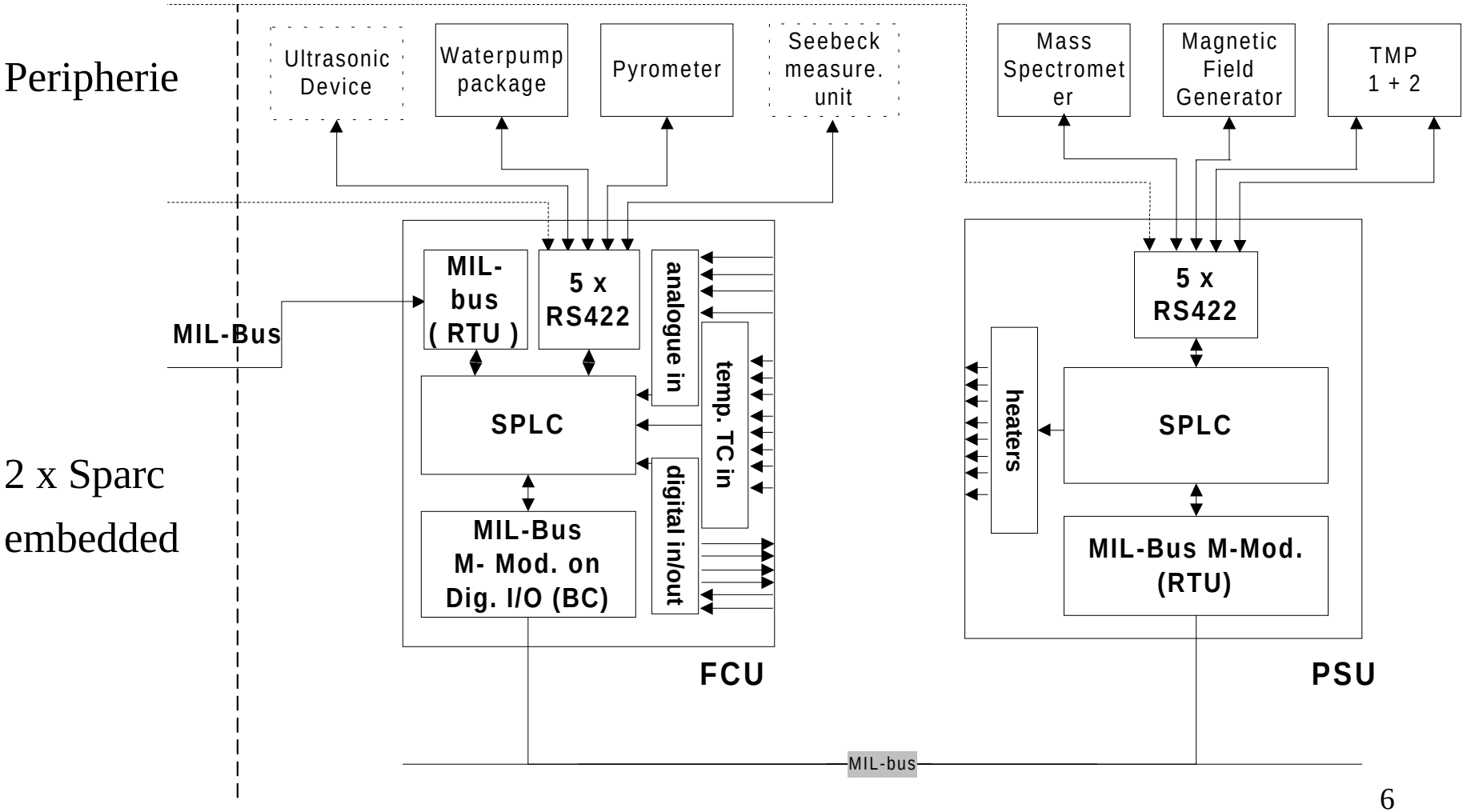
    Prozesse, Messages, Topologie, OS

Der Herstellungsprozess

- ... ist wie ein make-File
  - Aktionen ausführen
  - Bedingungen prüfen
  - Ergebnis überwachen
- ... macht im Batch das,
  - was man sonst manuell macht ...
- ... paßt sich an die Vorgaben an
- ... ein „Hauch“ von
  - Expertensystem und künstlicher Intelligenz



Beispiel: Verteiltes Echtzeitsystem (events + synchron)



Automatische Softwareproduktion und Test
ASaP

- Prinzipielles Problem
  - uns fehlen Sensoren zur Identifikation von Softwarefehlern
  - viel leichter, Systembeschreibung oder viel Code zu erstellen ...
    ... als zu verifizieren und zu validieren
- ASaP Lösung

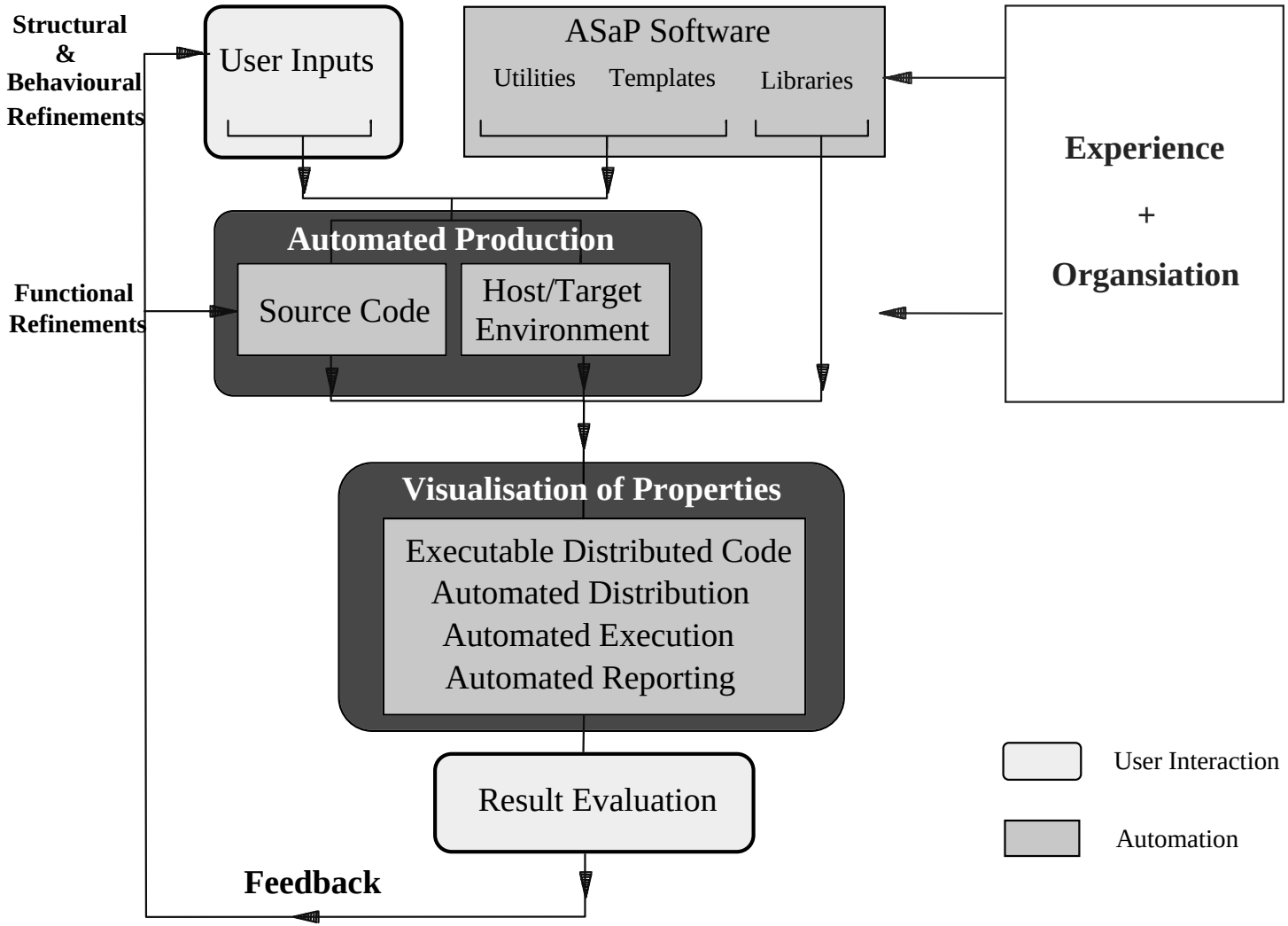
  durch Automation
  - Synergie zwischen Generierung, Verifizierung und Validierung
  - korrekten Code und lauffähiges System garantieren
  - schnelles Feedback vom System durch Visualisierung
- Automatische Codegenerierung
  - deckt nur einen Aspekt von sehr vielen durch Automation ab

Vorteile der Automation

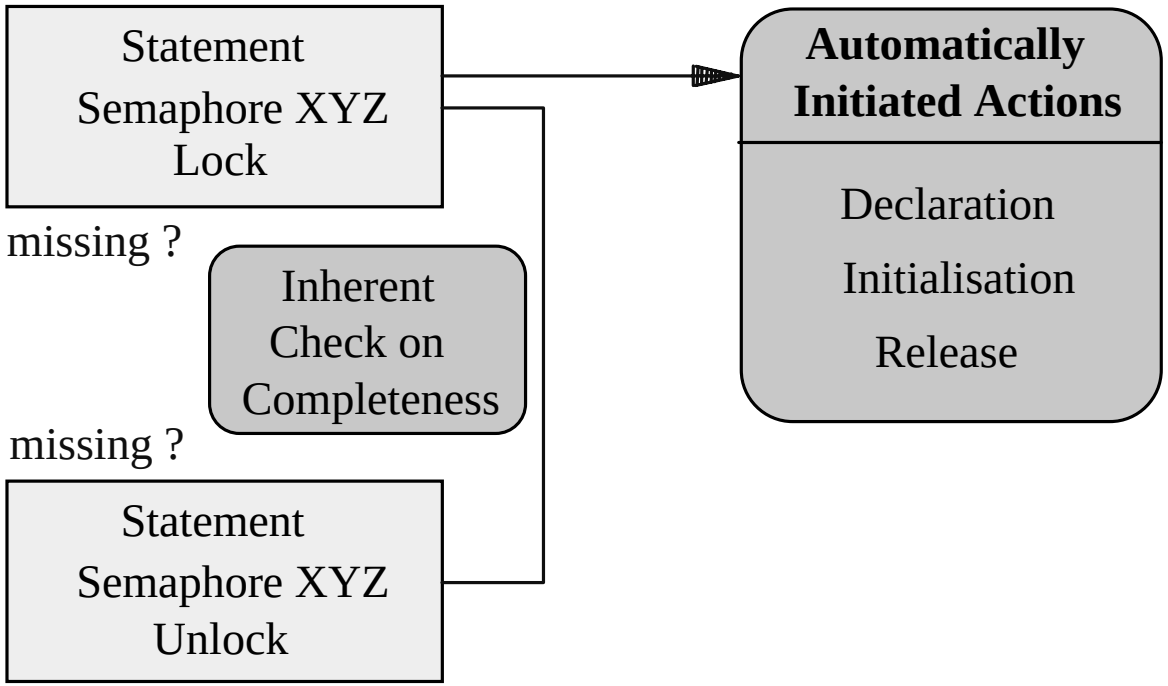
- mehr Effizienz durch Spezialisierung
  - Spezialisierung auf Teilbereich ohne Beschränkung der Allgemeinheit

    z.B. verteilte und/oder Echtzeitsysteme (großer Teilbereich)
  - Teilbereich $\rightarrow$ Konkretisierung $\rightarrow$ Produktions- / Prüfregeln
  - diametral zu gegenwärtigen Trend!
- größere Robustheit
  - Fehler behandeln, die man noch nicht kennt
  - mehr operationelle Schwachpunkte erkennen
- „und automatische Dokumentation ist auch mit drin ...“

Inkrementeller System-Entwicklungs-Zyklus von ASaP



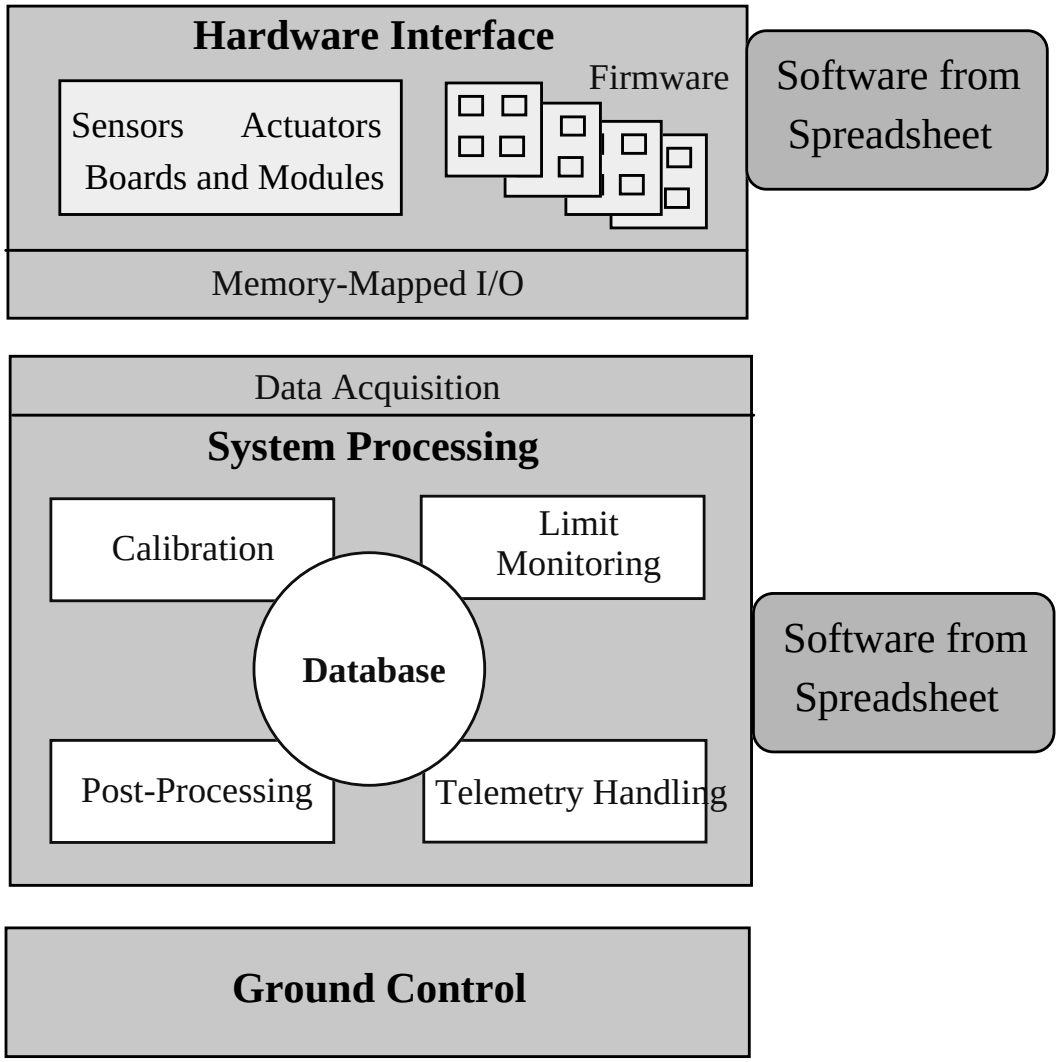
Beispiel 1: Synergie durch Spezialisierung



Semantic: each pair only once

Occurence of a statement: **check on complementary statement**
automated generation of complementary code

Beispiel 2: Automatische Erstellung von Software



File-Package 1

- Input
  - Spreadsheet

    Hardware-Interfaces, Datentyp, Sampling Rate, Board

    Kalibrierungsdaten, Grenzwerte, Sensortyp

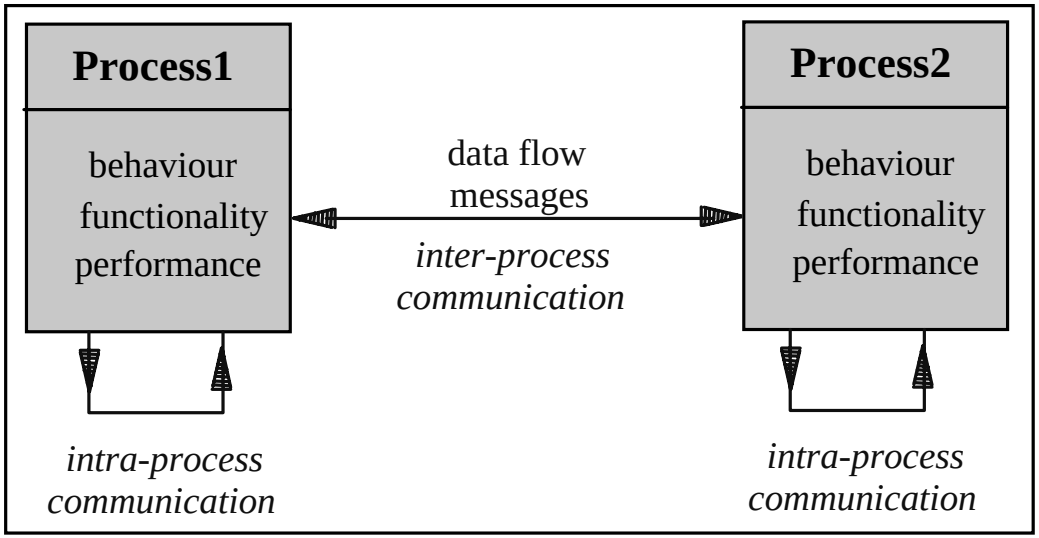
    IRQ, Telemetrie-Frames, CPU, ...
- Output
  - c- und h-Files

    Datenbankdeklaration, Look-Up Table (HW), Telemetrieframes

    Grenzwertüberwachung, spezifische Sensor-Kalibrierung, ...

Beispiel 3: Konfigurationsoptionen

Principal Configuration Options of a Distributed / Real-Time System



Intra-Process Activities
Finite State Machines
synchronous processing
asynchronous / sporadic processing
time-out
exception handling

Distribution
Mapping Process - Processor
Communication Channels
Topology
Fault Tolerance
OS

File-Package 2

- Input (aus Zeitgründen nicht gezeigt)
  - Spreadsheet

    Prozessdefinition, Finite-State-Machines, Datenverarbeitung

    Kanäle, Timing, Timeout, Datenmengen, Exception Handling, ...
- Output
  - c- und h-Files (aus Zeitgründen nicht gezeigt)
  - Visualisierung

    Input: State Transitions, Datenfluss (Übersicht + detailliert)

    Funktions-Hierarchien mit Parametern, ...

    Output: Timing Diagramme, MessageSequenceCharts + debug-info

    Test Coverage, Testfälle, Antwortzeiten, Buffernutzung,

    Profile, CPU-Last, Kanal/Bus-Auslastung, Exceptions, ...

Automatisches Testen und Verifizieren

- Quellcode schon vorhanden ?

- ... vollständige Automation geht auch hier !
  - Testfälle aus vorhandener Information ableiten
  - Testberichte automatisch erstellen

Mehr Qualität muß nicht teurer sein ...

wenn eine neue Technologie eingesetzt wird

Beispiel Automation durch SMD (Surface Mounted Devices)

- Fernsehapparat 1960

  Preis ca.100 Mh, wenig Funktionalität, hohe Ausfallrate

- Fernsehapparat 2000

  Preis ca. 20 Mh, hohe Funktionalität, sehr geringe Ausfallrate

Durch Automation erzielte Produktivität und Qualität

- Produktivität
  - in 30 Minuten das Äquivalent von ca. 5 Mann-Jahren (MJ)
    - ✡ verteiltes Echtzeitsystem 80,000 LOC (200,000 LOC)
    - ✡ in 30 Minuten das Äquivalent von ca. 1 MJ
    - ✡ verteilte synchronisierte Datenbank 16,000 LOC
  - in 1 Minute das Äquivalent von ca. 2 MJ
    - ✡ Operationen zu Datentypen, Interfaces etc.
- Fehlerrate
  - verteiltes Echtzeitsystem 2 = 2.5 .. 1 * 10^{-5}
  - verteilte Datenbank 0
- Literatur
  - üblich: 10^{-2} sehr gut: $<10^{-3}$

N.E.Fenton, 2000

Wie ist Automatisierung bei mir möglich?

- Analyse der „Herstellungsabläufe“ von Software
  - vergleichbar REFA-Ansatz
  - für ein möglichst breites Anwendungsgebiet wie
    Echtzeitsysteme, verteilte Systeme,
    Client-Server Anwendungen
    GUIs, Datenbanken, Prozess technische Systeme, ...

- Festlegen der optimalen Anwenderschnittstelle
  - Identifikation der Systemparameter („Gestaltung“)
    z.B. Prozesse, Topologie, Informationsaustausch
  - Einbettung in vorhandene Entwicklungsumgebung
    vorhandene Werkzeuge, ...
- ständige Optimierung der Abläufe
  - Bereitstellung von Analysewerkzeugen
  - Kontrolle von Produktivität und Qualität durch kontinuierliches Benchmarking