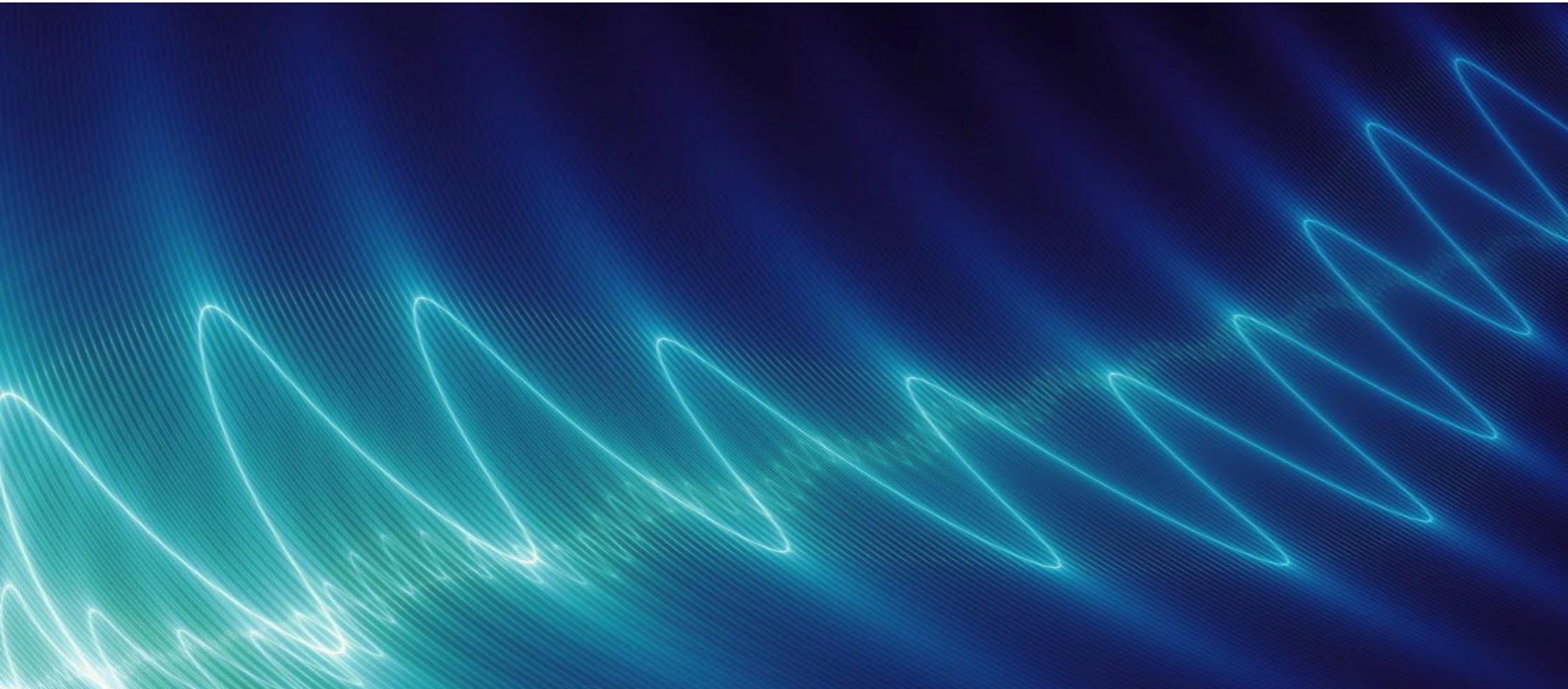
QAWARE
SOFTWARE ENGINEERING

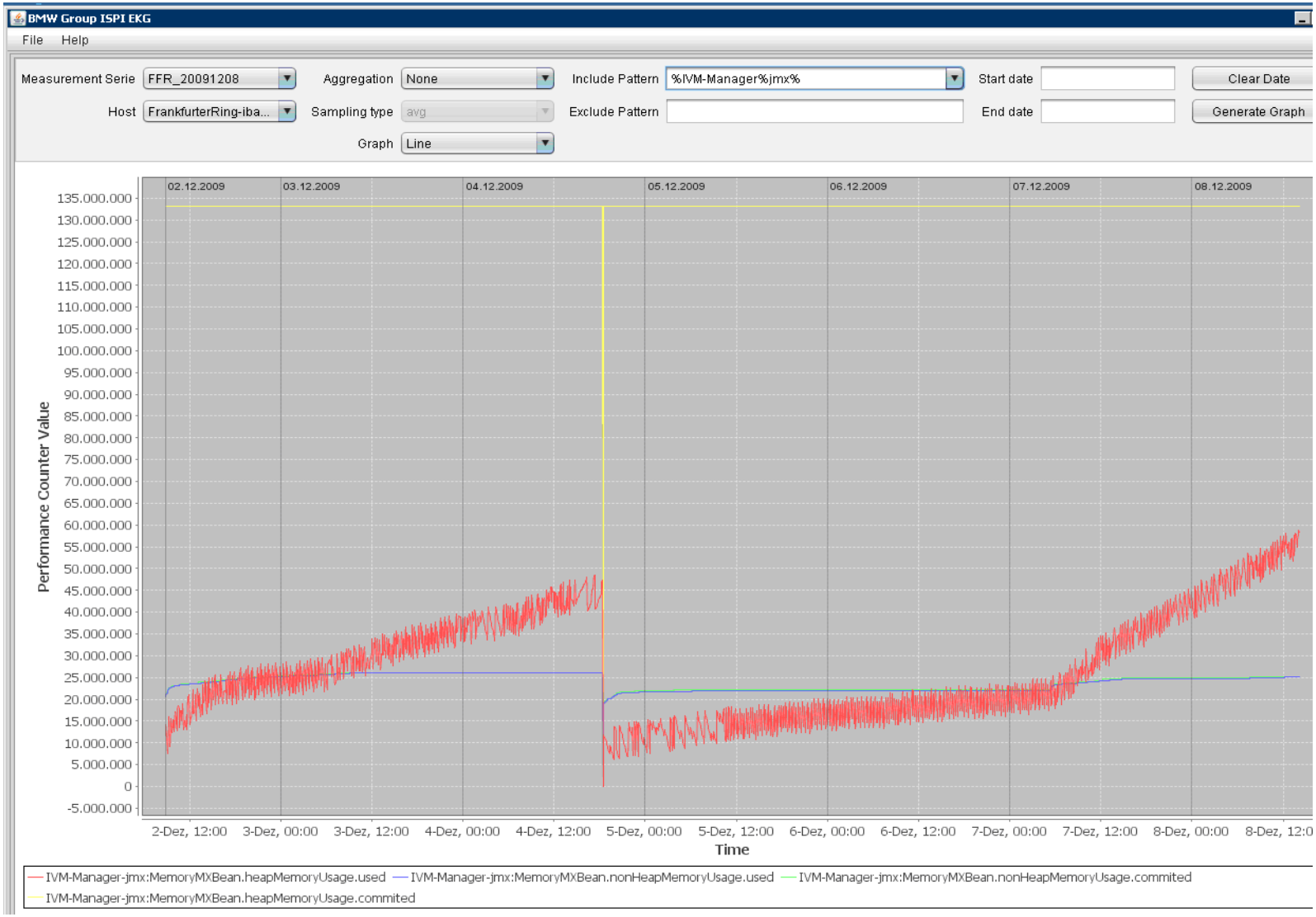
Dynamische Analyse mit dem Software-EKG

Kiel, 29.11.2012

Johannes Weigend



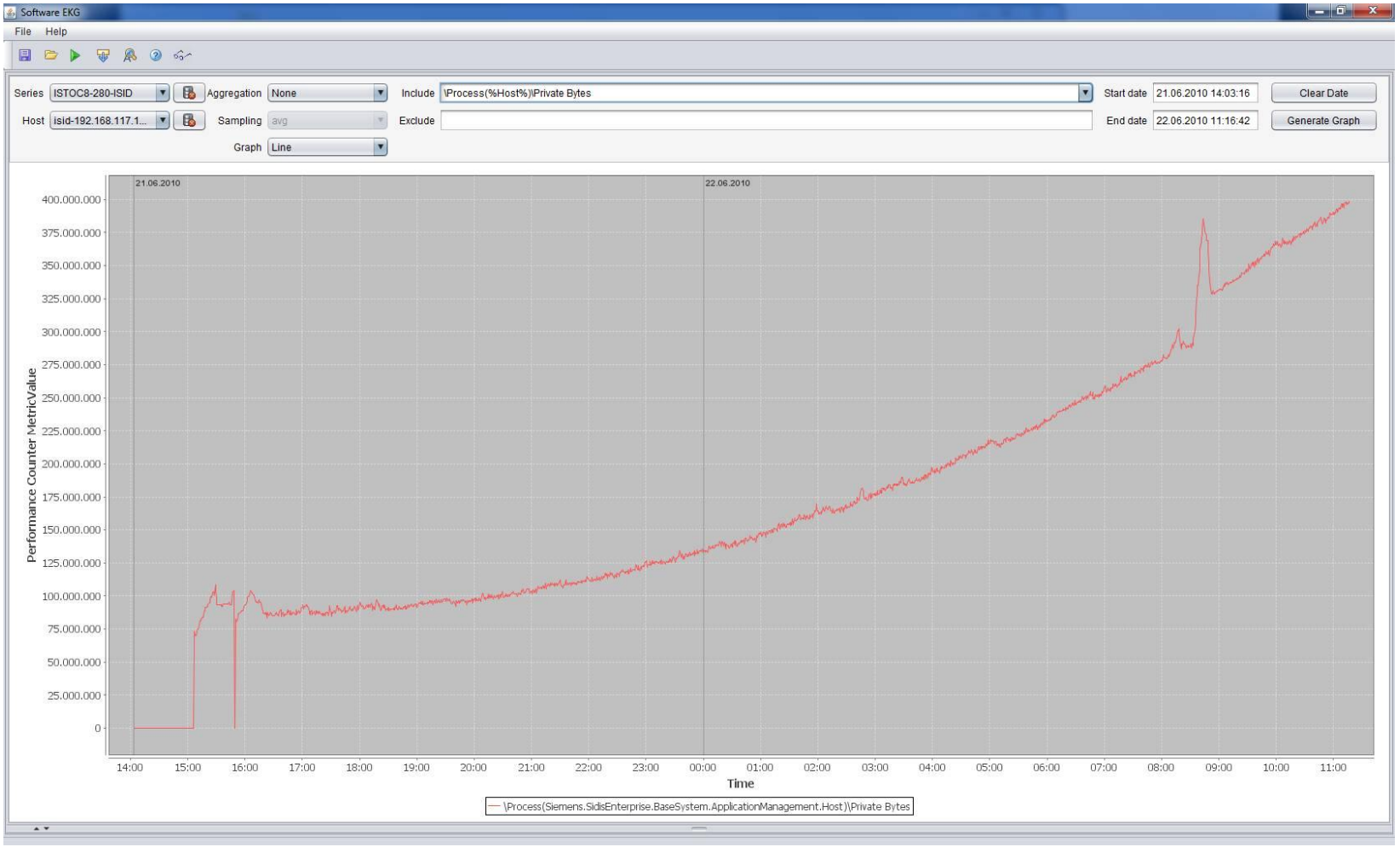
Mein Produktionssystem läuft ungefähr zwei Tage. Dann wird es immer langsamer und bleibt schließlich ganz stehen …



JVM Heap

2.12. 12.00

8.12. 12.00

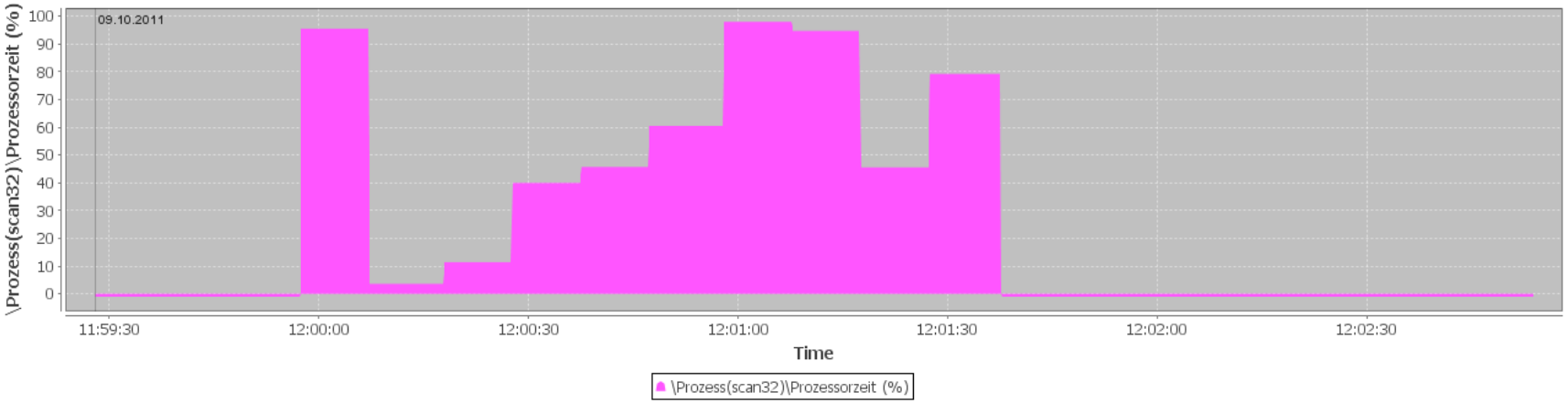


belegter Speicher (private bytes)

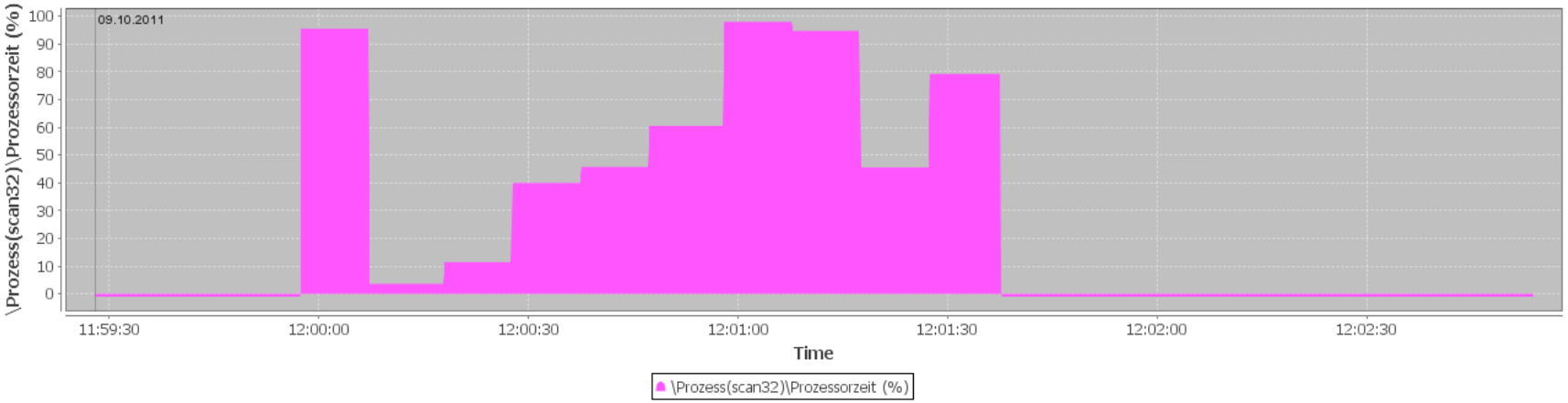
21.06. 14.00

22.06. 11.00

?


09.10.2011
\Prozess(scan32)\Prozessorzeit (%)
100
90
80
70
60
50
40
30
20
10
0
11:59:30
12:00:00
12:00:30
12:01:00
12:01:30
12:02:00
12:02:30
Time
\Prozess(scan32)\Prozessorzeit (%)

Der Virenscanner ist täglich um 12:00 für 2 Minuten aktiv



Mein Produktionssystem lief ein Jahr gut; der letzte Upgrade war vor 3 Monaten. Seit ein paar Tagen wird die Performance immer schlechter …

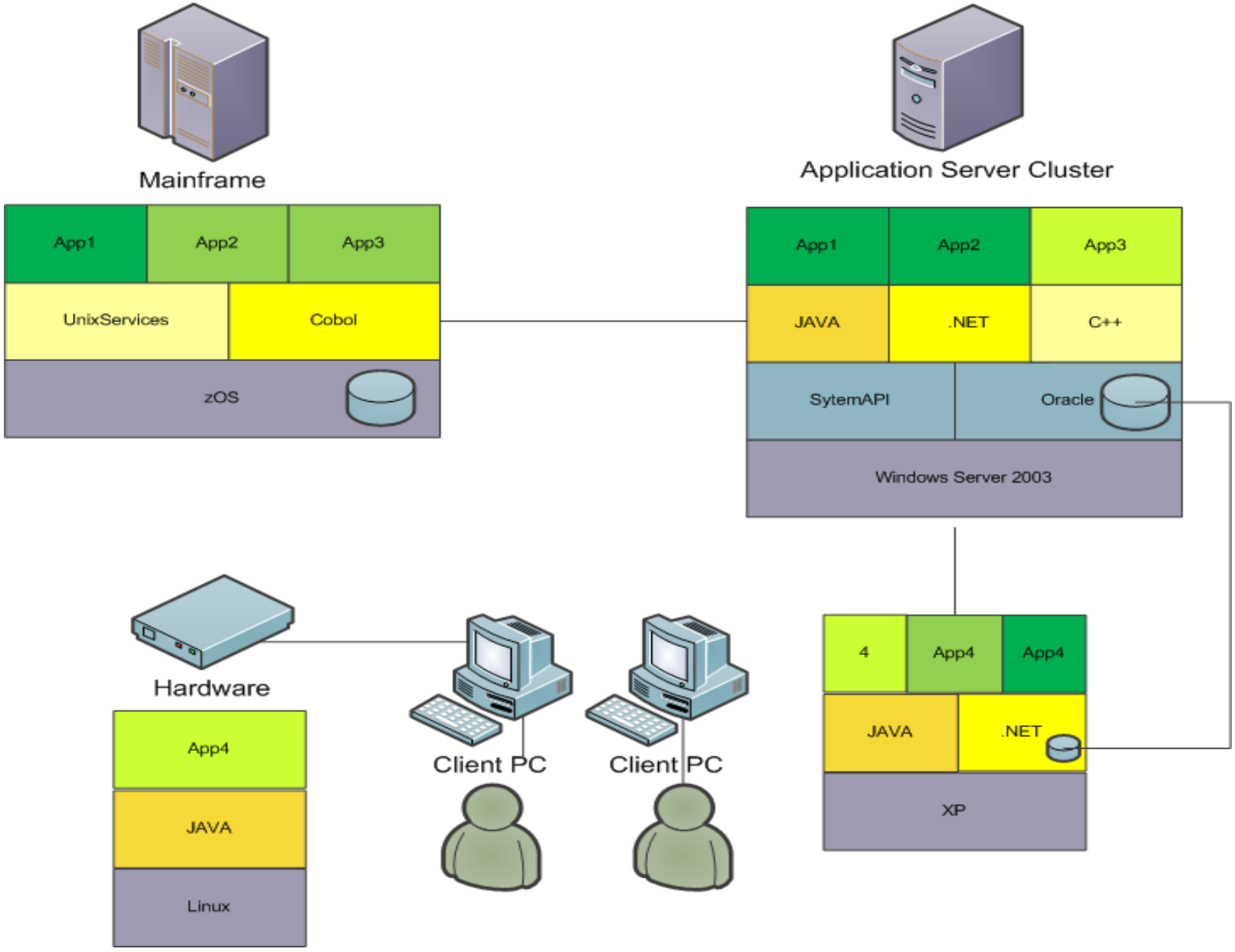
```
List<T> intersection(List<T> xs,
                     List<T> ys) {
  List<T> result = new ArrayList<T>();
  for (T x : xs) {
    for (T y : ys) {
      if (x.equals(y)) {
        result.add(x);
      }
    }
  }
  return result;
}
```

HÄSSLICH,
aber kein Problem,
solange
BEIDE LISTEN KURZ
sind

Mein Produktionssystem stürzt im Durchschnitt einmal im Monat ab. Der Fehler ist nicht reproduzier-bar …

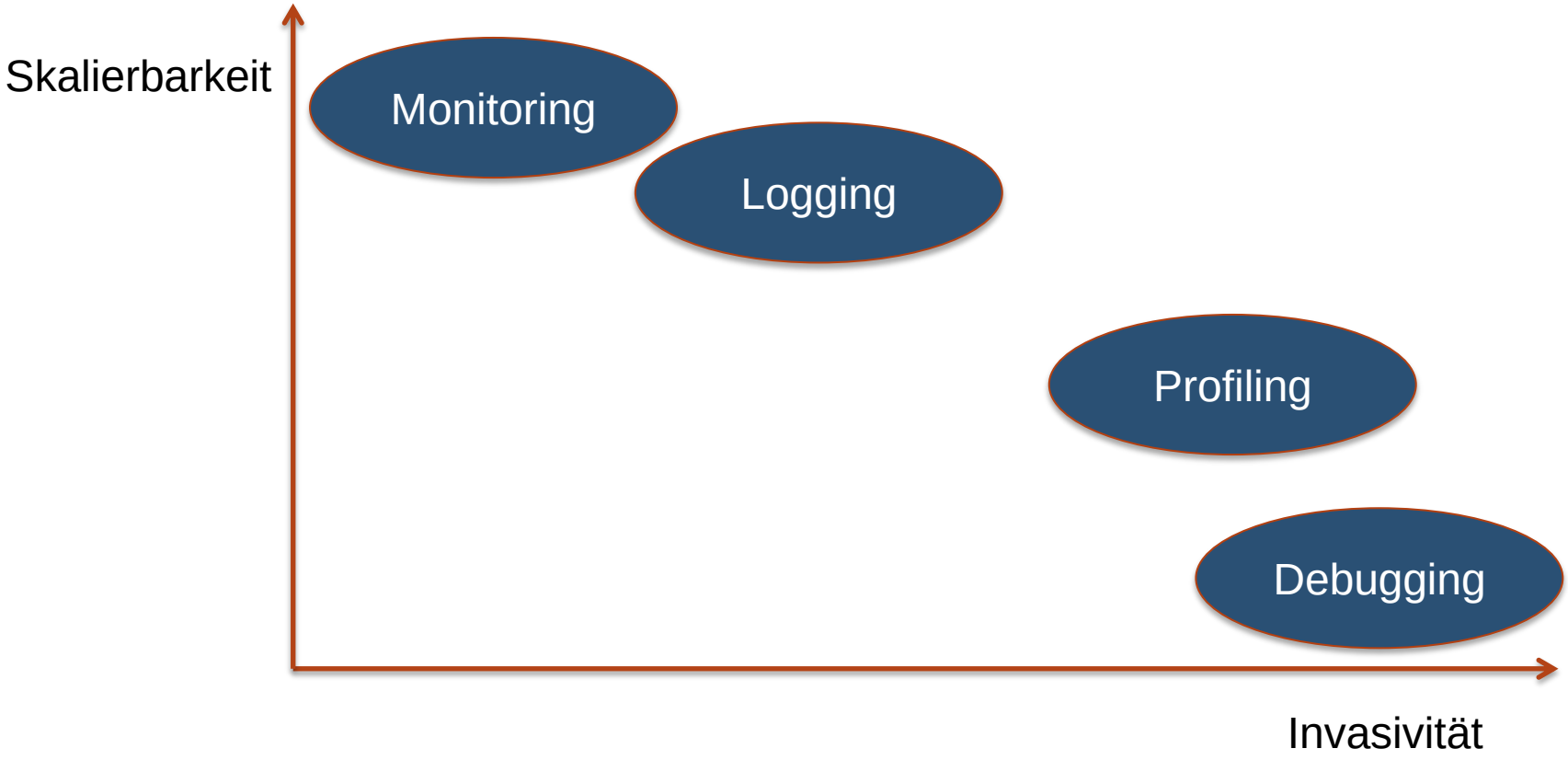
```
private Singleton instance = null;

public static Singleton getInstance() {
   if (instance == null) {
      synchronized(this) {
         if (instance == null) {
            instance = new Singleton();
               // initialize instance
         }
         return instance;
      }
   }
}
```

double-checked locking
JUST DOESN‘T WORK


Mainframe
App1
App2
App3
UnixServices
Cobol
zOS
Application Server Cluster
App1
App2
App3
JAVA
.NET
C++
SytemAPI
Oracle
Windows Server 2003
Hardware
App4
JAVA
Linux
Client PC
Client PC
4
App4
App4
JAVA
.NET
XP

1. Dynamische Software-Analyse
2. Software-EKG
3. Computer Aided Software Diagnosis (CASD)
4. Vorbeugen ist besser als Heilen
5. Naive Logging vs. Clever Logging
6. Design for Diagnosibility (DfD)

Dynamische Software-Analyse



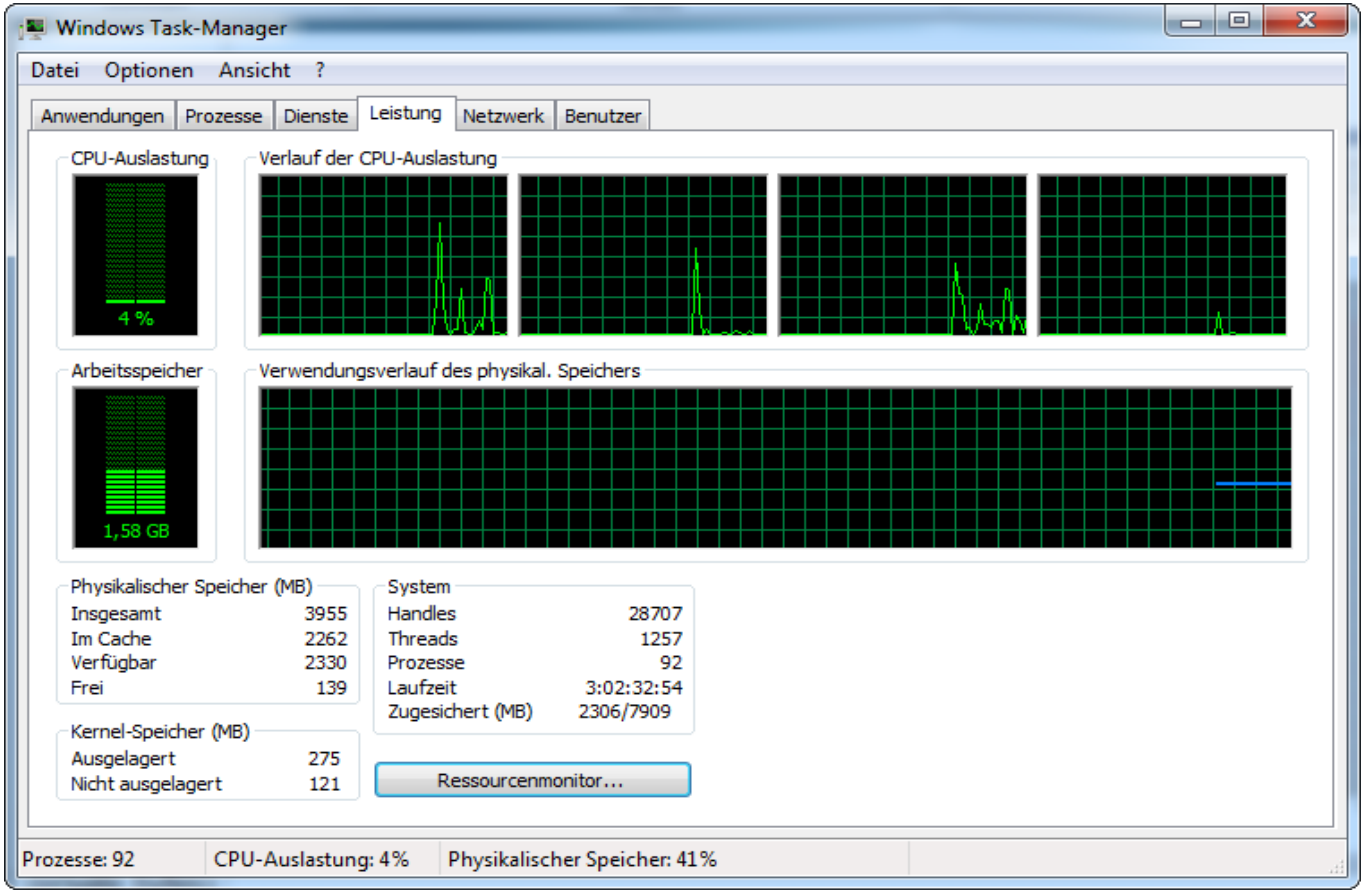
Wann passt welche Methode?

	Programmier-ung	White Box Integration	Black Box Integration, Abnahme	Produktion
Debugging	xx	x		
Profiling	x	xx	x	
Logging	xx	xx	x	x
Monitoring		x	x	xx

White Box

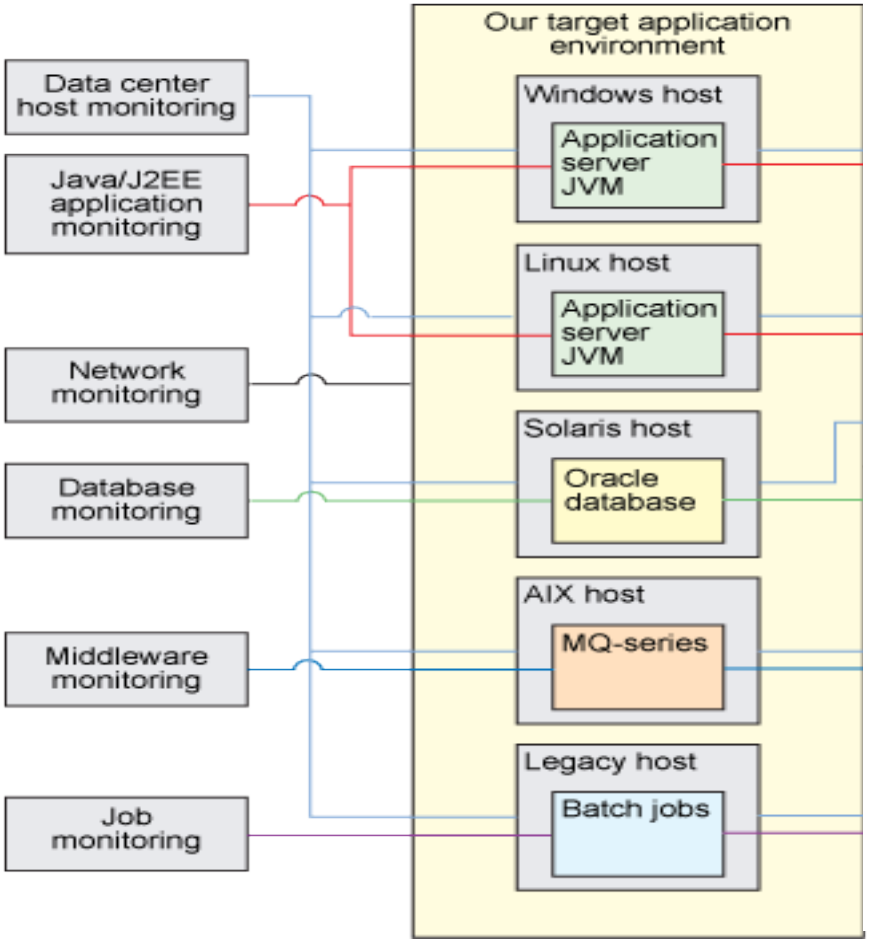
Black Box

Monitoring im Kleinen: Task Manager



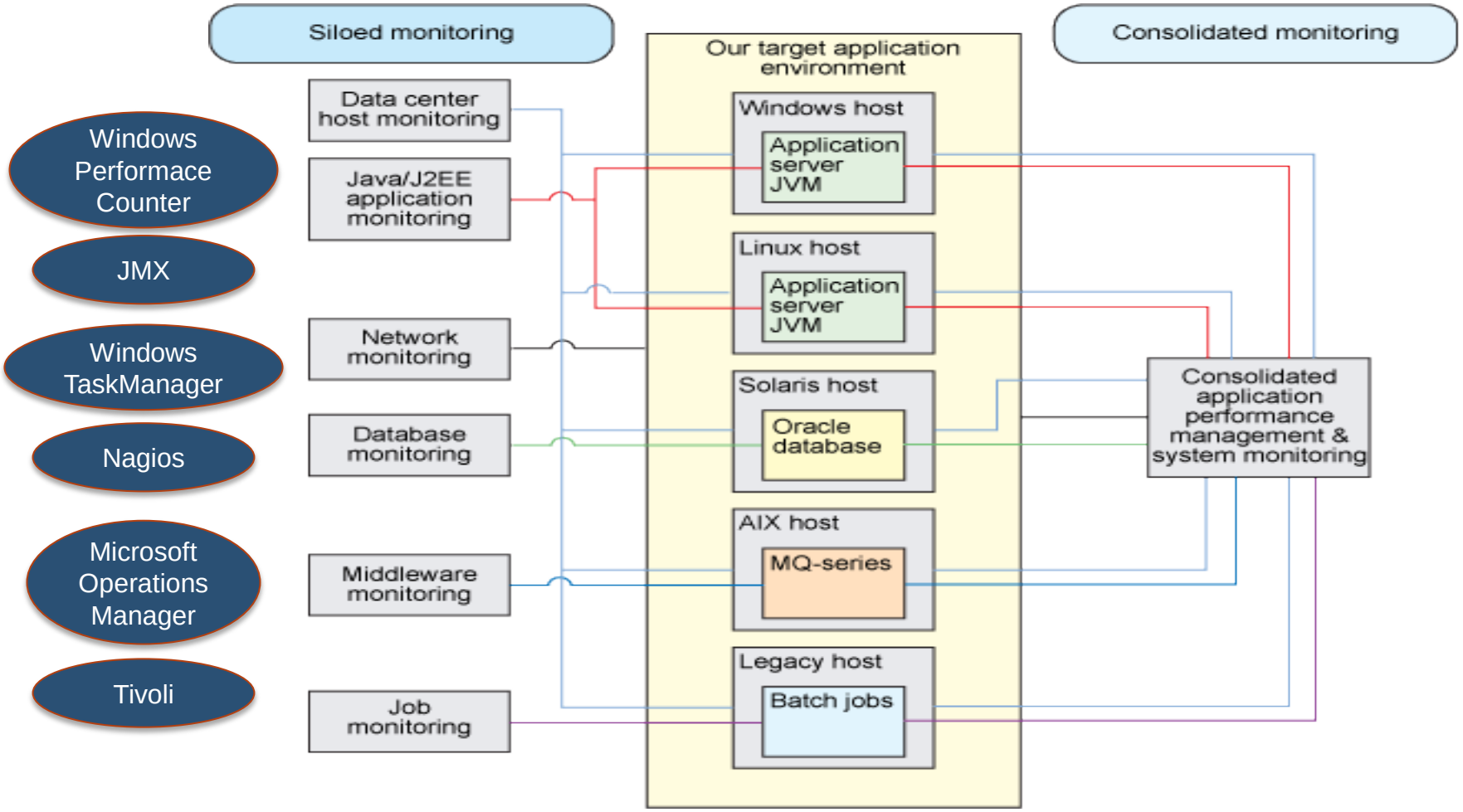
Monitoring ist viel mehr

- Windows Performace Counter
- JMX
- Windows TaskManager
- Nagios
- Microsoft Operations Manager
- Tivoli



Quelle: IBM

Siloed vs. Consolidated

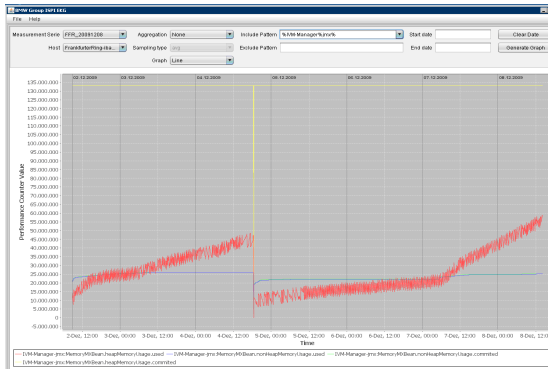


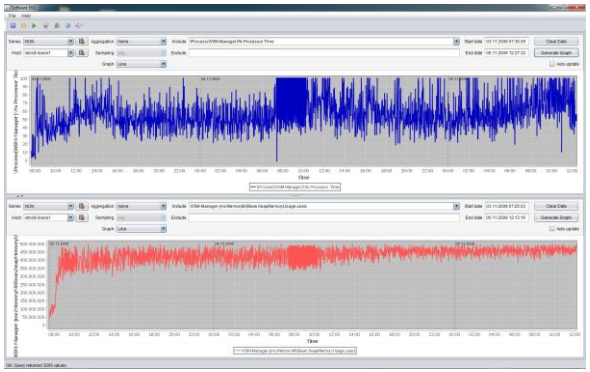
Quelle: IBM

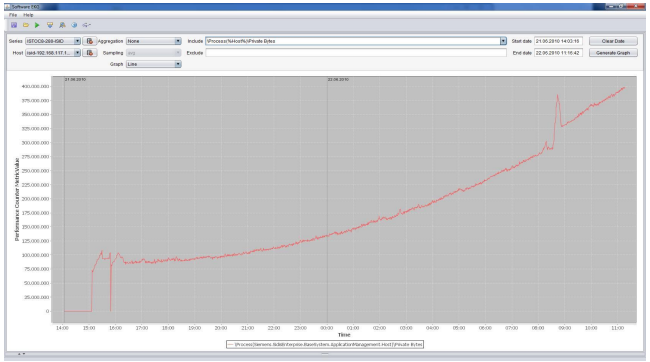
Consolidation – ja, aber…

- Mengenproblem: 10 Rechner mit je 10 Prozessen mit je 10 Threads, 10 Stunden lang 10 Messungen pro Minute ➔ 6 Mio. Messwerte.
- Speicherung, Dokumentation, Nachvollziehbarkeit?

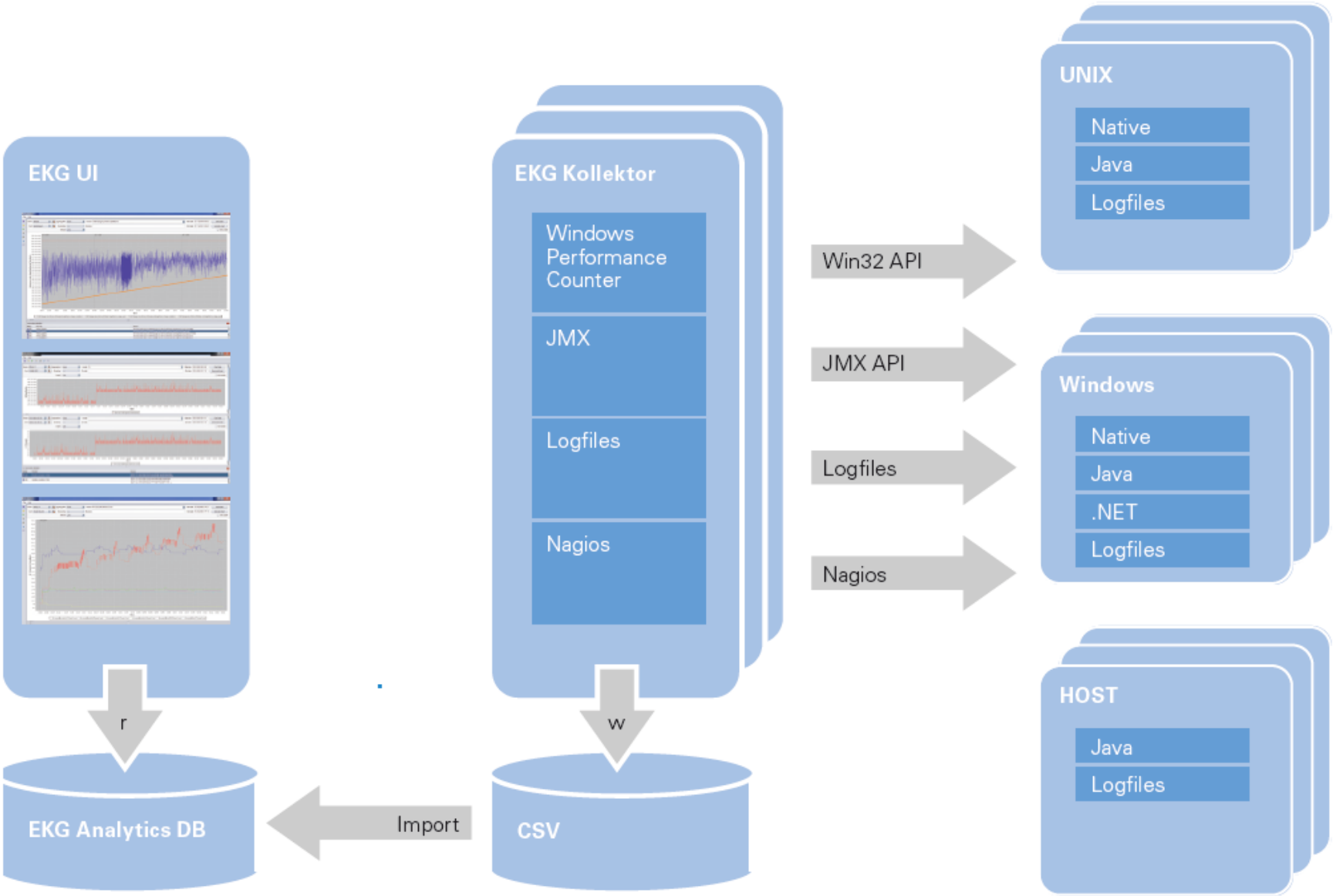
Mit dem SW-EKG Steck-nadeln im Heuhaufen finden



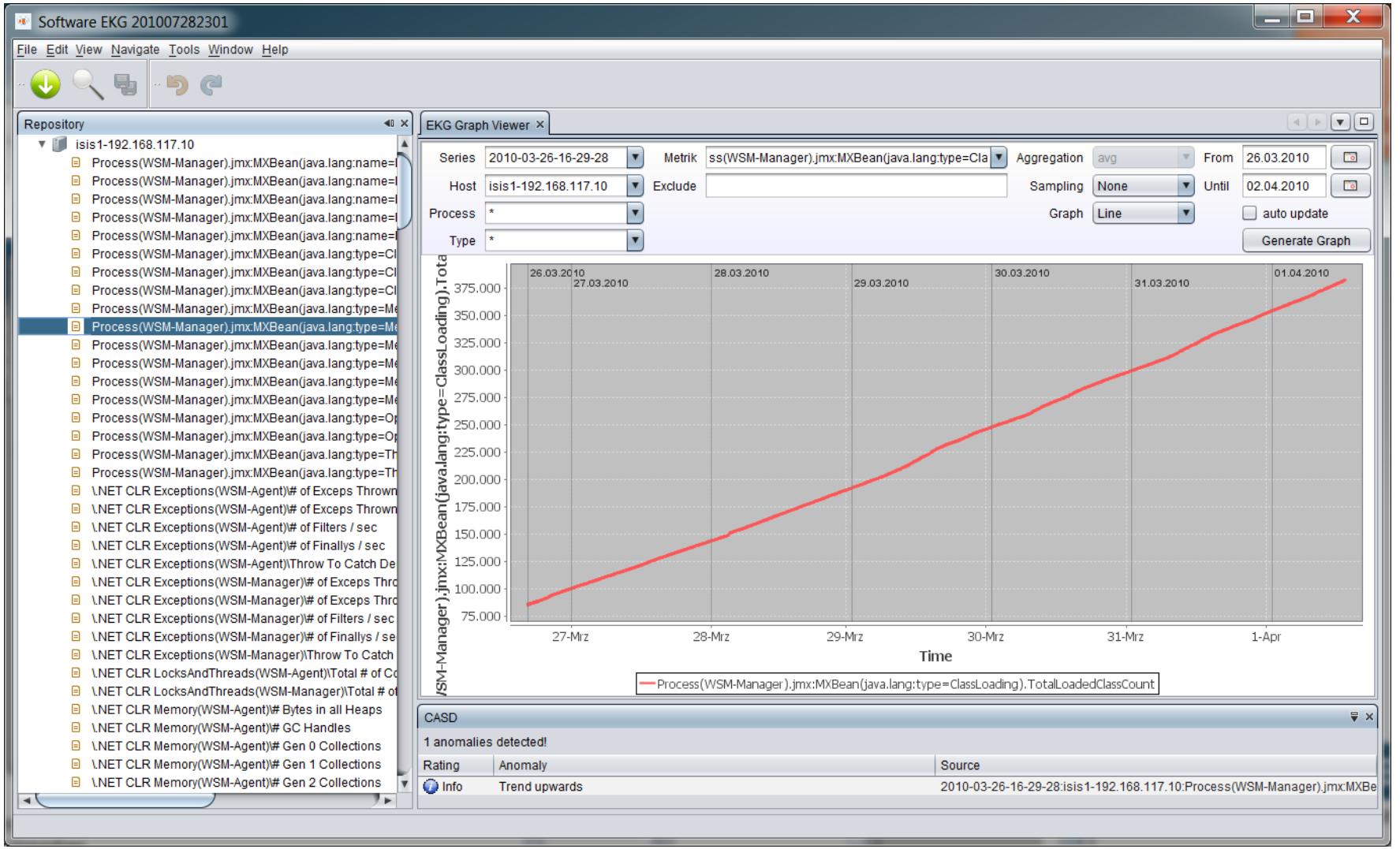




Das Software-EKG im Überblick



LIVE DEMO


Software EKG 201007282301
File Edit View Navigate Tools Window Help
Repository
isis1-192.168.117.10
EKG Graph Viewer
Series 2010-03-26-16-29-28
Host isis1-192.168.117.10
Process *
Type *
Metrik ss(WSM-Manager).jmx:MXBean(java.lang:type=Cla
Exclude
Aggregation avg
Sampling None
Graph Line
From 26.03.2010
Until 02.04.2010
auto update
Generate Graph
26.03.2010
27.03.2010
28.03.2010
29.03.2010
30.03.2010
31.03.2010
01.04.2010
375.000
350.000
325.000
300.000
275.000
250.000
225.000
200.000
175.000
150.000
125.000
100.000
75.000
27-Mrz
28-Mrz
29-Mrz
30-Mrz
31-Mrz
1-Apr
Time
Process(WSM-Manager).jmx:MXBean(java.lang:type=ClassLoading).TotalLoadedClassCount
CASD
1 anomalies detected!
Rating
Anomaly
Source
Info
Trend upwards
2010-03-26-16-29-28:isis1-192.168.117.10:Process(WSM-Manager).jmx:MXBe

Typische Einsatzszenarien

- In Produktion
  - Analyse von Performanz- oder Stabilitätsproblemen
  - Ermittlung von Optimierungsmöglichkeiten
  - Verständnis der Systemlandschaft, aller laufenden Komponenten und Schnittstellen
- In der Entwicklung und im Test
  - Kontrolle über die nicht-funktionalen Eigenschaften
  - Frühzeitige Erkennung von Ressourcenanstiegen (Speicher, CPU, Threads…)
  - Frühzeitige Ermittlung des tatsächlichen Ressourcenbedarfs (Release-Notes!)

Das Software-EKG ist nur gering invasiv

- Invasivität == Einfluss auf das Laufzeitverhalten der Anwendung durch die Überwachung / Messung
- Die Poll-Frequenz entscheidet über die Last
- Profiler und Debugger sind stark invasiv. Profiler können die Laufzeit um bis zu Faktor 10x verschlechtern
- Logging/Tracing im INFO oder DEBUG Level ist bei vielen Anwendungen das TOP-1 Performanzproblem

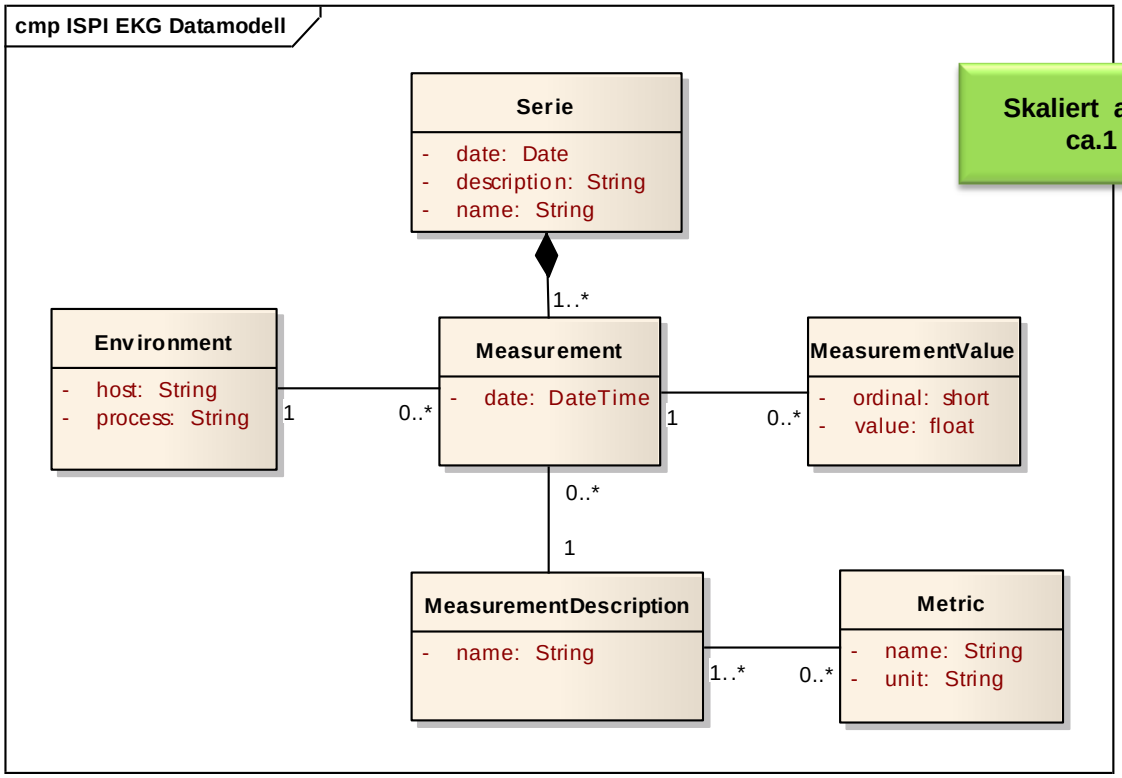
- → In Produktionssystemen können invasive Messwerkzeuge i.R. NICHT eingesetzt werden

Iterationen in der EKG-Datenhaltung

- Version 1.x: CSV-Dateien + serialisierte Hashtabelle
- Version 2.x: Teilnormalisierte Oracle-Datenbank
  - Probleme: Langsamer Import
  - Festplattenbedarf: Faktor 10-100 höherer Speicherbedarf als in CSV-Dateien
  - Oracle-CE ist auf 4GB beschränkt
- Version 3.x: CSV-Dateien + relationale Meta-Datenbank (Derby oder H2)
  - Strukturinformation in Relationaler-DB
  - Daten in CSV-Dateien (kopiert in eine eigenes Repository)
  - Langsam bei sehr großen CSV-Dateien mit vielen Spalten
  - Inkonsistente Unterstützung von regulären Ausdrücken (UI) durch SQL-Like
- Version 4.x: SOLR
  - Vollständig denormalisiertes Datenmodell pro Zeitreihe
  - Meta-Daten und Zeitreihendaten in einem SOLR-Dokument
  - Zeitreihendaten gezippt und BASE64 kodiert
  - Konsistente Query-Syntax und Experten-Query-Syntax (Lucene Query Language)
  - SCHNELL

Rückblick EKG 2: Die effiziente Datenhaltung war eine große Herausforderung im DB-Design und beim Design des CSV-Imports.

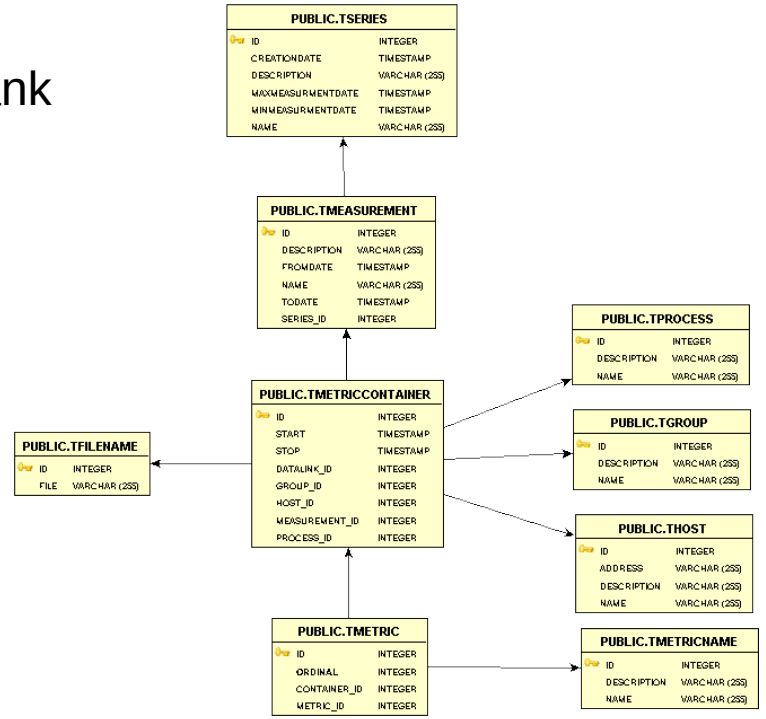
Date	\Memory\Available Bytes	\TCP\Segments/sec	\Processor(_Total)\% Processor Time	\Process(_Total)\Private Bytes	\Process(_Total)\Virtual Bytes
22.10.2009 14:41	2,187,980,800	-1	-1	2,739,781,632	10,000,031,744
22.10.2009 14:41	2,177,044,480	-1	0	2,745,896,960	10,008,424,448
22.10.2009 14:41	2,176,970,752	-1	0	2,745,896,960	10,008,424,448



Rückblick EKG 3: Index-Datenbank und zentrales File-Repository

Skaliert auf einem Rechner bis ca.100 Millionen Messwerten

- Datenhaltung in CSV-Dateien -> DB enthält keine Messwerte
- DB speichert Daten über Serien, Messungen und Datendateien
- **Dateien werden beim Import in ein Repository kopiert – Das Repository (incl. DB) ist kopierbar !**
- Index DB als H2 Single File Datenbank
- Rückgrat der Explorer-Funktionalität



EKG 4.0 – SOLR als NOSQL Datenbank

Skaliert auf einem Rechner bis ca.10 Milliarden Messwerten

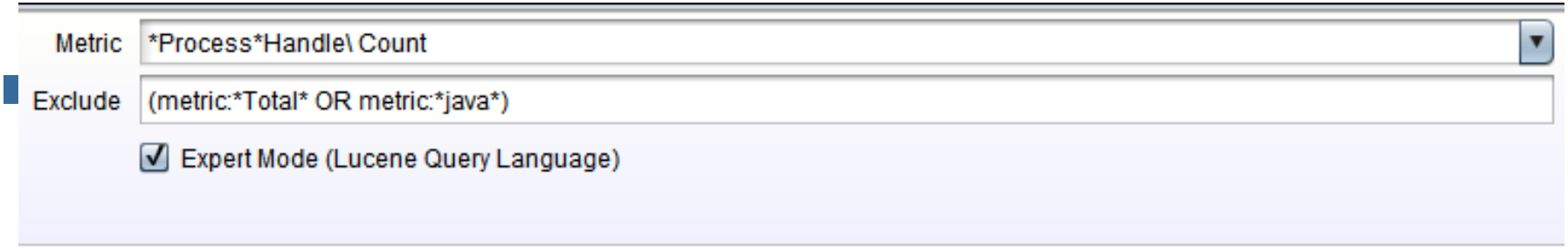
Apache Solr

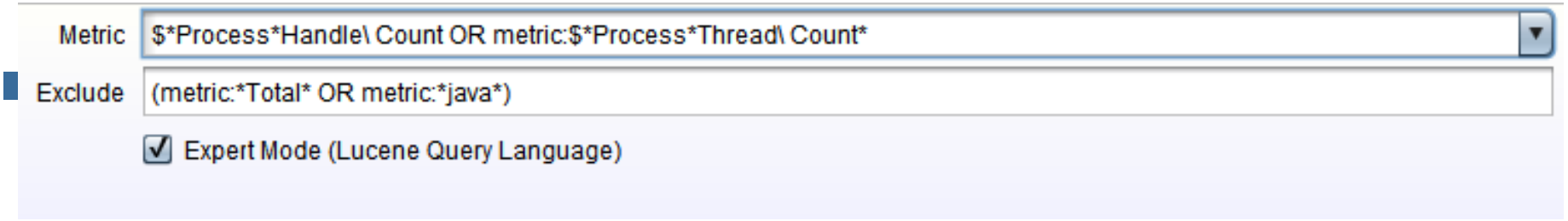
GZIP + BASE64 kodiert

record | bookmark

ISTOC8 Messung Plattform 3.1 : 1 Milliarde Messwerte in 400.000 Messreihen

Mit SOLR/Lucene sind auch komplizierte Abfragen möglich





Software-EKG = Werkzeug + Vorgehen + Kompetenz

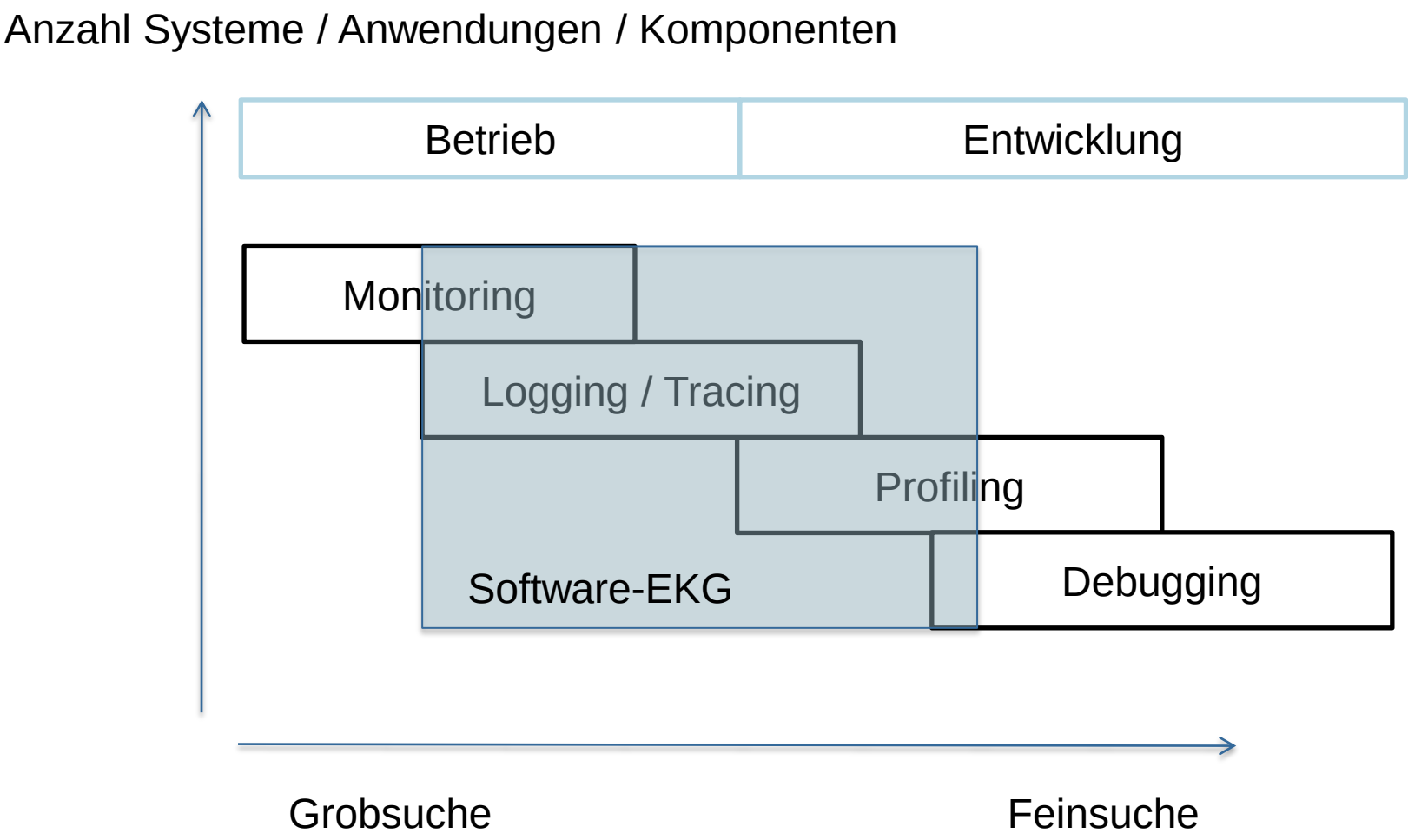
- Systematische Beobachtung verschiedener Messwerte in Korrelation mit dem Benutzerverhalten über kurze und lange Zeiträume.
- Speicherung aller Messdaten in einer Datenbank; komfortable Funktionen zum Verdichten und Visualisieren.
- Schnittstellen zu gängigen Messwerkzeugen (JMX, Performance Counter, Nagios)
- Erkennung von Anomalien durch vertikale Schnitte, Unterstützung durch Anomaliedetektoren

Software-EKG vs. Monitoring

	Software-EKG	Monitoring
Zweck	Fehlersuche	Überwachung, Alarm
Messfrequenz	hoch, bis 10/sec	niedrig, bis 1/min
Messwerte pro Prozess	10 – 100	< 10
Messwerte pro Rechner	10 - 100	< 10
JMX pro JAVA Prozess	10 – 100	nicht benutzt
Http pro Web-Prozess	10 – 100	nicht benutzt

dafür sind die Tools gedacht

Das Software-EKG verbindet Welten



SW-EKG: ein Beitrag zur NF-Methodik

	Programmier-ung	White Box Integration	Black Box Integration Abnahme	Produktion
Debugging	xx	x		
Profiling	x	xx	x	
Logging	xx	xx	x	x
Monitoring		x	x	xx
Software-EKG		xx	xx	x

White Box

Black Box

Einsatzmöglichkeiten

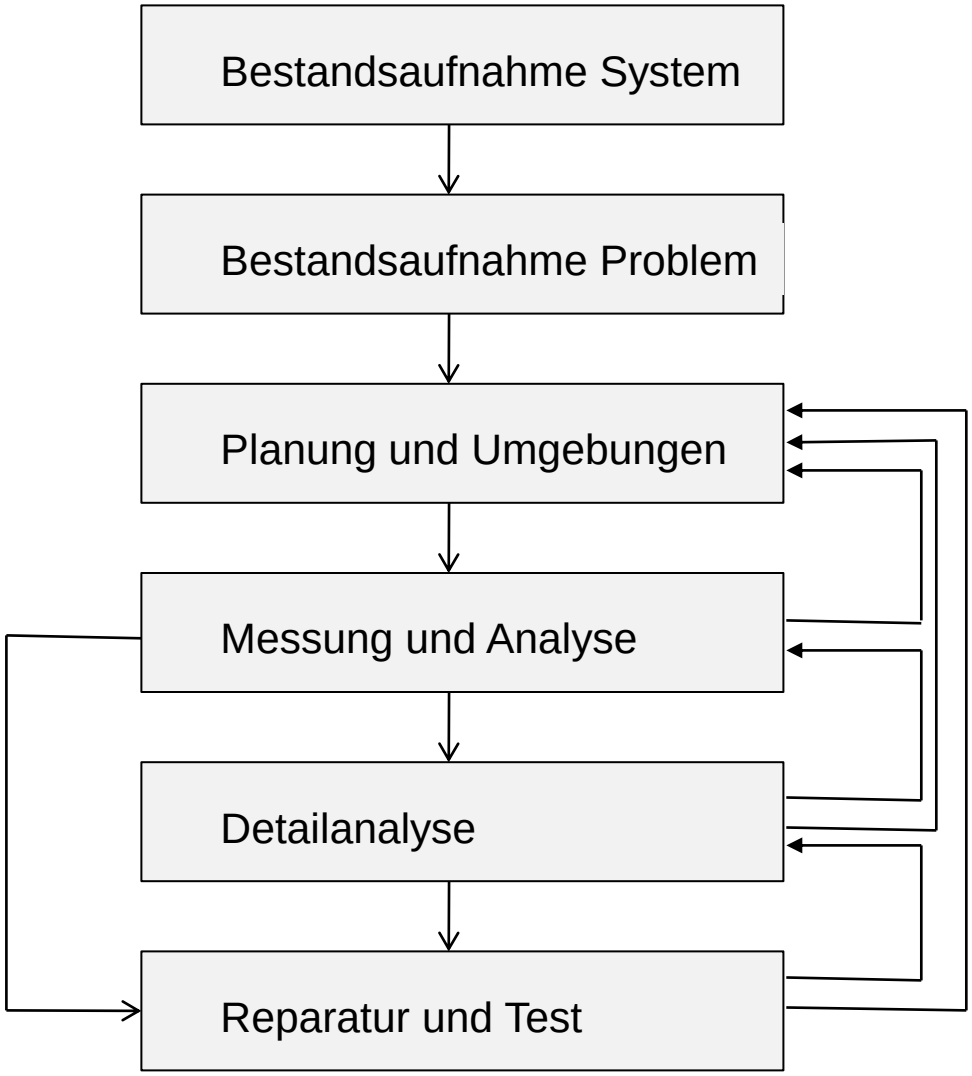
- Rettungsaktion
- Benchmarking
- Optimierung
- Prophylaxe
- Entwicklungsprozess

Vorbeugen ist besser als Heilen

Keine Annahmen über Software-Architektur, Plattform, Vorgehensmodell.

Eine Rettungsaktion (1)

Eine Rettungsaktion (2)



Erfolgsfaktoren:

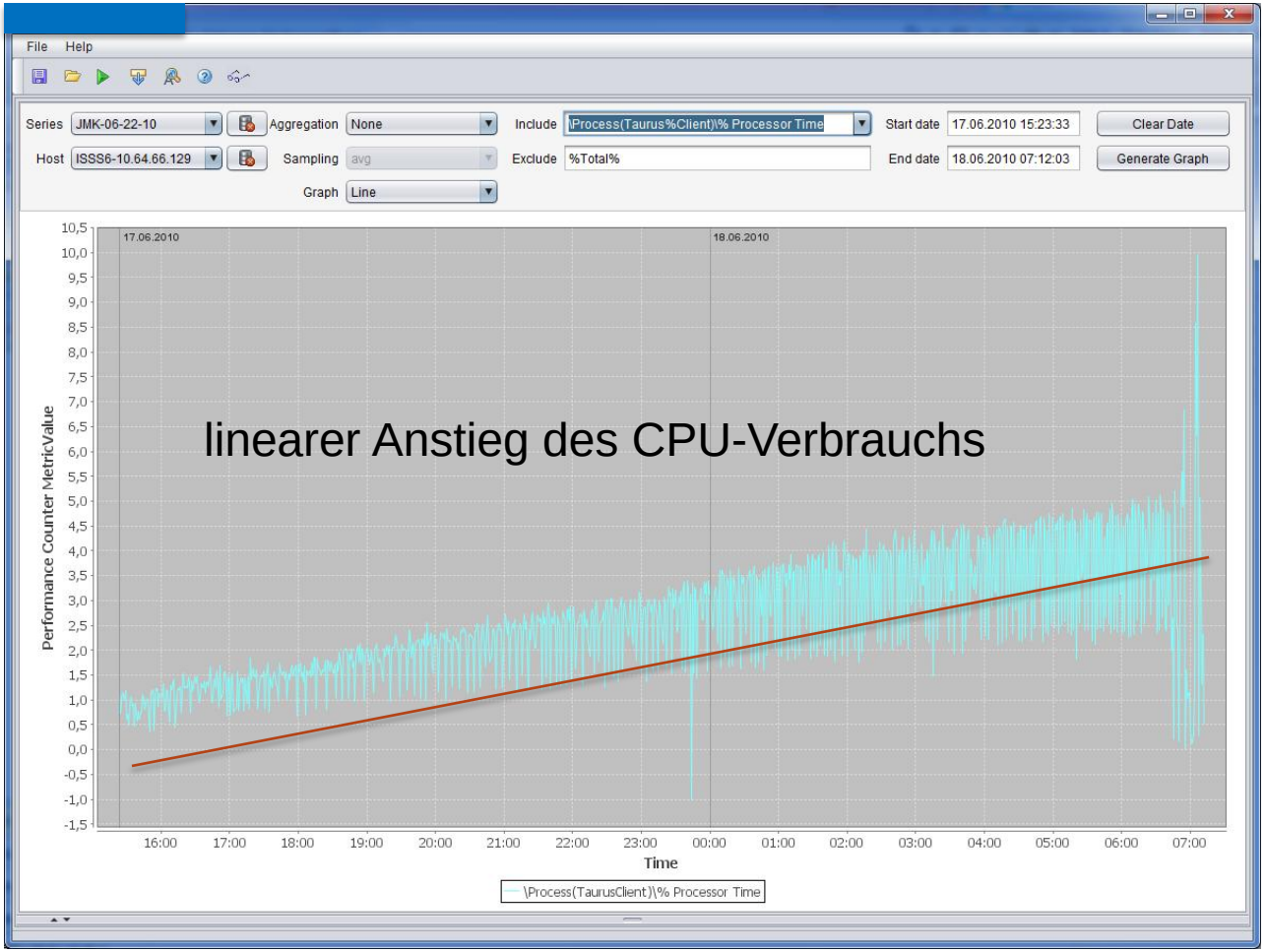
- Rückendeckung durch höheres Management
- Quick Wins, vor allem zu Beginn
- Kooperative Ansprechpartner für alle relevanten Themen
- Kompetenter Analyst
- Brauchbare Umgebungen
  ➔ HW-Virtualisierung

Computer Aided Software Diagnosis (CASD)

1.000 Messreihen über 3 Tage. Wie findet man die interessanten Stellen? Mit der Unterstützung von:

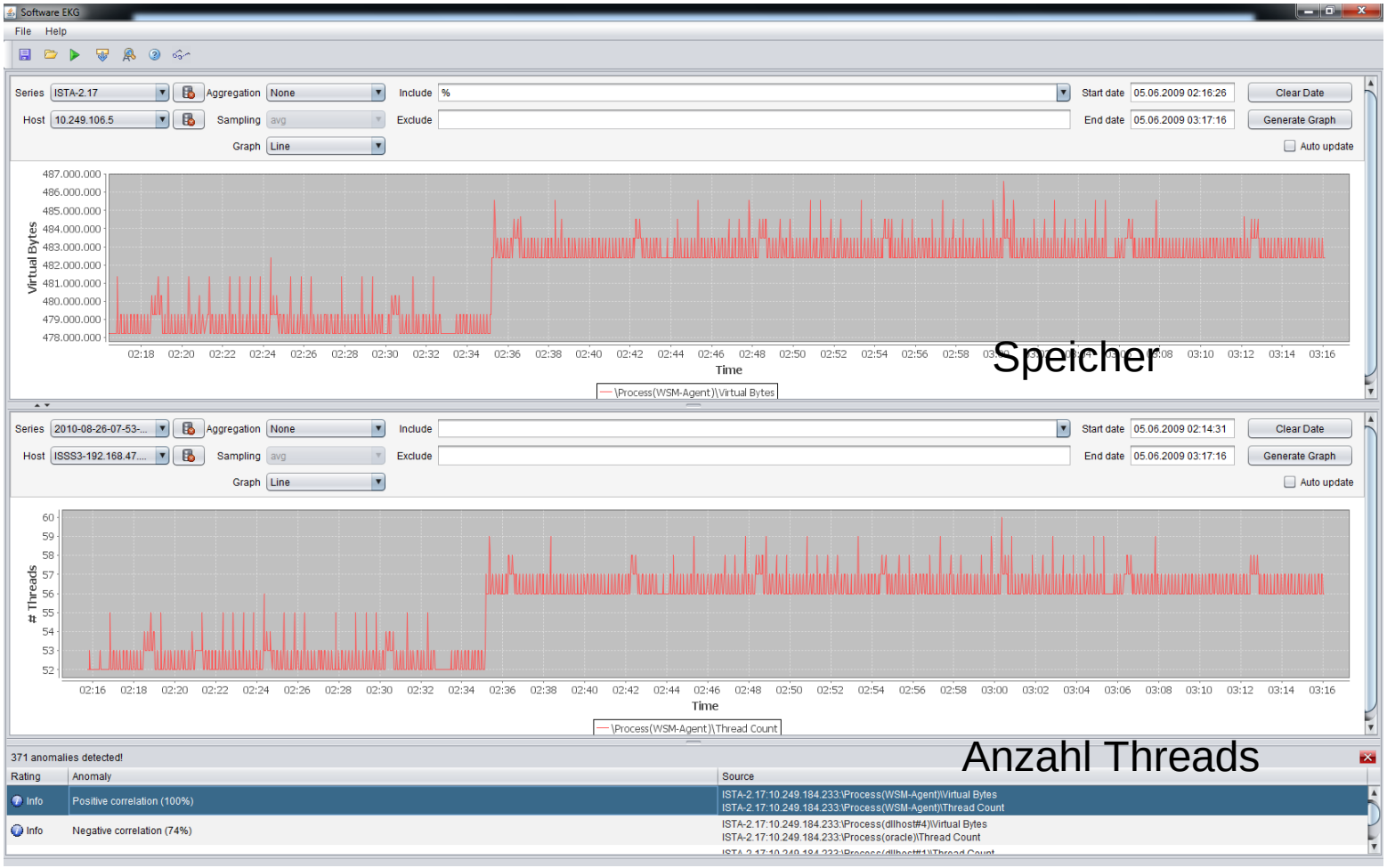
- Korrelationsdetektor
- Trenddetektor
- Frequenzdetektor

Trenddetektor



16:00 7:00

Korrelationsdetektor



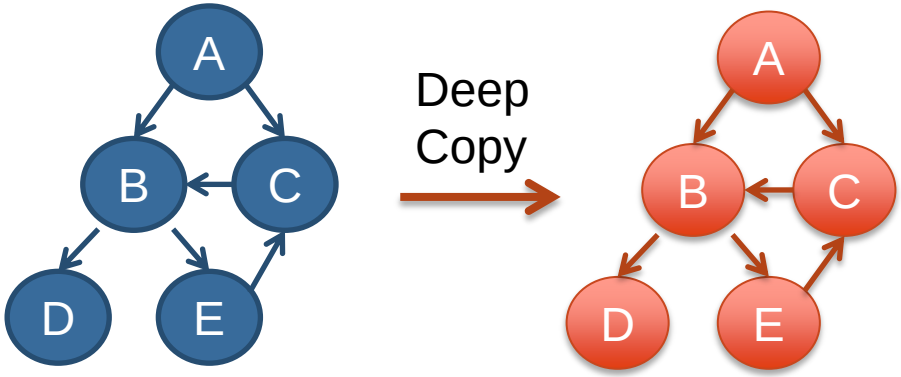
2:16

3:16

Typische Fehlerursachen

- **Leaks** (Speicher, Threads, Handles)
- **CPU-Verbrauch**: Naive Programmierung, Serialisierung (z.B. bei Deep Copy) Garbage Collection.
- **Locks** (Short Locks, Dead Locks): Naive Programmierung.
- **I/O-Last**: Logging, naiver Datenbankentwurf, naiver Datenbankzugriff, Ladezeiten
- **Parameter**: JVM, Datenbank, Betriebssystem, Netzverbindungen

Deep Copy Considered Harmful



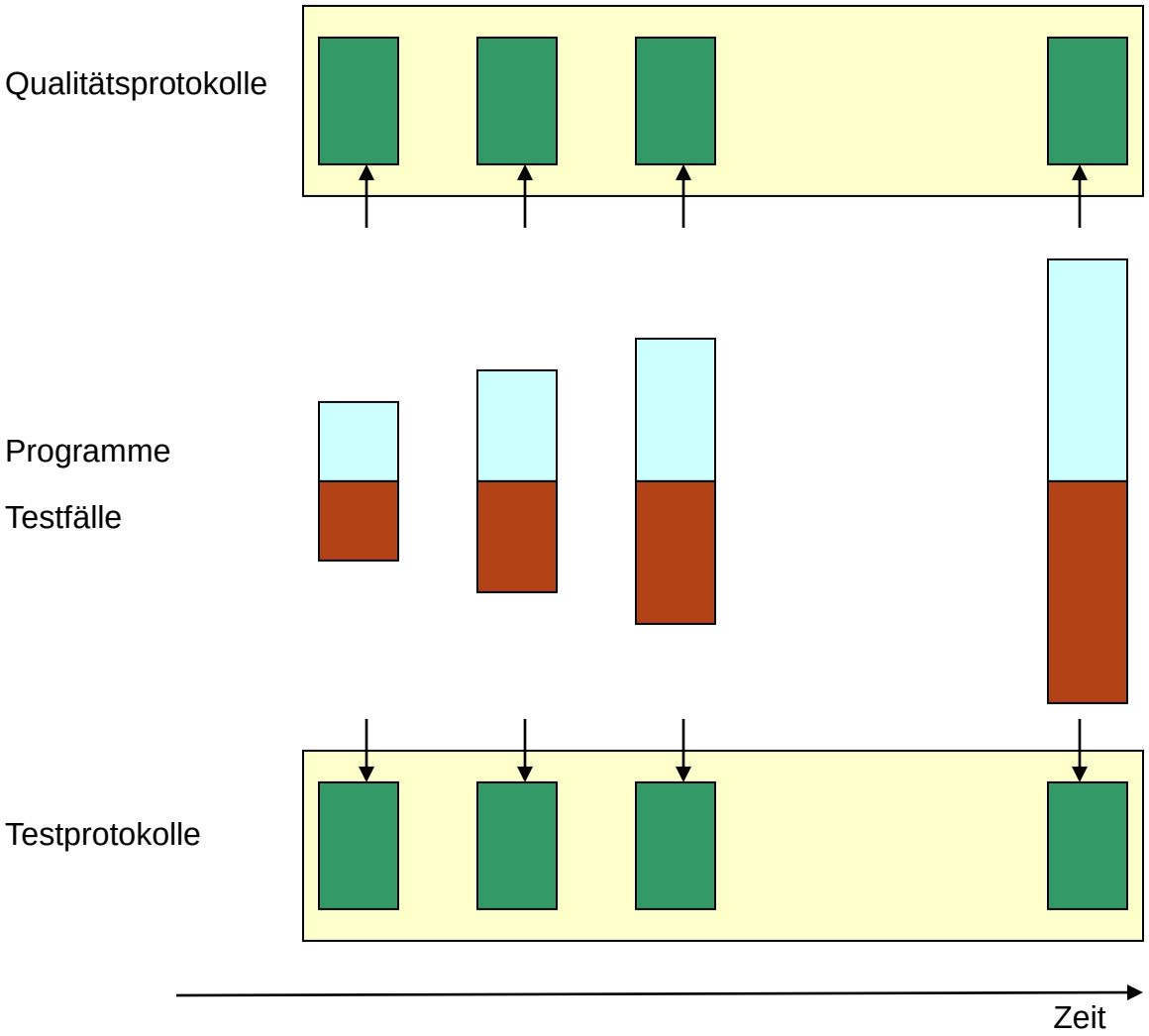

```
// JAVA Code - .NET Code ähnlich
Object p1 = … // im Test: System.getProperties();
ObjectOutputStream os = new ObjectOutputStream(out);
os.writeObject(p1);
ObjectInputStream oi = new ObjectInputStream(
        new ByteArrayInputStream(out.toByteArray()));
Object deepCopy = (Object) oi.readObject();
…
```

No Broken Windows!

Wie beugt man vor?

- Bewusstsein, Aufklärung
- Gezielte QS gegen Flüchtigkeitsfehler, fehlende oder mangelhafte Konzepte, mangelhafte Architektur
- **Software-Controlling** (systematischer Einsatz von EKG und anderen Prüfwerkzeugen in der Entwicklung)
- **Design for Diagnosibility** (DfD)
- **Ausbildung!**

Software Controlling



Statische Analyse

- Code-Qualität
- Metriken
- Komplexität
- Zyklenfreiheit
- Abweichungen von der Konstruktion (Design)
- Abweichungen vom Standard: z.B. Einhaltung der SW-Kategorien

Dynamische Analyse

- Korrektheit
- Testfallabdeckung
- Performance
- Robustheit
- Skalierbarkeit

Naive Logging Considered Harmful

```
2010-12-08 00:00:00,031 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306] ProcessHandlerImpl: [+] CONFIGUREPRINTERSPOOLER
2010-12-08 00:00:00,031 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.configurePrinterSpooler_3306*
2010-12-08 00:00:00,031 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.configurePrinterSpooler_3306*
2010-12-08 00:00:00,062 INFO [ScheduledThread-Manager.configurePxeService_3306] ProcessHandlerImpl: [+] CONFIGUREPXESERVICE
2010-12-08 00:00:00,062 INFO [ScheduledThread-Manager.configurePxeService_3306] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.configurePxeService_3306*
2010-12-08 00:00:00,062 INFO [ScheduledThread-Manager.configurePxeService_3306*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.configurePxeService_3306*
2010-12-08 00:00:00,062 WARN [ScheduledThread-Manager.configurePxeService_3306*] ConfigurePxeService: No PXEBootDevice available!
2010-12-08 00:00:00,062 INFO [ScheduledThread-Manager.configurePxeService_3306*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.configurePxeService_3306* Completed in: 0 ms
2010-12-08 00:00:00,093 WARN [ScheduledThread-Manager.monitorActiveManager_3306] ProcessHandlerImpl: Constraint check failed of process 'MONITORACTIVEMANAGER' (Script: 'stateHandler.isProductive() &&
!stateHandler.isActive() && !profileHandler.isLauncher()')
2010-12-08 00:00:00,093 WARN [ScheduledThread-Manager.monitorActiveManager_3306] ProcessHandlerImpl: Invalid conditions for process execution of 'MONITORACTIVEMANAGER'
2010-12-08 00:00:00,093 INFO [ScheduledThread-Manager.monitorActiveManager_3306] ProcessHandlerImpl: [x] MONITORACTIVEMANAGER
2010-12-08 00:00:00,093 WARN [ScheduledThread-Manager.monitorActiveManager_3306] ProcessHandlerImpl: Executing of process 'MONITORACTIVEMANAGER' failed, because of process conditions failed. Skipping...
2010-12-08 00:00:00,093 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306*] ConfigurePrinterSpooler: Printer spooler is running.
2010-12-08 00:00:00,109 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306*] ConfigurePrinterSpooler: No dublicate inventoried printer found.
2010-12-08 00:00:00,109 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.configurePrinterSpooler_3306* Completed in: 78 ms
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.configurePxeService_3306] ProcessHandlerImpl: [-] CONFIGUREPXESERVICE
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.monitorBusinessHours_16530] ProcessHandlerImpl: [+] MONITORBUSINESSHOURS
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.monitorBusinessHours_16530] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorBusinessHours_16530*
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.monitorBusinessHours_16530*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorBusinessHours_16530*
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.monitorBusinessHours_16530*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.monitorBusinessHours_16530* Completed in: 0 ms
2010-12-08 00:00:00,125 INFO [ScheduledThread-Manager.monitorBusinessHours_16530] ProcessHandlerImpl: [-] MONITORBUSINESSHOURS
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.configurePrinterSpooler_3306] ProcessHandlerImpl: [-] CONFIGUREPRINTERSPOOLER
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060] ProcessHandlerImpl: [+] MONITORUSV
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorUsv_33060*
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorUsv_33060*
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060*] MonitorUsv: UPS Monitoring is disabled - skipping...
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.monitorUsv_33060* Completed in: 0 ms
2010-12-08 00:00:00,140 INFO [ScheduledThread-Manager.monitorUsv_33060] ProcessHandlerImpl: [-] MONITORUSV
2010-12-08 00:00:00,171 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510] ProcessHandlerImpl: [+] MONITORCLUSTERCONFIGURATION
2010-12-08 00:00:00,171 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorClusterConfiguration_5510*
2010-12-08 00:00:00,171 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorClusterConfiguration_5510*
2010-12-08 00:00:00,187 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510*] EventHandlerDispatcherAsyncImpl: Pushing event #22166: IbaseConfigChanged(config=GenericConfiguration[CLUSTERS=[]])
2010-12-08 00:00:00,187 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510*] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before run() of: EventHandler#22166
2010-12-08 00:00:00,187 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.monitorClusterConfiguration_5510* Completed in: 16 ms
2010-12-08 00:00:00,187 INFO [EventHandler#22166] ExecutorServiceProviderImpl: #espi# Before run() EventHandler#22166
2010-12-08 00:00:00,187 INFO [EventHandler#22166] ExecutorServiceProviderImpl: #espi# After run() EventHandler#22166 Completed in: 0 ms
2010-12-08 00:00:00,203 INFO [ScheduledThread-Manager.monitorDhcpServer_2480] ProcessHandlerImpl: [+] MONITORDHCPSERVER
2010-12-08 00:00:00,203 INFO [ScheduledThread-Manager.monitorDhcpServer_2480] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorDhcpServer_2480*
2010-12-08 00:00:00,203 INFO [ScheduledThread-Manager.monitorDhcpServer_2480*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorDhcpServer_2480*
2010-12-08 00:00:00,203 INFO [ScheduledThread-Manager.monitorDhcpServer_2480*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.monitorDhcpServer_2480* Completed in: 0 ms
2010-12-08 00:00:00,203 INFO [ScheduledThread-Manager.monitorDhcpServer_2480] ProcessHandlerImpl: [-] MONITORDHCPSERVER
2010-12-08 00:00:00,218 INFO [ScheduledThread-Manager.monitorNetworkConnectivity_16530] ProcessHandlerImpl: [+] MONITORNETWORKCONNECTIVITY
2010-12-08 00:00:00,218 INFO [ScheduledThread-Manager.monitorNetworkConnectivity_16530] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorNetworkConnectivity_16530*
2010-12-08 00:00:00,218 INFO [ScheduledThread-Manager.monitorNetworkConnectivity_16530*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorNetworkConnectivity_16530*
2010-12-08 00:00:00,218 INFO [ScheduledThread-Manager.monitorNetworkConnectivity_16530*] ExecutorServiceProviderImpl: #espi# After call() ScheduledThread-Manager.monitorNetworkConnectivity_16530* Completed in: 0 ms
2010-12-08 00:00:00,218 INFO [ScheduledThread-Manager.monitorClusterConfiguration_5510] ProcessHandlerImpl: [-] MONITORCLUSTERCONFIGURATION
2010-12-08 00:00:00,250 INFO [ScheduledThread-Manager.monitorPrinterDocuments_1102] ProcessHandlerImpl: [+] MONITORPRINTERDOCUMENTS
2010-12-08 00:00:00,250 INFO [ScheduledThread-Manager.monitorPrinterDocuments_1102] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorPrinterDocuments_1102*
2010-12-08 00:00:00,250 INFO [ScheduledThread-Manager.monitorPrinterDocuments_1102*] ExecutorServiceProviderImpl: #espi# Before call() ScheduledThread-Manager.monitorPrinterDocuments_1102*
2010-12-08 00:00:00,265 INFO [ScheduledThread-Manager.monitorSoftwareEnvironment_16530] ProcessHandlerImpl: [+] MONITORSOFTWAREENVIRONMENT
2010-12-08 00:00:00,265 INFO [ScheduledThread-Manager.monitorSoftwareEnvironment_16530] ExecutorServiceProviderImpl: #espi# Queue-Size: 0 before call() of: ScheduledThread-Manager.monitorSoftwareEnvironment
```

Naive Logging

- Anwendung schreibt in eine Logfile (log4j, Java Logging)
- Log-Level niedrig ➔ zu wenig Info
- Log-Level hoch ➔ System unter Stress, viel zu viel Info
- Regeln für sinnvolles Logging?

Clever Logging

Die Anwendung aktualisiert ihren Zustand in einer Black Box konstanter Größe

Der Kollektor liest die Black Box in festen Abständen

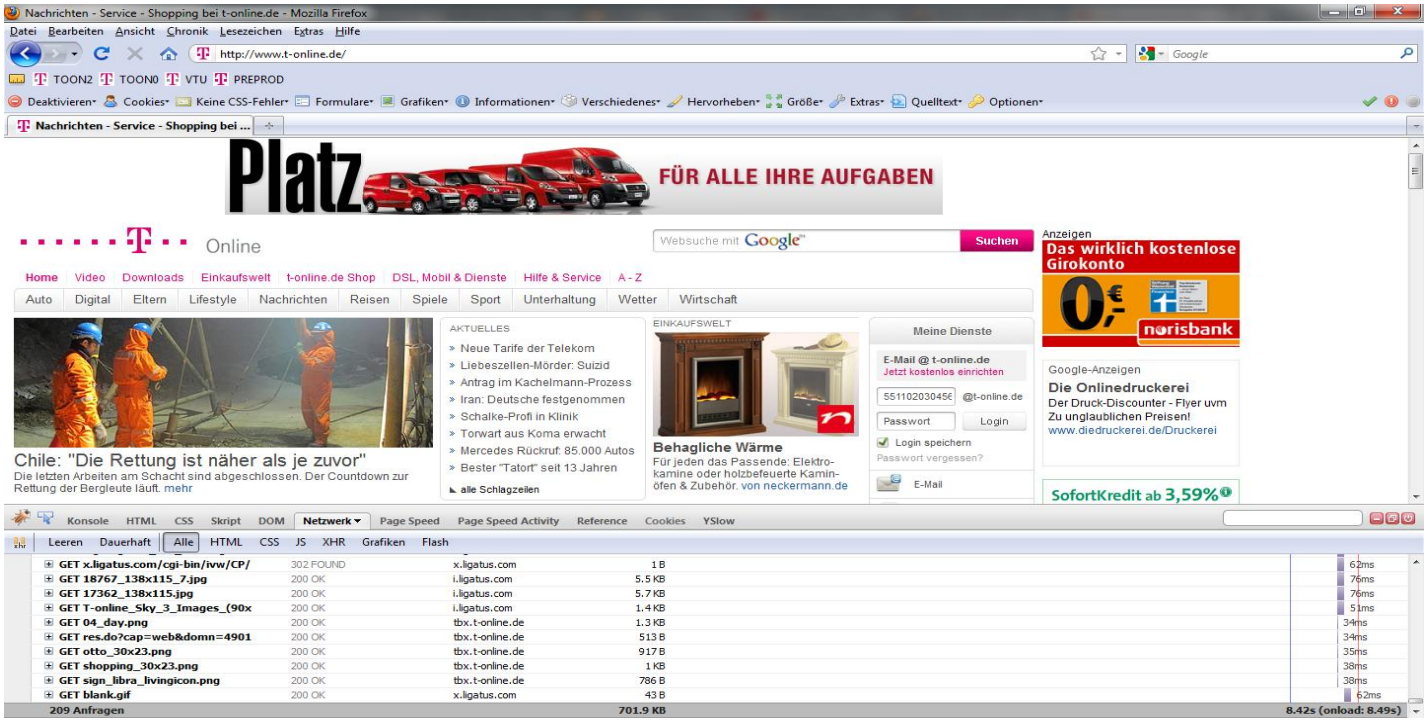
Design for Diagnosibility

Software-EKG von vornherein installieren

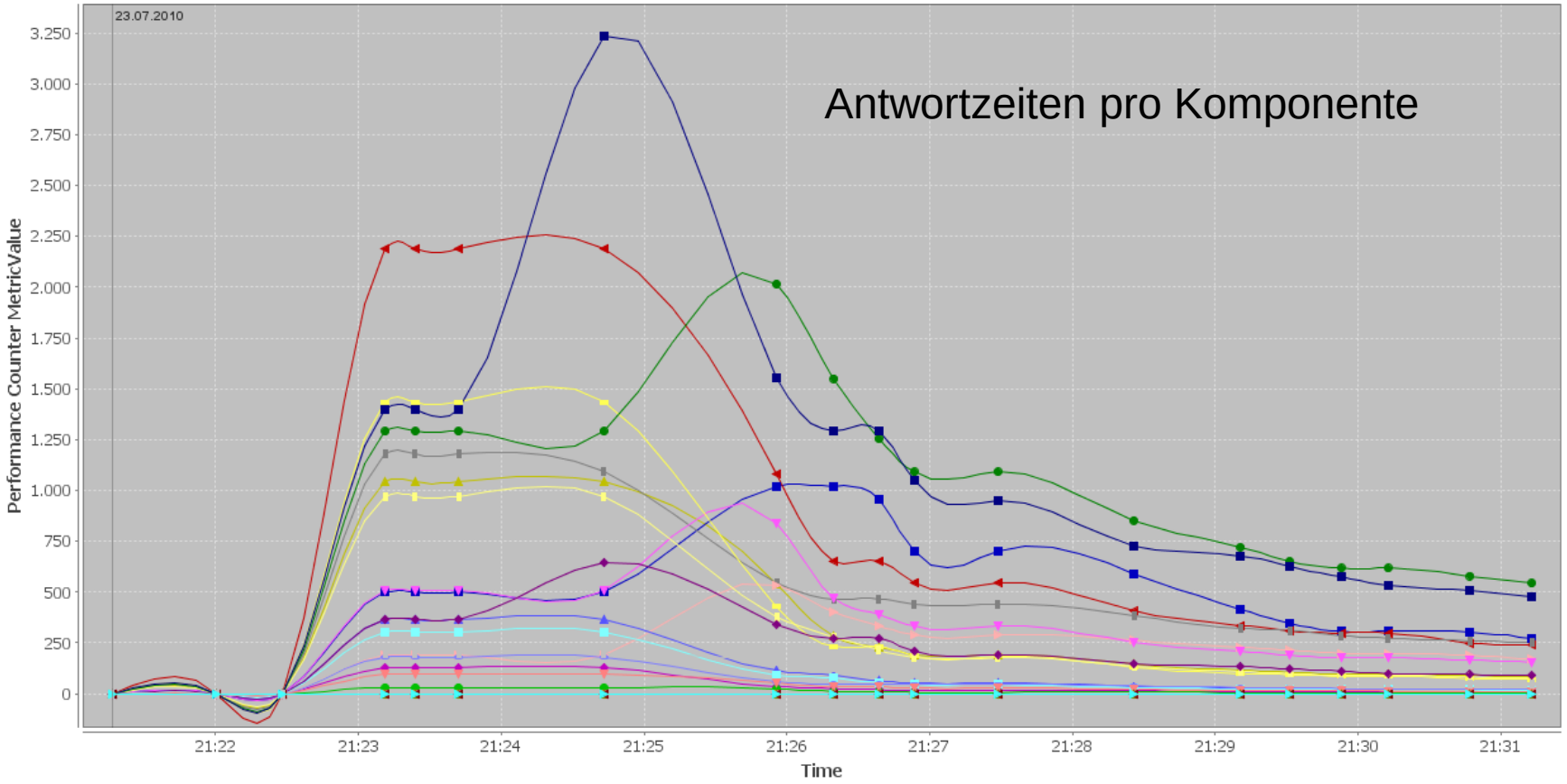
Logging-Konzept

Clever Logging mit Black Box

Ladezeit von 8,42 auf 5,32 Sekunden reduziert



Elefantenkurven



Nachzulesen in:

Dynamische Analyse mit dem Software-EKG
INFORMATIK-SPEKTRUM
Volume 34, Number 5 (2011), 484-495.

Johannes Weigend
Johannes Siedersleben
Josef Adersberger

Die vier Gebote für performante Software

1. **Gebot:** Optimiere niemals ohne zu messen
   (We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil** – Donald Knuth)

2. **Gebot**: Beachte die Komplexität von Algorithmen (Achtung bei $O(n^2)$ oder schlechter)
3. **Gebot:** Kontrolliere den Ressourcenverbrauch.
4. **Gebot**: Messe regelmäßig und optimiere gezielt.

Qualität und Agilität im Software Engineering

www.qaware.de