Testen in der Praxis

DI Christian Federspiel

Process Explorer - [Stranggießanlage]

Datei Anzeige Aktion Alarme Notlauf Fenster Hilfe

voestalpine VAI

VAICC Stranggießapplikation ld3cc4

Reini

Charge		Status	Sequenz	Gew	Pfa	Offen	Planzeile	Kühlpr	Werksmarke
835038		DTA		166,3	77		0823-029	550	47A5M21214Z#E
		V1					0823-030	550	35A5M26XX
734631		GIS	6783/75	167,4	68	07:43	0823-027	550	47A5M21214Z#E
		PRO	11787/75				0823-028	550	38A5M23ZYSI2#A
734630		SCH	6783/74	159,9	61	07:08	0823-025	550	35A5M26XX
		PRO	11787/74				0823-026	550	47A5M21214Z#E
835036		FER	6783/73	160,6	64	06:34	0823-023	550	49A4M21212ZSI2#E
		PRO	11787/73				0823-024	550	47A4M25ZZSI2#E

Strang A

PLC | Wechsel aktiv | Vorgabe gesperrt | L2

Pos [m]	Län	Start	Ende	Konus	Md	A	M	Spr
2800,64	2743	1277	1253	1,0 %	L1	✓		
2740,10	2500	1304	1277	1,0 %	S		L2	✓

Pfanne in Ladeposition

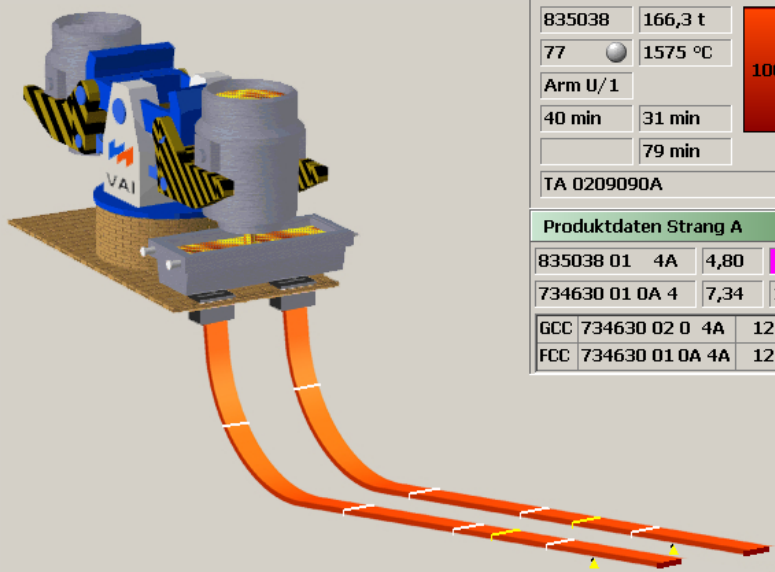
734631	3,0 t	2 %
68	1580 °C	
Arm O/2		

Verteiler

Ost/1	52	63 %
22,1 t	412 min	
1559 °C	07:50	
	:	
1553 °C	1532 °C	21 °C
RHA.TTC6		
GLUT. SP/CA-2		
RHA.TTC6		

Produktdaten Strang B

734631 23 4B	6,33	L2	
734630 21 0A 4	5,77	12,13	
GCC	734630 22 0 4B	10,05 m	
FCC	734630 21 0A 4B	12,08 m	



Pfanne in Gießposition

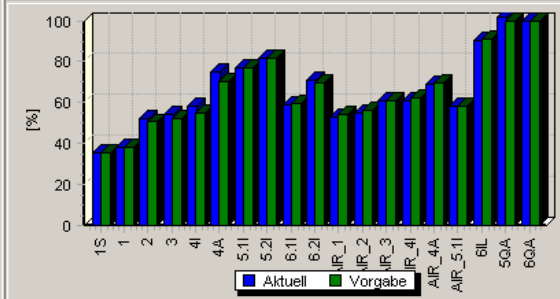
835038	166,3 t	100 %
77	1575 °C	
Arm U/1		
40 min	31 min	
	79 min	
TA 0209090A		

Produktdaten Strang A

835038 01 4A	4,80	L2	
734630 01 0A 4	7,34	12,24	
GCC	734630 02 0 4A	12,21 m	
FCC	734630 01 0A 4A	12,02 m	

S Kühl A | Oszi | P Kühl | VKK A

VASL4 -- B Kühlungswerte Bar-Grafik



S Kühl B | VKK B

Gießrohr		Kokille					Strang					Kühlung			
Typ	min	Nr	Format	Tief	Ab	Pulver	Vg	t/min	L(r)	L(t)	L(c)			Kurve	L1
TA 0306011 Typ F Cfreier	115	49	1277x215	149	1	META.SPH SL 470/	1,01	2,1	53,63	2771,9	0,00	✓	L2	VASL4_550	550
TA 0306011 Typ F Cfreier	124	42	1286x215	150	0	META.SPH SL 470/	1,01	2,1	52,83	2777,4	0,00	✓	L2	VASL4_550	550

05.11.2004 08:10:17 | Secondary Cooling: setting coolingPractice 550 for strandNr = 1

1.2004.1102.1 | ld3cc4 | 0 Watchdogs | 0 Alarme | 0 Defekte | MS | SDÜ | Neue Analyse | Server: 192.168.42.120; Port: 2070 | 08:13

Start | Process Explorer - [Str... | 08:13

Qualität der Fehlermeldungen 1

- Fehler: Charge 928120 hatte keine PfannenNr (von SekMet). Beladen von L1 ging nicht. Danach das Löschen/Erzeugen angeblich auch nicht
- Mängel: Zeiten am TTI Telegram Viewer

Qualität der Fehlermeldungen 2

- Heute um ca. 2:00 wurde die Charge 734586 auf der CC6 heringedreht. Die L1 Signale waren (angeblich) OK aber am L2 wurde die Pfanne nicht angezeigt. Der Bediner hat die Turret-in-Cast-Posuion und Pfanne-gealden Signale gesetzt (bei zweiteren musste er 2 mal klicken weil es nicht angenommen wurde !!!!!) worauf alles OK war. Er lies die Signale überschrieben (hab ich selbst gesehn, L1 Signale waren dann auch OK) aber nach 2min war die Charge weg und wir hatten stattdessen eine E-Charge im Giessen

VAI: Kosten von Fehler

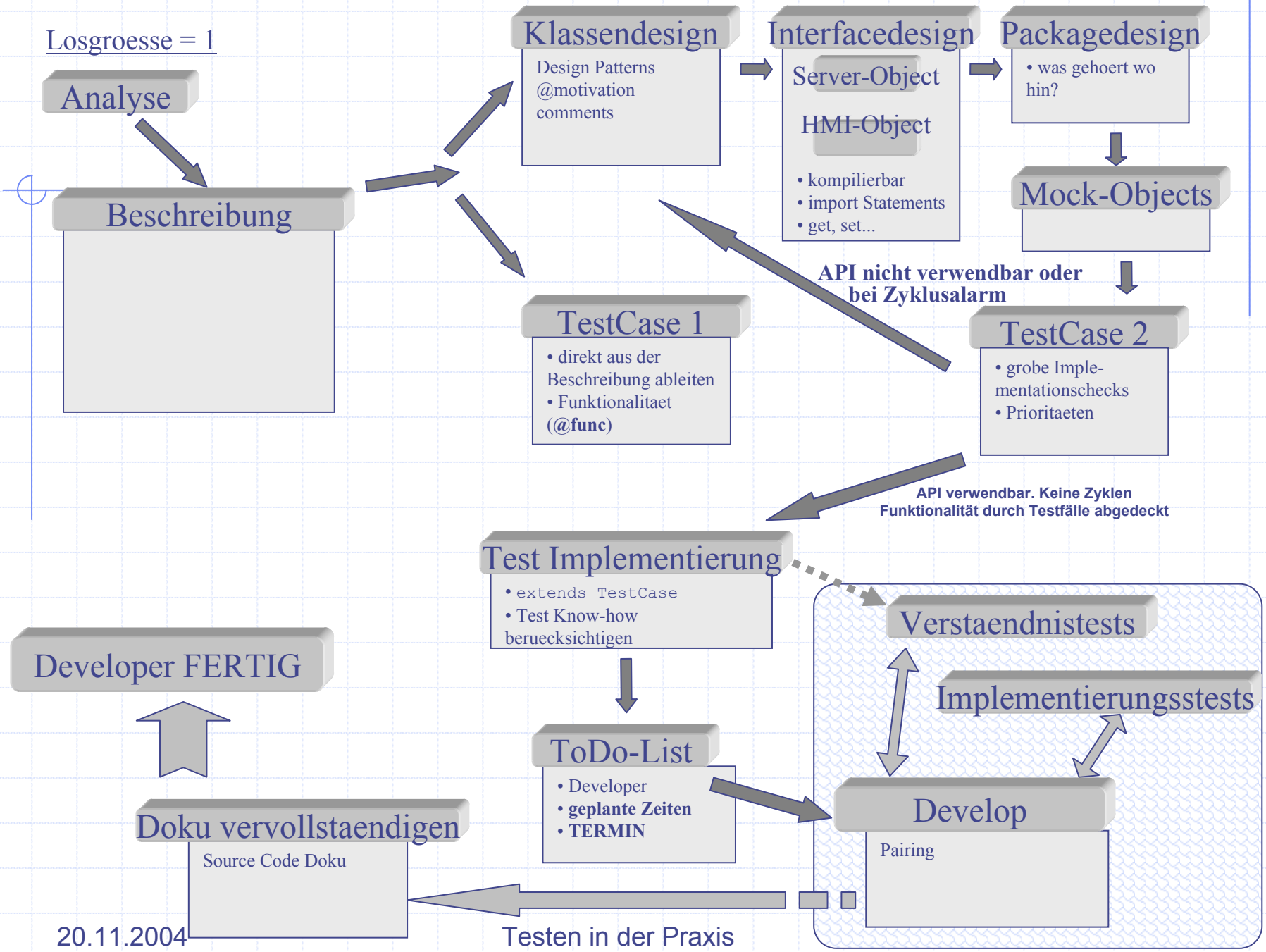
- Verteilte Systeme:
  - Durchschnitt: 8000 € / Fehler nach Auslieferung
  - Schwere Fehler: ca. 21.000 €
  - Imageschaden
    - Schaden beim Kunden meist um vieles höher!
  - Kunde mischt sich ein
    - Forciert seine Lösung, ohne ausreichendes Hintergrundwissen

➔ Architekturziel: Testbarkeit jeder Komponente

Mehrstufiges Testkonzept

- Automated Unit Testing
- Fehler Pattern Finder
  - FindBugs, PMD, JLint, JMetric, Sotograph…
  - Auf das Design abgestimmte selbstgeschriebene Tools
- Small Scale Simulators
  - Dynamic
  - Script file based
  - replay
- „Real World“- Simulator
- Simulation/Test Mode der Anlage

Erfahrung, Anforderungen


Losgroesse = 1
Analyse
Beschreibung
Klassendesign
Design Patterns
@motivation
comments
Interfacedesign
Server-Object
HMI-Object
• kompilierbar
• import Statements
• get, set...
Packagedesign
• was gehoert wo hin?
Mock-Objects
API nicht verwendbar oder bei Zyklusalarm
TestCase 1
• direkt aus der Beschreibung ableiten
• Funktionalitaet (@func)
TestCase 2
• grobe Imple-mentationschecks
• Prioritaeten
API verwendbar. Keine Zyklen
Funktionalität durch Testfälle abgedeckt
Test Implementierung
• extends TestCase
• Test Know-how beruecksichtigen
Verstaendnistests
Implementierungsstests
Developer FERTIG
ToDo-List
• Developer
• geplante Zeiten
• TERMIN
Develop
Pairing
Doku vervollstaendigen
Source Code Doku
20.11.2004
Testen in der Praxis

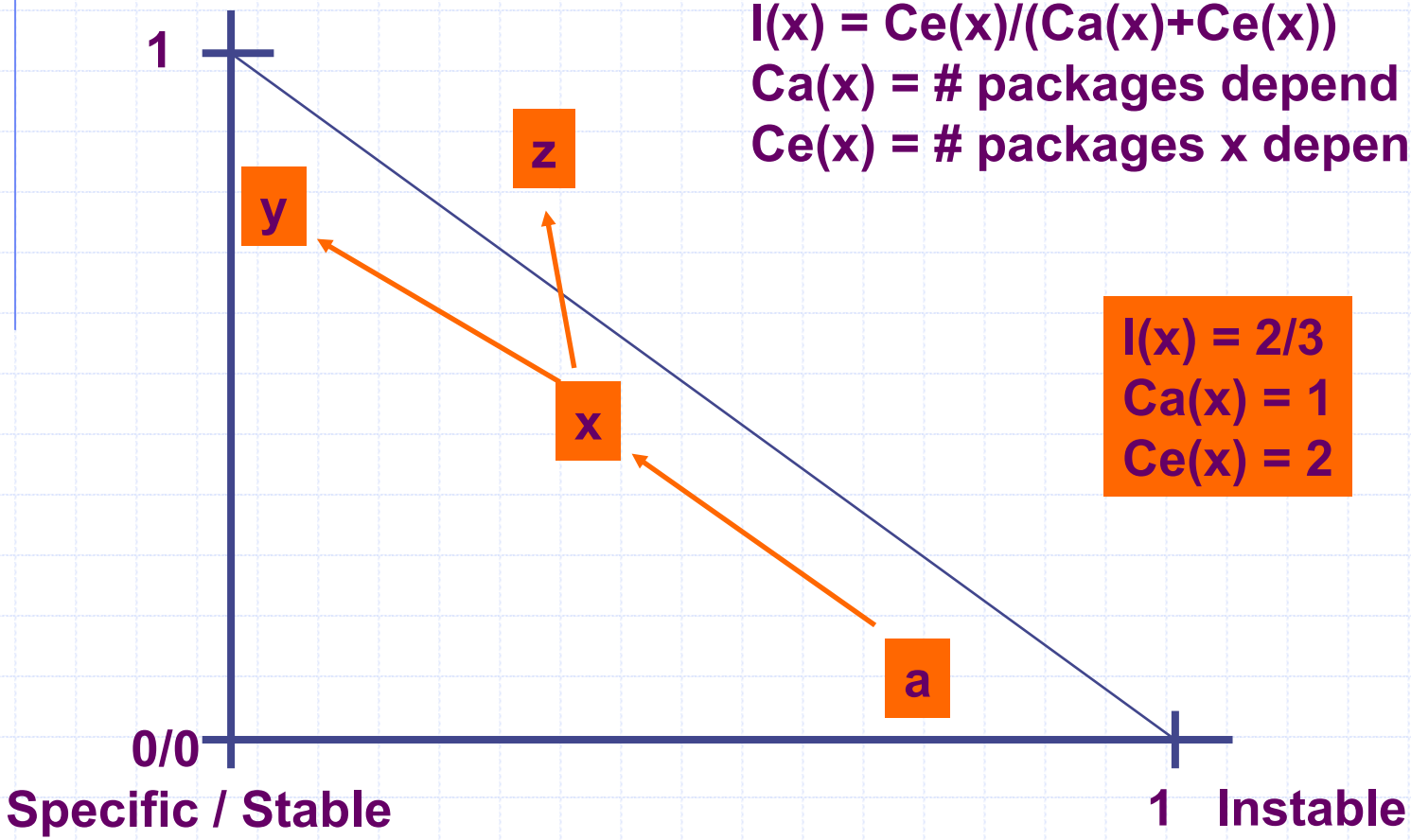
Architektur 1

- Steif = kleine Änderungen an einer Stelle ➔ viele Änderungen in abhängigen Modulen nach sich
- Zerbrechlich = Änderungen an einer Stelle ➔ vielen anderen Stellen neue Fehler. Geht einher mit der Steifheit
- Zäh = Mehrere Wege im System Änderungen vorzunehmen – designkonform oder Hack
- Unbeweglich = Eingeschränkte Wiederverwendung von Komponenten ➔ führt zu copy & paste

Ursache: Schlechte oder fehlende Architektur

Architektur 2

Abstract = #Interfaces / #all classes



I(x) = Ce(x)/(Ca(x)+Ce(x))
Ca(x) = # packages depend upon x
Ce(x) = # packages x depends on

I(x) = 2/3
Ca(x) = 1
Ce(x) = 2

Design Pattern

- Facade + Clover ➔ White box testing
- State+Mediator:
  - ```
    private void runException (MyRunnable r) {
      try {  r.run();  } catch (Exception e) {  return;  }
       fail(); //something went wrong
       throw new RuntimeException ("No Exception thrown");
    }
    ```
  - Test auf unerlaubten Zustand

  ```
  runException (new MyRunnable() { public void run() throws
    Exception { obj.callIllegalState();}});
  ```
- Listener
  - Wieviel Information steckt im Event? Komfort der Schnittstelle?
- Value Object
  ```

VAI-Zahlen für größere Projekte

- Unit-Testabdeckung: 30% - 50% vom Code
- Verhältnis Code zu Unit-TestCode:
  - Ca. 5 : 1 ohne Tools (Testfälle generieren)
    - Grober Richtwert für Management
- Änderungen:
  - Neue Release: Fehler pro 100 Änderungen
  - Ohne Unit-Tests: durchschnittlich 7 Fehler
  - Mit Unit-Test: durchschnittlich 2 Fehler

Erstellen einer Release

- Feature Freeze
  - 80% und stabil
  - Code Review
    - Tool-support - Sotograph
  - Tester
    - Budget- oder Fehlerrate gesteuert
    - Bugs in BugTracker (Mantis, Bugzilla, …)
- HMI: String Freeze
  - Doku
  - Übersetzungen

Laufzeit

Fehlermöglichkeit muss akzeptiert werden:

- Normale SW: 25 Fehler/1000 loc
- Wichtige SW: 2-3 Fehler/1000 loc
- Medizinische SW: 0.2 Fehler / 1000 loc
- Raumfahrt SW: < 0.1 Fehler / 1000 loc

- Recovery-fähig
- Überwachung durch einfachere Algorithmen/Modelle
- Schranken, wie oft ein Algorithmus aufgerufen werden darf
- Aspekte ?

Überprüfungen zur Laufzeit

- Plattenplatz, Memory, CPU-Load
- DB: Tablespaces, Locks
- Anzahl der Threads
- Timeout bei Methodenaufruf
- Automatischer Restart von Prozessen nach Absturz

Reaktion?

Formale Verifikation - JML

- Ansätze gut
- Derzeit nicht praxistauglich – zu aufwendig
- Problem: Formaler Code ist meist komplizierter als Code selbst

Developer with built in Quality