ISIS

beleuchten, was sich bisher im Dunkeln verbarg
verbessern, was bisher im Argen lag

Inhalt

- Was ist Softwarequalität
- Was ist ISIS
- Die Philosophie dahinter
- Die Innovation
- Werkzeuge
  - → Projektlogbuch
  - → Sotograph oder STAN
  - → BugCollector
  - → Clover oder EclEmma
- Die Wirkung
- Maßnahmen zur Prozessoptimierung
- Voraussetzungen für den sinnvollen Einsatz

Was ist Softwarequalität

- Die Qualität eines Softwaresystems besteht aus der
  - → *Äußeren Qualität*: der Funktionalität, der Korrektheit, der Bedienbarkeit, der Performanz (Sicht des Benutzers)
  - → *Inneren Qualität:* Verständlichkeit, Wartungs-, Erweiterungs- und Änderungsfreundlichkeit (Sicht des Entwicklers)
- Üblicherweise liegt der Fokus auf der äußeren Qualität, die innere Qualität wird sträflich vernachlässigt
  - → Wesentliche Ursache für die Softwarekrise
  - → Hohe Kosten bei Änderungen/Erweiterungen

Die Qualität eines Softwaresystems resultiert aus der Qualität des Entwicklungsprozesses

ISIS – eine Erfolgsgeschichte

- ISIS ist eine Methode zur Steuerung von Qualität und Effizienz in Software-Entwicklungsprojekten
- Wurde von andrena auf Basis langjähriger Prozesserfahrungen 2006 konzipiert
- Wurde Anfang 2007 in einem Pilotprojekt eingesetzt, optimiert und evaluiert
- Aufgrund des großen Erfolges hat andrena mittlerweile ISIS in allen internen Projekte eingeführt

ISIS: Das Ziel

Erzeugen von Software angemessener Qualität
bei gleichzeitig hoher Produktivität

ISIS: Die Philosophie dahinter

- Softwarequalität ist eine Ganzheit: innere und äußere Aspekte bedingen einander. Die Definition einer angemessenen Softwarequalität muss daher auf Metriken für beide Teile beruhen
- Um praktikabel und effizient zu sein, muss die Messung der Qualität
  - → auf wenigen,
  - → möglichst aussagekräftigen und
  - → leicht zu erhebenden

  Indikatormetriken beruhen
- Wichtig: Metriken sind immer nur eine beschränkte Sicht auf das Ganze

ISIS: Die Innovation

ISIS ist ein Vorgehensmodell aus der Verfahrenstechnik (Betrieb von Durchflussreaktoren) übertragen auf die Softwareproduktion:

1. Definition einer angemessenen Produktqualität als Zielgröße
2. Regelmäßiges Messen der Qualität
3. Feedback in den Prozess
4. Prozess und Produkt optimieren

ISIS: Prozessphasen

- Optimierung des Prozesses bis die Zielgröße stabil erreicht ist
  - → Primärer Fokus auf qualitätsverbessernden Maßnahmen
- Optimierung des Prozesses unter Halten der Zielgröße
  - → Primärer Fokus auf produktivitätsverbessernden Maßnahmen
- Stabiler Betrieb des Prozesses, Detailoptimierung
  - → Primärer Fokus auf Stabilität

ISIS: Werkzeuge

1. ProjektLogbuch (*andrena Eigenentwicklung)*
2. Automatisierte statische Analyse
   - → *STAN oder*
   - → *Sotograph (Lizenz)*
3. BugCollector (*andrena Eigenentwicklung*)
4. Testabdeckung
   - → *Clover (Lizenz) oder*
   - → *EclEmma (Freeware)*

ProjektLogbuch: Indikatormetriken

Kundenzufriedenheit	17 %
Anzahl der Programmierfehler	15 %
Schätzabweichung	11 %

Testabdeckung	13 %
Packages in Zyklen	11 %
Cyclomatic Complexity (Anzahl Methoden >5)	10 %
Average Component Dependency	9 %
Klassen größer 20 Methoden	6 %
Methoden größer 15 LOC	6 %
Compiler Warnungen	2 %

ProjektLogbuch: zentrales ISIS-Werkzeug

- Integriert:
  - → die Basiswerkzeuge und Metriken
- Verdichtet:
  - → Softwaremetriken zum Softwarequalitätsindex (SQI) als Maß für die Qualität des Softwaresystems
  - → Prozess- und Softwaremetriken zum Prozessqualitätsindex (PQI) als Maß für die Qualität des Prozesses
- Dokumentiert und visualisiert:
  - → für das Team und für Auftraggeber und Management

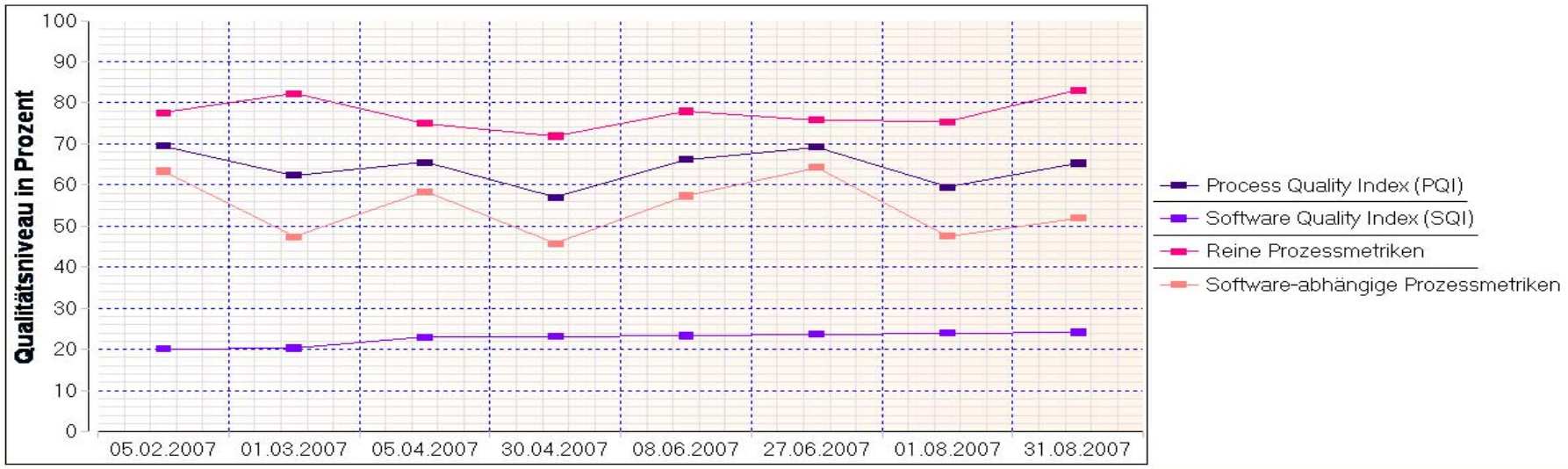
Das ProjektLogbuch ermöglicht transparente Produktion

Projektlogbuch: Beispiel aus der Produktion

Projekt Logbuch

Datei Messpunkt Ansicht Tools Hilfe

Datum	PQI	SQI	KuZ	%	Bugs	%	TA	%	PC	%	SAW	%	CC	%	ACD	%	KG	%	ML	%	CW	%	Beme
03.01.07	0	19	0	0	0	0	15.56	0	195	0	0	0	1105	0	35.13	0	401	0	5618	0	2733	0	neu
05.02.07	70	20	8	80	5	76	16.15	68	204	0	34	76	1103	100	26.58	100	406	100	5744	39	2608	0	neu
01.03.07	62	20	8	80	1	92	16.95	100	210	0	41	72	1109	51	26.42	23	408	78	5804	19	1338	47	neu
05.04.07	66	23	8	80	4	69	18.35	100	213	42	35	75	1114	47	26.31	24	409	72	5825	41	116	100	neu
30.04.07	57	23	7	70	5	60	18.59	58	213	18	12	91	1113	70	26.12	38	413	36	5845	34	111	100	neu
08.06.07	66	23	8	80	3	75	18.96	77	213	11	29	79	1110	100	26.46	0	417	82	5851	75	110	100	neu
27.06.07	69	24	7	70	2	78	19.73	100	212	45	24	82	1106	97	26.97	0	418	69	5859	49	93	100	neu
01.08.07	59	24	8	80	4	81	20.6	98	215	0	61	61	1109	59	27.54	0	421	54	5863	54	84	100	
31.08.07	65	24	8	80	1	91	20.89	67	209	83	32	77	1108	55	27.48	22	423	22	5863	39	85	31	3.3 ec



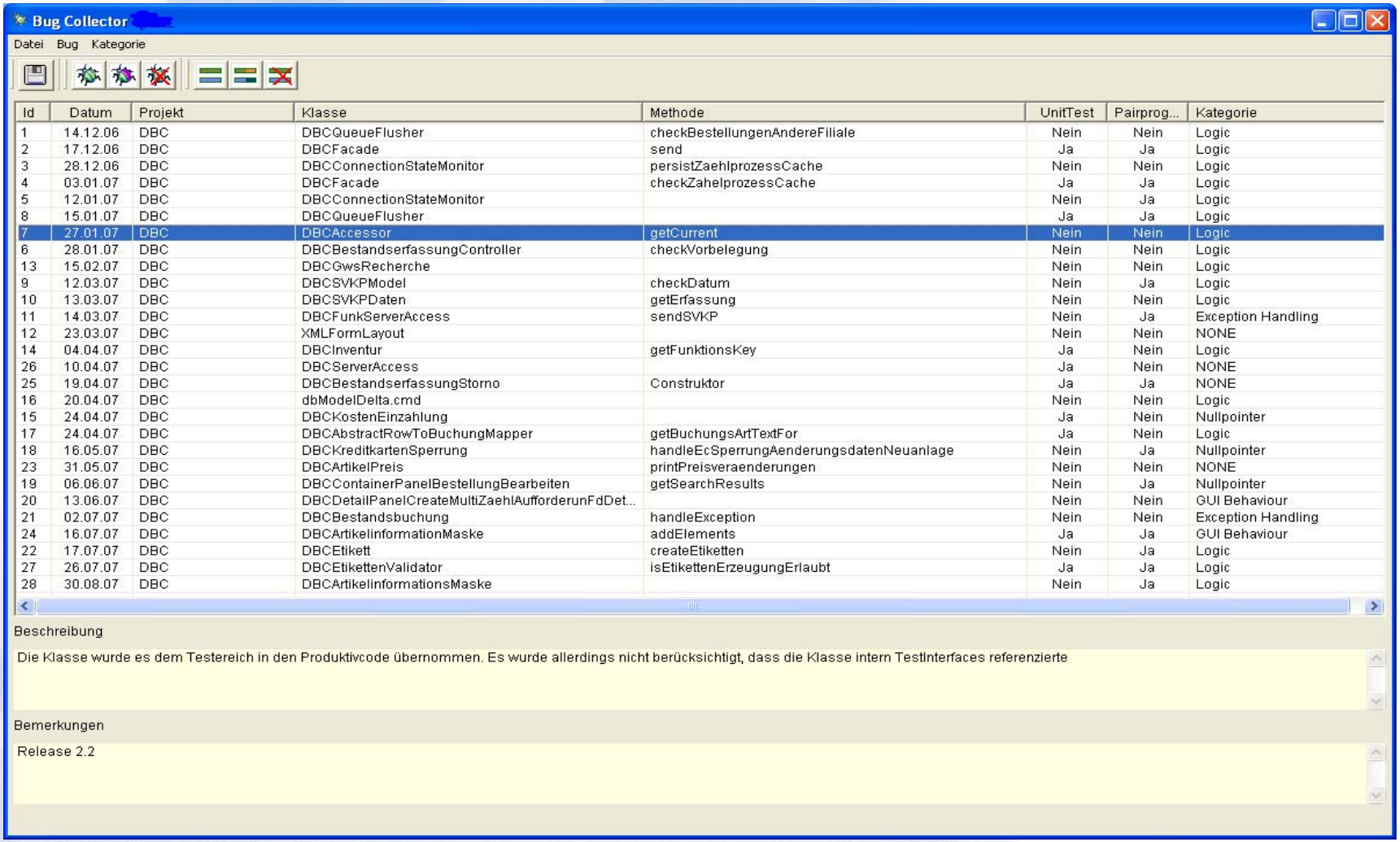
Statische Code-Analyse: Alternativen

- STAN
  - → Leichtgewichtig
  - → Vollständig in das ProjektLogbuch für die automatisierte Analyse integriert (auch als Standalone zu erwerben)
- Sotograph
  - → Das mächtigste und stabilste Tool auf dem Markt
  - → Wertvoll für die manuelle Sichtung des Codes und zur Kontrolle von Änderungen über Delta-Analyse

BugCollector

- Zweck: Sammeln und Analysieren von Programmierfehlern (Bugs), um aus ihnen zu lernen. Vermeidung von Verdrängung
- Definition Bug: von der Spezifikation abweichendes Verhalten in Code, welcher vom Entwicklungsteam freigegeben wurde
- Wichtig: Keine individuelle Zuordnung von Fehlern
- Erkenntnisse aus unserem Projekt: häufig Fehler bei alleiniger Programmierung und/oder fehlenden automatisierten Tests

BugCollector: Beispiel aus der Produktion



Id	Datum	Projekt	Klasse	Methode	UnitTest	Pairprog...	Kategorie
1	14.12.06	DBC	DBCQueueFlusher	checkBestellungenAndereFiliale	Nein	Nein	Logic
2	17.12.06	DBC	DBCFacade	send	Ja	Ja	Logic
3	28.12.06	DBC	DBCConnectionStateMonitor	persistZaehlprozessCache	Nein	Nein	Logic
4	03.01.07	DBC	DBCFacade	checkZahelprozessCache	Ja	Ja	Logic
5	12.01.07	DBC	DBCConnectionStateMonitor		Nein	Ja	Logic
8	15.01.07	DBC	DBCQueueFlusher		Ja	Ja	Logic
7	27.01.07	DBC	DBCAccessor	getCurrent	Nein	Nein	Logic
6	28.01.07	DBC	DBCBestandserfassungController	checkVorbelegung	Nein	Nein	Logic
13	15.02.07	DBC	DBCGwsRecherche		Nein	Nein	Logic
9	12.03.07	DBC	DBCSVKPModel	checkDatum	Nein	Ja	Logic
10	13.03.07	DBC	DBCSVKPDaten	getErfassung	Nein	Nein	Logic
11	14.03.07	DBC	DBCFunkServerAccess	sendSVKP	Nein	Ja	Exception Handling
12	23.03.07	DBC	XMLFormLayout		Nein	Nein	NONE
14	04.04.07	DBC	DBCInventur	getFunktionsKey	Ja	Nein	Logic
26	10.04.07	DBC	DBCServerAccess		Ja	Nein	NONE
25	19.04.07	DBC	DBCBestandserfassungStorno	Construktor	Ja	Ja	NONE
16	20.04.07	DBC	dbModelDelta.cmd		Nein	Nein	Logic
15	24.04.07	DBC	DBCKostenEinzahlung		Ja	Nein	Nullpointer
17	24.04.07	DBC	DBCAbstractRowToBuchungMapper	getBuchungsArtTextFor	Ja	Nein	Logic
18	16.05.07	DBC	DBCKreditkartenSperrung	handleEcSperrungAenderungsdatenNeuanlage	Nein	Ja	Nullpointer
23	31.05.07	DBC	DBCArtikelPreis	printPreisveraenderungen	Nein	Nein	NONE
19	06.06.07	DBC	DBCContainerPanelBestellungBearbeiten	getSearchResults	Nein	Ja	Nullpointer
20	13.06.07	DBC	DBCDetailPanelCreateMultiZaehlAufforderunFdDet...		Nein	Nein	GUI Behaviour
21	02.07.07	DBC	DBCBestandsbuchung	handleException	Nein	Nein	Exception Handling
24	16.07.07	DBC	DBCArtikelinformationMaske	addElements	Ja	Ja	GUI Behaviour
22	17.07.07	DBC	DBCEtikett	createEtiketten	Nein	Ja	Logic
27	26.07.07	DBC	DBCEtikettenValidator	isEtikettenErzeugungErlaubt	Ja	Ja	Logic
28	30.08.07	DBC	DBCArtikelinformationsMaske		Nein	Ja	Logic

Beschreibung

Die Klasse wurde es dem Testereich in den Produktivcode übernommen. Es wurde allerdings nicht berücksichtigt, dass die Klasse intern TestInterfaces referenzierte

Bemerkungen

Release 2.2

Testabdeckung: Alternativen

- Clover
  - → Lizenz
- EclEmma
  - → Freeware in Eclipse Umgebung
- Beide Werkzeuge
  - → messen die mittlere Testabdeckung und
  - → identifizieren die lokalen Defizite (Klassen- und Methodenebene)
  - → Unterstützen ein Defizitmonitoring

ISIS: die Wirkung

- Schaffung von Qualitätsbewusstsein durch intensive Diskussion über Softwarequalität, Metriken, Prozesse
- Zeitnahe Reaktion auf Qualitätsprobleme
- Lernorientierte Fehlerkultur
- Transparente Produktion
  - → Schafft Vertrauen bei Auftraggeber und Management
  - → Führt zu einer Verstetigung der Produktion
- In allen Projekten hat sich nach Einführung von ISIS die Codequalität deutlich verbessert
- ISIS macht Spaß, erzeugt hohes Selbstvertrauen, Motivation, Kreativität

ISIS: Maßnahmen zur Prozessoptimierung

- Strukturierung der Produktionskette
- Verbesserung der Anforderungsanalyse
- Verbesserung der Planung
- Verbesserung der Schätzmethode
- Konsequente pragmatische Anwendung von XP-Techniken
- Manuelle Entwicklertests als Ergänzung zu automatisierten Tests
- Konsequente Automatisierung

ISIS: Voraussetzungen

- Die Prozessoptimierung setzt in Teilen des Teams Erfahrungen zu
  - → sauberer objektorientierter Programmierung
  - → XP-Techniken
  - → Automatisierung

  voraus
- Es muss ein Mechanismus existieren, der die Zirkulation und Verbreitung von Wissen im Team gewährleistet (z. B. Pairprogramming)
- Für die Beherrschung der Werkzeuge Sotograph und Clover/EclEmma ist Spezialwissen erforderlich
- Voraussetzung ist die Bereitschaft zu einer lernorientierten Fehlerkultur