



andrena

OBJECTS

ISIS

Das Navigationssystem
für angemessene Qualität und hohe Effizienz

Inhalt

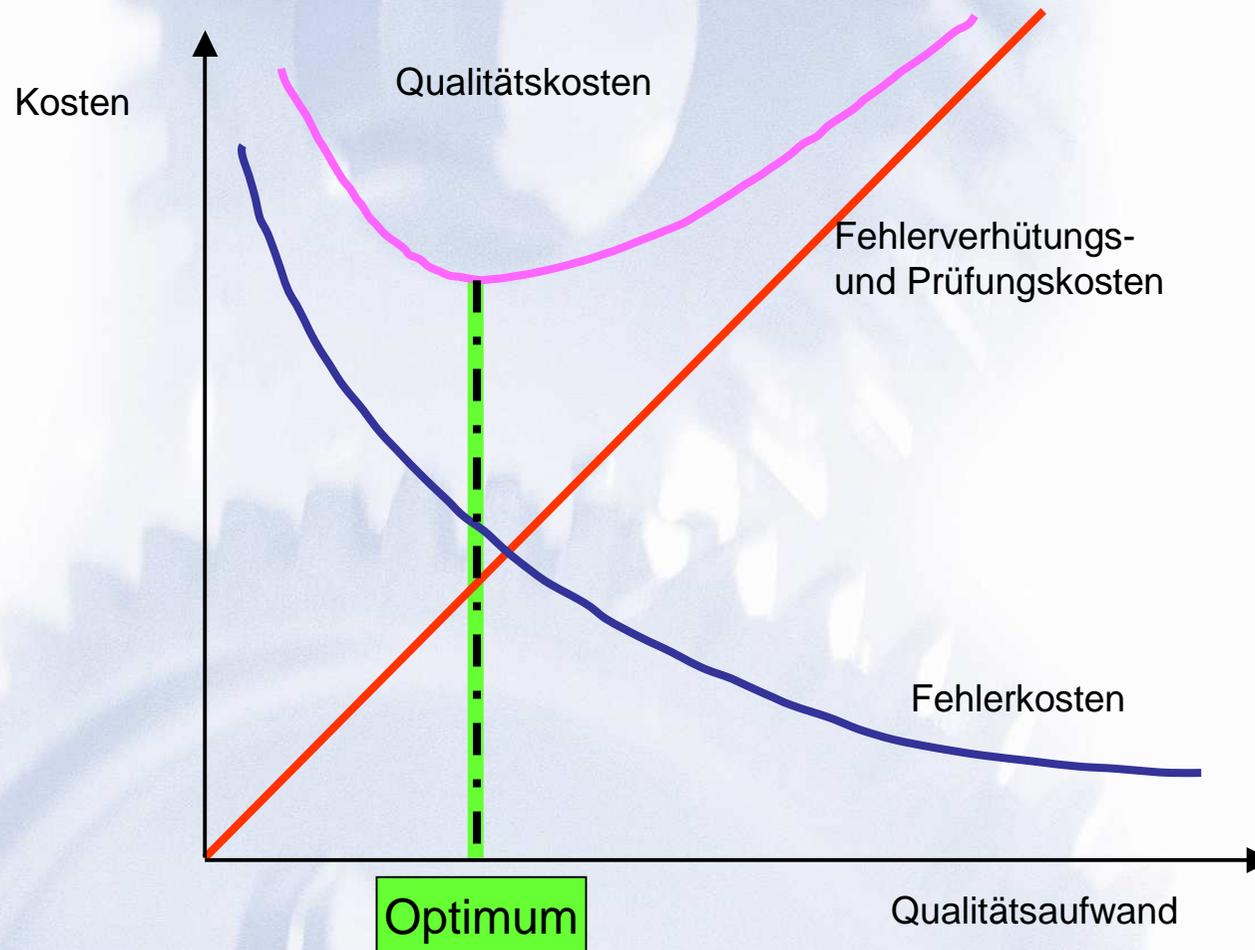
- Softwarequalität und Prozessqualität
- ISIS: das Ziel
- Messen der Prozessqualität
- Der Werkzeugzoo
- Die Wirkung
- Maßnahmen zur Prozessoptimierung
- Voraussetzungen für den sinnvollen Einsatz
- Spin-off: Messen der Softwarequalität
- Schlussbetrachtung

Softwarequalität: zwei Sichten

- Die Qualität eines Softwaresystems besteht aus der
 - *Inneren Qualität*: Verständlichkeit, Wartungs-, Erweiterungs- und Änderungsfreundlichkeit (Sicht des Entwicklers)
 - *Äußeren Qualität*: der Funktionalität, der Korrektheit, der Bedienbarkeit, der Performanz (Sicht des Benutzers)
- 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

Softwarequalität: Aufwandsoptimum



Kosten der Softwareproduktion minimieren
durch Steuerung der Prozessqualität

Messung der Prozessqualität

- Mehrere Methoden auf dem Markt
 - CMMI
 - SPICE
 - usw.
- Alle zu schwergewichtig, nicht praktikabel
- EN ISO 9001
 - Leichtgewichtig
 - Prozessorientiert (Messung der Kundenzufriedenheit als Steuergröße)
 - Für alle möglichen Produktionsbereiche geeignet
 - Zu dünn für Softwareentwicklung

ISIS: Messung der Prozessqualität

- Definition weniger **Indikatormetriken**, die
 - in Summe das Ganze abdecken
 - möglichst aussagekräftig und
 - leicht zu erheben sind
- Bewertung der Messdaten durch Projektion auf eine Skala von 0 bis 100 (Qualitätsniveaus)
- Verdichtung der gewichteten Qualitätsniveaus zu einem Leitindikator (Prozessqualitätsindex)

Die Auswahl der Indikatoren, die Bewertungsalgorithmen und die Gewichtung der Indikatoren basieren auf 200 Jahren Entwicklungserfahrung

ISIS: 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 %

ISIS: wo liegt das Optimum?

- PQI = 65 bis 85 und
- etwa 1,5 Bugs pro Doppelzentner Entwickler und Monat

ISIS: Der Werkzeugzoo

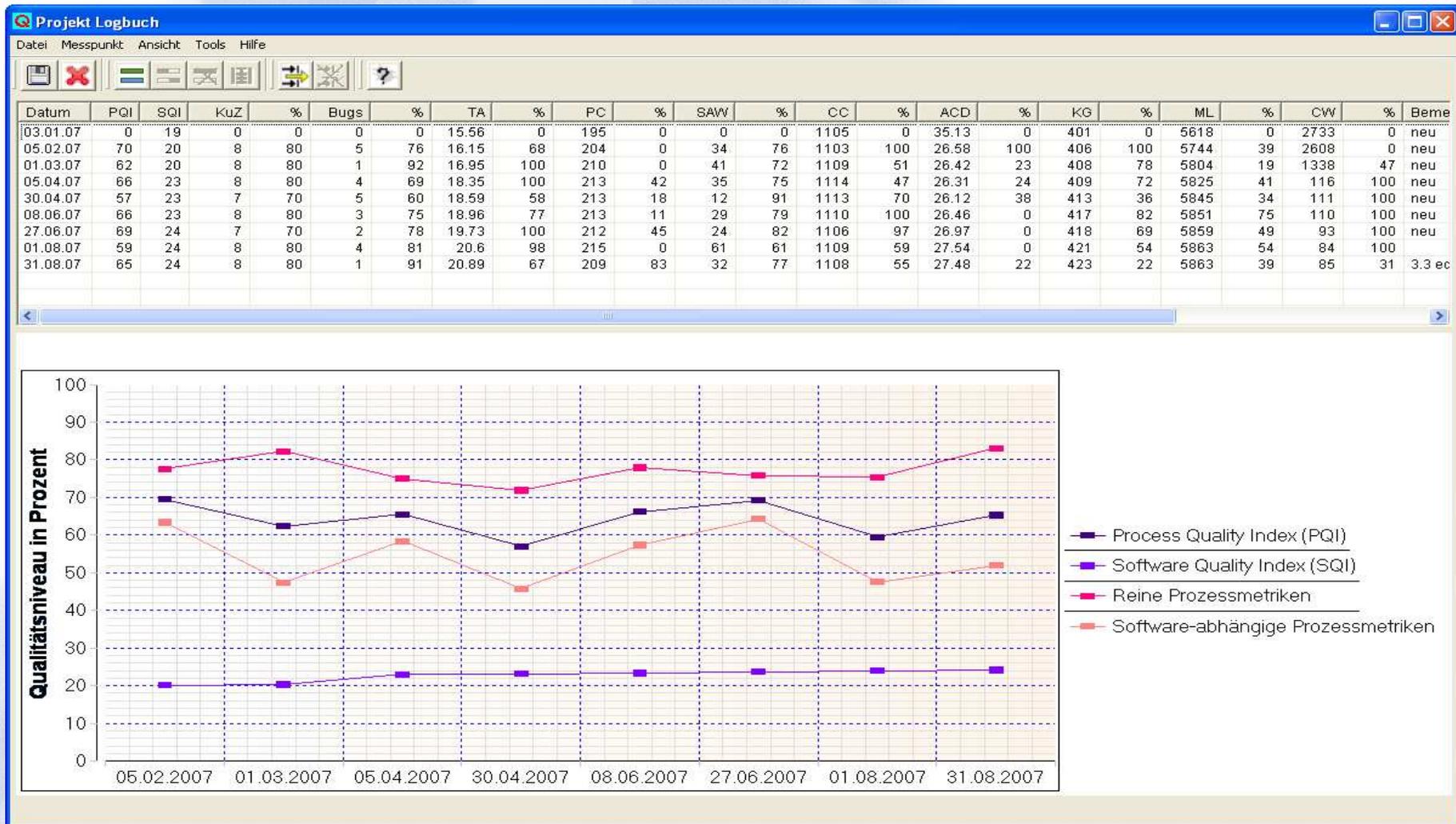
1. ProjektLogbuch (*andrena Eigenentwicklung*)
2. Automatisierte statische Analyse
 - *STAN* oder
 - *Sotograph* (*Lizenz*)
3. BugCollector (*andrena Eigenentwicklung*)
4. Testabdeckung (*EclEmma, Freeware*)

ProjektLogbuch: zentrales ISIS-Werkzeug

- Integriert die Basiswerkzeuge und Metriken
- Verdichtet die Indikatoren zu Indizes
- Historisiert die Qualitätsdaten
- Visualisiert

Das ProjektLogbuch ermöglicht **transparente Produktion**

Projektlogbuch: Beispiel aus der Produktion



Statische Code-Analyse: Alternativen

- STAN
 - Leichtgewichtig
 - Vollständig in das ProjektLogbuch integriert (automatisierte Analyse)
 - Auch als Standalone zu erwerben
- Sotograph
 - Das mächtigste und stabilste Werkzeug auf dem Markt
 - Teilweise in das ProjektLogbuch integriert (automatisierte Analyse)
 - 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

Bug Collector

Datei Bug Kategorie

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	checkZaehlprozessCache	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	Konstruktor	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	DBCReditkartenSperrung	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	DBCDetailPanelCreateMultiZaehlAufforderungFdDet...		Nein	Nein	GUI Behaviour
21	02.07.07	DBC	DBCBestandsbuchung	handleException	Nein	Nein	Exception Handling
24	16.07.07	DBC	DBCArtikelinformationMaske	addElement	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

EclEmma

- Freeware in Eclipse Umgebung
- misst die mittlere Testabdeckung und
- identifiziert die lokalen Defizite auf Klassen- und Methodenebene

ISIS: die Wirkung

- Schaffung von Qualitätsbewusstsein durch intensive Diskussion über Softwarequalität, Metriken, Prozesse
- Zeitnahe Reaktion auf Qualitätsprobleme
- 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

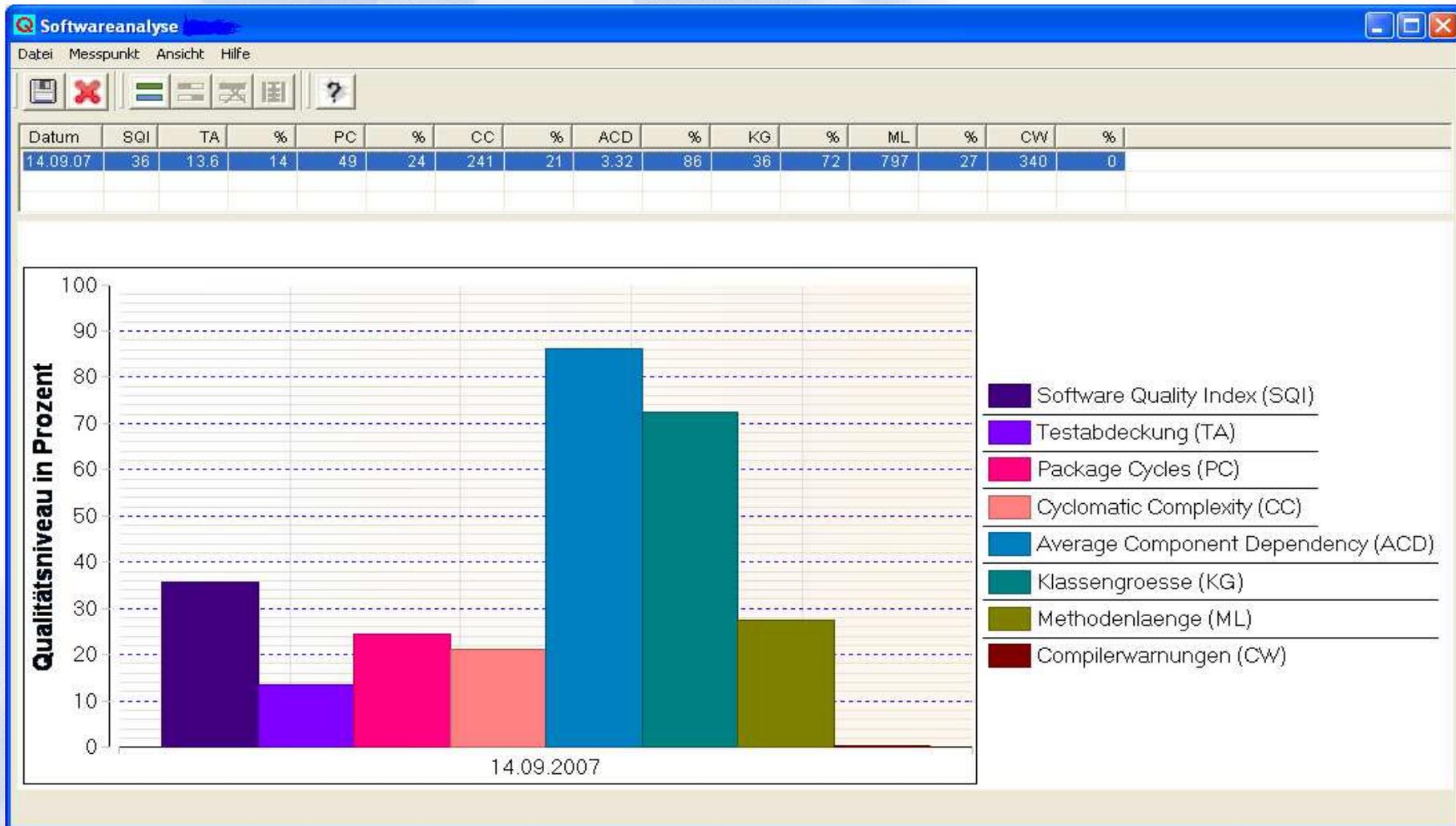
ISIS: Voraussetzungen

- Erfahrungen in Teilen des Teams Erfahrungen zu
 - sauberer objektorientierter Programmierung
 - XP-Techniken
 - Automatisierung
- Mechanismus, der die Zirkulation und Verbreitung von Wissen im Team gewährleistet (z. B. Pairprogramming)
- Inkrementelle Entwicklung, Iterative Planung und Steuerung, Retrospektiven (bei uns mit Scrum)
- Spezialwissen für die Beherrschung der Werkzeuge
- Bereitschaft zu einer **lernorientierten Fehlerkultur**

ISIS: Messen der Softwarequalität

- Softwarequalität kann sehr effizient gemessen werden
- Aufwand liegt bei 1 Personentag
- Einsatzmöglichkeiten:
 - Software Reviews
 - Monatliches Qualitätsmonitoring

ISIS: Beispiel Softwarequalität



Das **andrena** Qualitätsmanagementsystem

EN ISO 9001

+

ISIS

=

Agiles Qualitätsmanagementsystem