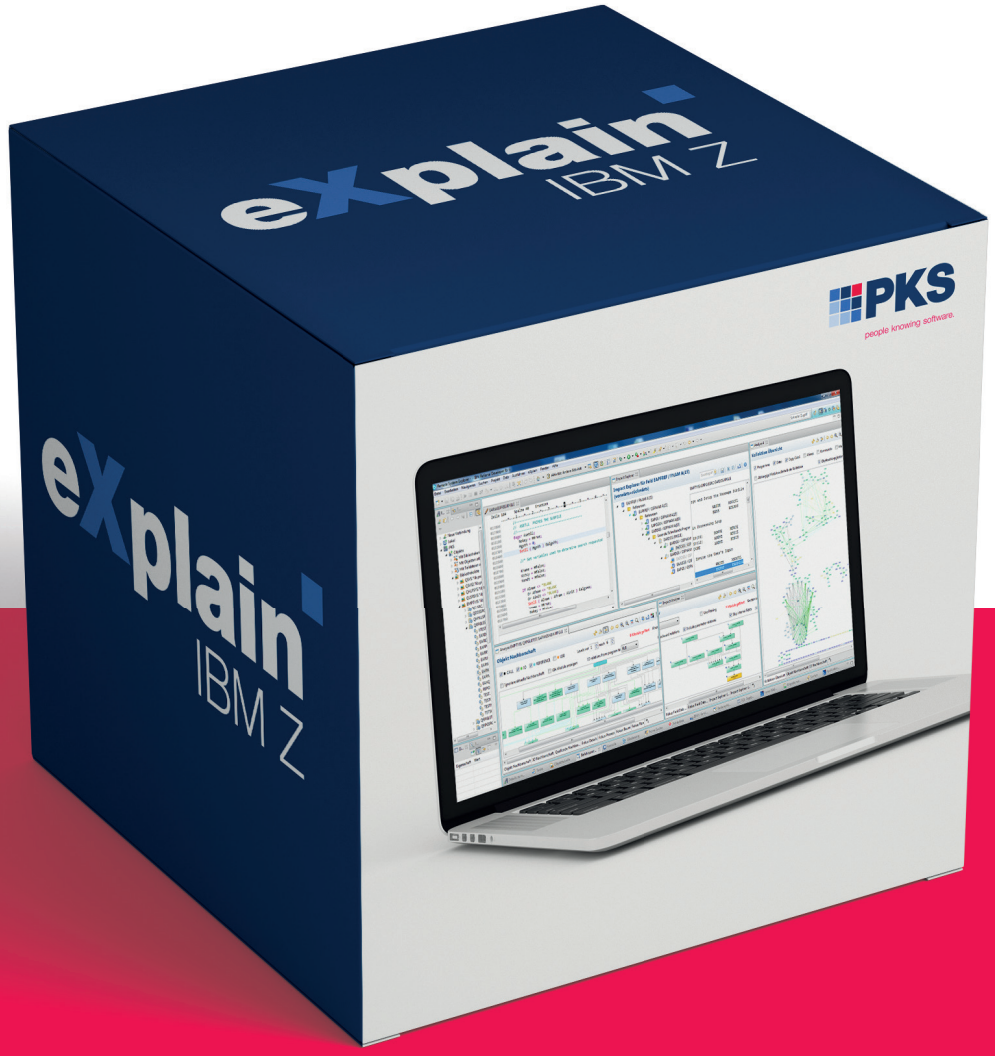
PKS
people knowing software.



Problemstellen in Legacy Code gezielt aufspüren:
Mit dem eXplain Quality Feature

1. Qualitätsanalyse von Softwareanwendungen

Alle Entwicklerteams mit IBM i und IBM Z Umgebungen leiden unter Legacy Code. Unter Legacy Code versteht man dabei klassischerweise etwas, das vor langer Zeit, oft von einer anderen Person, entwickelt wurde und als schwer oder schlecht wartbar wahrgenommen wird. Gibt man „Legacy“ in die Suchmaschine von Google ein, erhält man Ergebnisse wie Erbe, Vermächtnis und Hinterlassenschaft.

Mit dieser Definition ist die Verschlechterung der Code-Qualität zunächst einmal unabhängig von der Programmiersprache zu verstehen: mit der Dauer eines Projektes werden Änderungen am Code immer komplexer, die Chance, dass Vater oder Mutter eines Programms nicht mehr verfügbar ist, immer höher und das Risiko von Seiteneffekten steigt. Bei Entwicklern macht sich Unsicherheit, Frustration und Kontrollverlust breit. Das sind keine guten Voraussetzungen für motiviertes und produktives Arbeiten. Zusätzlich fehlen häufig durchgängige und/oder automatisierte Tests, die bei Änderungen deren Richtigkeit schnell und einfach erkennen lassen. Doch in den meisten Projekten ist es nicht möglich, den gesamten Legacy Code zu sanieren (z.B. den Code einem Refactoring zu unterziehen und ihn mit automatisierten Tests abzusichern). Die Aufwände wären zu groß und nur wenige Unternehmen sind in der Lage, das dafür notwendige Budget freizugeben. Es ist daher ein zielorientiertes Vorgehen sowie ein systematischer Blick auf die Legacy-Systeme notwendig, der die Entwicklungsteams in die Lage versetzt, strategisch wichtige Software-Komponenten und Problempunkte im Code automatisch zu erkennen und proaktiv sanieren zu können. Code oder Module gelten i.d.R. dann als strategisch wichtig, wenn sie für den Erfolg eines Unternehmens einen wesentlichen Beitrag leistet und eine schlechte Qualität hohe geschäftliche Nachteile oder operative Risiken mit sich brächte.

Problempunkte im Code mit statischer Codeanalyse aufspüren

Problempunkte lassen sich schon mit einfacher statischer Codeanalyse aufspüren. Statische Codeanalyse ist „State of the Art“ in der Softwareentwicklung und sollte daher auch von Cobol, RPG, PL/1 und Natural Entwicklern genutzt werden. Als Datenquelle dient dabei ausschließlich Code. Demzufolge kann statische Codeanalyse auch „nur“ Erkenntnisse liefern, die aus dem Code gewonnen werden können: Länge, Verschachtelungstiefen, Verletzung von Clean Code Regeln, Fehlermuster, … Diese Informationen reichen jedoch oft schon aus, um die wichtigsten Problempunkte im Code zu identifizieren und gezielte Maßnahmen zur Qualitätsverbesserung abzuleiten. Jedoch ist es immer gut, das Ergebnis einer statischen Code-Analyse mit dem Umfeld- und Fachwissen eines Experten zu bewerten und „zurechtzurücken“: so sollte man z.B. hinterfragen und analysieren, ob ein Modul z.B. gar nicht mehr geändert werden muss und im Test und Betrieb dieses Moduls vielleicht noch nie Fehler entdeckt wurden. In diesen Fällen ist die Refactoring-Strategie entsprechend anzupassen.

Fachliche Abgrenzungen erkennen – API-Fähigkeit herstellen – Ablösungen kontrollierbar machen

Viele Legacy-Anwendungen sollen in den kommenden Jahren schrittweise durch Neusysteme ersetzt werden. Für diese Projekte ist es essenziell notwendig, die relevanten Schnittstellen zwischen fachlich abgrenzbaren Einheiten im Code in deren Komplexität greifbar zu machen. Auch dies ist mit dem entsprechenden fachlichen Gliedern (clustern) von Legacy Systemen möglich und erleichtern somit Planung, Vorbereitung und Umsetzung von Software-Transformationen.

Den Generationswechsel kontrollierbar machen

Bedingt durch die fortschreitende Verrentung im IBM i und Z Umfeld denken viele Unternehmen darüber nach, externe Ressourcen einzubinden. Doch gerade beim Transfer der Wartung und Weiterentwicklung an externe Partner ist es seitens des Auftraggebers essentiell, die Code Qualität fortlaufend kontrollieren zu können, so dass nicht die Gefahr entsteht, dass der übergebene Code nach geraumer Zeit in einem noch schlechteren Zustand ist und dadurch die Pflege, Weiterentwicklung oder Ablösung unnötig verteuert oder wesentlich riskanter wird.

Mit dem eXplain Quality Feature steht ein Werkzeug zur automatisierten Komplexitätsmessung sowie fortlaufenden Qualitätskontrolle zur Verfügung. Dabei kann ein speziell optimierter und im Code messbarer Komplexitätsbegriff gemäß den Richtlinien des Kunden und Erfahrungswerten der PKS definiert und parametrisiert werden. Basierend auf Sprachgrammatiken und definierten Regeln können unerwünschte Konstellationen im Sourcecode erkannt und vermessen werden. Zudem können diese Informationen dem eXplain-Nutzer als Warnung gemeldet werden. Problematische Situationen wie fehlende Überprüfung von Returncodes nach bestimmten Aufrufen (ESQL, CICS, MQ, …) oder performancekritische Algorithmen können dadurch automatisiert erkannt werden.

Fazit

Problempunkte im Code lassen sich mit dem eXplain Quality Feature einfach, gezielt und dauerhaft aufspüren. Die begrenzten Budgets zur Sanierung und Modernisierung des Legacy Codes werden dadurch effizienter und effektiver eingesetzt. Außerdem unterstützt das eXplain Quality Feature die Übergabe und darauffolgende Kontrolle von Outsourcern.

2. Regeln und Metriken: Die beiden unterschiedlichen Arten, die Qualität einer Anwendung zu untersuchen bzw. zu bewerten

In eXplain gibt es zwei grundlegend unterschiedliche Ansätze, sich dem Thema Software-Qualität zu nähern:

1. Für jede Sprache gibt es einen Satz von Regeln, die von den Parsern überprüft werden und bei Regelverletzungen erfolgen entsprechende Einträge ins Repository.
   Typischerweise prüfen diese Regeln Eigenschaften ab wie z.B.

   - Definierte max. Schachtelungstiefe wird überschritten
   - Keine Fehlerbehandlung nach einem SQL-Call
   - IF ohne ENDIF

   Ein anderer Anwendungsfall ist die Überprüfung von Programmierrichtlinien, Namenskonventionen u.ä.

2. Außerdem ermitteln die Parser viele in der Informatik bekannte Metriken („Zahlenwerte“), die zu einem großen Teil auf Function Points, Halstead-Metriken u.ä. basieren.
Diese Metriken werden im Repository abgelegt und stehen für Auswertungen zur Verfügung.

   Darüber hinaus werden zusätzliche Metrikwerte basierend auf Clusterdefinitionen (die eine fachliche Sicht auf die Anwendung einbringen) errechnet und im Repository entsprechend gespeichert.
Eine besondere Rolle spielen hier die sog. Cluster-Interface-Metriken, die eine Aussage treffen über die Schnittstellen zwischen Clustern (Anwendungsbereichen), z.B. über die vorhandene Servicekapselung.

Die erzeugten QA-Informationen können in vielfältiger Weise innerhalb von eXplain genutzt werden, u.a. bei der Erzeugung von benutzerdefinierten Qualitäts-Reports oder bei komplexen Schnittstellenanalysen.

Das QA-Dashboard bietet jederzeit einen Überblick über die wichtigsten Parameter und v.a. auch über die Veränderungen einzelner Qualitätsmerkmale im Laufe der Zeit (Quality History).

3. Verfügbare Qualitäts-Regeln in eXplain

Die folgenden Regeln sind aktuell in eXplain vorhanden und können durch Parameter an die jeweilige Kundensituation angepasst werden.

Jede Regel kann einzeln aktiviert/deaktiviert und einem bestimmten Schweregrad zugeordnet werden.

Im Rahmen von Kundenprojekten kann die Liste der verfügbaren Regeln nach gemeinsamen Spezifikationen erweitert werden.

QA_CBL_AAN	Assignment Alpha Numeric
QA_CBL_ADWL	Accept Date Within Loop
QA_CBL_ANA	Assignment Numeric Alpha
QA_CBL_APL	Assignment Precision Loss
QA_CBL_CFNSTOF	Copybook Field Not Similar To Other Field
QA_CBL_CICSCRCNC	CICS Call Return Code Not Checked
QA_CBL_CICSCRCNCBF	CICS Call Return Code Not Checked By Function
QA_CBL_CICSCRCNR	CICS Call Return Code Not Requested
QA_CBL_CNLE	Condition Nesting Limit Exceeding
QA_CBL_CNSIS	Condition Not Set in Section
QA_CBL_DEANO	DataElement Alpha Numeric Overlapping
QA_CBL_DEINU	DataElement Initialized but Not Used
QA_CBL_DENU	DataElement Not Used
QA_CBL_DEUNI	DataElement Used but Not Initialized
QA_CBL_DPCU	Dynamic Program Call Used

QA_CBL_DSNA	Display Statement Not Allowed
QA_CBL_DSNA	Display Statement Not Allowed
QA_CBL_ENCI	End Not Correct Indented
QA_CBL_EWEE	Evaluate Without End-Evaluate
QA_CBL_EWS	Exit Without Section
QA_CBL_EWWO	Evaluate Without When Other phrase
QA_CBL_FMNA	Field Modification Not Allowed
QA_CBL_FNIBP	Field Not Initialized Before Perform
QA_CBL_FNU	File Not Used
QA_CBL_GSFUWQ	Group-Sub-Field Used Without Qualification
QA_CBL_GTIS	Go To Inter-Sectional
QA_CBL_GTOOL	Go To Out Of Loop
QA_CBL_GTU	Go To Used
QA_CBL_IBS	Initialize Big Structure
QA_CBL_IBS	Initialize Big Structure
QA_CBL_IBT	Initialize Big Table
QA_CBL_IBT	Initialize Big Table
QA_CBL_IBT	Initialize Big Table
QA_CBL_ICOSCO	If Cascade On Same Condition Operand (use EVALUATE)
QA_CBL_IMSCRCNC	IMS Call Return Code Not Checked
QA_CBL_IOWEH	IO Without Error Handling
QA_CBL_ISNCI	If Structure Not Correctly Indented
QA_CBL_ISNCI	If Structure Not Correctly Indented
QA_CBL_ISNCI	If Structure Not Correctly Indented
QA_CBL_IWEI	If Without End-If
QA_CBL_IWEI2	If Without End-If 2
QA_CBL_LNLE	Loop Nesting Limit Exceeding
QA_CBL_LOMSADDM	Length Of Move Source And Destination Dont Match
QA_CBL_MCFTM	Move Corr Field Type Mismatch
QA_CBL_MQCRCNC	MQ Call Return Code Not Checked
QA_CBL_NSU	NEXT SENTENCE used
QA_CBL_OCMCC	Open Cursor Missing Close Cursor
QA_CBL_OCWL	Open Cursor Within Loop
QA_CBL_PCANC	Program Call Argument Naming Convention
QA_CBL_PCMTO	Procedure Called More Than Once from program
QA_CBL_PCPNI	Program Call Parameter Not Initialized
QA_CBL_PCPNI	Program Call Parameter Not Initialized
QA_CBL_PCPNI	Program Call Parameter Not Initialized
QA_CBL_PCPNI	Program Call Parameter Not Initialized
QA_CBL_PCPNSTSV	Program Call Parameter Not Set To Specific Value
QA_CBL_PCRCNC	Program Call Return Code Not Checked
QA_CBL_PCSLE	Program Code Statements Limit Exceeding
QA_CBL_PESL	Period Ends Statement Levels
QA_CBL_PNC	Program Not Called
QA_CBL_PNP	Paragraph Not Performed

QA_CBL_PNP	Paragraph Not Performed
QA_CBL_PPFSWNC	Procedure Performend From Section With Naming Convention
QA_CBL_PSLE	Paragraph Statements Limit Exceeding
QA_CBL_SA	Self-Assignment
QA_CBL_SNP	Section Not Performed
QA_CBL_SNP	Section Not Performed
QA_CBL_SOPNE	Sections Or Paragrhs Not Exclusive
QA_CBL_SOPNE	Sections Or Paragrhs Not Exclusive
QA_CBL_SPCU	Static Program Call Used
QA_CBL_SQLCNC	SQL Code Not Checked
QA_CBL_SQLDCWOB	SQL Declare Cursor Without Order By
QA_CBL_SQLFU	SQL Function Used
QA_CBL_SQLMRFRNC	SQL Multiple Rows Fetch Result Not Check
QA_CBL_SQLNIBSOF	SQL No Initialization Before Select Or Fetch
QA_CBL_SQLNIBSOF	SQL No Initialization Before Select Or Fetch
QA_CBL_SQLNIBSOF	SQL No Initialization Before Select Or Fetch
QA_CBL_SQLNMBIOU	No Modification Before SQL Insert Or Update
QA_CBL_SQLNMBIOU	No Modification Before SQL Insert Or Update
QA_CBL_SQLSCIWFF	SQL Select Constant Into Without Fetch First Row
QA_CBL_SQLSRFWL	SQL Single Row Fetch Within Loop
QA_CBL_SRU	Stop Run Used
QA_CBL_SSLE	Section Statements Limit Exceeding
QA_CBL_SWCB	Section Without Comment Before
QA_CBL_SWE	Section Without Exit
QA_CBL_SWE	Section Without Exit
QA_CBL_SWP	Section Without Paragraph
QA_CBL_TIONCWL	Table Index Overflow Not Checked Within Loop
QA_DC	Dead Code detected
QA_MM	Missing Module
QA_NAT_BT	Usage of BACKOUT TRANSACTION in a database block
QA_NAT_COND	Usage of critical logical condition
QA_NAT_DBF_UPDATE	Usage of UPDATE instruction in a FIND WITH-Equal block
QA_NAT_DBF_WHERE	Usage of WHERE clause in FIND
QA_NAT_DBF_WITH	Usage of WITH clause in FIND
QA_NAT_DBH_FROM	Usage of FROM clause in HISTOGRAM
QA_NAT_DBH_RANGE	Usage of RANGE clause in HISTOGRAM
QA_NAT_DBH_WHERE	Usage of WHERE clause in HISTOGRAM
QA_NAT_DBR_FROM	Usage of FROM clause in READ
QA_NAT_DBR_RANGE	Usage of RANGE clause in READ
QA_NAT_DBR_WHERE	Usage of WHERE clause in READ
QA_NAT_EXTCALL	Usage of external modules
QA_NAT_HEX	Usage of HEX literal
QA_NAT_LINEREF	Usage of line references
QA_NAT_NO_LABEL	Usage of update and delete statements with no label
QA_NAT_ON_ERROR1	Usage of ON ERROR with program termination and without BACKOUT TRANSACTION

QA_NAT_ON_ERROR2	Usage of ON ERROR without program termination and BACK-OUT TRANSACTION
QA_NAT_SORT	Usage of SORT statement
QA_PLI_GOTOLIOPT	Enterprise PLI message IBM1092I
QA_PLI_GOTOUSED	Enterprise PLI message IBM1458I
QA_PLI_LINECMTOS	Enterprise PLI message IBM2464I
QA_PLI_LINECTMS	Enterprise PLI message IBM2475I
QA_PLI_MAXNESTDOSE	Enterprise PLI message IBM2426I
QA_PLI_MAXNESTIFSE	Enterprise PLI message IBM2427I
QA_PLI_MULCLGR	Enterprise PLI message IBM1120I
QA_PLI_ONERRNOSYS	Enterprise PLI message IBM2621I
QA_PLI_PROCNR	Enterprise PLI message IBM2800I
QA_PLI_SELCNOOTHCL	Enterprise PLI message IBM1059I
QA_PLI_SLENLTV	Enterprise PLI message IBM1211I
QA_PLI_STRALTT	Enterprise PLI message IBM1186I
QA_PLI_VARNI	Enterprise PLI message IBM1085I
QA_PLI_VARUNR	Enterprise PLI message IBM2418I
QA_PLI_VDECLWDA	Enterprise PLI message IBM1482I
QA_RPGLE_CFU	Cycle File Used
QA_RPGLE_EPL	EVAL Precision Loss
QA_RPGLE_GTIS	Go To Inter-SubRoutine
QA_RPGLE_GTOOL	Go To Out Of Loop
QA_RPGLE_GTU	Go To Used
QA_RPGLE_IOEIUNC	IO Error Indicator Used but Not Checked
QA_RPGLE_IOEIUNC	IO Error Indicator Used but Not Checked
QA_RPGLE_IOSNC	IO Status Not Checked
QA_RPGLE_IOSNC	IO Status Not Checked
QA_RPGLE_IOSNC	IO Status Not Checked
QA_RPGLE_IOWEH	IO Without Error Handling
QA_RPGLE_PCPDTM	Program Call Parameter Data Type Mismatch
QA_RPGLE_PDFU	Program Described File Used
QA_RPGLE_PDFUED	Program Described File Used but existing External Description
QA_RPGLE_SQLCNC	SQL Code Not Checked
QA_RPGLE_SQLCNC	SQL Code Not Checked
QA_RPGLE_SQLCNC	SQL Code Not Checked

4. Verfügbare Qualitäts-Metriken in eXplain

Generell können alle in eXplain vorhandenen Metriken mit Schwellwerten versehen werden, die festlegen, ab wann eine einzelne Metrik als „kritisch“ oder „gut“ eingeschätzt wird.

Diese Schwellwerte dienen u.a. zur Festlegung eines sog. „eXplain Ratings“ von 0-5, vergleichbar mit dem „Sterne-Rating“ von Ratingagenturen.

Durch das eXplain-Rating ergibt sich ein deutlich einfacherer Blick auf die Qualität einer Anwendung als beim direkten Arbeiten mit den sehr unterschiedlich dimensionierten reinen Zahlenwerten.

4.1. Verfügbare Qualitäts-Metriken in eXplain

Diese Metriken beziehen sich immer auf ein gesamtes Modul (Programm).

LINESTOTAL	Programmgröße (LOC)
PCT_LINESCOMMENT	% Kommentarzeilen
PCT_LINESDEAD	% Tote Zeilen
COMPLEXITY	Komplexität
MAINTAINABILITY	Wartbarkeit
NESTINGDEPTH	Schachtelungstiefe
QERRORS	Qualitäts-Fehler
EXPLAIN	eXplain-Rating
LINESCOMMENT	Kommentarzeilen
LINESDEAD	Tote Zeilen
GOTO_ERR	GOTO benutzt
IONC_ERR	IO ohne Fehlerbehandlung (ERROR)
IONC_WARN	IO ohne Fehlerbehandlung (WARNING)
PRECISIONLOSS	Verlust Dezimalstellen
PARAMMISMATCH	Programmaufufe Parameter Mismatch
WLOC_CYCLE	RPG Zyklus
WLOC_PGMDESCFILE	RPG Intern beschr. Datei

4.2. Clustermetriken

Diese Metriken beziehen sich jeweils auf einen einzelnen Cluster, also einen definierten fachlichen Anwendungsteilbereich.

LINESTOTAL	Programmgröße (LOC)
PCT_LINESCOMMENT	% Kommentarzeilen
PCT_LINESDEAD	% Tote Zeilen
COMPLEXITY	Komplexität
MAINTAINABILITY	Wartbarkeit
NESTINGDEPTH	Schachtelungstiefe
QERRORS	Qualitäts-Fehler
EXPLAIN	eXplain-Rating
LINESCOMMENT	Kommentarzeilen
LINESDEAD	Tote Zeilen
GOTO_ERR	GOTO benutzt (Severity ERROR)
IONC_ERR	IO ohne Fehlerbehandlung (ERROR)
IONC_WARN	IO ohne Fehlerbehandlung (WARNING)
PRECISIONLOSS	Verlust Dezimalstellen
PARAMMISMATCH	Programmaufufe Parameter Mismatch)
WLOC_CYCLE	RPG Zyklus
WLOC_PGMDESCFILE	RPG Intern beschr. Datei
AVG_LINESTOTAL	Durchschn. Programmgröße (LOC)
AVG_LINESCOMMENT	Durchschn. Kommentarzeilen
AVG_LINESDEAD	Durchschn. tote Zeilen
AVG_PCT_LINESCOMMENT	Durchschn. % Kommentarzeilen
AVG_PCT_LINESDEAD	Durchschn. % Tote Zeilen
AVG_COMPLEXITY	Durchschn. Komplexität
AVG_MAINTAINABILITY	Durchschn. Wartbarkeit
AVG_NESTINGDEPTH	Durchschn. Schachtelungstiefe
AVG_QERRORS	Durchschn. Qualitäts-Fehler
AVG_EXPLAIN	Durchschn. eXplain-Rating
AVG_GOTO_ERR	GOTO benutzt (Severity ERROR)
AVG_IONC_ERR	Durchschn. IO ohne Fehlerbehandlung (ERROR),
AVG_IONC_WARN	Durchschn. IO ohne Fehlerbehandlung (WARNING)
AVG_PRECISIONLOSS	Durchschn. Verlust Dezimalstellen
AVG_PARAMMISMATCH	Durchschn. Programmaufufe Parameter Mismatch
AVG_WLOC_CYCLE	Durchschn. RPG Zyklus
AVG_WLOC_PGMDESCFILE	Durchschn. RPG Intern beschr. Datei
SUM_IONC_ERRW	Sum. IO ohne Fehlerbehandlung (ERROR)
SUM_IONC_WARN	Sum. IO ohne Fehlerbehandlung (WARNING)
PROGRAMS	Anz. Programme

INTFCPROGRAMS	Anz. Interface-Programme
PCT_INTFCPROGRAMS	% Interface-Programme
SERVICEPROGRAMS	Anz. Service-Programme
PCT_SERVICEPROGRAMS	% Service-Programme
FILES	Anz. Dateien
INTFCFILES	Anz. Interface-Dateien
PCT_INTFCFILES	% Interface-Dateien
SERVICEFILES	Anz. Service-Dateien
PCT_SERVICEFILES	% Service-Dateien
IC_CALLS	IC-Aufrufe
IC_CALLS_NONSERVICE	Nicht-Service IC-Aufrufe
IC_IO_DB	IC IO DB
IC_IO_DB_NONSERVICE	IC IO DB non service
IC_IO_WORK	IC IO Workfile
IC_IO_WORK_NONSERVICE	IC IO Workfile non service

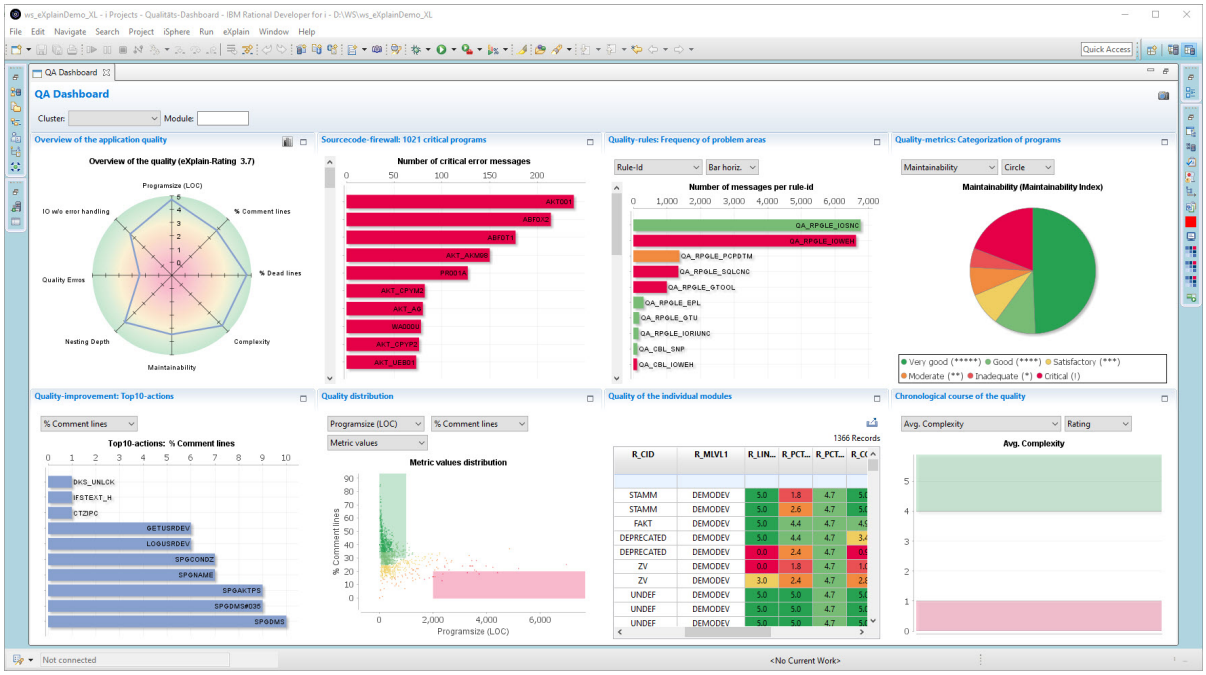
4.3. Clusterinterface-Metriken

Diese Metriken beziehen sich jeweils auf einen einzelnen Cluster, also einen definierten fachlichen Anwendungsteilbereich.

CALL_NROFCALLS	CALL: Anzahl der Programmaufrufe zwischen den Clustern
CALL_NROFPARENTMODULES	CALL: Anzahl der aufrufenden Programme im Parent-Cluster
CALL_NROFCHILDMODULES	CALL: Anzahl der aufgerufenen Programme im Child-Cluster
CALL_PERCENTAGEOFPARENTMODULES	CALL: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
CALL_PERCENTAGEOFCHILDMODULES	CALL: Prozentualer Anteil der aufgerufenen Programme im Child-Cluster
IO_NROFCALLS	IO: Anzahl der IO-Calls zwischen den Clustern
IO_NROFPARENTMODULES	IO: Anzahl der aufrufenden Programme im Parent-Cluster
IO_NROFCHILDMODULES	IO: Anzahl der verwendeten Dateien im Child-Cluster
IO_PERCENTAGEOFPARENTMODULES	IO: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
IO_PERCENTAGEOFCHILDMODULES	IO: Prozentualer Anteil der verwendeten Dateien im Child-Cluster

5. Das QA Dashboard in eXplain

Im QA Dashboard von eXplain können die für den Kunden bzw. den jeweiligen Entwickler relevanten Qualitätssichten kompakt zusammengestellt und visualisiert werden. Somit sieht man auf einen Blick, wie es um die jeweils im Fokus stehenden Komponenten, Metriken usw. steht. Auch für Team- und Projektleiter stellt das QA Dashboard einen guten Gesamtüberblick dar. Es kann jeweils der Scope an Code ausgewählt werden, auf den sich das Dashboard bezieht (z.B. ein einzelner Cluster oder eine definierte Menge von Programmen).



Die einzelnen Views von links oben nach rechts unten kurz erläutert:

Qualität der Anwendung im Überblick

Die wichtigsten Metriken werden in Form eines „Spinnennetzes“ dargestellt, so dass auf einen Blick ersichtlich ist, welche eher im grünen Bereich liegen oder als sehr kritisch eingestuft werden. Für die Einstufung werden die erwähnten Schwellwerte herangezogen, die kundenspezifisch festgelegt werden können.

Code-Firewall

Auflistung der Programme mit den meisten Verletzungen von Regeln mit Schweregrad 3 (kundenspezifisch konfigurierbar). Durch Anklicken eines Diagrammbalkens erhält man für diese Programm eine Übersicht aller betroffenen Sourcestellen bzw. aller verletzten QA-Regeln. Jede Verletzung kann direkt im Sourcecode angezeigt werden!
Diese Regeln mit hohem Schwergrad zeigen Sourcecodestellen auf, die u.U. zur Laufzeit zu schwer auffindbaren Defekten führen, wie z.B. Schleifenüberläufe oder nicht abgefangene Fehlersituationen.

Qualitäts-Regeln: Häufigkeit der Problemstellen

Übersicht über die definierten QA-Regeln und deren Häufigkeit, der definierte Schweregrad führt zur entsprechenden Einfärbung. Man erkennt dann, dass z.B. das hauptsächliche Problem der Anwendung eine zu hohe Schachtelungstiefe der Programmstrukturen ist.

Qualitäts-Metriken: Kategorisierung der Programme

Gibt einen Überblick darüber, welche Qualität die Programme hinsichtlich einer einzelnen Metrik haben. Die Kategorisierung erfolgt auf Basis der definierten Schwellwerte bzw. des dahinterliegende eXplain-Ratings. Es ergeben sich Aussagen der Art „70% aller Programme haben eine sehr gute Wartbarkeit, 10% aller Programme hingegen eine extrem schlechte Wartbarkeit“. Die jeweils zu einer gruppe gehörenden Programme können interaktiv als Grid angezeigt werden, u.s.w

Qualitäts-Verbesserungen: Top 10-Aktionen

Für jede Metrik können die 10 Programme angezeigt werden, welche die schlechtesten Metrikwerte erzeugen. Es bietet sich u.U. an, genau diese Programme bei anstehenden Verbesserungen ins Auge zu fassen und „die gröbsten Schnitzer“ zu beseitigen.

Qualitäts-Verteilung

Dies View dient dazu, zwei verschiedene Metriken gleichzeitig im Auge zu behalten, um sich bei Verbesserungsvorhaben auf das Wesentliche zu konzentrieren.
Während z.B. ein schlechter Kommentierungsgrad für sich allein betrachtet noch akzeptabel erscheinen mag, so ist dessen Auftreten in Kombination mit einer hohen Komplexität des Programmes sehr kritisch. Das Ziel ist also, Programme zu identifizieren, die gleichzeitig bei zwei Metriken sehr schlecht abschneiden.
Man erkennt im Diagramm, wie sich die Programme bzgl. der beiden ausgewählten Metriken verteilen und interaktiv können die gewünschten Teilmengen selektiert und bearbeitet werden.

Qualität der einzelnen Module

Darstellung der Metriken für jedes Programm in Gridform, beliebig sortierbar, u.s.w

Qualität im zeitlichen Verlauf

Ein sehr wichtiger Aspekt beim Thema „Qualität einer Softwareanwendung“ ist die Frage, wie diese sich im Laufe der Zeit verändert. Wird die Qualität bzgl. einer bestimmten Metrik immer schlechter? Haben durchgeführte Verbesserungsarbeiten tatsächlich den gewünschten Effekt?

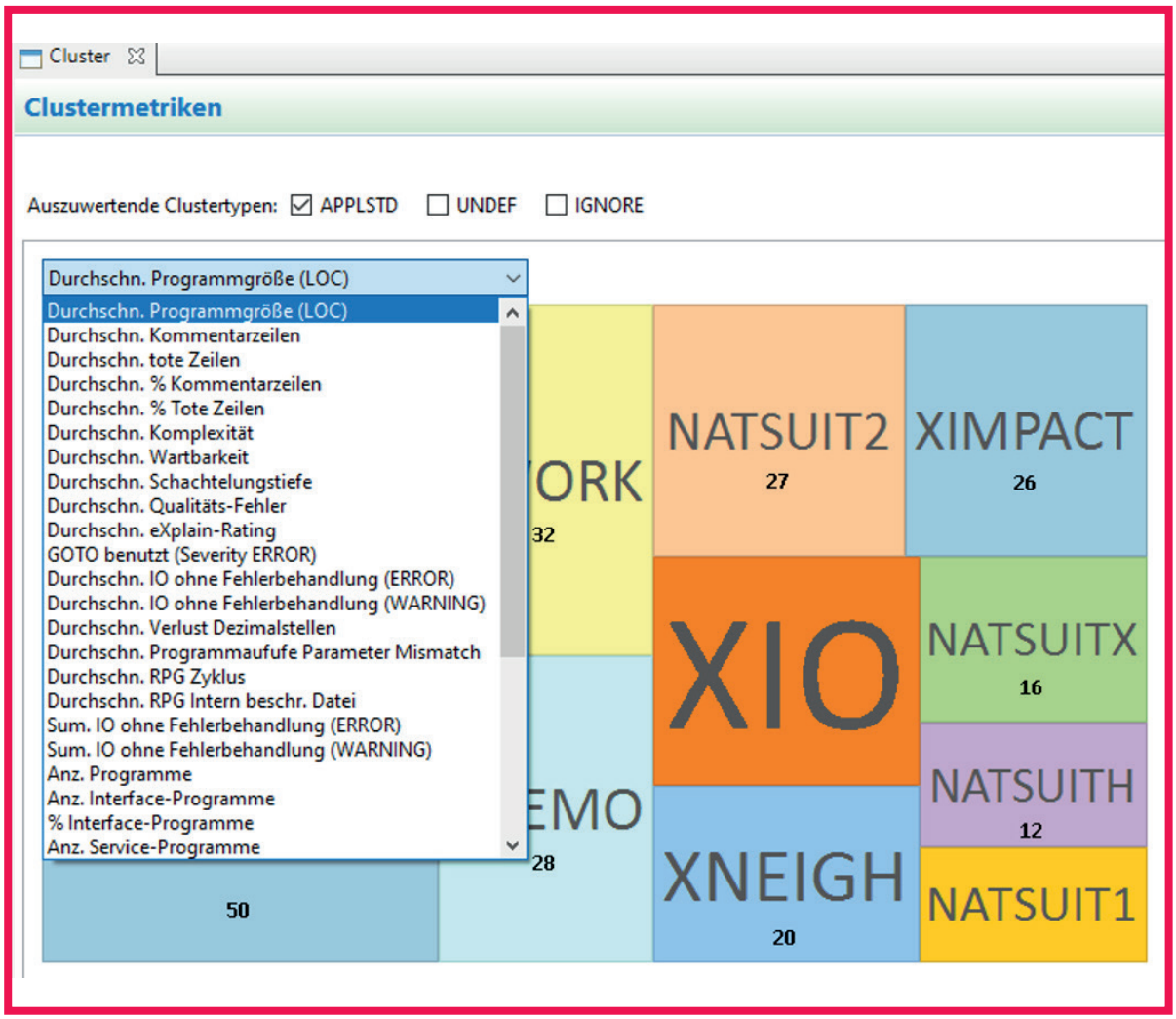
Durch regelmäßige Snapshots kann der aktuelle Zustand der Anwendung im Repository abgelegt werden, so sadd zeitliche Verläufe für jede einzelne Metrik in dieser View dargestellt werden können.

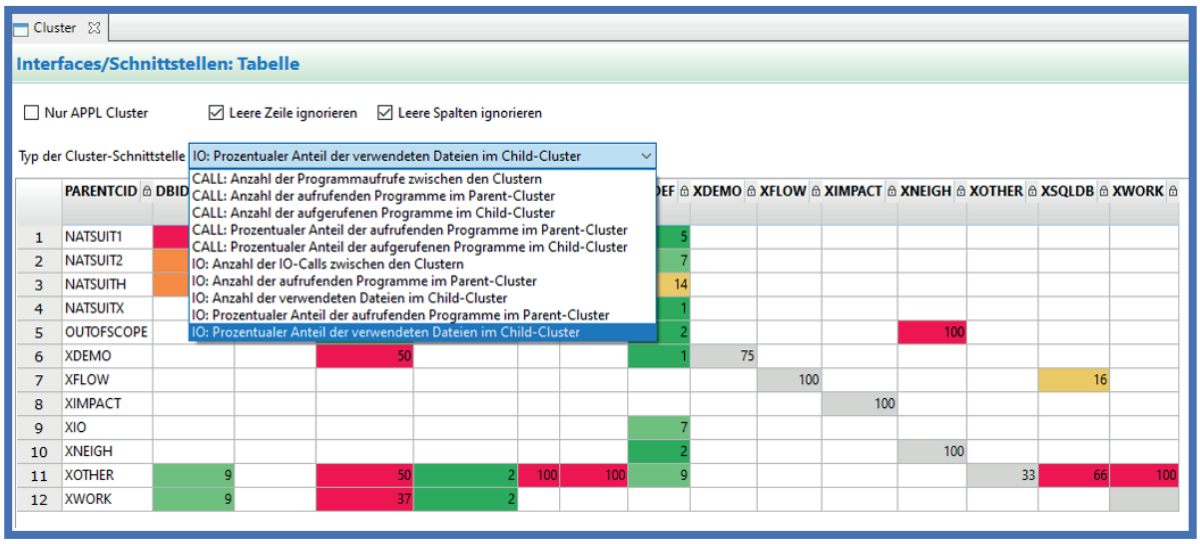
6. Die Kombination der fachlichen Sicht mit der Qualitätsanalyse

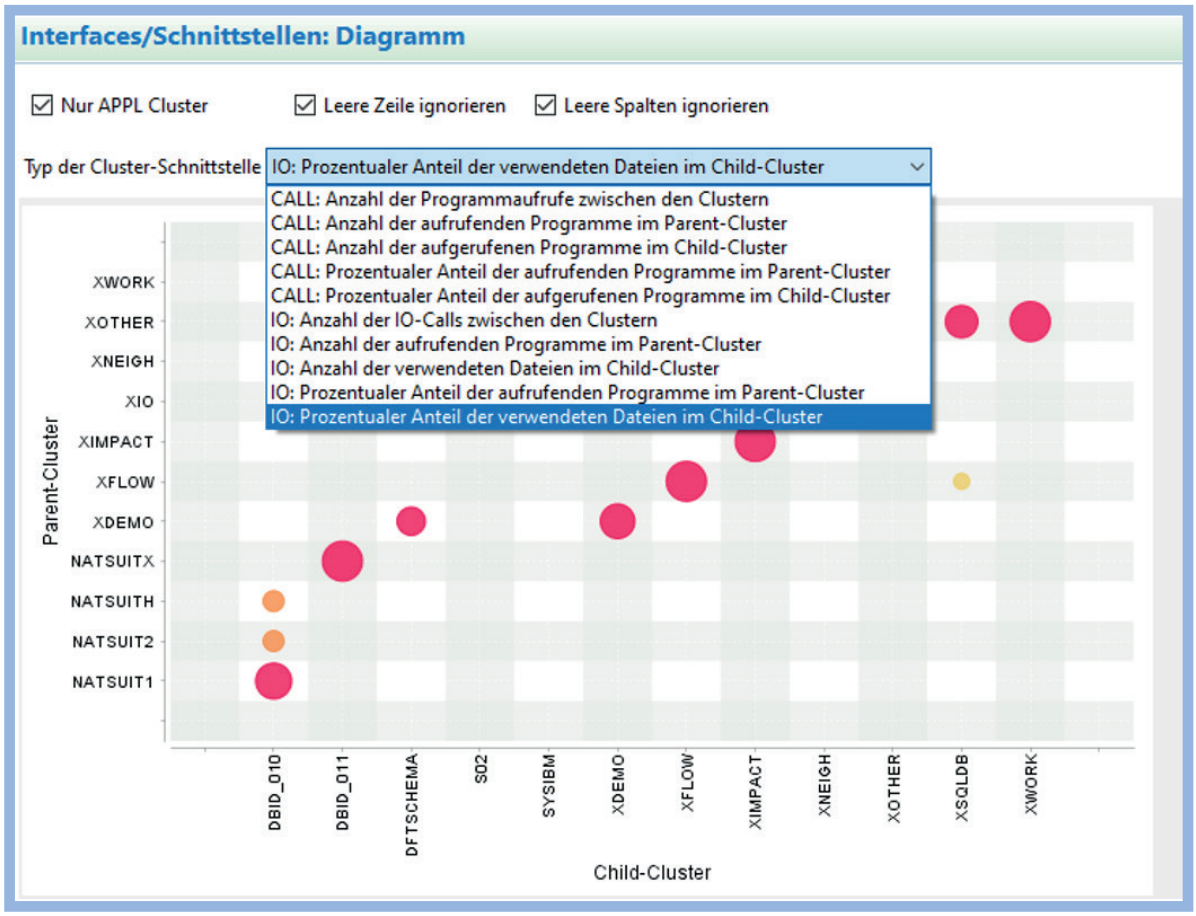
Die beschriebenen Qualitätsmerkmale der einzelnen Programme können mit den Informationen über fachliche Teilbereiche („Anwendungs-Cluster) kombiniert werden. Dadurch ergeben sich vielfältige neue Einsichten in die untersuchten Anwendungen.

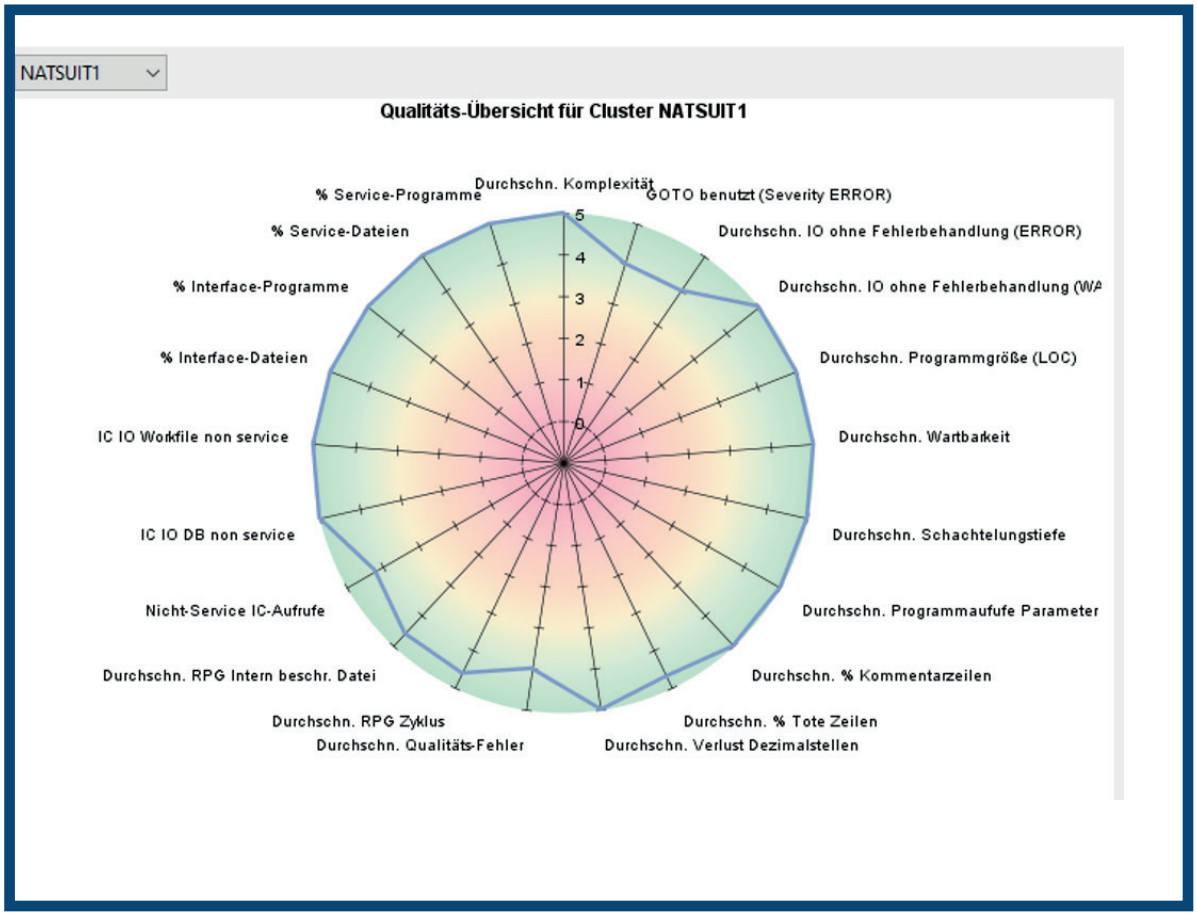
Eine besondere Rolle spielt die Analyse und Beurteilung der Schnittstellen zwischen fachlichen Teilbereichen und eXplain bietet hierfür eigens für diesen Zweck definierte Clusterinterface-Metriken an.

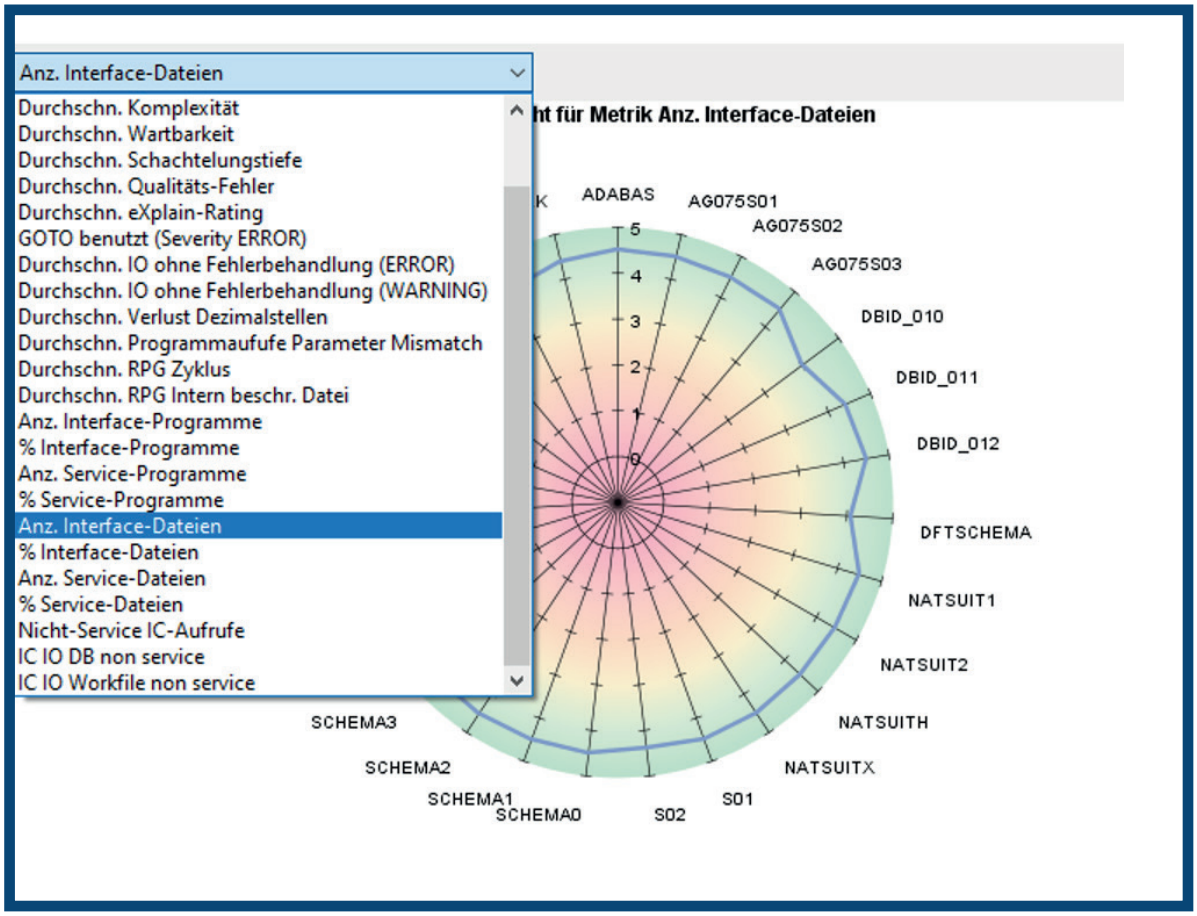
Im Folgenden einige Beispieldiagramme, die einen Eindruck der Möglichkeiten vermitteln.




Cluster
Interfaces/Schnittstellen: Tabelle
Nur APPL Cluster
Leere Zeile ignorieren
Leere Spalten ignorieren
Typ der Cluster-Schnittstelle
IO: Prozentualer Anteil der verwendeten Dateien im Child-Cluster
CALL: Anzahl der Programmaufrufe zwischen den Clustern
CALL: Anzahl der aufrufenden Programme im Parent-Cluster
CALL: Anzahl der aufgerufenen Programme im Child-Cluster
CALL: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
CALL: Prozentualer Anteil der aufgerufenen Programme im Child-Cluster
IO: Anzahl der IO-Calls zwischen den Clustern
IO: Anzahl der aufrufenden Programme im Parent-Cluster
IO: Anzahl der verwendeten Dateien im Child-Cluster
IO: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
IO: Prozentualer Anteil der verwendeten Dateien im Child-Cluster
PARENTCID
DBID
XDEMO
XFLOW
XIMPACT
XNEIGH
XOTHER
XSQLDB
XWORK
1 NATSUIT1
2 NATSUIT2
3 NATSUITH
4 NATSUITX
5 OUTOFSCOPE
6 XDEMO
7 XFLOW
8 XIMPACT
9 XIO
10 XNEIGH
11 XOTHER
12 XWORK


Interfaces/Schnittstellen: Diagramm
Nur APPL Cluster
Leere Zeile ignorieren
Leere Spalten ignorieren
Typ der Cluster-Schnittstelle
IO: Prozentualer Anteil der verwendeten Dateien im Child-Cluster
CALL: Anzahl der Programmaufrufe zwischen den Clustern
CALL: Anzahl der aufrufenden Programme im Parent-Cluster
CALL: Anzahl der aufgerufenen Programme im Child-Cluster
CALL: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
CALL: Prozentualer Anteil der aufgerufenen Programme im Child-Cluster
IO: Anzahl der IO-Calls zwischen den Clustern
IO: Anzahl der aufrufenden Programme im Parent-Cluster
IO: Anzahl der verwendeten Dateien im Child-Cluster
IO: Prozentualer Anteil der aufrufenden Programme im Parent-Cluster
IO: Prozentualer Anteil der verwendeten Dateien im Child-Cluster
Parent-Cluster
XWORK
XOTHER
XNEIGH
XIO
XIMPACT
XFLOW
XDEMO
NATSUITX
NATSUITH
NATSUIT2
NATSUIT1
DBID_010
DBID_011
DFTSCHEMA
SO2
SYSIBM
XDEMO
XFLOW
XIMPACT
XNEIGH
XOTHER
XSQLDB
XWORK
Child-Cluster


NATSUIT1
Qualitäts-Übersicht für Cluster NATSUIT1
Durchschn. Komplexität
GOTO benutzt (Severity ERROR)
Durchschn. IO ohne Fehlerbehandlung (ERROR)
Durchschn. IO ohne Fehlerbehandlung (WA
Durchschn. Programmgröße (LOC)
Durchschn. Wartbarkeit
Durchschn. Schachtelungstiefe
Durchschn. Programmaufufe Parameter
Durchschn. % Kommentarzeilen
Durchschn. % Tote Zeilen
Durchschn. Verlust Dezimalstellen
Durchschn. Qualitäts-Fehler
Durchschn. RPG Zyklus
Durchschn. RPG Intern beschr. Datei
Nicht-Service IC-Aufrufe
IC IO DB non service
IC IO Workfile non service
% Interface-Dateien
% Interface-Programme
% Service-Dateien
% Service-Programme
0
1
2
3
4
5


Anz. Interface-Dateien
Durchschn. Komplexität
Durchschn. Wartbarkeit
Durchschn. Schachtelungstiefe
Durchschn. Qualitäts-Fehler
Durchschn. eXplain-Rating
GOTO benutzt (Severity ERROR)
Durchschn. IO ohne Fehlerbehandlung (ERROR)
Durchschn. IO ohne Fehlerbehandlung (WARNING)
Durchschn. Verlust Dezimalstellen
Durchschn. Programmaufufe Parameter Mismatch
Durchschn. RPG Zyklus
Durchschn. RPG Intern beschr. Datei
Anz. Interface-Programme
% Interface-Programme
Anz. Service-Programme
% Service-Programme
Anz. Interface-Dateien
% Interface-Dateien
Anz. Service-Dateien
% Service-Dateien
Nicht-Service IC-Aufrufe
IC IO DB non service
IC IO Workfile non service
ht für Metrik Anz. Interface-Dateien
ADABAS
AG075S01
AG075S02
AG075S03
DBID_010
DBID_011
DBID_012
DFTSCHEMA
NATSUIT1
NATSUIT2
NATSUITH
NATSUITX
S01
S02
SCHEMA0
SCHEMA1
SCHEMA2
SCHEMA3
0
1
2
3
4
5

Warum Sie mit eXplain die beste Wahl treffen, wenn Sie ein Tool zur Qualitätsanalyse brauchen:

Neben den gängigen Funktionalitäten eines Quality Tools verfügt einzig eXplain über das „Unique Feature“ der Einbeziehung von fachlichen Sichten in das Thema Qualität: spezielle Clustermetriken und v.a. Cluster-Interface-Metriken, die es in dieser Form sonst nirgends gibt und welche unverzichtbar bei Projektvorhaben wie Konsolidierung und Rückbau von Schnittstellen, z.B. als Basis/Voraussetzung für teilweise Einführung von Standardsoftware sind.

Eine weitere Unterscheidung zu anderen Tools und Anbietern liegt in der Fähigkeit und Möglichkeit, eXplain an die speziellen Anforderungen der Kunden, Anwender und Unternehmen anzupassen:

- Erstellung zusätzlicher Regeln
- Erstellung spezieller Metriken, insbesondere in Zusammenhang mit den Schnittstellenanalysen

Nicht zuletzt stellt die Interaktion und Beratung der Entwicklerteams und IT-Architekten eine wesentliche Komponente dar, denn vom Wiegen allein wird die Sau nicht fett! PKS bietet Ihnen mit einem dedizierten Expertenteam die volle Unterstützung: im Projekt aber auch dauerhaft im Rahmen der Weiterentwicklung und des Betriebs.

Darüber hinaus unterstützen wir Sie auch bei den Schlussfolgerungen und den daraus abzuleitenden Maßnahmen (Analyse-Consulting), denn nur so lassen sich die „trockenen Ergebnisse“ eines Quality-Tools überhaupt bewerten und die für das Unternehmen relevanten und finanziell interessanten Aspekte ableiten und umsetzen.

PKS
people knowing software.

PKS - people knowing software.

Wir entwickeln geschäftskritische Businessanwendungen.
Heute, für morgen.

Roland Zurawka
Geschäftsführender Gesellschafter
+49 (0) 751 56140 222
roland.zurawka@pks.de

Heidi Schmidt
Geschäftsführende Gesellschafterin
+49 (0) 751 56140 229
heidi.schmidt@pks.de

Joachim Sieber
Projektverantwortlicher
+49 (0) 751 56140 224
joachim.sieber@pks.de

Bernd Butscher
Senior Developer
+49 (0) 751 56140 257
bernd.butscher@pks.de

www.pks.de