

Inhalt

- Fragestellungen
- Analysen und deren Anwendung
- Erfahrungen

Projektleiter



- Hat unsere Software eine klare, verständliche Struktur?
- Gibt es problematischen Code, der letzte Woche schlechter geworden ist?
- Hält sich unser Subcontractor an die Vorgaben?

Architekt



- Entspricht die Implementierung der geplanten Architektur?
- Hat jemand gestern Code implementiert, der die Architektur bricht?
- Konvergiert die IST- gegen die SOLL- Architektur während unserem Umbau?
- Werden meine Komponenten ausschliesslich über EJBs verwendet?

Qualitätsspezialist



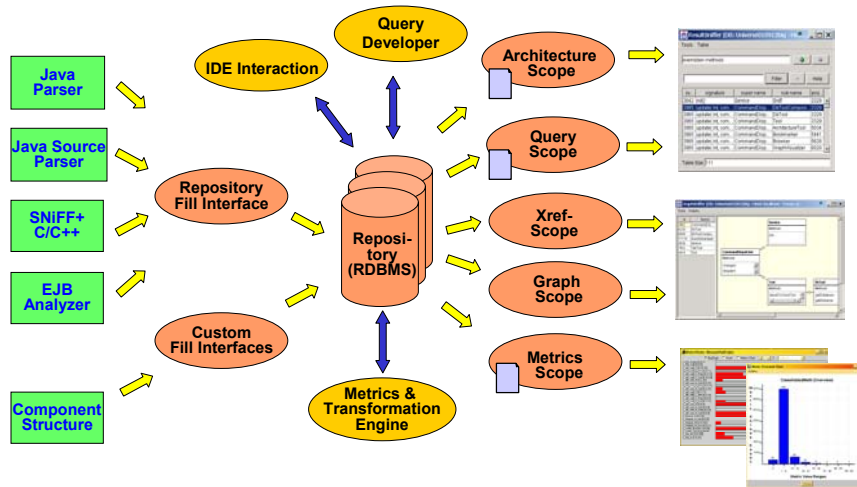
- Halten sich die Ingenieure an die Qualitätsvorgaben?
- Wie gehe ich mit der Datenflut um, die mein Metrikwerkzeug liefert?
- Kann ich einfach eigene Metriken definieren?

Softwareingenieur



- Wie brauchen Klienten meinen Code?
- Ich muss etwas basierend auf einer Komponente implementieren. Wie wird diese Komponente durch ihre typischen Klienten benutzt/erweitert?
- Welche Subsysteme/Packages/Klassen werden durch eine Änderung betroffen?

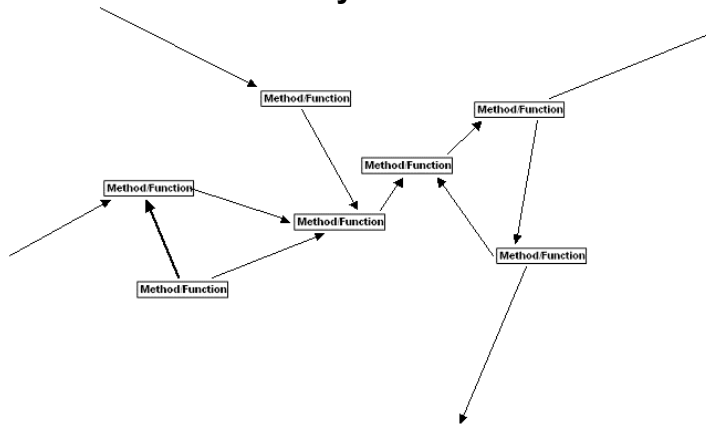
Architektur



Software -Analyse mit Sotograph

- Subsysteme
- Architekturanalyse
- Metrikbasierte Analyse
- Code Comprehension

Navigieren auf Abstraktionsebenen: Symbole

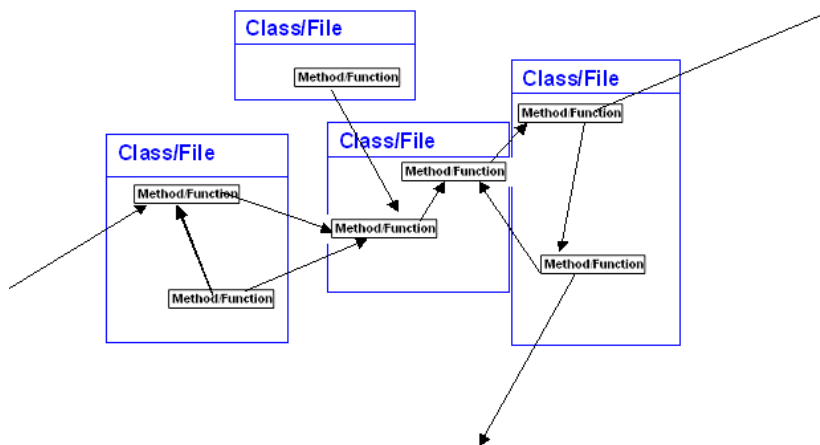


Pfeile stellen beliebige Abhängigkeitsbeziehungen dar:

- Call Referenzen (auch polymorph)
- Vererbung
- Typ Verwendung
- Lese/Schreibzugriff auf Variablen/Attribute

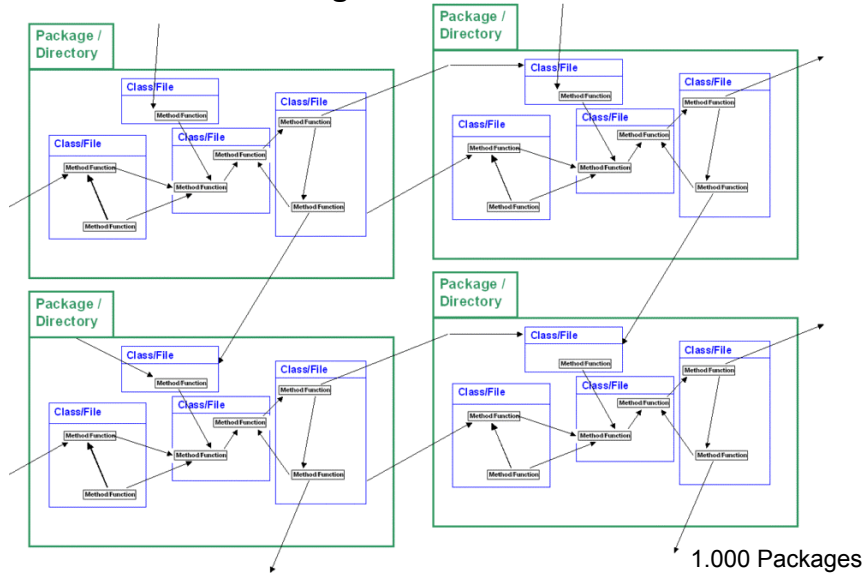
2.000.000 LOC
200.000 Symbole

Navigieren auf Abstraktionsebenen: Klassen / Files



20.000 Klassen

Navigieren auf Abstraktionsebenen: Packages / Directories

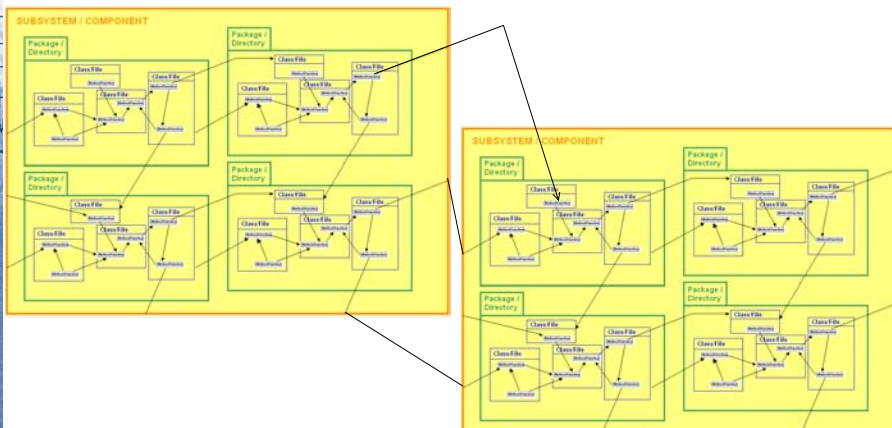


Dr. Walter Bischofberger

Software-Tomography GmbH © 2003

11

Navigieren auf Abstraktionsebenen: Subsysteme / Komponenten

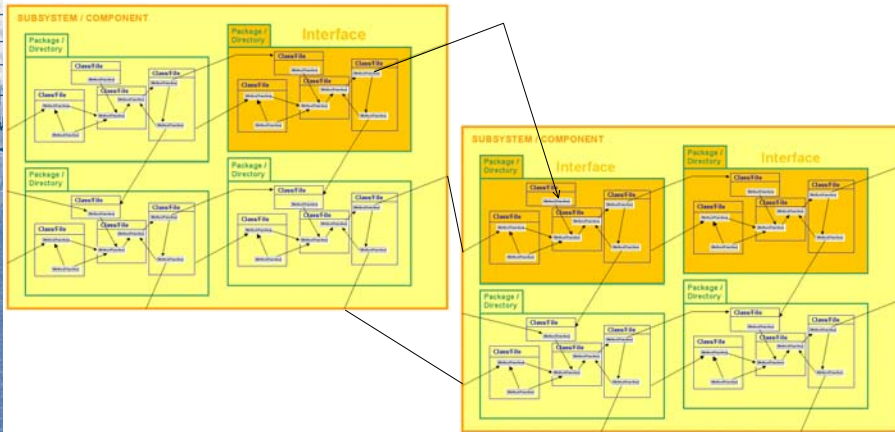


Dr. Walter Bischofberger

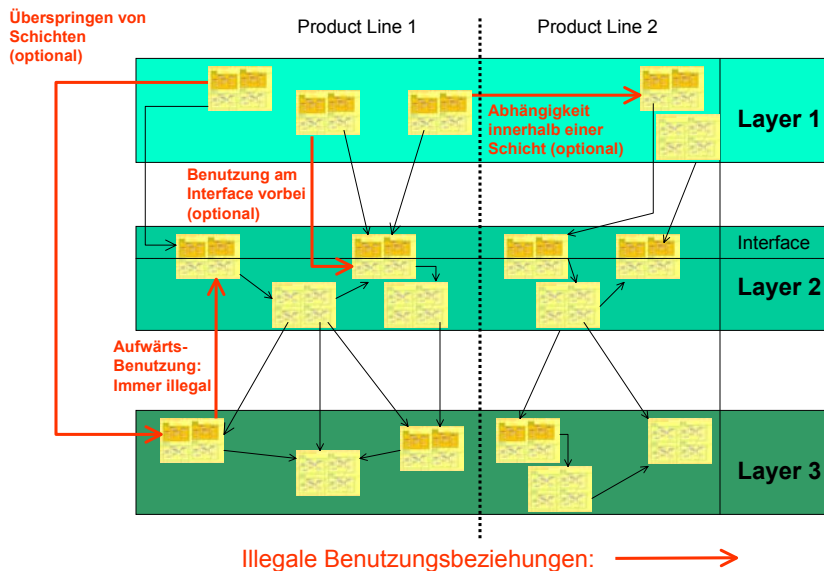
Software-Tomography GmbH © 2003

12

Navigieren auf Abstraktionsebenen: Subsysteme / Komponenten Interfaces



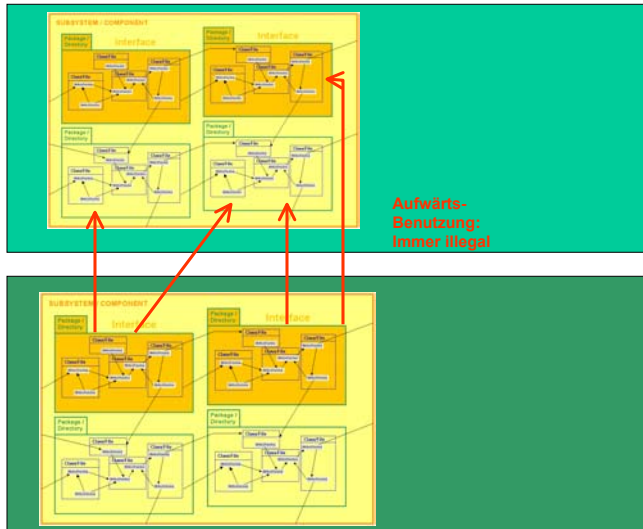
Schichtenarchitektur



Verfolgung illegaler Beziehungen:

Zoomen bis in den Code

Von Subsystemen
nach
Packages/Directories
nach
Klassen/Files
nach
Methoden/Funktionen
bis zum Sprung in die
entsprechende
Source-Code-Stelle



Illegale Benutzungsbeziehungen: →

Beispiel Architekturbeschreibung

```
ArchitectureModel Sotograph {
  Trend True; // Calculate differences between different version.
  Uses Default; // Architecture model is based on the subsystem model Default.
  ArchitectureLayer Manager {
    // Artifacts in layer are allowed to use all layers below.
    InterLayerUsage = True;
    // Artifacts within the layer are allowed to use each other.
    IntraLayerUsage = True;
    Subsystem Tools.manager;
  }
  ArchitectureLayer ToolsAndServices {
    InterLayerUsage = True;
    // Artifacts within the layer are NOT allowed to use each other.
    IntraLayerUsage = False;
    SubsystemsByPackagePath "sotogra/qualit/tools/[a-l-n-z].*";
    Subsystem Tools.metric;
  }
  ArchitectureLayer ToolInfrastructure { ... }
  ArchitectureLayer Frameworks { ... }
  ArchitectureLayer Toolkits {
    Subsystem Base.util;
    Subsystem Db;
  }
}
```


Illegale Beziehungen - Subsystemebene

Exceptions of Trend Architecture Model Sotograph for v1 = 0.95 and v2 = 0.96

sub...	subRefing	sub...	subRefed	errorKind	v1	v2	diff
65911	Tools.result	65908	Tools.manager	UPWARD	3	3	0
65912	Tools.subsystem	65900	Base.util	INTERFACE	42	42	0
65900	Base.util	65899	Base.tool	UPWARD	0	2	2
65910	Tools.query	65900	Base.util	INTERFACE	94	96	2
65900	Base.util	65908	Tools.manager	UPWARD	56	60	4
65902	Tools.architecture	65900	Base.util	INTERFACE	38	42	4
65899	Base.tool	65895	Base.guiutil	INTERFACE	31	36	5
65902	Tools.architecture	65913	Tools.trend	INTRA	74	80	6
65901	Db	65899	Base.tool	UPWARD	738	802	64

Table Size: 27

Referencing Subsystem Referenced Subsystem 0.95 0.96 Differenz

Illegale Beziehungen Details

■ Package-Ebene

Trend architecture exceptions between subsystems Tools.architecture and Base.util

pck...	pckgRefing	pck...	pckgRefed	errorKind	v1	v2	diff
288	architecture	281	jflex	INTERFACE	15	19	4
290	parser	281	jflex	INTERFACE	23	23	0

■ Referenz-Ebene

architecture exceptions between package architecture and jflex for v1 = 0.95 and v2 = 0.96

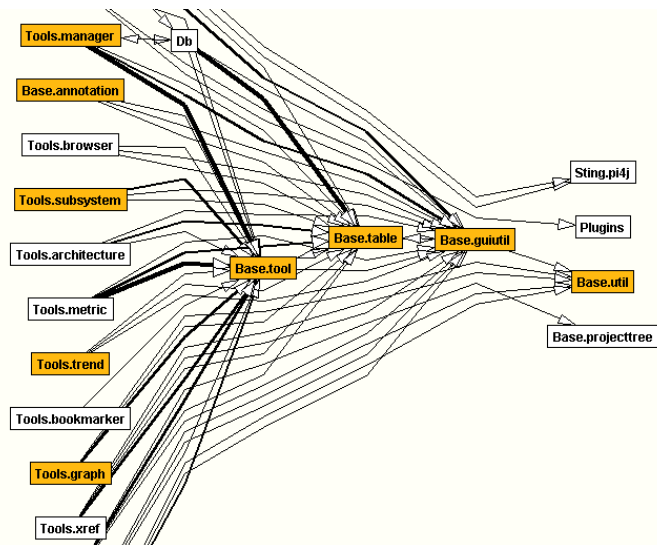
refi...	refingSymbol	refe...	refedSymbol	referenceType	loc...	ref...	changes
189...	me_parseArchitect...	189...	me_getLine()	CALL	LO...	104...	NEW
189...	me_parseArchitect...	189...	me_getColumn()	CALL	LO...	104...	NEW
189...	me_parseArchitect...	189...	cl_SyntaxErrorExc...	TYPEACCESS	LO...	104...	SAME
189...	me_parseArchitect...	189...	cl_SyntaxErrorExc...	TYPEACCESS	LO...	104...	SAME

Illegale Beziehungen Details

■ Source Code

```
private boolean parseArchitectureModel() {
    ArchitectureParser p= new ArchitectureParser(getDatabase());
    try {
        p.parse(getSelectedModel(), getArchitectureGUI().getSource());
        // getSelectedModel().display();
    } catch (SyntaxErrorException e) {
        sfCat.error("ArchitectureSyntaxErrorException - Line " + e.getLine() + "." + e.ge
getArchitectureGUI().selectEditorLineCol(e.getLine(), e.getColumn()-1, e.getColumn
getArchitectureGUI().showErrorMessage("Line " + e.getLine() + "." + e.getColumn()
return false;
    }
    return true;
}
```

Illegal referenzierte Subsysteme



Metrik- und regelbasierte Analyse

■ Arten

- Architekturmetriken
 - Z.B. Stabilitätsmetriken von Robert C. Martin
- Grössen- und Kopplungsmetriken
 - Z.B. Anzahl Klassen pro Package, Anzahl verwendeter Packages
- Komplexitätsmetriken
 - Z.B. Cyclomatic Complexity
- Regeln
 - Z.B. eine Klasse darf die von ihr abgeleiteten Klassen nicht kennen
- Bad Smells
 - Z.B. Flaschenhalse, unbenutzte Artefakte

Metrik- und Regelbasierte Analyse

■ Schwierigkeiten

- Sehr grosse Menge Messwerte
 - **Filtern**
- Verstehen der Ursache eines Messwerts
 - **Erklärungen**
 - **Visualisierung** im Kontext des Softwaresystems
- Das Untersuchen einer Version genügt oft nicht
 - **Trend-Unterstützung**
- Definieren eigener Metriken
 - Ermöglicht durch **Einfaches Datenmodell im Repository**

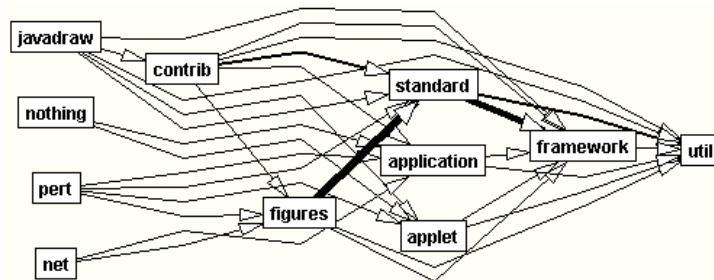
Strukturanalyse

■ Fragestellungen

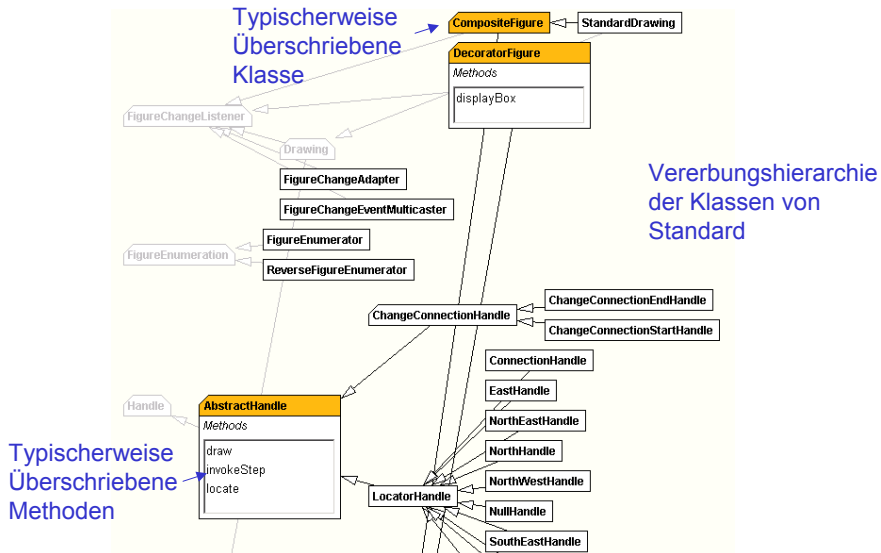
- Was ist die Struktur meines Softwaresystems?
- Wie wird Subsystem X typischerweise benutzt oder erweitert?
- Welche Teile meiner Anwendung sind betroffen wenn ich die Schnittstelle von Subsystem X ändere?
- Gibt es Bad Smells?
- Gibt es instantiierte Entwurfsmuster?

Was ist die Struktur meines Softwaresystems?

- Z.B. Vererbungsbeziehungen aus Packag-Ebene
 - Dicke der Pfeile zeigt Stärke der Kopplung



Typische Überschreiberschnittstelle von Pckg Standard



Dr. Walter Bischofberger

Software-Tomography GmbH © 2003

25

Impact Analyse

- Z.B. mit dem XrefScope
 - Erlaubt es beliebige Beziehungen zwischen Artefakten auf unterschiedlichen Abstraktions Ebenen zu untersuchen.
 - Typische Fragestellungen
 - Welche Subsysteme sind betroffen wenn ich eine Klasse verändere
 - Welche Klassen erben von einer Klasse dieses Packages
 - Oder für C: welche Directories hängen von einem File ab.

Dr. Walter Bischofberger

Software-Tomography GmbH © 2003

26

XrefScope

■ Welche Klassen erben von einer Klasse von Standard

Focus

Packages

Pa...	Package Name	Short Package Path
178	standard	CH.ifa.draw.standard

Reference Properties

Direction

Refers To Referred By Comprises Defined In

Reference Kinds

Call Polymorphic Call Inheritance Read EJB Call EJB Inheritance Containment Write Type Access

Select All Select None

Result Granularity

Classes

Overview

Classes referring to Package standard Eval Eval Detailed

Analyseabfragen

- Bad Smells
 - Flaschenhalse
 - Abstrahierbare Methoden und Attribute
 - Bidirektionale Abhängigkeiten
- Entwurfsmuster
 - Factories, Wrappers, Bridges, Composites, Template Methods, Singletons
- Testabdeckung
 - Klassen die nicht direkt von JUnit Testklassen angesprochen werden

Entwickeln eigener Abfragen, Regeln und Metriken

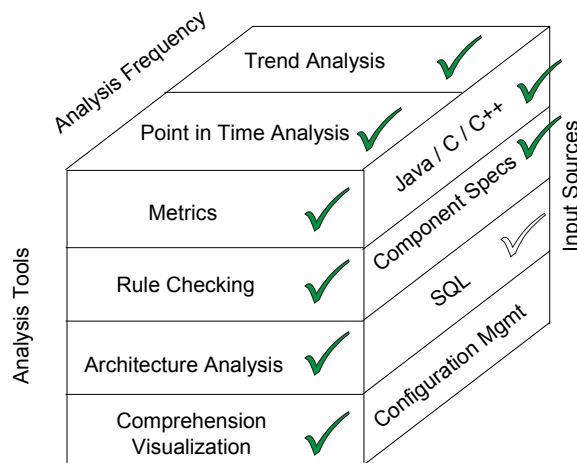
The screenshot shows the JHotDraw QueryDeveloper window. The left sidebar contains a tree view of queries, with 'CommonCallInt' selected under the 'Package' category. The main window displays a query editor with the following SQL code:

```

WHERE acalled.packageId = Focus.getId("packageId")
AND ccalled.symbolId = acalled.classId
AND acalled.signatureId = sref.refedSignatureId
AND sref.referenceType = 'CALL'
AND sref.refingSymbolId = acaller.symbolId
AND ccaller.symbolId = acalled.classId
AND acaller.classId <> acalled.classId
#if Input.get(filterConstructors) = "TRUE"
AND acalled.name <> ccalled.name
#endif
AND acaller.packageId = clientPackage.packageId;
    
```

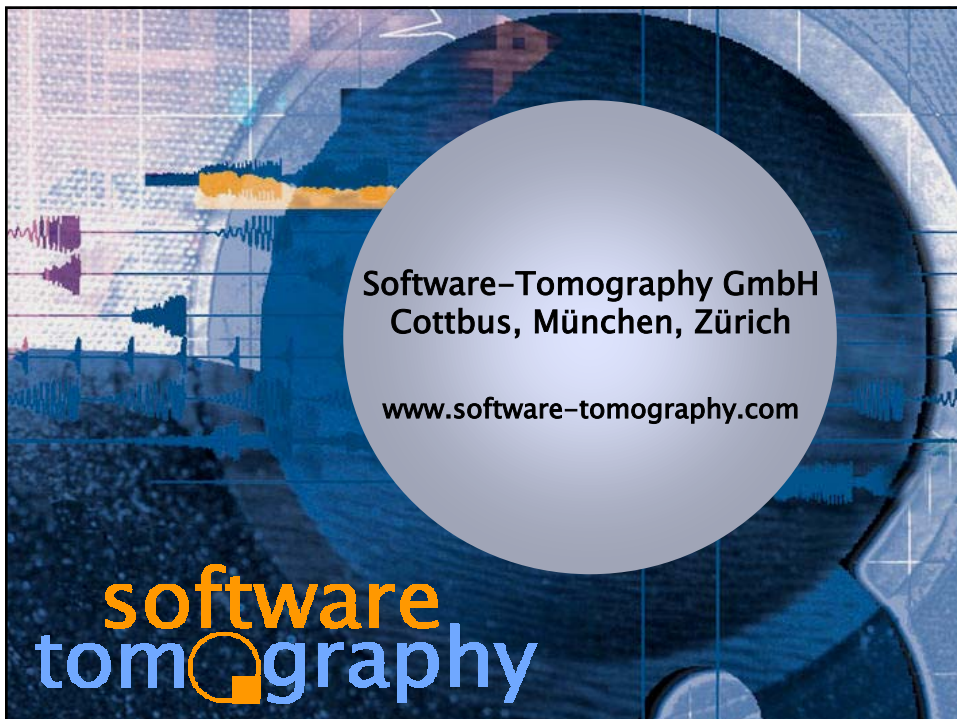
Below the SQL code, there are input fields for 'Client Kind' (set to 'All'), 'Number of Clients' (set to '3'), 'Group Result' (checked), and 'Filter Constructors' (checked). A description box at the bottom explains the query's purpose: "Shows different kinds of common call interfaces of the focus package. Similar to the 'call interface' query. The central difference is that This query only calculates the interface for the 'Number of Clients' best clients. Best client means client package with most call relationships to the focus package."

Dimensionen der Software-Tomographie



Erfahrungen aus der Praxis

- Software Qualitätsanalyse durch SQS
 - Sehr tiefgehende Qualitätsanalysen grosser, komplexer Softwaresysteme mit zwei Personenwochen Aufwand
 - Kann nur durch erfahrene Spezialisten mit Werkzeugunterstützung in so kurzer Zeit durchgeführt werden.
- Architektur-Reengineering bei einer Schweizer Grossbank
 - 2'000'000 LOC 600Mhz Rechner
 - Identifizierung des Problems
 - Überwachung der iterativen Verbesserung
 - Basisanalyse und Einarbeitung: drei Tage



Software-Tomography GmbH
Cottbus, München, Zürich

www.software-tomography.com

software
tomography

The image features a large, stylized graphic of a human head in profile, composed of various colored segments (blue, orange, purple, green) and overlaid with a grid pattern. A large, semi-transparent grey circle is centered over the head, containing the company name and website. The logo at the bottom left consists of the word 'software' in orange and 'tomography' in blue, with a stylized orange and blue 'o' in 'tomography'.