Automatisierte Ermittlung und Berwertung von Subsystemschnittstellen

Gliederung

- *1. Ermittlung von Subsystemschnittstellen*
- *2. Toolentwicklung*
- *3. Bewertung von Schnittstellen*
- *4. Ergebnisse*

Ermittlung von Subsystemschnittstellen

- *Subsystemschnittstellen*
  - *beschreiben die Teile eines Subsystems, die andere Subsysteme benutzen (können)*
  - *tatsächlich benutzte Schnittstelle: Menge der Symbole, die in anderen Subsystemen referenziert wird*

Ermittlung von Subsystemschnittstellen

- *Warum?*
  - *Subsysteme abgrenzen*
  - *Subsystemeinteilung verbessern*
  - *Teile austauschen*
  - *Informationen über die Architektur gewinnen*

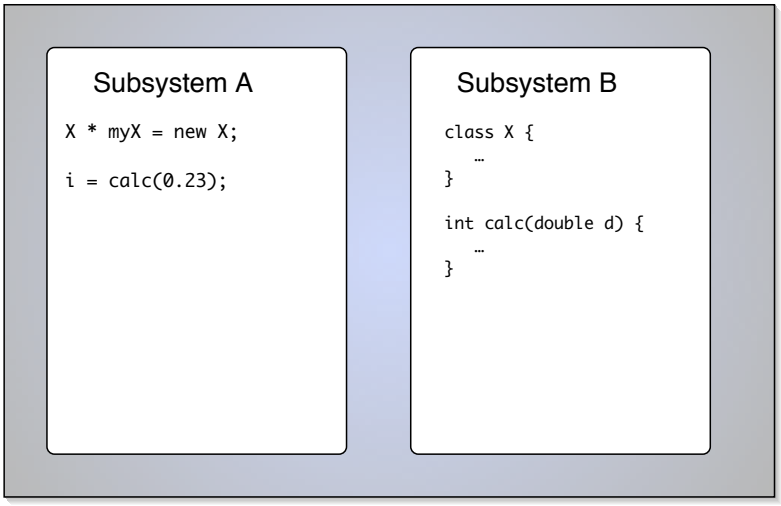
Ermittlung von Subsystemschnittstellen

- *Wie?*
  - *manuell (Studienarbeit): Erstellen von Subsystem-Headerdateien*
    - *umständlich*
    - *zeitaufwendig*
    - *fehleranfällig*
  - *automatisch:*
    - *schnelle Bestimmung der Schnittstellen*
    - *Vergleich von Subsystemeinteilungen möglich*

Ermittlung von Subsystemschnittstellen

- *grundlegendes Problem: Referenzen finden*

  "Welche Symbole (Klassen, Funktionen, etc.) eines Subsystems werden in anderen Subsystemen benutzt?"



Gliederung

- *1. Ermittlung von Subsystemschnittstellen*
- *2. Toolentwicklung*
- *3. Bewertung von Schnittstellen*
- *4. Ergebnisse*

Toolentwicklung

- *grundlegendes Problem: Referenzen finden*

- *Sourcecode muss geparst werden -> zu umständlich*
- *Idee: vorhandenes Tool nutzen, welches diese Referenzen ermittelt*
- *Betrachtung von Sourcecode-Analyse-Tools:*

objectiF

- *unterstützt großen Teil des Softwarentwicklungsprozesses*
  Use-Case-, Klassen-, Aktivitätsdiagramme, Dokumentation, Sourcecode
- *Reverse-Engineering Komponente*
- *Erweiterungemöglichkeiten:*
  COM-Interface für VisualBasic
- *ermittelt Referenzen*
- *aber:*
  - *Einlesen des XCTL-Systems mit Problemen verbunden*
  - *Informationen über Referenzen nicht von außen nutzbar*

SNiFF+

- *Cross-Referencer*
- *SNiFF-API:*
  - *Java Schnittstelle*
  - *liefert sämtliche vorhandene Informationen über:*
    - *das Projekt allgemein*
    - *Dateien*
    - *Symbole*
    - *Referenzen*
- *problemloses Einlesen der XCTL-Quellen*

SNiFF-API

- *class SniffController:*
  - *stellt Verbindung zu SNiFF her:*

```
connect();
disconnect();
```

  - *startet Datenübertragung:*

```
getFileData();
getProjectData();
```

- *class SymbolHandler:*
  - *Funktionen, die Informationen empfangen:*

```
handleClass();
handleFunction();
handle...();
```

- *FileHandler, PreprocessorHandler, ProjectHandler, ReferenceHandler, ScopeHandler*

Toolentwicklung

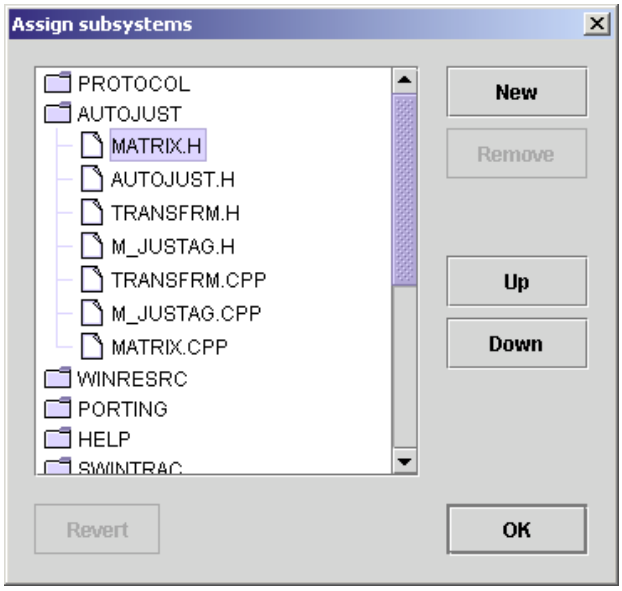
- *eigenes Tool: InterfaceFinder*
- *nutzt SNiFF*
- *Arbeitsweise:*
  - *Verbindung zu SNiFF+ herstellen*
  - *Datei-, Symbol- und Referenzinformationen übertragen*
  - *Zuordnung der Subsysteme*
  - *Generierung der Schnittstellen*
  - *Ausgabe der Schnittstellen*

Anforderungen an InterfaceFinder

- *Aufgabe:*
  Ermitteln der tatsächlich benutzen Subsystemschnittstellen
- *dazu notwendig:*
  - *Informationsübertragung von SNiFF*
  - *Datenspeicherung*
  - *Berechnen der Schnittstellen*
  - *Ausgabe der Schnittstellen*
- *Subsystemzuordnung*
- *grafische Benutzeroberflche*

Subsystemzuordnung

- *jede Datei wird einem Subsystem zugeordnet*
- *erste Zuordnung wird von Verzeichnisstruktur bestimmt*
- *Dialog erlaubt Veränderung der Zuordnung*

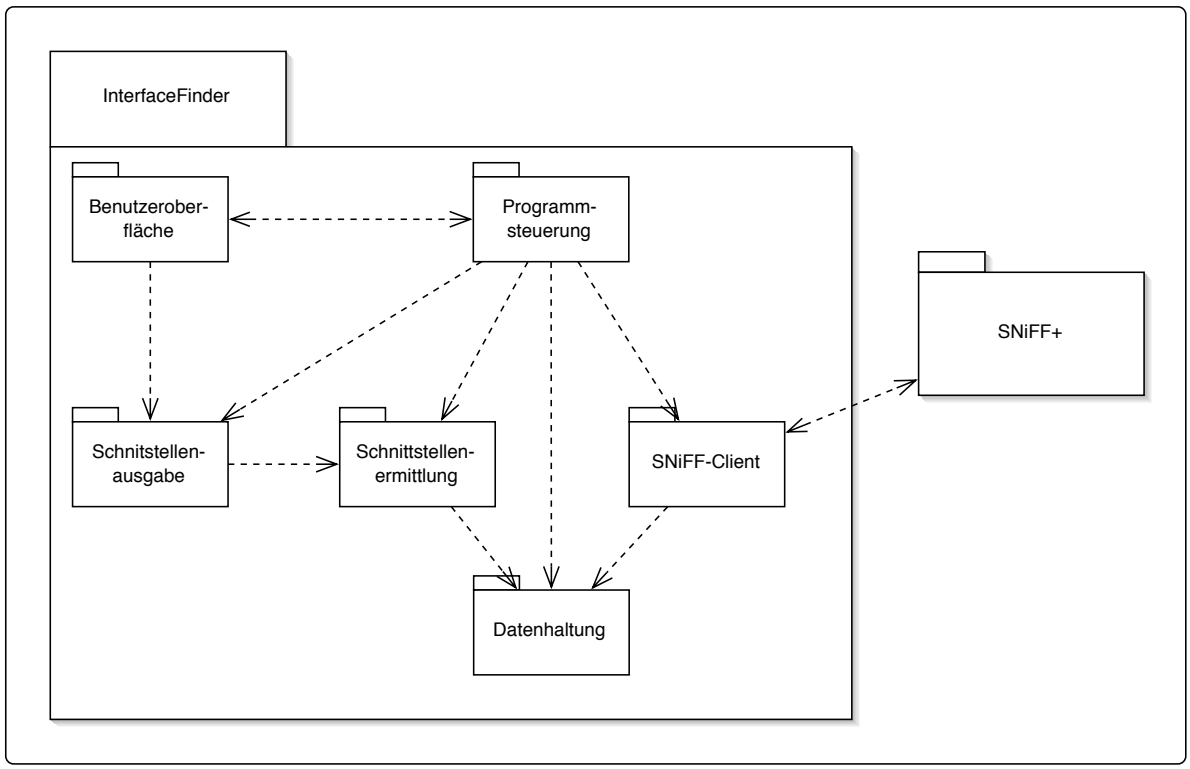


Ausgabe der Ergebnisse

- *tabellarisch*
- *Export in Textdatei*
- *Erweiterungen möglich*

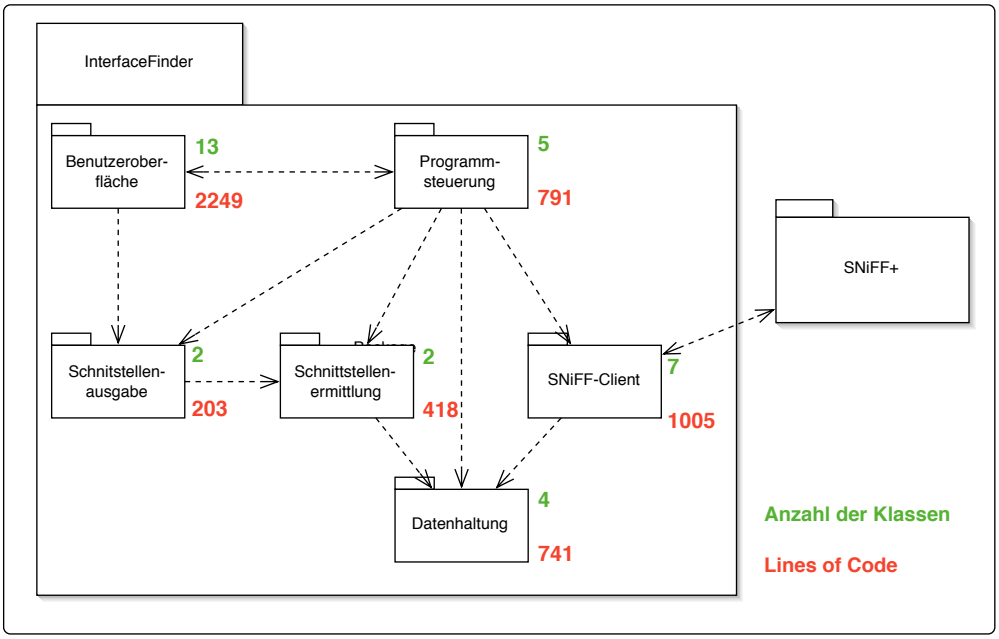
Type	Name	File
define	DegToRad	DATAVISA.H
define	ArToRLx	DATAVISA.H
define	ArToRLy	DATAVISA.H
class	TCurve	DATAVISA.H
enum	TOutputType	DATAVISA.H
class	TPlotData	DATAVISA.H
class	TBitmapSource	DATAVISA.H
typedef	LPCurve	DATAVISA.H
class	TDataBase	DATAVISA.H
typedef	LPDataBase	DATAVISA.H
struct	TDisplay	DATAVISA.H
typedef	TKSystem	DATAVISA.H
enum	TSaveFormat	DATAVISA.H
typedef	TData	M_DATA.CPP
struct	TCoorSystem	DATAVISA.H
struct	TBMContens	DATAVISA.H
struct	CPoint	DATAVISA.H
class	TCurveShowParam	DATAVISA.H
variable	lpDataBase	M_CURVE.CPP
enum	TOrder	DATAVISA.H

Architektur



Implementation

- *33 Java-Klassen*
- *5407 Lines of Code*



Test

- *getrennter Test von Funktionalität und Oberfläche*
- *Funktionalität:*
  - *Testsystem:*
    einfaches C++ Programm, zwei Subsysteme
    -> Schnittstelle ablesbar
  - *XCTL:*
    Vergleich der Ergebnisse der Studienarbeit (Headerdateien) mit berechneten Schnittstellen
  - *beide Tests erfolgreich*
- *Oberfläche*
  - *Testfälle: Aktionen und erwartete Reaktionen,*
    jedes GUI-Element wird mindestens einmal aktiviert
  - *auch erfolgreich*

Anwendung

- *Vorbereitung: SNiFF-Projekt anlegen*
- *InterfaceFinder starten*
- *Projekt öffnen (Sniff .proj-Datei)*
- *Subsystemzuordnung bearbeiten*
- *Berechnung abwarten*
- *Ergebnisse betrachten*

Demonstration

Gliederung

- *1. Ermittlung von Subsystemschnittstellen*
- *2. Toolentwicklung*
- *3. Bewertung von Schnittstellen*
- *4. Ergebnisse*

Bewertung von Schnittstellen und Subsystemeinteilungen

- *mehrstufige Betrachtung:*
  - *einzelne Schnittstellen*
  - *einzelne Subsystemeinteilung*
  - *Vergleich von Subsystemeinteilungen*
- *verschiedene Größen bestimmt*

Bewertung von Schnittstellen und Subsystemeinteilungen

- *einzelne Schnittstellen:*
  - *Struktur der Schnittstelle - Anzahl der verschiedenen Symbole*
  - *Rückschlüsse auf Programmierstil*
  - *Klassen vs. Funktionen*
- *einzelne Subsystemeinteilung:*
  - *Größe der Schnittstelle:*
  - *absolut = Anzahl der Symbole*
  - *relativ zur Subsystemgröße*
  - *-> Vergleich der Subsysteme*

Bewertung von Schnittstellen und Subsystemeinteilungen

- *Vergleich von Subsystemeinteilungen:*
  - *Einteilung als Ganzes bewerten*
  - *Durchschnitt der relativen Schnittstellengröße*
  - *Bezugswert: zufällige Einteilung*
  - *nur Vergleich von Einteilungen mit gleicher Anzahl der Subsysteme*

Bewertung von Schnittstellen und Subsystemeinteilungen

- *Untersuchung des XCTL-Systems :*
  - *normale Einteilung*
  - *zufällige Einteilung*
  - *Drei-Schichten-Einteilung*

Gliederung

- *1. Ermittlung von Subsystemschnittstellen*
- *2. Toolentwicklung*
- *3. Bewertung von Schnittstellen*
- *4. Ergebnisse*

Ergebnisse

- *InterfaceFinder*
  - *erfüllt Aufgabe*
  - *Einschränkungen*
  - *Erweiterungmöglichkeiten*
- *Bewertung von Schnittstellen*
- *Schnittstellen des XCTL-Systems*

Ergebnisse/InterfaceFinder

- *Einschränkungen:*
  - *C++ Sprachumfang nicht vollständig*
  - *Operatorüberladung*
  - *Undefined References*
  - *gleichnamige Symbole*
  - *Makros*
- *-> Einschränkungen beruhen auf Problemen von SNiFF*

Ergebnisse/InterfaceFinder

- *Erweiterungsmöglichkeiten:*
  - *Speichern von Subsystemeinteilungen*
  - *Statistik*
  - *Ausgabe*
  - *Sprachumfang*
  - *Ersetzen von SNiFF+*

Ergebnisse/Bewertung

- *Schnittstellen liefern Erkenntnisse*
  - *Architektur*
  - *Programmierstil*
- *XCTL-System:*
  - *Einteilung bestätigt*
  - *fehlende Einheitlichkeit*

Gliederung

- *1. Ermittlung von Subsystemschnittstellen*
- *2. Toolentwicklung*
- *3. Bewertung von Schnittstellen*
- *4. Ergebnisse*