

Die Columbo Architektur

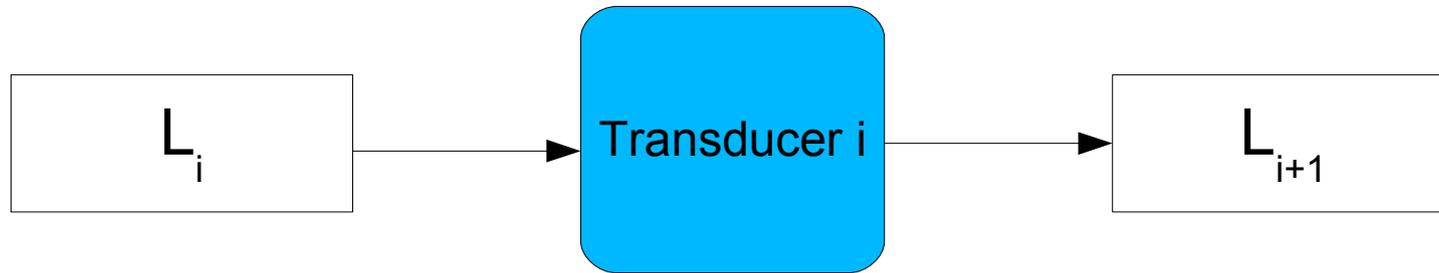
oder

Die Suche nach dem formalen Esperanto

Jens.Doll@uni-hamburg.de

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

1.1 Transduktionen



d_i :

Definitionen

L_i : w_i

β -Reduktion bzgl. der d_i

L_{i+1} : $\lambda d_1 d_2 \dots d_n w_{i+1}$

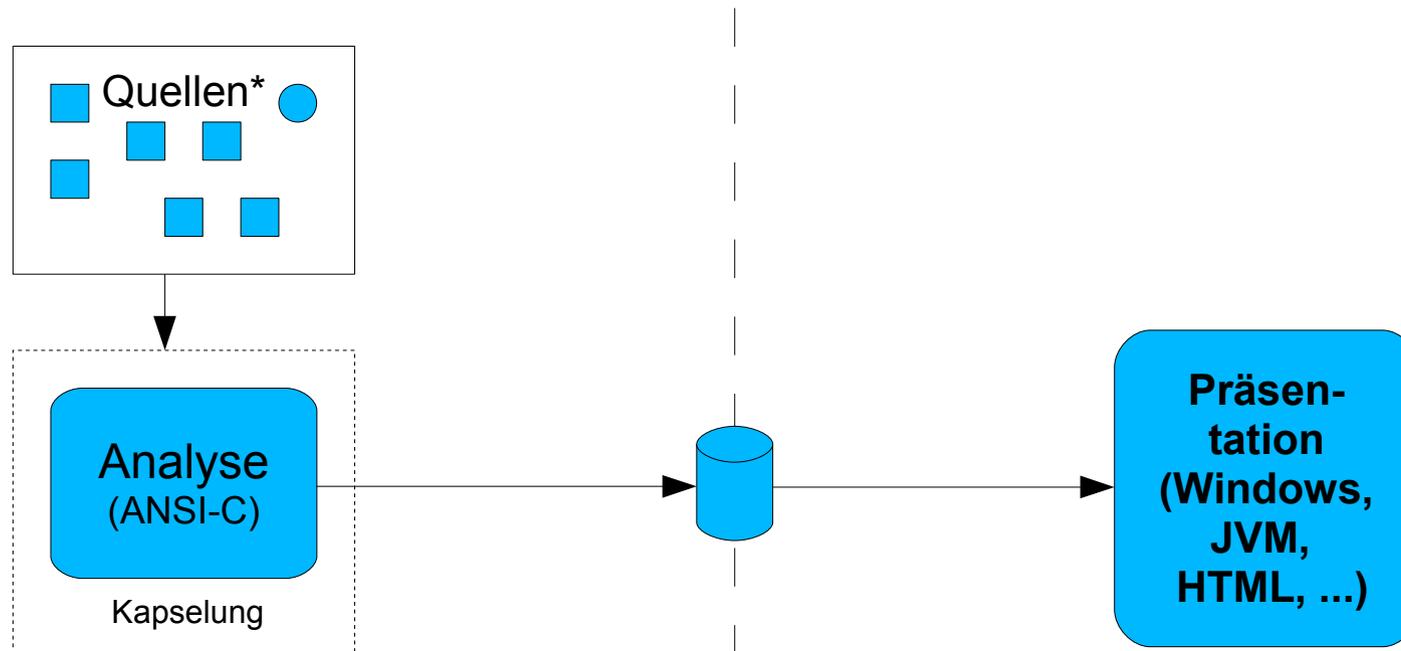
d_i sind α -Abstraktionen

endlich, regulär phrasenstrukturiert

(TRS für Normalisierung und für Fixpunktlogik)

- 1 Elemente
- 2 Aufbau
- 3 Prozeß
- 4 Ziel

1.2 Technische Details



Mainframe
oder
Server
oder
PC

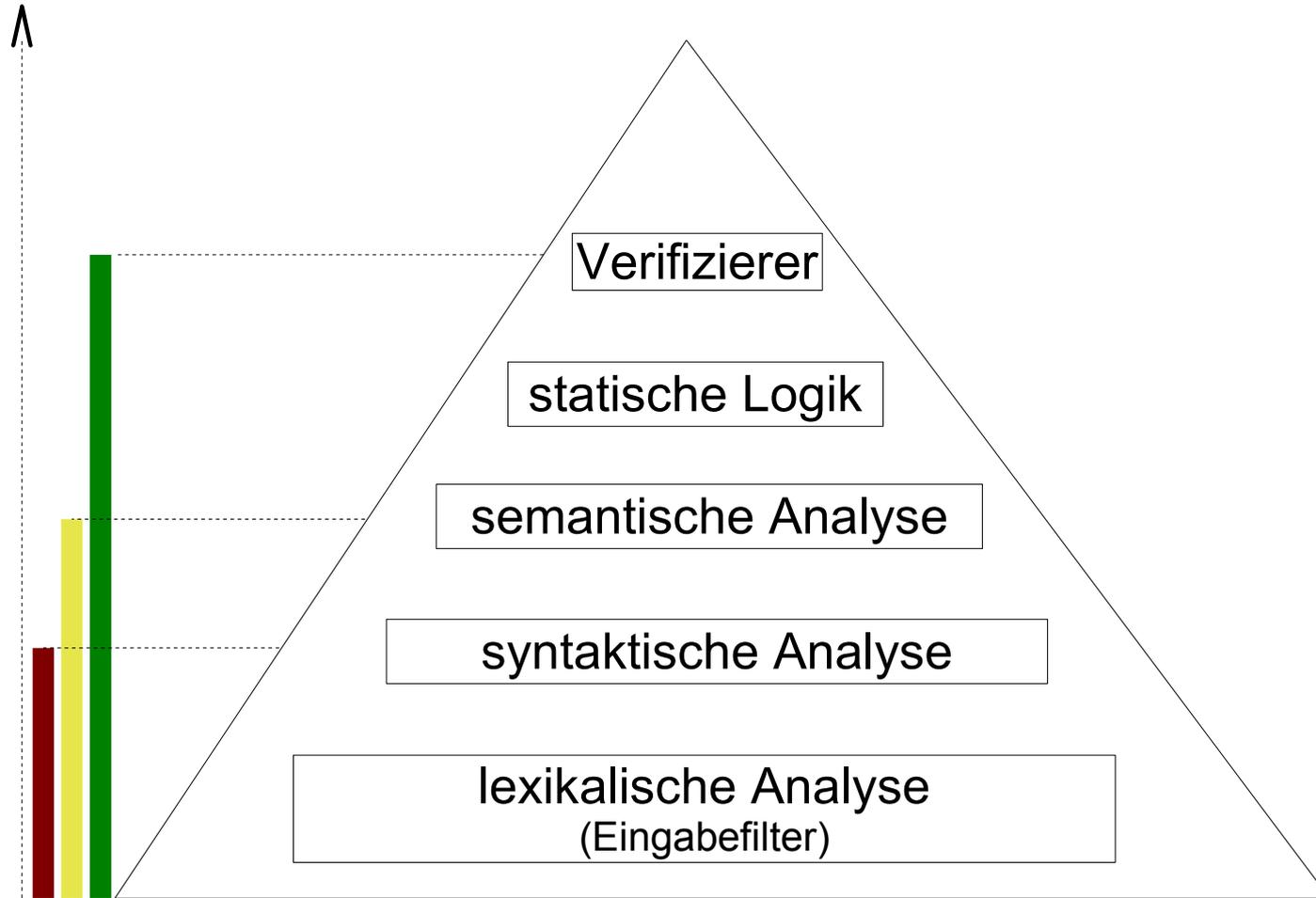
Desktop

XML-Schnittstellen für:
Hightite, Navigation, Semantik, CFA, DFA, ...

*inkl. SQL

- 1 Elemente
- 2 Aufbau
- 3 Prozeß
- 4 Ziel

2.1 Formalsprachliche Bausteine



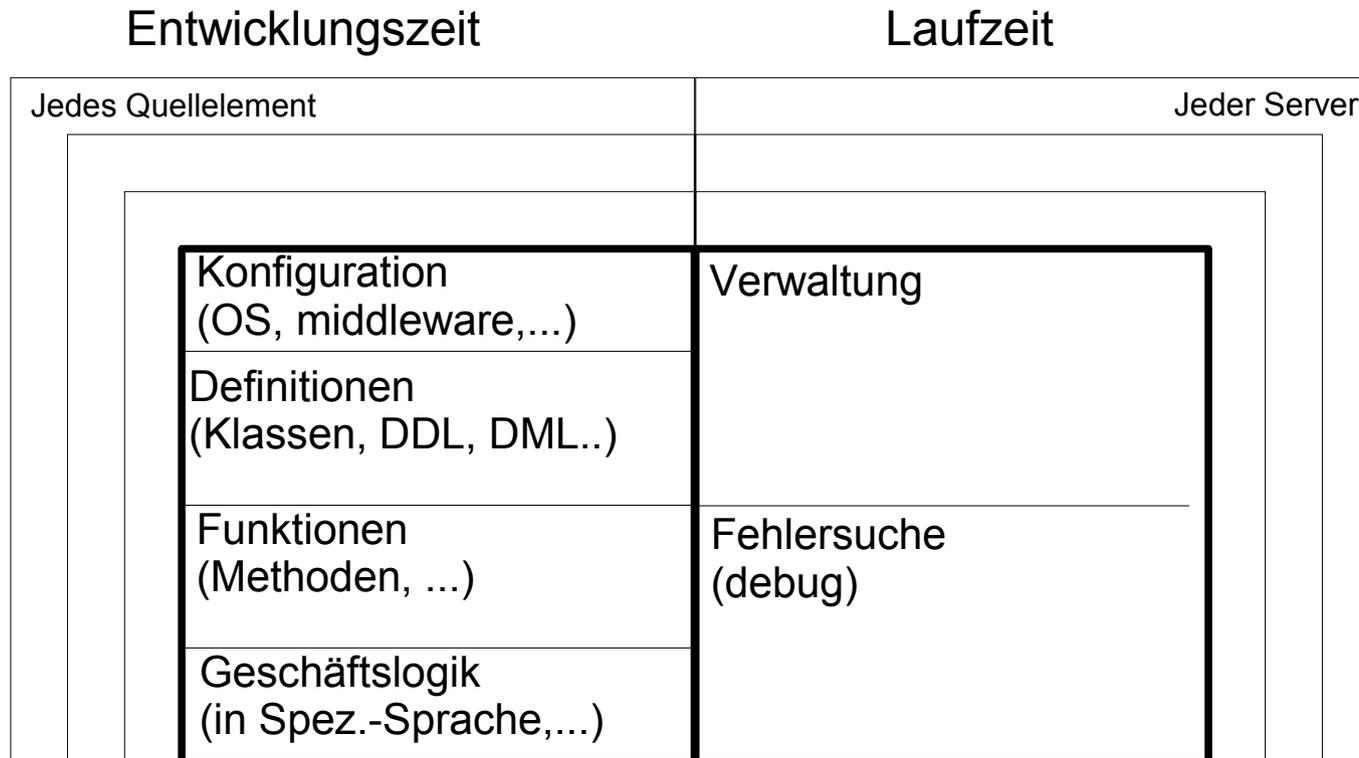
- 5 → **Spezifikationsabweichungen**
- 4 → **statische Fehler, CFA-, DFA-Grafik**
- 3 → **Goto-Programme, Aktionen**
- 2 → **Navigator, Syntaxprüfung**
- 1 → **Highlighting, Includes***

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

- █ = Stufe 1, Standard Edition = Produkt Elbe
- █ = Stufe 2, Professional Edition
- █ = Stufe 3, Enterprise Edition

*copy, import, include

2.2 Nutzerschnittstellen

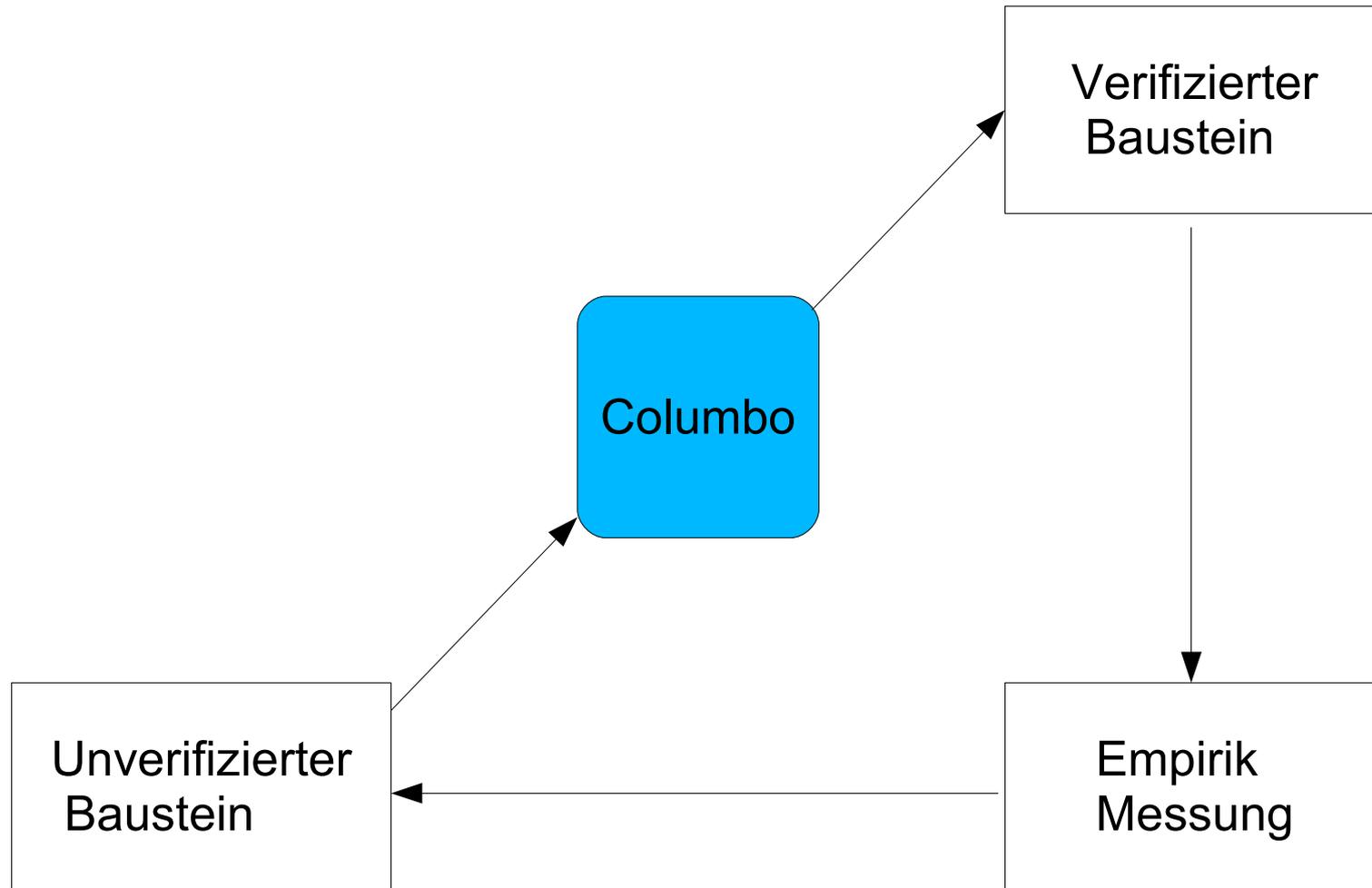


Gruppierung nach
Verzeichnis,
Bibliothek,
Schema

Gruppierung nach
Server,
Verzeichnis

- 1 Elemente
- 2 Aufbau
- 3 Prozeß
- 4 Ziel

3.1 Iterationen



1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

KVP=kontinuierlicher Verbesserungsprozeß

4.1 Stand des Projektes

Elbe 1.0

Columbo 0.7

Webinterface 0.3

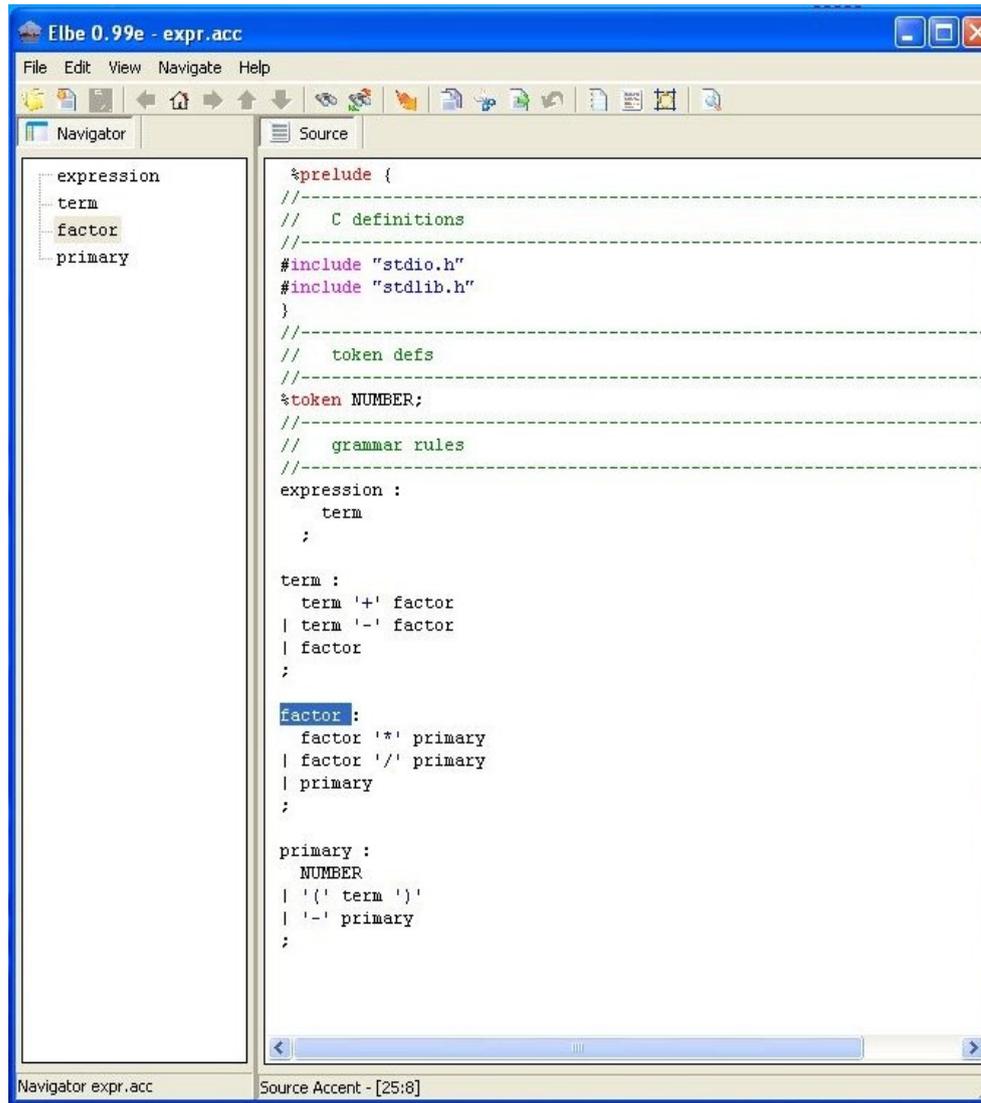
Columbo

SherlockHolmes

-Methode

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

4.2.1 Beispiel: Accent-Datei



```
Elbe 0.99e - expr.acc
File Edit View Navigate Help
Navigator Source
expression
term
factor
primary


```

$prelude {
//-----
// C definitions
//-----
#include "stdio.h"
#include "stdlib.h"
}
//-----
// token defs
//-----
%token NUMBER;
//-----
// grammar rules
//-----
expression :
 term
 ;

term :
 term '+' factor
| term '-' factor
| factor
;

factor :
 factor '*' primary
| factor '/' primary
| primary
;

primary :
 NUMBER
| '(' term ')'
| '-' primary
;

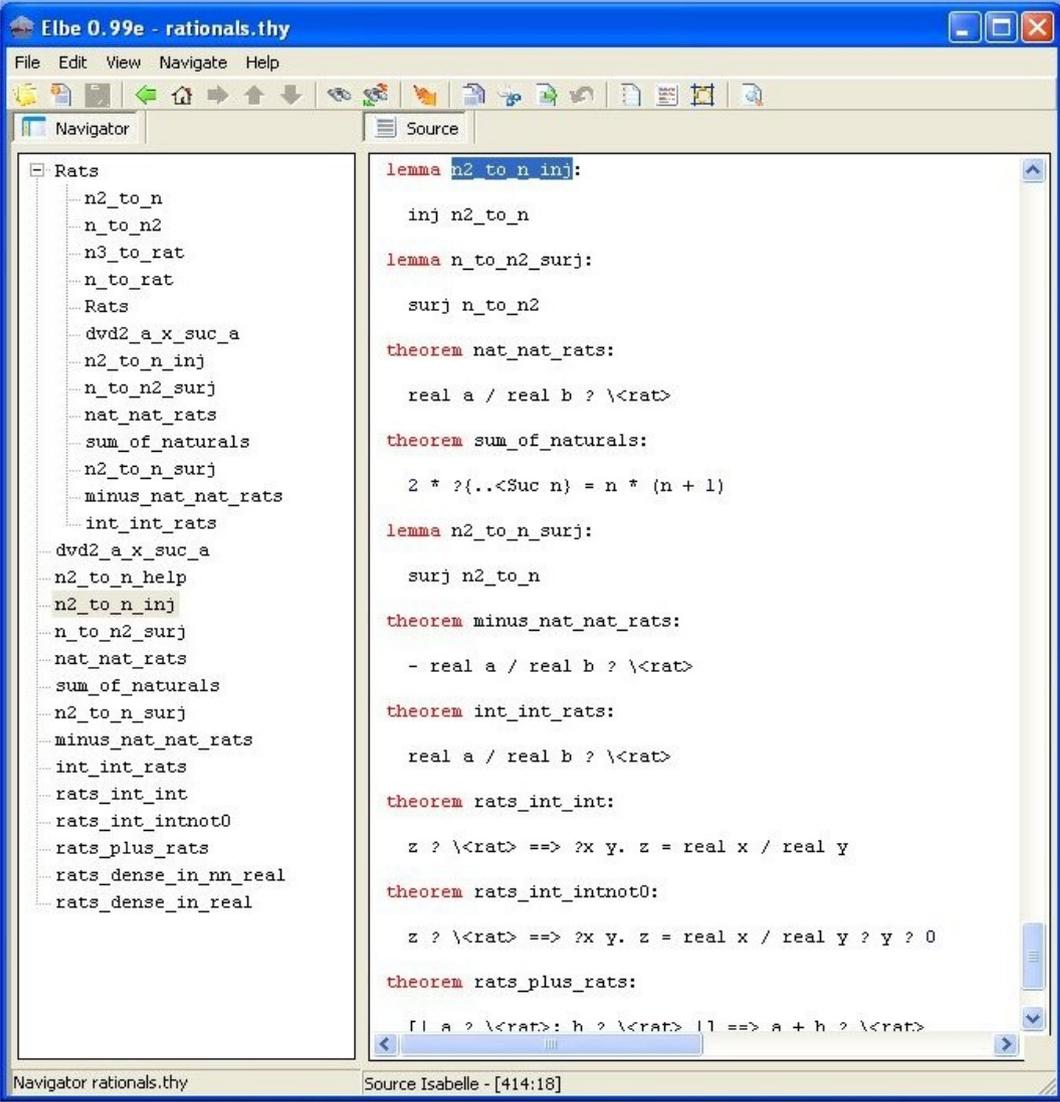
```


```

- 1 Elemente
- 2 Aufbau
- 3 Prozeß
- 4 Ziel

Download unter <http://cococo.de>

4.2.2 Beispiel: Isabelle-Datei



```
Elbe 0.99e - rationals.thy
File Edit View Navigate Help
Navigator Source
Rats
├─ n2_to_n
├─ n_to_n2
├─ n3_to_rat
├─ n_to_rat
├─ Rats
├─ dvd2_a_x_suc_a
├─ n2_to_n_inj
├─ n_to_n2_surj
├─ nat_nat_rats
├─ sum_of_naturals
├─ n2_to_n_surj
├─ minus_nat_nat_rats
├─ int_int_rats
├─ dvd2_a_x_suc_a
├─ n2_to_n_help
├─ n2_to_n_inj
├─ n_to_n2_surj
├─ nat_nat_rats
├─ sum_of_naturals
├─ n2_to_n_surj
├─ minus_nat_nat_rats
├─ int_int_rats
├─ rats_int_int
├─ rats_int_intnot0
├─ rats_plus_rats
├─ rats_dense_in_nn_real
└─ rats_dense_in_real

lemma n2_to_n_inj:
  inj n2_to_n

lemma n_to_n2_surj:
  surj n_to_n2

theorem nat_nat_rats:
  real a / real b ? \<rat>

theorem sum_of_naturals:
  2 * ?(<Suc n) = n * (n + 1)

lemma n2_to_n_surj:
  surj n2_to_n

theorem minus_nat_nat_rats:
  - real a / real b ? \<rat>

theorem int_int_rats:
  real a / real b ? \<rat>

theorem rats_int_int:
  z ? \<rat> ==> ?x y. z = real x / real y

theorem rats_int_intnot0:
  z ? \<rat> ==> ?x y. z = real x / real y ? y ? 0

theorem rats_plus_rats:
  [| a ? \<rat>; h ? \<rat> |] ==> a + h ? \<rat>
```

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

Download unter <http://cococo.de>

4.2.3 Beispiel: Webinterface

[source/Datei](#)



Quellcode-Bibliothek

Datei counter-goto.cob

```
*****
* Testbibliothek für COBOL
* Die Korrektheit wird nicht zugesichert.
* Falls nicht im Text anders spezifiziert gilt:
* (c) Context IT GmbH, Email: info@coroco.de
*****
identification division.
program-id. counter.
author. "JD".
date-written. 25.10.2004.
date-compiled.
data division.
working-storage section.
  77 i      pic 9(4).
linkage section.
  77 s      pic 9(8).
procedure division returning s.
  * computes s=Summe(1..100)
  move 1 to i
  move 0 to s.
  labl.
  if (i > 100) goto ende.
  add i to s
  add 1 to i
  goto labl.
  ende.
end-program counter.
```

 angemeldet



Untersuchungsergebnis

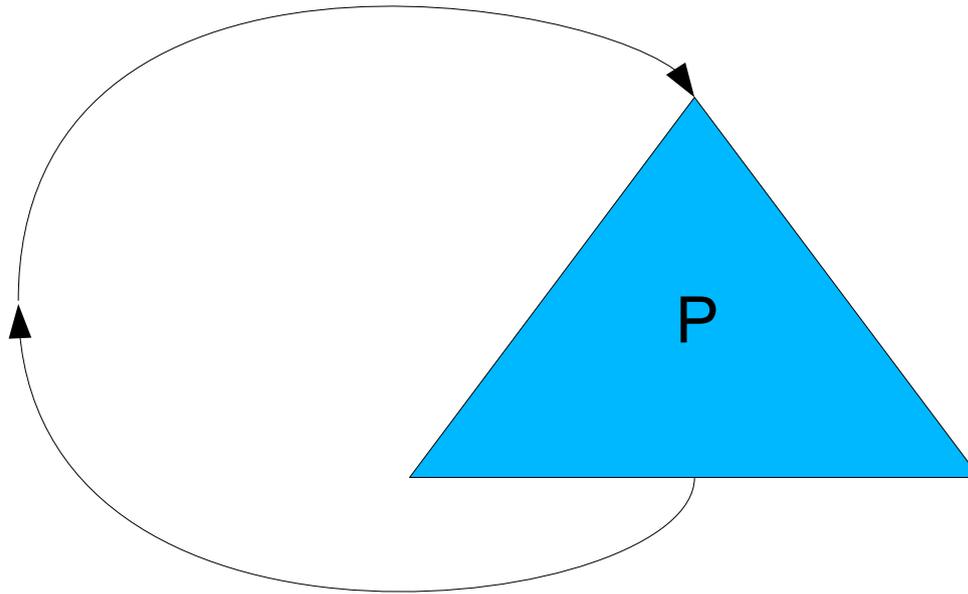
```
--- 1: Datei counter-goto.cob am 30.9.2010
10:55:55:29 ---
```

```
Funktion counter
-----
Anzahl Zeilen: 19
Anteil Kommentare (Datei): 5 %
Anzahl Statements: 5
Anzahl Entscheidungen: 7
Anzahl definierter Felder: 2
Anzahl interner Felder: 6
Analyseseit(real): 10 msec
Analysespeicherbedarf: 405 kB
Größe des Programmes
Arbeitsspeicher: 44 Bytes
Komplexität
McCabe Maß: 24
Aufwand
Halstead Maß: 34
Function Point Aufwand: 1.5 PM
C-Maß: 256.0 DB
```

```
Ergebnis s [0 .. 10^8-1]
s = 5050
Meldungen
W5309? Überlauf möglich
```

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel

4.3 Zielvorstellung („in the limit“)



Das Halteprogramm P
Q(P) Quelle
O(P) Objekt

...auf dem Weg zum eleganten Beweissystem = Elbe

Vielen Dank für Ihre Aufmerksamkeit

1 Elemente
2 Aufbau
3 Prozeß
4 Ziel