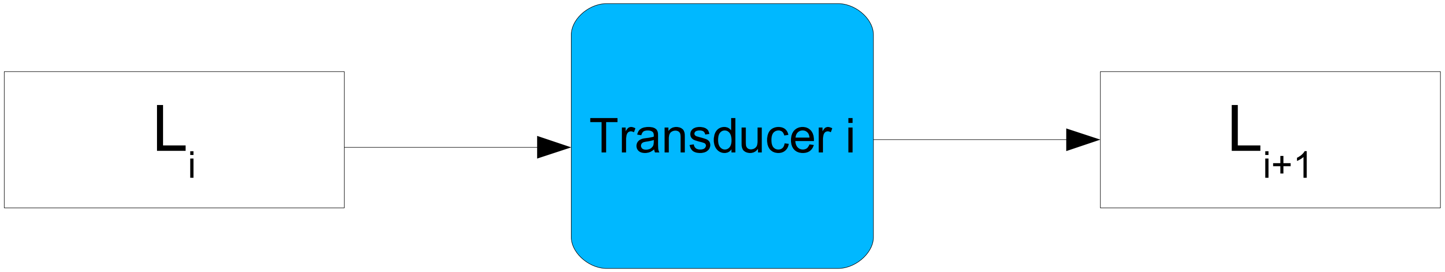
The Columbo Architecture

or

On Search for a Formal Esperanto

Context IT GmbH
www.cococo.de

1.1 Transductions



L_i

Transducer i

L_{i+1}

d_i: definitions

L_i: w_i β reductions for d_i

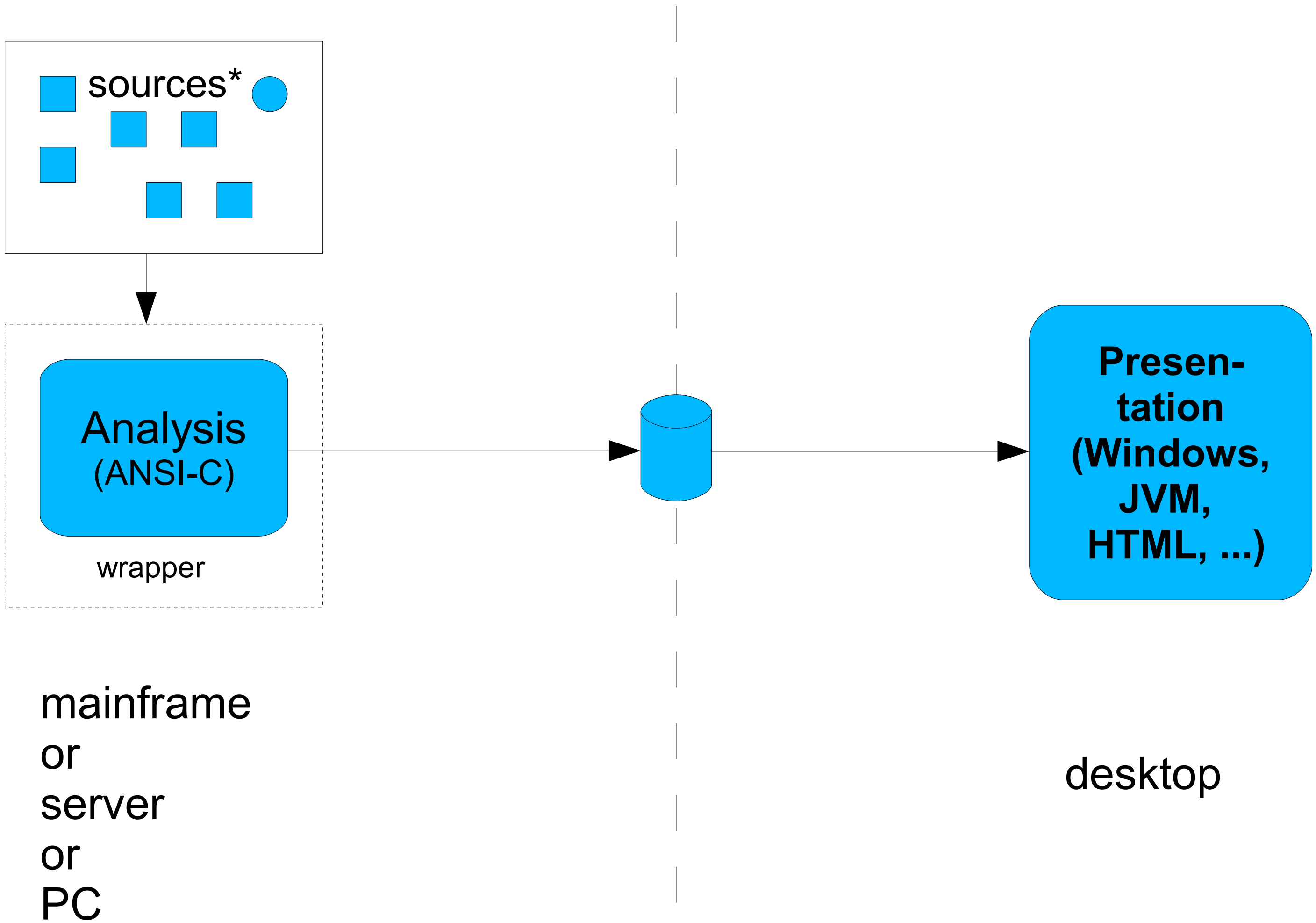
L_{i+1}: $\lambda\, d_1 d_2 \ldots d_n\ w_{i+1}$ d_i are α abstractions

finite, regular .... phrase structured

(TRS for normalisation and for fixed points logic)

1 Elements
2 Structure
3 Process
4 Target

1.2 Technical Details



mainframe
or
server
or
PC

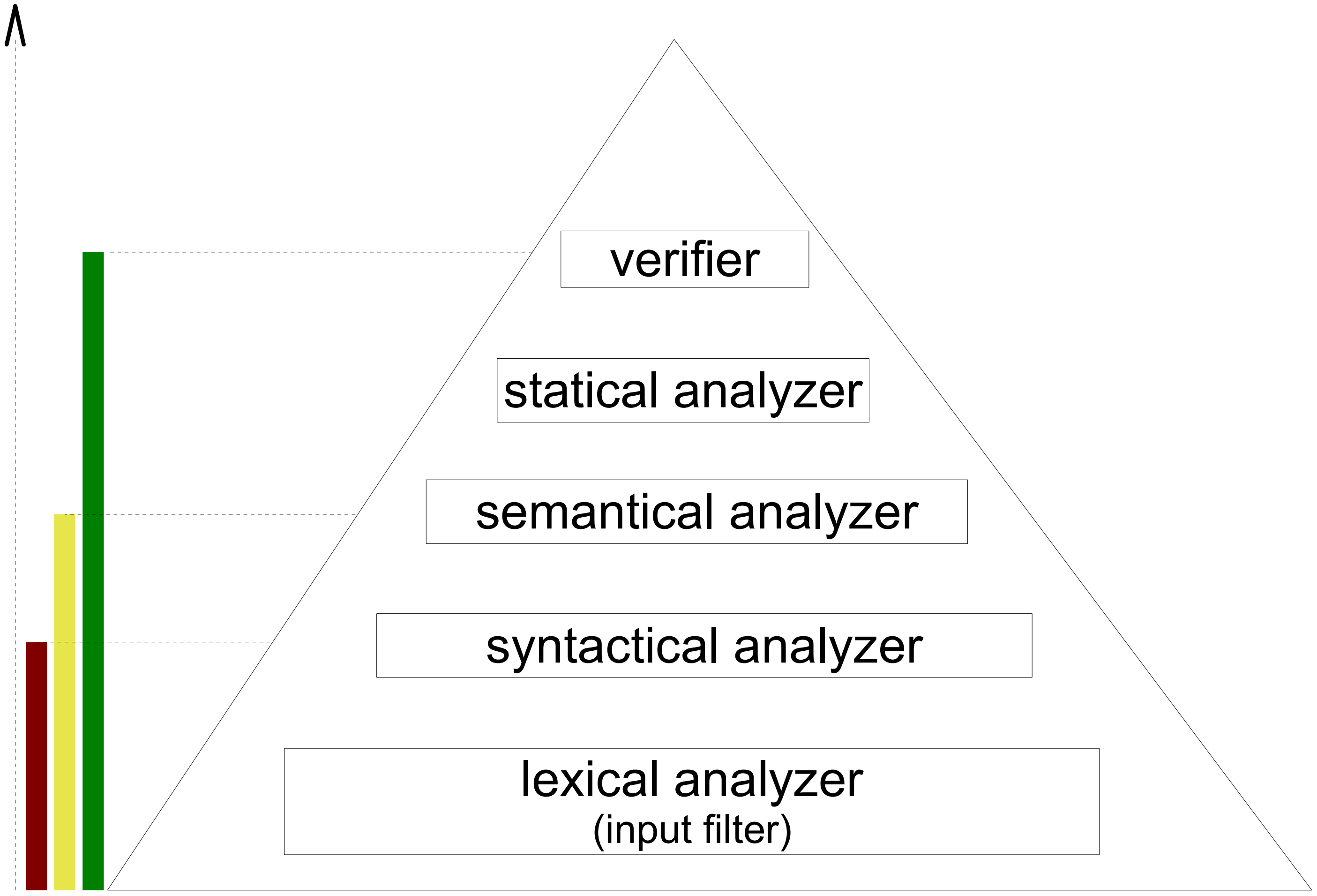
desktop

XML interfaces for:
highlite, navigation, semantics, CFA, DFA, ...

*incl. SQL

1 Elements
2 Structure
3 Process
4 Target

2.1 Language Elements



verifier

statical analyzer

semantical analyzer

syntactical analyzer

lexical analyzer
(input filter)

5 → spezification
errors

4 → statical errors,
CFA , DFA graphs

3 → Goto programs,
actions

2 → navigator,
syntax check

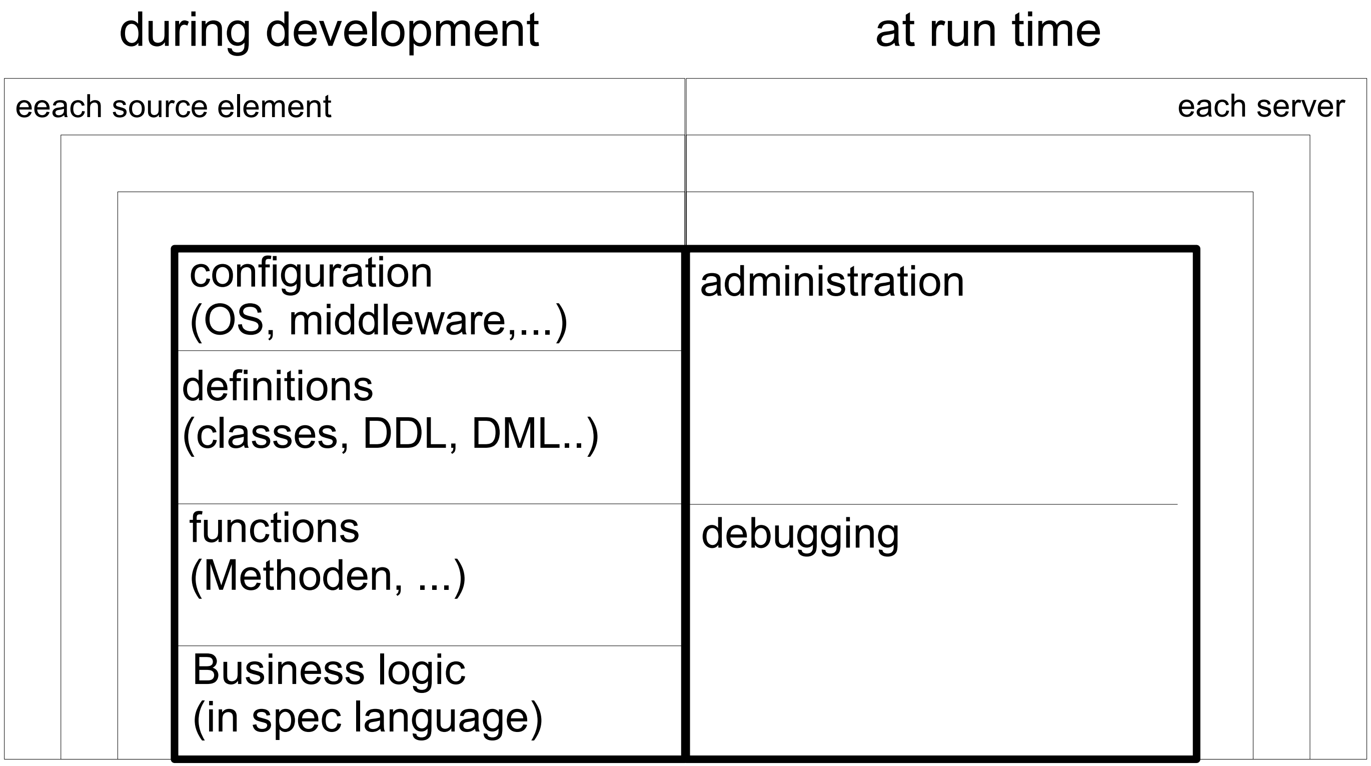
1→ highlight,
includes*

🟥 = level 1, standard edition = Elbe product
🟨 = level 2, professional edition
🟩 = level 3, enterprise edition

*copy, import,
include

1 Elements
2 Structure
3 Process
4 Target

2.2 User Interfaces



during development

at run time

eeach source element

each server

configuration
(OS, middleware,...)

definitions
(classes, DDL, DML..)

functions
(Methoden, ...)

Business logic
(in spec language)

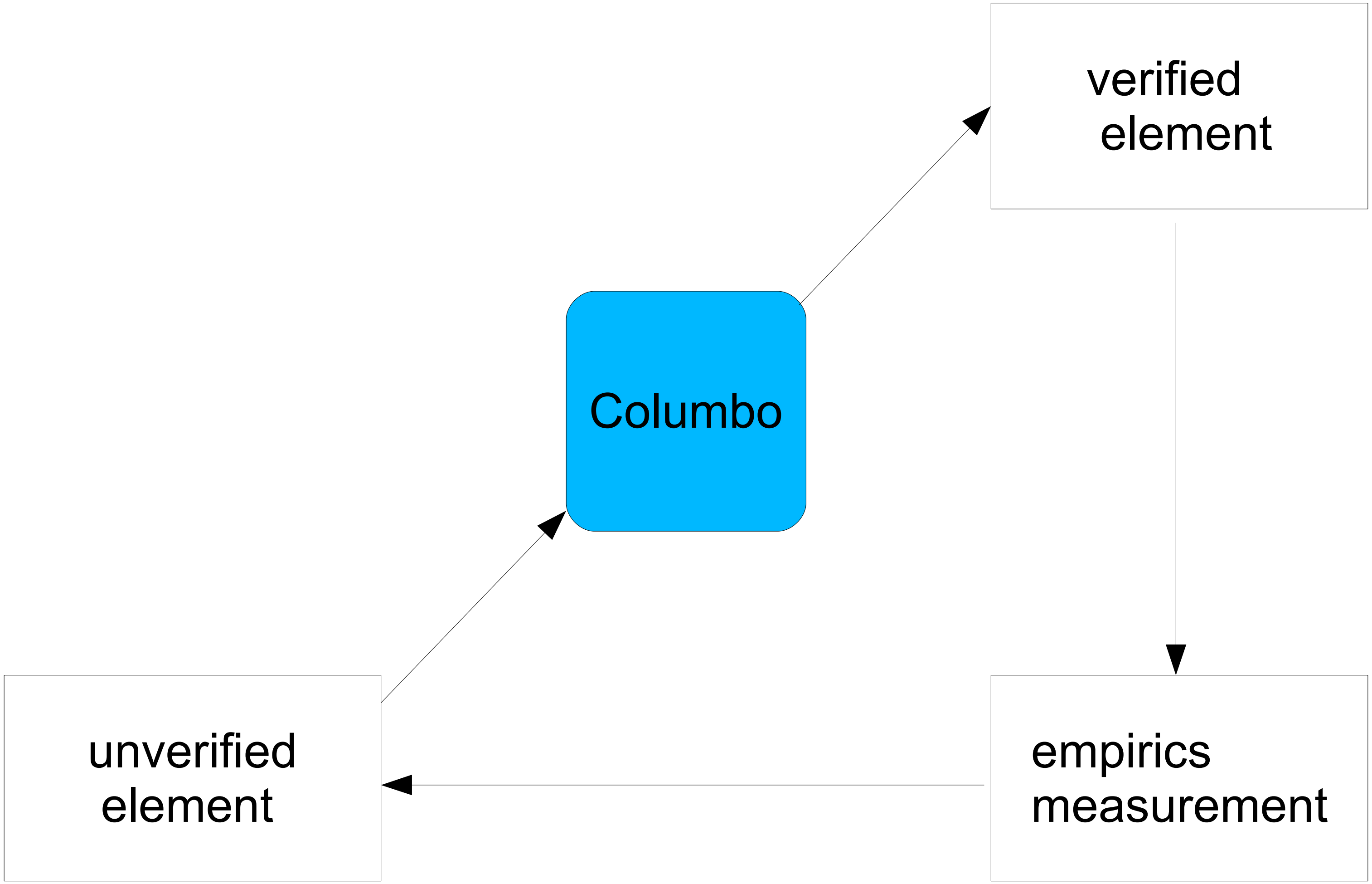
administration

debugging

Group by
directory,
library,
schema

Group by
server,
directory

3.1 Iteration



TQM=continous improvement

1 Elements
2 Structure
3 Process
4 Target

4.1 State of the Art

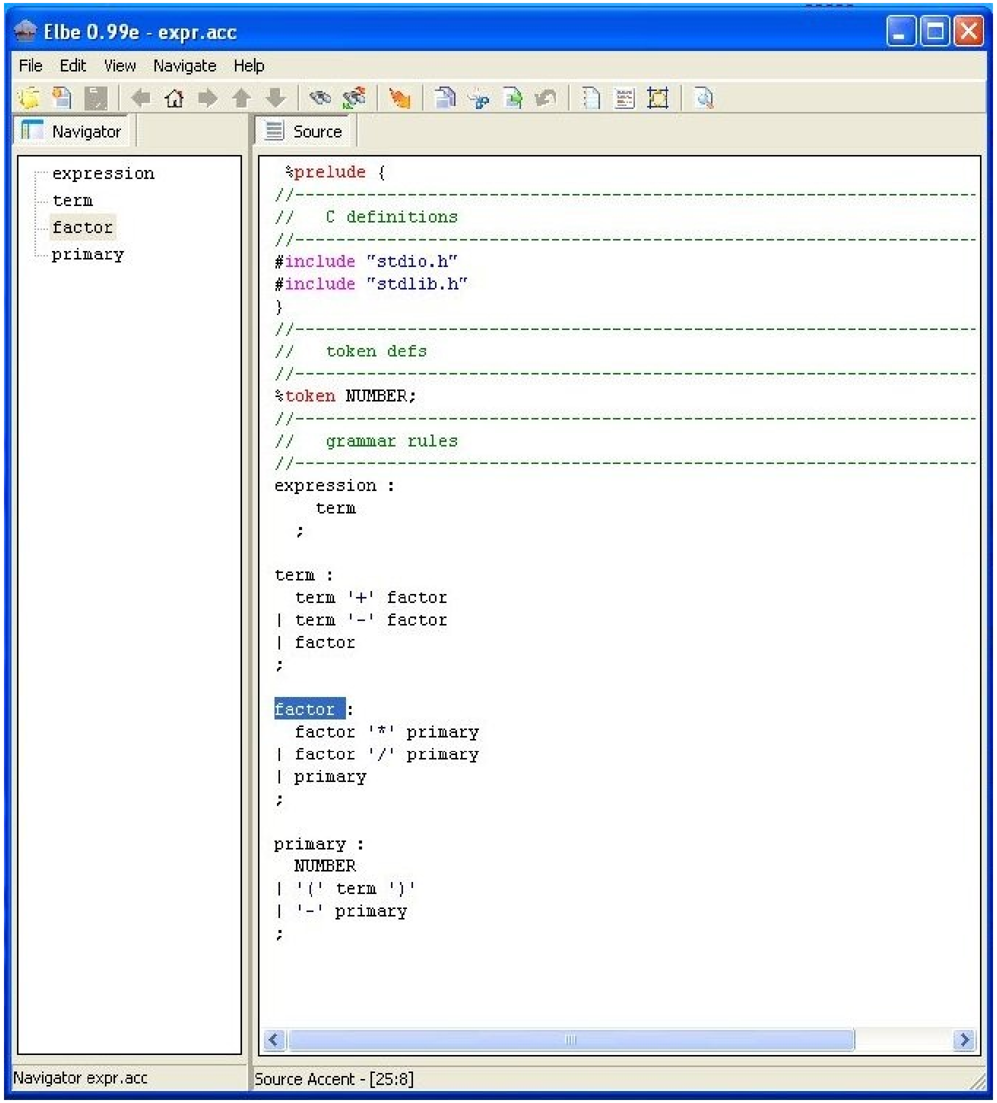
Elbe 1.0

Columbo 0.7

Webinterface 0.3

Columbo
Sherlock Holmes method

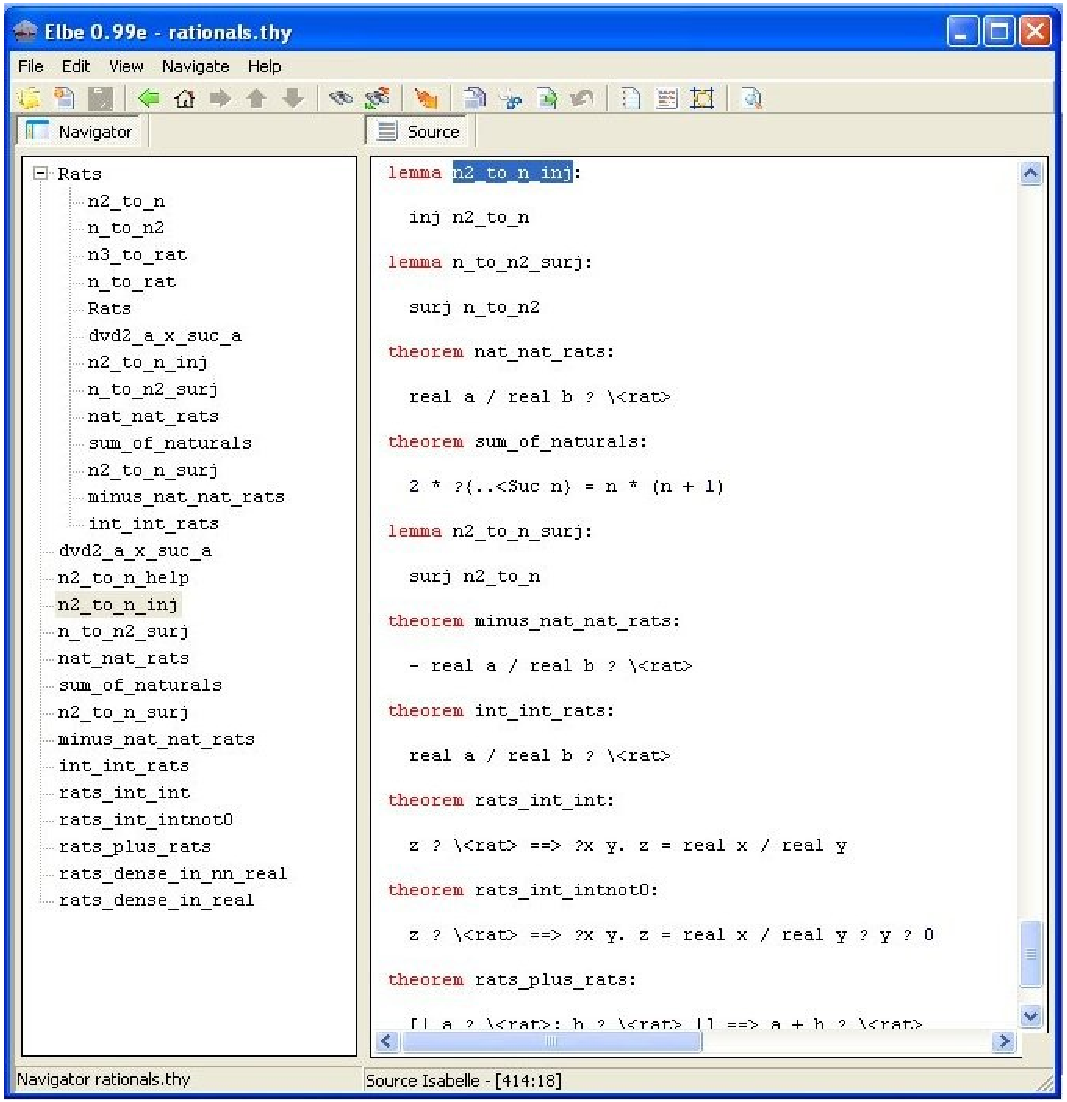
4.2.1 Example Accent file



```
 %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
;
```

Download under http://cococo.de

4.2.2 Example: Isabelle file



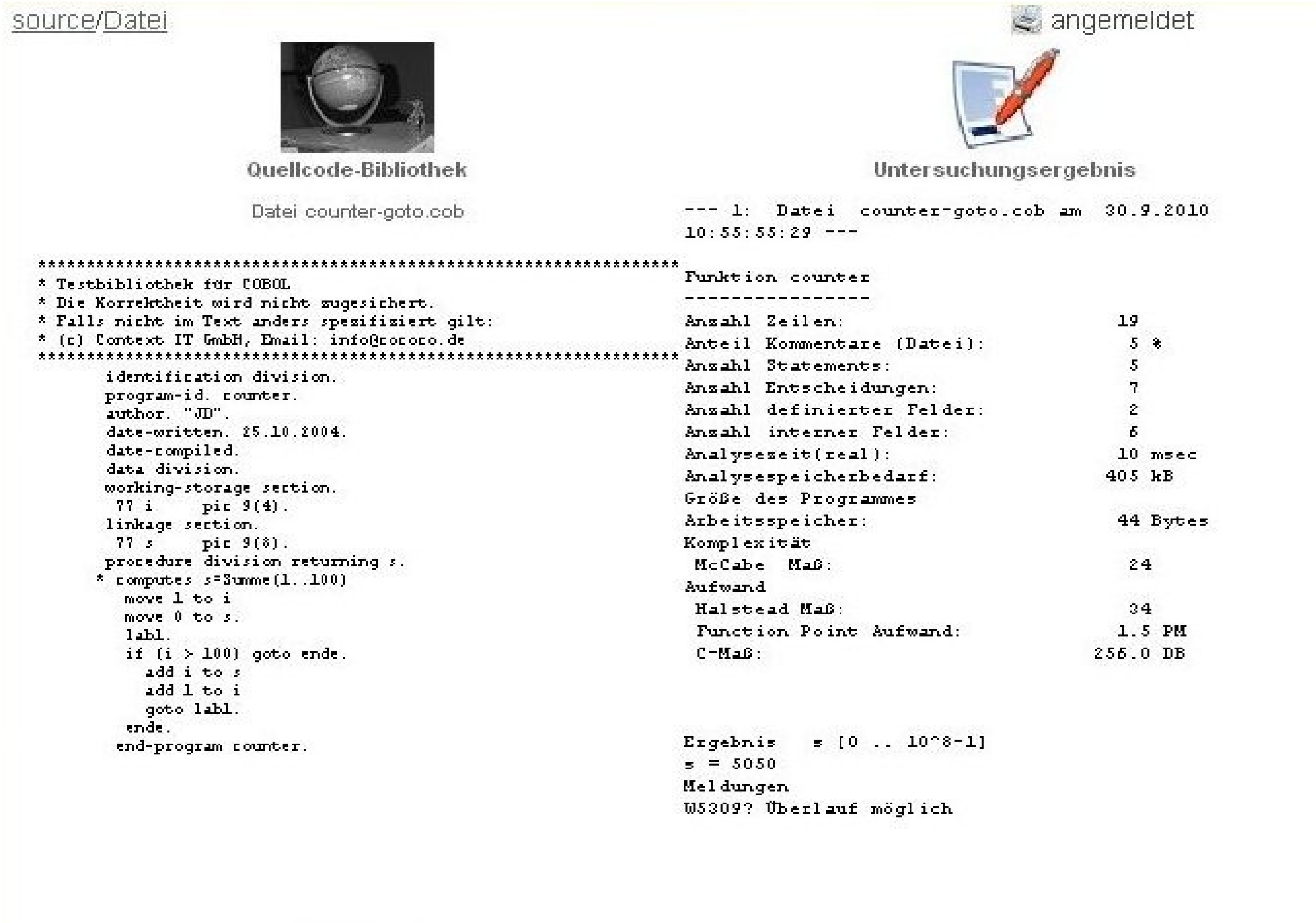
```
Rats
    n2_to_n
    n_to_n2
    n3_to_rat
    n_to_rat
    Rats
    dvd2_a_x_suc_a
    n2_to_n_inj
    n_to_n2_surj
    nat_nat_rats
    sum_of_naturals
    n2_to_n_surj
    minus_nat_nat_rats
    int_int_rats
  dvd2_a_x_suc_a
  n2_to_n_help
  n2_to_n_inj
  n_to_n2_surj
  nat_nat_rats
  sum_of_naturals
  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>; b ? \<rat> |] ==> a + b ? \<rat>
```

1 Elements
2 Structure
3 Process
4 Target

Download under http://cococo.de

4.2.3 Example: Web Interface

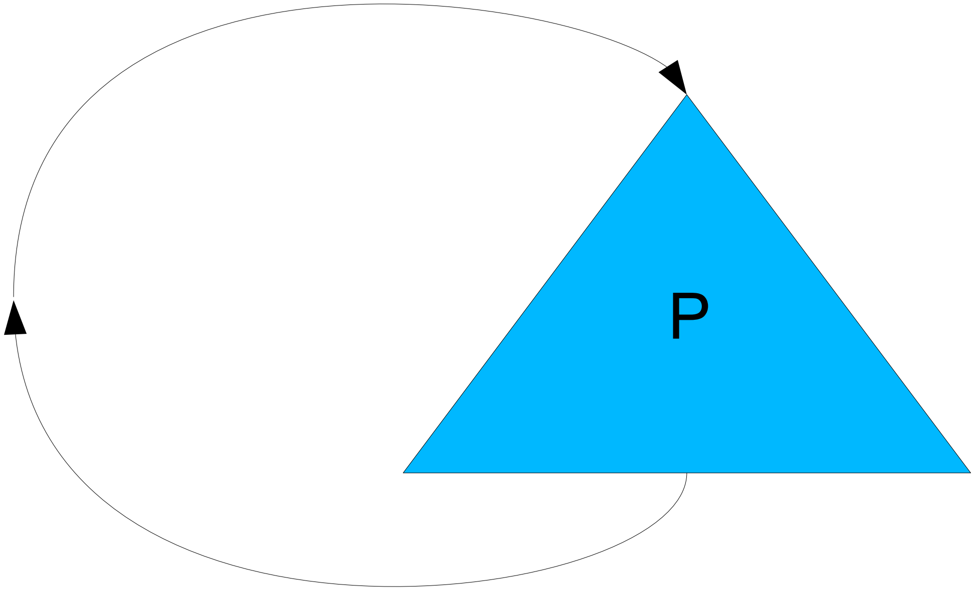


```
**********************************************************************
* Testbibliothek für COBOL
* Die Korrektheit wird nicht zugesichert.
* Falls nicht im Text anders spezifiziert gilt:
* (c) Context IT GmbH, Email: info@cococo.de
**********************************************************************
       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.
```

```
--- 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
Analysezeit(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 Elements
2 Structure
3 Process
4 Target

4.2 Goal („in the limit“)



the halting program P
Q(P) source
O(P) object

...towards an elegant proof system = Elbe

Thanks for watching!