Szoftver karbantartási lépések ellenőrzése

Majzik István
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék
http://www.inf.mit.bme.hu/

Tartalomjegyzék

Szoftver karbantartási terv

- Minőségirányítási rendszer fennhatósága alatt
- Rögzítendő eljárások: A karbantartás „életciklusa”
  - Hibajelentések, hibanaplók, módosítási igények elemzése
  - Módosítás specifikálása, tervezése és megvalósítása
  - Verifikáció, validáció és értékelés
  - Engedélyezés és jóváhagyás módja
- Fejlesztéssel azonos szintű szaktudás, eszközök, tervezés, megvalósítás, dokumentáció és irányítás szükséges
- Specifikus módszerek:
  - Adatrögzítés és -elemzés
  - Hatáselemzés
- Jellegzetes dokumentáció:
  - Szoftver változtatási feljegyzések
  - Szoftver karbantartási feljegyzések

Jellegzetes dokumentáció

- Szoftver-változtatási feljegyzések

  Változtatási tevékenységhez köthető
  - Módosítási igény
  - Módosítás specifikációja
  - Módosítás hatásának elemzése
  - Módosítás tervezése és megvalósítása
  - Módosítás verifikációja és validációja
- Szoftver-karbantartási feljegyzések

  Szoftverelemhez köthető
  - Hivatkozás a szoftver-változtatási feljegyzések elemeire
  - Változtatás következményeire vonatkozó információk
  - Regressziós tesztesetek, ismételt érvényesítés feladatai
  - Konfiguráció előzményei és változásai

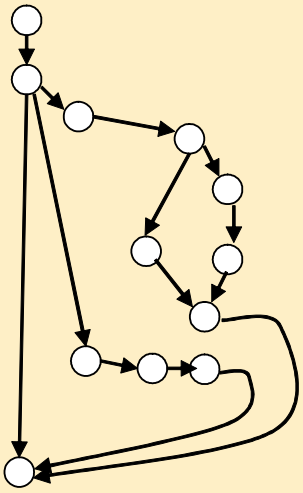
A szoftver karbantartáshoz kötődő technológiák

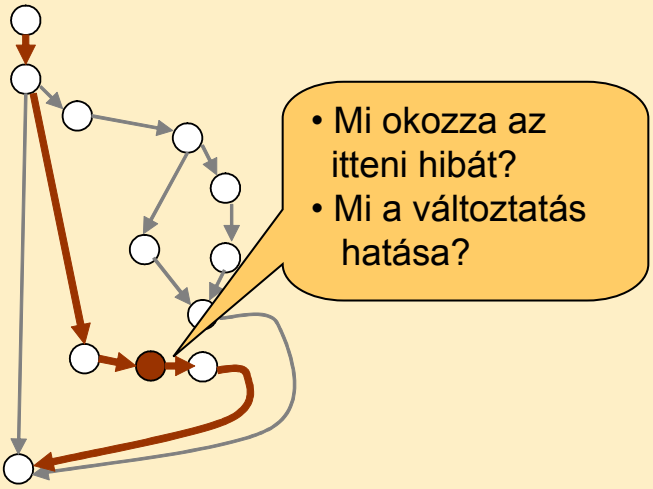
- Feladatok:
  - Hibakeresés (debuggolás)
  - Hibajavítás tervezése és megvalósítása
  - Hatáselemzés
  - Ellenőrzés (regressziós tesztelés)
- Megoldások, technológiák:
  - Program szeletelés
    - Hibakeresés segítése
    - Módosítások hatásainak felmérése
    - Teszt kiválasztás támogatása
  - Regressziós tesztelés

Tartalomjegyzék

Program szeletelés

- Debuggolás, ellenőrzés, tesztelés során a programnak csak egy részét (szeletét) szeretnénk figyelembe venni:
  - Mely sorok vannak hatással egy változó értékére?
  - Mit befolyásol, ha módosítunk egy utasítást?





A statikus szeletelés definíciója

- Statikus szeletelési kritérium: C = (V, I)
  - V a program változóinak egy halmaza
  - I a program egy kiválasztott utasítása
- Az M program C = (V, I) kritérium szerinti S statikus szelete:

  Egy olyan futtatható részprogramja M-nek, amelyre igaz:

  M-et és S-et futtatva tetszőleges program bemenetre

  az I utasításnál

  a V halmazbeli változók értékei megegyeznek
- Hátra haladó szeletelés: Kiválasztja az I-nél a V-beli változókat befolyásoló utasításokat

Szeletelési példa

```
     procedure SumEven
     int n, sum, j
1    sum = 0
2    j = 2
3    n=read()
4    while n > 0 do
5            sum = sum + j
6            j = j + 2
7            n = n - 1
     endwhile
8    write (sum)
```

Kritérium:
C=({j}, 6)

Befolyásoló utasítások:
- 2. utasítás (j értékadása)
- 4. utasítás (ciklus)
- 3. utasítás (4-esre hat)
- 7. utasítás (4-esre hat)

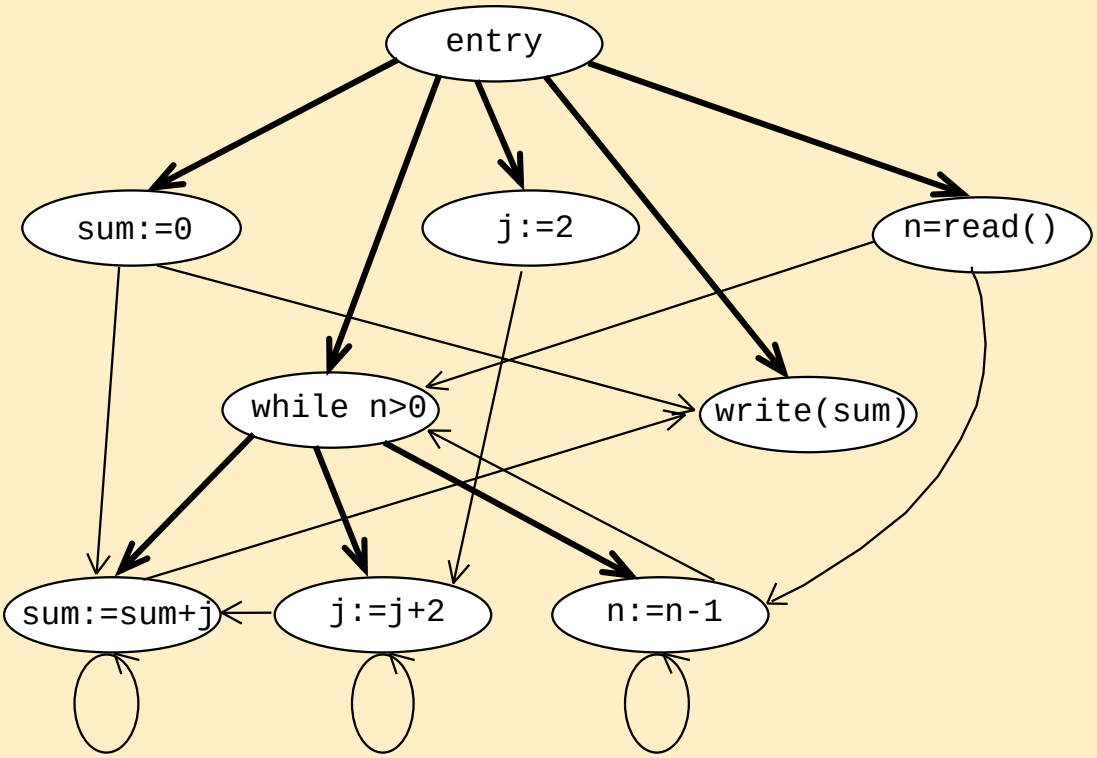
Szelet: {2, 3, 4, 6, 7}

C=({n},7) szerinti szelet:
{3, 4, 7}

Szeletelés alapja: Függőségek

- A program CFG-ben b utasítás vezérlésfüggő a-tól, ha:
  - b-be vezető út tartalmazza a-t is (elérhetőség),
  - a-ból indulva létezik elágazás, amelyen keresztül olyan út vezet, amely nem tartalmazza b-t (más irány)
- Egy b utasítás adatfüggő egy a utasítástól, ha:
  - az (a, b) pár egy du-párt alkot (definíció és felhasználás)
- Egy program függőségi gráfja (PDG, Program Dependence Graph) a következő:
  - egy egyedi kiindulási csomópontot tartalmaz (vezérlésfüggés),
  - a program minden egyes utasítása a PDG gráf egy csomópontja,
  - az a csomópontból vezet él b csomópontba, ha b vezérlésfüggő, vagy adatfüggő a-tól

A példaprogram függőségi gráfja



Vezérlésfüggés vastagított vonal, adatfüggés normál vonal

Statikus szelet meghatározása

Hátra haladó szelet (backward slice) képzése:

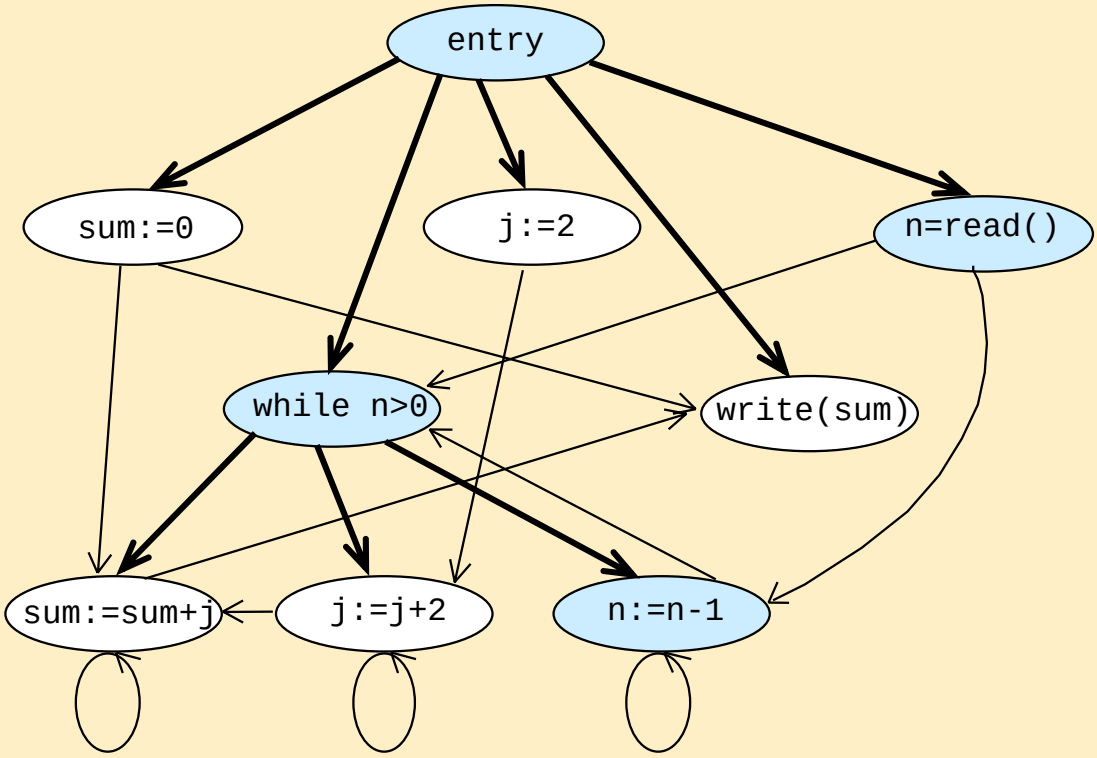
- PDG felépítése
- Indulás a szeletelési kritérium által megadott utasítástól visszafelé
- Szelet részei azok az utasítások, ahová az éleken visszafelé vezet út

Listás feldolgozás PDG-re (elérhetőségi probléma):

1. Listára felvenni a kritérium által adott utasítást
2. Kivenni egy lista-elemet és a szeletbe tenni
3. Az ebbe vezető élek végein lévő utasításokat, ha még nem voltak feldolgozva, felvenni a listára
4. Folytatás 2-ben, míg ki nem ürül a lista

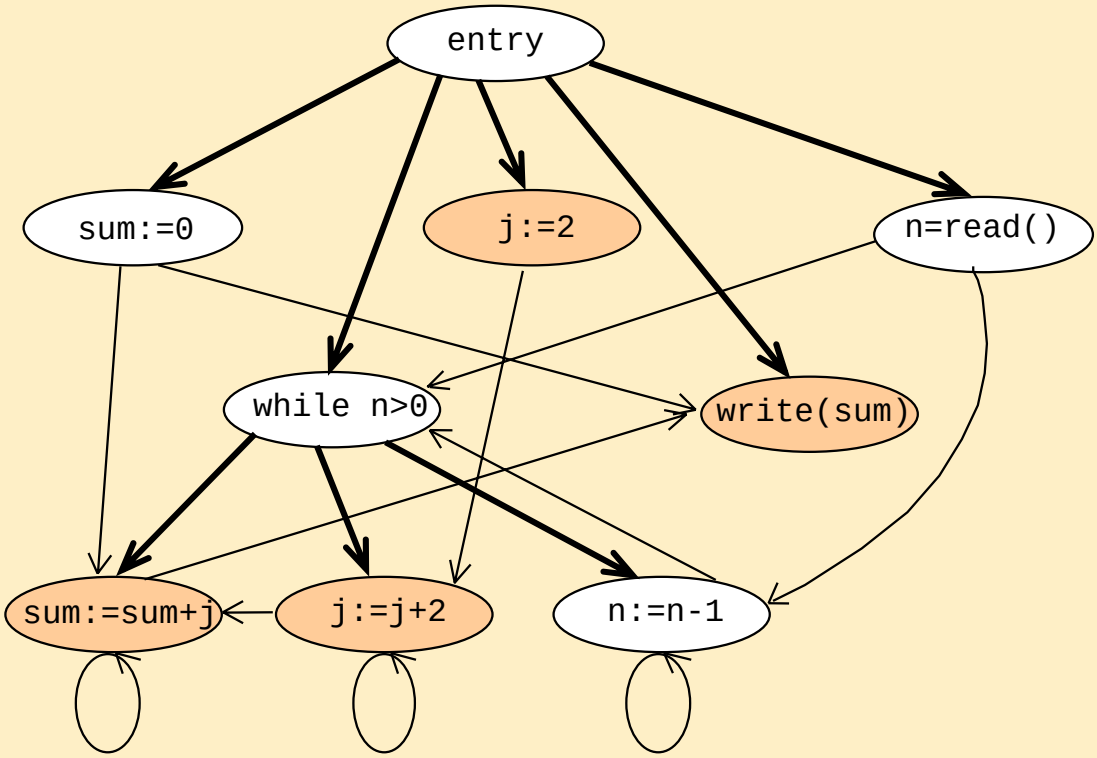
Hátra haladó szelet a példaprogramhoz

Hátra haladó szelet: Indulás a kritériumbeli utasítástól visszafelé
Példa: C=({n},7) kritérium szerinti statikus hátra haladó szelet:

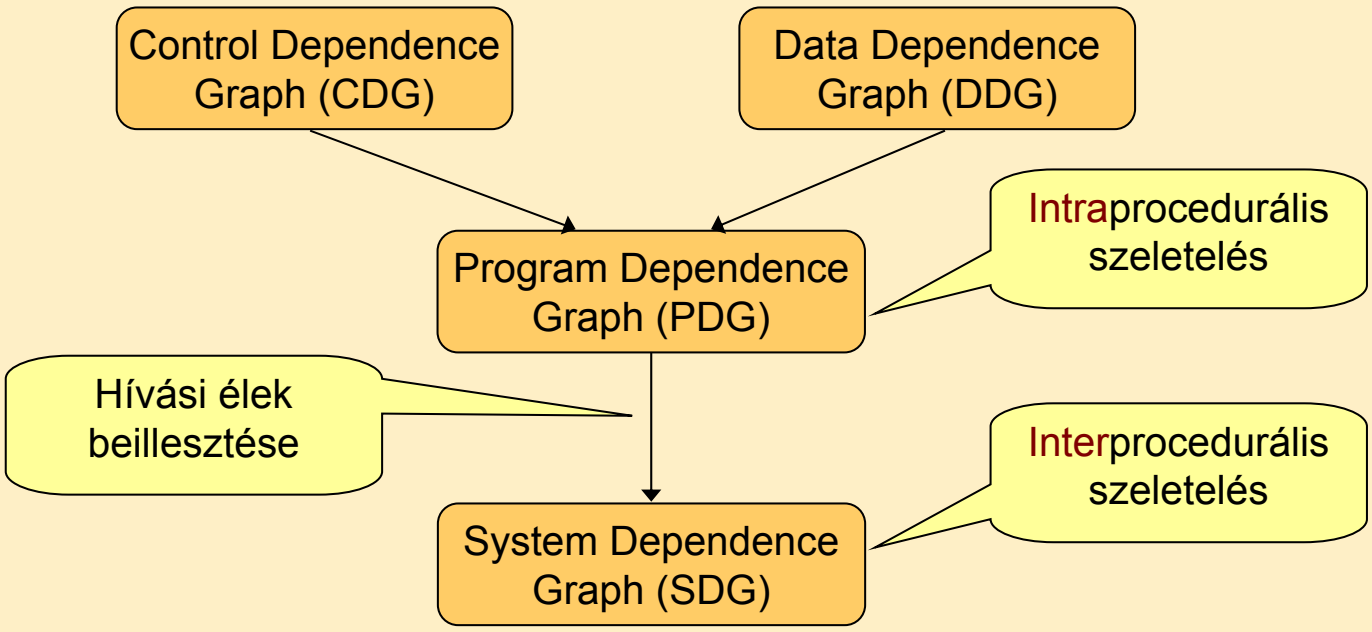


Előre haladó szelet a példaprogramhoz

Előre haladó szelet: Indulás a kritériumbeli utasítástól előrefelé
Példa: C=({j},2) kritérium szerinti statikus előre haladó szelet:

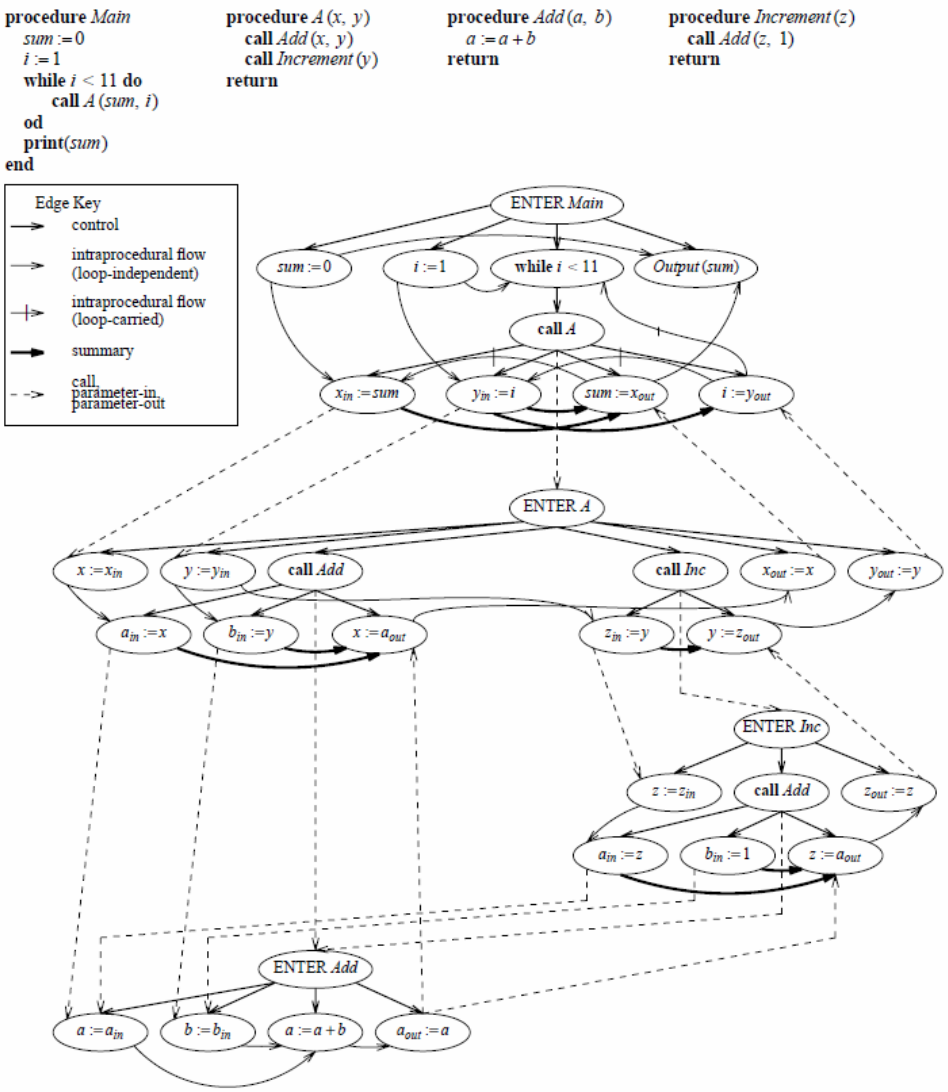


Struktúrák a szeleteléshez



A szeletek meghatározása elérhetőségi probléma

System Dependence Graph



Statikus szeletek használata

- Szeletelés kisebb programot eredményez
  - Könnyebb áttekintés, megértés hibakeresés során
  - Kisebb kódméret tesztelés során
- Jellegzetes kérdések, amire válaszolni lehet:
  - Hátra haladó szelet:
    - Mi lehet befolyással egy hibás eredményű utasításra?
  - Előre haladó szelet:
    - Mit befolyásol egy adott utasítás megváltoztatása?
    - Mit kell újratesztelni?
- Konkrét teszt esetre vonatkozó hibakeresés
  - Adott bemenettel történő programfutás volt!
  - Tovább csökkenthető a szelet mérete

Tartalomjegyzék

A dinamikus szeletelés definíciója

- Szeletelést egy adott bemenetre történő program futáson (számításon) végezzük el.
  - Ciklusok: Többször végrehajtott utasítások a futásban!
- Dinamikus szeletelési kritérium: C = (t, I^q, V)
  - t egy adott bemenet (teszt eset),
  - I^q egy kiválasztott utasítás (q-adik végrehajtása),
  - V a program I-beli változóinak egy halmaza.
- Az M program C=(t, I^q, V) szerinti S dinamikus szelete:

  Egy olyan futtatható részprogramja M-nek, amelyre igaz:
  M-et és S-et futtatva t program bemenetre
  az I utasítás q-adik végrehajtásakor
  V változóinak értékei megegyeznek

A példaprogram dinamikus szeletelése

```
   procedure SumEven
   int n, sum, j
1   sum = 0
2   j = 2
3   n=read()
4¹ while n > 0 do
5¹       sum = sum + j
6¹       j = j + 2
7¹       n = n - 1
   endwhile
8   write (sum)
```

Kritérium:
C=(n=1, 8^1, {sum})
Ciklus egyszer fut le (n=1).

Befolyásoló utasítások:
- 5^1: értékadás
- 3: beolvasás
- 1 és 2: értékadás

Dinamikus szelet:
{1, 2, 3, 5, 8}

A példaprogram dinamikus szeletelése

```
   procedure SumEven
   int n, sum, j
1   sum = 0
2   j = 2
3   n=read()
4⁰ while n > 0 do
5⁰       sum = sum + j
6⁰       j = j + 2
7⁰       n = n - 1
    endwhile
8   write (sum)
```

Kritérium:

C=(n=0, 8^1, {sum})

Ciklus egyszer sem fut le.

Befolyásoló utasítások:

- 3: beolvasás
- 1: értékadás

Dinamikus szelet:

{1, 3, 8}

Dinamikus szeletelés használata

- Programutak szerinti különbség:
  - Statikus szelet:
    Minden lehetséges programfutást (függőséget) figyelembe vesz (nincs meghatározott bemenet)
  - Dinamikus szelet:
    Meghatározott bemenetre (függőségre) szorítkozik, ami egy konkrét programfutást jelent,
    így a statikusnál általában kisebb méretű a szelet
- Tesztelés során kimutatott hiba keresése:
  A dinamikus szeleten dolgozunk
  - Konkrét teszt esetre történik a hibakeresés
  - Általában kisebb méretű mint a teljes kód

Szeletelési technikák áttekintése

- Szelet(elés) típusok:
  - Futtatható – nem futtatható
  - Statikus – dinamikus
  - Előre haladó – hátrafelé haladó
  - Interprocedurális – intraprocedurális
  - Amorf
    - A szelet nem az eredeti program utasításait tartalmazza, csak a hatása ugyanaz az adott helyen
- Az alkalmazandó típus a felhasználástól függ
  - Hibakeresés
  - Hatásanalízis, függőségi analízis
  - Program megértés
  - Tesztelés

Szeletelő eszközök

- WPS - The Wisconsin Program Slicing System
  http://www.cs.wisc.edu/wpis/html/
- CodeSurfer
  http://cayuga.grammatech.com/products/codesurfer
- Unravel
  http://www.nist.gov/itl/div897/sqg/unravel/unravel.html
- ClearMaker
  Y2K probléma
- ...