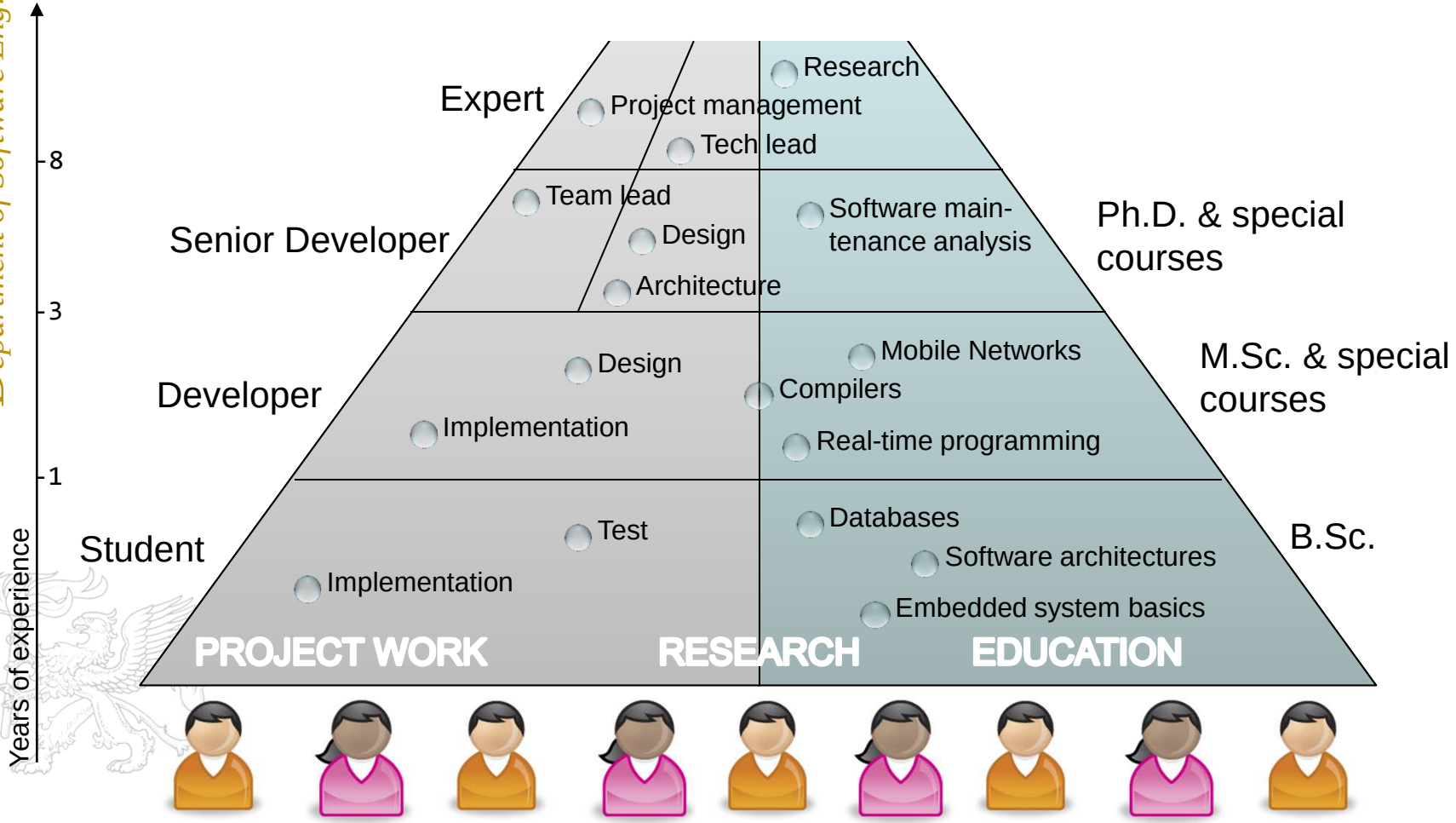
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Kutatás-fejlesztés a Szoftverfejlesztés Tanszéken

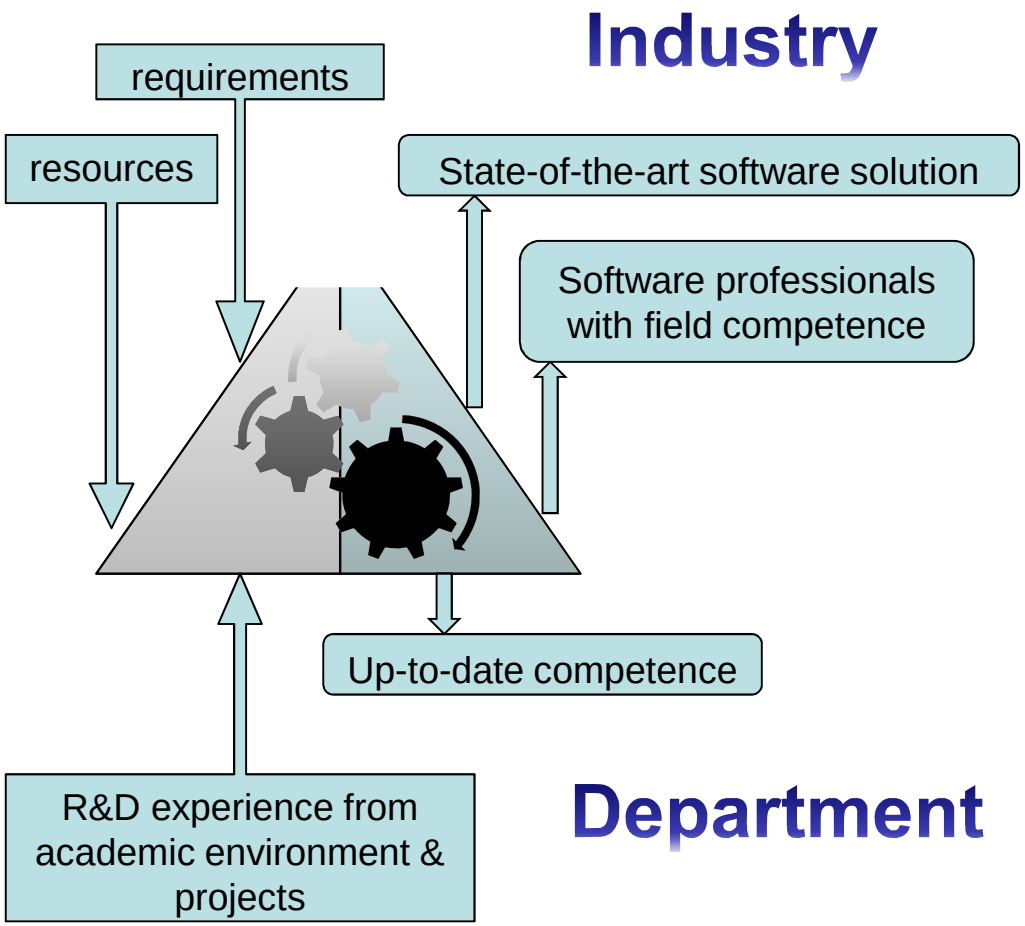
Gyimóthy Tibor

Mission: Project-based Education



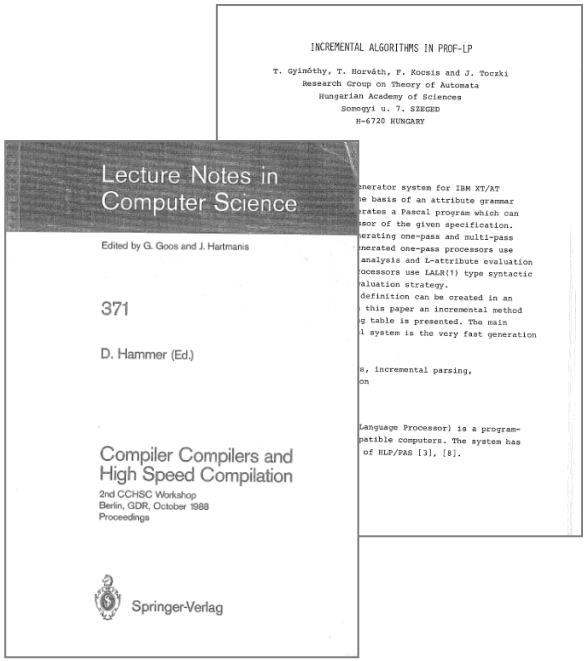
Creating Competence Groups

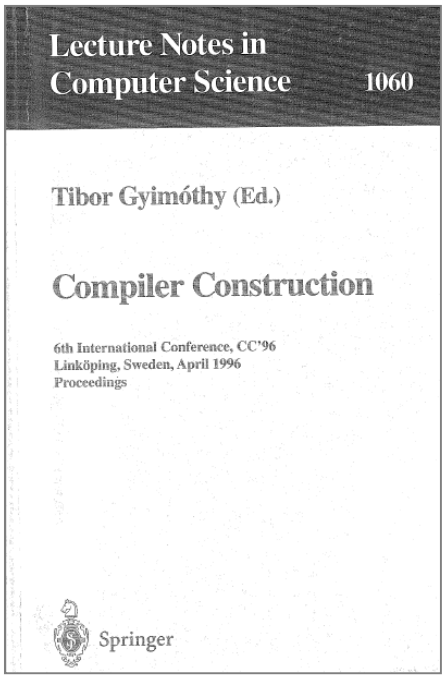
- Win-win situation
- Built around one research area
- Competence areas
  - Software Quality
  - Compilers & Embedded Systems
  - M2M Solutions



Compilers (and Related)

History





- PROF-LP: a compiler generator tool from 1982
- Cobol parser generated by PROF-LP used in Germany

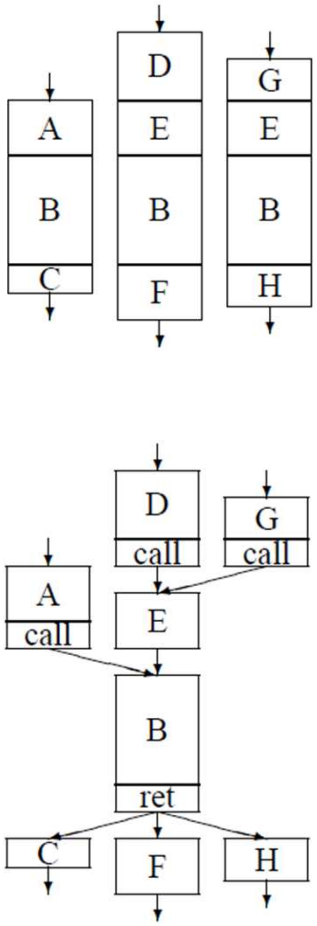
- Chaired Compiler Construction conference in 1996

Projects

- Columbus/CAN (1997-2001)
  - C++ parser written from scratch using ANTLR (LL parsing technology)
  - Supporting framework: internal representation, code analysis outputs, user interface, IDE integration
- Currently:
  - Research tools
  - Software maintanence

Projects (cont.)

- Post-link time optimization (2000-2003)
  - Targeted ARM binaries of production mobile devices
  - Utilized aggressive whole-program optimizations, including code factoring (procedural abstraction)
  - Achieved 5-11% size reduction
- Code factoring in GCC (2004-2005)
  - Adapted the binary code factoring optimizations to GCC
  - Official GCC branch opened



Projects (cont.)

- GCC-ARM improvements (2003-2004)
  - Several small patches to enhance the ARM port of GCC
- Symbian-GCC improvement (2004-2005)
  - Fixed and enhanced the official, old, forked compiler of Symbian 6.1
  - Produced 3 different compilers
  - Achieved 10% size reduction and 18% running time reduction

Related Projects

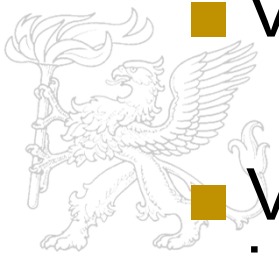
- XEEMU
  - Am Intel XScale / ARM power consumption simulator
- Flash file systems
  - For embedded Linux and NetBSD
  - JFFS2 with improved mount time
  - New designs: UbiFS, ChewieFS

Selected Publications

- Beszédes, Á., Gergely, T., Gyimóthy, T., Lóki, G. and Vidács, L. *Optimizing for Space : Measurements and Possibilities for Improvement.* In Proceedings of the 2003 GCC Developers' Summit (GCC 2003), pages 7-20. Ottawa, Canada, May 25-27, 2003.
- Beszédes, Á., Ferenc, R., Gyimóthy, T., Dolenc, A. and Karsisto, K. *Survey of Code-Size Reduction Methods.* In ACM Computing Surveys (CSUR), Volume 35, Issue 3 (September 2003), pages 223-267. Published by ACM Press
- Beszédes, Á., Ferenc, R., Gergely, T., Gyimóthy, T., Lóki, G. and Vidács, L. *CSiBE Benchmark: One Year Perspective and Plans.* In Proceedings of the 2004 GCC Developers' Summit (GCC 2004), pages 7-15. Ottawa, Canada, June 2-4, 2004.
- Lóki, G., Kiss, Á., Jász, J. and Beszédes, Á. *Code Factoring in GCC.* In Proceedings of the 2004 GCC Developers' Summit (GCC 2004), pages 79-84. Ottawa, Canada, June 2-4, 2004.
- Nagy, Cs., Lóki, G., Beszédes, Á. and Gyimóthy, T. *Code factoring in GCC on different intermediate languages.* In Proceedings of the 10th Symposium on Programming Languages and Software Tools (SPLST 2007), pages 81-95. Budapest, Hungary, June 14-16, 2007.
- Zoltán Herczeg, Ákos Kiss, Daniel Schmidt, Norbert Wehn, and Tibor Gyimóthy. XEEMU: An Improved XScale Power Simulator. In Integrated Circuit and System Design - Power and Timing Modeling, Optimization and Simulation - 17th International Workshop, PATMOS 2007, Gothenburg, Sweden, September 3-5, 2007, Proceedings, volume 4644 of Lecture Notes in Computer Science (LNCS), pages 300-309, 2007.
- Zoltán Herczeg, Daniel Schmidt, Ákos Kiss, Norbert Wehn, and Tibor Gyimóthy. Energy Simulation of Embedded XScale Systems with XEEMU. Journal of Embedded Computing, 3(3):209-219, August 2009.

Webkit: JavaScript végrehajtás

- Fókuszban
  - JIT (röpfordítás)
  - ARM platform
- Különböző JS motorok
  - JavaScriptCore (WebKit)
    - karbantartás
  - V8 (Blink)
    - gyorsítás
  - V4 (Qt QML small & simple expression interpreter)
    - fejlesztés



Webkit: Fuzzing

- Randomizált tesztelés
- Cél
  - stabilitás
  - biztonság (sokszor: böngésző összeomlás → memória probléma → biztonsági rés)
- Támogatott nyelvek
  - HTML, CSS, XML, SVG, MathML, JavaScript

Webkit:Grafika

- OpenGL ES alapú 2D motor
  - beágyazott eszközökön (is) hardveresen gyorsítve
- JPEG dekódolás párhuzamosítása
- HTML5 Canvas gyorsítása Chrome-ban

Webkit: Hálózat

- SPDY/HTTP 2.0 támogatás
- Network Process
  - párhuzamosítás
  - biztonság

Webkit: egyéb fejlesztések

- Teszteléshez használt böngésző fejlesztése
- Buildbot üzemeltetés és karbantartás
- LLVM fejlesztések

Software Quality

Software Quality

- Conference organisation
  - The 6th European Conference on Software Maintenance and Reengineering (CSMR2002)
  - The 21st IEEE International Conference on Software Maintenance (ICSM2005)
  - The 16th European Conference on Software Maintenance and Reengineering (CSMR2012)
  - The 8th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2011)
  - Various Scientific Committee memberships
- 5 best paper awards
- Most influental paper award (ICSM 2012)

Most Influental Paper Award

Software Quality

- AIM: Measure/Improve Software Quality
  - Reducing the number of bugs (post-release bugs are the most expensive ones)
  - Improving maintainability
    - Reducing costs of development and testing
  - Continuous quality monitoring and assessment
- How?
  - Code based software quality
  - Software testing, test optimization and process improvement
  - With tools – analyzers, monitoring framework
  - Methodology – Columbus Methodology
  - Expertise – industrial and academic knowledge and experience
  - By Services – Quality Assessment
  - Education
  - Innovation

Software Quality Projects

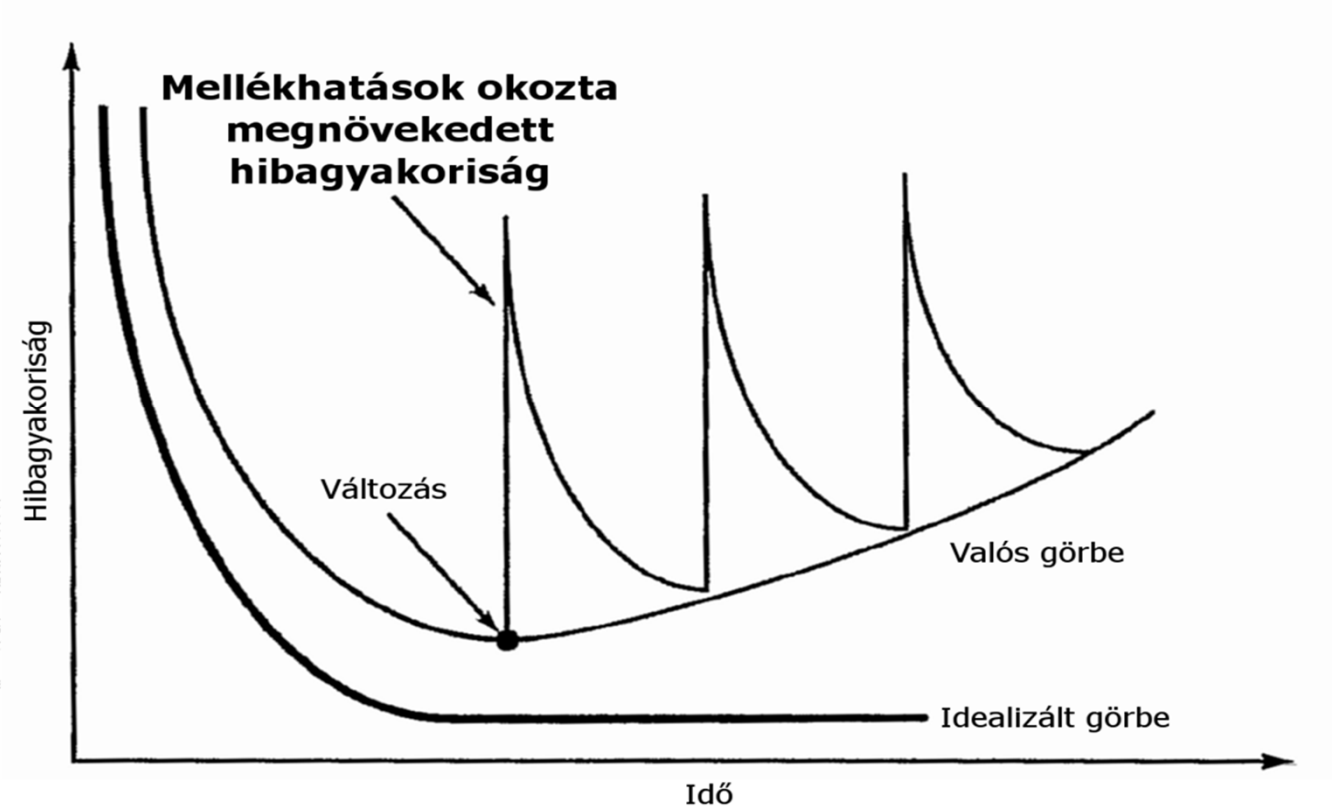
- Tools and services
  - Code quality monitoring
  - Change impact analysis
  - Architecture reconstruction
  - Software testing and optimization
- Analysis of large systems containing several million lines of source code
- Reference projects with banks, insurance companies and mobile phone manufacturers

Szoftverkarbantartást támogató módszerek

- A nagyméretű informatikai rendszerek minősége a hosszabb üzemeltetés során a változtatások hatására rohamosan romlik.
- Egyetlen hiteles „specifikáció” a forráskód.
- A szoftver minőségének biztosítására a tesztelés önmagában nem elegendő (a rendszer „korosodásával” a tesztelés egyre költségesebb lesz).

Motiváció és célkitűzések

- Szoftver minőségének romlása



Motiváció és célkitűzések (folyt.)

- Cél: a szoftverkarbantartást támogató módszerek és szoftver megoldások kidolgozása (szoftverkarbantartás a teljes szoftverfejlesztési életciklus 60-80%-át is elérheti).
  - *Programszeletelés*: a szoftver elemek közötti függőségek meghatározása.
  - *Columbus keretrendszer:* karbantartási módszerek integrálása.

Szoftverkarbantartás

- Legnehezebb probléma nagy rendszerek egyes részei közötti kapcsolatok feltárása a forráskód alapján.
- „Visszatervező” (reverse engineering) eszközök használata:
  - A rendszer komponenseinek azonosítása
  - Kapcsolatok feltárása
  - A rendszer ábrázolása magasabb absztrakciós szinten

Szoftverkarbantartás (folyt.)

- Jelenleg nincs általánosan elfogadott szoftverkarbantartási módszertan és azt támogató eszközrendszer.
- Modellek és azokat támogató analizátorok kellenek a szükséges adatok előállítására.
- A szoftverminőség folyamatos monitorozását támogató módszerekre van szükség.

Programszeletelés

- Egy programszelet tartalmazza a program azon utasításait, amelyek hatással lehetnek az ún. *szeletelési kritériummal* <p,V> meghatározott változók értékeire.
- Weiser eredeti 1979-es definíciója szerint a szelet egy olyan végrehajtható utasítás-sorozatot jelent, amelynek viselkedése az adott kritériumra megegyezik az eredeti programéval.

Programszeletelés (folyt.)

- „*Hátrafelé irányuló” (backward) szelet*: tartalmazza azokat az utasításokat, amiktől a szeletelési kritérium értéke függhet.
- „*Előrefelé irányuló” (forward) szelet*: azokat az utasításokat tartalmazza, amik függhetnek a szeletelési kritériumtól.
- *Statikus szelet*: a program összes lehetséges végrehajtását figyelembe vesszük a függőségek meghatározásakor.
- *Dinamikus szelet*: az adott utasítás egy végrehajtására számítjuk a függőségét.

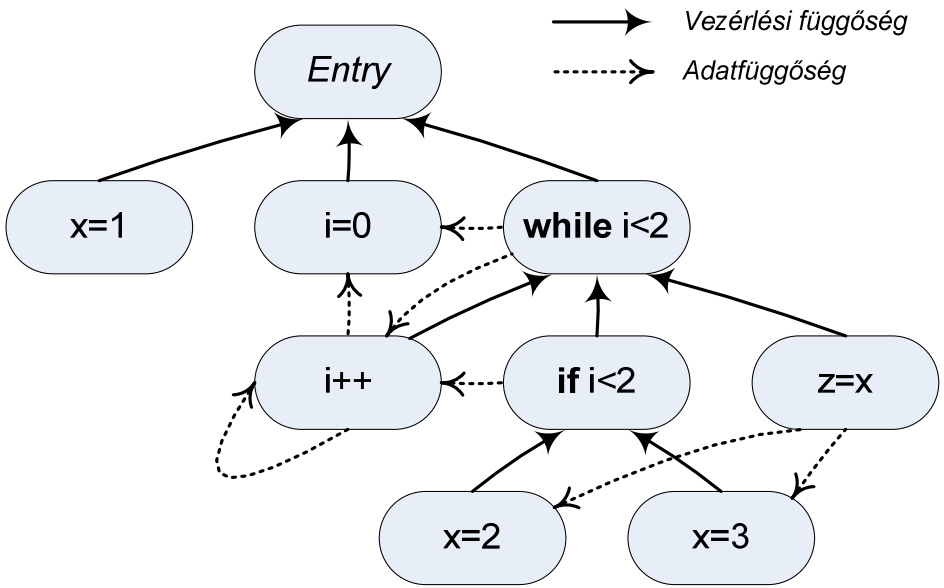
Programszeletelés (folyt.)

- Programszeletek számítása függőségi gráfok alapján (vezérlési-és adat élek).
- Pointerek, tömbök, nem-strukturált vezérlési szerkezetek és az eljárás-hívások okozzák a legtöbb problémát.
- Dinamikus esetben a függőségi gráf mérete kritikus.

Programszeletelés (folyt.)

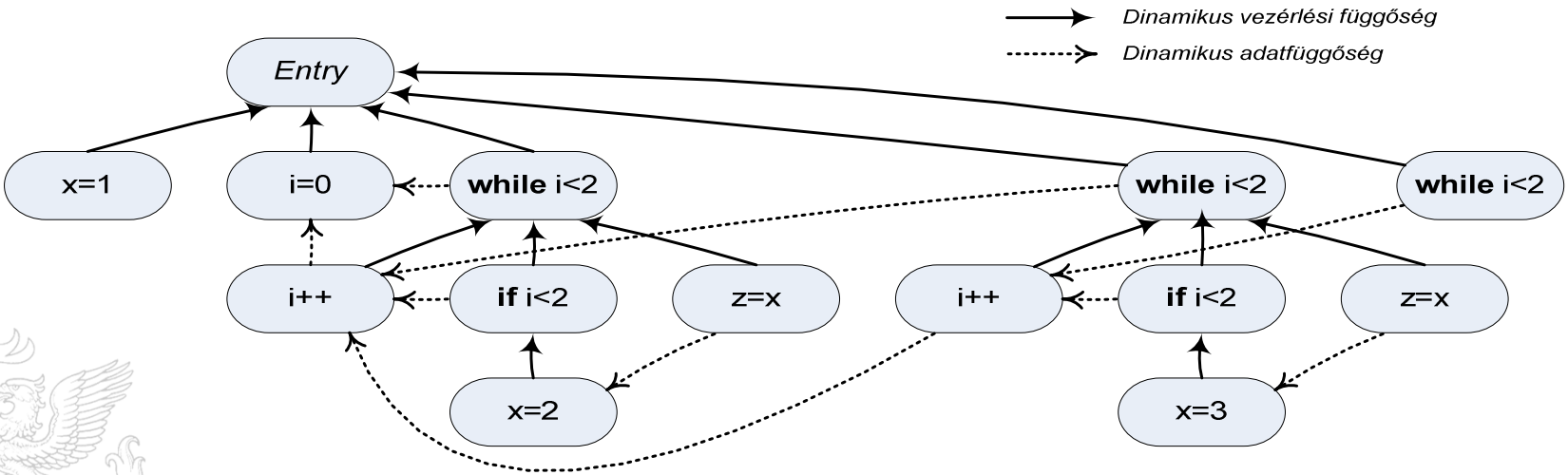
- Programfüggőségi Gráf (statikus szelet)

```
1. x=1;
2. i=0;
3. while (i<2) {
4.    i++;
5.    if (i<2)
6.       x=2;
      else
7.       x=3;
8.    z=x;
   }
```



Programszeletelés (folyt.)

- Dinamikus Függőségi Gráf (dinamikus szelet)



Dinamikus programszeletelés (folyt.)

- Eredmények összefoglalása
  - Egy új globális dinamikus szeletelő algoritmus kidolgozása.
  - Az algoritmus alkalmazása releváns szeletek számítására.
  - C és Java programok dinamikus szeletelése.
  - Uniós programszeletelés.

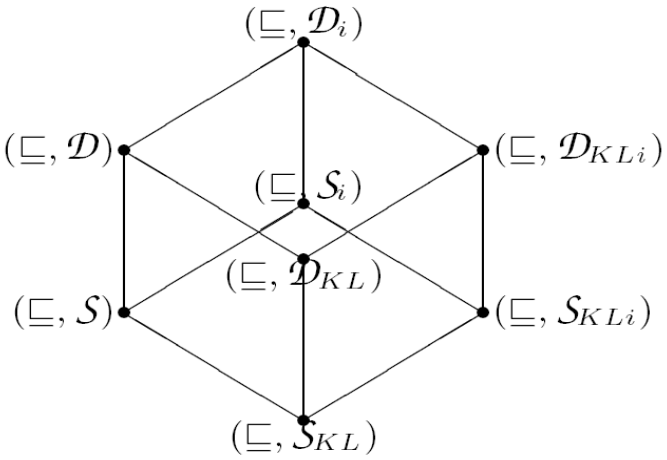
Dinamikus programszeletelés (folyt.)

- Eredmények hatása
  - Új lendületet adott a dinamikus szeleteléssel kapcsolatos munkáknak.
  - Konferencia legjobb cikke díj az European Conference on Software Maintenance and Reengineering konferencián 2001-ben.
  - 2004-ben az ICSE konferencia programszeletelés szekciójában mindhárom cikk hivatkozik a módszerünkre.
  - Több egyetemen továbbfejlesztették az algoritmusunkat.

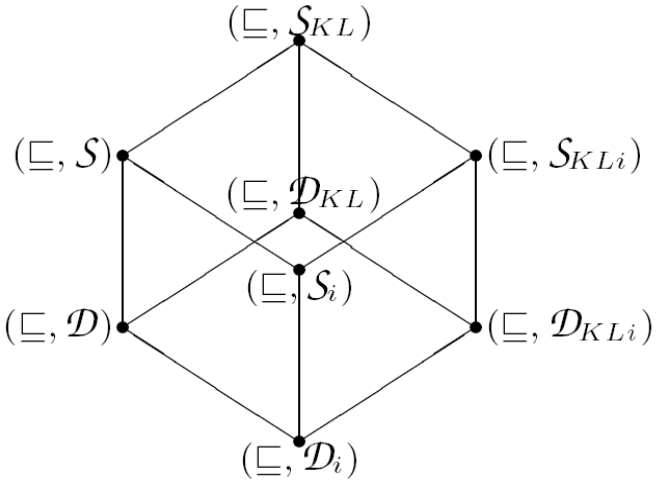
Programszeletelési módszerek tartalmazási kapcsolata

- Kimutattuk, hogy a Korel és Laski-féle dinamikus szeletelés és a statikus szeletelés között nincs egyértelmű alárendeltségi viszony.

Programszeletelési módszerek tartalmazási kapcsolata (folyt.)



Alárendeltségi reláció



Rang reláció

Programszeletelési módszerek tartalmazási kapcsolata (folyt.)

- Eredmények hatása
  - A projekciós keretrendszer új területet nyitott meg a programszeleteléssel kapcsolatos formális kutatásokhoz.
  - Megoldásra vár számos létező szeletelési módszer (pl. releváns, amorf, hybrid, feltételes) keretrendszerbe való illesztése.
  - Konferencia legjobb cikke díj a Source Code Analysis and Manipulation konferencián 2005-ben.

Forráskód minőségvizsgálata

- Statikus analízis alapján problémás kódrészletek felderítése.
- Objektumorientált metrikák alkalmazása nyílt-forrású rendszerek szoftverminőségének vizsgálatára.
- Eredmények megmutatták, hogy bizonyos metrikák (pl az ún. csatolási metrika-CBO) jól azonosítják a legtöbb hibát tartalmazó osztályokat.

Forráskód minőségvizsgálata (folyt.)

- Eredmények hatása
  - A módszer alapján elkészült a Mozilla szoftver folyamatos minőség monitorozó rendszere.
  - Az eredményeket 2005-ben publikáltuk az *IEEE Transactions on Software Engineering* folyóiratban. A publikációra eddig több mint 430 hivatkozás történt (Google Scholar).

Függőség elemzés

- Egyik legalapvetőbb technika programon belüli függőségek feltérképezése és használata
- Kód entitások között (pl. utasítások, eljárások, osztályok) függőségek elemzése
- Függőségi gráf
- Hatáshalmaz = gráfbejárás kezdeti hatáshalmazból
- Pontos vs. biztonságos
- Költséges vs. hatékony
- Egyszerű vs. szofisztikált
- Dinamikus: futási információk felhasználásával
- ***Nagy probléma: nagy programok, nagy komplexitású módszerek***

Static Execute After (SEA módszer)

- Static Execute After / Before

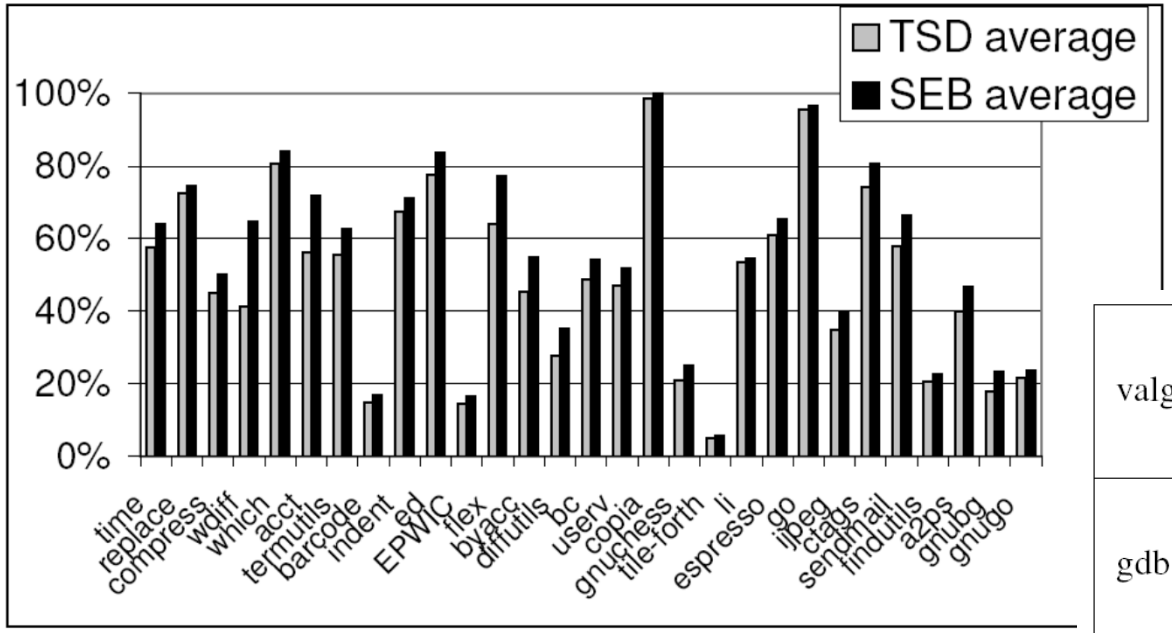
> ***B* modul *A*-tól függ**
>
> **⇔**
>
> ***B* modul *A* után hajtódik végre a program valamely lehetséges végrehajtása során**

- Három lehetőség:
  1. ***A*** hívja ***B***-t
  2. ***A*** visszatér ***B***-be
  3. ***C*** hívja ***A***-t majd utána ***B***-t

SEA (folyt.)

- SEA reláció számítható a program vezérlési folyam információja alapján (interprocedurális CFG)
  - Bővebb, mint a részletes PDG gráf, amely az adatfüggőségeket pontosan tartalmazza (vagyis konzervatívabb ezért *biztonságos*)
  - Egyszerűbb számítani, tömörebb méretben tárolható (összefüggő komponensek elhagyhatók)
  - Eljárás szinten praktikus az alkalmazása

Pontosság és hatékonyság



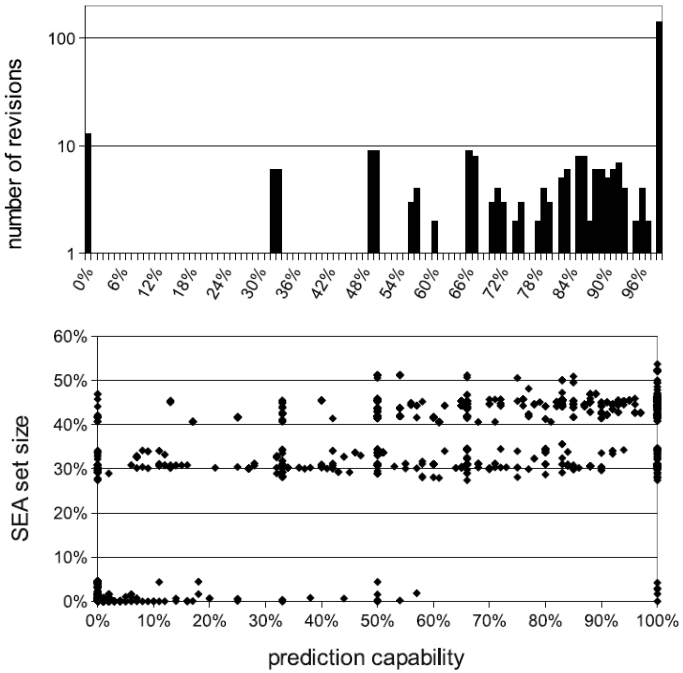
valgrind	SDG	vertices	1 920 150
		edges	6 947 024
	ICCFG	vertices	154 509
		edges	179 626
gdb	SDG	vertices	10 086 409
		edges	48 876 108
	ICCFG	vertices	160 340
		edges	185 611
gcc	SDG	vertices	18 775 143
		edges	81 492 908
	ICCFG	vertices	467 185
		edges	584 972
mozilla	SDG	vertices	N/A
		edges	N/A
	ICCFG	vertices	1 587 499
		edges	1 723 611

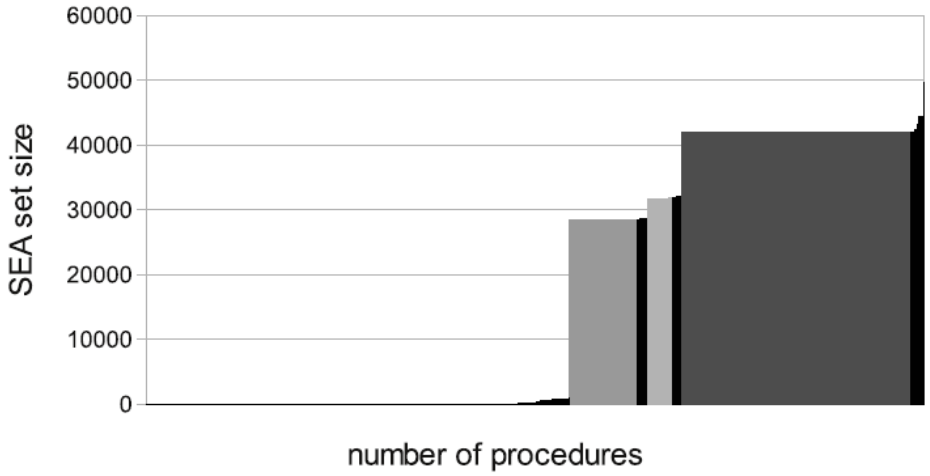
Kísérlet a WebKit rendszeren

- WebKit böngészőmotor
  - Google, Apple, Nokia, …
- 2 MLOC C++ kód, 90 ezer eljárás
- Regressziós teszt környezet
  - „Layout” tesztek
  - 25 ezer teszteset
  - 113 ezer revízió, átlagosan ¼ óránként új
  - Hiba-adatbázis: több mint 90 ezer bejegyzés
- 33 ezer revízió pár megvizsgálása
- 240 pár további vizsgálathoz

Eredmények

- Elemzési idő: statikus elemzés 2 óra, 1-10 perc SEA számítás 1 revízióra
- Előrejelzés teljessége: **83,9%**
- Függőségi halmazok sokszor nagyok (átlag: program méret **19%**-a)
- Legfőbb ok:
  - *Függőségi klaszterek* kialakulása





Alkalmazás tesztelésben

- Teszt szelekció
  - Eljárás szintű kód lefedettség alapján (77% össz.)
  - Csak a megváltozott (és új) kódon átmenő tesztek újrafuttatása
  - Új hibák 75%-át elkapja a szelekció is
  - Szelektált teszteset lista nagy lehet (40-50% átlagosan)
- Teszt priorizálás
  - Móho stratégiak tesztesetek különböző jellemzői alapján, pl. általános, speciális
- Hatásanalízis: nem csak a megváltozott, hanem annak hatásait is ellenőrizzük
- Új módszer: priorizálás az alapján, hogy a klasztereket milyen mértékben fedik a tesztesetek
  - Többi stratégiához képest 55-60%-os javulás

Best Paper Award (SCAM 2013)

Mission: Project-based Education