OBJEKTUM VEZÉRELT SZOFTVEREK ANALÍZISE

Ferenc Rudolf, ferenc@cc.u-szeged.hu
Beszédes Árpád, beszedes@cc.u-szeged.hu
Szegedi Tudományegyetem

Abstract

A korszerű objektum orientált szoftvertermékek a piaci igények kielégítése végett egyre nagyobbak és komplexebbek lesznek. Ráadásul a szoros határidők miatt elmaradó dokumentációk és tervek hiánya még inkább megnehezíti a rendszerek megértését. A forráskód megértésére a legnagyobb szükség a szoftver életciklus karbantartási szakaszában van, amikor különböző módosításokat kell eszközölni rajta, mint például a hibajavítások vagy új funkcionalitások hozzáadása. Ez a problémakör hozta létre a szoftverfejlesztés tudományon belül a létező rendszerek modellezését kutató (reverse engineering) irányzatot. A célja az, hogy egy bizonyos szintű szoftverspecifikációból egy magasabb szintű leírást hozzon létre, például esetünkben a programkódból a tervezési dokumentációt. Manapság a modellezésre leginkább a szabványos UML jelölésrendszert alkalmazzák. Természetes tehát az igény, hogy a visszanyert tervezési dokumentáció is UML diagramokra épüljön. Ezen túl egyéb, különböző szempontok szerinti vizualizációk is rendkívül fel tudják gyorsítani a forráskód megértését (pl. hívási gráfok). A legújabb kutatási témák közé tartozik a tervezési minták felismerése a forráskódból, melynek segítségével még magasabb szintről lehet megközelíteni szoftverrendszert. Egy másik nagy kutatási terület a szoftverminőség mérése, melyben a különböző metrikáknak nagy szerepük van. A Szegedi Tudományegyetem Informatika Tanszékcsoportjának dolgozói a helsinki Nokia Research Center-rel és a FrontEndART Kft.-vel együttműködve a fenti problémakört megcélozva egy Columbus nevű reverse engineering eszközt fejlesztettek ki, mely segítséget nyújt a forráskód megértésében és dokumentálásában. Az eszköz analizáló motorját felhasználva elkészült egy CPPAudit nevű program is, ami ellenőrizni tud különböző kódolási konvenciók betartását, illetve figyelmeztet veszélyes kód konstrukcióra.

Modern object oriented software systems, to meet the continuously growing expectations of the market, are getting more and more complex. Additionally, the lack of documentation and plans, which is missing because of tight deadlines, makes it even more difficult to understand the systems. The biggest need in comprehending the source code is in the maintenance phase of the software lifecycle, when different modifications need to be done, like bug-fixing and adding new functionality. This originated the reverse engineering research within the science of software engineering. The aim is to produce a higher level description from a given specification, for instance the design documentation from the source code in our case. Nowadays, standard UML notation is used in most cases for designs. So it is an obvious need to present the recovered design documentation with UML diagrams. Additional visualizations based on different aspects can also remarkably improve source code comprehension (e.g. call graphs). Among the newest research activities is the recognition of design patterns from source code, with which the software system can be approached from an even higher level. Another large research area is the measuring of software quality, where different metrics have a big role. The staff at the University of Szeged Informatics Department in cooperation with the Nokia Research Center in Helsinki and FrontEndART Ltd. developed a reverse engineering tool called Columbus, which helps in source code comprehension and documentation. Using its analyzer engine another tool called CPPAudit was made as well, which checks different coding conventions and gives warnings on dangerous code constructs.

1. Bevezetés

Az egyik legnagyobb probléma a nagy szoftverrendszerek fejlesztésénél és karbantartásnál a forráskód gyorsan növekvő mérete és komplexitása. Ennek eredményeképpen nagy szükség van a rendszer különböző részeinek és ezek összefüggéseinek megértésére [1, 2]. A nagy mennyiségű örökölt kód és a fejlesztésben részt vevő nagyszámú kódoló is szükségessé teszi a reverse engineering eszközök használatát [16]. A fordított irányú tervezés egy szoftverrendszer elemzésének folyamata, melynek célja, hogy: (a) beazonosítsa a rendszer komponenseit és ezek kölcsönhatásait és (b) egy magasabb szintű leírást hozzon létre egy más formában" [4]. A cikkben bemutatjuk a *Columbus* nevű eszközt [7], ami képes nagy, C++ programozási nyelven [12] írt szoftverrendszereket elemezni. Az eredményt egy szabványos formában prezentálja, melynek a neve *Columbus Séma a C++-hoz* [6]. Ez a séma leírja, hogy a különböző nyelvi elemek és kapcsolataik hogyan legyenek ábrázolva.

Egy valódi szoftverkarbantartás sikeres lebonyolításához egy megfelelően összeválogatott eszközkészletre van szükség (pl. elemzők, kódgenerátorok, metrikaszámítók, dokumentáló eszközök). Ahhoz, hogy ezen eszközök kommunikálni tudjanak egymással, szükség van egy közös sémára és egy elemzőre, ami a sémának megfelelő adatokat előállítja. Megfogalmazunk néhány követelményt, melyeknek meg kell felelni, amikor sémát, elemzőt és az eszközök integrációját elősegítő keretrendszert tervezünk.

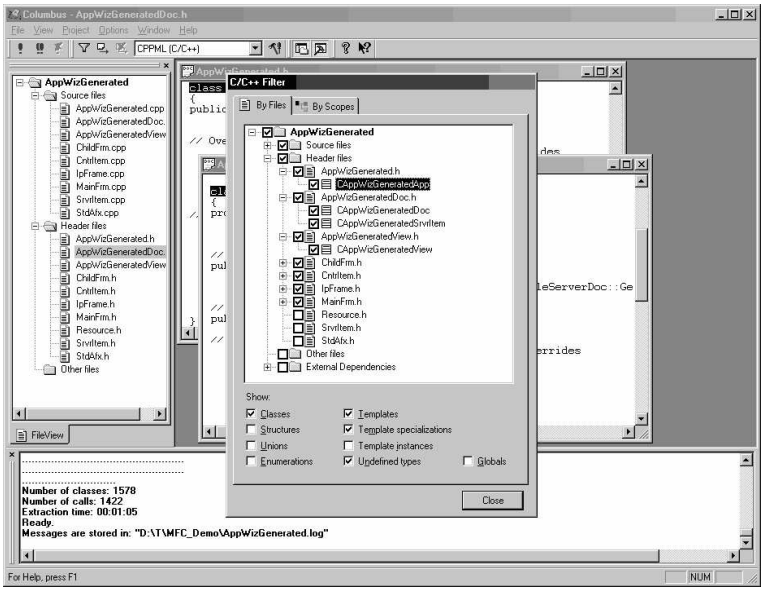
Követelmények a keretrendszerrel kapcsolatban: Mivel a valós projektek általában több fájlból állnak és azok több könyvtárba vannak rendezve, nagyon fontos a *projektkezelés*. Ahhoz, hogy ténylegesen össze tudjon fogni több eszközt, fontos a *bővíthetőség*. Mivel a kinyert információ mennyisége nagy rendszereknél óriási, ezért szükség van valamilyen *szűrés*re. Ezen kívül nagyon fontos még a *vizualizáció*, hogy a kinyert információt minél könnyebb legyen átlátni.

Követelmények a C++ elemzővel kapcsolatban: A legfontosabb követelmény a *helyesség* és a *teljesség*, hogy megbízható információkat kapjunk. Mivel a C++ nyelvnek sok *dialektus*a van, fontos ezeknek a kezelése. Ide tartozik a *hibatűrés* igénye is, vagyis ha ismeretlen kulcsszóba vagy szintaktikus hibába fut az elemző, akkor ne álljon le, hanem lépjen túl a hibán, és gyűjtsön össze lehetőleg minél több információt. Fontos még a *sebesség* is és az *előfeldolgozás* képessége. Ahhoz, hogy be lehessen integrálni az elemzőt a szokásos fordítási folyamatokba (makefile), szükség van *parancssori végrehajtás*ra.

Követelmények a C++ sémával kapcsolatban: Az első és legfontosabb elvárás a *részletesség*, vagyis hogy minden nyelvi elemet kényelmesen ábrázolni lehessen benne. Fontos követelmény a *modularitás/bővíthetőség* is, hogy a logikailag összetartozó nyelvi elemek külön csomagokban legyenek, és ezek szükség esetén lecserélhetőek legyenek például egy másik *dialektus*ra.

Ebben a cikkben bemutatjuk a Columbus keretrendszert, elemzőt és sémát a C++ nyelvhez, amik kielégítik a legtöbb fenti követelményt. Az eszköz ingyenesen használható tudományos és oktatási célokra és letölthető a *FrontEndART* honlapjáról [10].

2. A Columbus keretrendszer



1. ábra. A Columbus felhasználói interfésze

A *Columbus* (lásd az 1. ábrát) egy reverse engineering keretrendszer, melyet a Szegedi Tudományegyetem Informatika Tanszékcsoportjának dolgozói készítettek együttműködve a helsinki Nokia Research Center-rel és a FrontEndART Kft.-vel. Az eszköz képes nagy C/C++ projektek elemzésére és a Columbus Sémának megfelelő formátumú adatok kigyűjtésére.

A rendszer kifejlesztésének fő motivációja az volt, hogy létrehozzunk egy általános keretrendszert, amivel sokféle reverse engineering feladatot el lehet végezni egy általános felhasználói interfészen keresztül. Ennélfogva a Columbus egy olyan keretrendszer, ami támogatja a projektkezelést, adatkinyerést, adat ábrázolást, adattárolást és szűrést. Mindezen alapvető feladatokat a rendszer különálló *moduljai* (plug-in-jei) végzik el. A rendszerhez sok előre elkészített modul tartozik, de tetszőlegesen bővíthető külső modulokkal is.

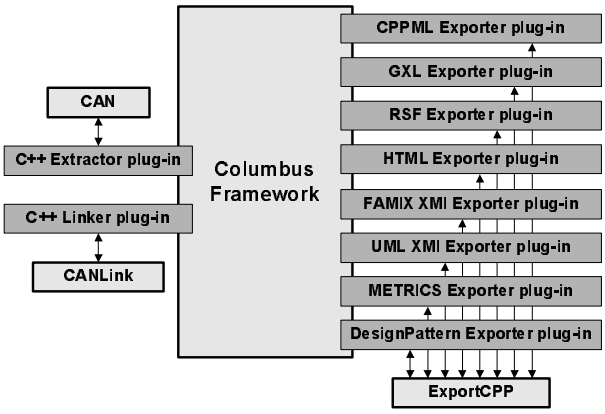
Az eszköz funkcióinak többségét parancssorból is el lehet érni (a CAN, CANLink, és ExportCPP nevű programok által, lásd a 2. ábrát), így beépíthető a szokásos programfordítási folyamatokba (pl. makefile-okba). Ezáltal naprakész dokumentációt kaphatunk a rendszerünkről.

2.1 A Columbus rendszer áttekintése

A Columbus rendszer működését három különböző modul típus valósítja meg (lásd a 2. ábrát):

– *Extraktáló modul*: feladata, hogy leelemezzen egy adott input fájlt és készítsen belőle egy kimenő fájlt, ami tartalmazza a kinyert információt.
– *Linkelő modul*: feladata, hogy összefésülje az extraktáló modul által készített fájlokat és felépítse a projekt teljes reprezentációját. Ez a modul felelős a kinyert információ szűréséért is.
– *Exportáló modul*: feladata, hogy feldolgozza és elmentse a linkelő modul által felépített és megszűrt belső reprezentációt valamilyen kimenő formátumba. A jelenleg

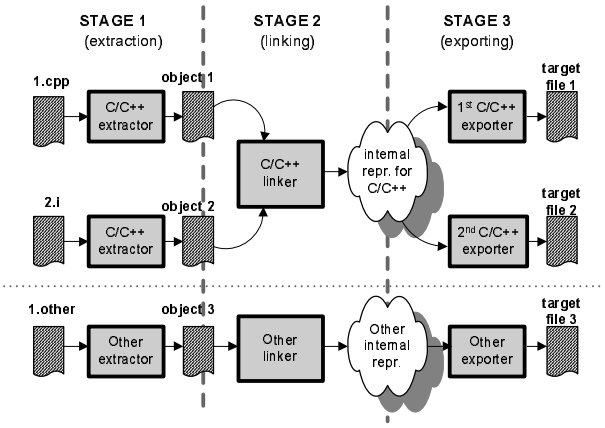
megvalósított formátumok: négy XML formátum (CPPML, GXL, UML XMI és Famix XMI), RSF, HTML, tervezési minták és metrikák (CSV).



2. ábra. A keretrendszer felépítése

Az alaprendszerben jelenlévő modulok mellé a felhasználó szabadon írhat saját modulokat is a Columbus API-t használva, melyekkel bővítheti a keretrendszer tudását.

2.2 Az adatkinyerés folyamata



3. ábra. Az adatkinyerés folyamata

Az adatkinyerés folyamata a *Columbus projekten* alapul. Egy projekt tárolja a bemenő fájlokat (és a beállításaikat: előfordított fejléc, előfeldolgozás, kimenő könyvtárak, stb.) melyek egy fa-nézetben vannak megjelenítve. Egy projekt egy valós szoftverrendszert ábrázol és tartalmazhat különböző nyelvű forráskódokat is.

A teljes kinyerési folyamat a 3. ábrán van ábrázolva. Az eljárás nagyon hasonló a hagyományos fordítóprogramoknál megszokottakhoz. Az *első fázis* az *adatkinyerés*. A Columbus fogja az input fájlokat, és egyenként átadja őket a megfelelő extraktáló modulnak, ami leelemzi és létrehozza a megfelelő kimenő fájlokat. A *második fázis*ban a linkelő modul automatikusan meghívódik, hogy *összefésülje* az első fázisban létrehozott fájlokat. Ebben a fázisban lehetőség van szűrni is az információt. A *harmadik fázis*ban, miután kiválasztottuk a megfelelő formátumot, az információ feldolgozása és *exportálása* kerül végrehajtásra.

Mindhárom fázis paraméterezhető különböző modul specifikus opciókkal. Egy fontos előnye még a Columbus rendszernek, hogy a fenti lépéseket inkrementálisan is el tudja végezni, vagyis ha egy adott lépés eredménye már rendelkezésre áll, akkor azt nem futtatja le újból.

2.3 Szűrés

Az analizált forráskód óriási mennyiségű információt eredményezhet, amit nehéz használható formában prezentálni a felhasználónak (a felhasználót általában egyszerre csak egy része érdekli a teljes rendszernek). A Columbus-ban jelen levő különböző szűrési mechanizmusok segíthetnek ennek a problémának a megoldásában.

Három különböző módszer áll rendelkezésre:

– *C++ elemkategóriák szerinti szűrés* (pl. osztályok, sablonok, felsorolás típusok, stb.). Ezzel az opcióval azok az elemek, amik nem taroznak bele a kiválasztott kategóriákba, ki lesznek szűrve.
– *Bemenő fájlok szerinti szűrés*. Azok a C++ elemek, amik a nem kijelölt bemenő fájlokból származnak ki lesznek szűrve. Ezzel a módszerrel minden olyan elemet, ami nem része szorosan a projektnek (pl. ami rendszerkönyvtárakból származik) könnyedén el lehet távolítani. A megmaradó C++ elemek egyenként ki/bekapcsolhatóak a projekt fanézetben.
– *Hatáskör szerinti szűrés*. A különböző C++ elemek, mint pl. az osztályok és névterek ki/bekapcsolhatók egy fanézetben, ami a rendszer hatáskör szerinti struktúráját ábrázolja.

2.4 Exportálás

A kinyert információt különböző formátumokba lehet kiexportálni. Ezek között vannak általánosak (CPPML, GXL, UML XMI, HTML, Metrikák, tervezési minták) és eszközfüggők (pl. Famix XMI, RSF). Az exportáló modulok száma folyamatosa növekszik. Az aktuálisan rendelkezésre állóak a következők:

CPPML. Ez az exportáló modul egy XML dokumentumot készít (a neve *CPPML* – C++ Markup Language) aminek a struktúrája a Columbus Sémát követi, vagyis lényegében a memóriában levő belső reprezentáció tükörképe.

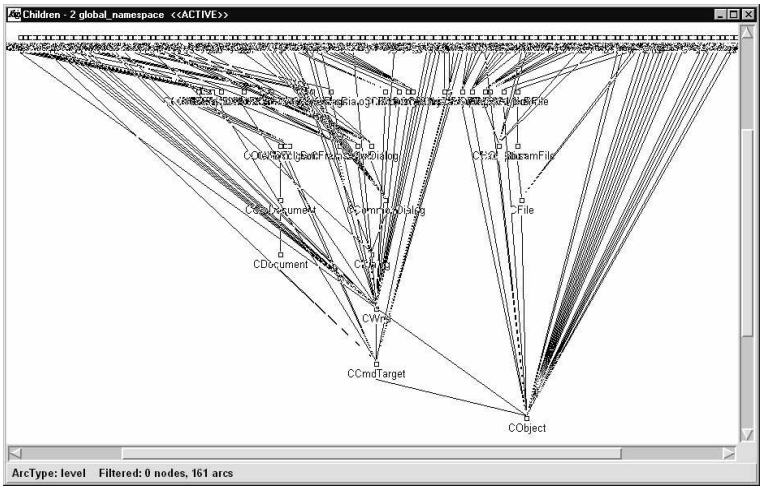
GXL. Ez az exportáló modul egy GXL dokumentumot készít. A *GXL* (Graph eXchange Language) [11] egy XML alapú gráf leíró formátum. Mivel a Columbus Séma alapvetően egy gráfot definiál, ebben a formában kényelmesen lehet ábrázolni. Ezen kívül lehetőség van még a függvényhívási gráfot is ebben a formában elkészíteni.

UML XMI. Ezzel a modullal szabványos UML XMI (Unified Modeling Language XML Metadata Interchange) dokumentumok készíthetők. Ezek a dokumentumok ezután betölthetők tetszőleges UML tervező eszközökbe, amik támogatják az XMI-t (pl. Rational Rose, Together) így megkapjuk az elemzett rendszer UML-ben ábrázolt terveit.

Tervezési Minták (Design Patterns). Ez a modul képes felismerni a Gamma-féle tervezési minták [9] nagy részét az elemzett szoftverrendszerben [3] (a minták tetszőlegesen módosíthatók és bővíthetők). Ezzel lehetőség nyílik tovább finomítani a visszanyert

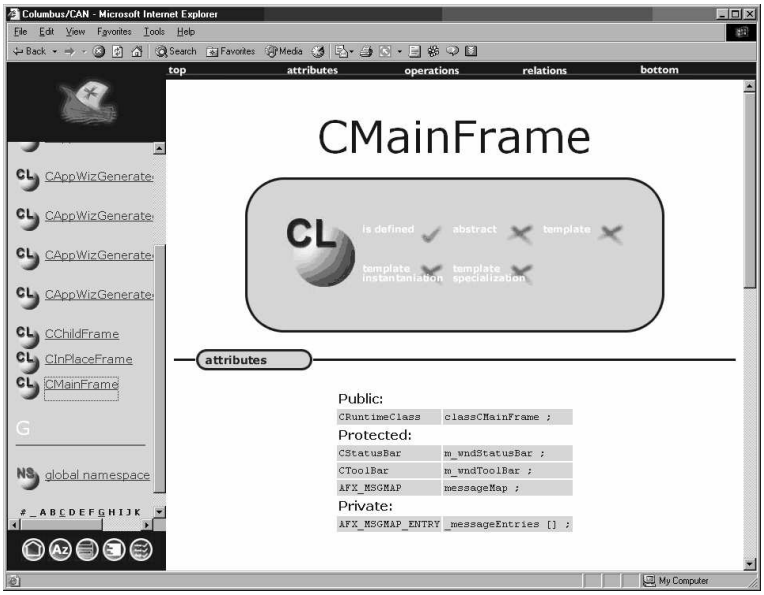
dokumentációt. Hasznos lehet még szoftverfejlesztés közben is, mivel le lehet vele ellenőrizni, hogy jól lett-e implementálva az adott tervezési minta.

RSF – rigi. Háromféle gráfot lehet készíteni a *rigi* eszközhöz [15]: (1) Columbus Séma gráf, (2) függvényhívási gráf és (3) UML osztálydiagram. A 4. ábrán látható egy Columbus által kinyert öröklődési hierarchia vizualizálása rigi-ben.



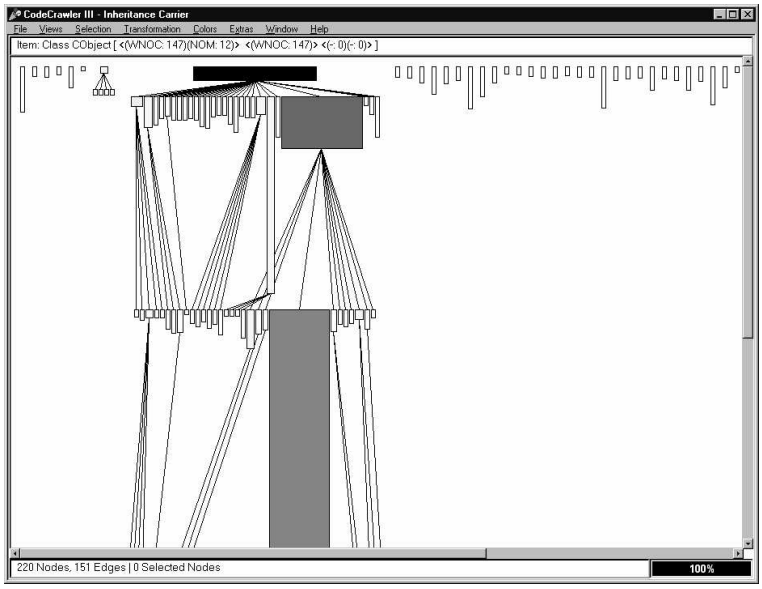
4. ábra. Vizualizáció rigi-ben

HTML dokumentáció. Ezzel a modullal hipertext dokumentáció készíthető HTML formátumban. A generált dokumentáció böngészhető, felhasználóbarát formában prezentálja az elemzett projektet. Minden fontos információ megtalálható az osztályokról strukturált módon. A dokumentációban háromféleképpen lehet gyorsan mozogni: (1) osztálynevek szerint, (2) hatáskör szerint és (3) öröklődés szerint. Az 5. ábrán látható egy Columbus által kinyert dokumentáció böngészőben megjelenítve.



5. ábra. A Vizualizáció böngészőben

Famix XMI – CodeCrawler. Ezzel az exportáló modullal *Famix* [5] XMI formátumú fájl készíthető, amit a *CodeCrawler* eszközzel lehet tovább feldolgozni (pl. megjeleníteni, metrikákat számítani). A 6. ábrán látható egy Columbus által kinyert öröklődési hierarchia vizualizálása CodeCrawler-ben.



6. ábra. Vizualizáció CodeCrawler-ben

Metrikák – CSV. Ez az exportáló modul 88 különböző metrikát számít ki az elemzett szoftverrendszerről, amit utána CSV (Comma Separated Value) formában ment el. Ez a fájl szinte minden ismert táblázatkezelőbe betölthető (pl. Microsoft Excel) és feldolgozható.

3. A Columbus Séma

A reverse engineering eszközök között létfontosságú a sikeres adatcsere. Ehhez azonban szükség van egy közös formátumra, amit alkalmazni lehet különböző eszközökben, mint pl. elemzőkben és metrikát számító programokban. Lennie kell egy szabványos sémának, ami leírja az adatok formátumát [8]. Ebben a cikkben ajánlunk egy sémát a C++ nyelvhez, amit Columbus Sémának nevezünk [6].

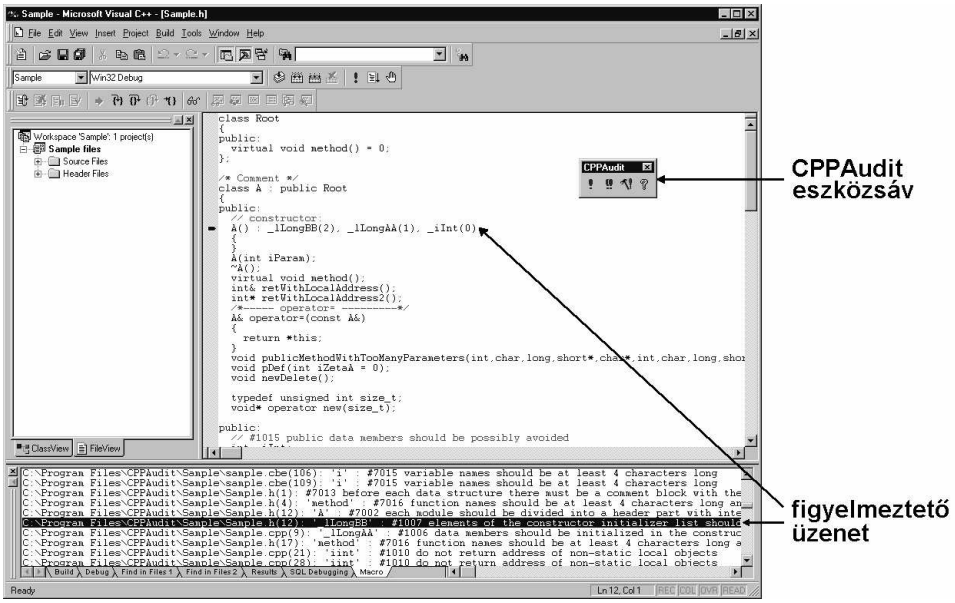
A Columbus Séma alacsony szinten (AST – Absztrakt Szintaxis Fa) ábrázolja a C++ nyelvet, de tartalmaz magasabb szintű elemeket is (pl. a típusok szemantikáját). Maga a séma UML osztálydiagramokkal van leírva, ezáltal könnyű implementálni és a példányait könnyű fizikailag reprezentálni (pl. GXL-lel). A séma moduláris, ami további rugalmasságot biztosít a jövőbeni bővítésekhez és módosításokhoz.

4. A CPPAudit eszköz

A Columbus analizáló motorját felhasználva elkészült egy *CPPAudit* nevű program is, ami ellenőrizni tud különböző kódolási konvenciók betartását, illetve figyelmeztet veszélyes kód konstrukcióra.

Az eszköz beépül a Visual Studio fejlesztőkörnyezetbe (lásd a 7. ábrát), ahonnan rendkívül kényelmesen használható. Egy egyszerű eszközsávon keresztül lehet indítani hasonlóan a

szokásos programfordításhoz. A futtatás eredményei különböző figyelmeztető üzenetek lehetnek, amik az alsó ablakban jelennek meg. Ezekre az üzenetekre kettőt kattintva a szerkesztőben megjelenik a kérdéses programsor.



7. ábra. CPPAudit ellenőrzés közben

Jelenleg 72 különböző szabályt ellenőriz, amik segítenek a hibák megelőzésében és az egységes kódolási stílus betartásában. Ilyen szabályok például:

– 'new' and 'delete' should correspond to each other
  (a 'new' és 'delete' utasítások feleljenek meg egymásnak)
– do not return address of non-static local objects
  (ne adja vissza nem statikus lokális objektum címét)
– public data members should be avoided
  (ne használjon publikus adattagokat)
– data members should be initialized
  (az adattagokat inicializálni kell)
– data members should be initialized in the constructor initializer list
  (az adattagokat a konstruktor inicializációs listában kell inicializálni)
– operators '++' and '--' should not be used more than once in one instruction
  (a '++' and '--'operátorokat ne használja többször egy utasításban)
– stb.

Az eszközt parancssorból is lehet futtatni, így beépíthető a szokásos programfordítási folyamatokba (pl. makefile-okba). Ezáltal akár rá is lehet kényszeríteni a programozókat a szabályok betartására.

5. Kísérletek

Ebben a fejezetben bemutatunk néhány kísérletet a Columbus analizáló képességeivel kapcsolatban. Három különböző, valós világból vett projektet elemeztünk:

– IBM Jikes fordítóprogram [13]. Ezzel a projekttel figyeltük a rendszer képességeit összetett osztályhierarchiák kezelésére.
– Leda graph library [14]. Ezzel a projekttel demonstráltuk az elemző képességeit bonyolult sablonok (template-k) kezelésére.
– StarOffice Writer [17]. Ez egy nagy C++ projekt, ami 9 449 darab forrásállományból áll (több mint 1.7 millió nem előfeldolgozott programsor!). Ezzel a projekttel ellenőriztük az elemző képességeit valós méretű, nagy projektek kezelésére.

A következő táblázat tartalmazza a projektek méreteit:

Projekt	**Fájlok száma**	**Méret**	**Sorok száma**
Jikes	77	3,5MB	94 611
Leda	508	2,9MB	116 752
Writer	9 449	66,5MB	1 764 574

A következő táblázat tartalmazza az elemzési időt (minden teszt egy Intel PIII-800 gépen lett elvégezve 384MB RAM memóriával Windows 2000 operációs rendszer alatt):

Extrakció	**Idő**	**Memória**
Jikes	00:01:02	19MB
Leda	00:05:50	49MB
Writer	01:55:09	139MB

A következő táblázat tartalmazza a beazonosított C++ elemek számát:

Projekt	**Osztályok**	**Névterek**	**Függvények**	**Attribútumok**
Jikes	275	1	3 471	1 643
Leda	1 563	1	10 802	8 287
Writer	4 988	99	61 533	23 862

Látható, hogy a memóriafogyasztás körülbelül arányos a bemenet komplexitásával (vagyis a bemenet különböző elemeinek számával, mint pl. az osztályok és függvények számával). Ebből és a StarOffice Writer sikeres elemzéséből következik, hogy a rendszer képes nagy, valós méretű projektek kezelésére.

6. Összegzés

A cikkben bemutattunk egy sémát a C++-hoz és egy reverse engineering eszközt, ami a séma szerint állít elő adatokat a forráskód elemzéséből. Az eszköz ingyenes kutatási és oktatási célokra. Célunk, hogy segítsük a kutatókat a munkájukban. A keretrendszert használva nem kell minden kutatónak magának megvalósítania a megfelelő elemzőt, hanem a Columbus-t használva jobban koncentrálhat a saját konkrét kutatási területére.

A munkának fő előnye a moduláris, közös séma, a hatékony elemző, ami a sémának megfelelő adatokat állít elő, valamint a bővíthető keretrendszer, ami egybefog mindent. A

keretrendszer több népszerű eszközzel is együttműködik (pl. Rational Rose, Together, rigi, CodeCrawler), és könnyedén bővíthető, hogy még több programot támogasson.

A jövőben szeretnénk a támogatott eszközök számát növelni, valamint hozzáadni a Java programozási nyelvet is. Ez persze egy Java Sémát is igényel.

Hivatkozások

[1] M. Armstrong and C. Trudeau. *Evaluating Architectural Extractors*. In Proceedings of WCRE'98, pages 30-39, Oct. 1998.

[2] B. Bellay and H. Gall. *An Evaluation of Reverse Engineering Tool Capabilities*. In Software Maintenance: Research and Practice 10, pages 305-331, Oct. 1998.

[3] Zs. Balanyi and R. Ferenc. *Mining Design Patterns from C++ Source Code*. In Proceedings of the 7th International Conference on Software Maintenance (ICSM 2003), to appear. IEEE Computer Society, Sept. 2003.

[4] E. J. Chikofsky and J. H. Cross II. *Reverse Engineering and Design Recovery: A Taxonomy*. In IEEE Software 7, pages 13-17, Jan. 1990.

[5] S. Demeyer, S. Ducasse, and M. Lanza. *A Hybrid Reverse Engineering Platform Combining Metrics and Program Visualization*. In Proceedings of WCRE'99, 1999.

[6] R. Ferenc and Á. Beszédes. *Data Exchange with the Columbus Schema for C++*. In Proceedings of CSMR 2002, pages 59-66, Mar. 2002.

[7] R. Ferenc, Á. Beszédes, M. Tarkiainen, and T. Gyimóthy. *Columbus – Reverse Engineering Tool and Schema for C++*. In Proceedings of the 6th International Conference on Software Maintenance (ICSM 2002), pages 172-181. IEEE Computer Society, Oct. 2002.

[8] R. Ferenc, S.E. Sim, R. C. Holt, R. Koschke, and T. Gyimóthy. *Towards a Standard Schema for C/C++*. In Proceedings of WCRE'01, pages 49-58. IEEE Computer Society, Oct. 2001.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley Pub Co, 1995.

[10] Homepage of FrontEndART Software Ltd. http://www.frontendart.com

[11] R. Holt, A. Winter, and A. Schürr. *GXL: Towards a Standard Exchange Format*. In Proceedings of WCRE'00, pages 162-171, Nov. 2000.

[12] International Standards Organization. *Programming languages – C++*, ISO/IEC 14882:1998(E) edition, 1998.

[13] IBM Jikes Project. http://oss.software.ibm.com/developerworks/opensource/jikes

[14] K. Mehlhorn and S. Naeher. *Leda: A platform for combinatorial and geometric computing*. In Cambridge University Press, 1997.

[15] H. A. Müller, K. Wong, and S. R. Tilley. *Understanding Software Systems Using Reverse Engineering Technology*. In Proceedings of ACFAS, 1994.

[16] A. Quilici. *Reverse Engineering of Legacy Systems: A Path Toward Success*. In Proceedings of ICSE'95, pages 333-336, 1995.

[17] The StarOffice Homepage. http://www.sun.com/software/star/staroffice