OOOK

V. Országos Objektumorientált Konferencia

Információs társadalom: kihívások és válaszok

Dobogókő, BM Pilis Üdülőszálló

2002. október 16-17.

Neumann János Számítógép-tudományi Társaság
NJSZT Objektumorientált Fejlesztők és Felhasználók Klubja

OOOK

V. Országos Objektumorientált Konferencia

Információs társadalom: kihívások és válaszok

Dobogókő, BM Pilis Üdülőszálló

2002. október 16-17.

Neumann János Számítógép-tudományi Társaság
NJSZT Objektumorientált Fejlesztők és Felhasználók Klubja

TARTALOMJEGYZÉK

Plenáris előadók:

Böszörményi László
University of Klagenfurt, Institute for InformationTechnology

Anastasius Gavras
EURESCOM

Steve Latchem
Aonix Europe Limited

Andrew Watson
Object Management Group

Scott Ambler
Ronin International Inc.

Henry S. Thompson
University of Edinburg, W3C

PROGRAMBIZOTTSÁG

Elnök: ***Kozma László***

Tagok

Alföldi István
Csink László
Dömölki Bálint
Juhász István
Kelen András
Kondorosi Károly
Kovács András
Papp Sándor
Soóky Árpád
Zsemlye Tamás

SZERVEZŐBIZOTTSÁG

Elnök: ***Juhász István***

Tagok
Aranyosné Varga Gabriella
Hetthéssyné Papp Gizella
Pádár Adrienn
Veress Péter

Köszöntő

Az V. Országos Objektumorientált Konferencia Programbizottsága és Szervezőbizottsága nevében tisztelettel köszöntjük résztvevőinket.

Az objektum elvű technológiák napjainkra meghódították az információs társadalmat. Az oktatásban , a kutatóintézetekben és számos alkalmazói területen kutatók és fejlesztők hada foglalkozik az objektum elvű technológiák elméleti és gyakorlati problémáival. Büszkék vagyunk arra, hogy Magyarország sem marad el a világ élvonalától ezen kutatásokban, alkalmazásokban. Erre talán az a több mint 50 előadás is bizonyíték, amely a konferencia két napján három szekcióban elhangzik majd. Az előadások nyelve magyar. Ezúton is tisztelettel kérjük az előadókat, hogy használják ki ezt az alkalmat is arra, hogy szűkebb szakmánk nyelvét tovább gazdagítsák frappáns magyar szakkifejezésekkel.

A konferencia rangját jelzi a meghívott előadások magas színvonala és nagy száma is. A nyitó plenáris előadásokat Mr. Andrew Watson, az Object Management Group technikai igazgatója és Dr. Böszörményi László a klagenfurti egyetem professzora tartja. Az egyes szekciókat Mr. Scott Ambler, a denveri székhelyű Ronin International Inc. elnöke, Mr. Anastasius Gavras, az EURESCOM illetve Mr. Steve Latchem, az Aonix Europe Limited képviselője nyitják meg izgalmasnak ígérkező előadásaikkal. A záró plenáris előadáson Mr. Henry S. Thompsont, a University of Edinburgh munkatársát, az XML és a World Wide Web Consortium szerkesztőbizottságának vezetőjét hallgathatjuk meg. A konferencia programját gazdagítja a Modellezés a gyakorlatban címmel szervezett kerekasztal beszélgetés is, ahol a téma iránt érdeklődő részvevőink kicserélhetik gondolataikat, és személyesen találkozhatnak ismét a meghívott előadók többségével.

Egy konferencia sikere az elhangzott előadások tartalmán, színvonalán felül a szervezők előkészítő munkáján és a szponzorok támogatásán is múlik. Ezen a helyen is köszönetünket fejezzük ki minden támogatónknak nagyvonalú felajánlásaikért, amelyek lehetővé tették többek között a neves meghívott előadók jelenlétét és a diákok, doktorandusz hallgatók részvételét rendezvényünkön. Köszönjük a szervezőbizottság és a programbizottság tagjainak áldozatos munkáját, a Neumann János Számítógép-Tudományi Társaság munkatársainak a technikai szervezésben nyújtott nélkülözhetetlen segítségét. Végül, de nem utolsó sorban köszönjük a konferencia részvevőinek a gondosan elkészített, tartalmas előadásokat. A konferencia munkájához sok sikert kívánunk.

Dobogókő, 2002. október 16.

Kozma László
a Programbizottság elnöke

Juhász István
a Szervezőbizottság elnöke

Áttekintő táblázat

2002.10.16. (szerda)			
10:00-10:30	Megnyitó		
	Kemény János Díj átadása		
10:30-11:30	Plenáris előadás		
11:30-12:30	Plenáris előadás		
12:30-14:00	Ebéd		
14:00-16:00	A	B	C
	Szekcióülések		
16:00-16:30	Kávészünet		
16:30-18:30	A	B	C
	Szekcióülések		
18:30	Vacsora		
19:30	Kerekasztal		
2002.10.17. (csütörtök)			
9:00-11:00	A	B	C
	Szekcióülések		
11:00-11:30	Kávészünet		
11:30-13:00	A	B	C
	Szekcióülések		
13:00-14:30	Ebéd		
14:30-16:30	A	B	C
	Szekcióülések		
16:30-17:00	Kávészünet		
17:00	A konferencia zárása, Plenáris előadás		
19:00	Állófogadás		

PROGRAM

2002.10.16. (szerda)

10:00 A konferencia megnyitása: Kelen András (Triad Computer Services)
A konferenciát üdvözli: Kozma László (ELTE)
A Kemény János Díj átadása: Remzső Tibor (NJSZT)

10:30 Plenáris előadás
Andrew Watson (OMG)
Model Driven Architecture: Integration in the new Millenium

11:30 Plenáris előadás
Böszörményi László (University of Klagenfurt)
Adaptive Software for Adaptive Multimedia

12:30 *Ebéd*

14:00 Szekcióülések

16:00 *Kávészünet*

16:30 Szekcióülések

18:30 *Vacsora*

19:30 *Modellezés a gyakorlatban* – Kerekasztal
Moderátor: Hutter Ottó
Válaszadók: Scott Ambler, Steve Latchem, Henry S. Thompson, Andrew Watson, Boros Péter, Vég Csaba, Zsuffa Zsolt

2002.10.17. (csütörtök)

9:00 Szekcióülések

11:00 *Kávészünet*

11:30 Szekcióülések

13:00 *Ebéd*

14:30 Szekcióülések

16:30 *Kávészünet*

17:00 A konferencia zárása: Kelen András (Triad Computer Services)
Plenáris előadás
Henry S. Thompson (W3C): Schemas, Infosets and Objects:
The Web Architecture of the Future StartsToday

19:00 *Állófogadás*

SZEKCIÓK

2002.10.16. szerda

A szekció

***Szekcióelnök*: Kondorosi Károly**

14:00 - 14:30	Antal Zsuzsa: MDA alapú modellezés
14:30 - 15:00	Bányai Attila: Az MDA szabvány megvalósítása minta alapú kódgenerálással
15:00 - 15:30	Csiszár Tibor - Kókai Tamás: A Roleorientált modellezés alapjai
15:30 - 16:00	Ugron Balázs - Kozma László - Hajdara Szabolcs: A strukturált programozástól az objektum elvű technológiáig

***Szekcióelnök*: Kondorosi Károly**

16:30 - 17:00	Sinkó Annamária: OO módszertanok áttekintése
17:00 - 17:30	Bognár Tamás: A komponens alapú szoftverfejlesztés a gyakorlatban
17:30 - 18:00	Bertalan Gábor : Életciklus-kezelés az MSZ ISO/IEC 12207 szabvány alapján
18.00 - 18.30	Kondorosi Károly - Várkonyi József - Benedek András: Modellezés a jogban

B szekció

***Szekcióelnök*: Zsemlye Tamás**

14:00 - 14:30	Zsemlye Tamás: Java 2 Micro Edition
14:30 - 15:00	Juhász Zoltán: Java, Jini és szolgáltatás-orientált programozás
15:00 - 15:30	Darmai Gábor: Java web szolgáltatások és alkalmazásuk
15:30 - 16:00	Németh László: Java J2EE alapú alkalmazás integrálás

***Szekcióelnök*: Zsemlye Tamás**

16:30 - 17:00	Ertner Péter - Frigó József: Java J2EE alapú portál technológiák
17:00 - 17:30	Fazekas Imre: Objektumok futásidejű kezelése
17:30 - 18:00	Boros Péter: Java alapú keretrendszer
18.00 - 18.30	Molnár István - Simon Géza: Model-View-Controller paradigma web-környezetben, J2EE alkalmazásokban

C szekció

***Szekcióelnök*: Csink László**

14:00 - 14:30	Hajdara Szabolcs - Kozma László - Ugron Balázs: Párhuzamos programok szintézise és objektum elvű kiterjesztése
14:30 - 15:00	Gyapay Szilvia - Pataricza András: UML modellek Petri-háló alapú erőforrás-allokációja és ütemezése
15:00 - 15:30	Ferenc Rudolf - Beszédes Árpád - Gyimóthy Tibor: Columbus: C++ tervrekonstrukciós eszköz és séma
15:30 - 16:00	Pataricza András - Varró Dániel: UML modellek automatikus transzformációi

***Szekcióelnök*: Csink László**

16:30 - 17:00	Huszerl Gábor - Majzik István - Pap Zsigmond - Pataricza András - Petri Dániel - Varró Dániel: Keretrendszer nagymegbizhatóságú. biztonságkritikus rendszerek fejlesztéséhez és teszteléséhez
17:00 - 17:30	Szabolcsi Judit: A D programozási nyelv
17:30 - 18:00	Kis Gergely - Orosz József - Pintér Márton - László Zoltán - Thomas Genssler: C# programok metaszintű manipulációja
18.00 - 18.30	Alberti István - Balássy György: A .NET Framework és a C# nyelv objektum orientált szemszögből

SZEKCIÓK

2002.10.17. Csütörtök

A szekció

Szekcióelnök: **Dömölki Bálint**

9:00 - 9:50 ***Steve Latchem(Aonix Europe Limited) : Repositories for Components and Web Services***

10:00 - 10:30 Jónás Richárd: Komponesalapú webalkalmazásfejlesztési rendszer

10:30 - 11:00 Tilly Károly - Baranyi Szabolcs: Invariáns kezelői felületek

Szekcióelnök: **Dömölki Bálint**

11:30 - 12:00 Andriska Zoltán: Nyomkövetés az UML modellből előállított kódban

12:00 - 12:30 Szulman Péter - László Zoltán - Markus Bauer: Objektum-orientált programok struktúra-modelljének egyszerűsítése és megjelenítése

12:30 - 13:00 Veréb Krisztián: Objektum alapú keresési és indexelési technológia képadatbázisokhoz

Szekcióelnök: **Markovits Péter**

14:30 - 15:00 Molnár Balázs: Web szolgáltatások és Java objektumok Oracle környezetben

15:00 - 15:30 Petrohán Zsolt: Adatbázis alapú Java alkalmazások gyors fejlesztése üzleti objektum keretrendszerek segítségével (BC4J, UIX)

15:30 - 16:00 Markovits Péter: Objektumorientált adatbázis struktúrák és az XML

16:00 - 16:30 Petrohán Zsolt: Adatbázisok és folyamatok tervezése UML segítségével Oracle környezetben

B szekció

Szekcióelnök: **Tarnay Katalin**

9:00 - 9:50 ***Anastasius Gavras (EURESCOM): Cooling the hell out of distributed telecom applications' deployment***

10:00 - 10:30 Papp Sándor: A nyílt szolgáltatás-átlépő kezdeményezés (OSGI) és az objektum-orientált környezet

10:30 - 11:00 Tarnay Katalin: Szolgáltatásfelfedező protokollok

Szekcióelnök: **Tarnay Katalin**

11:30 - 12:00 Jaskó Szilárd : Szolgáltatásfelfedezés modellezése

12:00 - 12:30 Dulai Tibor - Muhi Dániel: Service Location Protocol (SLP)

12:30 - 13:00 Muni Dániel : Session Initiation Prtotocol (SIP)

Szekcióelnök: **Papp Sándor**

14:30 - 15:00 Kollarics Róbert: Hálózati kommunikáció objektum-orientált megvalósitása

15:00 - 15:30 Fazakas Antal: OO technológiák a protokoll tesztelésben

15:30 - 16:00 Csontos Péter: Új programozási paradigmák a láthatáron

16:00 - 16:30 Espák Miklós: Új paradigmák a szoftverfejlesztésben

C szekció

Szekcióelnök: **Bertalan Gábor**

9:00 - 9:50 ***Scott Ambler (Ronin International Inc.): RUP in Practice***

10:00 -10:30 Gulácsi Ferenc: Nyílt forráskódú fejlesztés kereskedelmi alkalmazásoknál. A Progress objektumorientált fejlesztési projektje.

10:30 - 11:00 Vég Csaba: Alkalmazásfejlesztés - a gyakorlatban

Szekcióelnök: **Bertalan Gábor**

11:30 - 12:00 Kókai Tamás - Csiszár Tibor: Roleorientált szoftverfejlesztés a gyakorlatban

12:00 - 12:30 Vég Csaba: „Információs környezetek" a szervezésben és tervezésben

12:30 - 13:00 Bognár Zsolt: Kritikus alkalmazások fejlesztése OO CASE eszközökkel

Szekcióelnök: **Kollár Lajos**

14:30 - 15:00 Kázmér Adorján: Többrétegű alkalmazás fejlesztése OO módszerrel

15:00 - 15:30 Dobán Orsolya - Patricza András: Szoftverfejlesztési Folyamatok Optimalizálása

15:30 - 16:00 Dolla Gábor: Elektronikus dokumentum küldése bárkinek

16:00 - 16:30 Benkő Tamás - Fokt Attila: Információ integrálás logikai alapokon

ELŐADÁS VÁZLATOK

Plenáris Előadások

László Böszörményi

Adaptive Software for Adaptive Multimedia

Adaptation is becoming an increasingly important tool for resource and media management in distributed multimedia systems. Best-effort scheduling and worst-case reservation of resources are two extreme cases, none of them well suited to cope with large-scale, dynamic multimedia systems. The middle course can be met by a system which dynamically adapts its data, resource requirements, and processing components to achieve user satisfaction.

The need for adaptive systems implies enhanced requirements on software technology. An adaptive system has to be able dynamically reconfigure itself, allocate new resources, migrate and/or replicate software components etc. The consequences of these new requirements have not been explored yet appropriately.

In the course of the ADMITS project (Adaptation in Distributed Multimedia IT Systems) various aspects of adaptive distributed multimedia are investigated. As a part of the project an Adaptive Multimedia Server (AMS) is developed, which is able to dynamically change the number and the functionality of its nodes on demand. The architecture of AMS serves in the talk as an example for what we call "adaptive software

A SZEKCIÓ

2002. október 16.
14:00 – 16:00
Szekcióelnök: Kondorosi Károly

Antal Zsuzsa

MDA alapú modellezés

Napjaink legnagyobb kihívása az integráció.

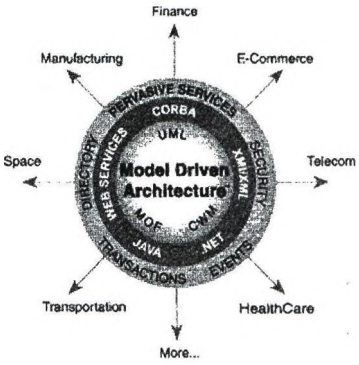
Információs vagyonunk különböző platformon, operációs rendszeren, hálózati protokollon működő, eltérő technológiákkal készített alkalmazásokban fekszik. Természetes igény biztosítani ezeknek a heterogén rendszereknek az együttműködését, és felkészülni arra, hogy a jövőben készülő rendszereink ennek az integrált megoldásnak a részeiként valósuljanak meg.
A Model Driven Architecture (MDA) az Object Management Group (OMG) legújabb lépése az integráció problémájának megoldására.

Az OMG már megalakulásakor azt a célt tűzte ki magának, hogy segítséget nyújtson a heterogén környezetben élő informatikai világnak az integráció megteremtésében. Számos munka, elfogadott és alkalmazott szabvány segítette elő ezt a törekvést – gondoljunk csak a CORBA vagy az UML szabványokra. Az MDA-val újabb nagy lépést tett az OMG az integráció problémájának megoldására, hiszen a saját és mások korábbi munkájára építkezve egy új megközelítést, új módszert adott a rendszerek specifikálására és megvalósítására.

Az MDA alapú fejlesztés során az a cél, hogy a készítendő rendszer funkcionalitását és viselkedését egy olyan modellben írjuk le, amely nem tartalmaz a megvalósítási technológiához kapcsolódó döntéseket. Az így létrehozott platform-független modellt kell leképezni egy olyan modellé, amely az alkalmazott technológia lehetőségeit kihasználva, már platformtól függően valósítja meg a rendszer viselkedését, és alkalmas arra, hogy a kódgenerálás és a kódolás kiindulási alapja legyen.

Az MDA újdonsága abban rejlik, hogy az egyes modellek megalkotása, illetve a modellek közötti leképezések megvalósítása elfogadott, illetve kifejlesztés alatt álló szabványok segítségével, akár automatizáltan történik.

A technológiától független működési modell, valamint a modellezés és a modellek közötti leképezések szabványos módja teszi lehetővé, hogy a meglévő alkalmazásainkat könnyebben integrálhassuk és egy olyan hosszú távra is érvényes *modell vezérelt architektúrát* alakítsunk ki, amely előre felkészült az új technológiákkal készülő jövőbeni rendszereink befogadására.
Az előadás célja bemutatni az MDA alapelveit, felépítését, a szabványosítás aktuális helyzetét valamint a gyakorlati alkalmazásának lehetőségeit.



Bányai Attila

Az MDA szabvány megvalósítása minta alapú kódgenerálással

Az előadás betekintést nyújt az Object Management Group (OMG – www.omg.org) új, Model Driven Architecture (MDA) nevű szabványába, valamint egy ennek megfelelő megvalósításba. Az MDA koncepció egyik legfontosabb eleme, hogy a modellezésnek két szintjét különbözteti meg, a platform független (Platform Independent Model – PIM) és a platform specifikus modellezést (Platform Specific Model – PSM). A platform fogalmába az MDA keretében a programozási nyelvektől, adatbázis kezelő rendszerektől, hardvertől, operációs rendszertől kezdve sok minden beletartozik. Az idők során a platformok változnak, azonban az alkalmazások jelentős részénél készíthető olyan robosztus, időtálló modell (PIM), ami túlél(het)i a platformváltásokat.

A megközelítés számos módszertanban megfigyelhető, többek között a Unified Process-ben is, ahol az elemzés egyik feladata a robosztus, időtálló struktúra kialakítása egy idealizált, platform független környezetben, míg a tervezés feladata a PSM-hez hasonlóan az elemzési modell leképzése a rendelkezésre álló platformra.

A PIM, PSM és a kód előállítására, szinkronban tartására a piacon találhatóak megoldások, a jövőben azonban még számos új terv- és kódgenerátor megjelenése várható.

Az előadás bemutatja a terv- és a kódgenerátorok közötti legfontosabb különbségeket, valamint egy lehetséges MDA megvalósítást, amelynek a magját a minta alapú kódgenerálás adja.

Csiszár Tibor – Kókai Tamás

A Roleorientált modellezés alapjai

Az általunk kifejlesztett Roleorientált módszer a nagy mennyiségű adatot kezelô, gyakran változó informatikai rendszerek hatékony fejlesztését segíti elô. Cikkünk a fejlesztés életciklusának csupán egyetlen, de talán az egyik legfontosabb részére koncentrál, nevezetesen a modell megalkotásához szükséges fogalmakat mutatja be.
A bevezetôben részletesen kifejtésre kerül a módszer világszemlélete, valamint sorra vesszük a modell által teljesítendô kritériumokat is.
Modellezô módszerünk tulajdonképpen a Role-ok és közöttük lévô Relation-ok azonosítása szolgáló eljárás. Ennek megfelelôen egy fejezet a Role fogalmával és a Role-ok lehetséges csoportosításával fogalkozik.
Írásunk legnagyobb fejezete a kapcsolatkezelésrôl szól. Bemutatjuk az általunk használt rekurzív és nem rekurzív kapcsolatokat, illetve kitérünk a segítségükkel megadható constraint-ekre is.
Módszerünk még korántsem befejezett. Az utolsó részben a még elôttünk álló feladatokat vesszük sorra.

Megjegyzés: Az érdeklôdôk figyelmébe ajánljuk a témához szorosan kapcsolódó a Roleorientált szoftverfejlesztés a gyakorlatban című elôadást, amely az elmélet alkalmazását mutatja be.

Ugron Balázs – Kozma László - Hajdara Szabolcs

A strukturált programozástól az objektum elvű technológiáig.

A nagy programrendszerek készítése megbízható minőségben, elfogadható határidőn belül a szoftverkészítők régi nagy problémája. Időrendben haladva a szoftverkészítés fejlődési fázisai röviden a következők:

- 1969 előtt: a szoftverfejlesztések ad hoc módon történtek, bár például E. W. Dijkstra már 1965 körül kidolgozta a strukturált tervezés alapelveit;
- 1969-71: felülről lefelé haladó tervezés, lépésenkénti finomítás;
- 1972-73: strukturált programozás;
- 1974-75: megbízhatóság, szisztematikus tesztelés, helyességbizonyítási módszerek;
- 1976-77: követelményanalízis, formalizált specifikációk;
- 1978-80: automatikus fejlesztőeszközök;
- 1980-89: CASE eszközök;
- 1990 után: szakértői rendszerek eszközeinek bevitele a szoftver technológiákba;
- Napjainkban: az objektum elvű technológia rohamos terjedését éljük;
- A közeli jövő: a csoportmunkát is támogató, vizualizációs eszközökre építő objektum elvű technológia.

A 80-as évek végén, a 90-es évek elején a szakemberek azt jósolták, hogy az ezredfordulóig a szoftverfejlesztésben az objektum elvű koncepciók széleskörű elterjedésére számíthatunk. E

jóslat napjainkban válik valósággá. Ugyanakkor a 90-es években a hálózatok területén is robbanásszerű fejlődés következett be. A nagy hálózatok az ezredfordulóra váltak általánossá. Az ezt megalapozó elméleti kutatások az elosztott rendszerek területén régóta sikerrel folynak. Napjainkban az ilyen rendszerek fejlesztését támogató környezetek rohamosan terjednek, gondoljunk csak a Java nyelvre és annak fejlesztő környezeteire. Megindult e területen a szabványosítási folyamat is. Ennek eredményeként született meg a CORBA (Common Object Request Broker Architecture) szabvány, amelynek elsődleges célja az elosztott objektum-elvű alkalmazások számára az együttműködést és a hordozhatóságot biztosítani.

A SZEKCIÓ

2002. október 16.
16:30 – 18:30
Szekcióelnök: Kondorosi Károly

Sinkó Annamária

OO módszertanok áttekintése

A nagy programrendszerek készítése megbízható minőségben, elfogadható határidőn belül a szoftverkészítők régi nagy problémája. Időrendben haladva a szoftverkészítés fejlődési fázisai röviden a következők:

- 1969 előtt: a szoftverfejlesztések ad hoc módon történtek, bár például E. W. Dijkstra már 1965 körül kidolgozta a strukturált tervezés alapelveit;
- 1969-71: felülről lefelé haladó tervezés, lépésenkénti finomítás;
- 1972-73: strukturált programozás;
- 1974-75: megbízhatóság, szisztematikus tesztelés, helyességbizonyítási módszerek;
- 1976-77: követelményanalízis, formalizált specifikációk;
- 1978-80: automatikus fejlesztőeszközök;
- 1980-89: CASE eszközök;
- 1990 után: szakértői rendszerek eszközeinek bevitele a szoftver technológiákba;
- Napjainkban: az objektum elvű technológia rohamos terjedését éljük;
- A közeli jövő: a csoportmunkát is támogató, vizualizációs eszközökre építő objektum elvű technológia.

A 80-as évek végén, a 90-es évek elején a szakemberek azt jósolták, hogy az ezredfordulóig a szoftverfejlesztésben az objektum elvű koncepciók széleskörű elterjedésére számíthatunk. E jóslat napjainkban válik valósággá. Ugyanakkor a 90-es években a hálózatok területén is robbanásszerű fejlődés következett be. A nagy hálózatok az ezredfordulóra váltak általánossá. Az ezt megalapozó elméleti kutatások az elosztott rendszerek területén régóta sikerrel folynak. Napjainkban az ilyen rendszerek fejlesztését támogató környezetek rohamosan terjednek, gondoljunk csak a Java nyelvre és annak fejlesztő környezeteire. Megindult e területen a

gondoljunk csak a Java nyelvre és annak fejlesztő környezeteire. Megindult e területen a szabványosítási folyamat is. Ennek eredményeként született meg a CORBA (Common Object Request Broker Architecture) szabvány, amelynek elsődleges célja az elosztott objektum-elvű alkalmazások számára az együttműködést és a hordozhatóságot biztosítani.

Bognár Tamás

A komponens alapú szoftverfejlesztés a gyakorlatban

Az előadás betekintést nyújt a komponens alapú fejlesztés módszereiről, technikai , üzleti és gazdasági előnyeiről, felhasználási területeiről. Átfogóan ismertet egy komponens alapú módszertant a Select Perspective -et, a hozzá tartozó eszköz-megvalósítással.

A Select Perspective fejlesztési életciklust biztosít komponens alapú fejlesztésekhez, mely segíti a párhuzamos fejlesztési tevékenységeket és lerövidíti a piacrakerülés idejét.

A Perspective SW-fejlesztési életciklus munkafolyamatokból áll, melyek iteratív és inkrementális fejlesztési megközelítésen alapulnak.
Az iterációk lehetővé teszik, hogy a projekt igazodjon a változó követelményekhez.
A növekmények a projekt mérföldköveinek tekinthetők, a megoldás működő és tesztelt szeleteit adják.

Az Aonix Select Perspective objektum-orientált, iteratív és inkrementális szoftverfejlesztési módszertana hosszú múltra tekint vissza. 1998-ig nem sokban tért el a szokványos módszertanoktól (pl. UP), de ekkor átalakult, hogy magába foglalja a komponens alapú fejlesztést.
A Select Perspective használati eset és UML központú, azonban az üzleti alkalmazások fejlesztéséhez UML nem mindig elegendő. Ezért a Select Perspective kiterjed az üzleti folyamatok -, valamint a relációs adatbázisok modellezésére is.

A Select Perspective központjában az előállít – kezel - felhasznál (Supply – Manage - Consume) folyamat áll. A komponensek szempontjából három, jól elkülöníthető szerepet különböztet meg. Ezek : komponens felhasználó, komponens kezelő és a komponens előállító.
A folyamat a komponens felhasználónál kezdődik. Ö ismerkedik meg a valós világból vett problémával, majd modellezi azt, és jól elkülöníthető komponensekre bontja. Meghatározza az egyes komponensek interfészeit és vázolja a komponensek együttműködését. Ebben a fázisban komponensek határa még nem végleges, csak kísérleti jellegű.

Amint a szükséges komponensek specifikálva vannak, a kezelő a komponens menedzser segítségével megvizsgálja, hogy létezik-e már ez, vagy valamilyen hasonló komponens, megpróbálja őket megtalálni a komponens adatbázisban. A megtalált komponenseket újrahasznosíthatjuk, a hiányzókat pedig el kell készíteni. A kezelő publikálja a hiányzó komponensek specifikációit, a komponens előállító számára.

A komponens előállító részletesen kidolgozza a specifikáció alapján a komponens terveit. Elképzelhető, hogy a tervezés során megváltozik a komponens interfésze, ezt publikálni kell a komponens kezelő. Majd a komponens implementálása következik, és a kész, futtatható komponens (akár a forráskódjával együtt) szintén a komponens kezelőnek rendelkezésére áll. Amint a hiányzó komponensek elkészültek, a komponens felhasználó frissíti a terveket a megváltozott specifikációk alapján, összeállítja a rendszert és teszteli.

Az Object Management Group (OMG) Model Driven Architecture [www.omg.org/mda] (MDA) szabványa megvalósítható a komponens alapú fejlesztés felöli megközelítéssel is, ami a platform független modell és a platform függő modell határát a komponens specifikációja és részletes tervei között húzza meg. A szoftver az elemi feladatokat megvalósító komponensek együttműködésével valósul meg, ezen a szinten programnyelv és platform függetlenül. Ugyanakkor a komponensek belsejének részletes terve a generált kód pontos megfelelője kell, hogy legyen, amit folyamatos, automatizált szinkronban tartásukkal érhetünk el.

Bertalan Gábor

Életciklus-kezelés az MSZ ISO/IEC 12207 szabvány alapján.

Az előadás betekintést nyújt a szoftver életciklusát leíró MSZ ISO/IEC 12207 szabványba.

Szoftverek esetén is van értelme általában életciklusról beszélni, de különösen az objektum-orientált
paradigmára épülő rendszereknél, ahol az iteratív-inkrementális fejlesztés a ciklikusság folytonosságát kiemeli.

A 2000. májusában honosított MSZ ISO/IEC 12207 nemzetközi szabvány egységes fogalmi keretet hoz létre a szoftver-életciklus folyamatokra olyan jól definiált terminológiát használva, amely a szoftveripar számára kiindulásul szolgálhat.

A szabvány olyan folyamatokat, tevékenységeket és feladatokat tartalmaz, amelyek szoftvert tartalmazó rendszerek, különálló szoftvertermékek és szoftverszolgáltatások beszerzése, valamint szoftvertermékek szállítása, fejlesztése, üzemeltetése és karbantartása során alkalmazandók.

Témák:

- A szabvány használatának előnyei
- A szabvány alkalmazása
- A szabvány illesztése
- A szabvány korlátai
- Fogalom meghatározások
- A szabvány szerkezete
  - Az életciklus fő folyamatai
  - Az életciklus támogató folyamatai
  - Az életciklus szervezeti folyamatai
- Eszköz támogatás (Követelmény-kezelés, konfiguráció-kezelés, stb.)

Kondorosi Károly – Várkonyi József – Benedek András

Modellezés a jogban

A jogalkotás és jogalkalmazás a társadalmak örökzöld problémája. Aligha kétséges, hogy a „jogállam" ma világszerte komoly diszfunkciókat mutat. A problémák egyik - ha nem is legfontosabb - okát abban kereshetjük, hogy a jogállam alapját képező szabályrendszert minden szinten szabad szöveges reprezentációban kezelik. Ez a reprezentáció pedig nem közvetíti pontosan a jogalkotói szándékot a többi szereplő (jogalkalmazók, jogalanyok) felé, sőt, előállítása a jogalkotót sem kényszeríti egyértelműségre és a konzisztencia fenntartására. Igaz ez akkor is, ha a jogi szaknyelv, valamint esetlegesen alkalmazott „template" jellegű szerkezeti előírások ösztönös formalizációs törekvéseket mutatnak.

A szerzők által kidolgozott JOGSEGÉD *koncepció* lényege, hogy a jogalkotó a jogszabályok hiteles formális modelljét is létrehozza és közreadja, a jogalkalmazók és jogalanyok pedig ezt felhasználhatják. A formalizmussal szemben elvárás, hogy könnyen érthető, ugyanakkor informatikai eszközökkel könnyen támogatható legyen. A JOGSEGÉD *rendszer* pedig a modellek kezelését „CAD" jelleggel támogató szoftverrendszer.

Tekintettel arra, hogy a jog nemzetközi megítélés szerint is egyike az informatika behatolásának leginkább ellenálló területeknek, a szerzők jogászokkal és jogász doktoranduszokkal közösen több meglévő jogszabály kísérleti modelljét dolgozták ki. Az alkalmazott formalizmus a szöveges függvényforma, illetve annak „ha ... akkor ..." alakú szabályokkal kifejezett reprezentációja volt, amely közvetlen feldolgozható PROLOG alapú szakértői keretrendszerrel. A kedvező visszajelzések, a kritikák és a tapasztalatok alapján más formalizmusokat (metamodelleket) is megvizsgálandónak és kiértékelendőnek ítéltünk. Fő értékelési szempontoknak az érthetőséget, elterjedtséget, illetve a feldolgozási hatékonyságot tekintettük. Vizsgálataink alapján a további, jelenleg is folyó kísérletek egyik ígéretes irányának az objektum-modellek alkalmazását tartjuk.

Előadásunkban az előzmények áttekintése és a tapasztalatok értékelése után bemutatjuk az objektum-modellekkel folytatott kísérleteket a jogalkotásról szóló törvény, illetve a parlamenti házszabály modellezésére, valamint az „On-line JOGSEGÉD" koncepciót, amely a jogalkotókat, jogalkalmazókat és jogalanyokat on-line közösségnek tekintve vázolja fel egy elektronikus jogszolgálat jövőképét.

A SZEKCIÓ

2002. október 17.
9:00 – 13:00
Szekcióelnök: Dömölki Bálint

Steve Latchem (Aonix Europe Limited)

Repositories for Components and Web Services

Component Based Design and Development has been stated as the "engine" for the delivery of the new Internet and Web Service architectures of the future. Critical to its successful implementation are the concepts, principles and process surrounding the supply, management and consumption of components and services.

In this presentation, I present the critical processes, activities, dependencies and quality and accreditation measures required to successfully adopt the Supply, Manage, Consume Model, including the repository technology capabilities that are required. The presentation includes mini-examples from Aonix's clients who have implemented the process successfully.

In addition, the use of outsourced component and managed service suppliers, and massive parallel development of components for software solutions are discussed.

Jónás Richárd

Komponensalapú webalkalmazásfejlesztési rendszer

Napjainkban az Internet rohamos fejlődése üzleti és más szempontokból is indokolja, hogy megjelenjünk a Weben, azaz egyre fontosabb kérdés lesz, hogy hogyan tudunk hatékonyan webalkalmazást fejleszteni. Akkor lehetünk sikeresek e téren, ha minél gyorsabban tudjuk a követelményeket összegyűjteni, és a követelményekből minél hamarabb tudunk alkalmazást fejleszteni. Figyelembe kell venni, hogy manapság az alkalmazásokat fejlesztőcsoportok elosztott fejlesztőeszközzel készítik, és azt is, hogy a gyors fejlesztés egyik kulcsfontosságú eleme az újrafelhasználhatóság biztosítása. Tehát meg kell oldanunk, hogy a fejlesztők ne csak a saját tudásukat legyenek képesek felhasználni, hanem mások ötleteit, megoldásait, szoftverkomponenseit is elérjék, újrafelhasználják.

A komponensek számtalan definícióira lelünk, ha a szakirodalomban meg szeretnénk keresni mi, is a komponens. Számunkra a komponens egy olyan újrafelhasználható kódrészlet, amely képes egy weboldal egy részének előállítására. Egy weblap minden része szerkezetileg és funkcionálisan jól elhatárolt feladatot old meg, melynek eredményeképpen létrejön a lap maga. Világos, hogy egy weblap által elvégzett tevékenység részekre osztható, ezen részek lesznek

esetünkben a komponensek. Minden komponens rendelkezik egy modellel, egy nézettel és egy vezérlővel, tehát a komponenseink a jól ismert MVC modellt követik. Hogyan definiálhatunk komponenseket?

A komponens modell része meghatározza azokat az adatokat, amellyel foglalkozni szeretnénk. Az adatok lehetnek konstans adatok, vagy lehetnek egy adatbázis-lekérdezés eredményének sorai is. Így világos, hogy olyan webalkalmazások készítését is támogatjuk, amelyek a tartalmukat adatbázisból nyerik. A komponens nézet része meghatározza, hogy az előbb definiált adatok hogyan jelenjenek meg a böngészőben. Ez biztosítja számunkra olyan mobil eszközök támogatását is, amelyek más megjelenítési stratégiával rendelkeznek (PDA, mobiltelefon). A komponens vezérlő része meghatározza a komponens részeinek kiértékelésének sorrendjét. Tehát biztosítja az adatok szelektív megjelenítését, új adattagok létrehozását, származtatását, stb. Az említett hármas egyidejű definiálásával egy konkrét komponens hozható létre.

Először a weblapot, mint megoldandó problémát, részekre kell bontanunk, majd a részeket, mint problémákat komponensek definícióival meg kell oldanunk. Az elkészítendő weblap a komponensek helyes összerakásával áll elő. A komponenseket konténerekbe szervezhetjük, így egyre bonyolultabb problémára biztosítunk megoldást. Természetesen a konténerek is belepakolhatók konténerekbe, így a weblap egy hierarchikus szerkezetű komponens-reprezentációját kapjuk meg. A konténer tekinthető komponensnek (szintaktikailag), hiszen a konténer adatrészét azon komponensek (vagy konténerek) alkotják, amelyeket a konténer tárol. A nézet meghatározza, hogy a konténer hogyan helyezze el a komponenseit, illetve a vezérlő dönt arról, mely komponensek élnek, mi legyen a kiértékelésük sorrendje, stb.

A komponensek újrafelhasználhatóságát biztosítja az öröklődési mechanizmus, amely segítségével új komponenseket tudunk származtatni meglévő komponensekből. A rendszer egyszeres öröklődést támogat, a szülő komponens tetszőleges része felüldefiniálható, így felhasználhatjuk a meglévő komponens modell, nézet vagy vezérlő részét.

A rendszerben a komponenseket és a konténereket az XML nyelv segítségével definiáltuk. Így a komponens adatrészében a konstans adatok XML dokumentum-töredékként jelennek meg, míg az adatbázisból jövő adatokat a *query* elemben megadott SQL lekérdezés végrehajtása után kapjuk meg. A nézetet XHTML nyelven kell leírnunk, hiszen a nézet a komponens XML definíciójának egy része. Miután elkészítettük a komponens definícióját, egy komponens-azonosító megadása után, elhelyezhetjük egy adatbázisba, ahonnan az azonosító, mint név alapján a rendszer bármikor elő tudja keresni. A konténereket is adatbázisban tároljuk, így mind a komponensek, mind a konténerek letesztelhetők, kereshetők, és újrafelhasználhatók.

A komponensek és konténerek definiálását, módosítását és végrehajtását az Interneten keresztül végezhetjük el egy webalkalmazás segítségével, így ezt a fejlesztőeszközt nem kell telepíteni a kliens gépekre, bár megjegyzendő, hogy sok fejlesztő esetén nagyobb hardvert igényel.

Elkészült egy implementáció is, amely MySQL adatbázistkezelőn, Apache webszerveren és Java 2 SDK 1.4-es fejlesztőkörnyezeten alapul. A webszerver JSP lapokon, JavaBean-eken keresztül biztosítja a komponensek adatbázisbeli elérését az MM MySQL JDBC driver segítségével. A rendszer egy Athlon 1.2GHz-es processzorral, 384MB fizikai memóriával ellátott számítógépre lett telepítve. Egy öt felhasználót szimuláló többszálas Java alkalmazás segítségével teszteltük a rendszert úgy, hogy a felhasználók 0 és 100 ezredmásodperc közötti véletlen

időnként érték el a szervert. A mérések szerint az átlagos válaszidő 0.05 másodperc, másodpercenkénti 12-15 kérés esetén. A teszteredmények azt mutatják, hogy az elkészített rendszer nem csak prototípus építésére, de alkalmazási futtatási környezetként is hatékonyan használható.

Tilly Károly – Baranyi Szabolcs

Invariáns kezelői felületek

Az egyre nagyobb számítási teljesítményű, olcsó személyi számítógépek elterjedésével az informatika a mindennapi élet részévé vált. Számos független szoftverergonómiai vizsgálat tanúsága szerint a felhasználó jártassága az adott szoftver kezelésében a hatékonyságot (a műveletvégzés sebességét és a hibázások arányát) alapvetően befolyásolja. Egy szoftver használhatósága első sorban a következő két módszer segítségével növelhető:

1. A felülethez való hozzáférés bonyolultságának csökkentése.
2. A felhasználó tudásának növelése.

Az (1.) módszert követve eljutunk az intelligens kezelői felületekhez. Ezek minimális beavatkozást igényelnek, és próbálnak az emberi kommunikációban megszokott természetes modalitások segítségével kommunikálni. Ígéretes irányzat, de számos technikai, és elméleti problémát vet fel, melyek megoldása rendkívül bonyolult, ezekre egyenlőre jó esetben is csak részmegoldások léteznek.

Az (2.) irányzat a felhasználó oktatását, adott alkalmazások kezelésének lehető legpontosabb begyakorlását jelenti. A módszer igen erős korlátja a megtanulandó felületek nagy száma, és az, hogy ugyanazon szolgáltatás több különböző felületen érhető el a gyártótól, eszköztől és divatirányzattól függően. A megtanulandó felületek *(F)* számát befolyásoló három fő tényező a következő:

Szolgáltatás (S): A kezelői felületen megjelenő a felhasználó számára adott funkciót jelentő parancs vagy programegység. A szolgáltatások nagy számának egyik oka az adatszerkezetek okozta eltérések, melyek gyakran irrelevánsak a felhasználó számára (pl. egy dokumentumot szeretnénk elolvasni és nem érdekel, hogy .doc vagy .pdf formátumú). Az adatszerkezetek invarianciája egy adott absztrakciós szint fölött érhető el, ezért bevezetjük az absztrakt szolgáltatás fogalmát, amely funkcionálisan írja le a szolgáltatás jelentését a felhasználó szemszögéből.

Eszközfüggés (E): Ugyanaz a szolgáltatás a kezelői felületet megjelenítő eszköztől függően (pl. GUI, HTML vagy WAP) legtöbbször logikailag is eltérő módon jelenik meg.

Implementációfüggés (I): A jelenlegi direkt manipuláción alapuló grafikus ablakozó felületek egyszerű kezelhetőséget ígérnek, de ehhez meg kell tanulni az adott alkalmazás felépítését, menüstruktúráját, elrendezését, vagyis sajátos „gondolkodásmódját". A felhasználóknak jelenleg több tucat hasonló funkciójú, de különböző felületű szoftver használatát kell megtanulniuk, sőt újra tanulniuk valahányszor egy új verzió megjelenik, illetve ha más gyártó termékét szeretnék használni.

A felhasználó által megtanulandó felületek száma okozta mentális terhelés tehát a következő összefüggéssel jellemezhető: $F = O(SEI)$. Az invariáns kezelői felületek alapgondolata, hogy a megtanulandó felületek száma *(F)* jelentősen csökkenthető, ha a felhasználó számára ugyanazon szolgáltatás logikailag mindig ugyanolyan kezelői felülettel rendelkezik, eszköztől és implementációtól függetlenül.

A szolgáltatások és a megjelenítés szétválasztásával a felhasználónak elegendő megtanulni az egyes felületek kezelésének sajátosságait, logikai felépítését, navigációs mintáit. A felületek tervezését és megtanulását elősegíti, ha elemeiket eszközfüggetlen, absztrakt szinten, funkcionalitásuk szerint rendszerezzük, és tervezési mintákat alakítunk ki. Az absztrakt szolgáltatások segítségével a szolgáltatás és az implementáció is teljesen szétválasztható. Az absztrakt szolgáltatás mint specifikáció szerepel, az egyes implementációk csak hatékonysági, minőségi jellemzőkben térnek el, funkcionalitásban nem. A nagyszámú szolgáltatás áttekintését és használatát elősegíti a szolgáltatások és megjelenítők rendszerezése. A rendszertan biztosítja, hogy a közös részeket csak egyszer kell megtanulni. Ezzel a mentális terhelés a legjobb esetben akár $F=O(S+E)$ nagyságrendűre csökkenthető. Másrészt az absztrakt szolgáltatásoktól élesen elkülönülnek az őket implementáló komponensek, az ún. *szolgáltatók*, amelyek az adott keretek között korlátlanul újrafelhasználhatók.
Jelen cikkünkben bemutatunk egy, a kezelői felületek invarianciáját biztosító, absztrakt szolgáltatásokon alapuló rendszert, továbbá összefoglaljuk az implementáció, és a mintaalkalmazások során szerzett tapasztalatainkat. Bemutatjuk az absztrakt szolgáltatások egy leírásmódját és az arra épülő rendszertant.

Andriska Zoltán

Nyomkövetés az UML modellből előállított kódban

Az előadás egy konkrét megoldást mutat be az UML modellből előállított Java kód futási idejű naplózására. Az UML modellből a kódgenerálás során az előállított kódba olyan hívásokat illesztünk be, melyek egy külső komponens segítségével naplózzák az alkalmazás futási idejű eseményeit. A naplózás a modellben definiált csomagstruktúra alapján engedélyezhető és tiltható, valamint a mélység is konfigurálható.
Az előadás bemutatja a megoldás elvi vázát, a megoldást lehetővé tevő minta alapú kódgenerálást, valamint egy konkrét példaalkalmazáson keresztül annak hasznosságát. A megoldás segítséget nyújt pl. a nyomkövetésben, a hibakeresésben, a unit tesztelésben, és a teljesítményhangolásban, így a fejlesztési időben jelentős csökkenés érhető el.
Az előadás külön kitér a minta alapú kódgenerálás néhány további értékes jellemzőjére: a termelékenység növekedésre, minőségbiztosításra, kockázat csökkentésre.

Szulman Péter – László Zoltán – Markus Bauer

Objektum-orientált programok struktúramodelljeinek egyszerűsítése és megjelenítése

A felhasználók új igényei, a szoftver termék megváltozott környezethez való igazítása, új technológiákra, más platformra való áttérés, az újrahasznosítható komponensek kinyerése, az idő során beépülő tervezési hiányosságok kiszűrése felvetik napjainkban a Reengineering igényét.

A Reengineering magában foglalja a rendszer megértését, visszafejtését egy olyan absztrakciós szintig, ahol az igényelt változásokat már végre lehet hajtani. A reengineering során a vizsgált

rendszer felépítésének megértése kulcsfontosságú. A gyakran hiányos dokumentáció miatt ez történhet a forráskódból kinyert, UML-modellhez hasonló, struktúramodell segítségével. Ez a modell egy objektum-orientált programnál a rendszer entitásait tartalmazza, amelyek lehetnek osztályok, metódusok, csomagok, alrendszerek, illetve a köztük lévő kapcsolatok: metódushívások, öröklések, tartalmazás stb.
A Karlsruhe-ban működő FZI (Forschungszentrum Informatik an der Universität Karlsruhe) intézetében fejlesztették ki a Goose eszközrendszert, amely a reengineering fázisai közül a modell kinyerését, előkészítését, vizualizációját és a szoftverben lévő tervezési problémák automatikus szűrését támogatja.

Már átlagos bonyolultságú (néhány ezer entitás, több tízezer kapcsolat) szoftver esetén is, a kinyert modell összes entitásának és relációjának egyidejű megjelenítése megnehezíti a megértést, nehezen tudunk a lényegi pontokra koncentrálni. Szükségünk van tehát a modell leegyszerűsítésére, absztrakciójára. Nagy szoftverek esetén ez elképzelhetetlen eszköztámogatás nélkül. Elkészült egy olyan eszköz, amely a kinyert struktúramodell egyszerűsítését teszi lehetővé egy parancsrendszer segítségével. A parancsok entitások és kapcsolatok különböző szempontok szerint történő beszúrását, törlését illetve csoportosítását végzik. A transzformációk egymásutánjával az eredeti rendszer leegyszerűsített nézeteit kapjuk, amelyek már egy magasabb absztrakciós szint entitásait és kapcsolatait tartalmazzák. Ezeket a transzformációkat akár egyetlen parancs-szkriptben is megadhatjuk az eszköznek.
A modell megjelenítésével még áttekinthetőbbé válik a rendszer felépítése, amely a Goose eszközkészlet vizualizációs programjával lehetséges.
A kidolgozott módszerek és az ezekre épülő eszközkészlet alkalmazhatóságát számos esettanulmány igazolja.

Veréb Krisztián

Obkejtum alapú keresési és indexelési technológia képadatbázisokhoz

A képek tárolása, de legfőképp azok adatbázisból történő visszakeresése nagyban különbözik a nem multimédiás jellegű egyéb adatok tárolásától és visszakeresésétől. Az újabb, objektum-relációs vagy teljesen objektum-orientált szemléletű adatbázisok terjedésével pedig a problémákra újabb megoldási lehetőségek jelentkeznek. Felmerül a kérdés, mit nevezünk képadatbázisnak. Természetesen léteznek, speciális képek tárolására szolgáló adatbázisszerű megoldások, de itt most nem ilyen speciális megoldásokkal foglalkozom, hanem olyan általános adatbázis-kezelőkben megvalósítható megoldásokkal, ahol felmerül nagyobb képadatbázisok létrehozásának kérdése (pl. Oracle9i ORDBMS). Általánosan elmondható, hogy egy adatbázis-kezelő rendszer jónak tekinthető, mikor az adatbázis a képek BLOB-kénti tárolásán felül már többet is nyújt. Például megoldja a képek méretezését, bináris le- és feltöltését stb.

A képi adatbázisokból történő képkinyerés alapsémája a következő: Adott egy adatbázis, mely képeket tartalmaz. Adott egy kérdező kép, kereső kép (query image), egy minta, és azt szeretnénk tudni, található-e az adatbázisban olyan kép, mely legjobban hasonlít a minta képre. Mivel azon képek száma, melyek feltehetően identikusak, megegyezőek a mintával meglehetősen kicsi, így valamilyen távolságfogalmat kell bevezetni. Ezek alapján tehát az adatbázisban az olyan keresések végezhetők el egy adott keresőképpel, mint például az identikus keresés (a

távolság 0), vagy az epszilon keresés (ahol a távolság kisebb mint egy adott epszilon), illetve az NN keresés (legközelebbi szomszéd a távolság alapján). A kereséshez valamilyen indexelési technikát lehet alkalmazni.

Az adatbázisokban tárolt képek indexelése azok tulajdonságvektorain alapszik. A vektorok leggyakrabban valós koordinátájú többdimenziós <v1,...,vd> alakú vektorok alakjában írhatók fel, ahol d a dimenziók száma. Adott illesztő algoritmusok és konkrét megvalósítások esetén természetesen a d rögzített. Az is belátható, hogy így egy adott illesztés esetén a keresett kép szignatúrája is a d dimenziós valós koordinátájú vektorok teréből adódik.

A multimédiás indexelési technikák két nagy csoportba oszthatók. Az első az adatpartíció indexelés, a másik pedig a térpartíció indexelés. Az első az adatok eloszlása alapján osztja fel a teret, a másik pedig előre meghatározott vonalak mentén osztja fel a teret, függetlenül az adatok előfordulásától. A térpartíciós indexelés nem tud túl hatékony lenni azon esetekben, mikor a képek közel azonosak, azaz az indexeik távolsága nem túl nagy, és egy nagyobb csoportba csoportosulva nem töltik ki az elméleti teret. Ilyen esetek elkerülése érdekében érdemesebb az adatpartíciós indexelést alkalmazni.

Az adatpartíció indexelés az R-fából származtatható, mely eredetileg kétdimenziós adatok indexelésére szolgált a GIS-ben. Később az R-fákat kiterjesztették többdimenziós adatokra is. De nagyon sok egyéb kiterjesztés is létezik, melyek mind azon alapulnak, hogy nem minden régiónak van ugyanakkora szerepe a visszakeresésekkor.

Az objektum-alapú adatpartíció indexelés az objektumok hierarchiában betöltött "szemantikájuk" szerinti indexelésen alapszik. Az indexelendő tulajdonságvektorok alakja a leggyakrabban egy többdimenziós vektor. Az indexelés lelke ezen vektorok valamilyen rendezési elv alapján történő rendezése a többdimenziós térben. Mivel a vektorok mérete is nagy elemszámú kép esetén már jelentőssé válik, így szükségessé válik a vektorok többdimenziós terének valamilyen elven történő felosztása, és ezen klasszifikáció segítségével többszintű indexszerkezet felépítése. Az első, és legfontosabb dolog, hogy az adatbázisban tárolni kívánt képek típusait meg kell határozni. Ez a tipizálás valójában egy hierarchikus osztályozás, mely a képek nem mérhető, ún. asszociatív tulajdonságain alapszik. Ez az osztályozás az adatbázistervezők feladata. A fontos, hogy ennek az adott képhez tartozó adott típusnak ismertnek kell lennie az adatbázisba történő beszúráskor, illetve a visszakereséskor. Mint említettem, ez a klasszifikáció valójában egy hierarchikus osztályozás, azaz az adatbázisban tárolható összes létező kép típusára fel kell készülni, és azok típusait egy öröklődési fában reprezentálni.

Ismeretes, hogy bármely kereső rendszer esetében az általános keresésektől sokkal hatékonyabbak, célravezetőbbek azok a keresések, mikor a keresési kritériumok némelyikét, vagy mindegyikét konkrétan meg lehet határozni. Egy, az objektum-hierarchiából származó típusfa típusaiba tartozó képeket tároló adatbázisban ilyen specializáció az, hogy meghatározható, a fa mely csomópontjához rendelt osztályba tartozik a kép. Mivel az öröklődési fa ISA kapcsoatokból áll, ha a gyökér szinten található a kép, akkor az bármelyik kép lehet az adatbázisból, viszont a fában haladva a levélelemek felé egyre pontosítható, mely képek tartoznak bele egy adott keresésbe, s melyek nem. (Ez végül is abból az OO szemléletből adódik, hogy egy gyermek mindig szerepelhet szülője helyett, hiszen örökli annak tulajdonságait.) Így tehát bevezethető egy olyan keresés, melynél a keresőkép típusa (osztálya) meghatározza a keresendő képek osztályait.

Ez a tipizált keresés, mely a fent említett objektum-hierarchián alapszik, és jól alkalmazható képi adatbázisokban, a már ismert identikus, epszilon, és NN keresések kiegészítéseként.

A SZEKCIÓ

2002. október 17.
14:30 – 16:30
Szekcióelnök: Markovits Péter

Molnár Balázs

Web szolgáltatások és Java objektumok Oracle környezetben

Az objektum orientált alkalmazások fejlesztésével, üzemeltetésével szemben a vállalati környezet speciális elvárásokat támaszt. Olyan környezetnek, ami ilyen alkalmazásások megbízható és kellő sebességű futtatását vállalja, a szabványok teljeskörű támogatása mellett ezen elvárásoknak is meg kell felelnie. Az Oracle alkalmazásszerver olyan környezet, amely megfelel ezen kritériumoknak. Az előadásban szó lesz a vállalati alkalmazások tranzakciós és perzisztens objektumainak kezeléséről, az alkalmazásfunkciók web szolgáltatásként történő publikációjának lehetőségeiről, illetve a futtatókörnyezet felügyeletéről is.

Petrohán Zsolt

Adatbázis alapú Java alkalmazások gyors fejlesztése üzleti objektum keretrendszerek segítségével (BC4J, UIX)

Az Oracle Business Components for Java
A Sun Microsystems azért hozta létre a J2EE-modelleket (blueprints), hogy segítsen a fejlesztőknek nagy teljesítményű J2EE alkalmazásokat készíteni. A modellek tervezési mintákat és sémákat adnak, melyek segítségével a fejlesztők általában sok programkódot írnak meg teljes körű e-üzleti alkalmazásaikhoz.
Az Oracle9*i* JDeveloper a Business Components for Java üzleti komponenstárral biztosít J2EE-keretrendszert a tervezési minták megvalósításához. Az üzleti komponenstár jelentősen növeli a fejlesztők hatékonyságát, mert megfelelően kezeli az objektum–relációs leképezést és a J2EE-architektúra különböző rétegei közti kapcsolatokat.

A Business Components for Java üzleti komponenstár deklaratív és kódközpontú eszközökkel segíti az üzleti feldolgozás végrehajtási logikájának definiálását, ugyanakkor gondoskodik az alapul szolgáló infrastruktúráról.

UIX

UIX keretrendszer gondoskodik a kifejlesztet üzleti logikánk igényes megjelenítésről.
XML-ben definiálhatjuk a megjelenítési felületet, ami JSP technológiára épül.

Markovits Péter

Objektumorientált adatbázis struktúrák és az XML

Az Oracle adatbázis kezelő relációs alapokon működő rendszer, SQL elérési felülettel. Ezeken az alapokon építkezve azonban már a fél évtizede felmerült az objektumorientált adatstruktúrák adatbázisban történő kezelésének szükségessége, mintegy ezzel megteremtve a hátteret és az adatelérési interfészt az objektum orientált fejlesztési környezetek számára.
Az Oracle objektum-relációs kiegészítése kezdetben Object Option néven választható önálló modulja volt az adatbázis kezelőnek, azonban évek elteltével integrált elemévé változott és kiteljesedtek benne mindazok a szolgáltatások amelyek valódi objektumorientált adatkezelést tesznek lehetővé. Természetesen ezt a szolgáltatást az Oracle úgy valósította meg, hogy átjárás és átlátást biztosít a hagyományos tisztán relációs struktúrák és az objektum-relációs szerkezetek között.
Az XML, mint adatleíró nyelv, megjelenésével az adatbázis kezelőket is új kihívás elé állította. Az XML, jellegénél fogva tetszőleges mélységben egymásba ágyazott, fa struktúrával jól leírható adatszerkezeteket tartalmaz. Ezeknek az adatoknak a feldolgozása és az adatbázis tábla szervezésű adatstruktúráinak való megfeleltetése a legkomolyabb feladat, amellyel az XML kapcsán a relációs adatbázis kezelőknek meg kell birkózniuk. Ebben és az XML beágyazott adatszerkezeteinek kezelésében nagy segítséget nyújthat az Oracle előbb említett objektum-relációs megközelítése.
Ezzel és a legújabb Oracle 9i Release 2 adatbázis verzió által bemutatott XMLDB adatbázis kiegészítéssel teljeskörűvé válik az objektumorientált, a hagyományos SQL alapú strukturált relációs és az XML-ben reprezentált dokumentum alapú világok közötti átjárás.

Petrohán Zsolt

Adatbázisok és folyamatok tervezése UML segítségével ORACLE környezetben

Mivel az UML egységes modellezőnyelv a szoftverfejlesztés modellezésének de facto szabványává vált, a JDeveloper teljes mértékben támogatni fogja az UML-t. A JDeveloper UML-támogatása fokozatosan kerül bevezetésre. A 9i verzió két UML-modellezőt tartalmaz: a Class Modelert és az Activity Modelert.

A JDeveloperben az UML-modellezés fő célja az e-üzleti alkalmazások fejlesztésének támogatása, kihasználva az Oracle9*i* alkalmazáskiszolgáló és az Oracle9*i* adatbázis-kezelő lehetőségeit és teljesítményét.

A Class Modeler az Oracle Business Components for Java üzleti komponenstár (BC4J) és a natív Java osztályok modellezését és integrált kódgenerálását támogatja, és szoros kapcsolatot biztosít a modell és a kód között.

Az Activity Modeler az e-üzleti alkalmazások integrálását segíti. A generálás az Oracle Advanced Queueing (AQ) és az Oracle Workflow felé történik, JMS eléréssel és XML üzenetekkel.

A JDeveloper modellezői támogatják a szabványos UML-t, amely a JDeveloperben speciális funkciókkal bővült. A modellezők az XMI-t is támogatják a más UML-eszközökkel való kapcsolat biztosítására.

B SZEKCIÓ

2002. október 16.
14:00 – 16:00
Szekcióelnök: Zsemlye Tamás

Zsemlye Tamás

Java 2 MicroEdition

Juhász Zoltán

Java, Jini és szolgáltatásorientált programozás

Az Internet egyre általánosabb elterjedésének és az elosztott rendszerek programozástechnikai fejlődésének köszönhető a szolgáltatás-centrikus szoftver fogalmának megjelenése. Az informatikai világ új kedvence a Web Service architektúra, mely lehetővé teszi web-alapú alkalmazások könnyebb együttműködését.

A Web Service architektúra egy adott megjelenési formája az általános szolgáltatás-orientált architektúrának (Service-Oriented Architecture: SOP). Az architektúrától felépítésétől azonban érdekesebb az a kérdés, hogy ilyen rendszerek programozása kíván-e egy újabb, más programfejlesztési stílust, paradigmát. A válasz erre a kérdésre igen.

Az előadás témája a szolgáltatás-orientált programozás (SOP) alapelveinek, tulajdonságainak, jellemzőinek és előnyeinek áttekintése. A szolgáltatás-orientált programozás lehetővé teszi elosztott rendszerek, alkalmazások gyorsabb integrációját a komponens technológia felhasználásával. A szolgáltatás specifikáció és implementáció szétválasztásával szélesebb körű újrafelhasználás valósulhat meg, valamint lehetővé válik ezen szolgáltatások egyszerű távoli elérése is.

A szolgáltatás-orientált programozás elveinek jobb megértése céljából az előadás során röviden ismertetésre kerül a Web Service architekúra, valamint a Jáva nyelv és a Jini technológia is. Megvizsgalom továbá a Jáva és Jini technológiák szerepét, tulajdonságait a szolgáltatás-orientált architektúrák és programozás terén, valamint összehasonlítom az így nyert SOA platform lehetőségeit a Web Service architektúráéval.

Darmai Gábor

Java web azolgáltatások és alkalmazásuk

A mai nagyvállalatok informatikai infrastruktúrája általában rendkívül bonyolult, sok tíz esetleg száz "egyedi" szoftver alkalmazás heterogén, rengeteg interfészt tartalmazó, átláthatatlan hálózatából épül fel. Ezek az alkalmazások legtöbb esetben valamilyen egyedi kommunikációs protokollal rendelkeznek, ezért minden egyes új kommunikációs csatorna implementálása a vállalat rengeteg pénzébe és idejébe kerül.

Sok esetben egy jól szervezett szerver konszolidáció (az alkalmazások "összeolvasztása") megoldást nyújt a fenti problémára, de mivel az alkalmazások többsége már önmagában is elég drága és bonyolult, ezeknek cseréje illetve újra implementálása szóba sem jöhet. A másik – jobb – megoldás az lehet, hogy a meglévő alkalmazásainkat ellátjuk egy szabványos kommunikációs réteggel. A szabványos réteg segítségével elkerülhetjük a kommunikációs interfészek számának óriási mértékű növekedését, hiszen minden alkalmazás minden más alkalmazással csak ezen az egyetlen szabvány interfészen fog kommunikálni.
Ez a szabványos réteg a Web szolgáltatás interfész, amely a CORBA, RMI és COM technológiák után várhatóan tényleges áttörést hoz az alkalmazásintegráció és az Internet alkalmazásának területén. A Web szolgáltatások technológia főleg az egyszerűségének köszönheti sikerét, illetve annak, hogy ez talán az egyetlen olyan informatikai szabvány, amelyet minden vezető szoftvergyártó támogat. Noha a Web szolgáltatásokat sokszor a Java nyelvvel kötik össze, tudnunk kell, hogy ez a technológia teljes mértékben platform és programozási nyelv független.
A Web szolgáltatások technológia nem definiál komponens modellt, csupán a kommunikáció mikéntjét írja elő. Ezzel biztosítja, hogy a kommunikációs interfész mögött elhelyezkedő alkalmazások belső tulajdonságai teljes egészében rejtve maradnak a külvilág elöl, így azok tetszőleges technológiával készülhetnek illetve tetszőlegesen módosíthatóak.
A Web szolgáltatások két fő felhasználási területe az alkalmazásintegráció és az Internet. Mind a két alkalmazási terület a Web szolgáltatások különböző előnyeit használja ki. Az alkalmazásintegráció esetében a legfontosabb tulajdonság a kommunikációs protokoll egyszerűsége és szabványossága. Az Interneten történő használata esetén az előbbi két előny mellett nagy szerepet kap a Web szolgáltatások önleíró tulajdonsága (WSDL), illetve a Web szolgáltatások regisztrációs adatbázisa, a UDDI címtár. A WSDL (Web Service Description Language) és a UDDI (Universal Description, Discovery and Integration) a Web szolgáltatások szintaktikai és szemantikai leírására szolgálnak, és lehetővé teszik a Web szolgáltatások automatikus felderítését és használatát.

Németh László

Java J2EE alapú alkalmazás integrálás

Az üzleti alkalmazások integrálása napjaink égető problémájává vált. Az informatikai rendszereknek rövid átfutási idővel kell gyorsan változó üzleti folyamatokat támogatni. A megvalósuló új szolgáltatásoknak a már használt vegyes alkalmazásokra és infrastruktúrára kell támaszkodni a beruházások védelme érdekében.
Vállalatiank legtöbbjénél az alkalmazások már kapcsolódnak egymáshoz. A kapcsolatok általában egy-egy alkalmazást kötnek össze, páronként változó módon. Könnyen belátható, hogy az így kialakult rendszer nem látható át, karbantarthatósága kritikus. A "spagetti" integrációban résztvevő rendszerek legapróbb módosítása is beláthatatlan, nem várt eredményeket hozhat.
Az előző megoldásnál sokkal tudatosabb az üzenettovábbításra alapozott message oriented middleware-ek használata. Ebben az esetben az alkalmazások már csak egy másik alkalmazáshoz, a middleware-hez csatlakoznak. A middleware feladata az üzenetek megfelelő továbbítása és esetleges transzformációja.
A gyorsan változó üzleti igényekre sem a spontán módon kialakuló "spagetti" integráció, sem a hagyományos üzenettovábbításra és konverzióra alapuló megoldás nem nyújt megfelelő választ.

A valós üzleti logika egyik esetben sem jelenik meg kielégítően, rejtve marad az alkalmazásokban, csatolókban és a rendszert használó emberek fejében. Gyakori jelenség, hogy az üzleti folyamatok az alkalmazások közti nem megfelelő kapcsolat miatt a szükségesnél jóval tovább tartanak, rontva a vállalatok hatékonyságát.
A korszerű, szabványos technológiákra alapuló, harmadik generációs integrációs middleware eszközök megfelelő megoldásokat kínálnak a hagyományos alkalmazások elérésére, az üzleti folyamatok leképezésére és végrehajtására, az integrált rendszeren kialakított kompozit alkalmazások elkészítésére.
Az előadás áttekinti a J2EE architektúrára alapozott integrációs megoldásokat, ismerteti a J2EE specifikáció alkalmazás integrációhoz szorosan kötődő részeit, vizsgálja a technológiai trendeket és a várható specifikációkat.

B SZEKCIÓ

2002. október 16.
16:30 – 18:30
Szekcióelnök: Zsemlye Tamás

Ertner Péter – Frigó József

Java J2EE alapú portál technológiák

A nagyvállalati Internet és Intranet portálok területén a J2EE alapú portál technológiák az utóbbi években egyeduralkodóvá váltak. A szoftver szállítók minden eddiginél teljesebb megoldás szállítására törekszenek, mit érdemes használni ezekből a termékekből, és mi a marketing fogás? A meglévő funkciók közül melyek azok, amelyek megléte és biztonságos működése minimális elvárásként megfogalmazható?
A piacon ezek kívül megjelentek alacsony költségvetésű vagy szabad szoftver alapokon készülő ingyenes megvalósítások is, ezek milyen mértékben elégítik ki az elvárásokat?
Az előadás azokat az elveket és technikákat mutatja be, amelyek segítségével a fejlesztés kézben tartható és az elkészült alkalmazás megfelel az elvárható igényeknek.
Szót ejtünk a J2EE portál technológiák felépítéséről, típusairól. Elemezzük az MVC fejlesztési technika (Model-View-Controller) alkalmazásának előnyeit, és megvizsgálunk konkrét megvalósításokat.
Említést teszünk azokról a problémákról, amelyek az ilyen alapon történő fejlesztéseket technológiai szempontból jellemzik. Az előadásban kitérünk az IQSOFT Rt. projektjeiben megtapasztalt buktatókra, javaslatokat teszünk ezek elkerülésére.

Fazekas Imre

Objektumok futásidejű kezelése

A folyamatos üzemelést igénylő számítógépes rendszerek számára szükséges olyan technológia megalkotása, mellyel a futás közbeni fejlesztőség megoldható. Ez OO környezetben objektumok együttesét jelenti, melyek kapcsolatban állnak, kommunikálnak egymással. Amikor valamilyen fejlesztést végzünk, bizonyos objektumok megváltoztatják viselkedésüket, új objektumok jelennek meg. Ezért a futásidejű szoftverfejlesztés mögött mindig objektumcsere áll. Dinamikus környezetet kell teremteni, ahhoz, hogy futás közben bizonyos objektumok viselkedése, adatmodellje változhasson. Az előadásomban olyan technológiát szeretnék bemutatni, mely biztosítja:

Tetszőleges alkalmazások esetében történő alkalmazhatóságot
Osztályok, interfészek, metódusok, adattagok definiálását és módosítását futásidőben
Perzisztens objektumok kezelését
Konzisztens és teljesítménybeli javulást eredményező működést

Egy szoftver rendszer esetében jogos igény a futás időben történő profiling elvégzése, vagy szoftverkomponensek analizálhatósága, strukturális és viselkedésbeli korrekciók végrehajtása az adott szoftver leállításának vagy teljesítménycsökkenésének előidézése nélkül (pl: szűk keresztmetszet keresés és korrekciók). Ideális esetben általunk definiált *Ágensek* és *Wrapper*-ek végeznék el az általunk definiált műveleteket vagy megoldást találnának általunk specifikált problémákra egy futó alkalmazásba beépülve oly módon, hogy ne legyen szükség különösebb programozói tevékenységre, vagy tudásra. Az ágensnek egy alkalmazás objektumai között kellene „mászkálgatni", és működő objektumokba beépülni, lehetőséget biztosítva a tényleges működés követésére, ellenőrzésére, fejlesztésére.
Olyan technológiát mutatok be, mely megoldásként szolgál a problémákra és lehetővé teszi:
tetszőleges alkalmazáson történő alkalmazhatóságot
Ágensek, Warepper-ek definiálását, koordinálását, irányítását futás időben
megbízható konzisztens működést a teljesítménycsökkenés minimalizálásával
teljes kontrollt és felügyelet mind a szoftver mind az ágensek felett
nagyon egyszerű felhasználói interfészen keresztüli végrehajtást

Boros Péter

Java alapú keretrendszer

A tudomány mai állása szerint kiválóan tudunk elosztott és komponens alapú rendszereket fejleszteni, de ez valahogy mégsem tűnik elegendőnek ahhoz, hogy megtörténjen az alapvető technológiai áttörés, amelyről ezzel kapcsolatban mindenki beszélt. Vajon mi okozhatja, hogy nem igazán sikerül a különböző technológiák nyújtotta előnyöket maradéktalanul kihasználni. Nos ezt többek között az alábbi két alap problémára lehet visszavezetni:

A fejlesztési technológiák és azok egymáshoz viszonyított szerepe nincs tisztázva és keretbe foglalva → Keretrendszer(ek), amely pontosítja a feladatok megoldására igénybe vehető technológiákat és azok használatának módját. A módszertanok által kiemelt fontosságúnak tartott architekturális tervezés fizikai megvalósítása lehet egy ilyen keretrendszer és biztosíthatja az architekturális vívmányok újra felhasználásának lehetőségét.
A modellezés orientált fejlesztés hiánya, amely szükségszerű, ha a fejlesztés során a modell csupán másodlagos dokumentáló és nem pedig vezérlő szerepet tölt be. → MDA (Model Driven Architecture). A modell a rendszer fejlesztés minden fázisában a mérnöki tevékenységek motorja marad. Ehhez természetesen nem csak a fejlesztés menetét kell átgondolni, hanem a modellről alkotott elképzeléseinket is. A generálás pedig a korábbi egyszerű értelmezéssel ellentétben komolyabb hangsúlyt és nehezebben megoldható feladatokat kap.
A keretrendszerek fejlesztése kettős feladat elé állítja a készítőit. Egyrészt biztosítani kell, hogy a keretrendszer önmagában megállja a helyét és lehetőség szerint részenként továbbfejleszthető legyen. Ez biztosíthatja, hogy a
Az előadásom során igyekszem bemutat az ezen a területen szerzett konkrét tapasztalatainkat, és bemutatni egy olyan keretrendszert, valamint modellezési filozófiát, amelytől hosszútávon a fent említett célok elérését reméljük.

Molnár István - Simon Géza

Model-View-Controller paradigma web-környezetben, J2EE alkalmazásokban

A Sun Java™ Blueprints dokumentumok a Model-View-Controller (MVC) tervezési paradigmát javasolják interaktív alkalmazások fejlesztéséhez. Az MVC értelmében egy interaktív program három fő részre bomlik: az alkalmazás modellje (*model*), melynek feladata az adatok reprezentációja és az üzleti logika megvalósítása; a nézetek (*view*), amelyek az adatokat különféle formákban és bontásokban prezentálják; illetve a felhasználói adatbevitelt fogadják; végül a vezérlő (*controller*), mely a kérések irányításával, és a program menetének vezérlésével, diszpécseri szerepet tölt be.
A különféle webes fejlesztő keretrendszerek mindegyike valamilyen módon az MVC paradigmát valósítja meg.

Az MVC elvet követve számos előnyhöz jutunk. Elkülöníthetők a tervezési jellemzők adat-perzisztencia, és -viselkedés; megjelenés és vezérlés területekre. Lecsökkenthetők az ismétlődő kóddarabok, egy helyre gyűjthetők a vezérlési szerkezetek, egyszerűsödnek a módosítások, és elkülöníthetők a különböző tudású, illetve szakirányú fejlesztők részterületei.

Előadásunkban részletezzük ezeket az előnyöket, majd az elv ismertetésén túl bemutatjuk az iparban leggyakrabban alkalmazott webes MVC megvalósításokat: Java Server Faces (JSF-Sun), Jakarta-Struts (Apache).

B SZEKCIÓ

2002. október 17.
9:00 – 13:00
Szekcióelnök: Tarnay Katalin

Anastasius Gavras (EURESCOM)

Cooling the hell out of distributed telecom applications' deployment

Papp Sándor

A nyílt szolgáltatás-átlépő kezdeményezés (OSGI) és az objektum-orientált környezet

1. Az OSGI kezdeményezés célja és eredete.
2. Az OSGI architektúrája.
3. Az OSGI és a middleware viszonya (Java VM, Java Embedded Server, UpnP)
4. Fogyasztói elektronikai rendszerek együttműködése objektum-orientált megközelítésben
5. A lakossági átlépő (Residential Gateway) működtetése nyílt, szétosztott technológiában.

Tarnay Katalin

Szolgáltatásfelfedező protokollok

Merre tart a WEB?
Szolgáltatásfelfedezés és - keresés
Protokollok osztályozása
Szolgáltatásfelfedezés infrastrukturája
WEB trader
Példa
Értékelés

Jaskó Szilárd

Szolgáltatásfelfedezés modellezése

Ahhoz, hogy a telekommunikációban áttörést érhessünk el teljesen újszerű gondolkodásmódot kell elültetni az emberek és főként a mérnökök szemléletmódjában. Egy ilyen újszerű „filozófiai" irányzat a szolgáltatásokra épülő kommunikáció. A számítógépes hálózatokon nem címekre (http, IP stb.), hanem szolgáltatásokra kell rákeresni és azok közül a

felhasználó a céljainak leginkább megfelelőt veheti igénybe. Ezen hálózatok heterogén rendszerbe olvasztása nagy munka, felelősség és egyúttal nagy lehetőség. A szolgáltatásfelfedezést két nézőpontból lehet modellezni az egyik a felhasználói oldal, a másik a technikai megvalósítás oldala. Akkor optimális a fejlesztés, ha ezt a két modellt össze tudjuk olvasztani, úgy hogy ne vagy csak minimálisan sérüljenek a modelleknek az erényei és ne növekedjenek a hibalehetőségek.

A gondolat megszületése után nagyon fontos, hogy kerülők nélkül le tudjuk „közérthető" formában írni a gondolatainkat. Ennek fontosságát azt hiszem nem kell taglalnom. Munkám során én is ezt a szisztémát követtem, amikor definiáltam a Bluetooth Szolgáltatás Felfedező Protokoll (SDP, Bluetooth Service Discovery Protocol) dinamikus működését. A felhasznált eszközök a következők voltak: CFSM (Communication Finite State Machine), Globális Állapotátmenet Gráf, SDL (Specification and Description Language), MSC (Message Sequence Chart), HMSC (High Message Sequence Chart). Ezek közül a módszerek közül hármat emelek ki részletesebben, mivel az SDL, MSC és a HMSC segítségével megkönnyíthetjük a tesztelési fázis előkészítését.

Az SDL segítségével a rendszerek dinamikus viselkedését lehet ábrázolni és vizsgálni. Természetesen egy rendszer dinamikus leírása nem olyan egyszerű feladat és legtöbbször több módszert kell ötvözni, hogy a leírás teljes legyen. Az SDL-nek egy nagy hibája a sok erénye mellett, hogy nehézkesen lehet ábrázolni vele a rendszer időbeni működését, a telekommunikációs alkalmazások esetén viszont ez egy súlyos fogyatékosság. Ebből kifolyólag szinte nélkülözhetetlen MSC-ben is ábrázolni a működést, hiszen ez az ábrázolási szisztéma direkt erre lett kitalálva. A HMSC-t az MSC-nek a kiterjesztett változata, általában nagy rendszerek esetén szokás alkalmazni.

A munka végeztével az SDP teljes működését leírtam és ezzel lehetővé tettem a rendszer további vizsgálatát illetve tesztelését.

Dulai Tibor – Muhi Dániel

Service Location Protokol (SLP)

Szolgáltatásfelfedezés
Az SLP architekturája
Szolgáltatások és üzenetek
Alkalmazások
Az elektronikus ügynök
Mobil rendszer
Értékelés

Muhi Dániel

Session Initiation Protocol (SIP)

SIP-re alapozott IP kommunikáció
SIP hálózat elemei
Funkciók
SIP szolgáltatások áttekintése
SIP alapú telefonia
Ujabb szolgáltatások
SIP mobilitás

B SZEKCIÓ

2002. október 17.
14:30 – 16:30
Szekcióelnök: Papp Sándor

Kollarics Róbert

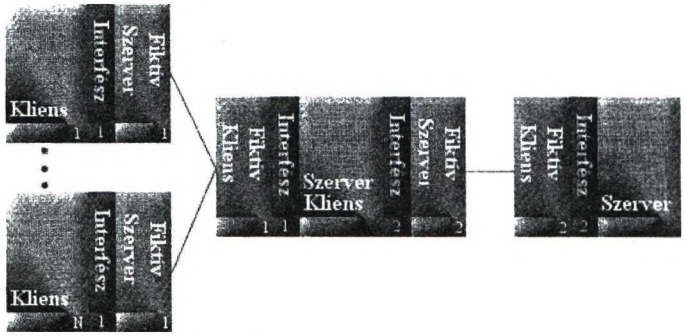
Hálózati Kommunikáció objektum-orientált megvalósítása

Az előadás a hálózati kommunikáció egy objektum orientált megvalósítását mutatja be, amely nyelv és protokoll független. A modell bemutatása egyben felvázol egy fejlesztési folyamatot, amelynek betartása esetén a választott protokollra épülő hálózati kommunikáció generálható.

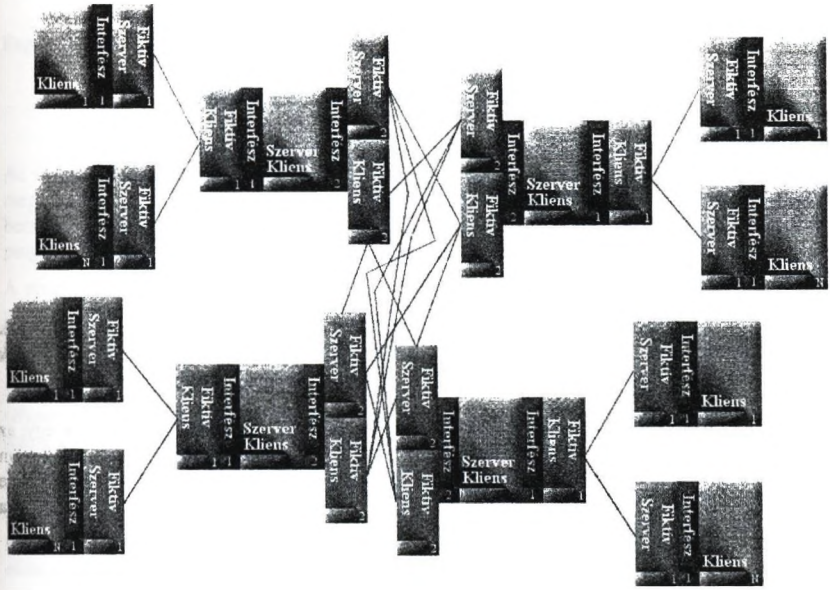
Ennek következtében a fejlesztőknek csak az alkalmazás logikájára kell összpontosítani és a kommunikációt megvalósító részeket pedig csak legenerálni.

Hálózati kommunikáció megvalósításának elemzése.
Egy általános modell felépítése.
A modell alkalmazásának részletezése.
A generikus részek megállapítása.
A fejlesztési folyamat meghatározása.
Egy Java-ban írt példaprogramon keresztül a technológia bemutatása.
TCP/IP, RMI, CORBA alapú hálózati kommunikáció generálása.

Az előadás további részében az előbb ismertetett technológia alkalmazásának bemutatása többrétegű és elosztott rendszerek esetén.



A három rétegű megvalósítás



Az n-rétegű megvalósítás

Fazakas Antal

OO technológiák a protokoll tesztelésében

Konformancia tesztelés
A TTCN tesztnyelv
A tesztnyelv fejlődése objektum orientált irányba
TTCN-3 jellemzői
GFT, Graphical Format of TTCN
Inline operátorok
Tesztek osztályozása
Értékelés

Csontos Péter

Új programozási paradigmák a láthatáron

Bevezetés (motiváció)

OOP és az abból eredő dolgok (pl. UML hiányosságai)
Milyen irányban célszerű továbbmozdulni?

Aspektus-orientált programozás

Az aspektus-orientált programozás egy olyan programozási paradigma, amely a kilencvenes évek közepén született meg. Manapság a programozási nyelvekkel kapcsolatos kuatások középpontjában áll, és minden bizonnyal a közeljövõben széleskörûen el fog terjedni.

Az aspektusok olyan funkcionálisan összefüggõ programrészletek gyûjteménye, amelyek egy, az AOP lehetõségeket nélkülözõ nyelven írt programban a forráskód különbözõ részein szétszórtan találhatók meg. A paradigma célja, hogy ezen funkcionálisan összefüggõ részek egy helyre gyûjtésével a programtermék (kód és/vagy terv) bonyolultságát csökkentse, a karbantarthatóságát, olvashatóságát javítsa.

Főbb koncepciók és eszközök:
- AspectJ (XEROX) és más nyelvre adaptált változatai
- HyperJ (IBM)
- Egyéb: DemeterJ, Mozart, ComposeJ, ConcernJ, JADE

Intencionális programozás

Történelmi háttér
Célok
Alapkoncepció
Az ökológiai párhuzam
Megvalósulás jelenlegi állapota
Kapcsolat más területekhez (OOP, AOP, generatív programozás, tervezési minták)
Hasonló projektek:
- Sun Jackpot
- IBM Eclipse
- Az UML fejlődése
    - action semantics
    - complete UML based software platform
    - The Reuse Initiative
    - ágensek: JacZone WayPointer

Espák Miklós

Új paradigmák a szoftverfejlesztésben

A szoftverfejlesztési módszerek és a nyelvi modellek ma is folyamatos fejlődés alatt állnak. Az utóbbi évtizedben jelentős hangsúlyt kapott számos olyan elv, amely a programfejlesztést megkönnyíti. Az irányzatok rendkívül sokfélék. Az előadás keretében azt próbáljuk meg bemutatni, melyek azok az újonnan felismert szemléletmódok és technikák, amelyek a programfejlesztés jövőjét nagy valószínűséggel meg fogják határozni.

A program szervezésére vonatkozó általános elvek közül a talán a legfontosabb a nézőpontok szerinti szétválasztás (*Separation of Concerns*). Számos publikációban rámutattak már arra, hogy az osztályszintű absztrakció nem megfelelő bizonyos típusú nézőpontok hatékony elkülönítésére. Főként akkor, ha ezek ún. *crosscutting* tulajdonsággal rendelkeznek. Az ilyen nézőpontok elkülönítésére több technikát bemutattak, közöttük az adaptív programozást (*AP*), a hipertereket (*Hyperspaces*) és az aspektusorientált programozást (*AOP*). A program szervezésére szolgáló egy másik alapelv Demeter törvénye. E törvény betartásának alapvető hatása, hogy osztályaink lazán kapcsolódnak egymáshoz, és azok implementációs részletei be lesznek zárva. Ennek következtében egy osztály működésének megértéséhez nem szükséges számos másik osztály részletes megismerése. Az adaptív programozás alapjait is Demeter törvénye jelenti. Emellett a rendszeres kódátszervezéseket (refactoring) megkívánó extrém programozás (XP) szintén elősegíti ennek az elvnek a betartását.

Hosszú életciklusú szoftvereknél szintén fontos szempont, hogy minél könnyebben fejleszthetőek legyenek, minél kevésbé legyenek érzékenyek alapvető, akár a programtervet érintő változtatásokra. Azokon a területeken, ahol a szoftverrel szemben támasztott követelmények dinamikusan változnak, az XP használata jelenti a legjobb megoldást. Adaptív programozás esetén nem kell tervezéskor osztályaink minden részletét rögzítenünk, így az osztályok felépítéséhez kötődő követelményi változásokhoz később azonnal igazítható lesz a program. A deklaratív metaprogramozás célkitűzése, hogy összekapcsolja szoftverfejlesztés magasabb fázisait a megvalósítási szinttel. Ezzel automatizálhatóvá válik a különböző fázisok együttes fejlesztése (*co-evolution*). Lehetővé válik terv generálása az implementációból és viszont, valamint magasszintű tervezési feladatok elvégzése (pl. kódátszervezések, generatív programozás, stb.).

Több olyan irányzat is megjelent, amely az objektumorientált modell egyes gyengéire, hiányosságaira mutat rá. A személyiségjegyek (*Personalities*) segítségével elkerülhetjük az adatok és a viselkedés öröklődésének együtt járásából eredő problémákat. Másik felismerés, hogy az osztályhierarchiabeli viszonyokat egyszerre több helyen is implementálni kell, így a kód redundánssá válik, és a fölösleges részletektől nagyobb, nehezebben áttekinthető és módosítható lesz. Ezen segít pl. az AP és Composition Filters nevű aspektusorientált eszköz is.

C SZEKCIÓ

2002. október 16.
14:00 – 16:00
Szekcióelnök: Csink László

Hajdara Szabolcs - Kozma László – Ugron Balázs

Párhuzamos programok szintézise és objektum elvű kiterjesztése

Párhuzamos programok szintézisére különböző matematikai eszközök állnak rendelkezésünkre. Ilyen eszközök például a klasszikus elsőrendű logika, a temporális vagy más néven időlogikák és a különböző fajta algebrák stb.
A konkurens programok olyan speciális párhuzamos programok, amelyekben nemdeterminisztikus szekvenciális programok, az ún. folyamatok működnek együtt egy közös cél megvalósítása érdekében. Ezek a folyamatok vagy közös, osztott változókon keresztül cserélnek egymással információt, vagy minden folyamat saját, helyi adattárolóval rendelkezik és üzenetváltásokkal tartják a kapcsolatot egymással.
A szintézis során először specifikáljuk a feladatot, majd ennek segítségével futtatható kódot generálunk.
Az objektum elvű kiterjesztés első lépéseként megadjuk a rendszerben szereplő osztályok specifikációját. Ezt oly módon tesszük meg, hogy leírjuk az egyes objektumok belső életciklusát. Ehhez a temporális logika eszközeit használjuk fel. Ezen módszer korlátai sokkal szűkebbek, mint a folyamatok specifikációja során, és a kódgenerálás is sokkal kevésbé általános.
A kódgenerálás problémája a nagy komplexitású feladatok esetén csak nehezen, vagy egyáltalán nem oldható meg.

Gyapay Szilvia – Pataricza András

UML modellek Petri-háló alapú erőforrás allokációja és ütemezése

Az informatikai rendszerek növekvő komplexitásával egyre gyakrabban merül fel az igény erőforrás-allokációs és ütemezési problémák hatékony megoldására. Azonban a valós életből vett problémák leírása vagy túl matematikai ahhoz, hogy ipari környezetben használható legyen, vagy túl terjengős. Ezen problémák elkerülése végett célszerű a feladat valamely szabványos, félformális leíró nyelven való megfogalmazása, melyet esetünkben az *Unified Modeling Language (UML)* nyelv *General Resource Model (GRM)* profile-ja testesít meg [1]. Ebből a leírásból modelltranszformáció segítségével automatikusan generált *Petri-háló* modell már alkalmas a matematikai vizsgálatok elvégzésére.

A Petri-háló alapú modellezés előnye kidolgozott matematikai háttere mellett, hogy támogatja a párhuzamos folyamatok modellezését, valamint grafikus megjelenése és szemantikája révén kitűnően bővíthető, azaz alkalmas erőforrás-allokációs és ütemezési problémák leírására. Ugyanakkor, a valós problémákat leíró Petri-hálók extrém mérete könnyen állapottér-

robbanáshoz vezethet az analízis során, hiszen ilyenkor gyakran a teljes állapottér kiterítése szükséges.

A Veszprémi Egyetem kutatócsoportja a termelésoptimalizálás területén használatos produkciós hálók területén egy ***matematikailag erősen rokon*** problémakörben fundamentális eredményeket ért el. A *produkciós hálóknál* szükséges és elégséges feltételrendszert adtak a teljes megoldástér egzakt meghatározására, valamint hatékony algoritmusokat dolgoztak ki az e fölött értelmezett függvények optimalizálására [2].

A két iskola eredményeinek összekapcsolásával (a Petri-hálók modellező erejének és a produkciós hálók szintézisét végző hatékony algoritmusok kombinálásával [3]) kidolgozott hatékony analízis és optimalizálási módszerek automatikus keretrendszert biztosítanak az UML-alapú erőforrás-allokációs és ütemezési problémák megoldására.

Ferenc Rudolf – Beszédes Árpád – Gyimóthy Tibor

Columbus: C++ tervrekonstrukciós eszköz és séma

A korszerű objektum orientált szoftvertermékek méretének és komplexitásának gyors növekedése azt a természetes igényt eredményezte, hogy a nagyméretű rendszerek különböző részei közötti kapcsolatokat könnyebben értsük meg. Ezen összefüggések azonosítása után a rendszer fejlesztőjének szüksége lehet arra, hogy átszervezze a rendszert úgy, hogy az jobban megfeleljen az egyéni követelményeknek. A szegedi Mesterséges Intelligencia Tanszéki Kutatócsoport egy, a létező rendszerek modellezésére szolgáló (reverse engineering) eszközt fejlesztett ki, melynek a neve Columbus.

A Columbus projekt fő célkitűzése az, hogy egy megbízható analizáló eszközt szolgáltasson, amely különböző információkat ismer fel és gyűjt ki nagy C++ rendszerekből. Ezen információk egy általunk megtervezett sémának felelnek meg, ami leírja az összegyűjtött információ struktúráját a konkrét fizikai reprezentációtól függetlenül. A sémát törekszünk nemzetközi konferenciákon általánosan elfogadtatni, hogy ez által megkönnyítsük az információ cserét különböző szoftver fejlesztő, vizualizáló, analizáló és egyéb programok között. A séma szerint tárolt információból könnyedén előállíthatók például UML osztály-diagramok, különböző dokumentációk és metrikák.

A program képes komplex projektek elemzésére és a séma szerint kigyűjtött adatok különböző formátumokba történő exportálására, valamint egyéni igényeknek megfelelően könnyedén bővíthető. Az eszköz számos nemzetközi konferencián elismerést nyert és kutatási/oktatási célokra ingyenesen elérhető.

Pataricza András – Varró Dániel

UML modellek automatikus transzformációi

Az 1990-es évek objektum-orientált metodikáinak eklektikája után napjainkban a Unified Modeling Language (UML) már egyértelműen az objektum-orientált rendszertervezés szabványos, grafikus modellezési nyelvének tekinthető. E nyelv fejlődésében mérföldkővé válhat az UML szabvány (fejlesztés alatt álló) 2.0-ás változata, mely céljául tűzi ki a nyelv matematikailag precízzé tételét és finomhangolhatóságát domain specifikus alkalmazások esetén. E célból az UML 1.x-es verzióira jellemző egysíkú (monolit) nyelvstruktúrát egy központi metamodellezési mag köré épült nyelvcsalád váltaná fel, ahol az egyes résznyelvek (diagramok) szemantikája külön-külön definiálható.

Egy nyelvcsaládon alapuló UML architektúra esetén létfontosságú az UML modellek transzformációnak tervezhetősége. Egyrészről egy UML modell fejlesztése, finomítása során a tervező maga is (modellen belüli) transzformációk segítségével építi a rendszertervet. További transzformációk szükségesek azonban (i) az egyes diagramok közötti transzformációkat specifikálásához, illetve, (ii) az egyes UML diagramok szemantikáját definiáló matematikai modellekre történő leképezéshez.

Előadásomban ismertetem a (fejlesztés alatt álló) **VIATRA** (Visual Automated model TRAnsformations) modelltranszformációs rendszert [1-3], melynek legfőbb céljai a következők:

- **modellek** és modell családok (metamodellek) **grafikus leírása**, mely jól illeszkedik az ipari szabványokhoz (például UML, MOF, XMI)
- **modellek transzformációinak matematikai precizitású, de grafikus leírása**
- a **transzformáció implementációjának automatikus generálása** az előbbi grafikus specifikációból kiindulva
- egy **automatizált transzformációs gép**, mely a bemeneti modellből és a transzformáció programjából automatikusan állítja elő a kimeneti modellt
- egy **automatikus visszavetítési mechanizmus**, mely a kimeneti matematikai modellen elvégzett analízis eredményeit az eredeti modllen belül is láthatóvá teszi.
- a **modelltranszformációk formális helyesség- és teljességbizonyítása**.

C SZEKCIÓ

2002. október 16.
16:30 – 18:30
Szekcióelnök: Csink László

Huszerl Gábor – Majzik István – Pap Zsigmond – Pataricza András – Petri Dániel – Varró Dániel

Keretrendszerek nagymegbízhatóságú, biztonságkritikus rendszerek fejlesztéséhez és teszteléséhez

Az UML a vizuális programozás területén áttörést jelentő modellező nyelv, amelyet az OMG ipari szabványban is rögzített. Az UML definíció összefoglalja és integrálja a korszerű szoftverfejlesztési paradigmákat (objektumorientáció, modularizálás, tervezési minták használata, stb.). Az automatikus kódgenerálással végződő vizuális programozás az emberi hibaforrások jelentős részét kiküszöböli az implementáció végső lépéséből, azonban ez sem oldja meg a tervezés során a rendszertechnikai hibák elkerülését. A vizuális programozás során is lehet szintaktikusan helyes, de szemantikusan hibás modelleket építeni. E szemantikus hibák azonban olyan komplexek is lehetnek, hogy azokat nagy biztonsággal csak matematikai analízis útján lehet felfedni.

A fellépő hibák fő okai nemzetközi statisztikák szerint a következők:

- *A specifikáció hibái*: A komplex rendszerek specifikációja teljességének és ellentmondásmentességének biztosítása hagyományos módszerekkel már nem lehetséges.
- A vezérlő elektronika állandó, katasztrofális fizikai meghibásodásainak aránya a technológiai fejlődés következtében visszaszorul, viszont a rendszert érő zajok, zavarok hatására bekövetkező *tranziens hibák hányada növekszik.*
- A működésbeli meghibásodások jelentős része *szoftvertervezési hibák* hatására következik be, ezeken belül dominálnak a *szoftver rendszertechnikai hibák.*

Előadásunk során egy jelenleg is folyó kutatásba adunk bepillantást [1,2], mely egy, a beágyazott- vagy reaktív-rendszerek UML bázisú tervezéshez illeszkedő, kiegészítő (add-on) matematikai modellanalízis programrendszert, és a szolgáltatásbiztonságot garantáló módszertan kidolgozását tűzte ki célul, melyek:

- a specifikáció teljességét és ellentmondás-mentességét vizsgálják,
- a rendszer megkívánt működéseinek helyességét hibák fellépése esetében is bizonyítják, a környezetben illetve a vezérlőrendszerben bekövetkezett hibák hatását vizsgálva,
- a rendszeren belüli folyamatok logikai vezérlésének helyességét igazolják.

A fentieket a rendszer nyílt formában valósítja meg olyan módon, hogy:

- a rendszer egyes komponenseinél a szabványos adatcsere-formátumok kizárólagos alkalmazásával lehetővé teszi a tervező és ellenőrző eszközök széles választékának használatát, valamint
- a felhasználó igényeknek megfelelően újabb ellenőrzési metodikák beiktatását

Szabolcsi Judit

A D programozási nyelv

Miért van szükség még egy új programozási nyelvre?

A szoftveripar hosszú utat tett meg a C nyelv létrehozása óta (a C már 30 éves). A C++ nyelv sok új tulajdonsággal ruházta fel a C-t, de megtartotta a vele való kompatibilitást, ezzel az eredeti tervek gyengeségeit is továbbvitte. Emellett az új struktúrák régihez való illesztése (az örökölt C programkódok újrafelhasználhatóságának biztosítása miatt) elbonyolította a C++-t.

A standard C leírása közel 500 oldal, a standard C++-é pedig közel 750 oldal. A C++ programozók csak a nyelv egy-egy részét, „szigetét" használják, így hiába hordozható a kód fordítók között, ha programozó és programozó között nincs sok átfedés.

Ezek mellett a modern fordítási technológia is elérte azt a szintet, amikor már mellőzhetőek a register kulcsszó, a preprocesszor és a többi, primitív compilerek által igényelt optimalizálás.

A régi, 16 bites gépek helyett a D nyelv minimum 32 bites memória címzést kezel és kis módosítással átalakítható a 64 bites architektúrákra is. (Ezért nem szükséges a közeli/távoli mutatókkal való trükközés.)

A nyelv tervezésekor lefektetett alapelvek, célkitűzések

- A C és C++ programozók számára könnyen elsajátítható legyen. (Habár a D nem kompatibilis a C/C++-szal, de a C++ –ból könnyű D-re átírni a kódot.)
- Egy gyakorlott programozó nagyobb erőfeszítés nélkül megérthesse a nyelvet.
- A D compiler-t könnyű legyen megvalósítani, mivel a tokenizálás független a szintakszistól, és a szemantikus analízis mindkét előzőtől.
- A C-ben mindenütt jelenlévő pointerek helyettesítése ill. elrejtése, a hibák kiküszöbölése végett.
- Design by Contract és a unit tesztelés alkalmazása a megbízható programok előállításáért.

A nyelv új tulajdonságai címszavakban

- szemétgyűjtés (garbage collection)
- szerződések, elő- és utófeltételek (Design by Contract megvalósítása, amely Bertrand Meyer ötlete)
- nem szükséges elődeklaráció
- modulok
- valódi típusdefiníciók
- a C/C++-ban megszokottnál érthetőbb tömbdeklarációk
- beépített sztringkezelő függvények
- bit típusú változók
- asszociatív tömbök
- szinkronizálás

- egyszerű öröklődésű osztályok (többszörös öröklődés és nem virtuális tagfüggvények nincsenek)
- minden objektumnak a heap-en foglal helyet
- In, Out és Inout paraméterek
- inline assembler
- verziók
- a hibakeresés és a robosztus technikák támogatása
- fordítási és futási idejű ellenőrzés
- unit tesztelés

Kiss Gergely – Orosz József – Pintér Márton – László Zoltán – Thomas Genssler

C# programok metaszintű manipulációja

Mára az objektum-orientált fejlesztés "de facto" szabvánnyá nőtte ki magát a szoftverfejlesztésben. Az utóbbi néhány évben azonban meg kellett tapasztalnunk a korlátait is. Példaként említhetjük a szoftveradaptáció, a felhasználói követelményekhez való folyamatos alkalmazkodás nehézkességét, vagy azt, hogy az objektumokban gyakran különböző viselkedések keverednek össze.

Ezekre a problémákra keres megoldást számos új módszer és paradigma, köztük a refaktorálás (refactoring), a generikus (más néven generatív) programozás, a személyiségjegyek (personalities), a metaprogramozás, az adaptív és az aspektus-orientált programozás. Az említett módszerek azonban a forráskódot már egy, a puszta szövegnél magasabb (meta) szinten közelítik meg. Így olyan eszközöket igényelnek, amelyek lehetővé teszik a forráskód meta szintű feldolgozását. Ilyen szoftverek már léteznek C++-hoz, JAVA-hoz, azonban még nem készültek el a viszonylag új - ámbár villámgyorsan terjedő - C# nyelvhez. Ezért tűztük ki célul egy, a C# nyelven íródott programok szintaktikai és szemantikai analízisét, valamint metaszintű manipulációját lehetővé tevő könyvtár elkészítését.

Előadásunkban röviden áttekintjük és jellemezzük a jelenleg elérhető legkorszerűbb technológiákat, majd bemutatjuk az általunk elkészített keretrendszert, a **RECODER.C#**-ot, amely a JAVA nyelvhez íródott RECODER könyvtáron alapul. A **RECODER.C#** képes a C# osztályok szintaktikus és részleges szemantikai analízisére, valamint e statikus információkból elérhetővé teszi az osztályok metastruktúráját (névterek, típusok, osztályok, metódusok, mezők, attribútumok). Az elemzés mellett a programcsomag azt is lehetővé teszi, hogy a metastruktúrából ismét kódot generáljunk, így lehetőséget teremt a programok statikus transzformációjára.

Az előadásban bemutatjuk a **RECODER.C#** működését, és kitérünk azokra a konstrukciókra, ahol a C# kód analízisa nehézséget jelent. Az előadás végén pedig ízelítőt adunk a könyvtár lehetséges alkalmazásaiból: röviden szó lesz a refaktorálásról, a tervezési minták (design patterns) automatizált alkalmazásáról, valamint ennek fordítottjáról - a tervezési minta és a negatív tervezési minta (antipattern) keresésről. Foglalkozunk még a JAVA programok C# nyelvre fordításának lehetőségével, illetve ennek fordítottjával is. Mindezeket megpróbáljuk a gyakorlatban is bemutatni.

Albert István – Balássy György

A .NET Framework és a C# nyelv objektum-orientált szemszögből

Az előadás objektum orientált szemszögből mutatja be a C# nyelvet, a Microsoft legkorszerűbb, komponens orientált programozási nyelvét. Ennek a futótűzként terjedő programozási nyelvnek már a tervezésekor figyelembe vették az OOP paradigmáit, ezért aki azokban járatos, gyorsan megismerkedhet az új nyelvvel – az előadás is az objektum orientált világ fogalmain keresztül vezeti be a hallgatókat a .NET első számú nyelvébe.
Az előadás második részében a .NET Framework-öt helyezzük a középpontba. Bemutatjuk funkcióit, szolgáltatásait, megnézzük, hogy a Microsoft az implementáció során hogyan alkalmazta a C# lehetőségeit és az objektum orientált programozási technika eszközeit. Látni fogjuk, hogy az objektum orientáltság hogyan határozta meg a .NET Framework architektúráját. A Framewörk talán egyik legfontosabb tulajdonsága, hogy számos ponton kibővíthető, az előadásban megvilágítunk néhányat ezen pontok közül.

C SZEKCIÓ

2002. október 17.
9:00 – 13:00
Szekcióelnök: Bertalan Gábor

Scott Ambler (Ronin International Inc.)

Enterprise RUP

Gulácsi Ferenc

Nyílt forráskódú fejlesztés kereskedelmi alkalmazásoknál. A Progress objektumorientált fejlesztési projektje.

A Progress Software Corporation 2000 decemberében indította el webre alapozott, nyílt forráskódú fejlesztői hálózatát.
A Progress egyedi megoldása azzal függ össze, hogy a cégfilozófiában a fejlesztő eszközök és az adatbázis alkalmazások értékesítése az alaptevékenység. Alkalmazást – eltérően a konkurensektől – nem fejlesztenek, ezt több ezer fejlesztő-partnerükre bízzák. Nem csoda, hogy a legnagyobb vállalati alkalmazás fejlesztő tábora a Progressnek van (megjegyzés: a Microsoft értelmezésünk szerint nem ide soroltatik, hiszen az ERP, CRM stb. alkalmazások előállítói elsősorban nem Visual Basicben fejlesztenek).

A speciális filozófiára és a nagy fejlesztő táborra, valamint a több millió felhasználóra alapozta a Progress nyílt forráskódú fejlesztési projektet, amelyet Progress Open Source Software Exchange rövidítéséből "POSSE" névre keresztelt. (www.possenet.org)

Ezzel a Progress hármas célt szeretne elérni:

1. A késztermékek, azaz a szoftverek a lehető legpontosabban illeszkedjenek a végfelhasználó igényeihez, aki saját maga is részt vehet a munkában, hatással lehet a „végtermékre".
2. A nyílt forráskódú fejlesztés révén jobb, gyorsabb, és lényegesen megbízhatóbb fejlesztő környezetek jönnek létre, összehasonlítva a hagyományos, „zárt" fejlesztői modellekkel.
3. A negyedik generációs Progress fejlesztői nyelv, illetve fejlesztő környezet – amely sok szempontból kiemelkedik a hagyományos negyedik generációs nyelvek közül – szélesebb közösséghez jut, kölcsönös előnyöket nyújtva ezáltal minden résztvevő partner számára.

Hasonlóan a linuxhoz, amelynek népszerűségét az ingyenessége mellett az átláthatóságából adódó megbízhatósága adja, a Progress alkalmazások felé is lényegesen nagyobb bizalom nyilvánulhat meg azokból a szektorokból (pl. kormányzati szféra), amelyeknél fontos az átláthatóság, a biztonság.

Az előadásban bemutatjuk a Progress fejlesztő környezetet és azt a hátteret, amelyre építve már meg is jelent az első kereskedelmi forgalomban kapható termék, a Progress Dynamics.

Vég Csaba

Alkalmazásfejlesztés – a gyakorlatban

A követelmények felmérésére, illetve azok informatikai alapszerkezetének, a „szakterületi modell" összeállítására napjainkra az objektumorientált szemlélet már kialakult módszerekkel és gyakorlattal rendelkezik. Továbbra is probléma azonban, hogy ezt az alapszerkezetet milyen módon ültethetjük át informatikai közegbe és technológiákra. Mivel egy alkalmazást többféleképpen is le lehet programozni, ezért itt túlságosan nagy szerepet kap az emberi tényező, így más fejlesztők teljesen más módon és minőségben is megoldhatják ugyanazt a problémát.

Az alkalmazásfejlesztés gyakorlatának alapkérdése, hogy hogyan biztosítható a fejlesztés folyamatossága és menete. A tapasztalatok szerint a fejlesztésnek a tervezést és programozást felölelő gyakorlati része két, teljesen különböző feladatkörre bontható, az architekturális és az alkalmazási programozó tevékenységeire.

Az arhitekturális fejlesztő (régebben: „rendszerprogramozó") feladata az alkalmazás technológiai alapszerkezetének a kialakítása, amelyet a Rational Unified Process alaparchitektúrának

(„baseline-architecture") nevez. Fontos, hogy az architekturális fejlesztő („architect") a rendszert, mint egységes egészt tekintse.

Az alkalmazási programozó feladata az alkalmazásnak a követelményekben megfogalmazott részleteinek a leprogramozása, neki tehát elsősorban a részletekre kell koncentrálni.

Az architekturális fejlesztés eredményét, az alaparchitektúrát alapvetően két részre bonthatjuk.

- Az együttesen alkalmazott létező vagy újonnan fejlesztett *frameworkök* (osztálykönyvtárak) a tipikus működtető segédosztályok gyűjteménye.
- A *tervezési megoldások* gyűjteménye a megvalósítás egyes területeire adnak recepteket, például így oldjuk meg egy felhasználói felület vezérlésének a leprogramozását, így oldjunk meg az adatok adatbázisból való feltöltését és visszaírását, stb.

Az előadás egy egyszerű példán keresztül bemutat az egyes technikai kérdésekre is egy-egy megoldást.

Kókai Tamás – Csiszár Tibor

Roleorientált szoftverfejlesztés a gyakorlatban

Jelenleg egy nagy mennyiségű adatot kezelô és gyakran változó szoftver rendszer fejlesztésével foglalkozunk. A feladatot kezdetben az objektum-orientált paradigma és az UML segítségével szerettük volna megoldani, azonban gyakran ütköztünk olyan problémákba, amiket nehezen, vagy egyáltalán nem tudtunk megoldani. A fenti tapasztalatok hatására kezdtünk el kidolgozni egy új szoftverfejlesztési módszert, melyben megpróbáltuk egyesíteni a relációs adatbáziskezelés és az OOP elônyeit.
A cikkünk bevezetôjében bemutatjuk a feladatosztály jellemzôit és a jelenlegi módszerek - mások által is tapasztalt - hiányosságait.
Jelen keretek között a módszer részletes ismertetése helyett a figyelem felkeltése az elsôdleges célunk. Ennek megfelelôen írásunk további része áttekintô jellegel mutatja be a szoftverfejlesztés legfontosabb lépéseit a modellalkotástól kezdve, query-k definiálásán keresztül a view-k tervezésig. Említést teszünk a Client program működéséről, valamint bemutatunk egy konkrét példát is.
Az utolsó fejezet a módszer elônyeit, illetve hátrányait foglalja össze, valamint kitérünk a még elôttünk álló feladatokra is.

Vég Csaba

„Információs környezetek" a szervezésben és tervezésben

Az információs környezetek (context) elvével egy olyan egyszerű alapszerkezetet határozhatunk meg, amely keretébe egyszerűen és áttekinthetően helyezhetjük el alkalmazásunk elemeit. A környezetek olyan, „aggregátum"-jellegű objektumok, amelyek a bennük elhelyezkedő objektumok közötti viszonyokat kezelik. A környezetek egyben „közeg" szerepűek is, így egyik feladatuk az objektumok közötti hatások átvitele és vezérlése. A környezetek ugyanakkor az

objektumok között különleges jellemzőkkel rendelkeznek, például speciális módon keverednek a származtatási és a tartalmazási viszonyok.

Az objektumorientált szemléletben egy tipikus probléma például a következő: a Pont és az Egyenes osztályok esetén hova helyezzük el azt a metódust, amely azt vizsgálja, hogy egy pont illeszkedik-e az egyenesre? A metódust a Pont vagy az Egyenes osztályban is definiálhatjuk, de statikus métódusként egy utility osztályban is elhelyezhetjük. Hol helyezzünk el egy Kör és egy Egyenes metszéspontjait meghatározó metódust? Másik példánkban egy játékprogram egy szobájában elhelyezkedik egy bomba, amely felrobbanása sérülést okozhat a játékoson. Hogyan tervezzük meg ebben az esetben az interakciót?

Az információs környezetek elvét alkalmazva ekkor célszerű megkeresnünk az objektumok környezetét. Síkidomok esetén ez maga a Sík lesz, így esetünkben célszerű statikus metódusként elhelyezni az illeszkedés, illetve a metszéspontok számítását. A játékprogramban a bomba és a játékos közös környezete a Szoba, így a bomba a Szobának kell, hogy jelentse a felrobbanását, a Szoba felelőssége lesz a hatást átvinni a játékosokra, így pl. egy varázslat alatti szoba el is nyelheti a káros hatást.

Az „információs környezetek" elvének egyes technikai elemei részben már megjelentek egyes programozási nyelvekben, illetve technológiákban, pl.: a JavaScript kliens-oldali böngésző-objektumai általában elérik a külső objektumot, a Java nyelv „belső osztályai" a külső osztály környezetében helyezkednek el, így elérik azt, illetve közvetlenül hivatkozhatnak azok attribútumaira és műveleteire, a Java BeanContext-ek a Bean-ek hierarchikus környezetei, stb.

A környezetek néhány jellemzője a következő:

- A környezetek a csomag- („modul-") hierarchiákhoz hasonlóak, de változókkal és műveletekkel („szolgáltatásokkal") is rendelkeznek;
- a környezetek általában hierarchikus, ritkább esetben „dag" szerkezetet alkotnak,
- az alkörnyezet eléri a tartalmazó „szülő" környezet(ek)et.
- a környezetek viszik át, egyben meg is szűrhetik az egyenrangú objektumok közötti hatásokat, ill. értelmezhetik azok egymáshoz való viszonyait,
- a környezeteknek lehetnek aktuális értékeik, pl. az aktuálisan kezelt üzleti objektumok
- a környezetek rendelkezhetnek a külső környezet aktuális értékeivel és szolgáltatásaival, ill. átdefiniálhatják azokat

Bognár Zsolt

Kritikus alkalmazások fejlesztése OO CASE eszközökkel

Az előadás célja, hogy ismertesse a kritikus alkalmazások közös jellemzőit, és a fejlesztés során felmerülő általános problémákat.

Kritikus alkalmazásoknak azokat a szoftvereket tekintjük, ahol a hibás működésből fakadó veszteségek elviselhetetlenek a szervezetünk számára. Ezek a hibák okozhatnak mind anyagi, erkölcsi veszteségeket és legrosszabb esetben még emberéleteket is követelhetnek.

A kritikus alkalmazások fejlesztése elképzelhetetlen alapos elemzés, és a részletekig menő tervezés nélkül. Az alkalmazás speciális jellege miatt szigorú elvárásokat támasztunk többek között a szoftvertervező-eszközzel szemben.

A kritikus alkalmazásokra jellemző, hogy valamelyik szabványnak kell megfelelni. Ilyenek pl. katonai, légi-közlekedési, valós idejű alkalmazások, de ide sorolható egy pénzintézet folyamatos működését biztosító rendszer is. Az ilyen fejlesztésre alkalmas rendszereknek könnyen kell alkalmazkodni az egyes szektorok előírásaihoz (DO178B, MSZ ISO/IEC 12207, V-Modell, AQAP ...). Ehhez célszerű, ha az eszköz nagymértékben testreszabható, de az a legjobb, ha maga az eszköz egy meta-eszköz.

C SZEKCIÓ

2002. október 17.
14:30 – 16:30
Szekcióelnök: Kollár Lajos

Kázmér Adorján

Többrétegű alkalmazás fejlesztése OO módszerrel

Bevezető

Az előadás a Pénzügyi Szervezetek Állami Felügyelete számára készített háromrétegű rendszer megvalósításakor alkalmazott objektum orientált szoftverfejlesztési módszert mutatja be.

A projekt kereteinek ismertetése

- A projekt céljának, felépítésének ismertetése: A PSZÁF tevékenysége, a rendszer célkitűzései, néhány általános követelmény kiemelése.
- A projekt során megvalósult két legnagyobb alrendszer sajátosságainak ismertetése
- Az alkalmazott eszközök, technológiák rövid bemutatása

A szoftverfejlesztési folyamat ismertetése

Jelen előadás elsősorban a szoftverfejlesztési ciklus tervezési, fejlesztési szakaszaira koncentrálva mutatja be a célként kitűzött feladat megoldására kialakított módszer alapelveit.

A kialakított tervezési szabvány ismertetése

A Software Through Pictures (STP) case eszközben kialakított tervezési módszer, folyamat ismertetése.

- Háromrétegű alkalmazás tervezési alapelvei
- Modularitás, funkcionális dekomponálás alapelvei
- A testreszabott OO tervezési folyamat bemutatása

Kódgenerálás

Az ACD kódgenerátor gyakorlati alkalmazása.

- Rendszerterv és programkód konzisztenciájának fenntartása.
- Milyen feladatok elvégzésére célszerű a generátort alkalmazni? Mennyire legyen „okos" a generátor?

Fejlesztés

- Mit tehet a programozó és mit nem?
- A fejlesztés szabályozásáról

Az eredményekről

A kialakított módszer használhatóságára vonatkozó tapasztalatok.

Dobán Orsolya – PatariczaAndrás

Szoftverfejlesztési Folyamatok Optimalizálása

Kihasználva az eddigi kutatási eredményeket, melyek szerint az UML nem csak a végtermék fokozatosan finomodó leírására, hanem a választott projektmenedzsment életciklus modellje, valamint a fejlesztési környezet (például háttér infrastruktúra, fejlesztők ismeretei, stb.) szabatos specifikálására is alkalmas, célunk, hogy összekapcsoljuk a szoftverfejlesztési projekt költségét meghatározó valamennyi tényezőt (céltermék, technológia, környezet, stb.) egyetlen egységes UML modellé, azokat a modell különböző moduljaiként implementálva.

- A *termékmodell* elemi komponenseihez a költségbecslési faktorok értékét (pl. funkcionális bonyolultság vagy kódhossz) hozzárendelve már automatikusan felépíthető a megvalósítandó célrendszer műszaki szempontból teljes költségmodellje.
- A korábbiak szerint önálló UML modulként lehet megfogalmazni az adott *fejlesztői környezet és folyamat* költségbecslés szempontjából releváns modelljét, kibővítve az adott költségbecslési metodika által alkalmazott elemi költségfaktorokkal.

Kihasználva azt, hogy az UML modell formális jellegű szabatossága miatt mód van abból közvetlenül matematikai modell származtatására, több tervezési alternatíva esetén lehetőség adódik azok költségmodelljének automatikus generálására. Ezen matematikai problématérben az optimumot ismert algoritmusokkal megkeresve támogatható a költség szempontjából optimális műszaki megoldás kiválasztása.

Az utóbbi két évtizedben **költségbecslő modellek** tucatja született meg. Ezek már a tervezés igen korai fázisától kezdve, az első durva rendszerspecifikáció megalkotása után lehetővé teszik a költségek és a munkaráfordítás becslését. A költségbecslő modellek közül a gyakorlatban az amerikai COCOMO II. modell az egyik legelterjedtebb.

Az UML-ben végzett tervezés természetéből adódóan a fokozatos finomítás módszerét követi. Ezen fokozatosan finomított, tehát a tervezés előrehaladtával egyre részletesebb diagramokból kiindulva a fejlesztés során egyre pontosabb költségbecslések végezhetőek. Ezen előny kihasználásaként a célunk a rendszerszintű **költségbecslések automatizálása**, ezáltal a becslések okozta plusz ráfordítások elkerülése.

A szoftverfejlesztési projektek egyik leggyakoribb problémája a fejlesztési folyamat során felmerülő, a kiindulásként felhasznált rendszer**specifikáció változás**a. Összességében jelenleg csak rendkívül szubjektív becslések adhatók arra, hogy akár a specifikáció változtatásából, akár a későbbi igények kielégítéséből milyen mértékű költség-, illetve munkaterhelés adódik.
Célunk egy olyan általános metodika kidolgozása, mely lehetővé teszi a specifikációbeli változások gyors követését, költségkihatásának elfogadható pontosságú becslését.

Célunk továbbá a költségbecslésre alapozva egy olyan **optimalizáló eljárás** készítése, mely lehetővé teszi a projekt tárgyi és humán erőforrásainak optimális allokálását, valamint a felmerülő szoftverarchitektúrák közötti optimális választást, ésszerű minimumra szorítva így a fejlesztési időt.

Dolla Gábor

Elektronikus dokumentum küldése bárkinek

Az előadás egy konkrét kérdéskört (elektronikus számla küldése) elemezve keres általános megoldást cégek közötti elektronikus dokumentumok küldésére.
Business registry-k áttekintése (UDDI, ebxml)
Transport protokollok áttekintése (SOAP, ebxml)
Elektronikus dokumentum szabványok áttekintése (RosettaNet, XBRL, OFX, stb)

Az előadás célja a világban zajló folyamatok megismertetése.
Az előadás másik célja, egy olyan közösség megszervezése, amelyik:
- előmozdítja a fenti szabványokon alapuló magyarországi B2B folyamatokat
- megteremti a hazai B2B szabványokat
- lobbizik a nyílt szabványokon és nyílt arhitektúrákon alapuló elektronikus dokumentumküldésért (pl elektronikus adóbevallás)

Benkő Tamás – Fokt Attila

Információ integrálás logikai alapokon

Az előadás bemutatja a *SILK* rendszert, egy információ-integrálást támogató eszköz-készletet.
A jelenleg használatos integrációs technológiák alkalmazása különféle korlátozásokat von maga után. Az üzleti folyamat *(workflow)* alapú integrálás nagyon jól alkalmazható néhány

kitüntetett folyamat esetén, ugyanakkor merevnek tűnik, mert a folyamatkör mindenkoribővítése további fejlesztési lépést, újabb programozási feladatot jelent. Az *adattárházak* építése útján végzett integrálás akkor megfelelő, ha nem feltétlenül van szükség a forrásokból származó adatokhoz történő azonnali, közvetlen hozzáférésre, hiszen egy ilyen rendszer az *off-line* működésből adódóan lassan reagál a változásokra. Az adattárház működtetéséhez továbbá jelentős infrastruktúra is szükséges (adattárak, nagy sávszélességű kommunikációs vonalak, folyamatos migrálás).

Ezzel szemben a vállalati adatintegrálási technológia *(enterprise information integration)* közvetlen kapcsolatot biztosít különböző típusú adatforrásokhoz, bárhol, bármilyen alkalmazás esetén, egy szabványos adatlekérdezési nyelv segítségével, biztosítva a rugalmas adathozzáférést és -feldolgozást.

A SILK rendszer egy ilyen, ismeretkezelésen alapuló eszköz-készlet, amelynek segítségévela heterogén madatforrások egységes rendszerré integrálása hatékonyabbá tehető. *Modellalapú* megközelítésmódjával lehetővé teszi mind a heterogén információforrások dinamikuslekérdezését *(mediálás)*, mind azok homogénebb formájúra és tartalmúra történő átalakítását*(integrálás)*.

A SILK rendszer szabványos ipari megközelítésekre és ajánlásokra épít. Modellezésre statikus *UML* leírásokat és *OCL*-ben megadott korlátokat *(constraint)* használunk. Modellek cseréjére az *XMI* formátumot választottuk. Adatlekérdezés céljára egy *SQL* jellegű, *OCL*-en alapuló nyelv használható. A SILK rendszer adatforrások széles skáláját kezeli: közvetlenül képes lekérdezni relációs adatbázisokat, részben struktúrált adatokat (mint az *XML*), továbbá helyi alkalmazások vagy web-szolgáltatások formájában elérhető információforrásokat.

A SILK központi része egy *modelltár*, ami egyrészt az adatforrásokra és az adatkapcsolatokra vonatkozó metaadatokból, másrészt pedig az alkalmazási környezet és a végfelhasználók *adatnézeteiből* és az ezekhez kapcsolódóan összeállítható lekérdezésekből tevődik össze. A rendszer eszközöket nyújt a modelltár feltöltésére, a modellek ellenőrzésére, hasonlóság-elemzésére és egyesítésére. Lehetőséget nyújt továbbá az információforrások lekérdezésére . különböző absztrakciós szinten lévő (adatforrás- vagy fogalmi) modelleken keresztül. Az előállt modellek más integrációs módszerek (mint pl. adattárház vagy üzleti folyamat alapú technológiák) segítésére is felhasználhatók.

A SILK logikai alapokon működő szoftver, a rendszer belső moduljai *Prolog* használatával készültek (következtetés, elemzés, lekérdezés-optimalizálás). A rendszerhez készült grafikus kezelőfelületet és az adatforrásokhoz történő kapcsolódást *Java* környezetben implementáltuk.

Az eszközkészlet a SILK projekt keretében az Európai Unió IST 5. keretprogramjának támogatásával, az IQSOFT Rt. koordinálásával készült.

Záró Plenáris Előadás

Henry S. Thompson

Schemas, Infosets and Objects: The Web Architecture of the Future Starts Today

A core requirement placed on the design of the W3C XML Schema effort was that it bring relevant key principles of object-oriented languages to bear on the document structure definition problem. In doing so, we also made it possible to understand and exploit a new way of thinking about the relationship of HTTP streams, XML documents, OO languages, OO data models and the real world with which applications interact. In this talk I'll explain all this, and draw some conclusions about how to make the best use of XML in designing forward-looking applications today. Oh, and of course it's all really about Web Services

SZPONZOROK


ORACLE
9i
UNBREAKABLE


SOFT


Sun
microsystems


SYSDATA
a Siemens Company


TRIAD

CÉGISMERTETŐ

Az Oracle Corporation (Nasdaq: ORCL) a világ második legnagyobb szoftvervállalata, az internetes kereskedelmet és a webes alkalmazásokat támogató csúcstechnológiájú, globális e-business megoldások vezető szállítója. Éves árbevétele meghaladja a 9 milliárd dollárt. Internetes platformja, különböző eszközei és internetes alkalmazásai, valamint a hozzájuk kapcsolódó tanácsadási, oktatási és támogató szolgáltatások 145 országban állnak rendelkezésre.

A vállalat központja a kaliforniai Redwood Shoresban található. Az Oracle az első olyan szoftvercég, amely teljes termékválasztékára kifejlesztette és bevezette a 100 százalékban internet alapú vállalati szoftvereket, az adatbázistól és a kiszolgálótól a vállalati ügyviteli alkalmazásokig, valamint az alkalmazásfejlesztési és döntéstámogató eszközökig. Egyedül az Oracle képes az egész világra kiterjedő elektronikus üzletviteli biztosító, teljes körű vállalati megoldásokat megvalósítani, amelyek az ügyfélkapcsolatok kezeléséről (front office) a vállalatirányítási alkalmazásokon (back office) keresztül a platform infrastruktúrájáig mindenre kiterjednek.

Az Oracle webhelyének címe (URL):
http://www.oracle.com

Alapítás éve: 1977
Alkalmazottak száma: Összesen: 41 000
Árbevétel: 2002-es pénzügyi év: 9,7 mrd USD

Egyre több vállalat tér át az elektronikus üzletvitelre (e-business), és számukra az Oracle internetes alkalmazásai gazdaságos megoldást kínálnak a piaci lehetőségek kiszélesítésére, az üzleti folyamatok hatékonyságának növelésére, továbbá új ügyfelek szerzésére és megtartására. A drága és rugalmatlan kliens-szerver architektúrát az internet egyszerűségével felváltó cégek innovatív alkalmazások széles választékát állíthatják rendszerbe, amelyek közönséges webböngészővel is elérhetők.

Az Oracle az egyedüli megoldásszállító, amely az e-business termékek teljes skáláját kínálja:

- *az internetes feldolgozást támogató platformot, amelyen létrehozhatók és rendszerbe állíthatók a weben alapuló megoldások,*
- *az internetes használó vállalati alkalmazások teljes választékát és*
- *szaktanácsadási szolgáltatásokat az e-business stratégia kialakítására, valamint az e-business alkalmazások megtervezésére, teszt szabására és/vagy megvalósítására.*

Az Oracle 1993-tól lányvállalata, az Oracle Hungary Kft. révén van jelen a magyar piacon. A siker egyik biztosítéka, hogy az Oracle Hungary Kft. következetesen alkalmazza az Európai Üzleti Kiválóság Modellt (EFQM), amely segíti a stratégiai és az üzleti tervezést és rendelkezik minden olyan elemmel, amely fontos a cég igazán eredményes működéséhez. Ennek elismeréseképpen a hazai informatikában elsőként kapott Nemzeti Minőségi Díjat.

ORACLE®

ORACLE HUNGARY 1124 Budapest, Alkotás u. 17-19.
Telefon: 224-1700, fax: 214-0070, www.oracle.com

IQSOFT Rt. 1135 Budapest Csata u. 8., Tel:236-6400, Fax: 236-6464,
WEB: www.iqsoft.hu, e- mail:info@iqsoft.hu

Az Információs Rendszerek Építője

Az IQSOFT Rt. 1990-ben alakult. Az 1990-es 30 főről mára a dolgozók létszáma közel 100 főre bővült és további 40 - 50 fős, kiképzett szakértői csapat vonható be a fejlesztési feladatok megvalósításába. 2001-ben árbevételünk meghaladta az 1, 7 milliárd Ft-ot.

Az IQSOFT 1999-ben csatlakozott a KFKI Számítástechnikai Csoporthoz. A KFKI Csoporton belül az IQSOFT az alkalmazások integrációjára és egyedi alkalmazás fejlesztések megvalósítására szakosodott cég.

Az IQSOFT Rt., a vezető szállítók és szabad forráskódú szervezetek eszközeivel az adott feladathoz, költségvetéshez és vállalati információs környezethez legjobban illeszkedő megoldásokat kínál ügyfeleinek. Az IQSOFT a hazai és nemzetközi trendek ismeretében kiemelt jelentőséget tulajdonít szakmai munkájában a portálépítő és alkalmazásintegrációs tevékenységnek és az ehhez szorosan kapcsolódó vállalati/intézményi internetes és mobil kommunikációs környezet kialakításának.

A feladatok megoldásához tovább bővítettük az internet/intranet alapú környezetek kialakítását támogató, világszínvonalú alkalmazások kínálatát. Az IQSOFT képviseli Magyarországon a BEA, a Documentum és az Autonomy cég termékeit.

- **A BEA WebLogic termékeket portálok, kritikus internet alapú integrált környezetek és többrétegű alkalmazások építéséhez,**
- **a Documentum web és elektronikus tartalomkezelést biztosító termékeit, valamint**
- **az Autonomy nyelvfüggetlen, koncepció megközelítésű, automatikusan kategorizáló, a tartalom alapján kereső szoftver komponenseit a vállalati portálok és tudáskezelő rendszerek kialakításához és üzemeltetéséhez ajánljuk.**

Kiemelt partnerenként működünk együtt az IBM-el és az Oracle-el. Portfoliónkban változatlanul kiemelt jelentőségű a teljes szoftverfejlesztési életciklust támogató Rational eszközcsalád.

Az IQSOFT megalakulása óta a hazai szoftverpiac meghatározó tényezője. E pozíciót elsősorban szakértelmével vívta ki magának. A társaság sikereihez a korszerű technológiák alkalmazása is jelentősen hozzájárult. Az IQSOFT büszke arra, hogy nevéhez fűződik az Oracle adatbáziskezelő magyarországi elterjesztése, az objektumorientált szoftver technológia valamint a Java alapú rendszerek üzleti célú felhasználásának hazai bevezetése. Az IQSOFT az egyedi alkalmazásfejlesztés mellett kész megoldásokat kínál, ezek közé tartozik az OLIB könyvtári és tudáskezelő rendszer, a saját fejlesztésű, a befektetési alapok portfolió-kezelését támogató IQ*FMS integrált értékpapír nyilvántartó rendszer valamint a speciálisan a bűnüldözéssel kapcsolatos feladatok elvégzését támogató i2 termékcsalád.

Az IQSOFT szerződéses munkáiban a legkorszerűbb technológiák (pl. objektumorientált, CORBA, Java, web szolgáltatások) felhasználásával vállalkozik az informatikai feladatok elvégzésére. A cég nemzetközi K+F projektekben való aktív részvétele lehetővé teszi, hogy az IQSOFT - a számítástechnika fejlődési irányait nyomon követve - a magyar felhasználók számára is jól hasznosítható eredményeket közvetítsen, új megoldásokat ajánljon.

Szolgáltatásaink között szerepelnek: IT stratégia tanácsadás, üzleti folyamatok felmérése és újratervezése, követelményelemzés, rendszertervezés, egyedi alkalmazások építése, vállalaton belüli és vállalatok közötti alkalmazás integráció, tartalomkezelés, tudáskezelés valamint funkcionális és teljesítménytesztelés, illetve a kiemelt jelentőségű oktatási tevékenység.

Az IQSOFT az ISO 1998-ban teljes szakmai tevékenységére megkapta az ISO9001 minőségbiztosítási tanúsítványt, amelyet 2001-ben az ISO 9001:2000 szabvány szerint újítottunk meg.

KFKI
GROUP

A SUN ÉS A HÁLÓZATI SZÁMÍTÁSTECHNIKA TÖRTÉNETE

A Sun Microsystems-t 1982-es megalakulása óta egyazon jövőkép - **"A hálózat maga a számítógép"** - hajtotta előre jelenlegi pozíciójáig. A vállalatot számítástechnikai oktatás fellegvárának számító Stanford Egyetem három diákja alapította, innen származik neve: **Stanford University Network.** Mára a cég a nagyvállalati hardverek, szoftverek és szolgáltatások vezető szállítójává vált szerte a világon. Termékeinek fejlesztése során - fennállása óta - mindvégig arra törekedett, hogy biztosítsa ügyfelei számára a hálózati informatika nyújtotta előnyöket, nyílt, rugalmas és méretezhető, a szervezetek információs igényeit maradéktalanul megoldani képes rendszereket hozzon létre. Az évek során a Sun termékskálája folyamatosan bővült s napjainkra megtalálhatók benne a munkaállomásoktól kezdve a munkacsoport szintű, a nagyvállalati, mainframe szintű szervereken és adatközpont megoldásokon és vékony klienseken át a hálózati háttértároló rendszerek csakúgy, mint a Sun ONE szoftverportfólió, mely magában foglalja például a biztonságos Solaris operációs rendszert, a portálszervert, a címtárszervert, elektronikus kereskedelmi alkalmazásokat, internetes kommunikációt segítő szoftvereket, integrációs szervert, a JAVA programozási nyelvet, valamint az arra épülő fejlesztőeszközöket.

A SUN MAGYARORSZÁGON

A Sun 1992 őszétől képviselteti magát Magyarországon. 1995 őszétől a közép-európai képviseleti iroda mellett megkezdte működését a Sun Microsystems Magyarország Kft. A magyarországi irodában összesen 55 munkatárs látja el a kereskedelmi, rendszermérnöki, marketing és szerviz feladatokat.
A cég szervizszolgáltató részlege 1996-tól a Sun Microsystems Magyarország Kft. részeként, önálló egységként működik és nyújt különböző szolgáltatásokat a felhasználók részére.

A cég nemzetközi hálózatának, illetve a rendszerintegrációval és -támogatással foglalkozó cégekkel kialakított jó kapcsolatoknak köszönhetően a Sun teljes szolgáltatáscsomagot nyújt a felhasználóknak. Partnereinkkel közösen olyan megoldásokat biztosítunk, amelyek az üzleti tevékenység minden részletét összehangolják. A Sun által kínált "testre szabott számítástechnika" jelentős költségmegtakarítást eredményez, mivel az új rendszerek bevezetése mellett a már meglévő eszközök is megtarthatók, s ez az út vezet a befektetések megtérülésének (ROI) maximalizásához.

A vállalat értékesítési csatorna politikájában hazánkban is a közvetett - tehát a partnereken keresztül megvalósuló - értékesítési modell a meghatározó.

Szolgáltatások

A Sun teljes körű tanácsadási, oktatási és professzionális szervizszolgáltatásokat biztosít.
Vásárlóink számára testreszabott ajánlatokat kínálunk a termékek teljes életciklusán átívelő hatékonyság és a termelékenység maximalizálása érdekében.

Professional Services
Bárhogyan is változzanak az Ön igényei, a Sun Professional Services segít tisztázni az üzleti jövőképet, valamint világos technológiai célokká fordítja azt.

Világszínvonalú tanácsadóink rendelkeznek azokkal a speciális ismeretekkel, amelyek az internet- vagy intranet-megoldások sikeres bevezetéséhez, valamint a nyílt számítógépes rendszerekre való átálláshoz szükségesek. Segítünk megtervezni és kialakítani a vállalati informatikai környezet szilárd alapjául szolgáló egyedi megoldásokat - beleértve az ERP-infrastruktúra, a Java-platform és Internet-alapú megoldásokat.

Sun Technikai Központ
A Sun Microsystems Magyarország Kft. hivatalos viszonteladói által szállított Sun rendszerek műszaki kiszolgálását az Enterprise Services magyarországi részlege látja el. Szolgáltatásai két csoportba sorolhatók; egyrészt közreműködik a szállítandó rendszer installálásában, másrészt ellátja a szállított szoftver és hardver elemek garanciális és garanciaidőn túli szervizét.

Sun Educational Services
A Sun számítógépekhez, illetve a Solaris operációs rendszerhez kapcsolódó oktatást a **Sun Educational Services** magyarországi képviselete szervezi, és a Component Soft Kft., a Sun Microsystems Magyarország Kft. hivatalos oktató központja végzi. Az 1994. évi indulás óta több mint 1000 hallgató fejezte be sikeresen a különböző kurzusokat. Jelentős részüket egy-egy fontos informatikai beruházás alkalmával nagy vállalatok, intézmények delegálták. A hallgatók visszajelzései azt mutatják, hogy az elsajátított ismeretek jelentősen bővítették tudásukat, és segítették a mindennapok problémáinak gyors megoldását.

SYSDATA
a Siemens Company

SYSDATA
A Siemens Company

A Sysdata a Siemens program- és rendszerfejlesztéssel foglalkozó csoportjának – Program and System Engineering (PSE) Group - magyarországi leányvállalata.
A Siemens a 90-es évek elején Közép-Európában hozta létre azt az önálló cégekből álló hálózatot, amelyben mára nyolc országban több mint 5000 fejlesztőmérnök dolgozik, és amely az Egyesült Államokban és Németországban képviseletet működtet. Kompetenciánk a PSE-ben a szoftverfejlesztés köré épül, tevékenységünk a rendszer- és termékfejlesztésen keresztül a tanácsadási, rendszerintegrációs és oktatási feladatok elvégzéséig terjed.
A PSE alapvető feladata a Siemens más ágazatainak támogatása magas színvonalú szoftverfejlesztői kapacitás, illetve megbízható informatikai háttér biztosításával. Meghatározó megrendelőink a Siemens konszern ágazatai, melyeken keresztül megoldásaink különböző rendszerek elemeiként a világpiacra jutnak. Siemens termékek részeként nálunk készült szoftvereket használnak világszerte a gazdasági élet különböző szegmenseiben, az információ és kommunikáció világában, ipari rendszerekben, az energiagazdálkodásban, közlekedésben, orvostechnikában, létesítménytechnikában, valamint az űrkutatásban.
Magyarországon a Siemens részeként a Sysdata 1993-as alapítása óta képviseli a PSE-t, ennek köszönhetően a nemzetközi hálózatban felhalmozódott tudás, valamint a felépített erőforrások a magyar vállalkozások rendelkezésére állnak. A Siemens Nemzeti Vállalatcsoporthoz tartozó cégekkel együtt fellépve ügyfeleink részére szoftverfejlesztést, informatikai tanácsadást, rendszerintegrációt, üzemeltetési és oktatási feladatokat végzünk.
Munkatársaink száma ma Magyarországon meghaladja az 500 főt, mellyel a Sysdata a legnagyobb fejlesztőközpont. Cégünk két irodával rendelkezik, 2000 óta a budapesti központ mellett Szegeden is működtet fejlesztőbázist.

Triad Számítástechnikai és Szolgáltató Kft.
Budapest, XI. Karolina út 65. , Telefon: 209-2748, 372-3124, fax: 209-0931
Web: www.triad.hu, e-mail: info@triad.hu

TRIAD

A Triad Számítástechnikai és Szolgáltató Kft. jogelődje, az azonos nevű GMK, 1986-ban alakult meg. A Kft. 1991-ben alakult és lényegében átvette a GMK tevékenységét.

Az 1986-os alakulást követően a cég elsősorban konzultációs és szoftverfejlesztési területen tevékenykedett.
1990-es egyesült államokbeli piackutató munkánkat követően a tevékenység kiegészült kereskedelmi és értéknövelt viszonteladói területtel.

Három legfontosabb tevékenységi körünk:

- Számítógéppel támogatott, teljes életciklust lefedő szoftver- és rendszerfejlesztés, az ehhez kapcsolódó eszközök forgalmazása, technológia tanácsadás, oktatás.
- Nagyteljesítményű nyomtatók, digitális másolók forgalmazása és az ehhez kapcsolódó szolgáltatások nyújtása.
- Elektronikus számlamegjelenítés és fizetési technológia szállítása, a hozzá kapcsolódó fejlesztés és konzultáció.

Bár a három tevékenység látszólag távol áll egymástól, mindhárom azonos piaci magatartást, hasonló kereskedelmi és marketing tevékenységet igényel és végső soron lényegében azonos felhasználói kör számára ajánlható.
Cégünk specialitása, hogy valamennyi tevékenységünk felső kategóriájú termékekhez kapcsolódik és így ügyfeleink elsősorban nagyvállalatok, pénzintézeti szolgáltatók, államigazgatási szervek, biztosítók, stb. Valamennyi tevékenységünk során alapvető követelmény ügyfeleink megbízható, pontos kiszolgálása.
A Magyarországon kezdetben ismeretlen technológiák és beszállítók ellenére jelenleg a legnevesebb szállítók termékeivel szemben is sikerrel tudunk fellépni.
Kedvező árainkon túlmenően gépeink és szoftvereink megbízhatósága, szolgáltatásaink minősége (rendelkezésre állás, szerviz, folyamatos alkatrész és fogyóeszköz ellátás, kiképzés, tanácsadás, speciális igények kielégítése, stb.) is segített abban, hogy a legmagasabb követelményeknek is meg tudjunk felelni.

BM Pilis Üdülőszálló

Szalonna-sütő
Turista szálló
Konferencia-terem
Szalonna-sütő
Focipálya
Sörözö Bowling
Üdülő
Uszoda
Szauna Presszó Recepció
Ön itt áll
Teniszpályák
Röpl. p.
Pilis Hotel
Étterem
Presszó
Recepció
Játszótér
Sport p.
Személyzeti szálló
P

Budapest felöl BEJÁRAT Téry Ödön Út FŐBEJÁRAT Dobogókö felé