BELGRADE COLLEGE OF COMPUTER SCIENCES

Informacione i Internet tehnologije

Seminarski rad:
Programerski alati

Predmetni nastavnik:
Goran Radić

Student:
Mitrović Nenad 38/04

Datum predaje
21.12.2004.

SADRŽAJ

UVOD

Sa napretkom informacionih tehnologija i povećanjem zahteva tržišta, pojavila se potreba za mnogo bržim razvojem aplikacija. U te svrhe se razvila čitava grana softverske industrije koja se bavi kreiranjem isključivo programerskih alata. Neke od njih su kreirale velike kompanije kao što su Microsoft i Borland, dok su drugi alati nastali kao open source projekti ogromne zajednice "slobodnih" programera.

MICROSOFT VISUAL STUDIO .NET

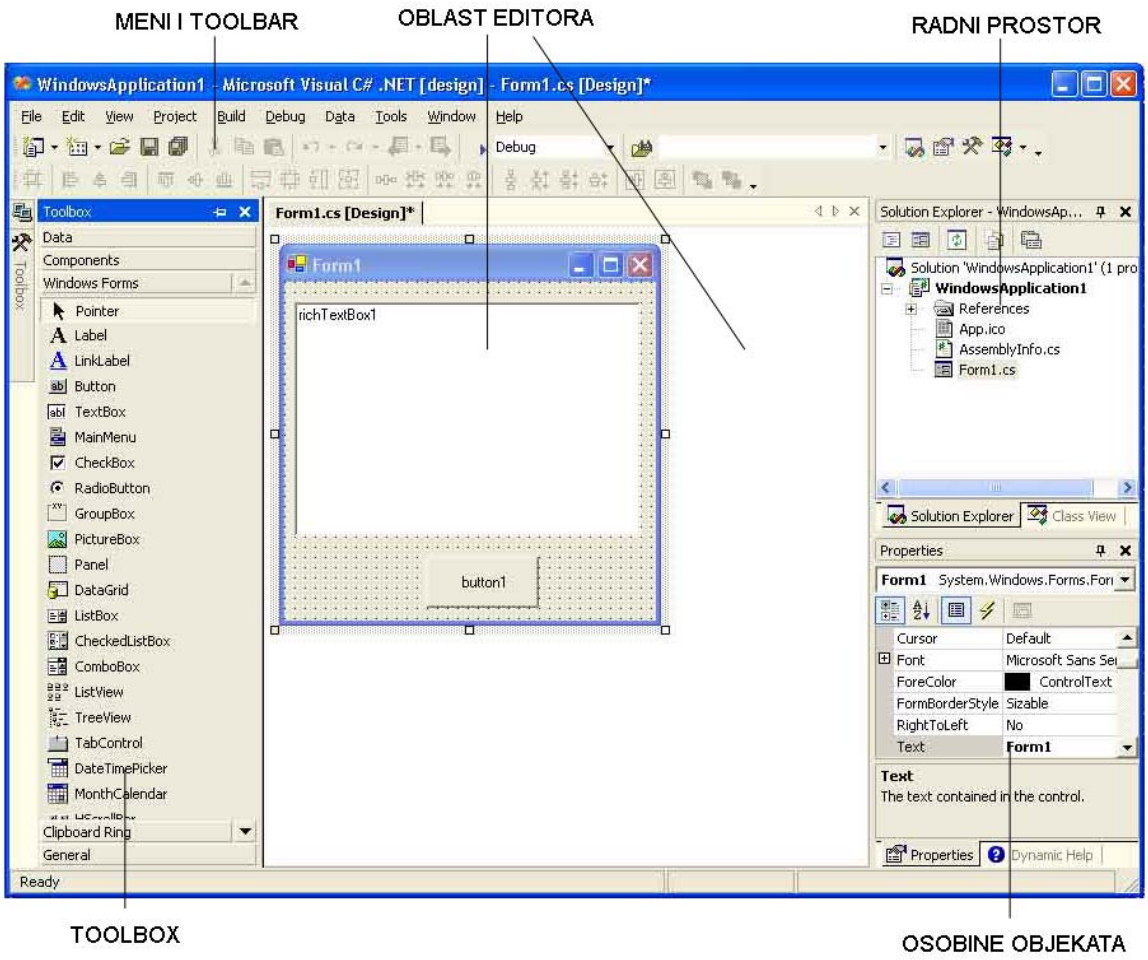
.NET (dot net) je Microsoftova platforma za razvoj nove generacije aplikacija i servisa za Windows, Web i mobilne uređaje. Osnovna ideja celog .NET-a je rešavanje integracije - fundamentalnog problema sa kojim se suočavaju današnji programeri.

Ne tako davno integracija je podrazumevala samo osiguravanje razmene informacija između računara povezanih u lokalnu mrežu. Iznenada se pred programere pojavio novi problem. Organizacije se više nisu zadovoljavale razmenom podataka u okviru svojih internih sistema, već su ih želele integrisati sa sistemima dobavljača i kupaca. Microsoft .NET rešava te probleme korišćenjem otvorenih internet standarda i tehnologija, kao što su na primer **XML** servisi.

REVOLUCIJA U PROGRAMIRANJU

Prošlo je vreme kompajliranja programa iz komandne linije. Pošto nijedan program nema budućnost ukoliko nema dovoljno kvalitetan grafički interfejs, Microsoft se zaista potrudio da ispravi neke svoje greške iz prethodnih verzija **Visual Studia**. U ovoj verziji integrisano razvojno okruženje (eng. **Integrated Development Environment - IDE**) je na vrhunskom nivou. Ono predstavlja panel kroz koji upravljamo projektom, u kome pišemo programski kod, pokrećemo debugger i pristupamo dokumentaciji, odnosno programima za pomoć. Kada se otvara više dokumenata, njihovi nazivi se pojavljuju na vrhu radne površine, tako da im je na taj način olakšano pristupanje.

Editor za kod, osim podrške za programske jezike, podržava kreiranje **HTML** dokumenata, **CSS**-a (**Cascading Style Sheets**) i **XML**-a. Greške u sintaksi su podvučene, pa izgledaju kao gramatičke greške u Wordu.



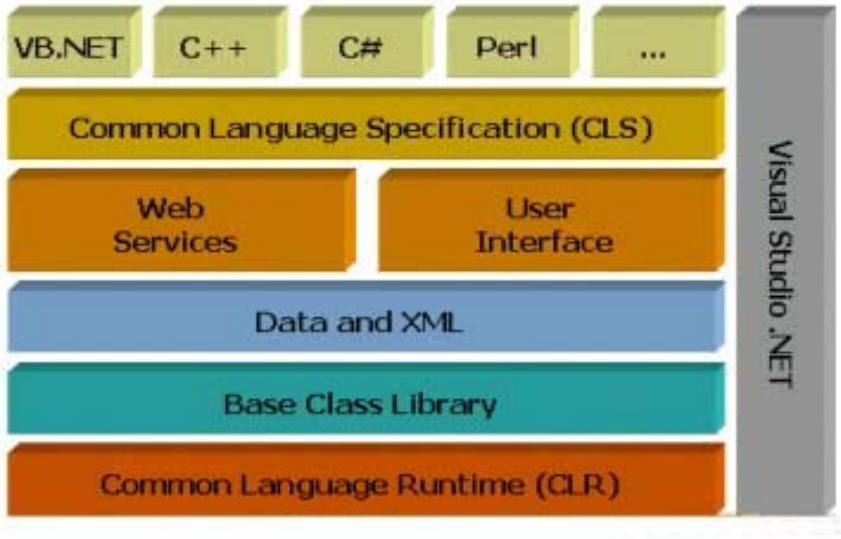
Slika 1. Visual Studio

REŠENJA (SOLUTIONS)

Za razliku od verzije 6, **Visual Studio .NET** predstavlja pravu ujedinjenu razvojnu okolinu. Kada je prilikom razvoja projekta u Visual Studiju 6 bilo potrebno paralelno koristiti C++ i Visual Basic programski kod, moralo je da se otvore tri aplikacije: Visal C++, Visual Basic i - povrh svega - **MSDN** za pristup dokumentaciji. Visual Studio .NET ujedinjuje sve razvojne alate u jednu okolinu, a projekat u kome se koristi više od jednog programskog jezika se zove rešenje (**solution**).

.NET FRAMEWORK

.NET Framework je platforma za razvoj, instalaciju i pokretanje .NET aplikacija i servisa. Sastoji se od dva glavna dela: **CLR**-a (**Common Language Runtime**) i **FLC**-a (**Framework Class Library**).



Slika 2. .NET Framework

CLR upravlja memorijom, izvršava kod, brine se o sigurnosti koda, prevodi kod u mašinski jezik (kompajliranje) i upravlja ostalim sistemskim servisima. CLR je dizajniran za maksimalno iskorišćavanje performansi računara na kome se izvršava programski kod. CLR koristi **Just-in-time** (**JIT**) kompajler koji prevodi kompletan CLR pseudo-kod u mašinski.

FLC (**Framework Class Library**) je hijerarhiski set biblioteka klasa. U FCL-u se nalazi komponentizovana verzija **ASP**-a (**Active Server Pages**) koja se zove **ASP.NET**. Služi za kreiranje Web skriptova koji se nalaze na strani servera. Zatim, **ADO.NET** podsistem za rad sa bazama podataka i **Windows Forms** okoline za razvoj naprednih Windows aplikacija. FCL je objektno orijentisan i usko povezan sa CLR-om.

VISUAL BASIC .NET

Visual Basic.NET sa prethodom verzijom Visual Basic 6 deli samo ime, a zapravo je to potpuno novi programski jezik. Ukratko, Visual Basic.NET je moćan, objektno orijentisan programski jezik. Osim standardnih karakteristika objektno orijentisanih jezika (nasleđivanje, interfejsi ...) podržana je i višenitnost (multithreading), pa Visual Basic programeri sada mogu pisati aplikacije koje nezavisno izvršavaju više zadataka.

VISUAL C++ i C#

Microsoft je bio "pravedniji" prema C/C++ programerima, pa je programski jezik namenjen .NET platformi nazvao **C#** (**si šarp**). **C#** svoje temelje nalazi upravo u ova dva spomenuta jezika. To je pre svega moderan programski jezik i kao takav sadrži sve osobine koje bi novi jezik morao imati: jednostavnost, doslednost, objektnu orijentisanost, kompatibilnost i fleksibilnost. Prvenstveno je namenjen razvoju programa za .NET platformu, iako se može upotrebljavati i za razvoj svih ostalih aplikacija, uključujući i servise.

Standardni C++ programski jezik je dobio mogućnost izvršavanja u dva režima: **managed** (upravljani) i **unmanaged** (neupravljani) režim. Ukoliko se govori o managed režimu to znači da se C++ program izvršava u CLR-u, kao i bilo koji druga .NET aplikacija. Ukoliko je ipak bitno da se program izvršava van CLR-a, onda postoji mogućnost klasičnog (unmanaged) režima rada.

.NET ZA LINUX

Projekat Mono je inicijativa određenog dela Linux zajednice za razvijanje open source Linux verzije Microsoft .NET okruženja za razvoj. Jedan od ciljeva je uvesti programski jezik **C#** u postojeći arsenal open source alata za razvoj softvera i da se omogući kreiranje .NET programa nezavisnih od operativnog sistema. Projekat Mono obezbeđuje C# kompajler koji proširuje mogućnosti **GNOME** razvojne platforme, dozvoljavajući programerima da stvaraju .NET kompatibilne aplikacije.

Drugi deo projekta Mono se stara o kompletnoj implementaciji biblioteka klasa kompatibilnih sa Microsoftovim **CLI** (eng. **Common Language Infrastructure**), što omogućava programerima da stvaraju klijentske aplikacije i Web servise, koristeći mogućnosti baza podataka dostupnih na open source sistemima.

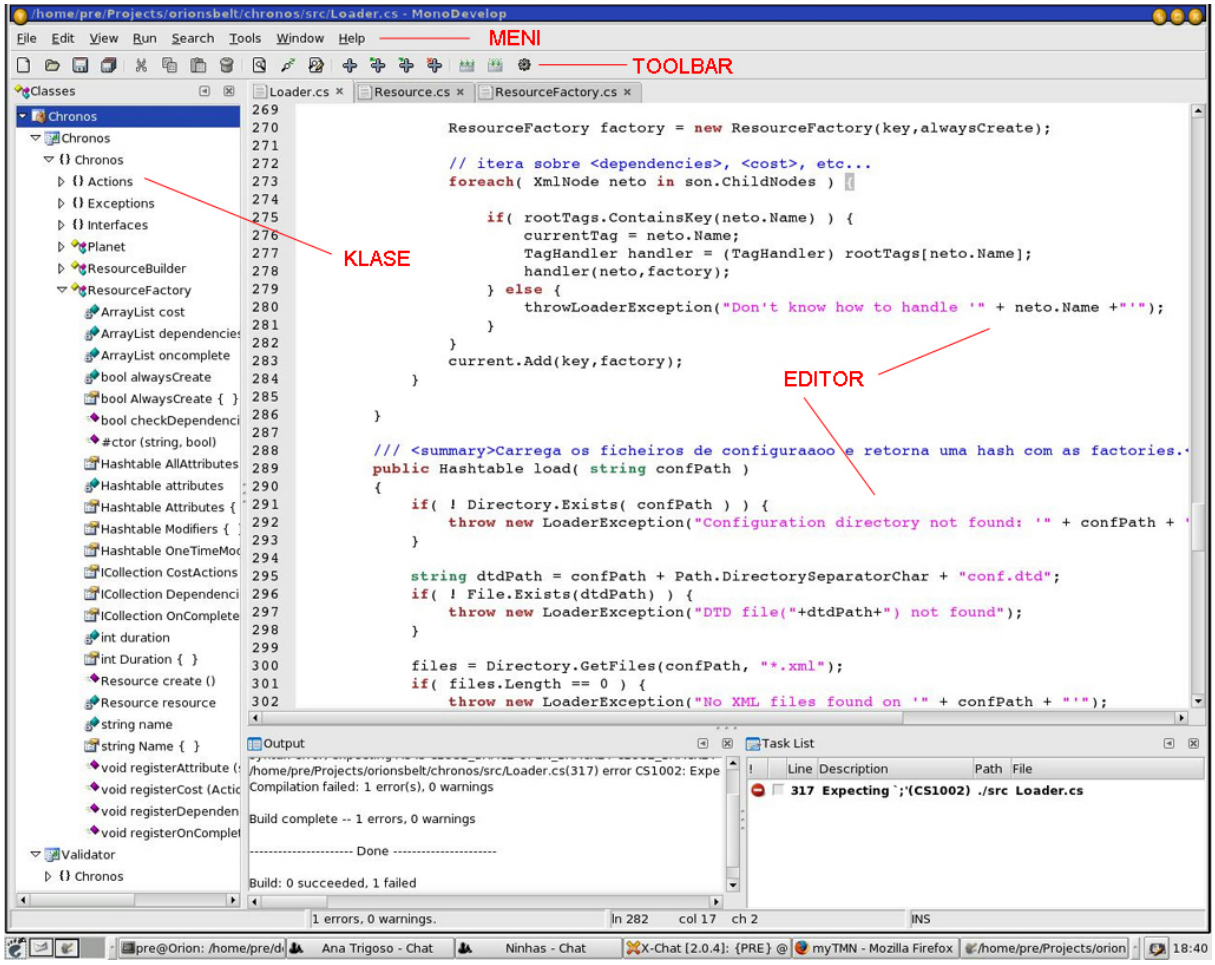
Treći deo projekta omogućava Linux verziju **CLR**-a (**Microsoft Common Lanugage Run-Time**) i **JIT** (**Just-in-Time run time engine**), što treba da omogući Linux sistemima da izvršavaju .NET aplikacije kreirane bilo na Windows ili Linux platformi.

Na površini osnovnog sistema se nalaze dva **API**-ja koji definiše korisnički interfejs, Web servise i pristup bazama podataka, dok drugi sloj originalno pripada projektu Mono, a uključuje stvari koje omogućavaju funkcionisanje **GTK** (**Graphical Toolkit**), Cairo grafički sistem i Mono database sloj.

MONODEVELOP

Pored implementacije poznatog API-ja, postoji još jedan faktor koji bi trebao da privuče Windows programere ka Mono platformi - **IDE** (**Integrated Develpment Environment**). Mono projekat je pripremio aplikaciju pod imenom **MonoDevelop**, koja je portovana verzija open source **SharpDevelop** aplikacije. MonoDevelp ima novi korisnički interfejs, a ponaša se i izgleda kao **GNOME** aplikacija.

MonoDevelop ima sve što je potrebno jednom integrisanom razvojnom okruženju i rad sa njim je lak, koliko to može biti. Na primer, u slučaju pisanja Web stranica sa MonoDevelp alatom, prikaz Web strane se može u realnom vremenu videti u oknu Browser, koji će se menjati onako kako se unose promene u kodu.



Slika 3. MonoDevelop

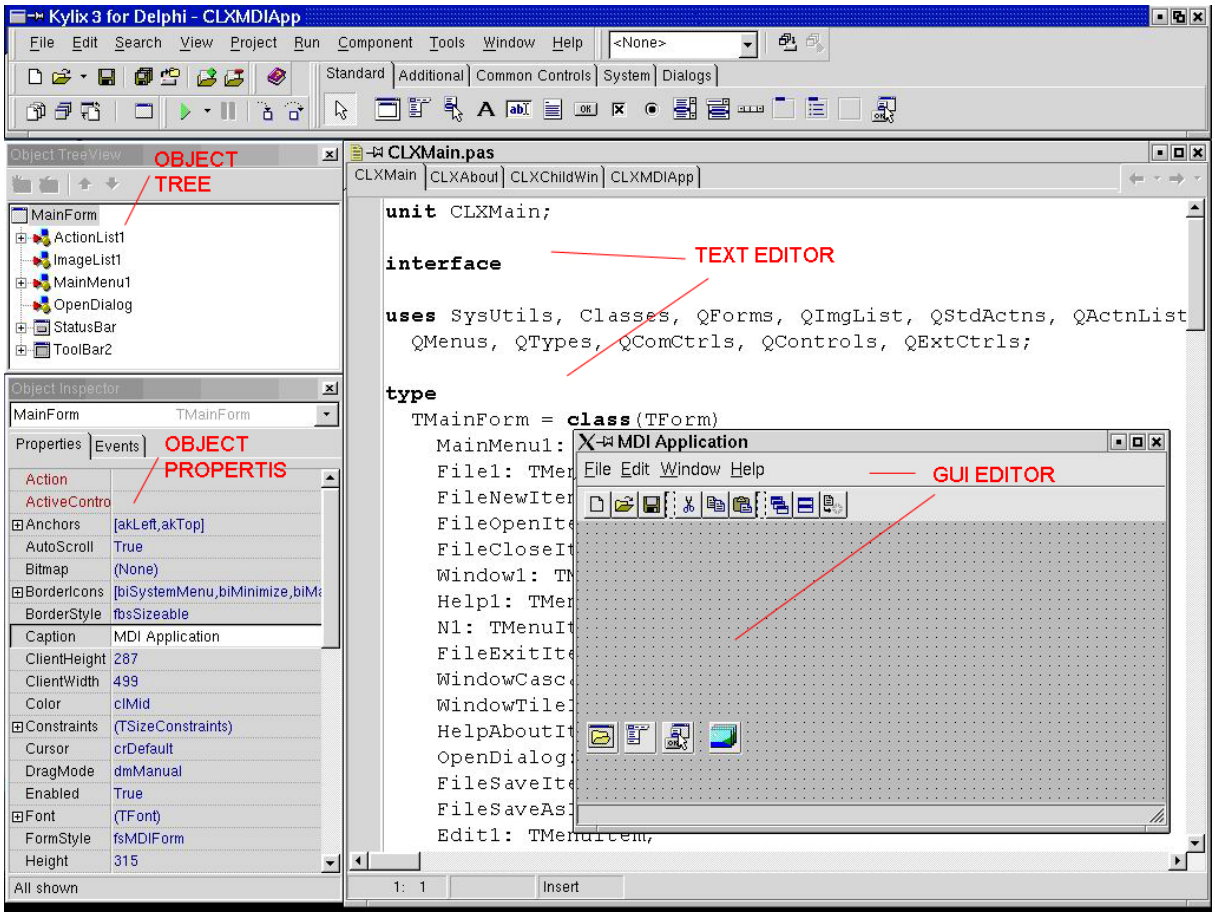
Deo projekta Mono je i potprojekat koji treba da se postara o kvalitetnoj dokumentaciji, uključujući i Mono specifične API-je.

Prema argumentima protivnika projekta Mono, on je dar sa neba za Microsoft, koji u pravom trenutku može iskoristiti patentna prava i zaustaviti razvoj Linuxa za više godina. Prema ovim protivnicima, Microsoftova najbolja strategija je da pomogne uspeh projekta Mono, pripremajući zamke. One bi se mogle sastojati u tihom patentiranju ključnih funkcija .NET platforme, što je već dobrim delom i učinjeno. Nakon toga Microsoft treba samo da čeka pravi trenutak, koji bi mogao biti u situaciji kada veliki deo razvoja bude prešao na ovu platformu.

Mono projekat je sada deo kompanije **Novell**, a ona je poznata kao stari neprijatelj **Microsofta**. Takođe, očigledno je da sa Mono platformom postoje veliki planovi.

KYLIX

Delfi programski jezik, kao što je poznato, nije ništa drugo nego **objektno orijentisan Paskal**. Dugo su Delfi programeri čekali na svoj alat kojim će moći izrađivati aplikacije i za Linux operativni sistem.



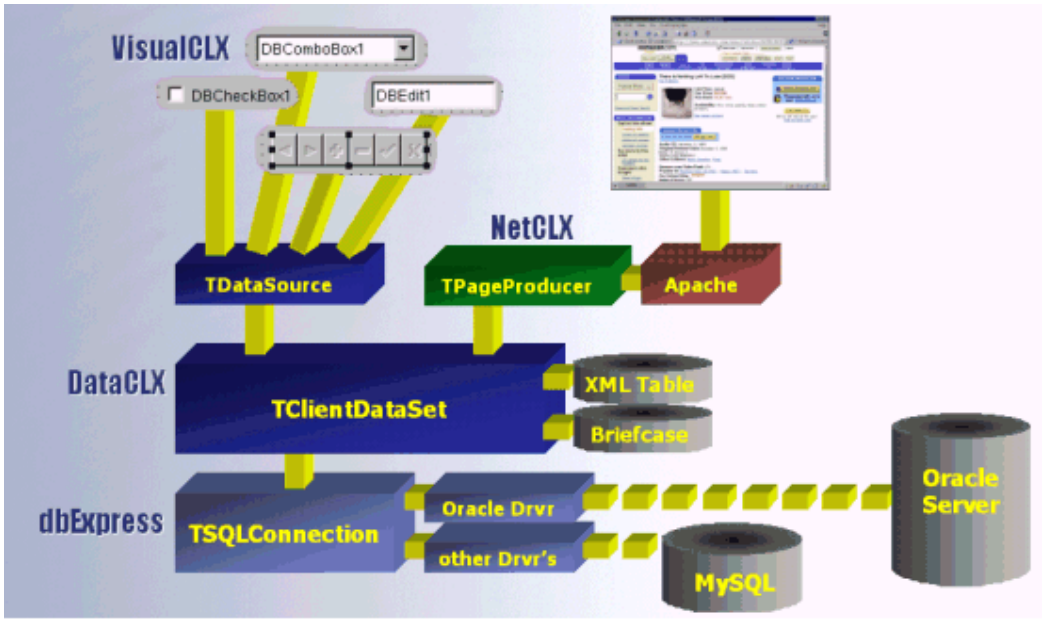
Slika 4. Kylix

Razvoj moćnog alata za Delfi pod Linux platformom počeo je još 1999. godine. Činjenica je da su **Delfi** i **Kylix** međusobno kompatibilni, tako da je moguće napraviti aplikaciju koja će u isto vreme biti pogodna i za Linux i Windows operativni sistem. Ovo je naročito korisno za programere koji simultano pišu aplikacije za Windows i Linux platformu.

Kylix se odlično povezuje i sa **.so** (**shared object**) datotekama, tako da je moguće koristan deo koda napisati u bilo kom jeziku, ako iz bilo kojeg razloga ne odgovara objektni Paskal koji koristi Kylix. Ovaj alat bez poteškoća radi u oba najpopularnija grafička okruženja u Linuxu - **KDE** i **GNOME**.

Nove verzije Kylixa imaju nadograđenu IDE podršku koja sada developerima omogućava izradu aplikacija pisanih u **Delfi** ili **C++** programskom jeziku. Naime, one uključuju IDE za oba spomenuta programska jezika. Praktični način na koji se ova mogućnost koristi jeste da se pod KDE ili GNOME grafičkim okruženjem pokrene Kylix i zatim selektuje Delfi ili C++ način rada. Novost je i novi prozor **Object TreeView**, koji prikazuje logičke odnose između vidljivih i nevidljivih komponenti na formi. Object Treeview je sinhronizovan sa **Object Inspector** i **Form Designer** prozorom. Bitna novost je prisutna i na novi glavni meni Kylixa, u kome je moguće izabrati pet različitih tipova projekta - **aplikacija**, **modul**, **forma**, **frejm** i **Unit**.

Neki će se programeri možda odlučiti za izradu frameworka cross-platformskih aplikacija uz pomoć ugrađene **Component biblioteke** (**CLX**). Sa ovom bibliotekom moguće je pisati Linux CLX aplikacije korišćenjem C++ ili Delfi koda i njegovo kompajliranje s bilo kojim Borlandovim Windows razvojnim paketom - C++ Builderom ili Delfijem, i to za izradu Windows aplikacija i obratno.



Slika 5. CLX Library

JAVA

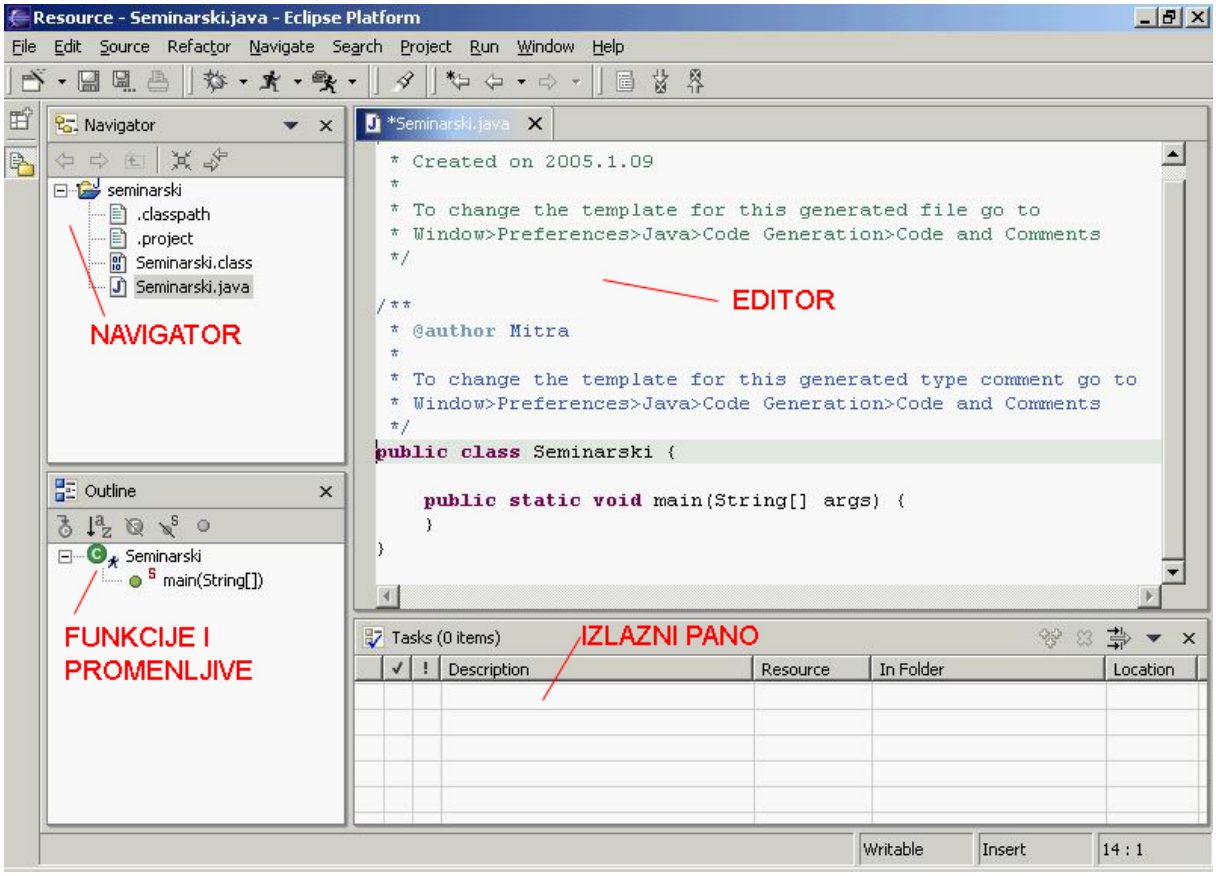
Java je programski jezik koji je razvio **Sun Microsystems** i vrlo je sličan programskom jeziku **C++**, ali pojednostavljen zbog želje da se izbegnu uobičajene greške prilikom programiranja. Prvobitno ime Jave bilo je **OAK**, a namera njenih tvoraca - stvaranje programskog jezika za palm-topove i male kućne aparate. Godine 1995. OAK je doživeo neuspeh, pa je preimenovan u Javu i modifikovan za primenu u to vreme naglo rastućem **World Wide Web** okruženju tj. **Internetu**.

Datoteke s Java izvornim kodom (**.java datoteke**) prevode se (kompajliraju) u format nazvan **bytecode** (**.class datoteke**) koga na kraju izvršava **Java interpreter**. Kompajlirani Java kod može se izvršavati na većini današnjih računara zahvaljujući Java interpreterima i okruženjima za pokretanje Java programa (poznatijih kao **Java Virtual Machine**).

Najveća greška programera početnika, a nažalost ponekad i onih iskusnih, predstavlja vezivanje za razvojno okruženje, a ne za programski jezik. Tako je većina C/C++ programera vezana za Visual Studio, dok Pascal programeri sebe obično nazivaju Delphi programerima, prema razvojnom okruženju. Na svu sreću, kada je u pitanju Java, nijedno razvojno okruženje nije dominantno, pa su svi programeri u Javi uglavnom okrenuti ka karakteristikama i mogućnostima samog jezika.

ECLIPSE

Eclipse Fondation je neprofitabilna organizacija koja uz pomoć programera iz celog sveta razvija ovo programersko okruženje. Glavni finansijer celog projekta je **IBM**, koji je ujedno i jedan od najjačih zagovornika Java tehnologija. Glavna specifičnost ovog okruženja je korišćenje **SWT** biblioteke kao osnovne umesto **Swing**-a, što se po želji može promeniti. Sam SWT je odlična biblioteka vizuelnih komponenti, nešto brža od Swing biblioteke i ima mogućnosti kojih u standardnoj SUN-ovoj biblioteci nema (npr. nečetvrtasti prozor).



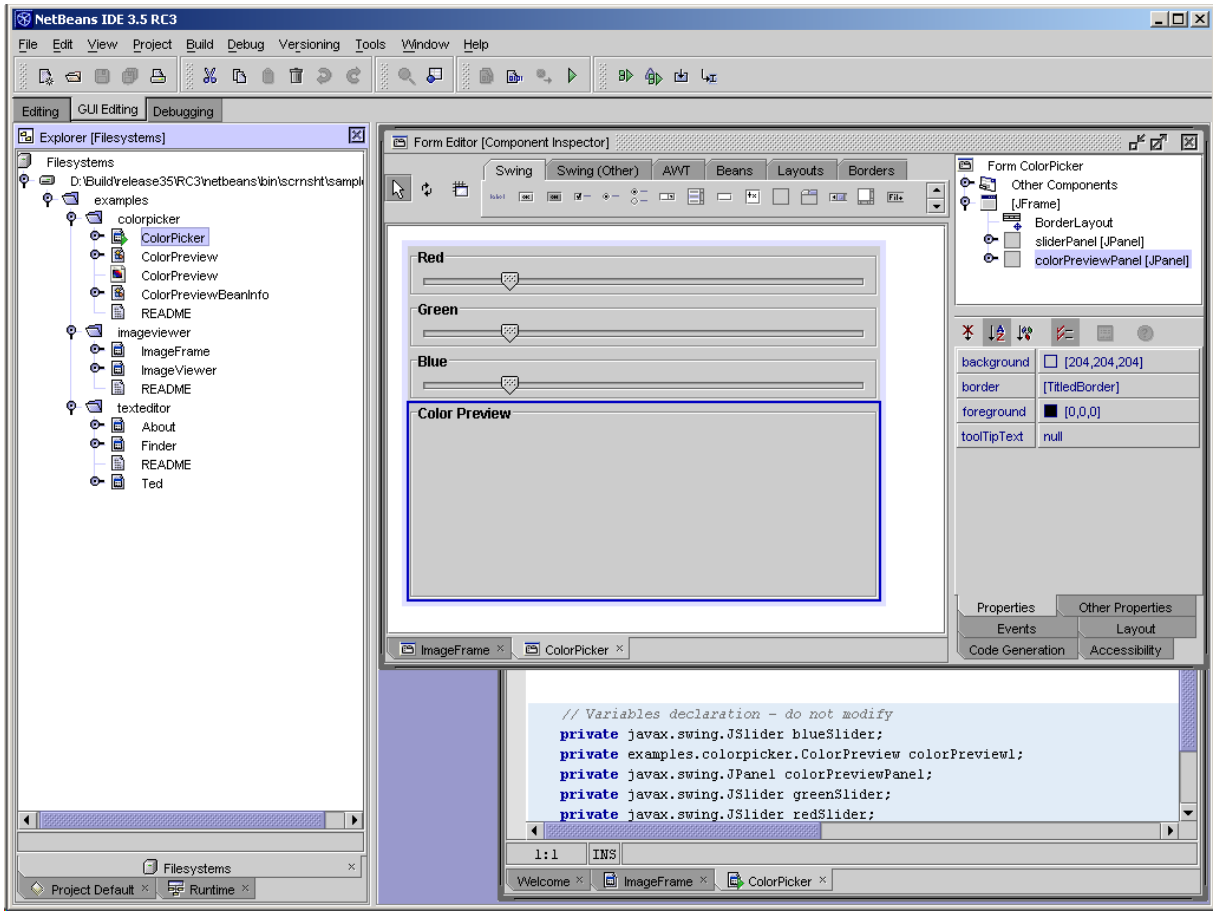
Slika 6. Eclipse

Dobra strana Eclipse-a je i veliki broj **plug-inova** koji se razvijaju nezavisno od samog okruženja, a odlično se integrišu u samo okruženje. U tabeli se vidi da Eclipse podržava vizuelno kreiranje GUI-a, što je ostvareno preko nekih od njenih dodataka.

Eclipse ima veliku bazu korisnika i jedno je od najpopularnijih razvojnih okruženja za Javu. Lako se integriše sa drugim Java tehnologijama. Po mogućnostima i kvalitetu prevazilazi i većinu komercijalnih izdanja.

NETBEANS

NetBeans je slično Eclipse-u, besplatno razvojno okruženje, open source, iza koga stoji jedan gigant. U slučaju NetBeans-a to je niko drugi do **SUN Microsystems**, tvorac **Jave**.

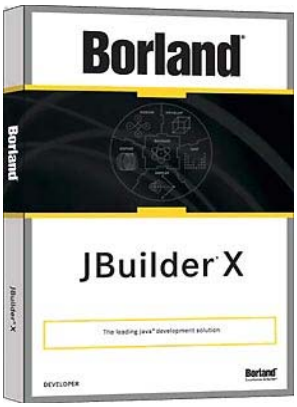


Slika 7. NetBeans

NetBeans trenutno ima manje korisnika nego Eclipse, ali zahvaljujući njegovom kvalitetu baza korisnika se intenzivno povećava. NetBeans su se pokazali kao veoma zgodan alat za automatsko kreiranje GUI-a. Dobro se integriše sa **Tomcatom** (Web serverom kompletno pisanom u Javi). Ima veliku bazu plug-inova, tako da se lako dodaju nove mogućnosti. Zgodna specifičnost je to što korisnik može da piše svoje module i integriše ih u okruženje i tako dobija polugotovo grafičko okruženje za svoje Java aplikacije.

Kao nedostatak se može istaći loša podrška za uvoz projekata iz drugog okruženja, kao i teže korišćenje projekta rađenog u NetBeans-u u nekom drugom razvojnom okruženju.

JBUILDER



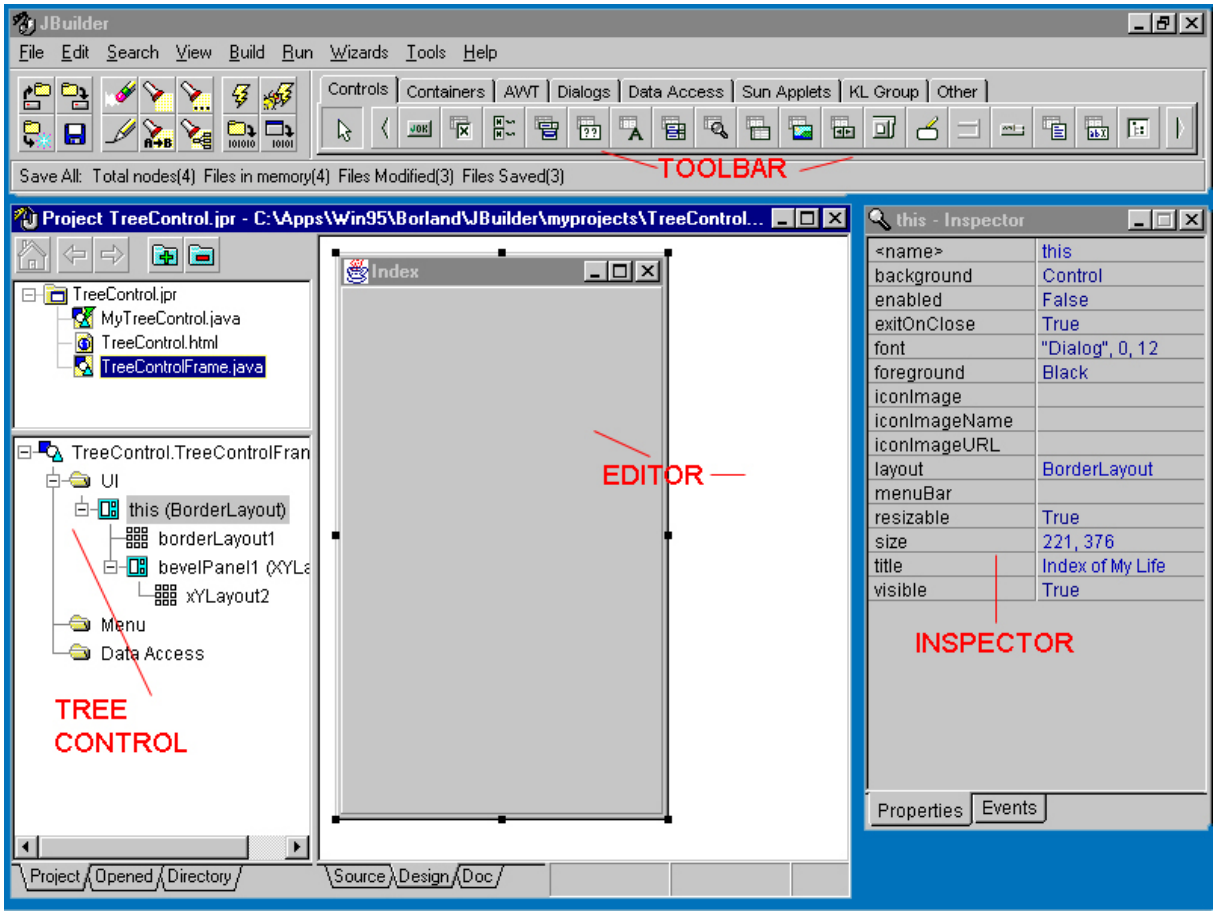
Slika 8. JbuilderX

Borland je još jednom opravdao očekivanja najzahtevnijih programera. Najnovija verzija JBuildera nosi oznaku **X**. Poznat kao firma, u to vreme još sa imenom **Inprise**, koja je napravila revoluciju u programiranju svojim programskim alatom za Delphi programere, Borland je nastavio tradiciju pravljenja vrhunskih alata i kao za sve svoje proizvode i **JBuilderX** je kreirao u tri verzije, u ovom slučaju **Developer, Enterprise i Mobile**.

- **Developer verzija** je namenjena programerima koji kreiraju manje projekte. Ne sadrži mogućnosti koje su potrebne za kreiranje aplikacija za složena mrežna okruženja u velikim korporacijama, ali je ipak dovoljna za većinu standardnih programa.

- **Enterprise verzija** sadrži sve što je potrebno za kreiranje bilo kakve aplikacije, u bilo kakvom okruženju. Čak i nedostatak proširivanja mogućnosti preko plugin-ova se ne može tako okarakterisati jer jednostavno ne postoji mogućnost koja bi mogla da se doda ovom razvojnom okruženju za Java programere. Jedna ista verzija radi na svim platformama, čak je isti i način instalacije.

- **Mobile verzija** je markentinški potez Borlanda koji se mogao i očekivati u trenutku naglog širenja mobilne telefonije i smart uređaja. Optimizovana je za kreiranje programa (i igrica) za ove aparate podržavajući **J2ME** (**Java2 Mikro Edition**).



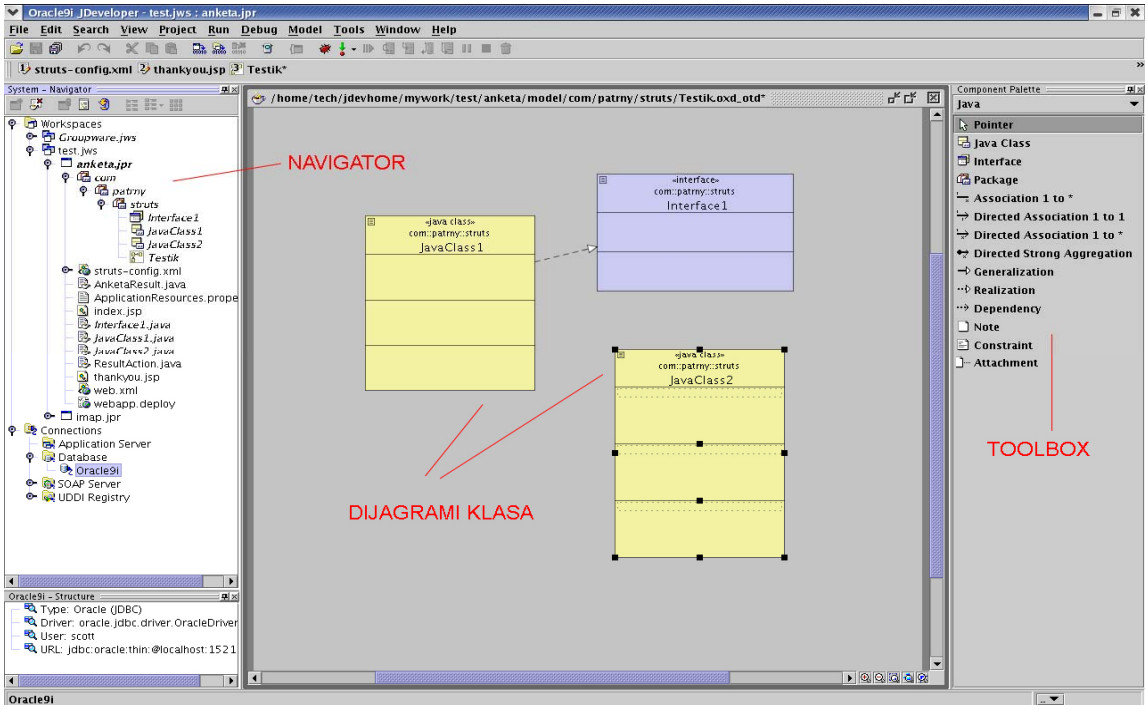
Slika 9. JBuilder

JDEVELOPER

Oracle korporacija je neverovatan, ali i opravdan tržišni uspeh doživela kreiranjem istoimene i danas jedne od najpopularnijih baza podataka. Ona je optimizovana za rad u višeplatformskom okruženju i zato zahvalnost duguje postojanju programskog jezika Jave. U početku su druge firme pisale pomoćne alate za Oracle, ali kako se firma širila počela je sa kreiranjem programa za svoju bazu podataka koji su se pokazali tako dobrim da su kasnije izrasli u nezavisne alate.

Jedan od ovih programa je i **JDeveloper**, alat za kreiranje Java aplikacija koji dolazi uz Oracle, kao svakodnevni i podrazumevani program za developere baza podataka. Međutim, može se nabaviti i instalirati kao samostalan proizvod. Sadrži ugrađeni **UML modeler** koji ima mogućnost kreiranja koda i to funkcioniše dvosmerno. Bilo kakva promena u modeleru se odražava u kodu i obratno!

Nedostaci su loš rad sa projektima iz drugog okruženja i nemogućnost kreiranja **jar**-a (arhive za Java datoteke).



Slika 10. JDeveloper

PREGLED MOGUĆNOSTI ALATA ZA KREIRANJE JAVA APLIKACIJA

		Eclipse	NetBeans	JDeveloperer	JBuilder
MOGUĆNOT EDITORA	AutoComplete	Da	Da	Da	Da
	Reformat	Da	X	X	Da
	Live Template	Da	Da	Da	Da
	Predlog za ispravljanje greške	Da	Da	Da	Da
	Zagrade	Da	Da	Da	Da
	Methode separator	X	X	Da	X
MOGUĆNOST OKRUŽENJA	Uvoz postojećeg projekta	Da	X	X	Da
	Generisanje novog projekta	Da	Da	Da	Da
	Korišćenje projekta u drugom okruženju	Da	X	X	Da
	Debug	Da	Da	Da	Da
	Generisanje Jar-a	Da	Da	X	Da
	Generisanje exe, bin...	X	X	X	Da
	Integracija sa Tomcatom	PlugIn	Da	Da	Da
	Rad sa ANT-om	Da	Da	Da	Da
	Generisanje JavaDoc-a	Da	Da	Da	Da
	Plugins	Da	Da	Da	X
	Besplatan	Da	Da	Da	Da
	Komercijalan	X	X	X	Da
GUI	Generisanje GUI-a	Da	Da	Da	Da
	Generisanje GUI-a binarno	PlugIn	X	Da	Da
	Generisanje GUI-a u kodu	PlugIn	Da	Da	Da
OS	Windows verzija	Da	Da	Da	Da
	Linux verzija	Da	Da	Da	Da

Tabela 1. Java tools

VISUAL MODELING

Sa razvojem objektno-orijentisanih jezika javila se potreba za vizuelim modelovanjem, tj.grafičkim predstavljanjem objekata, programskog toka, aktivnosti, komunikacije među objektima itd... Modeli odnosno dijagrami su korisni za razumevanje problema, kreiranje dokumentacije, dizajniranje programa i baza podataka i za međusobnu komunikaciju svih koji su uključeni u konkretni projekat(anilitičari, dizajneri, programeri, krajnji korisnici).

Kako softverski sistemi postaju sve kompleksniji, ne možemo da ih razumemo u potpunosti. Modeliranje je idealni način da shvatimo svu apstrakciju složenog problema i posmatramo samo one delove koji su nam u tom trenutku bitni.

Za modelovanje se koristi **UML** (**Unified Modeling Language**), Universalni Jezik za Modelovanje. Glavna osobina UML jezika je skalabilnost - možemo grafički da predstavimo samo jedan deo koda ili jedan proces, a možemo i celokupan projekat sa svim detaljima. Veoma interesantno je i to što je potpuno dozvoljeno da se dodaju informacije ili simboli koji ne pripadaju UML specifikaciji, kao što su simboli iz oblasti kojoj je program namenjen. Zvanično telo zaduženo za standardizaciju UML-a je **OMG** (**Object Management Group**).

Postoji ukupno 13 zvaničnih tipova dijagrama. Svaki od ovih tipova namenjen je određenom aspektu programa ili procesa.

- **Dijagram aktivnosti** - služi za opisivanje procedure, poslovnih postupaka i toka posla. Ovaj dijagram je sličan **dijagramu toka**, ali pomoću njega mogu da se prikažu paralelne aktivnosti.

- **Dijagram klasa** - najčešće je korišćen i služi za prikaz klasa i njihovih odnosa. Pomoću ovih dijagrama predstavljamo celu hijerarhiju klasa ili samo njen deo. Sama klasa predstavlja se samo imenom ili sa svim detaljima (polja klase i metodi u klasi). Dijagramom možemo da predstavimo sve varijante klasa, uključujući apstraktne klase i interfejse.

- **Dijagram komponenti** - kokristi se ređe, a postoje različita mišljenja o tome da li je komponenta isto što i klasa, kao i o tome kakve su razlike. UML standard omogućava da oni koji žele da rade sa komponentama imaju dijagram za prikazivanje strukture komponenti i njihovih veza.

- **Dijagram komunikacije** - koristi se za prikaz veza između učesnika, to je jedna vrsta dijagrama interakcije. Dijagram komunikacije se ređe koristi od **dijagrama sekvence**, koji ima sličnu namenu. Razlika između ova dva dijagrama je u tome što se dijagram sekvence odnosi na redosled izvršavanja, a dijagram komunikacije se više odnosi na veze među podacima. Prilikom međusobnog konsultovanja mnogi menadžeri više koriste dijagram komunikacije, jer se lakše menja i bolje oslikava potrebe programa, dok se u kasnijem radu i u dokumentacije više koristi dijagram sekvenci, jer daje više informacija.

- **Dijagram stanja** - opisuje ponašanje sistema, odnosno nekog njegovog dela. Često se koristi i za opisivanje životnog veka nekog objekta. Ređe se primenjuje za veće sisteme ili skupove klasa.

- **Dijagram objekata** - prikazuje stanje objekata sistema u nekom trenutku. Za razliku od dijagrama klasa, koji prikazuje klase, kijagram objekata prikazuje dijagram inistanci klasa, odnosno objekte, zato se i zove dijagram objekata. Ovaj dijagram je od velike pomoći kada dijagram klasa nije dovoljan; tada treba dati nekoiko karakterističnih dijagrama objekata, koji će predstaviti stanje sistema u nekom trenutku i tako dati dodatne informacije o klaama i njihovom funkcionisanju. Ovaj dijagram ima sličnu upotrebu kao i dijagram komunikacije, samo što ovke nema poruka.

- **Dijagram paketa** - služi da u velikim sistemima prikažemo odgovarajuće podceline. Paketi se inače koriste u svim objektno orijentisanim jezicima, pa ih tako imamo u Javi pod istim nazivom ili u .NET-u pod imenom **namespace**. Zbog ove činjenice dijagram paketa se veoma lako i efikasno prenosi u programski kod. Dijagrami paketa su izuzetno korisni jer nam daju jasnu sliku o organizaciji projekta i njihovoj strukturi. Ovaj dijagram bi trebalo da se koristi kad god imamo malo veći sistem.

- **Dijagram pregleda interakcija** - u stvari je spoj **dijagrama aktivnosti** i **dijagrama sekvenci**. Ovo je novi tip dijagrama ubačen u verziji 2 UML standarda. Njegova glavna odlika je u objedinjenom prikazu, koji daje potpunu sliku interakcija. Ono što smeta je spajanje dva različita tipa prikaza u jednu celinu, što otežava tumačenje dijagrama.

- **Dijagram raspoređivanja** - predstavlja fizičku organizaciju sistema i koji se deo programa gde izvršava. Ovi dijagrami su izuzetno važni u slučajevima distributivnih programa, odnosno kod Web baziranih programa.

- **Dijagram sekvence** - prikazuje jedan scenario u kome učestvuje određeni broj objekata sa prikazom njihovih komunikacija. Ovaj dijagram je veoma zgodan za prikaz pojedinačnih slučajeva.

- **Dijagram složene strukture** - omogućava da izvršimo hijerarhijsko razlaganje klasa. Za razliku od dijagrama paketa, koji nam opisuje grupisanje u trenutku prevođenja, dijagram složene strukture pokazuje grupisanje u trenutku izvršavanja.

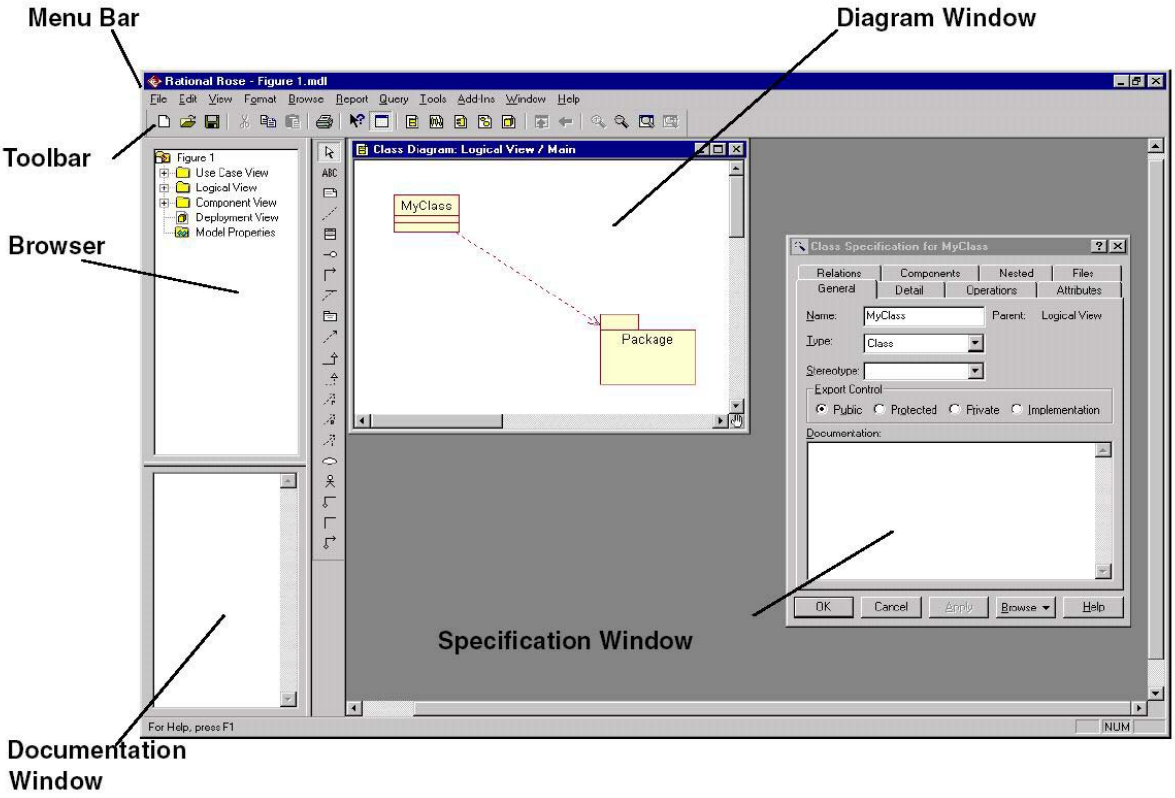
- **Dijagram slučajeva korišćenja** - prikazuje jedan slučaj korišćenja sistema od strane korisnika. Ovo je pogled na sistem spolja, sa tačke korisnika. Za razliku od drugih dijagrama, dijagram slučajevakorišćenja trebalo bi da bude propraćen odgovarajućim tekstualnim opisom slučaja. Ovaj tekstualni opis je čak i važniji od samog dijagrama. Ovi dijagrami i njihovi tekstualni opisi bi trebalo da budu osnova komunikacije između korisnika i tvorca sistema.

- **Vremenski dijagrami** - Predstavljaju još jedan tip dijagrama interakcije. Vremenski dijagrami se koriste za prikazivanje vremenskih ograničenja između različitih stanja objekata.

Svojim kvalitetom su se izdvojila dva proizvoda koja služe za Visual Modeling: **Microsoftov Visio** i **IBM**-ov **Rational Rose**. Ovde se mora spomenuti i EclipseUML koji radi kao dodatak za Eclipse platformu za kreiranje Java aplikacija koji je besplatan, ali ima zavidan set mogućnosti.

RATIONAL ROSE

Rational Rose je softversko rešenje za visualno modelovanje koje dozvoljava kreiranje, analiziranje, dizajn i modifikaciju elemenata i njihovo uklapanje u grafičku representaciju celokupnog sistema.



Slika 11. Rational Rose

OSOBINE

Rational Rose pruža sledeće mogućnosti za analizu, dizajn i konstruisanje aplikacija:

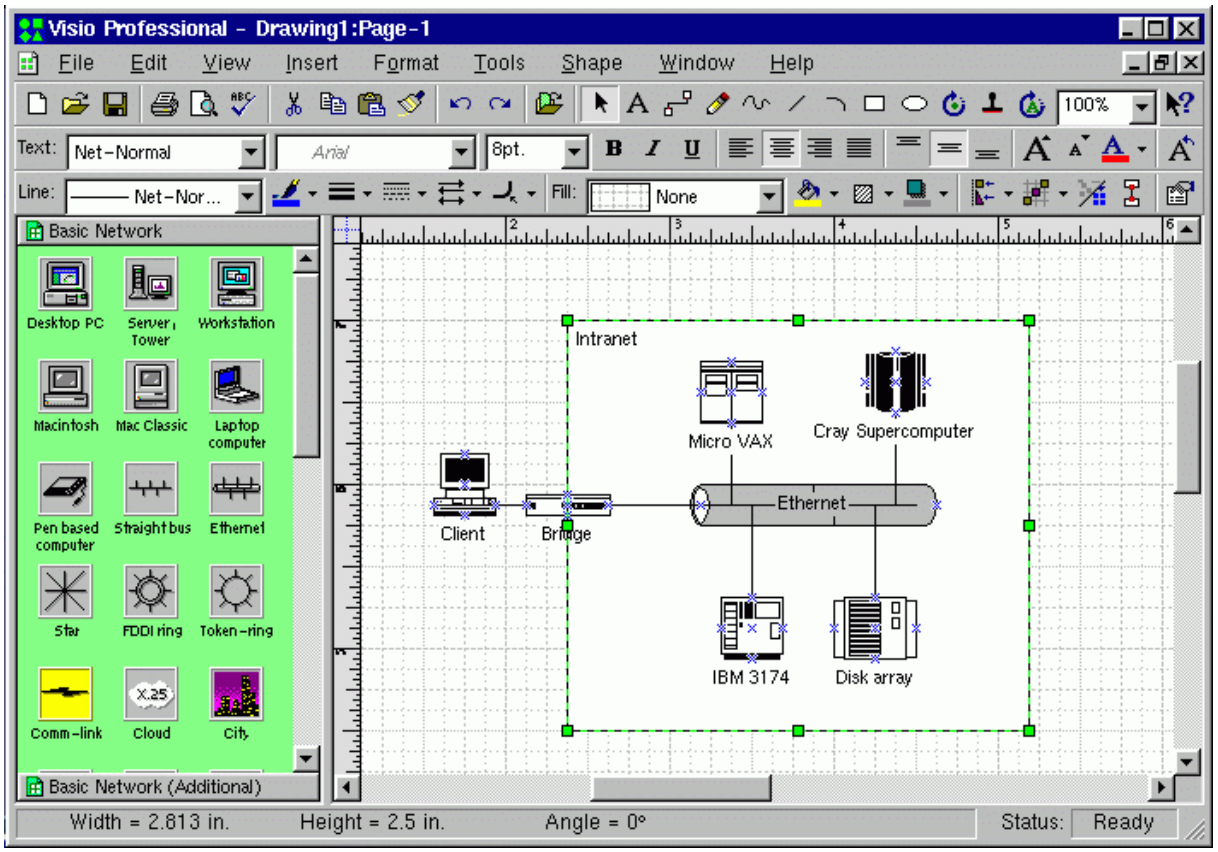
- Objektno-orijentisano modelovanje
- Podrška za UML, COM, OMT i Booch 93 (metode za vizuelno modelovanje)
- Paralelni višekorisnički razvoj
- Generisanje dokumentacije
- Rational Rose skripting za proširenje i integraciju
- Multiplatformska mogućnost

Od ogromnog značaja je i mogućnost generisanja koda u trenutno najpopularnijim programskim jezicima, kao i u CORBA-i (Common Object Request Broker Architecture) - arhitektura i tehnologija distribuiranih objekata koja je zamišljena da bude nezavisna od korišćenog programskog jezika.

MICROSOFT VISIO

Microsoft ne bi bio to što jeste kada ne bi zakoračio u sve oblasti informacionih tehnologija. Ti koraci znaju da budu zaista veliki. Microsoftovi programeri potrudili su se da deo paketa **Microsoft Office-a 2003** sa imenom **Visio** predstavlja ozbiljnu pretnju konkurenciji.

Ovaj alat za visual modeling dolazi i nezavisno od Office-a ukoliko nekom nije potreban ceo paket. U skladu sa novim tehnologijama (takođe Microsoftovih) ovaj prizvod je u stanju da kreira kod u C# i Visual Basic programskim jezicima i radi samo na Windows platformi. Projekte je moguće eksportovati u HTML obliku. Za svaku pohvalu je sposobnost Visio za povezivanje sa Rational Rose.



Slika 12. Microsoft Visio

PREGLED MOGUĆNOSTI ALATA ZA VISUAL MODELING

		Rational Rose	Visio	EclipseUML
DIJAGRAMI	Vlasnik	IBM	Microsoft	Omondo
	Dijagrami interakcije	Da	Da	Da
	Klasni dijagrami	Da	Da	Da
	Dijagrami stanja	Da	Da	Da
	Dijagrami aktivnosti	Da	Da	Da
	Sekvencni dijagrami	Da	Da	Da
	Dijagrami komunikacije	Da	Da	Da
	Dijagrami raspoređivanja	Da	Da	Da
GENERISANJE KODA	CORBA-IDL	Da	Da	X
	C++	Da	Da	X
	Java	Da	Da	Da
	Drugi jezici	Ada, Delphi, C#, C, C++	C#, Visual Basic	X
	Operativni sistemi	Windows, Unix, Linux	Windows	Svi OS koji podržavaju Javu
	Ostalo	Podrška za COM, paralelni višekorisnički razvoj, Rational Rose scripting, generisanje dokumentacije	Modelovanje baza podataka, međusobna veza sa Rational Rose, export projekata u HTML, podrška za VB scripting	Integracija sa Eclipse-om, podrška za razvoj baza podataka, generisanje dokumentacije

Tabela 2. UML Tools

INDEX SLIKA

INDEX TABELA

LITERATURA

www.microsoft.com

www.borland.com

www.eclipse.org

www.rationalrose.com

www.mono-project.com

www.oracle.com