RAZVOJ OKRUŽENJA ZA REALIZACIJU INTEGRISANOG SISTEMA ZA SIMULTANO PROJEKTOVANJE

DEVELOPING FRAMEWORK FOR REALIZATION OF INTEGRATED CONCCURENT DESIGN SYSTEM

Milan Zdravković[1], Miodrag Manić[1], Miroslav Trajanović[1]

Mašinski fakultet Univerziteta u Nišu

Sadržaj - Na Mašinskom fakultetu u Nišu, sprovedeno je istraživanje u cilju afirmacije izabranih tehnologija za realizaciju pouzdane, robusne i sigurne troslojne klijent - server arhitekture, koja treba da ispuni uslove za funkcionalan rad integrisanog sistema za simultano projektovanje.

Abstract - *This paper describes research which goal is to choose, test and affirmate technologies for realization of functional, robust and secure three-tier client - server architecture, which has to be framework for functional integrated concurent design system. Research is conducted on Faculty of Mechanical Engineering at University of Nis.*

1.UVOD

U radu, koji je nastao kao jedan od rezultata ovog istraživanja, dat je kratak rezime osnovnih funkcionalnih i tehnoloških zahteva, uz osvrt na izabrani model sistema, opis izabranih tehnologija za realizaciju sistema sa diskusijom o razlozima za njihovo usvajanje i arhitektura demo klijent - server aplikacije za simultano konstruisanje osnosimetričnih delova - Web/CADROT, koja je napravljena radi testiranja i provere opravdanosti donetih odluka.

Sadašnji sistemi za računarsko projektovanje su realizovani u vidu samostalnih aplikacija, koje se izvršavaju na računaru korisnika. Konstruisani proizvod se arhivira u vidu datoteke u specifičnom formatu i to, u lokalnom sistemu datoteka operativnog sistema računara, ili na nekom fajl serveru. Ovaj način rada ne omogućava grupni rad projektanata, jer samo jedan učesnik u procesu projektovanja, može u odredjenom trenutku da radi na modelu. Pored ovoga, svaki korisnik mora da ima sistem za projektovanje, instaliran na svom računaru.

Cilj sistema za simultano projektovanje je da omogući simultano odvijanje rada većeg broja inženjera na jednom modelu, uz primenu pravila koordinacije rada na odredjenom nivou, koja su integrisana na aplikacionom serveru.

Jedna od svrha projektovanja sistema jeste i efikasno korišćenje tipično moćnih resursa, potrebnih u oblasti projektovanja proizvoda. Veliki licencni troškovi uvođenja i održavanja složenih sistema za projektovanje, kao što je Pro/ENGINEER, se mogu uštedeti projektovanjem malih, mobilnih sistema za konstruisanje usko specijalizovane klase proizvoda, kao što su, na primer, osnosimetrični delovi.

Klijentski programi sa integrisanim jezgrom za modeliranje ni u kom slučaju ne mogu zameniti Pro/ENGINEER, ali se njihovom upotrebom u velikoj meri može smanjiti broj inženjer sati za skupom grafičkom stanicom i Pro/ENGINEER licencom, čime će se značajno smanjiti troškovi projektovanja proizvoda.

2. FUNKCIONALNI I TEHNOLOŠKI ZAHTEVI SISTEMA ZA SIMULTANO PROJEKTOVANJE

Osnovni principi za projektovanje sistema za simultano projektovanje se definišu na osnovu prethodno usvojenih funkcionalnih i tehnoloških zahteva koje sistem treba da ispuni.

Osnovni funkcionalni zahtevi, koji treba da budu ispunjeni u ovoj fazi razvoja sistema su:

1. osnov sistema predstavlja objektno orijentisani model proizvoda, definisan u sistemu CADROT, razvijenom u LIPS laboratoriji Mašinskog fakulteta u Nišu,
2. klijent aplikacija treba da omogući komponovanje osnovnih primitiva, u cilju dobijanja željene geometrije. Treba da sadrži:

- grafički korisnički interfejs,
- izbor svih funkcija CADROT sistema,
- sredstva za komunikaciju sa serverom, u cilju stalnog ažuriranja modela proizvoda u OO bazi podataka,
- vizuelizaciju proizvoda na ekranu.

3. Server treba da omogući distribuisanje objektnog modela izabranog proizvoda ka klijentima, proveru prava pristupa odredjenom proizvodu, ili nekoj njegovoj komponenti i da održava komunikaciju klijenta sa perzistentnim, "živim" objektima u OO bazi.

Iako se ovi funkcionalni zahtevi mogu realizovati primenom različitih tehnologija, pažljivim izborom se sistemu može produžiti vek i smanjiti rizik od zastarevanja. Njegova modularna priroda nameće potrebu za dodatnim oprezom, jer je izvesno da će se sistem stalno dograđivati u cilju povećanja njegove

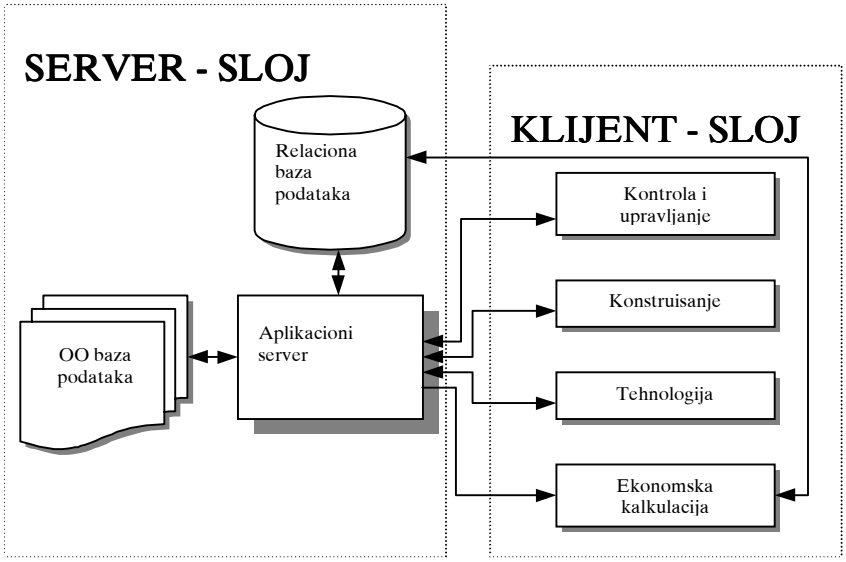
funkcionalnosti, čak i kad on postane potpuno operativan. Na osnovu istraživanja stanja na tržištu tehnologija za objektno orijentisano i distribuirano programiranje, usvojeni su sledeći tehnološki zahtevi:

1. isključivo korišćenje programskog jezika Java, na nivou klijenta. Klijent treba da bude Java aplet, koji će distribuirati web server. Jedini preduslov za korišćenje klijenta treba da bude instaliran web browser;
2. server je, u ovoj iteraciji sistema, implementiran u programskom jeziku Java;
3. komunikacija između klijent apleta i servera se odvija putem IIOP (Internet Inter-ORB Protocol) protokola, odnosno preko CORBA standarda za komunikaciju;
4. arhiviranje modela proizvoda se vrši u objektno orijentisanom sistemu za upravljanje bazama podataka ObjectStore.

3. JEDAN PRISTUP REALIZACIJI SISTEMA ZA SIMULTANO PROJEKTOVANJE

Da bi se proces projektovanja lakše modelirao, podeljen je najpre, na četiri *sfere* (slika 3.1). Definisano je da se u svakoj od njih, osim u sferi ekonomske kalkulacije, model proizvoda može *ažurirati* na specifičan način. Ove četiri sfere odredjuju četiri različita tipa klijent aplikacija.

Tok informacija kroz sistem je prikazan strelicama. Svaki od četiri tipa klijenta je povezan sa aplikacionim serverom tokom informacija, specifičnih za svaku sferu projektovanja. Tokovi informacija se odnose na *čitanje, zapis* i *izmenu* modela proizvoda.



Slika 3.1 Tokovi informacija u integrisanom sistemu za simultano projektovanje proizvoda

Proces projektovanja proizvoda započinje u sferi kontrole i upravljanja. Na aplikacionom serveru se otvara *nalog za projektovanje proizvoda*, odnosno instancira se klasa Proizvod. U okviru otvaranja naloga proizvoda, definišu se i neki opšti atributi objekta klase Proizvod i otvaraju nalozi na aplikacionom serveru za *nosioce radnih grupa*.

Zadatak nosioca radne grupe za konstruisanje je da, na sastanku sa razvojnim timom definiše koncept konstrukcije sklopa ili podsklopa, na osnovu definisanih opštih atributa, *baze znanja* preduzeća i raspoloživih *resursa*. Na osnovu koncepta, definiše se sastavnica proizvoda. Za jedan ili više elemenata sastavnice, nosioc otvara naloge za konstruisanje sa definisanim *pravima pristupa i izmene* odgovarajućoj sferi informacija. Rad u svakoj sferi projektovanja obavljaju *radne grupe*, koje određuju nosioci. Svaki član radne grupe ima svoje *radno mesto*, koje predstavlja računar, na kome se pokreće specifična klijent aplikacija.

U okviru zadatka, nosioc radne grupe, na osnovu sastavnice sklopa, definiše i *granične uslove* dela. Granični uslovi predstavljaju niz deklaracija kojima se, opisnim jezikom definišu ograničenja za konstrukciju dela s obzirom na zahteve funkcionalnosti i poziciju unutar sklopa (oblik, dimenzije, tolerancije). Granični uslovi se odnose na *funkcionalne elemente*, odnosno površine, unutar modela proizvoda, dok ostale elemente rotacionog dela možemo nazvati *konstruktivnim*.

Rad članova radne grupe za konstruisanje se odvija simultano, pri čemu svaki član od nosioca dobija odgovarajući zadatak. Zadatak se odnosi na konstruisanje jednog dela i sadrži i granične uslove, definisane za taj deo. Preko klijent aplikacije, svaki član radne grupe ima *pravo čitanja i izmene* za deo za koji mu je otvoren nalog, i *pravo čitanja* za ostale delove sklopa ili podsklopa. Preko klijent aplikacije je omogućen i pristup bazi znanja preduzeća.

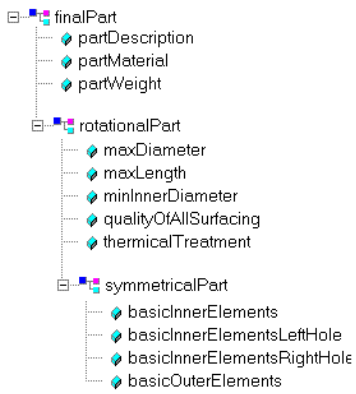
4. USVOJENO PROGRAMSKO OKRUŽENJE INTEGRISANOG SISTEMA ZA SIMULTANO PROJEKTOVANJE

Na osnovu analize postavljenog problema i istraživanja postojećih tehnoloških sredstava, usvojeno je programsko okruženje za razvoj integrisanog sistema za simultano projektovanje. Programsko okruženje se sastoji od objektno orijentisanog modela proizvoda, sistema za upravljanje OO bazama podataka, CORBA aplikacionog interfejsa za distribuiranje objekata i JAVA programskog jezika za implementaciju thin klijenta koji se može izvršavati u browseru i server aplikacije.

4.1 Objektno orijentisani model proizvoda

Nakon proučavanja korisničkih zahteva, definisanja ciljeva, obavljena objektna analiza rezultuje objektnim, funkcionalnim i dinamičkim modelom. Objektni model se sadrži od uočenih objekata sa svojim atributima, klasifikovanim unutar objektne hijerarhije nasleđivanja. Funkcionalni model opisuje ponašanje objekata, odnosno, definiše njihove metode. Dinamički model definiše stanja objekata izazvana različitim stimulansima, tokom izvršenja aplikacije.

Na osnovu ovih definisanih modela, vrši se njihova implementacija, odnosno, razvoj svih komponenata sistema, klasa i relacija.



Slika 4.1 Stablo osnovnih klasa objektnog modela proizvoda sa svojim atributima

U prvoj fazi modeliranja sistema za simultano projektovanje u objektno orijentisanom okruženju usvojen je *redukovani model* proizvoda. Prvo usvojeno ograničenje je da se modeliranje vrši na nivou mašinskog dela *integralne* geometrije, čime se isključuju podsklopovi i sklopovi. U ovoj fazi, podržan je samo model rotacionog, odnosno, osnosimetričnog dela, sa dodatnim nerotacionim elementima.

Na istom nivou u hijerarhiji (nasleđuje klasu *Object*) nalaze se tehnički elementi (oblici, forme), koji se mogu podeliti na *konstruktivne* i *tehnološke*. Na slici je prikazano stablo klasa oblika, sa roditeljskom klasom *partFeature*, sa svim atributima. Klase koje neposredno nasledjuju klasu *partFeature*, su klasa osnovnih elemenata (*basicElement*), dodatnih rotacionih elemenata (*additionalRotElement*) i dodatnih nerotacionih elemenata (*additionalNonConcentricElement*).

4.2 Objektno orijentisani sistemi za upravljanje bazama podataka

Elementarni zahtev koji OO sistem za upravljanje bazama podataka treba da ispuni je da omogući definisanje složenih objekata, korišćenjem klasa, atributa i metoda, uz poštovanje svih objektnih paradigmi, kao što su nasleđivanje, učaurenje itd. Funkcionalnost koju OO baza dobija ovim osobinama je nasleđena od ostalih OO tehnologija i predstavlja jedinstvenu osnovu svih objektnih sistema za upravljanje bazama podataka.

Sve informacije u OO bazi se organizuju i održavaju u obliku "živih", realnih entiteta, modeliranih u OO okruženju, za razliku od relacionih baza, kod kojih se realni entiteti dekomponuju u primitive (npr., brojevi), koji se mogu struktuirati unutar tabele.

S obzirom na definisane zahteve sistema za simultano projektovanje, u ovom delu je data kratka diskusija o nekim aspektima rada OODBMS sistema, koji su od naročitog značaja za njegovu realizaciju.

4.2.1 Perzistentni objekti

Jedan od osnovnih ciljeva korišćenja OO baza je potreba da određeni objekti budu stalno na raspolaganju nekoj spoljnoj aplikaciji, bez obzira na to da li se ona trenutno izvršava, ili ne, odnosno, da se objektni načine *perzistentnim*. Osobina perzistentnosti se može zasnivati na određenoj klasi, gde se može definisati da je svaki objekt koji je nasledjuje - perzistentan. Alternativni koncepti su jedinstveno definisanje osobine perzistentnosti za svaki objekat i perzistentnost svih objekata kojima se pristupa sa objekta koji je perzistentan.

Jedna od sličnih osobina objekata je i njegova *egzistencija*. OODBMS može posedovati sredstva za eksplicitno brisanje objekata iz baze, pri čemu se, istovremeno moraju i obrisati sve reference drugih objekata na njega. Jedna od alternativnih strategija može biti i održavanje objekta u bazi dok god postoje reference na njega.

4.2.2 Čuvanje verzija objekata

Većina OODBMS sistema podržavaju koncept po kojem se jedan objekat može predstaviti sa više različitih *verzija*, ili stanja. Postoje dva načina održavanja više verzija objekta, koji se primenjuju zavisno od potreba aplikacije koja koristi podatke iz baze:

- Linearno čuvanje verzija omogućava čuvanje prethodne verzije objekta koji je upravo izmenjen. čuvanje prethodnih verzija nekog

objekta je od krucijalne važnosti za razne tehnike projektovanja, kod kojih postoji mogućnost da projektant može da se vrati na bilo koju tačku u procesu projektovanja, bilo da bi ispravio grešku nastalu u sledećim koracima, ili iz nekog drugog razloga.

- Grananje verzija objekata je značajno kod simultanog rada na određenom objektu, gde više korisnika može pristupiti nekom objektu i promeniti njegovo stanje. Pod pretpostavkom da se posao svih korisnika objekta zasniva na njegovoj konzistentnoj, nepromenljivoj, osnovnoj verziji, generiše se linearno više grana verzija, nakon čega se, u odredjenom trenutku, podaci iz ovih grana integrišu u jedinstveni objekt, koji sadrži sve atribute i stanja, definisane u procesu rada na objektu svih korisnika. Na osnovu skupa odgovarajućih verzija, u svakom trenutku se može odrediti *konfiguracija* tog objekta.

Iako se čuvanje verzija može kontrolisati na najnižem programskom nivou, većina OODBMS sistema podržava mogućnost linearnog čuvanja verzija, tako što, nakon svake izmene objekta, kreira njegovu novu verziju.

4.2.3 Načini upravljanja sistemom

Upravljanje objektima, skladištenim unutar OODBMS sistema se može vršiti na pasivan ili aktivan način.

Kod pasivnog upravljanja imamo slučaj da baza podataka, odnosno, objekti u njoj, u sebi ne sadrže implementaciju metoda definisanih u klasama. Implementacije ovih metoda se nalaze u samoj aplikaciji koja radi sa objektima. Izvršenje aplikacije, dakle, rezultuje u transferu svakog objekta koji učestvuje u radu aplikacije iz procesa koji se izvršava unutar OODBMS, u proces koji se izvršava unutar adresnog prostora klijent aplikacije. Ovaj saobraćaj je potpuno transparentan za programera.

Aktivno upravljanje podrazumeva da se u OO bazi podataka skladišti implementacija ponašanja objekta. Ovo omogućava objektima da izvršavaju svoje metode unutar server procesa, na OODBMS sistemu. Najvažnije prednosti ovakvog upravljanja sistemom su:

- objektima se može pristupati i manipulisati od strane proceduralnih programa, preko standardnih programskih interfejsa;
- metode svih objekata su skladištene na jednom mestu, zbog čega možemo biti sigurni da sve aplikacije koriste najsvežije metode;
- svaki objekat koji učestvuje u radu aplikacije ne mora da bude prebačen u adresni prostor aplikacije.

4.2.4 Paralelan rad

Sistemi za upravljanje bazama podataka poseduju mehanizme za upravljanje paralelnim radom više korisnika baze tako što osiguravaju da paralelan pristup podacima ne izazove razna nekonzistentna stanja baze ili same aplikacije koja joj pristupa, usled pogrešnih pretpostavki zbog parcijalno ažuriranih podataka koje aplikacija "vidi".

Tokom takozvanih "dugih" transakcija, postoji mogućnost da se one prekinu, usled prekida rada celog sistema ili nekih drugih lokalnih problema. Ovaj scenario može izazvati nekonzistentno stanje baze, usled njenog delimičnog ažuriranja. Postoje mehanizmi koji sprečavaju da se ovo desi. Unutar jednog logičkog skupa transakcija, zavisnih jednih od drugih, ili će se izvršiti svaka njegova fizička komponenta ili neće ni jedna. Poštovanje ovog pravila omogućuje stalno logički konzistentno stanje baze.

Jedna od strategija može biti dozvola više čitanja i samo jednog ažuriranja istovremeno, s tim što se klijentu koji čita informaciju stavlja do znanja da se informacija upravo ažurira. Svaka klijent aplikacija dobija svoju lokalnu kopiju objekta. Ako neka od njih vrši izmenu objekta, objekat se ne skladišti na server, sve dok se ne završe sve paralelne transakcije čitanja. Sa druge strane, svi klijenti koji trenutno čitaju podatke, dobijaju informaciju da se objekat trenutno ažurira tako da znaju da pred sobom imaju zastarelu informaciju. Kada se svi paralelni procesi čitanja završe, može se završiti i proces izmene, tako što se izmenjeni objekat permanentno sačuva u bazi.

4.3 Distribuirani objekti - CORBA

U ovom trenutku, na tržištu modela za distribuirano programiranje, glavni rivali su CORBA specifikacija OMG asocijacije, DCOM model kompanije Microsoft i RMI Java aplikacioni interfejs, sastavni deo Java programskog jezika. Pored efikasne i pouzdane komunikacije između distribuiranih objekata, od izabrane arhitekture se zahteva i potpuno funkcionalan rad u heterogenom okruženju različitih softverskih i hardverskih platformi.

Na osnovu istraživanja obavljenog u cilju poređenja postojećih tehnologija za distribuirano programiranje, zaključeno je da CORBA tehnologija predstavlja najzrelije rešenje za rad distribuiranih objekata u sistemu za simultano projektovanje.

Osnovne prednosti CORBA arhitekture se baziraju na njenoj univerzalnosti. CORBA predstavlja specifikaciju arhitekture za rad sa distribuiranim objektima. Drugim rečima, ona definiše samo način, na koji se vrši komunikacija između fizički udaljenih bjekata. Sama implementacija se vrši na osnovu definisanih principa, i to je zadatak raznih razvojnih timova (Inprise Visibroker, IONA Orbix, Joe ...).

Ovaj princip projektovanja distribuirane arhitekture, zasnovan na opštim, ali precizno utvrđenim pravilima omogućava CORBI jednaku funkcionalnost na svim operativnim sistemima i hardverskim platformama,

kao i univerzalni pristup korišćenjem svih objektno orijentisanih programskih jezika. CORBA ORBovi su razvijeni za sve važne operativne sisteme, uključujući i veći broj Microsoftovih operativnih sistema od onog koji podržava DCOM - Microsoftov proizvod!

Iako Microsoft ima ugovore sa raznim kompanijama za razvoj DCOM modela na VMS i nekoliko Unix sistema, DCOM model je u ovom trenutku potpuno funkcionalan samo na operativnim sistemima Windows 95 i Windows NT 4. Šta više, s obzirom da se komunikacija DCOM objekata odvija preko Microsoftove implementacije RPC protokola, određene teškoće u implementiranju mehanizama sigurnosti dovode u pitanje bezbednu komunikaciju DCOM objekata na Windows 95 operativnom sistemu. Takođe, činjenica da je razvoj DCOM modela centralizovan i strogo ograničen licencnim pravima Microsofta, onemogućava slobodan rad nezavisnih proizvođača na dodatnim opcijama i alatima, koji programerima mogu da budu veoma korisni.

Konačno, u odnosu na CORBA specifikaciju, DCOM model još uvek ne raspolaže svim servisima koji mogu poboljšati funkcionalnost i komfor u radu. Jedan primer za to su servisi poruka i transakcija, koje Microsoft u ovom trenutku uvodi u DCOM standard. Ovi servisi su deo CORBA standarda još od 1994 godine, kada je definisana CORBA 2.0 specifikacija.

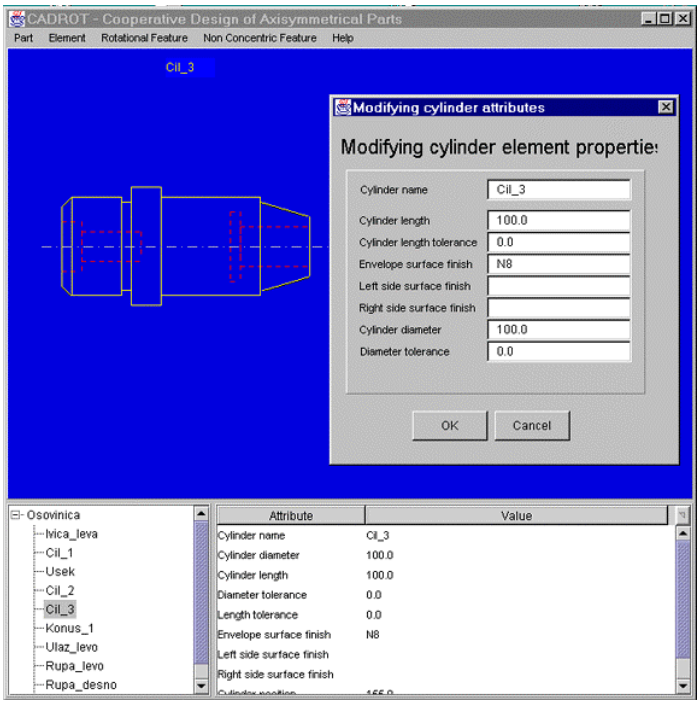
RMI aplikacioni interfejs predstavlja rešenje za rad sa distribuiranim Java objektima. Iako se Java programi mogu izvršavati na velikom broju operativnih sistema, funkcionalnost RMI metoda je ograničena obavezom korišćenja Java programa i na klijent i na server strani. Sa jedne strane, ovo omogućava upotrebu već napisanih programa i procedura, a sa druge, ograničava performanse sistema, s obzirom na to da se Java programski jezik interpretira, a ne izvršava. Čak i sa JIT kompajlerima, Java programi se ne mogu izvršavati brže od programa, napisanih u drugim objektno orijentisanim jezicima, kao što je C++. Ovo onemogućava upotrebu u sistemima sa intenzivnim procesorskim radom, kod kojih su performanse programa krucijalne, kao što su CAD sistemi.

4.4 Portabilni JAVA klijenti

U okviru portabilnog JAVA klijenta, potrebno je implementirati grafički korisnički interfejs, koji će omogućiti korisniku sistema rad prema deklarisanim funkcionalnim zahtevima, metode za vizuelizaciju dela i metode za komunikaciju sa serverom (stub kod) i OO bazom podataka.

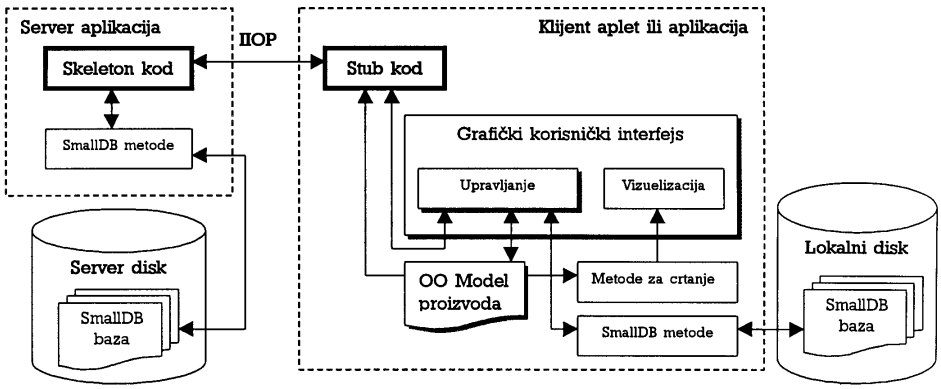
Pored ovoga, u memorijskom prostoru klijenta se instanciraju klasa *finalPart* i klase *partFeature*. Instancirani objekti se brišu iz memorijskog prostora, tek nakon završetka rada na delu. U međuvremenu, ažuriranje modela proizvoda u OO bazi se može vršiti eksplicitnim zadavanjem komande Save, ili pre nastalog konflikta, u slučaju da konstruktor naruši ograničenja, zadata u fazi apstrakcije dela, od strane šefa radne grupe.

Na slici 4.2, prikazani su neki od elemenata grafičkog korisničkog interfejsa, sa maskom za izmenu parametara konstruktivnog elementa - cilindar.



Slika 4.2 Maska za izmenu atributa konstruktivnog elementa - cilindar

Na slici 4.3 je prikazana šema komunikacionih puteva između pojedinačnih elemenata integrisanog sistema.



4.3. Šema komunikacionih puteva između pojedinačnih elemenata integrisanog sistema

5. ZAKLJUČAK

Kao rezultat ovog rada nastao je demo sistem za simultano konstruisanje osnosimetričnih delova. Namera autora je bila da testira i opravda korišćenje određenih tehnoloških sredstava, za koja se, u prethodnom istraživanju ispostavilo da su superiorna u odnosu na predložene alternative. Programski jezik Java, sa mnoštvom dodatnih biblioteka omogućava jednostavno programiranje klijent aplikacije ili apleta, naročito uz pomoć nekog od vizuelnih alata za pomoć u realizaciji korisničkog interfejsa. Korišćeni vizuelni alat - Borland JBuilder 2.0 se pokazao kao najozbiljniji izbor, u okruženju suparnika Symantec Visual Café i Microsoft J++.

Osnovni problem u radu na klijent strani je bila još uvek loša podrška radu sa grafičkim elementima, od strane osnovne JAVA JDK specifikacije. Iako verzija 1.2 JDK omogućava rad sa 2D i 3D grafičkim primitivima kao i manipulaciju s njima, iskustva autora upućuju na nepremošćenu razliku između stvarne i deklarisane podrške određenih web browsera JDK specifikaciji.

S obzirom na to da se, u klijent aplikaciji intenzivno koriste grafičke komponente, ona je razvijena korišćenjem Borland-ovih i Sun-ovih Swing klasa, koje se isporučuju u okviru vizuelnog JBuilder alata.

Već navedene prednosti CORBA standarda su jasno opredelile pravac razvoja komunikacionog puta za objekte koji sadrže model proizvoda. Jednostavno programiranje interfejsa za objekte, uz pomoć IDL jezika i lako inkorporiranje CORBA funkcija u Java program, rezultuju mogućnostima za rad u heterogenom okruženju sa moćnim performansama.

Konačno, back-end objektno orijentisana baza podataka omogućava skladištenje objekata na jednostavan način, bez prethodnog prilagođenja modela proizvoda paradigmama relacionog modela. Pored toga, komercijalni OODBMS sistemi omogućavaju i jednostavnu realizaciju nekih funkcionalnih osobina ovakvog sistema kao što su čuvanje verzija objekata, autorizacija korisnika, administracija i unutrašnja organizacija arhive objekata.

Što se tiče funkcionalnih osobina sistema za simultano konstruisanje, predložen je modularni koncept, koji razgraničuje uloge određenih inžinjerskih specijalnosti u procesu projektovanja, pri čemu se odvija i paralelan rad unutar domena jedne specijalnosti, kao što je, npr. konstruisanje. Pritom je naglašena potreba da se jasno definiše najprostija celina proizvoda, u čijem stvaranju učestvuje samo jedan inžinjer. Što je ona jasnije definisana, manji je rizik od narušavanja integriteta objektnog modela, kao i različitih konflikata.

6. LITERATURA

1. Milan Zdravković, "Realizacija sistema za simultano projektovanje osnosimetričnih delova u objektno orijentisanom okruženju", Seminarski rad, Mašinski fakultet Univerziteta u Nišu, 1999
2. Dragan Domazet, “Predlog projekta: Simultano projektovanje proizvoda korišćenjem univerzalnih mobilnih korisničkih programa (Web/CADROT sistem za simultano konstruisanje rotacionih delova posredstvom Interneta”, poverljivo, Singapore, 1998
3. Dragan Mišić, “Simultano projektovanje rotacionih delova i tehnoloških procesa obrade rezanjem”, Magistarski rad, Mašinski Fakultet Univerziteta u Nišu, Niš, 1998
4. Alan Pope, “The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture”, http://www.qds.com, 1997
5. Richard Grimes, “Professional DCOM Programing”, Wrox Press Ltd., Birmingham, 1997
6. Gregory McFarland, Andy Rudmik, “Object-Oriented Database Management Systems – A Critical Review/Technology Assessment”, Data & Analysis Center for Software - Department of Defense Information Analysis Center, http://www.dacs.dtic.mil, 1998