RAZVOJ INFORMACIONIH SISTEMA U INTERNET OKRUŽENJU KORIŠĆENJEM SOFTVERSKIH KOMPONENTI SA POSEBNIM OSVRTOM NA PRIMENU U VOJNOJ ORGANIZACIJI

Pejanović J. *Miloš,* Generalštab Vojske Srbije, Uprava za telekomunikacije i informatiku (J-6), Centar za komandno-informacione sisteme i informatičku podršku, Beograd

UDK: 004.738.5:355.1

Sažetak:

Razvoj personalnih računara i internet tehnologije uzrokuje neprekidne promene u metodološkim pristupima i konceptima razvoja informacionih[1] sistema. Većina postojećih informacionih sistema, zbog heterogenosti, ima problem integrisanja podsistema. Radi prevazilaženja ovog problema proizvođači softvera nude različita rešenja. U ovom radu se istražuju pristupi i predlaže optimalan put, sa posebnim osvrtom na primenu u vojnoj organizaciji.

Primenom savremenih pristupa u razvoju informacionih sistema na konceptu distribuiranih komponentnih sistema dolazi se do skupa predloženih rešenja različitih proizvođača. Rešenja se odnose na mehanizme koji bi trebalo da omoguće da komponente pisane u različitim jezicima međusobno sarađuju u heterogenim sistemima koji se nalaze u različitim čvorovima u mreži računara.

U radu je prikazan koncept komponentnih distribuiranih informacionih sistema u internet tehnologiji i njihove mogućnosti. Na kraju se predlaže rešenje sa specifičnostima implementacionog okruženja u vojnoj organizaciji.

Ključne reči: *internet tehnologija, distribuirani sistemi, komponentni sistemi, softverske komponente, vojna organizacija, heterogeni sistem, metodološki pristup.*

Uvod

Danas je nezamisliv razvoj distribuiranih informacionih sistema bez računarskih mreža i koncepta objektnoorijentisanog razvoja aplikacija. Time se uvode konkretniji standardi u računarskoj tehnologiji,

[1] U užem smislu, u ovom radu se misli na poslovne ili bazične informacione sisteme.

koji omogućavaju povezivanje udaljenih komponenti distribuiranog hardvera ili softvera. Prelazak sa strukturnog i modularnog programiranja u objektnoorijentisano programiranje redukuje cenu softvera. Takve *softverske komponente* integrišu se sa aplikacijama, a aplikacije u celoviti sistem. Zbog toga je razvoj *komponentnih modela* postao popularan poslednjih godina. Brz tehnološki razvoj, posebno pojava internet tehnologije, uzrokuje promene u navedenim metodološkim pristupima. Problem kod većine postojećih informacionih sistema jeste heterogenost i veliki broj neintegrisanih podsistema. Zbog toga se teži definisanju određenih standarda koji umanjuju ove probleme. Oni omogućavaju da se komponente softvera mogu koristiti ako su već jednom napisane ili da se mogu nabaviti od drugih proizvođača. Standard se odnosi na interfejs koji bi trebalo da omogući da distribuirane komponente, pisane u različitim jezicima, međusobno sarađuju. To je od velikog značaja za definisanje metodoloških pristupa u razvoju informacionih sistema kod velikih organizacionih sistema, kao što je vojna organizacija.

Posebno su značajna iskustva u primeni metoda i standarda u razvoju informacionih sistema sa primenom internet tehnologije. Pojava različitih proizvođača stvara dileme u opredeljenju za pristup u razvoju kompleksnijih informacionih sistema, posebno kada je reč o primeni u vojnoj organizaciji.

Pristup u razvoju informacionih sistema

Cilj procesa razvoja sistema je izrada kvalitetnog softvera koji će zadovoljiti potrebe korisnika. Pored toga, razvoj softvera treba da predstavlja proces s predvidivim vremenskim trajanjem i budžetom. Postoji više različitih procesa razvoja sistema, a primena određenog procesa zavisi od domena problema, tehnologije koja se može koristiti u implementaciji i sposobnosti i veštini projektnog tima.

U razvoju informacionih sistema važnu ulogu ima izbor odgovarajućih metoda i alata.[2] Za velike sisteme, kao što je, na primer, vojna organizacija, preporučuju se standardizovani postupci i metodologije za razvoj informacionih sistema. Postoje različiti metodološki pristupi u razvoju informacionih sistema: **sistemski integralni pristup razvoju** (od projektovanja, implementacije do uvođenja i održavanja) i **razvoj informacionih sistema kao tehničko-tehnološke strukture** (standardna mreža računara i servisa). **Kombinacija** ova dva pristupa vodi efikasnom rešenju u razvoju informacionih sistema, pri čemu treba imati u vidu **koncept „otvorenih sistema“** koji omogućavaju da različiti informatički standardi

[2] **Metode** predstavljaju skup postupaka, dok su **alati** sredstva u procesu razvoja informacionih sistema (na primer: dijagrami toka podataka, dijagrami objekti – veze, objektni dijagrami – dijagrami klasa, dijagrami promene stanja, Petrijeve mreže, itd.).

i servisi mogu da funkcionišu na tim sistemima (uključujući operativne sisteme, baze podataka, računarske komunikacije i korisničke interfejse).

Osnovni problem u razvoju složenijih informacionih sistema jeste savladavanje njihove složenosti. Pri tome, analiziraju se dva aspekta: **podela celokupnog razvoja na faze** i **dekompozicija sistema**. Podela razvoja na faze može se posmatrati kroz poznati konvencijalni životni ciklus informacionih sistema: **1. planiranje razvoja, 2. analiza i specifikacija zahteva, 3. projektovanje, 4. implementacija (kodiranje i testiranje), 5. uvođenje, 6. održavanje**.

Pojedinačne metode koriste skup alata, tehnika i aktivnosti za rešavanje problema u razvoju informacionih sistema. U pojedinim fazama životnog ciklusa informacionih sistema mogu se koristiti različiti alati i metode (tabeli 1).

Tabela 1[3]

Metode i alati za razvoj složenih informacionih sistema

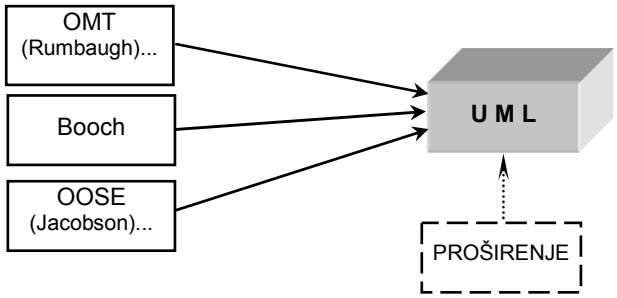
Konvencionalne metode	Objektne metode
1. Planiranje razvoja informacionih sistema (nije metoda već služi da razloži sistem na podsisteme)	
2. Analiza sistema	
Specifikacija	
Model procesa (strukturna sistem analiza)	Analiza „slučajeva korišćenja“ (*Use Case*)
Model podataka (model objekti – veze)	Objektno modeliranje. Dijagrami „klasa–objekti“
Specifikacija aplikacija (model aplikacionog složenog objekta)	Modeliranje dinamike sistema, preko dijagrama prelaza stanja (na primer, Petrijeve mreže)
Modeliranje funkcija (dijagrami toka podataka)	
3. Projektovanje	
Logičko i fizičko projektovanje	Projektovanje sistema, podela na podsisteme, definisanje arhitekture
Projektovanje programa	Projektovanje objekata
4. Implementacija (kodiranje i testiranje)	
5. Uvođenje	
6. Održavanje	

Aplikativno najpogodniji je **objektni pristup**. Kod različitih pristupa postoji problem izbora metoda projektovanja, odnosno stvaranja optimalnog metodološkog pristupa razvoja informacionih sistema. Takođe, postoji problem razvoja zbog parcijalne automatizacije u uslovima masovnog uvođenja personalnih računara. U takvim uslovima često se pristupa

[3] Izvor: B.Lazarević, S.Nešković, Objektnoorijentisana tansformaciona metoda razvoja informacionih sistema, SYMOPIS 90.

razvoju informacionih sistema kao infrastrukture i definisanju standarda, ne čekajući globalni projekat. Nakon izvesnog vremena postoji mogućnost da dođe do redundancija i haosa, zbog čega je potrebno ostvariti kontrolu i upravljati razvojem. Koordinacija nad lokalnim podsistemima, pored standarda, podrazumeva **rečnik podataka,** što znači postojanje „informacionog sistema o informacionim sistemima". Ovakav pristup u razvoju informacionih sistema zahteva dosta znanja.

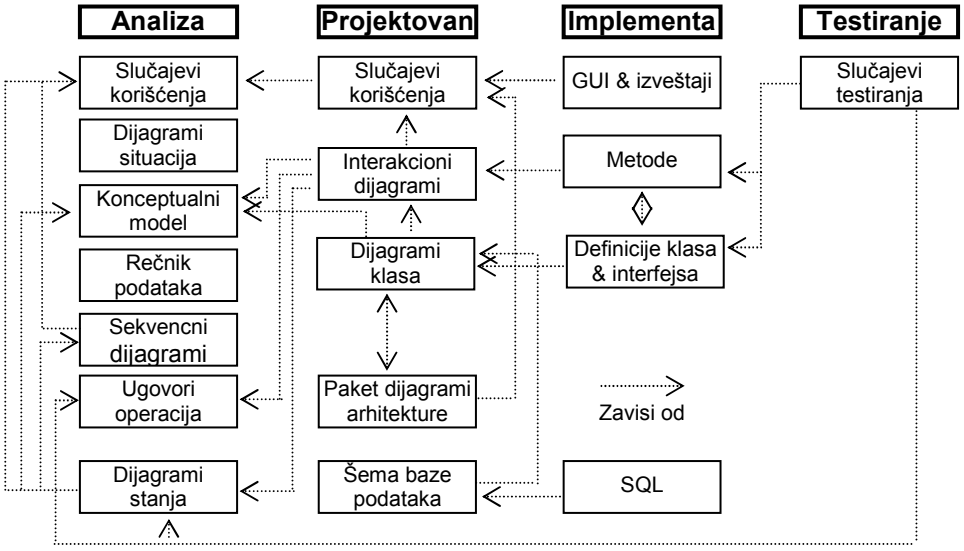
Postoje različite objektnoorijentisane metode koje predlažu različite pristupe objektnoorijentisanoj analizi i projektovanju informacionih sistema. Među njima su najpoznatije **OMT metod** *(Object Modeling Technique),* **Boochov metod i OOSE metod** (*Object – Oriented Software Engineering)* [4]. U OMT metodu posmatraju se tri različita modela: objektni, dinamički i funkcionalni. Oni predstavljaju različite poglede na sistem i međusobno su komplementarni. Ovaj metod daje prednost pristupu u kojem analiza sistema započinje definisanjem entiteta i formiranjem objektnog modela. Ovaj pristup naziva se **pristup orijentisan podacima** *(data driven approach).* U Boochovom metodu definisani su različiti dijagrami: dijagram klasa, dijagram objekata, dijagram interakcije objekata i dijagram prelaza stanja. Njima su predstavljene statičke i dinamičke karakteristike sistema. U ovom metodu posebno je naglašen onaj deo procesa razvoja sistema koji se odnosi na projektovanje i implementaciju sistema [5]. U OOSE metodu definisane su tri vrste objekata. To su objekti entiteta *(entity objects),* interfejs objekti *(interface objects)* i upravljački objekti *(control objects).* Za opis dinamike sistema uvedeni su slučajevi korišćenja *(use cases).* Ovaj metod daje prednost pristupu u kojem definisanje zahteva i analiza sistema započinju definisanjem slučajeva korišćenja i formiranjem modela slučajeva korišćenja. Ovaj pristup naziva se **pristup orijentisan slučajevima korišćenja** *(use case approach)* [4]. Uvidom u navedene objektnoorijentisane metode dolazi se do zaključka da sve te metode imaju jedan zajednički skup elemenata koji se koristi u modeliranju sistema (slika 1).



Slika 1 – Razvoj UML

Objedinjeni jezik modeliranja (*Unified Modeling Language – UML*) nastao je sredinom 90-ih,[4] prvenstveno objedinjavanjem tri pomenute metode, OMT metode, Boochove metode i OOSE metode. Analizom koncepata obuhvaćenih u objektnoorijentisanim metodama došlo se do skupa modela i elemenata modela koji se koriste pri analizi, projektovanju i dokumentovanju elemenata sistema.

Da bi se UML primenio na procese razvoja sistema definisana su proširenja UML-a: UML proširenje za proces razvoja softvera i UML proširenje za modeliranje poslovnih sistema. UML je široko primenljiv bez proširenja i uvode se u projekte samo ukoliko su neophodna za uvođenje novih oznaka i terminologije. UML proširenje oslanja se na UML metamodel. UML metamodel definiše koncepte i način njihovog korišćenja u modelima UML-a. UML proširenje predstavlja predefinisani skup stereotipova, označenih vrednosti i notacija, koji zajedno proširuju i oblikuju UML za primenu u specifičnom domenu ili procesu.



Slika 2 – Zavisnost između elemenata u fazama razvoja softvera[5]

UML sistem za opis procesa razvoja softvera sadrži više različitih, ali međusobno povezanih modela. Svaki model predstavlja odgovarajuću fazu životnog ciklusa sistema. Definisani su sledeći stereotipovi modela [4]:

– model slučajeva korišćenja,

[4] Prva verzija UML data je 1995. godine pod nazivom Unified Method, autora G. Boocha i J. Rumbauhga. Novije verzije prihvatile su i vodeće svetske kompanije u proizvodnji softvera, kao što su ORACLE, IBM, Digital Equipment, Microsoft i drugi.

[5] Metode razvoja softvera – Larmanova metoda, prof.dr Vidojko Ćirić, mr Siniša Vlajić, FON.

– model analize,
– model projektovanja, i
– model implementacije.

Metode razvoja informacionih sistema koje su prethodno opisane mogu se sistematizovati i opisati na jedinstven način pod nazivom Larmanova metoda [9], po pravilu kroz nekoliko faza: 1. specifikacija zahteva, 2. analiza, 3. projektovanje, 4. implementacija i 5. testiranje (slika 2).

Razvoj informacionih sistema u vojnoj organizaciji

Vojni organizacioni sistem je složen i sastoji se od hijerarhijski povezanih organizacionih jedinica. Svoje ciljeve i zadatke ostvaruje preko svojih upravljačkih funkcija. Delovanje tih funkcija zavisi isključivo od brzine, pouzdanosti u pristupu i obradi potrebnih informacija. U Vojsci[6] su do sada postojala parcijalna istraživanja iz oblasti razvoja informacionih sistema. Rezultati tih istraživanja poznati su stručnim službama u Vojsci, delimično se primenjuju u praksi i mogu se koristiti kao polazna osnova za istraživanja u razvoju vojnih informacionih sistema.

Razmatranjem pristupa u razvoju informacionih sistema u internet okruženju korišćenjem softverskih komponenti nameće se potreba da se u različitim uslovima i ograničenjima obezbedi maksimalna efikasnost organizacijskih sastava Vojske. To zahteva preispitivanje postojećih pristupa i metoda u razvoju informacionih sistema u vojnoj organizaciji i uvođenje novih i savremenih, koje mogu biti prihvaćene kao sopstveni standard. Problem razvoja informacionih sistema u internet okruženju korišćenjem softverskih komponenti svodi se na **iznalaženje načina određivanja skupa metodoloških postupaka, koncepata i pristupa iz repozitorijuma metoda koje podržavaju distribuirane informacione sisteme u internet tehnologiji**. To znači da je potrebno osmisliti metodološki pristup koji će biti adaptivan u skladu sa specifičnostima vojne organizacije.

Većina stranih oružanih snaga u razvoju svojih informacionih sistema koriste standardizovane postupke i metodologije koje se primenjuju u njihovom okruženju. U slučajevima kada to nije posebno definisano teži se korišćenju najsavremenijih metodoloških postupaka u skladu sa tehničko-tehnološkim razvojem. Ekonomski najrazvijenije zemlje, a time i njihove armije, vodeće su u razvoju i modernizaciji informacionih sistema. U prethodnim decenijama su informacije o njima bile relativno nedostupne, posebno u istočnoevropskim armijama. Danas je veoma izražen uticaj internet tehnologije na utvrđivanju standarda i metodologija u razvoju informacionih sistema. Time dolazi do mogućnosti prevazilaženja problema nastalih zbog razvoja mnoštva parcijalno razvijenih informacionih sistema, tako što se definiše standard koji omogućava komuniciranje različitih komponenti i povezuje ih u jedinstvenu globalnu mrežu.

[6] Vojska Srbije.

Na primer, američka vojska je publikovala standard za razvoj svojih tehničkih sistema i informacionih sistema pod nazivom *Joint Technical Architecture* **(*JTA*)**. Ovaj standard se bavi problemom razvoja tehničkih sistema uopšte. Postoji nekoliko verzija ovog dokumenta, koji se neprekidno usavršava. Osnovni cilj JTA jeste da obezbedi što veću efikasnost u izvršavanju vojnih operacija sa bilo kojim snagama, bilo gde u svetu. Kritična potreba je da se obezbedi takav sistem informacionih tehnologija koji će omogućiti sadejstvo i razmenu informacija. Proučavajući u praksi različita iskustva iz konflikata i operacija, došlo je do nove vizije u američkom ministarstvu za odbranu (*DoD*). Kao konceptualni obrazac javlja se JV 2010 (*Joint Vision 2010*), tako da JTA predstavlja krucijalni dokument ministarstva za odbranu u ostvarenju JV 2010. Ovaj dokument se stalno usavršava i otvoren je za pristup svim zainteresovanim. JTA obezbeđuje osnovu za neprekidnu i dobro povezanu međuoperatibilnost sistema DoD. JTA definiše područje servisa, interfejse i standarde (JTA elemente) primenjive u svim DoD sistemima, pri čemu su primenjivi u upravljanju, razvoju i lociranju novih ili postojećih sistema kroz DoD. JTA sadrži dva glavna dela: JTA jezgro i JTA dodatke. JTA jezgro sadrži minimalni skup JTA elemenata primenjivih u svim DoD sistemima za obezbeđenje međuoperatibilnosti. JTA dodatak sadrži dodatne JTA elemente primenjive u specifičnim funkcionalnim domenima (familijama sistema). Na primer, verzija 3.1. ovog dokumenta uključuje dodatak za C4ISR[7] domene, zatim *Combat Support domain, Modeling and Simulation domain i Weapon Systems domain*. Može sadržavati i poddomene neke domene [7]. JTA je komplementaran i konzistentan sa drugim DoD programima i inicijativama u razvoju „efektiva“ i međuoperatibilnih informacionih sistema. JTA je otvoreni dokument koji sadrži razvojne tehnologije i objedinjene bazične standarde i javno je dostupan.

Kada govorimo o standardima u razvoju vojnih informacionih sistema postoji veliki značaj primene TCP/IP protokola u razvoju računarskih mreža, što u suštini predstavlja razvoj telekomunikacione infrastrukture sa multiservisnim sposobnostima [8].

Komponentni distribuirani informacioni sistemi sa objektno-orijentisanim pristupom u internet okruženju

Informacioni sistemi koji imaju karakteristike distribuiranog hardvera, i/ili distribuiranog upravljanja i/ili distribuiranih podataka, predstavljaju **distribuirane informacione sisteme.** Distribuirani hardver podrazumeva dva ili više međusobno povezanih računara pomoću sistemske magistrale, lokalne mreže ili komunikacione mreže [1].

[7] *Command, Control, Communications, Computers, Intellgence, Surveillance, and Reconnaissance*

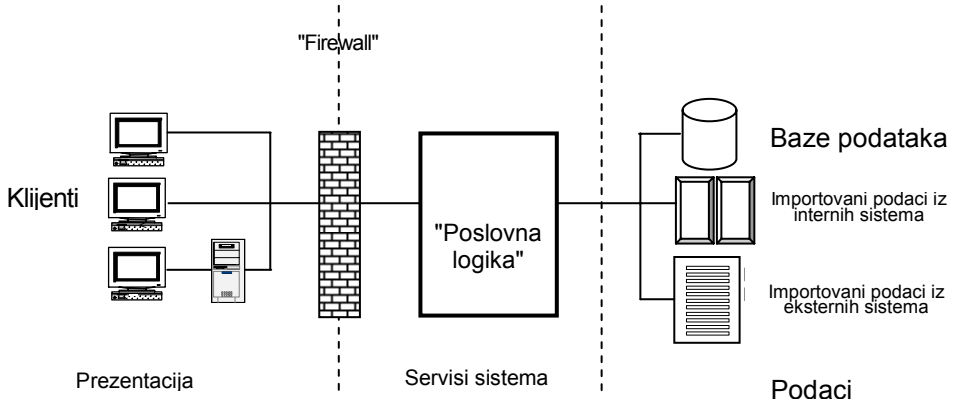
ISO/OSI[8] referentni model predstavlja polaznu osnovu pri povezivanju otvorenih sistema. Arhitektura referentnog modela služi kao osnova svih budućih razvoja standarda za potrebe distribuiranih informacionih sistema. U suštini, to su računarske mreže sa informacionim sistemima koji imaju karakteristike distribuiranih resursa.

Razvoj personalnih računara i komunikacionih tehnologija, pre svega **internet tehnologije,** uslovio je potrebu za nadogradnjom postojećih sistema i povezivanjem sa korisnicima sistema ili drugim sistemima u okviru i izvan vojne organizacije.

Razvoj globalne računarske mreže bazirane na internet tehnologiji i **klijent–server arhitekturi,** kao i njena masovna upotreba, bitno utiču na redefinisanje standarda za razvoj objektno orijentisanih distribuiranih informacionih sistema.

Pored postojećeg skupa metoda i tehnika za razvoj informacionih sistema postavlja se pitanje metodološkog pristupa razvoja u internet okruženju. Rešenje je nađeno u troslojnim *(*slika 3*)* i višeslojnim sistemima.

Na ovaj način ostvaruje se nezavisnost u smislu odvojenog razvoja klijentskih aplikacija, dodavanja novih operacija unutar poslovne logike ili intervencija na samoj bazi. Komponente aplikacije mogu biti distribuirane po različitim računarima: ulazimo u eru arhitekture distribuiranih objekata. **Distribuirani objekat** definiše se na jednom sistemu, a može se koristiti na drugom. U ovoj komunikaciji može učestvovati jedan ili više kompjutera. Cilj je da se obezbedi integracija servisa sa različitih platformi.

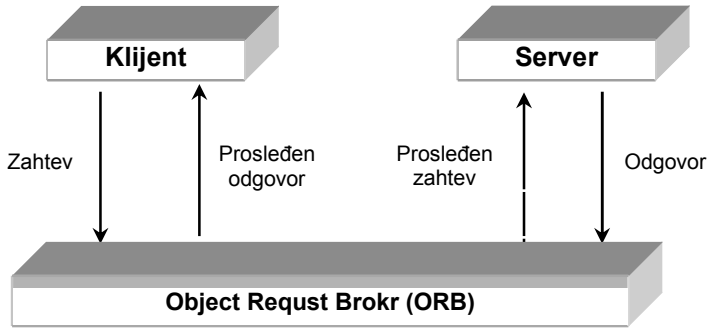


Slika 3 – Troslojna arhitektura

[8] Referentni model za povezivanje otvorenih sistema zasnovan je na konceptu hijerarhijske organizacije komunikacione arhitekture u sedam diskretnih slojeva.

Primenom distribuiranih objekata mogu se kreirati klijent/server aplikacije sa troslojnom ili višeslojnom arhitekturom, koje omogućavaju implementaciju sinhronih ili asinhronih rešenja za internet i intranet. Novi trendovi u primeni internet tehnologije postavili su nove zahteve pred serverske aplikacije koje moraju podržati transakcije u novom okruženju. Proširivanje uloge servera vodi stvaranju kompleksnijih aplikativnih servera. Transakcije se mogu izvršavati direktno ili svrstavajući se u red čekanja. U direktnim se uspostavlja sinhronizovana komunikacija između klijenta i servera, čime se dobija utisak da je uspostavljena direktna veza.[9] Noviji pristup koji se oslanja na CORBA model *(Common Object Request Broker Architecture)* i Object Request Broker (*ORB*) jeste primer direktnog izvršenja transakcija. S druge strane, pristigle transakcije se ne moraju izvršavati sinhronizovano, već mogu dolaziti u red transakcija na serveru i zatim se izvršavati po pravilima koja važe za taj red (npr. izvršenje po nastanku nekog događaja ili u nekom vremenskom trenutku i sl.). Isto važi i za poruke o rezultatima transakcija.

Transakcije koje se mogu izvršiti kompletno, na jednom čvoru mreže, nazivaju se lokalnim transakcijama. Priroda distribuiranih sistema uslovila je pojavu distribuiranih transakcija, za čije je izvršenje potrebno više čvorova mreže. One su kompleksnije i teže za implementaciju. Implementacija aplikacije zahteva upravljanje deljivim resursima servera (kao što je memorija ili procesorsko vreme) i upravljanje kontekstom. To je zahtev koji može usporiti razvoj aplikacije, jer bi se svaka transakcija programirala.



Slika 4 – Klijent–server komunikacija kroz ORB

Takve transakcije nazivaju se programirane transakcije (*eng. Programmatic transactions*). Aplikacije koje ih koriste su teške i skupe za održavanje. To je jedan od razloga zbog kojih je razvoj **komponentnih modela** postao popularan i doživeo veliki razvoj poslednjih godina. Komponente kojima je **implementiran mehanizam izvršavanja transakcija** na

[9] Tipičan primer ovakvog izvršavanja transakcija uočava se u CGI aplikacijama koje podržavaju web serveri.

ovaj način se mogu koristiti u izradi novih aplikacija. Komponentni model omogućava lakše izmene i održavanje, a istovremeno se skraćuje vreme potrebno za implementaciju, jer postoje različite gotove komponente koje se mogu iskoristiti. Ovakav pristup implementaciji transakcija naziva se deklarativnim pristupom. Programeri su oslobođeni brige o realizaciji mehanizama za kontrolu izvršenja transakcija. Značajni komponentni modeli realizovani su u tehnologijama COM+ (*eng. Component Object Model*), EJB (*eng. Enterprise Java Beans*), CORBA *(Common Object Request Broker Architecture)* i SOA *(*Service Oriented Architecture*)* [2].

Tehnologije za implementaciju komponentnih distribuiranih informacionih sistema sa objektnoorijentisanim pristupom u internet okruženju

S obzirom na to da se sistemi zasnovani na distribuiranim objektima razvijaju godinama unazad, na tržištu postoji niz različitih platformi (tehnologija) za implementaciju višeslojne arhitekture. Pri odabiru distribuirane tehnologije treba obratiti pažnju na sledeće parametre:

– prenosivost (portabilnost) klijentske i serverske platforme,
– prenosivost programskog jezika,
– performanse pri izvršavanju,
– jednostavnost razvoja,
– sigurnost.

Rezultati testiranja sa različitim komunikacionim modelima za udaljeno pozivanje komponenata rezultirali su sledećim tehnologijama:

– COM+/DCOML,
– CORBA/IIOP,
– EJB/RMI/IIOP,
– SOAP.

COM+ / DCOM

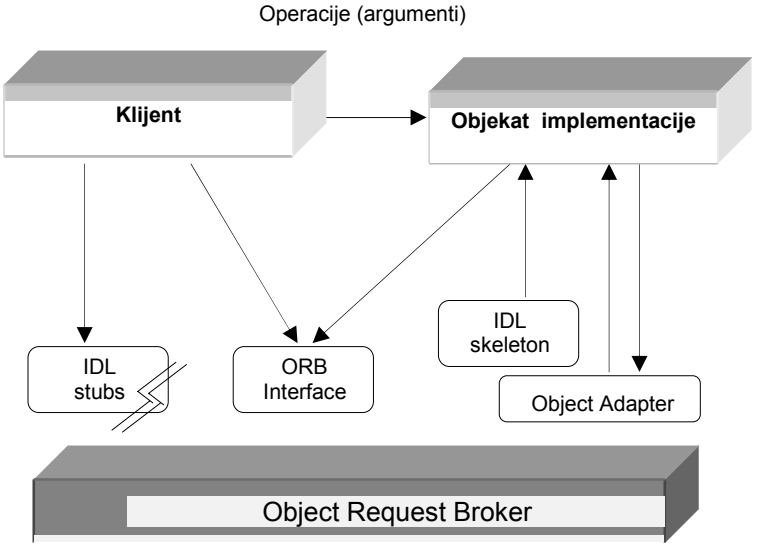
COM+ predstavlja evoluciju starije tehnologije COM *(Component Object Model).* COM je specifikacija za objekte koja definiše interfejs preko koga različiti objekti mogu da komuniciraju. COM je nezavisan od programskog jezika ukoliko implementira COM interfejs i teoretski može da se implementira na različitim operativnim sistemima. Međutim, ne podržava ih niko osim Microsoft Windowsa. Da bi se omogućilo da COM objekti sa različitih sistema međusobno razmenjuju informacije, COM specifikacija je proširena i nastao je DCOM (Distributed COM), koji poseduje znatno kompleksniji model konfiguracije i sigurnosti.

CORBA / IIOP

CORBA *(Common Object Request Broker Architecture)* jeste konkurentska specifikacija koju je kreirala Object Managment Group, OMG – grupe kompanija *(Object Managment Group)* koje razvijaju srednji sloj. CORBA je nezavisna od jezika i implementirana na većem broju platformi nego COM. Međutim, postoje nekompatibilnosti između implementacija različitih proizvođača. CORBA predstavlja magistralu objekata (*object bus)* koji omogućava klijentu da poziva metode sa objekta na serveru uz nezavisnost programskog jezika i lokacije objekta. Interakcija je omogućena preko ORB *(Object Request Brokers)* komponenata na klijentu i na serveru, a komunikacija se odvija preko IIOP *(Internet Inter-ORB Protocol)*[3].

EJB / RMI over IIOP

EJB *(Enterprise JavaBeans)* jeste specifikacija koju je izdao Sun Microsystems za Java Platformu. EJB je nezavisna od platforme, ali ne i od jezika. Svi EJB objekti moraju biti napisani u jeziku java. Za komunikaciju između različitih sistema EJB koristi varijantu IIOP, nazvanu RMI preko IIOP *(Remote Method Invocation over IIOP)*. RMI je protokol rezervisan samo za javu.



Slika 5 – Mogućnosti CORBA objekata definisane su pomoću IDL (Interface Definition language)

SOAP

SOAP *(Simple Object Access Protocol)* jeste kompletno kreiran na postojećim, proverenim i široko prihvaćenim tehnologijama kao što su HTTP i XML za prenos podataka između aplikacija. Pošto je XML univerzalni standard, sve platforme mogu da pristupe i obrade informaciju. Pristup različitim aplikacijama na raznim platformama sa SOAP-om postaje jednostavan, Java aplikacija na *Unix-u* jednostavno može da poziva metode COM objekta na *Windows* serveru. Klijentska aplikacija na *iMac-u* pristupa objektu na mainframe računaru. Sve to postaje transparentno i ne zahteva bilo kakvu posebnu administraciju.

Internet tehnologija omogućava korišćenje različitih servisa u komunikaciji između elemenata računarske mreže. Osnovna ideja je da se elementi informacionog sistema u Vojsci definišu kao komponente različitog softvera koje se mogu koristiti, ako su već napisane i postavljene na različite čvorove u mreži. Bez obzira na to da li su na istom serveru ili su distribuirani kroz mrežu, komponente komuniciraju. Osnovno pitanje u razvoju ovakvih informacionih sistema je tehnološko opredeljenje. U tabeli 2 predstavljen je uporedni prikaz prethodno opisanih tehnologija.

Tabela 2

Uporedni prikaz tehnologija za implementaciju komponentnih distribuiranih informacionih sistema

	DCOM	**IIOP**	**RMI/IIOP**	**SOAP**
Format	Binarni	Binarni	Binarni	Unicode
Platforma	Windows	Unix	Nezavisan	Nezavisan
Programski jezik	Nezavisan	Nezavisan	Java	Nezavisan
Izrada	Moderna (RAD)	Složen	Moderna (RAD)	Jednostavna
Sigurnost	Win NT Security	CORBA security service	JAVA security	HTTP/SSL, XML signature
Omogućava pristup kroz firewall	Ne	Ne	Ne	Da
Lociranje		Referenca		URL
Obrada grešaka		IDL Exception		SOAP Fault messages
Događaji	COM+ Events	CORBA Event service		N/A
Otkrivanje servisa		CORBA Naming/Traning service	RMI registry	UDDI
Opis tipova podataka		IDL		XML Schemas

Glavni **problem kod starijih protokola**, kao što su DCOM, IIOP i RMI/IIOP, nalazi se u nekompatibilnosti (teško prilagođavanje datih protoko-

la), tako da različite aplikacije međusobno ne mogu da komuniciraju. Druga bitna činjenica jeste da ne funkcionišu u prisustvu *firewalla*, što znači da aplikacije sa raznih lokacija ne mogu uvek međusobno da komuniciraju. Postavlja se pitanje smisla funkcija CORBA usvajanjem *Enterprise Java Beans* specifikacije, pošto u tom slučaju posedujete java aplikacije i na klijentu i na serveru. Pored toga, *Sun* je razvio RMI za komunikaciju java-java aplikacija, što je direktna konkurencija za CORBA i dovešće do daljih podela u ovom pristupu.

CORBA je specifikacija koju kompanije implementiraju po specifičnom zahtevu (želji), a pitanje komuniciranja između ORB-ova različitih proizvođača još nije u potpunosti rešeno. Ne postoji administriranje, sve se svodi na programiranje. Kada se želi povećati nivo sigurnosti sopstvenih komponenti potrebno je instalirati novi API i vršiti programiranje. CORBA je zasnovana na deljenim objektima (*shared objects*) koji čuvaju svoje podatke između dva poziva i tako zauzimaju memoriju. Broj klijenata koji istovremeno mogu da se povežu na server je sigurno mnogo manji nego kod *Microsofta*, gde se primenjuje princip deljenih podataka (*shared data*), a objekat između dva poziva oslobađa memoriju.

Web servisi predstavljaju osnovne gradivne blokove budućih informacionih sistema, a u suštini su aplikacije koje su raspoložive na mreži i koje mogu da urade ono što je u tom trenutku potrebno. Drugim rečima, to su resursi koji se adresiraju primenom URL-a koji vraćaju informaciju korisniku koji želi da ih koristi. Glavni komunikacioni protokol je SOAP, tj. XML preko HTTP-a. Osnovni pokretač ovih promena je XML, koji kroz svoju jednostavnost omogućuje praktičnu nezavisnost aplikacija i sistema jer je razumljiv i za čoveka i za mašinu. Web servisi objavljuju se na jedinstvenoj lokaciji i nude se kao usluge. Ukoliko se poseduju već napisane aplikacije koje nude stabilna i proverena rešenja, jednostavno je moguće objaviti aplikaciju kao uslugu korisnicima u vojnoj intranet mreži. Zbog te specifičnosti ova tehnologija se brzo raširila i u potpunosti se prihvata za implementaciju kako otvorenih, tako i zatvorenih informacionih sistema, kao što su vojni.

Odnos SOAP-CORBA i CORBA-SOAP prikazan je u tabeli 3.

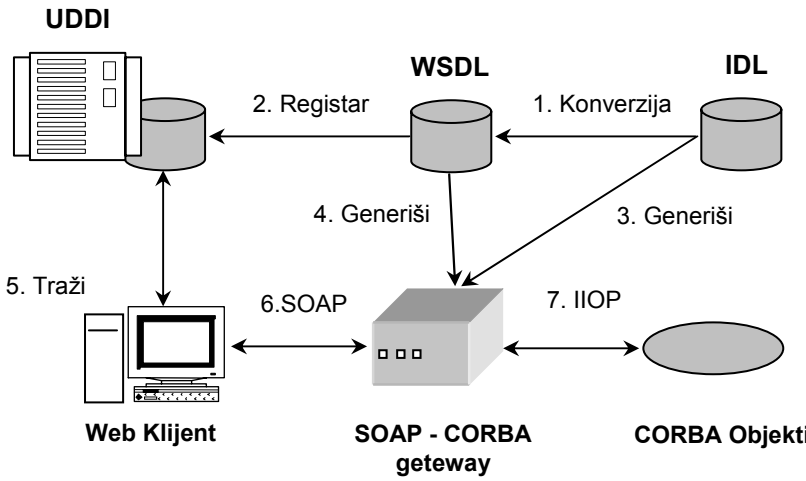
Tabela 3

Uporedni prikaz CORBA i SOAP

	CORBA	**SOAP**
Podrška raznim platformama	CORBA 1.0 teško je radila sa firewallovima, dok CORBA 2.0 radi preko TCP/IP protokola i teži uključivanju web-a.	Zasniva se na internet specifikacijama, te se lako nadograđuje na postojeće web okruženje koje je podržano na svakoj platformi.
Format podatak za prenos	Pošiljalac i primalac moraju da poseduju puno značenje konteksta poruke u prenosu, jer koristi binarno kodovanje podatka bez podataka o podacima (*metadata*)	Koristi XML koji omogućava jednostavnu obradu poruka u svakom koraku procesa. Jednostavno debagovanje i komprimovanje (zbog svoje ponovljivosti).
Interoperabilnost	Postoji više različitih implementacija CORBA, koje poseduju problem interoperabilnosti	Pošto je zasnovan na HTTP i XML omogućava jednostavnu interoperabilnost između različitih SOAP sistema.

	CORBA	SOAP
Skalabilnost	U CORBA postoji jednostavan mehanizam koji omogućuje ORB zahtev bez pamćenja stanja.	HTTP protokol ne pamti stanja između dva poziva.
	Izbor između mehanizma za pamćenje (stateful) ili bez pamćenja (stateless) zavisi od sistemske izvedbe.	SOAP server može da upravlja stanjem, korišćenjem klijentskih kolačića (*cookies)* ili specijalnih identifikacionih objekata sa SOAP pozivima.
		SOAP će posedovati mehanizam sesije kako bi se omogućili transakcioni zahtevi.
Životni ciklus	Određena instanca CORBA objekta identifikuje se referencom objekta instance.	SOAP indentifikuje objekte samo pomoću URL-a.
	CORBA se koristi za transparetnu komunikaciju između objekata aplikacije.	Životni vek SOAP objekta na serveru određen je vremenskim intervalom upravljanja stanjem.
Transportni protokoli	CORBA2 implementacija koristi Internet Inter-ORB protokol (IIOP), koji spada u GIOP (General Inter-ORB Protocol) preko TCP/IP.	HTTP protokol je definisan za prenos poziva metoda, a mogu se koristiti i SMTP, FTP.
	Dodatni portokol se zove DCE CIOP (DCE Common Inter-ORB Protocol) koji takođe podržan u CORBA2.	Drugi transportni protokoli kao što je SMTP nisu široko prihvaćeni pošto dopuštaju pozive u jednom smeru.
Sigurnost	Corba Security Service obezbeđuje sigurnosnu arhitekturu koja podržava veliki broj sigurnosnih politika kako bi se izašlo u susret različitim potrebama.	Na najnižem nivou SOAP poruke se mogu transportovati preko HTTPS kako bi se sprečilo prisluškivanje (snooping) i obezbedila identifikacija klijenta i servera.
	Servis specificira autentifikaciju, autorizaciju i enkripciju poruka.	Standard XML Key Management Specification (XKMS) obezbeđuje kvalitetniju sigurnost koja je neophodna za autentifikaciju korisnika određenog web servisa.
Upotreba	Programiranje CORBA sistema je kompleksno.	HTTP i XML olakšavaju implementiranje i debagovanje.
Vreme kompilacije	Kompilacija CORBA „Hello World“ traje više od minuta na 400 mhz/128 Mb.	SOAP ekvivalentu je potrebno manje od sekunde.
Prvi koraci	Sa CORBA je potrebno opsežno zananje, dugotrajno programiranje.	Sa SOAP-om je moguće uraditi nešto korisno za manje od 30 minuta.

Web pristup servisima izgrađenim u CORBA može se realizovati pomoću gatawaya koji automatski vrši konverziju između SOAP i CORBA IIOP poruka. Pošto su SOAP poruke čisti tekstualni dokumenti, tj. XML, konverzija se, u stvari, vrši između XML (slika 6).



Slika 6 – Šema (XML Schemas) i/ili DTDs (Document Type Definitions) prema odgovarajućem IDL elementima.

Set protokola web servisa stalno se razvija i koristi se da definiše, otkriva i implementira web servise. Osnova ovih protokola leži u sledeća četiri nivoa:

Service Transport: Ovaj nivo je odgovoran za prenos poruka između aplikacija i u njega su trenutno uključeni HTTP, SMTP, FTP, i novi protokoli kao što je BEEP – *Blocks Extensible Exchange Protocol*.

XML Messaging: Ovaj nivo je odgovoran za razumevanje poruka za razmenjivanje koje se implementiraju u XML formatu i trenutno se koristi XML-RPC (*XML – Remote Procedure Call*) i SOAP (*Simple Object Access Protocol*).

Service Description: Ovaj nivo definiše javni interfejs za određeni Web servis, i trenutno se opisuje kroz WSDL (*Web Service Description Language*).

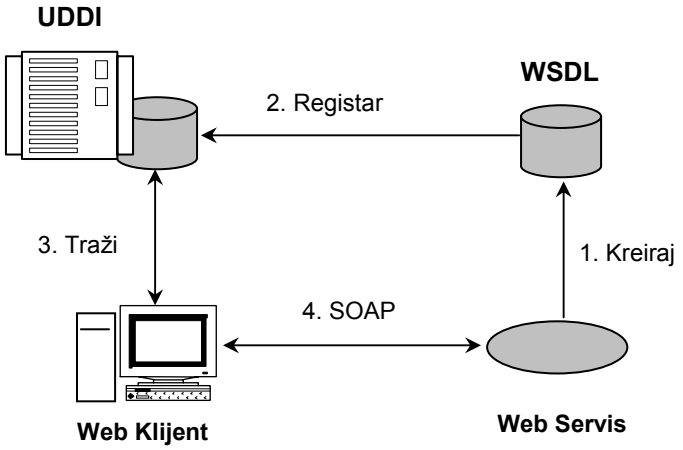
Service Discovery: Ovaj nivo je odgovoran za centralizovanje servisa u zajednički i jedinstveni registar koji obezbeđuje jednostavno objavljivanje i pronalaženje servisa. Otkrivanje servisa trenutno se obrađuje kroz UDDI (*Universal Description, Discovery, and Integration*).

Pored pomenutih, set sadrži nove protokole koji se još razvijaju, uključujući WSFL (*Web Services Flow Language*), SOAP-DSIG (*SOAP Security Extensions: Digital Signature*) i USML (*UDDI Search Markup Language*).

Nije potrebno razumevanje kompletnog seta protokola da bi se radilo sa Web servisima. Ukoliko se poznaju osnove HTTP-a, dovoljno je započeti sa *XML Messaging* nivoom.

Web servisi mogu se kreirati od već gotovih aplikacija ili od početka, pri čemu se programski jezici, gotove komponente i platforme mogu koristiti po svom izboru.

Nakon što se se usluga osmisli i kreira potrebno je (slika 7):



Slika 7 – Kreiranje web servisa

1. Opisati servis pomoću WSDL jezika.

WSDL je zasnovan na XML i predstavlja opis interfejsa web servisa koji treba da sadrži:

– opis interakcije koju servis nudi,
– opis argumenata i rezultata koji su uključeni u interakciju,
– adresu za lociranje servisa,
– komunikacioni protokol,
– format podataka koji se koristi u porukama.

Definisanje interfejsa je, u stvari, ugovor između servera i klijenta.

2. Registrovati servis na UDDI registru, internet lokaciji predviđenoj za objave servisa i time omogućiti i svom softveru da koristi tuđe usluge i zainteresovanim korisnicima da pronađu vaš servis.

UDDI predstavlja centralizovanu lokaciju koja obezbeđuje mehanizam za registrovanje i pronalaženje servisa. Koristi SOAP za komunikaciju i omogućuje klijentima da pronađu servis i serveru da ga objavi.

3. Omogućiti pristup svom web servisu posredstvom SOAP-a, koji koristi XML jezik za specificiranje pozivnih parametara i rezultata rada servisa.

SOAP predstavlja okruženje za razmenu poruka zasnovanih na XML-u u mreži, a služi se aplikativnim HTTP protokolom koji koriste web serveri.

Implementacija komponentnih distribuiranih informacionih sistema sa objektnoorijentisanim pristupom u internet okruženju

Brzina razvoja interneta dovela je do pojave mnogih tehnologija i standarda koji su istom tom brzinom i nestali. Sa aspekta istraživanja u ovom radu može se zaključiti da se izdvajaju dva pristupa i to kroz: J2EE (*Java 2 Platform Enterprise Edition*) okruženje i *Microsoftov.* NET skup tehnologija.

J2EE

J2EE je skup industrijskih standarda sa zadatkom da podrži i pojednostavi projektovanje, dizajn i implementaciju višeslojnih složenih web aplikacija u otvorenom distribuiranom okruženju koje predstavlja internet (Firma *Sun Microsystems* predvodi udruženje koje radi na daljem razvoju J2EE standarda, kao pogleda na distribuirane informacione sisteme zasnovane na korišćenju web servisa) [6].

Osnova svakog J2EE rešenja je programski jezik java, koji predstavlja platformski neutralan jezik, što znači da svaka platforma predstavlja njeno prirodno okruženje.

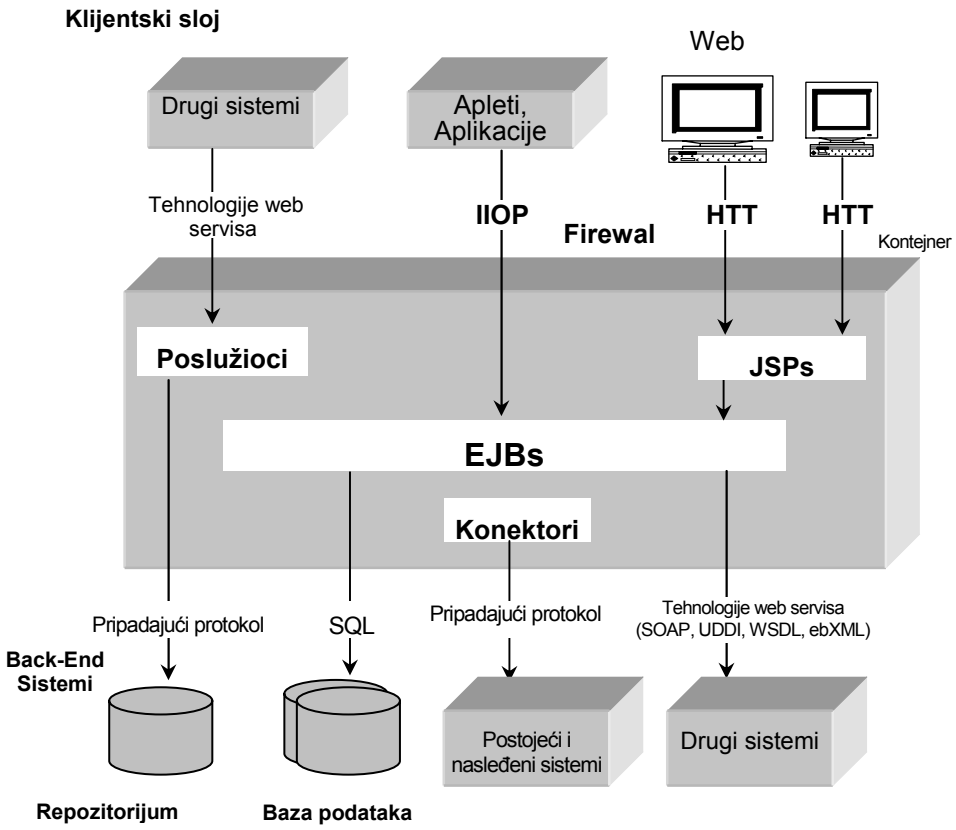
Razvojni inženjeri pišu izvorni kod na javi, koji se zatim prevodi u tzv. *bytecode*, koji predstavlja intermedijalni jezik koji mora biti interpretiran na svakoj platformi specifično pomoću JRE (*Java Runtime Enviroment*) da bi bio izvršen. Aplikacija mora biti pisana na javi da bi bila po J2EE standardu. Komponente koje sačinjavaju aplikaciju prevode se u bytecode i u vremenu izvršavanja interpretiraju pomoću JRE-a.

Sama J2EE aplikacija je zatvorena pomoću komponente *kontejner* koja pruža gotove odgovarajuće servise telu aplikacije, kao što su transakcioni servis, servis zaštite u transportu i servis trajnog skladištenja.

Poslovni sloj vrši obradu specificiranu logikom posla nad datim poslovnim podacima. Za veće J2EE aplikacije ima smisla koristiti EJB komponente, koje realizuju logiku posla i rad sa poslovnim podacima. Veza prema bazi podataka ostvaruje se preko JDBC (*Java Database Connectivity*), preko SQL/J ili zatečenih mehanizama korišćenjem JCA (*Java Connector Architecture*). Sa slike se vidi da se veza sa poslovnim partnerima ostvaruje i korišćenjem web servisa (SOAP, UDDI, WSDL, ebXML) preko java API-ja za XML (JAX API).

Drugi sistemi mogu se povezati na J2EE aplikaciju preko web servis tehnologija. *Poslužilac (Servlet),* kao specifičnost java okruženja, može prihvatiti zahtev drugog sistema za web servisom. Poslužilac koristi JAX API da bi obradio i izvršio zahtev web servisa.

Sa slike 8 se vidi da J2EE model podržava i rad sa tzv.„debelim“ (*thick*) klijentima, na primer, preko EJB komponente sa apletima ili drugim aplikacijama korišćenjem internet inter-ORB protokolom (IIOP). Web čitači i uređaji mobilne telefonije povezuju se na J2EE aplikaciju preko *JavaServer Pages* (JSP), koja ima zadatak da pripremi HTML ili WML stranicu, koja se zatim šalje čitaču [6].



Slika 8 – Koncept *Java 2 Platform Enterprise Edition*

.NET

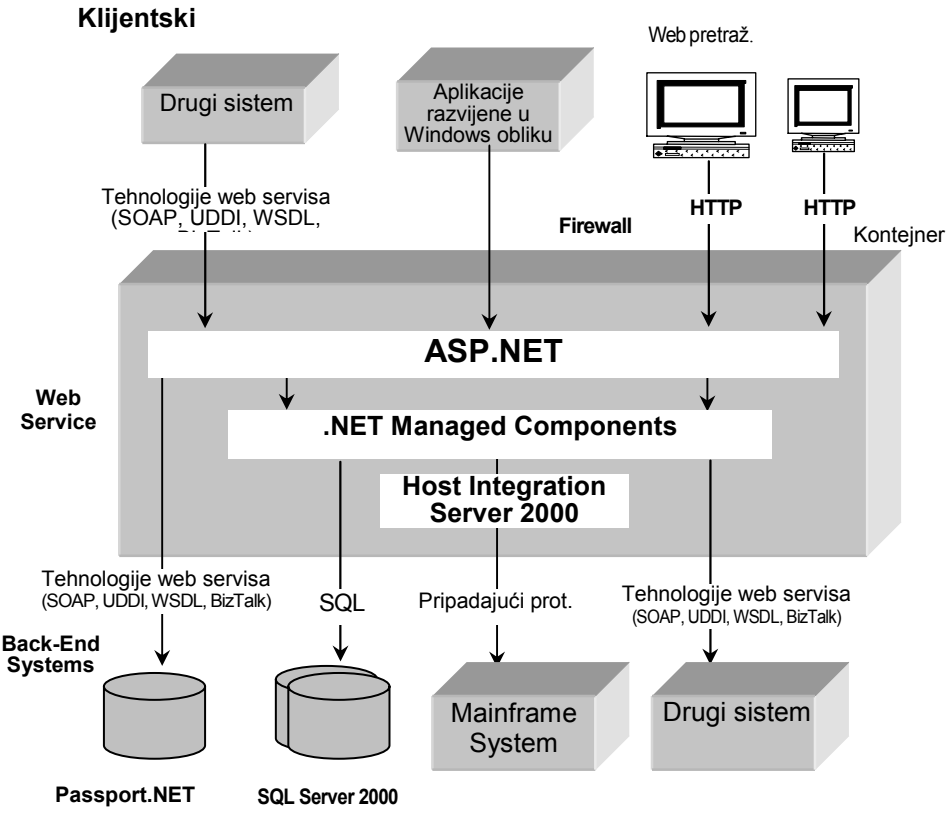
.NET (dot-net) predstavlja *Microsoftov* pogled na arhitekturu i tehnologiju rada informacionih sistema zasnovanih na korišćenju web servisa. Za razliku od J2EE pristupa, .NET nije samo skup standarda već, pre svega, skup komercijalnih tehnologija koje omogućavaju razvoj aplikacija orijentisanih na web servise. Možda je najpreciznije reći da. NET platforma polazi od nadgradnje zatečenih tehnologija kompanije *Microsoft*, odnosno, do same arhitekture. NET informacionog sistema došlo se postupkom reinženjeringa. U slučaju J2EE arhitektura je rezultat dogovora grupacije firmi okupljenih oko *Sun Microsystems*, a tek zatim se razvijaju odgovarajuće tehnologije za podršku razvoja sistema.

Dok je J2EE oslonjen na koncept jednog univerzalno prihvaćenog jezika – jave, raspoloživog na svakoj konkretnoj platformi, kao gradivnog elementa J2EE aplikacije, .NET dozvoljava da gradivni elementi. NET aplikacije budu sastavljeni od programa napisanih na različitim programskim jezicima, ali istog proizvođača.

Kao i u slučaju J2EE aplikacije, sama aplikacija je zatvorena u *kontejner* koji aplikaciji pruža važne servise, kao što su transakcioni i sigurnosni. Telo. NET aplikacije gradi se od. NET poslovnih komponenti koje sarađuju sa komponentama mehanizma za rad sa bazom podataka, ADO. NET. Za vezu sa softverskim nasleđem postoji *MS Host Integration Server 2000* sa svojim skupom servisa.

Veza sa drugim sistemima ostvaruje se uz pomoć web servis tehnologija, kao što su SOAP, UDDI i WSDL. Poslovni partneri se na. NET aplikaciju takođe povezuju korišćenjem ovih protokola, a *Microsoft* poseduje i sopstvenu tehnologiju za podršku zahtevnim poslovnim aplikacijama, *BizTalk*, koja, nažalost, nije kompatibilna sa ebXML-om. Raniji „debeli“ klijenti, web čitači i mobilni uređaji povezuju se na. NET aplikaciju posredstvom unapređene ASP tehnologije, ASP. NET, koja generiše potreban HTML ili WML kod. Obezbeđena je i podrška za tradicionalne Windows aplikacije.

Zahvaljujući pristupu reinženjeringa postojećih MS tehnologija, *Microsoft* nudi kao elemente. NET-a *SQL Server 2000* kao server baza podataka, *Exchange 2000 Server* kao kolaboracionu platformu i server unificiranih poruka, *Commerce Server 2000* za elektronsku trgovinu, *Application Center Server 2000* za rad klastera servera (serverskih „farmi“), *Host Integration Server 2000* za integraciju aplikacije sa softverskim nasleđem, *Internet Security and Accelaration (ISA) Server 2000* kao osnovni mehanizam zaštite (*firewall* – sigurnosni zid i *proxy*), *BizTalk Server 2000* za podršku integraciji složenih poslovnih procesa, bilo unutar organizacione jedinice ili sa drugim sistemima, posredstvom interneta.



Slika 9 – Razvojno okruženje. NET aplikacije

Microsoft je razvio *Visual Studio. NET* i snažno integrisano razvojno okruženje za. NET aplikacije. C# je programski jezik veoma sličan javi, koji je posebno pogodan za razvoj. NET aplikacija, koji je *Microsoft* razvio kao konkurenciju javi. Za podršku pri razvoju i radu sa već razvijenim web servisima *Microsoft* je razvio i *Hailstorm*, servis čiji su predmet rada drugi servisi.

Odnos J2EE – .NET

Obe platforme su korisne i vode ka istom cilju. Odluka o izboru između ove dve tehnologije (razvojna alata) zavisi od već postojeće baze inžinjerskog znanja u organizaciji, postojećih informacionih sistema i veza sa drugim sistemima. Odluka zavisi od odgovora na ova pitanja, a ne od minorne razlike u funkcionalnosti.

Karakteristike za obe platforme:

– za obe platforme potrebno je edukovanje razvojnog tima (java podučavanje za J2EE, objektnoorijentisano podučavanje za. NET),

– servise je moguće kreirati koristeći obe platforme,

– skalabilnost obe platforme teoretski je neograničena.

Uporedni prikaz funkcionalnosti obe platforme dat je u tabeli 4.

Tabela 4

Uporedni prikaz funkcionalnosti Microsoft. NET i java 2 platforme

Service or Feature	**Microsoft. NET Platform**	**Java 2 Platform EE**
Language	VB, C++, C#, Java, Jscript, Perl…	Java
OS Platform & Runtime	Windows - CLR	Any – JRE, JVM
Mobile Platform	.NET Compact Framework	Java 2 Micro Edition
GUI/In-proc Component	.NET class	JavaBeans
Server-side Component	.NET, sa COM+ services	EJB
Persistent Objects	ADO. NET DataSet	EJB Entity Beans
Web Page Generation	ASP. NET	JSP
„Code Behind“	ASP. NET	Java Servlet
Relational Data Access	ADO. NET	JDBC, SQL/J
Hierarchical Data Access	ADO. NET	
Queuing	System. Messaging	JMS
Asynchronous Invocation	COM+ Queued Components	Message Beans (EJB 2.0)
Eventing	COM+ Events	
Remoting	SOAP/HTTP/DCOM	RMI-over-IIOP
Naming	ADSI	JNDI
HTTP Engine	IIS	Apache
XML	System. XML	JAXP, JAXM, JAXB, JAXR…
Web Services	(.NET) XML Web Services	Sun ONE, IBM, BEA, Oracle Legacy Integration HIS (COMTI), BizTalk, MSMQ, WS JCA, JMS, WS, CORBA, JNI
Shared Context	Passport	The Liberty Alliance, JXTA
Security API	System. Security	JAAS

Istraživanjem u ovoj oblasti dolazi se do sledećih rezultata:

– SOAP je novi pristup u profesionalnom razvoju distribuiranih aplikacija. Rešava glavne probleme kod platformske zavisnosti i jezičke zavisnosti. Može da radi preko interneta i predstavlja otvoreni standard, koji se održava od strane W3C (*The World Wide Web Consortium*);

– nedostatak SOAP je što interoperabilnost sa CORBA nije uvek 100% izvodiva, odnosno postojeće implementacije ne zadovoljavaju uvek

traženu pouzdanost. S tim u vezi, OMG ima zadatak unapređivanja CORBA u smislu kompatibilnosti sa navedenim komunikacionim protokolom. To može biti jedan od uzroka nedovoljnog broja dostupnih web servisa na internetu;

– problem kod SOAP je i postojanje verovatnoće obilaženja firewalla, koji obezbeđuju sigurnost autorizovanja ili blokiranja poziva na određenim portovima. Time bi praćenje HTTP protoka informacija moglo postati još složenije;

– implementacija SOAP moguća je u skoro svakom programskom jeziku, na svim popularnim platformama. Postojeća distribuirana okruženja proširuju se za podršku SOAP-u;

– *Microsoft* je, izdavši. NET Visual Studio, zajedno sa konceptom. NET runtime-a, napravio ozbiljan korak ka usvajanju SOAP-a za standard u razvoju distribuiranih aplikacija. *Sun*[10] je najavio nešto slično u vidu svoje ONE (*Open Network Environment*) platofrme. IBM (*WebSphere*) i ostali industrijski giganti na ovom polju takođe najavljuju opsežnu SOAP podršku;

– SOAP poseduje potencijale da kreira transparentni web za servise i aplikacije kojima se može pristupiti na zahtev svakoga sa svake tačke, što će dovesti do eksplozivnog rasta novih servisa i prihoda od njih;

– SOAP ne pokušava da zameni CORBA ili bilo koji drugi distribuirani sistem. U poređenju sa CORBA SOAP je manje moćan, ali je impresivan u svojoj jednostavnosti i proširivosti, što je veoma korisno s aspekta potrebe da ga kompanije i programeri što više prihvate. Jednostavne stvari se mnogo brže usvajaju i prihvataju;

– CORBA i web servisi kreirani su iz različitih razloga, korišćenjem različitih tehnologija, iako su po prirodi komplementarni. CORBA omogućava razvijanje infrastrukture srednjeg sloja, sa snažnim i skalabilnim funkcionalnostima i servisima za kreiranje kritičnih sistema;

– platforme za razvoj web servisa mogu da obezbede neophodnu tehnologiju za iskorišćavanje postojećeg CORBA razvoja;

– kreiranje novog koda po CORBA arhitekturi dovedeno je u pitanje, s obzirom na to da je to arhitektura 90-tih i većina CORBA snadbevača ne nastavlja investiranje u ovu tehnologiju;

– velika prednost za SOAP i web servise jeste što imaju podršku velikih kompanija, kao što su IBM i Microsoft, kao i već postojeću veliku strukturu koja je izgrađena za WEB i HTTP;

– SOAP se i dalje razvija, pa će se pojaviti čitav niz novih protokola koji će pojednostaviti i olakšati primenu web servisa u svim sferama poslovanja, uključujući i vojne informacione sisteme.

[10] Sun Application Server, kasnije prerastao u Glassfish server (v1 do v3), već godinama sadrži kompletnu podršku za gotovo sve tipove standardnih web servisa. Takođe, Apache Jakarta projekat daje web servis podršku za Tomcat apl.server kroz biblioteke klasa, za PHP.

Zaključak

Ubrzani razvoj informacionih tehnologija i porast protoka informacija utiče na metodološke pristupe i koncepte u razvoju informacionih sistema. U poslednjem nizu godina došlo je do razvoja računarskih mreža i internet tehnologije, što je od velikog značaja i za pristup u razvoju vojnih informacionih sistema.

Za velike organizacione sisteme, kao što je vojna organizacija, poželjno je da u razvoju informacionih sistema postoje standardizovane metode i alati. Analizom poznatih metoda može se zaključiti da većina ima zajednički skup elemenata koji se koristi u modeliranju sistema. Zato je u radu predloženo korišćenje objedinjenog jezika modeliranja – UML, kao standarda za dokumentovanje, projektovanje i razvoj informacionih sistema.

Međunarodna organizacija za standardizaciju ISO je, kao osnovu razvoja distribuiranih informacionih sistema, definisala referentni model za povezivanje otvorenih sistema. On je zasnovan na konceptu hijerarhijske organizacije komunikacione arhitekture u sedam diskretnih slojeva. Zaključuje se da je potrebno obratiti pažnju na značaj TCP/IP u okviru telekomunikacione infrastrukture sa multiservisnim sposobnostima, koji predstavljaju jezgro projektovanih distribuiranih informacionih sistema. Zbog postojanja sopstvene (interne) telekomunikacione infrastrukture može se preporučiti da se i vojni informaciono-komunikacioni sistemi zasnivaju na navedenim standardima.

Pri određivanju strategije razvoja informacionih sistema u Vojsci, kao i za većinu kompleksnih organizacionih sistema koji sadrže heterogene hardverske i softverske platforme, od velikog je značaja mogućnost povezivanja postojećih rešenja.

Istraživanjem i analizom primene navedenih standardizovanih koncepata, tehnologija i razvojnih platformi, uključujući iskustva iz stranih oružanih snaga, zaključuje se da je koncept razvoja informacionih sistema u internet okruženju korišćenjem softverskih komponenti preporuka za strategijsko opredeljenje u prevazilaženju nekompatibilnih i nepovezanih postojećih parcijalnih softverskih rešenja. Osnovna ideja zasniva se na web servisima, koji se mogu kreirati od već gotovih aplikacija ili od početka, pri čemu se programski jezici, gotove komponente i platforme mogu koristiti po svom izboru.

Ranije su aplikacije za međusobnu udaljenu komunikaciju koristile pozive udaljenih procedura (Remote Procedure Calls – RPC) poput DCOM i CORBA ili su bile potpuno nepovezane. SOAP protokol omogućava komunikaciju između aplikacija na različitim operativnim sistemima, na različitih platformama, pisanih u različitim programskim jezicima. Inače, svetska organizacija W3C[11] objavila je kao preporuku SOAP verzije 1.1. i nastavlja s razvojem preporuke SOAP verzije 1.2.

[11] http://www.w3.org/TR/soap/

Aplikacije razmenjuju poruke dogovorenog formata. Poruke su formatirane kao XML dokumenti, pa je njihova obrada i provera jednostavna i može ih realizovati bilo koji program namenjen za rad sa XML dokumentima. SOAP klijent kreira XML dokument koji sadrži odgovarajući zahtev. Taj dokument formatiran je u skladu sa SOAP specifikacijom. Dokument dolazi do SOAP protokola koji obrađuje pristigle zahteve i na osnovu pristiglih zahteva pokreće odgovarajuću aplikaciju. Po završenoj obradi SOAP protokol vraća poruku odgovora SOAP klijentu. Dakle, zaključuje se da se transformacija podataka odnosi jednostavno na korišćenje XML formata.

Web servisi predstavljaju osnovne gradivne blokove budućih informacionih sistema, odnosno, u suštini, to mogu biti aplikacije raspoložive na vojnoj mreži. Pri opredeljenju za vrstu servisa zaključuje se da nije odlučujuće da li će biti implementiran standardni web servis ili specifični, posebno razvijen, na primer za vojnu primenu. Drugim rečima, to su resursi koji se adresiraju primenom URL-a i koji vraćaju informaciju korisniku. Analize opredeljuju komunikacioni protokol SOAP, tj. XML preko HTTP-a. Takođe, zaključuje se da je osnovni pokretač ovih promena XML, koji kroz svoju jednostavnost omogućuje praktičnu nezavisnost aplikacija i sistema, jer je razumljiv i za čoveka i za mašinu.

Literatura

[1] Sloman, M. and Kramer, J.: Distributed Systems and Computer Networks, Prentice – Hall, 1987.

[2] Siegel, J., CORBA 3, Object Management Group, COMPUTER, str. 114, May 1999.

[3] OMG, Common Object Request Broker Architecture: Core Spenfication, Nov. 2002, Version 3.0.

[4] Booch, G., Rumbaugh, J., Jacobson. J., Unified Modelling Language. Version J.3, Rational Software Corporation, 1999.

[5] Booch, G., Object–Oriented Analysis and Design with Application, 2. edition, Benjamin/Cummings, 1994.

[6] Java, Communications of the ACM vol. 41 No 6, June 1998.

[7] Department of Defense USA, Joint Technical Architecture, 2001.

[8] VJINFO2001 zbornik radova sa seminara o primeni informatike u Vojsci, Beograd, april 2001.

[9] Craig, L., Applying UML and Patterns, Prentice Hall, PTR, New Jersy.

DEVELOPMENT OF INTERNET-BASED INFORMATION SYSTEMS USING SOFTWARE COMPONENTS WITH THE EMPHASIS ON THE APPLICATION IN THE MILITARY ORGANIZATION

Summary:

The development of personal computers and Internet technology causes continuous changes in methodological approaches and concepts of development of information systems. Most existing information systems, due to their heterogeneity, have a problem of integration of subsystems. In order to overcome this problem, software vendors offer different solutions. In this work we explore different approaches and propose an optimal way, with a special emphasis on its application in the military organization.

By applying modern approaches in the development of information systems on the concept of distributed component systems, we come to the set of proposed solutions from different manufacturers. The solutions are related to the mechanisms which should ensure that components written in different languages cooperate with each other in heterogeneous systems that are in different nodes in the computer network.

This work describes the concept of component distributed information systems of Internet technology and their capabilities and offers a solution specifying the implementation environment in the military organization.

Access to the development of information systems

In the development of information systems, an important role is given to the choice of appropriate methods and tools. For large systems such as military organizations, standardized procedures and methodologies for the development of information systems are recommended. There are different methodological approaches in the development of information systems: ***a systematic integrated approach to development*** *(from design, implementation to implementation and maintenance) and* ***development of information systems as technical – technological structures*** *(standard computer and network service). The* ***combination*** *of these two approaches leads to the* ***concept*** *of* ***„open systems“*** *that allow different standards and IT services to operate on these systems. The UML system description of the process of software development has many different but interconnected models: use case model, analysis model, design model and implementation model. The previously mentioned methods of development of information systems can be systematized and described in a unique way called the Larmanova method, usually through several stages: 1.Specification request 2.Analysis, 3. Design, 4.Implementation and 5. Testing.*

Development of information systems in a military organization

The problem of development of information systems in the Internet environment by using software components is reduced to ***finding ways of determining a set of methodological procedures, concepts and approaches from the repositories of methods that support distribu-***

***ted information systems in Internet technology**. Therefore, it is necessary to design a methodological approach that will be adaptive in accordance with the specific characteristics of military organizations.*

While developing their information systems, most foreign armed forces use standardized procedures and methodologies applied in their environment. In cases where it is not specifically defined, there is a tendency to use modern methodological procedures in accordance with technical and technological development.

Component distributed information systems with an object-oriented approach in the Internet environment

*The usage of distributed objects can create client/server applications with a three-layer or multi-layer architecture which enables the implementation of synchronous or asynchronous solutions for the Internet and intranet. An example of a direct execution of transactions is an approach that relies on the CORBA model (Common Object Request Broker Architecture) and the Object Request Broker (ORB). However, the applications that use them are difficult and expensive to maintain. It is one of the reasons why the development of **component models** has become popular and experienced great development in recent years. Components with the **implemented mechanism for performing transactions** can be thus used in developing new applications. The component model allows easier changes and maintenance, and also shortens the time required for implementation, because there are various ready-made components that can be used. This approach to the implementation of transactions is called the declarative approach. Developers are free of care for the realization of mechanisms for controlling the execution of transactions. Important component models are implemented in COM + technologies (Component Object Model), EJB (Enterprise Java Beans), CORBA (Common Object Request Broker Architecture) and SOA (Service Oriented Architecture).*

Technologies for the implementation of component distributed information systems with an object-oriented approach in the Internet environment

The results of testing with different communication models for remote component calls resulted in the following technologies: COM + / DCOML, CORBA / IIOP, EJB / RMI / IIOP and SOAP. Web Services represent the basic building blocks of future information systems, and in fact are applications available on the network and suitable for executing what is necessary at that moment. The main communication protocol is SOAP, ie. XML over HTTP. They can be created from the already made applications or from the start, while programming languages, finished components and platforms can be used by the user's choice. SOAP represents the environment for message exchange based on XML in the network and it uses the HTTP protocol application used by Web servers.

From the aspect of this study, it can be concluded that there are two approaches through the J2EE (Java 2 Platform Enterprise Edition) environment and the Microsoft. NET set of technologies. SOAP is a new approach to professional development of distributed applications. It solves major problems with the platform and language dependences. It is able to work over the Internet and represents an open standard maintained by the W3C (the World Wide Web Consortium). The drawback of SOAP is interoperability which is not always 100% practicable with CORBA, ie. the existing implementation does not always meet the required reliability. This may be one of the causes of an insufficient number of available Web services on the Internet. The implementation of SOAP is possible in almost any programming language, on all popular platforms, so it has the potential to create a transparent web of services and applications that can be accessed on demand by everyone from every point, which will lead to the explosive growth of new services and therefore profits. SOAP does not try to replace CORBA or any other distributed system. Compared to CORBA, SOAP is less powerful, but it is impressive in its simplicity and scalability, which is very useful from the aspect of need for a wider acceptance by companies and by developers. SOAP is still developing, and a number of new protocols will appear, simplifying and facilitating the implementation of Web services in all business areas, including military information systems.

Conclusion

The possibility to connect the existing solutions is of great importance in determining the development strategy of information systems in the Army as well as in other complex organizational systems that contain heterogeneous hardware and software platforms. The research and analysis of the application of these standardized concepts, technologies and development platforms, including the experience of foreign armed forces lead to the conclusion that the concept of development of information systems in the Internet environment by using software components is a recommendation for strategic determination to overcome incompatible and unrelated existing partial software solutions. The basic idea is based on Web services which can be created from finished applications or from the start, while programming languages, finished components and platforms can be used by the user's choice.

The applications for mutual remote communication used to use Remote Procedure Calls – RPC such as DCOM and CORBA, or they used to be completely unrelated. The SOAP protocol allows communication between applications on different operating systems, on different platforms, written in different programming languages. The world organization W3C[12] has already published the SOAP version 1.1 as re-

[12] http://www.w3.org/TR/soap/

commendation. and it continues with the development of the recommendation of the SOAP version 1.2.

The applications exchange messages of the agreed format. Since the messages are formatted as XML documents, their processing and testing is simple and can be realized by any program designed to work with XML documents. The SOAP client creates an XML document that contains the corresponding request. This document is formatted in accordance with the SOAP specification. The document comes to the SOAP protocol which processes incoming requests and based on the received requests launches the appropriate application. Upon completion of processing, the SOAP protocol returns a response message to the SOAP client. It is, therefore, concluded that the data transformation relates to the use of the XML format.

Web Services represent the basic building blocks of future information systems, ie. in fact they may be available applications on the military network. When the decision on a type of service is concerned, it is not decisive whether the standard Web service will be implemented or a specific, particularly developed one, for example for military use. In other words, these are resources which are addressable using a URL and which return the information to the user. Analyses determine the SOAP communication protocol ie. XML over HTTP. It also concluded that the main initiator of these changes is XML which through its simplicity allows the practical independence of applications and systems because it is understandable both for a man and a machine.

Key words: *Internet technology, distributed systems, component systems, software components, military organization, heterogenic system, methodological approach.*

Datum prijema članka: 25. 11. 2009.

Datum dostavljanja ispravki rukopisa: 08. 02. 2010.

Datum konačnog prihvatanja članka za objavljivanje: 09. 02. 2010.