VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

DIPLOMSKI RAD

Metodološki postupak procesa razvoja WEB aplikacija

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Sadržaj

1. UVOD

Cilj ovog diplomskog rada jeste proučavanje poslovnih procesa, jezika za opis poslovnih procesa kao i metamodela za definisanje poslovnih procesa.

Ovaj rad predstavlja deo projekta razvoja Sistema za automatizaciju i praćenje razvojnih projekata u preduzeću „Soprex“ u Beogradu.

Dat je predlog modela za definiciju procesa i modela za izvršavanje procesa odnosno instance procesa koji se zasnivaju na opštim modelima elemenata procesa, njihove kompozicije i toka izvršavanja.

U okviru rada, tema proučavanja je i „workflow“ odnosno tok rada, kao i opšti model koji predstavlja predlog za implementaciju datih rešenja.

Sama implementacija predloženog modela je u toku pa neće biti predstavljana u okviru rada.

U drugom poglavlju je opisana metodologija razvoja informacionih sistema. Govori se o konvencionalnom i objektno orijentisanom pristupu, a poseban deo je posvećen pristupu razvoja softvera vođenim modelom (MDA). U okviru objektno orijentisanog pristupa su opisani Jedinstveni proces razvoja softvera i Jedinstveni jezik za modelovanje UML.

Treće poglavlje se bavi konceptom poslovnih procesa i metodologijama za modelovanje poslovnih proces.

Četvrto poglavlje je posvećeno modelovanju poslovnih procesa. U osnovi modelovanja poslovnih procesa, nalaze se jezici za modelovanje poslovnih procesa (Business Process Modeling Language). Oni predstavljaju apstraktne modele poslovnih procesa i odgovarajućih entiteta. Jezici za modelovanje poslovnih procesa definišu formalizme za izražavanje apstraktnih i izvršnih elemenata procesa koji se odnose na sve aspekte poslovnih procesa u jednom preduzeću, uključujući aktivnosti različitih kompleksnosti, transakcije, upravljanje podacima, konkurentnost, upravljanje izuzecima i neke složenije procesne semantike. Danas postoji veliki broj jezika za modelovanje poslovnih procesa koji se međusobno razlikuju u koncetraciji na različite aspekte poslovnih procesa, čime odgovaraju na različite potrebe u njihovom modelovanju.

U petom poglavlju je opisano upravljanje radnim tokom (Workflow managment), gde se definiše radni tok, proces, instanca procesa, kao i softverski servis za izvršavanje definicije procesa odnosno „engine“.

Šesto poglavlje se bavi Web servisima kao tehnologijom koja se uobičajeno koristi za interoperabilnost i integraciju aplikacija i informacionih sistema.

U sednom poglavlju se daje pregled modela koji su nastali kao rezultat izučavanja prethodnih tema i modela koji su predloženi od strane OMG grupacije.

2. METODOLOGIJA RAZVOJA

2.1. Model i modelovanje

Sistem možemo opisati kao skup objekata i njihovih međusobnih veza. Dejstvo okoline na sistem predstavlja ulaz sistema, a uticaj sistema na okolinu njegov izlaz. Model predstavlja apstraktnu predstavu realnog sistema To je subjektivni doživljaj objektivne stvarnosti.

Softver predstavlja model realnog sistema. Da bi softver bio dobar, mora predstavljati vernu sliku sistema, odnosno mora biti dobar model sistema. Loš softver je skup softver jer ne zadovoljava zahteve korisnika i zahteva velike vremenske, novčane i ljudske resurse za održavanje. Održavanje softvera uključuje otklanjanje grešaka, prilagođavanje promenama i zahtevima korisnika. Postoje tri tipa održavanja softvera: korektivno, adaptivno i perfektivno održavanje. Korektivno održavanje podrazumeva otklanjanje grešaka (debugging). Adaptivno održavanje se vrši u slučaju promena sistema i okruženja u kome softver funkcioniše. Da bi sveli na minimum ove promene, softver mora biti apstraktan. Perfektivno održavanje uključuje zadovoljavanje korisnikovih želja, odnosno implementiranje promena u zahtevima korisnika. U tom smislu, softver nikada ne sme biti direktno vezan za korisnikove zahteve, već se dobrim modelom mora vezati za sistem u kome korisnik deluje.

Problem razvoja softvera se može posmatrati sa dva aspekta: metodološkog i aspekta softverske arhitekture, koji su u međusobnoj korelaciji. Metodološki aspekt postavlja pitanje kako pristupiti razvoju informacionog sistema. Metodološki aspekt definiše specifični model kojim će se opisati sistem, i specifičnu metodologiju kao proces razvoja informacionog sistema. Aspekt arhitekture postavlja pitanje u kakvom softverskom okruženju se implementira sistem.

Modeli predstavljaju intelektualni alat za opisivanje sistema. Modelovanje[1] je proces kojim se sistem predstavlja pomoću datog modela.

Metodologija se definiše kao utvrđeni proces razvoja IS, gde se u različitim fazama, na različit način, primenjuju definisani modeli za opis sistema.

[1] Za modelovanje se kaže da je više umetnost i veština nego nauka.

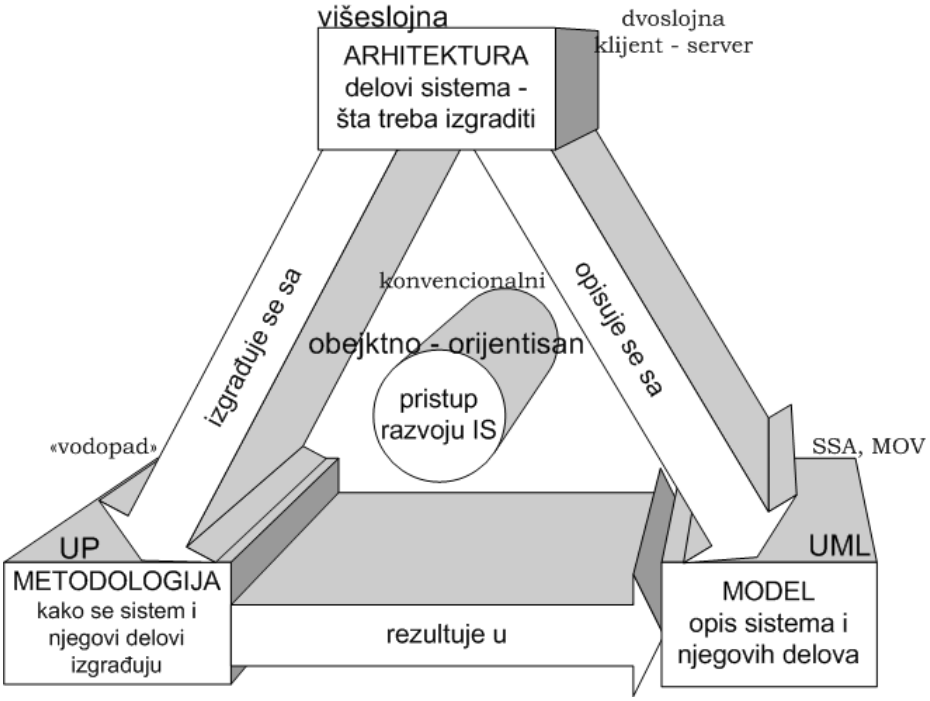
2.2. Pistupi u razvoju informacionih sistema

Jedan od najvećih problema u razvoju informacionih sistema je njihova složenost, koja se savladava korišćenjem dva osnovna metodološka principa:

- dekompozicija složenog sistema na manje celine čime određujemo njegovu arhitekturu i
- podela procesa razvoja na faze čime određujemo životni ciklus sistema.

Presek ova dva principa su modeli sistema koji nastaju u procesu razvoja, a kojima se opisuje arhitektura sistema. Na osnovu ovoga možemo reći da osnovni elementi koji definišu pristup razvoju informacionih sistema i koji najviše utiču na ostvarenje funkcionalnog i tehnološkog jedinstva su:

- ***Arhitektura*** koja definiše slojeve i delove IS kao i njegovo okruženje,
- ***Metodologija*** koja određuje *proces životnog ciklusa sistema*, odnosno određuje niz aktivnosti i načine njihovog izvođenja tokom «života» informacionog sistema i
- ***Modeli***, pogledi na sistem iz različitih uglova i sa različitih nivoa apstrakcije



Slika 1. Pristupi razvoju informacionih sistema

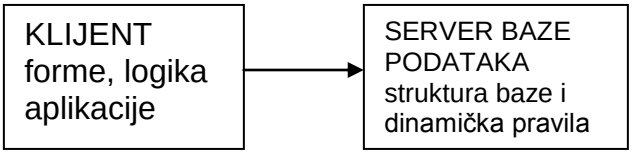
Vremenom došlo je do razvoja više pristupa u razvoju informacionih sistema:

- konvencionalni i
- objektno-orijentisani pristup
- modelom vođen razvoj softvera

Svaki od njih se razlikuje po primenjenom modelu, metodologiji i arhitekturi.

2.2.1. Konvencionalni pristup

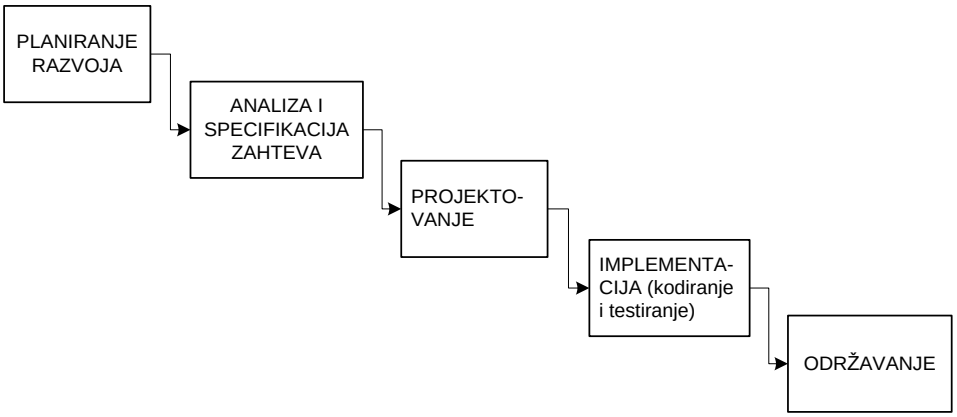
Kod konvencionalnog pristupa ***arhitektura*** je dvoslojna, klijent-server, gde se sva aplikativna logika vezuje za korisnički interfejs, odnosno nalazi se na klijentu, dok su podaci u bazi podataka na serveru.



Slika 2. Klijent – server arhitektura

Karakteristika ove arhitekture je slaba modularnost, odnosno nemogućnost ponovnog korišćenja logike.

Metodologija obuhvata frontalni razvoj i «vodopad» životni ciklus. Faze životnog ciklusa su sledeće:



Slika 3. „Vodopad“ životni ciklus

Modeli:

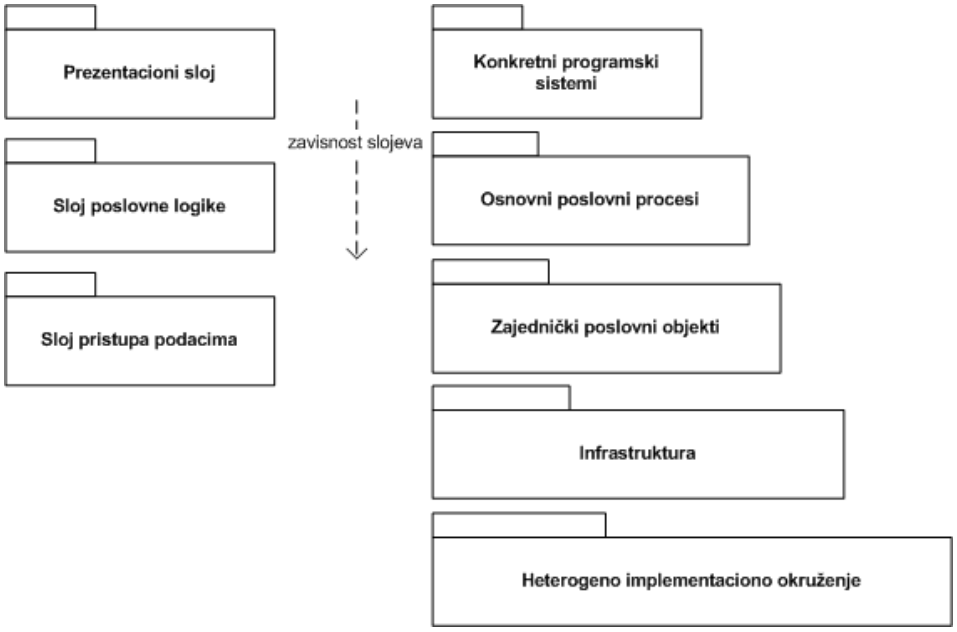
- Model ***strukturne sistemske analize*** kao skup fundamentalnih funkcija sistema, njihovih ulaza i izlaza
- Model ***objekti-veze*** koji određuje model podataka i
- Model ***složenog objekta*** kao modela aplikacije.

2.2.2. Objektno orijentisani pristup

Kod objektno orijentisanog pristupa *arhitektura* otklanja nedostatke arhitekture konvencionalnog pristupa u smisli da izdvaja aplikativnu logiku iz korisničkog interfejsa u jedan ili više slojeva

- Troslojna arhitektura podrazumeva raslojavanje sistema na sloj korisničkog interfejsa, sloj poslovne logike (srednji sloj) i sloj podataka. Ova arhitektura omogućava lakše menjanje i održavanje sistema jer promene u jednom sloju ne utiču na druge slojeve.

- Višeslojna arhitektura, koja nastaje daljim raslojavanjem sloja poslovne logike u distribuirane softverske komopnente, omogućava ponovno korišćenje tih komponenti koje se mogu menjati bez uticaja na kod koji ih koristi.



Slika 4. Troslojna i višeslojna logička arhitektura

Metodologija koja se ovde koristi je Jedinstveni proces razvoja softvera.
Modeli se opisuju Jedinstvenim jezikom za modelovanje (UML).

2.2.2.1. Jedinstveni proces razvoja softvera

Jedinstveni proces razvoja softvera (**U**nified Software Development **P**rocess UP) je softverski razvojni proces koji podrazumeva skup aktivnosti potrebnih da se korisnički zahtevi transformišu u softverski sistem. On predstavlja okvir generičkog procesa koji može biti specijalizovan za veliku klasu softverskih sistema. Jedinstveni proces je zasnovan na komponentama, tj. softverski sistem je izgrađen od softverskih komponenti koje su između sebe povezane odgovarajućim interfejsom.

Jedinstveni proces karakterišu sledeće osobine:

- sa njim se upravlja na osnovu slučajeva korišćenja (use-case driven)
- usmeravan je arhitekturom (architecture-centric)
- sastoji se od iteracija

Jedinstveni proces se nametnuo kao jedan od najboljih pri realizaciji objektno-orijenatisanih sistema. Njegova glavna karakteristika je *iterativni* razvoj u smislu da je organizovan u serije kratkih i vremenski ograničenih mini-projekata (iteracija), tako da je rezultat svake iteracije testirani, integrisani i izvršni deo sistema.

Iterativan ciklus je zasnovan na stalnom uvećavanju i usavršavanju sistema kroz svaku iteraciju, te se zato ovaj pristup zove *iterativno-inkrementalan* razvoj softvera.

Jedinstveni proces razvoja softvera definiše faze, kao skup međusobno povezanih aktivnosti na rešavanju nekog problema odnosno ispunjenju nekog zadatka u razvoju sistema. Sa aspekta vremenske dimenzije rad je organizovan kroz četiri glavne faze:

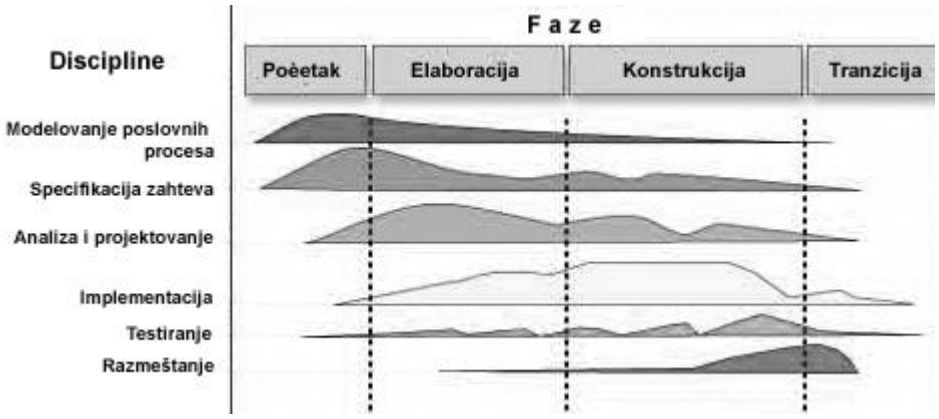
1. Početak (*Inception*) – definiše se opseg i vizija projekta, kao i studija koja je nalik na studiju izvodljivosti projekta. Navode se osnovni slučajevi korišćenja
2. Razvoj (*Elaboration*) – poboljšanje vizije, razvoj plana projekta, razrada slučajeva korišćenja, nacrt arhitekture sistema kao i prve interacije njene implementacije
3. Građenje (*Construction*) – iterativna implementacija preostalih elemenata na osnovu koje se dobija Beta verzija, priprema za prelaz
4. Prelaz (*Transition*) – Beta verzija se prosleđuje do korisnika radi testiranja, gde se nakon uočenih problema pravi konačna verzija. U ovoj fazi se takođe vrši i obuka korisnika.

Svaka iteracija Jedinstvenog procesa se sastoji od sledećih radnih tokova (disciplina):

- Modelovanje poslovnog procesa – opis strukture i dinamika organizacije
- Specifikacija zahteva – definisanje i opis slučajeva korišćenja
- Analiza i projektovanje – dijagrami klasa, sekvenci, kolaboracije i dijagrami promene stanja
- Implementacija – kodiranje, testiranje i integracija komponenti

- Testiranje – verifikuju se rezultati implementacije preko opisa test slučajeva, procedure i praćenja grešaka
- Razmeštanje - Konfiguracija sistema koji se isporučuje.

Važno je napomenuti da svaka iteracija u svakoj fazi sadrži gotovo sve discipline sa tim što se u zavisnosti i od faze akcenat menja tokom vremena što možemo i videti na sledećem dijagramu.



Slika 5. Faze Jedinstvenog procesa razvoja softvera

2.2.3. Modelom vođen razvoj softvera

Modelom vođen razvoj softvera (MDE – Model Driven Engineering) je pristup u kojem su centralni koncepti model i njihova transformacija. Modeli su opisani upotrebom dobro definisanih jezika za opis (metamodela), i nastaju kao rezultat svake faze razvoja, pri čemu predstavljaju realan sistem posmatran sa različitih aspekata i nivoa apstrakcije. Skup modela koji predstavljaju sistem u fazi analize transformišu se u skup modela u fazi projektovanja itd. Stoga se sam proces razvoja softvera može posmatrati kao niz transformacija modela sve do izvršne verzije IS na ciljnoj tehnološkoj platformi, a da se informacioni sistem sastoji iz skupa povezanih konkretizovanih modela u nekom ciljnom okruženju. Svaki od ovih modela je potencijalno opisan različitim jezicima za opis. Savremeni informacioni sistemi se implementiraju korišćenjem više metamodela (relacioni model, objektni model, XML Schema...). Takođe modeli koji nastaju u ostalim fazama razvoja softvera, posebno tokom analize i projektovanja, se realizuju pomoću različitih metamodela (SSA, UML, PMOV i drugi jezici za opis funkcija i strukture sistema).

Jedan od osnovnih problema koji se javlja pri razvoju IS, i kasnije u njegovom održavanju, jeste savladavanje heterogenosti komponenti IS koja je uzrokovana različitim metamodelima kojima su te komponente realizovane i opisane. Međutim indukcijom se može zaključiti da se na najvišem nivou apstrakcije svi metamodeli

zasnivaju na nekoliko fundamentalnih koncepata. Kao što je metamodel jezik u kojem se projektuje konkretan IS , tako je i meta-metamodel jezik koji opisuje različite jezike za opis. Takav meta-metamodel bi obuhvatio, odnosno opisao sve koncepte postojećih jezika za opis čime bi se omogućila transparentna komunikacija, odnosno transformacija između tih jezika.

Najčešće korišćen okvir za MDE je Modelom vođena arhitektura (MDA – Model Driven Architecture) koja podržava specifikaciju na taj način da se specifikacija funkcionalnosti sistema odvaja od specifikacije implementacije te funkcionalnosti na odgovarajućoj tehnološkoj platformi.

2.2.3.1. Modelom vođena arhitektura

Modelom vođena arhitektura (Model Driven Architecture – MDA) predstavlja pokušaj da se fokus razvoja softvera prebaci sa pisanja koda na kreiranje modela. Na odgovarajućim nivoima apstrakcije, ovi modeli mogu da pomognu naručiocima projekta i timovima za razvoj da prevedu svoje razumevanje domena problema u mnogo pouzdaniji kod mnogo brže.

Da bi lakše predstavili principe modelom vođene arhitekture u nastavku teksta daćemo jedno poređenje.

Zamislimo da imamo veliku mapu jedne zemlje u razmeri 1 prema 1. Na ovakvoj mapi imali bi sve detalje koji su nam potrebni ali bi potrošili značajno vreme da na njoj nađemo ono što tražimo. [5]

Dok su kartografi pravili različite mape u zavisnosti od detalja koje žele da prikažu, programeri su se tradicionalno fokusirali na jedan model, kod programa, što možemo uporediti sa mapom sa početka primera.

Modeli se često koriste da bi se prikazali kompleksni sistemi. Modeli se mogu posmatrati na različitim nivoima apstrakcije, i komplementarni pogledi na modele se mogu kombinovati da bi se dobio mnogo precizniji pogled na sistem od jednog jedinstvenog modela. Dugo su se modeli koristili samo u fazi analize da bi se bolje shvatio problem ali čim bi se pristupilo ozbiljnijem poslu na njih bi se zaboravilo, te oni više ne bi bili ažurirani i ne bi predstavljali stvarno stanje projekta.

2001. godine, Object Management Group (OMG) je započela inicijativu pod nazivom Model Driven Architecture (MDA) sa vrlo ambicioznim ciljem da se fokus prilikom izrade projekata prebaci sa pisanja koda na modelovanje. Kao što su jezici višeg nivoa zamenili asembler, biblioteke koda i okviri (framework) su omogućili ponovo korišćenje izolovanih delova koda, i na kraju, uzori su zamenili kod koji je specifičan za jedan projekat.

Modelovanje u MVA (Modelom Vodjenoj Arhitekturi) započinje generisanjem skup preduslova na osnovu kojih se izrađuje model sistema koji zadovoljava ovaj skup. Ovaj početni model opisuje zahteve ali se ne vezuje za specifične platforme. Tek kasnije, u

narednim fazama projekta, na osnovu skupa pravila, vrši se transformacija ovog modela nezavisnog od platforme (Platform Independent Model - PIM), u model specifičan za određenu platformu (Platform Specific Model - PSM).

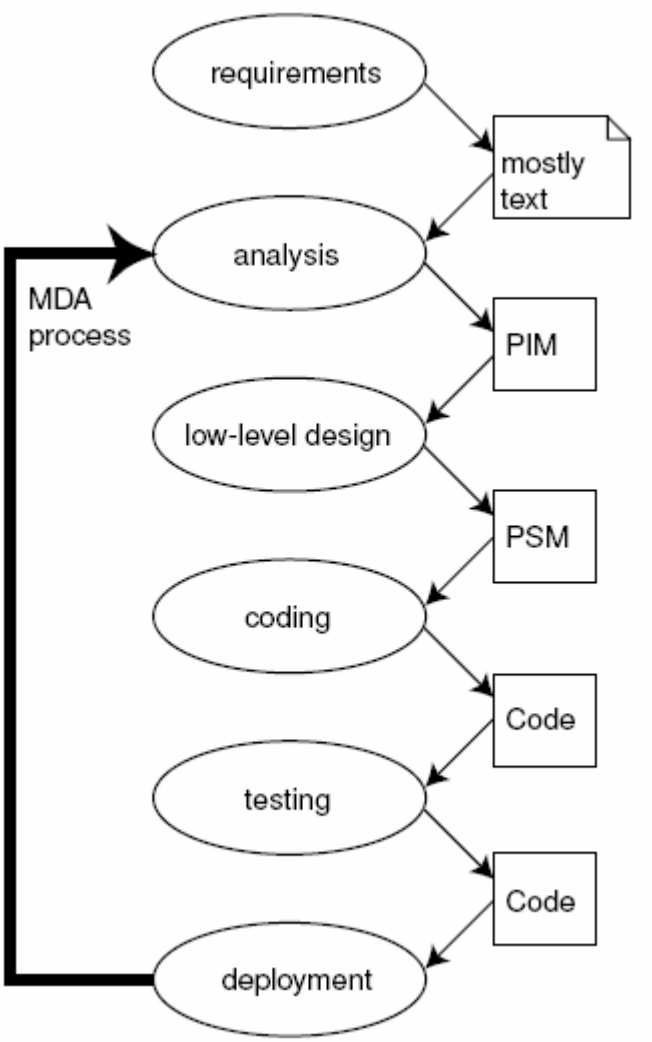
MVA specifikacija namerno koristi termin platforma, da bi se naglasilo da se ovo ne odnosi samo na operativni sistem već i na programski jezik.

Konceptualno, ove transformacije smanjuju nivo abstrakcije čineći neke aspekte modela nezavisnog od platforme konkretnim. Na ovaj način se ipak ne može odmah doći do konkretnog koda u nekom programskom jeziku već se mora izvršiti još jedan skup transformacija nad PSM modelom da bi se on mogao mapirati u kod.

Koristeći MVA, okruženje za modelovanje kreira programski kod iz modela, te se kod ne piše nikada direktno (ili u retkim slučajevima kada je to potrebno zbog optimizacije). Kako se menja sistem, menjaju se i odgovarajući delovi modela i tek onda se dati model ponovo sinhronizuje sa kodom. Na ovaj način, modeli i modelovanje postaju središte napora projektanata. MVA pravi jasnu razliku između poslovnog modela (Computation-Independent Model - CIM), poslovnog modela u specifičnom kontekstu razvojne tehnologije (PIM) i poslovnog modela koji je usko povezan za poslovni proces koji se projektuje kao i kod specifičan za platformu (PSM).

Životni ciklus MVA se sastoji sledećih šest koraka:

- snimanja zahteva u CIM
- kreiranje PIM-a
- transformaciju PIM-a u jedan ili više PSM-a, dodajući pri tom pravila koja su specifična za platformu
- prevođenje PSM-a u kod
- testiranje
- prenošenje sistema u konačno okruženje (instaliranje).

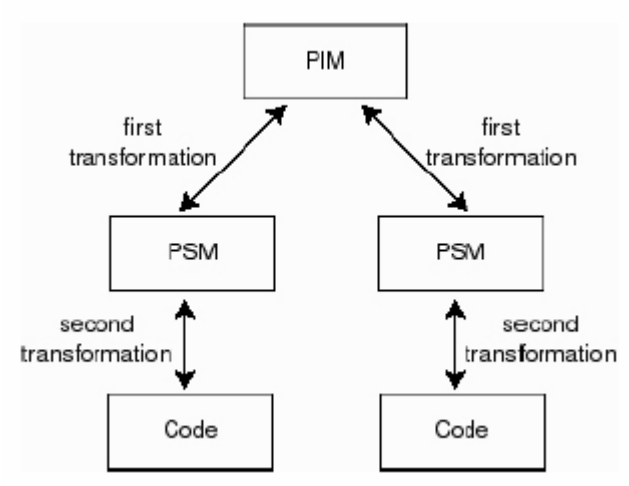


Slika 6. MDA životni ciklus razvoja softvera

Treba još napomenuti da se ovde ne završava proces izrade projekta. Naime, onog trenutka kada se softverski sistem pusti u eksploataciju mogu se javiti različiti problemi. Sistem koji je modelovan je mogao da se promeni ili je jednostavno moguće da se nisu dovoljno istestirali svi slučajevi koji mogu da nastupe u eksploataciji. Kada se prepoznaju ovi problemi pristupa se izmeni PIM-a u cilju usklađivanja sistema sa novo prepoznatim stanjem. Tek kada se završe izmene na modelima ovog nivoa apstrakcije pristupa se ostalim fazama do trenutka kada se ponovo vrši instalacija na ciljni sistem.

Ključni element MVA je transformacija koja se vrši isključivo uz pomoć softverskih alata. Mnogi CASE alati za objektno orjentisani razvoj, koji su zastupljeni na tržištu podržavaju automatsku transformaciju iz PSM u programski kod. Ono što trenutno nedostaje su alati za transformaciju PIM-a u PSM.

Na sledećoj slici možemo videti odnos PIM, PSM i programskog koda:



Slika 7. Odnos između PIM, PSM i programskog koda

MDA *framework* sastoji se od sledećih gradivnih elemenata:

- Modeli izgrađeni korišćenjem standardnog, dobro definisanog jezika, koji su konzistentni, precizni i sadrže dovoljno informacija o posmatranom sistemu
- Jedan ili više standardnih, dobro definisanih jezika za modelovanje
- Definicije kako se PIM modeli transformišu u PSM modele
- Jezik u kojem se pišu definicije transformacije. Pošto alati za transformaciju interpretiraju definicije transformacije, ovaj jezik mora biti formalan
- Alati za izvršavanje definicija transformacije PIM u PSM
- Alati za transformisanje PSM u programski kod

Na kraju možemo navesti osnovne prednosti MDA pristupa:

Portabilnost. MDA pruža portabilnost i razvoj nezavisan od platforme zahvaljujući postojanju platformski nezavisnih modela (PIM), čijom se transformacijom generišu različiti PSM modeli u zavisnosti od izabranog ciljnog okruženja.

Produktivnost. Projektanti, programeri i administratori mogu koristiti jezike za modelovanje sa kojima su familijarni, a samom primenom MDA pristupa i dalje je zadržana dobra komunikacija i integracija različitih timova. Velika produktivnost može se postići i korišćenjem potpuno automatizovanog generisanja programskog koda na osnovu PSM. Produktivnost se drastično povećava sa korišćenjem nešto složenijih modula za transformaciju PIM u PSM.

Interoperabilnost. Primenom MDA, identične specificirane poslovne funkcije su implementirane u raznim implementacionim tehnologijama a što je garantovano primenom rigoroznih metoda ovog pristupa. Interoperabilnost nije obezbeđena samo postojanjem modula za transformaciju PIM u PSM već i mogućnošću izgradnje modula za transformaciju (*bridges*) iz jednog u drugi PSM.

Lakše održavanje i dokumentovanje. MDA pristup podrazumeva da mnogo informacija o sistemu mora biti uključeno u PIM. Takođe, podrazumeva se da izgradnja PIM zahteva mnogo manje napora nego pisanje programskog koda.

3. POSLOVNI PROCESI

3.1. Šta je poslovni proces?

Poslovni proces definiše kako organizacija postiže svoju svrhu. Prema OMG-u (Object Management Group) proces predstavlja „tok rada i informacija u poslovanju“. On se sastoji iz atomskih koraka, koji su međusobno povezani poslovnim pravilima.

Korak poslovnog procesa je osnovni građevinski blok koji predstavlja jedinicu posla i ne može se razložiti na manje korake. Takođe se i naziva aktivnošću ili transakcijom, a kako je atomski on jedinstveno identifikuje sve događaje u procesu.

Ljudski akteri su sastavni deo svakog procesa koji nije u potpunosti automatizovan, tako da neautomatizovani koraci imaju korisnički interfejs kojim je proces nadgledan i kontrolisan od strane aktera. Korisnički interfejs omogućava akteru da unosi i modifikuje atribute, kao i da pokreće, suspenduje, uspostavlja, zaustavlja i završava procese.

Takođe, svaki proces ima pristup i mogućnost izmene stanja svakog poslovnog entiteta koji je vezan za taj proces.

Davenport i Short poslovni proces definišu kao skup logički povezanih zadataka koji se izvršavaju da bi se dostigao unapred definisan poslovni rezultat. Proces je struktuiran skup aktivnosti dizajniranih sa ciljem da proizvode određeni rezultat za nekog kupca ili tržište.

Poslovni proces predstavlja posao, podeljen u više koraka ili aktivnosti, koji su neophodni da bi se obavila poslovna transakcija. Da bi se izvršile aktivnosti u okviru poslovnog procesa, može biti neophodna akcija od strane aplikacije ili čoveka. Tipično za poslovne procese je da su po svojoj prirodi dugotrajni, kao i da uključuju više strana i/ili aplikacija u okviru ili van organizacije.

Poslovni proces može biti veoma složen po svojoj strukturi. Može imati više učesnika, kao što su ljudi, organizacije i sistemi, koji izvršavaju više zadataka. Da bi ostvarili postavljeni zadatak, učesnici moraju koordinisano izvršavati definisane zadatke, najčešće grupisane u podprocese. U nekim situacijama, podprocesi se izvršavaju paralelno, u drugim su sekvencijalni. Neki procesi zahtevaju ponovno izvršavanje podprocesa. Većina procesa sadrži tačke odluke, koje dovode do grananja toka u zavisnosti od postavljenih, zadovoljenih ili ne zadovoljenih uslova. U okviru nekih procesa, učesnici moraju proslediti određene informacije. Prenos informacija može imati ulogu okidača, odnosno otpočinjanja novog zadatka. Neki procesi su ad-hoc procesi, što znači da njihovi podprocesi nemaju definisane okidače. Učesnici ne moraju da završe sve definisane zadatke pre nego što oni, ili neki drugi učesnik, započnu izvršavanje drugog zavisnog podprocesa.

3.2. Metodologije modelovanja poslovnih procesa

Metodologija se može definisati kao:

- analiza principa metoda, pravila i postulata koje koriste discipline,
- razvoj metoda, koje se primenjuju preko disciplina,
- određena procedura, ili skup procedura.

Metodologija uključuje sledeće koncepte, odnosno određene discipline ili polja istraživanja:

- kolekcija teorija, koncepata, ili ideja;
- komparativna studija različitih pristupa;
- kritika pojedinačnih metoda.

Metodologija predstavlja više od prostog skupa metoda, jer se odnosi više na obrazloženja i filozofske pretpostavke, nego na određenu studiju.

Model predstavlja uzor, plan, predstavljanje, ili opis dizajniran da prikaže strukturu objekta, sistema, ili koncepta.

Proces može izgledati drugačije kada se opisuje od strane različitih učesnika. Potrebno je da metodologija modelovanja poslovnih procesa bude sposobna da reprezentuje različite aspekte opisa procesa. Dobra metodologija treba da omogući predstavljanje procesa na način jednostavan za prebacivanje u oblik prepoznatljiv učesniku.

Modeliranje poslovnih procesa uobičajeno ima širi opseg u odnosu na modeliranje pojedinačnog softverskog sistema. Svrha modelovanja je da analitičar sistema može jasno da predstavi opseg sistema koji se modeluje i koji će biti na odabrani način implementiran.

Stefan Haberl je razvio skup od sedam kriterijuma za evaluaciju metodologija modelovanja poslovnih procesa [Haberl]:

1. sposobnost da modeluje sve složene situacije vezane za poslovni proces:
   - sekvence
   - grananja
   - petlje
   - konkurentne elemente - paralelne grane i sinhronizaciju
   - kontrolu vremena izvršavanja
   - krajnji rok za izvršavanje
   - kontrolu grešaka
   - agregaciju
2. mogućnost za razlikovanje uloga, kao i mogućnost da se ulogama dodeljuju različiti zadaci
3. postojanje nedvosmislene grafičke prezentacije jezika

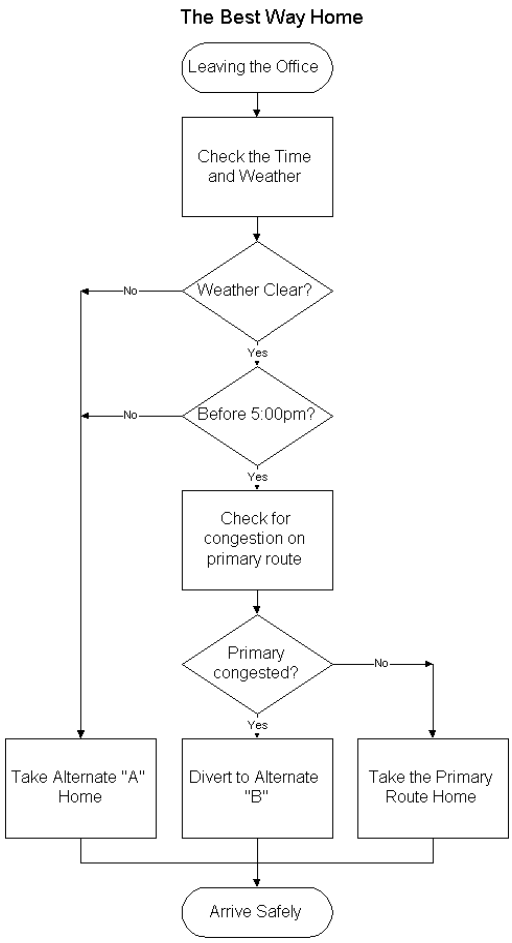
4. podržavanje transakcionog modela, koji omogućava opis situacije kada se proces mora vratiti u prethodno dobro stanje
5. definisanje kako će instance procesa biti pokrenute i identifikovane tokom izvršavanja
6. mogućnost za specifikaciju karakteristika poslovnog procesa, koji su važni za eksterne korisnike (što uključuje kvalitet usluga i cene)
7. jezik ne treba da sadrži detalje vezane za komunikacioni protokol.

Već dugi niz godina ljudi modeluju procese. Tehnike koju su koristili opisuju kako organizacije izvršavaju svoje poslove. Modeli su korišćeni u različite svrhe: za obuku novih zaposlenih, kao pomoć u procesu reinženjeringa, za kreiranje simulacionih modela za testiranje procesa, za razvoj sistema koji će automatizovati modelovani proces, kao vodič programerima za razvoj informacionih sistema, ili za direktno izvršavanje pomoću procesne mašine [Kerschberg].

Tokom godina razvijene su brojne metodologije modelovanja poslovnih procesa. Neke od njih, korišćene poslednjih trideset godina, su:

Dijagrami toka (Flow Charts)

Dijagrame toka koriste programeri da bi opisali logiku, ili putanju izvršavanja programa. Simboli i semantika dijagrama toka je ograničena na određeni broj kontrolnih struktura namenjenih programerima. Dijagrami toka su razvijeni da bi opisali put izvršavanja u okviru jednog procesa. Nemaju snagu da na pravi način modeluju grupu procesa koji među sobom sarađuju.



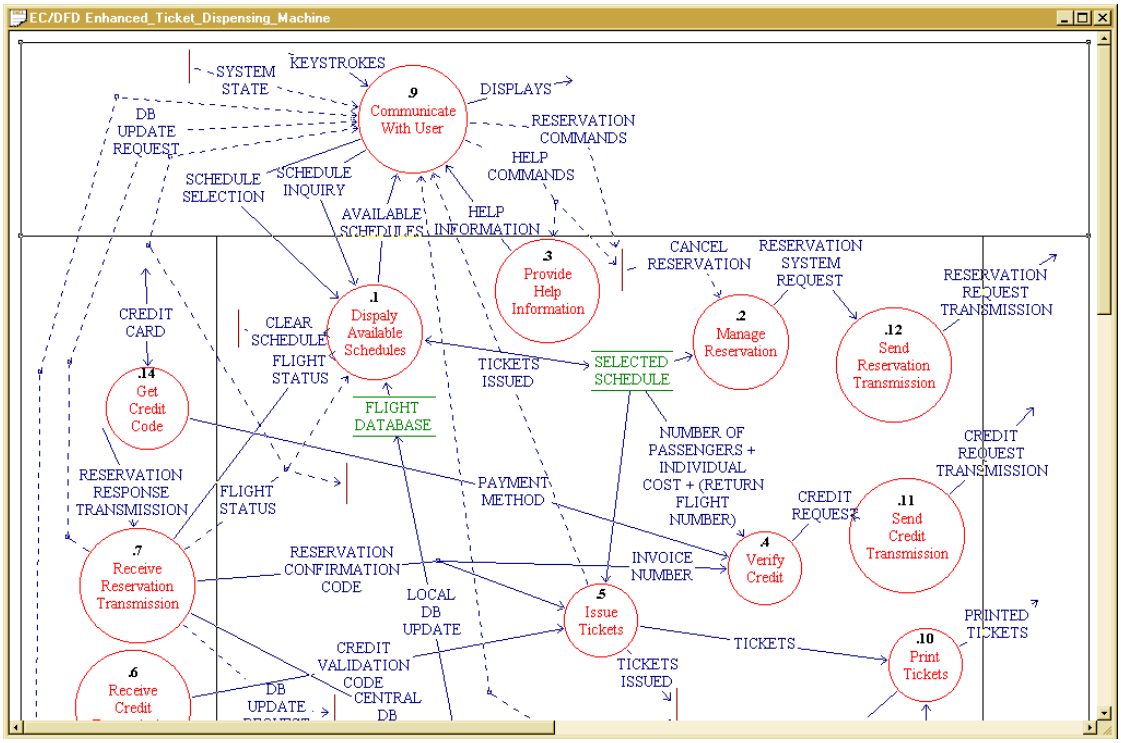
Slika 8. ANSI Flow Chart

Dijagrami toka podataka (Data Flow Diagrams)

Ovaj tip dijagrama se bavi podacima u okviru informacionog sistema. Počeo je da se koristi 1978. godine, posle izlaska knjige autora Tom DeMarco "Strukturna sistem analiza i sistemske specifikacije". Dijagrami toka podataka prikazuju niz koraka obrade podataka. Svaki korak dokumentuje jednu akciju, koja transformiše ili distribuira podatke. Nemaju mogućnost da opišu sekvence procesa ili mehanizme za kontrolu procesa.

Dijagrami toka kontrole (Control Flow Diagrams)

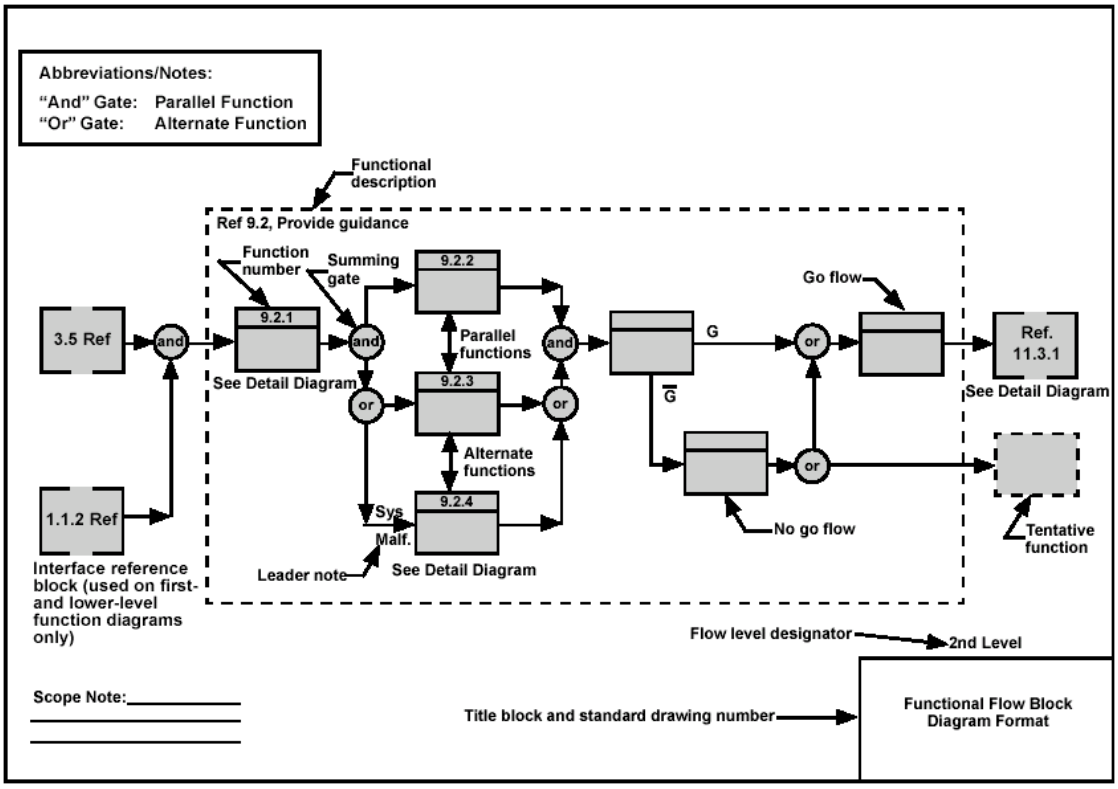
Ova vrsta dijagrama je nastala kao proširenje dijagrama toka podataka. Koristi se da opiše aplikacije zasnovane na događajima, više nego za aplikacije zasnovane na podacima.



Slika 9. Kombinovani Data/Control Flow Diagrami

Funkcionalni blok dijagrami toka (Functional Flow Block Diagrams)

Ovaj tip dijagrama se uobičajeno koristi u klasičnim inženjerskim sistemima da prikaže redosled izvršavanja sistemskih funkcija. Dijagrami prikazuju pravilan redosled u izvršavanju funkcija, kao i konkurentnost prilikom izvršavanja. Konkurentne funkcije mogu se izvršavati paralelno (AND) ili alternativno (OR).



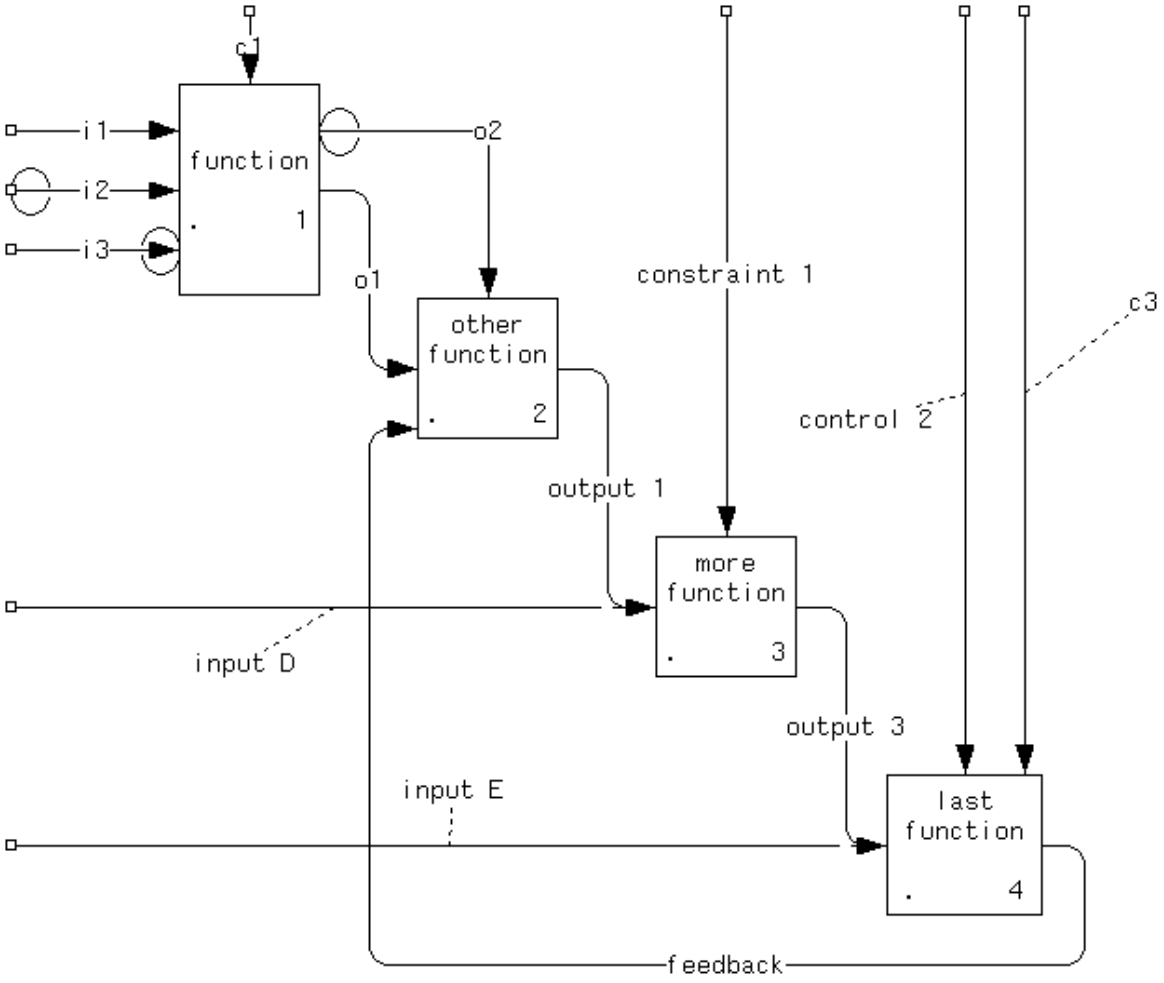
Slika 10. Funkcionalni blok dijagram toka primer [Kerschberg]

Gantt/PERT dijagrami

I Gantt dijagrami (napravljeni 1917. godine kao metod koji omogućava vremensko planiranje poslova vezanih za konstrukciju brodova) i PERT dijagrami (napravljeni 1950. godine za američku mornaricu sa ciljem da omoguće prikaz složenih sekvenci u izvršavanju poslova) se koriste za upravljanje projektima. Veoma retko se koriste da modeliranje procesa, jer ne poseduju mogućnost opisa bilo koje druge informacije o procesu, izuzev redosleda izvršavanja zadataka. Ne podržavaju modeliranje ulaznih i izlaznih informacija, kao ni mehanizme za kontrolu procesa.

IDEF

IDEF je verovatno najkorišćenija tehnika za tradicionalno modeliranje poslovnih procesa. Postoji nekoliko tipova IDEF modela. Poznat je IDEF0 model aktivnosti, koji omogućava modeliranje zadataka koje izvršava jedna organizacija, i koji uključuju ulaze, izlaze, i kontrolu svakog zadatka. Zadaci, ili aktivnosti, mogu biti prikazani kao zadaci na visokom nivou, i mogu se razložiti u podaktivnosti.



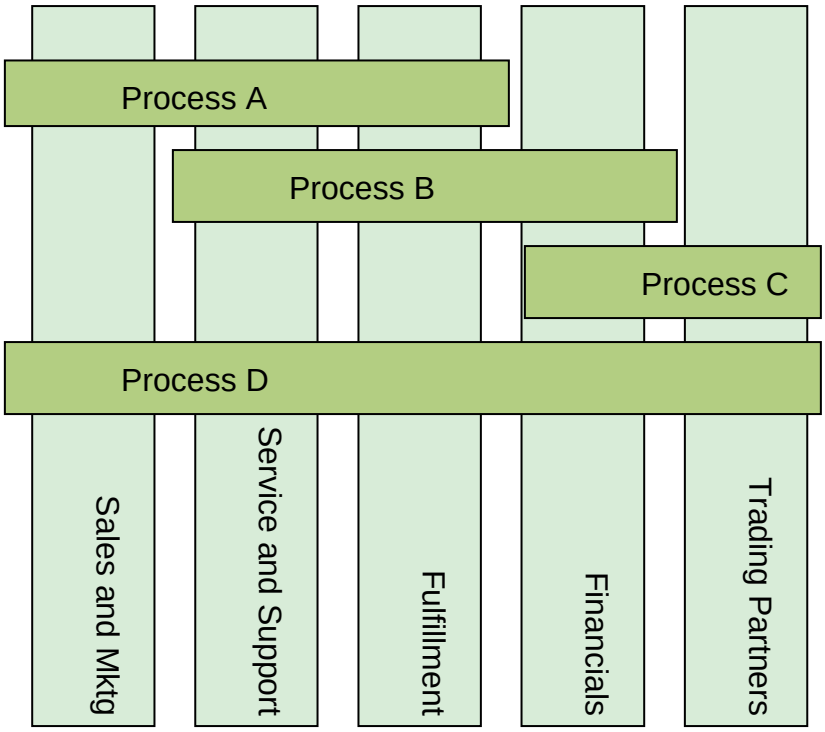
Slika 11. Primer IDEF0 diagrama

Poslednjih nekoliko godina razvijene su, ili su u razvoju, niz metoda, koje treba da omoguće modeliranje savremenih poslovnih procesa. Neke od tih metodologija su ebPML, BPMN, BPML, XLANG, WSFL, BPEL4WS, EDOC, XPDL, UML 2.0. U radu će biti detaljno opisane BPMN i BPEL metode za modeliranje poslovnih procesa.

3.3. Business Process Modeling

Pojam Business Process Modeling (BPM), ili kako se jos naziva Business Process Management, odnosi se na dizajniranje, upravljanje i izvršavanje poslovnog procesa, a njegova snaga se nalazi u objedinjavanju i proširenju postojećih procesno orijentisanih tehnika i tehnologija.

Za poslovne analitičare, BPM znači sagledavanje organizacije kao skupa procesa koji se mogu definisati, kojim se može upravljati, i koji se mogu optimizovati. Umesto tradicionalne orijentacije, prema kojoj se poslovi dele po organizacionim celinama, BPM se orijentiše prema poslovima, bez obzira u kojoj se organizacionoj jedinici izvršavaju. Za tehničko osoblje, BPM predstavlja grupu tehnologija fokusiranih na definisanje, izvršavanje i nadgledanje procesne logike. Bez obzira na različite perspektive, obe grupe, i poslovni analitičari, i tehničko osoblje, imaju cilj da unaprede poslovne procese.



Slika 12. Umesto tradicionalnih funkcija usmerenost na procese

BPM tehnologije mogu da ubrzaju procese, mogu da učine procese pouzdanijim, stalnijim, i otpornijim na greške. Poboljšanje poslovnih procesa ujedno znači i poboljšanje samog posla.

Iako različite organizacije koriste drugačije procese, još uvek postoji velika sličnost u različitim poslovima. Na primer, mnogi procesi zavise od grupe ljudi koja radi zajedno u cilju da obavi određene zadatke, često po unapred definisanom redu. Takođe je uobičajeno da procesi zavise od različitih vrsta softvera koji rade zajedno, često na

prilično struktuirani način. Ovaj softver može da donosi odluke, od prostih kao što je biranje jedne od opcija, do složenih odluka baziranih na pravilima. Praćenje procesa koji se izvršavaju takođe je korisno, kao što je postojanje načina da se ovi procesi opišu. Ove sličnosti između različitih organizacija je ono što čini BPM mogućim. Izdvajajući zajedničke elemente poslovnih procesa, implementiranjem automatizovane podrške za ove sličnosti, BPM tehnologije nude opšte koristan pristup za napredovanje procesa.

BPM treba koristiti samo za aplikacije koje su procesno orijentisane, odnosno koje su [Havey]:

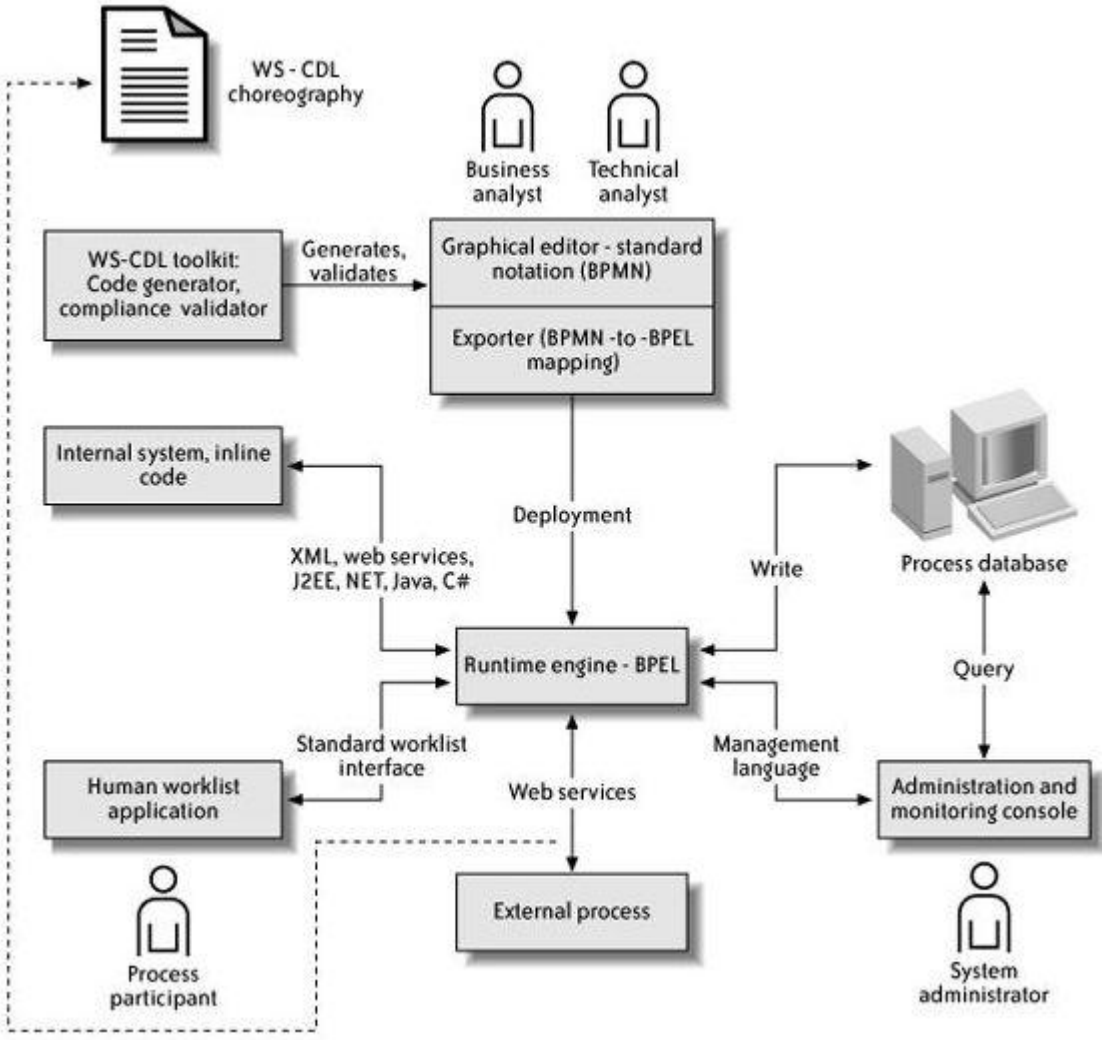
- dugotrajne, jer im je potrebno i više sati ili dana ili nedelja ili čak meseci da se izvrše,
- čuvaju stanje u bazama podataka,
- veći deo vremena čekaju na akciju koja će pokrenuti narednu aktivnost,
- čiji procesi su odgovorni za upravljanje i koordinaciju komunikacija između različitih uloga sistema i ljudi.

Procesne aplikacije treba da zadovolje barem deo navedenih karakteristika.

Razlozi za korišćenje BPM su sledeći:

- formalizacija postojećeg procesa,
- bolje razumevanje i na osnovu toga ugrađivanje poboljšanja poslovnog procesa,
- efikasnije izvršavanje poslovnih procesa zbog korišćenja BPM softvera,
- povećanje produktivnosti i smanjenje učešća ljudi u izvršavanju procesa,
- omogućavanje ljudima da reše komplikovane probleme,
- pojednostavljenje praćenje propisa.

3.4. BPM arhitektura



Slika 13. BPM arhitektura

Najbitniji deo, ili srce sistema, predstavlja procesna mašina, koja izvršava poslovni proces napisan u BPEL jeziku. Poslovni i tehnički analitičari dizajniraju procese koristeći grafički editor, koji podržava BPMN notaciju. Uobičajeno, editor omogućava kreiranje BPEL koda na osnovu BPMN dijagrama.

Na izvršavanje procesa može uticati i komunikacija sa ljudima i sa aplikacijama. Ljudi, koji učestvuju u procesu, koriste aplikacije dijagrama toka, da bi se povezali sa procesnom mašinom, i na taj način pregledali i izvršavali aktivnosti, koji čekaju da budu izvršene. Postoje dva tipa interakcije sa aplikacijama: interna i eksterna. Internim aplikacijama, koje se nalaze u okviru kompanijske mreže, ali van adresnog prostora od procesne mašine, se pristupa pomoću neke od integracionih tehnologija, kao što su Web servisi, J2EE, COM, uobičajeno koristeći XML kao format poruka. Eksterne

aplikacije su uobičajeno Web servisi, sastavni delovi poslovnih procesa drugih kompanija.

BPM sistemski administratori koriste grafičke konzole da bi administrirali i nadgledali procese koji se izvršavaju. Konzole sa procesnim mašinama komuniciraju preko jezika za upravljanje. Procesne mašine čuvanje stanje procesa u bazi podataka. Da bi izvršili neki upit, sistemski administratori mogu da pristupaju bazi procesa.

4. MODELOVANJE POSLOVNIH PROCESA

Sistemsko razmišljanje je osnova koja nam pomaže u shvatanju kompleksnosti sveta kroz razumevanje odnosa njegovih elemenata pre nego samih elemenata. Razumevanje strukture kompleksnih pojava i mogućnosti da se one sagledaju kao celina pre nego skup njihovih delova je fundament sistemskog razmišljanja.

Gledano sa poslovnog aspekta, sistemsko razmišljanje se fokusira na ponašanje cele organizacije, a ne samo na strukturu i njene komponente. Modelovanje poslovnih objekata predstavlja primenu sistemskog razmišljanja na poslovne koncepte u cilju razumevanja i notiranja svih aktivnosti unutar i između organizacija i individua sa jednostavnošću i preciznošću.

Modeli, kao mehanizmi za notiranje rezultata modelovanja, se grade za potrebe razumevanja postojećih ili budućih poslovanja i IT sistema. Modelovanje se uvek odnosi na naglašavanje tj. izostavljanje detalja u smislu naglašavanja bitnih i izostavljanja suvišnih detalja. Na pitanje šta je bitno a šta ne, ne postoji univerzalan odgovor, već to zavisi od svrhe modela, tj. kome je on namenjen. Na osnovu toga možemo reći da je model samo jedan od pogleda na sistem čiji je nivo detalja kontrolisan radi boljeg razumevanja suštine samog sistema.

U prošlosti, razvoj informacionih sistema je uglavnom bio fokusiran na tehničke aspekte kao što su model baze, tok aplikacije, dizajn korisničkog interfejsa itd. Iako veoma važan, ovaj rad nema smisla ukoliko nije usmeren na rešavanje konkretnih poslovnih potreba. Naravno, deo poslovanja koji je premeštan u domen informacionih sistema je tada bio znatno manji nego danas, a potreba za obuhvatanjem sve većeg dela poslovanja u okviru jednog informacionog sistem nameće potrebu za obuhvatnijim metodama modelovanja i samim jezicima i modelima za njihovo notiranje. Objektna tehnologija koje je decenijama korišćena za simulaciju kompleksnih sistema je uvedena i u domen informacionih sistema, tako da sada jezici kao UML omogućavaju primenu tih koncepata i u modelovanju poslovnih procesa.

Evidentna je kompleksnost poslovnih procesa pod kojima se podrazumeva mnogo pojmova, kao što su definicije, instance, uloge, bezbednost, korisnički interfejs, itd. Činjenica da svi elementi koji figuriraju među različitim poslovnim procesima su deljeni, omogućava implementaciju jednog generičkog procesa iz kog svi ostali mogu biti izvedeni. Za sada ne postoji jedinstveno mišljenje i definisan standard šta čini jedan poslovni proces, ali postoji određeni napor koji ulažu OMG, BPMI i drugi čiji koncepti dobijaju javno priznanje i prihvatanje.

Da bi se bilo koji proces adekvatno opisao, mnogo različitih vrsta informacija mora biti inkorporirano u njegov model. Shodno ovome različiti jezici za modelovanje poslovnih procesa pokušali su da odgovore na ove zahteve, ali su pritom akcentovali različite momente kako su nastali kao potreba rešavanja različitih problema iz različitih domena. Imajući prethodnu činjenicu u vidu, ovaj rad je skoncentrisan na uporednu

analizu različitih jezika za modelovanje poslovnih procesa, kroz upoznavanja sa svakim od njih. Poseban osvrt ostvaren je na organizacione, funkcionalne aspekte, kao i aspekte ponašanja i informacija koji je, pre svega, realizovan analizom meta modela svakog od jezika (ako postoji). Za potrebe analize korišćen je i generički meta model [List Korherr 2006] koji je od strane autora predložen kao osnova za svaku analizu jezika za modelovanje poslovnih procesa, neophodnu za odabir adekvatnog jezika za neki konkretan poslovni proces. Ovaj model je upravo skoncentrisan na prethodno pomenute aspekte poslovnih procesa, a detaljnije je objašnjen u uporednoj analizi jezika.

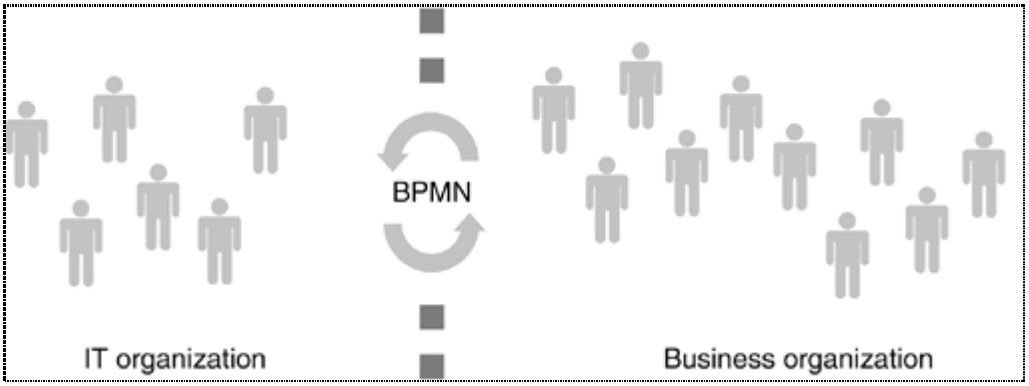
4.1 BPMI standardi

BPMI je neprofitna organizacija stvorena 2000. godine od strane kompanije Intalio, sa ciljem da kreira standarde za BPM. Članice ove organizacije su i BEA, Fujitsu, IBM, IDS Scheer, Pegasystems, PeopleSoft, SAP, SeeBeyond, Tibco, Virtria, WebMethods. I BPMI je članica nekoliko organizacija: W3C, OASIS, OMG, i WfMC. Kroz članstva u ovim organizacijama, BPMI može da učestvuje u diskusijama vezano za skoro sve standarde koje se odnose na BPM.

BPMI je odgovorna za sledeće funkcionalne specifikacije:

- **Business Process Modeling Notation (BPMN)** - Predstavlja jezik napravljen sa ciljem da omogući grafičko kreiranje dijagrama toka poslovnog procesa od strane poslovnih analitičara i divelopera.
- **Business Process Modeling Language (BPML)** - Predstavlja XML jezik čija je svrha da kodira tok izvršavanja poslovnog procesa u formu odgovarajuću za interpretaciju od strane mašine za izvršavanje procesa. Verziju 1.0 je napisao Assaf Arkin iz kompanije Intalio novembra 2002. godine. Prema priznanju BPMI organizacije, BPML je izgubio bitku u odnosu na BPEL jezik, koji se uglavnom koristi kao XML jezik za izvršavanje procesa.
- **Business Process Query Language (BPQL)** - Cilj kreiranja jezika je da bude osnova za praćenje i nadgledanje poslovnih aktivnosti (Business Activity Monitoring - BAM). Nije još publikovan.
- **Business Process Semantic Model (BPSM)** - Treba da predstavlja zajednički model za sve modele poslovnih procesa. Nije još publikovan.
- **Business Process Extension Layers (BPXL)** - Predstavlja standardni skup proširenja za transakcije, poslovna pravila, upravljanje zadacima i ljudsku interakciju. Nije još publikovan.

4.2 BPMN



Slika 14. BPMN kao interfejs između poslovne organizacije i IT organizacije

Stephen White iz IBM kompanije je napisao verziju 1.0 BPMN specifikacije 2004. godine.

BPMN je razvijen je sa ciljem da obezbedi notaciju za definisanje poslovnih procesa, lako razumljivu od strane svih poslovnih korisnika, od poslovnih analitičara koji treba da skiciraju procese, do tehničkih korisnika koji su odgovorni za implementaciju tehnologije pomoću koje se izvršavaju procesi, i na kraju do poslovnih ljudi, koji će upravljati i nadgledati poslovne procese. BPMN omogućava da se prevaziđe praznina između dizajniranja poslovnog procesa i njegove implementacije. BPMN predstavlja intuitivnu notaciju, koja se lako usvaja i koristi.

BPMN se definiše preko Business Process Diagram (BPD), koji je baziran na tehnici blok-dijagrama i koji je prilagođen za kreiranje grafičkih modela operacija poslovnih procesa. Business Process Model (BPM) predstavlja mrežu grafičkih objekata, odnosno aktivnosti i kontrola kojima se definiše njihov redosled izvršavanja.

BPMN dijagram je sastavljen od skupa grafičkih elemenata. Elementi omogućavaju lak razvoj jednostavnih dijagrama, različiti su jedan u odnosu na drugi i izgledaju blisko većini poslovnih analitičara. Na primer, aktivnosti su predstavljene kao pravougaonici, a odluke kao rombovi.

Četiri osnovne kategorije BPD elemenata su:

- objekti toka,
- objekti veza,
- swimlines,
- artifakti.

4.2.1 Objekti toka

Objekti toka predstavljaju osnovne grafičke elemente za definisanje ponašanja poslovnih procesa. Objekti toka su Event, Activity i Gateway elementi.

Event element se prikazuje pomoću kruga i predstavlja nešto što se dešava tokom izvršavanja poslovnog procesa. Event element utiče na izvršavanje toka procesa i obično ima uzrok (okidač) i posledicu (rezultat). Event elementi su kategorizovani po fazi u kojoj se dešavaju u procesu (Start, Intermediate i End) i po tipu (Basic, Message, Timer, Rule, Exception, Cancellation, Compensation, Link, Multiple, Termination).

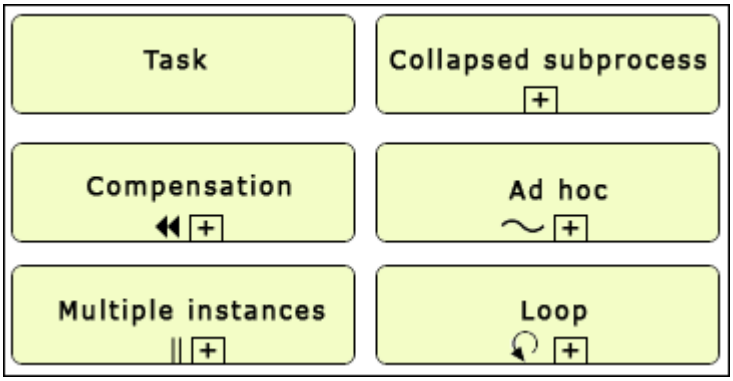
Start	Intermediate	End	Name
○	◎	●	Basic
✉	✉	✉	Message
◷	◷		Timer
▤	▤		Rule
	Ⓝ	Ⓝ	Exception
	⊗	⊗	Cancellation
	◀◀	◀◀	Compensation
→	→	→	Link
✸	✸	✸	Multiple
		◑	Termination

Slika 15. Prikaz kompletnog skupa Event elemenata

Tabela: Opis BPMN Event elemenata			
Type	**Start**	**Intermediate**	**End**
Basic	Nosilac položaja ili početak podprocesa.	Nosilac položaja	Nosilac položaja ili kraj podprocesa.
Message	Proces se startuje po prijemu poruke. (na primer, poziv Web servis metode implementirane u procesu).	Proces čeka poruku (na primer, čeka odgovor na poslati zahtev).	Poruka koja se šalje određenom procesu (na primer, za poziv Web servisa).
Timer	Startni događaj definiše vreme kada će biti okinut (na primer, svake nedelje u podne).	Tačka kada je došlo definisano vreme.	
Rule	Uslov, definisan u okviru procesa, je ispunjen (na primer, proces se startuje kada cena dostigne određeni nivo).	Uslov je ispunjen. Koristi se samo za upravljanje izuzecima.	
Exception		Podiže ili hvata grešku.	Generiše grešku.
Cancellation		Zaustavlja izvršavanje aktivnosti.	Prekida transakciju.
Compensation		Okida i izvršava akciju kompenzacije.	Izvršava akciju kompenzacije.
Link	Link startni događaj povezuje se na link završni događaj od procesa na istom nivou.	Veza od ili do druge aktivnosti.	Povezuje se na link start događaj od procesa na istom nivou.
Multiple	Dva ili više okidača mogu pokrenutu proces, ako se desi bilo koji od njih, proces se startuje. Okidači mogu biti poruke, timer, pravilo, ili tip.	Dva ili više okidača mogu da nastave da čekaju proces, ako se bilo koji od njih izvrši, proces se nastavlja.	Kada se proces završi, očekuje se nekoliko rezultata (na primer, potrebno je poslati nekoliko poruka).
Termination			Prekida sve aktivnosti u procesu. Izvršava kompenzaciju ili upravlja izuzecima.

Activity element se predstavlja pomoću pravougaonika sa zaobljenim uglovima i on je opšti pojam za posao koji se obavlja. Activity element može biti prost ili složen. Složena aktivnost, koja se naziva i proces, takođe može imati svoj skup prostih ili složenih aktivnosti, kao i druge BPMN elemente. Tipovi ovog elementa su: Task i Sub-Process (predstavljen je kao pravougaonik sa zaobljenim uglovima sa znakom plus u

centru donjeg dela). Znak plus predstavlja prikaz složenog procesa, detaljni o njemu se prikazuju na odvojenom dijagramu. Proces može biti označen sa simbolima koji predstavljaju kompenzaciju, višestruko pojavljivanje, petlje i ad hoc obradu. Kompenzacija predstavlja aktivnost koja poseduje logiku za vraćanje u prethodno stanje. Ad hoc proces sadrži skup aktivnosti koje se mogu izvršiti u bilo kom redosledu. Petlja aktivnost može biti konfigurisana kao standardna petlja (while ili until petlja), ili da podržava višestuke instance (foreach petlja).



Slika 16. BPMN Activity elementi

Tipovi Task elementa:

Service	Poziv Web servisa
Receive	Čekanje poruke (alternativa za izgradnju događaja)
Send	Slanje poruke
User, Manual	Posao se izvršava od strane čoveka-učesnika
Script	Logika sadržana u programskom ili skripting jeziku (na primer, izvršavanje dela Java koda)
Reference	Koristi definiciju drugog posla u okviru procesa; ne duplira definiciju, već je deli

Gateway element predstavljaju glavni način za kontrolu tokova u okviru BPMN dijagrama. Predstavlja se kao romb i koristi se za označavanje uslovnih grananja ili spajanja puteva u okviru procesa.

Symbol	Name
	Exclusive OR
	Exclusive OR
	Exclusive OR (Event-based)
	Exclusive OR
	Complex
	Parallel

Slika 17. BPMN Gateway elementi

Gateway Exclusive OR element, koji je baziran na podacima, koristi kontrolne strukture if-then-else i switch sa međusobno isključivim slučajevima. U situaciji kada se vrši grananje u okviru procesa, tačno jedan uslov mora biti ispunjen.

Drugi tip Gateway Exclusive OR elementa, koji je baziran na događaju, koristi pick kontrolnu strukturu. U situaciji kada se vrši grananje u okviru procesa, svaka grana vodi do jednog čvora sa događajem. Kontrola prolazi kroz granu gde se okida prvi događaj, a sve ostale grane se ignorišu.

Gateway Inclusive OR element je sličan gateway Exclusive OR elementu, koji je baziran na podacima, izuzev što dozvoljava prolazak kroz svaku granu switch kontrolne strukture, čiji uslov je ispunjen.

Gateway Complex element se uglavnom koristi kod spajanja puteva u okviru procesa, kada se proverava definisani izraz da bi se odredila putanja kojom će se nastaviti izvršavanje procesa.

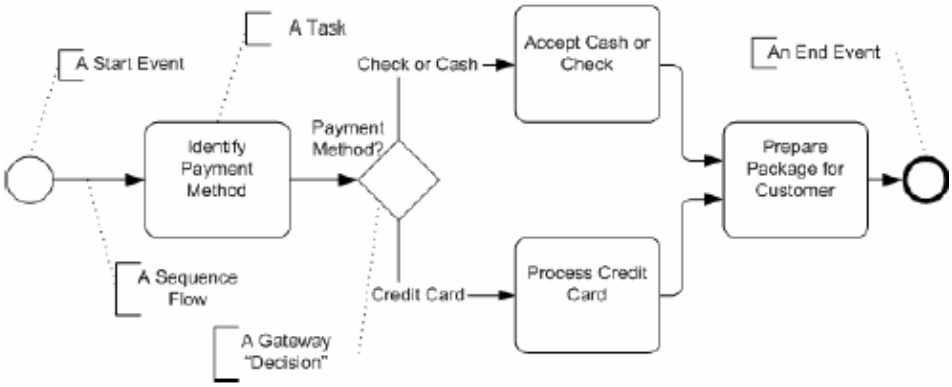
Gateway Parallel element, koristi flow kontrolnu strukturu, i kod grananja u okviru procesa, dozvoljava prolazak kroz svaku od definisanih putanja. Kod spajanja, blokira nastavak izvršavanja procesa, sve dok se ne izvrše sve definisane grane.

4.2.2 Objekti veza

Objekti toka se povezuju međusobno korišćenjem objekata veza. Postoje tri osnovna tipa elemenata koji obezbeđuju ovu funkciju.

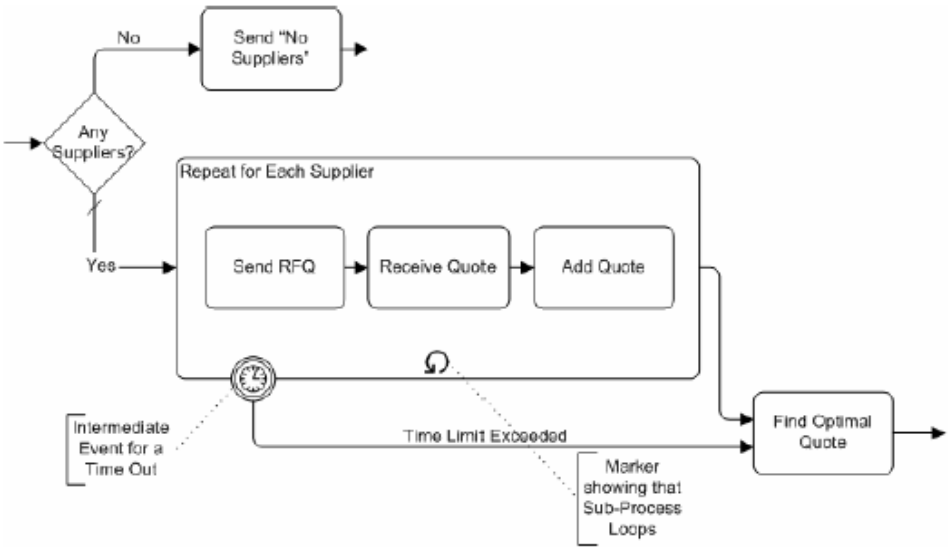
Sequence Flow		Element je predstavljen punom linijom sa punom strelicom i koristi se da prikaže redosled izvršavanja aktivnosti u okviru procesa.
Message Flow		Element je predstavljen isprekidanom linijom sa nepopunjenom strelicom i koristi se da prikaže tok poruka između dva odvojena učesnika u procesu, koji primaju i šalju poruke.
Association		Element je predstavljen linijom od tačaka sa otvorenom strelicom i koristi se da poveže podatke, tekst i druge artifakte sa objektima toka, odnosno da prikaže ulaze i izlaze iz aktivnosti.

Za modelare koji žele da kreiraju dijagrame sa relativno niskim nivoom preciznosti za potrebe dokumentacije ili komunikacije, osnovni elementi pružaju mogućnost za jednostavno kreiranje razumljivih dijagrama, kao što je primer na sledećoj slici.



Slika 18. Primer jednostavnog poslovnog procesa

Za one kojima je potrebna veća preciznost prilikom kreiranja dijagrama, dodatni detalji mogu biti inkorporirani u osnovne elemente.



Slika 19. Primer procesa opisan sa više detalja

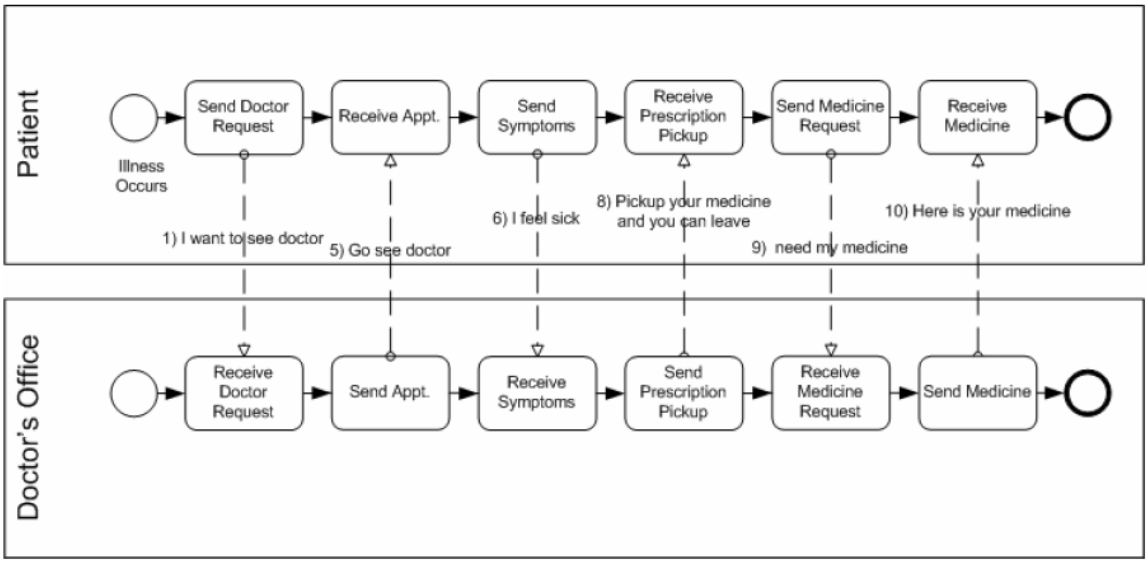
4.2.3 Swimlanes

Swimlines čine Pool elementi i, u okviru Pool elementa, Lane elementi. Pool prikazuje aktivnosti jednog učesnika, najčešće preduzeća. Lane element u okviru Pool elementa prikazuje sastavne delove jednog učesnika, najčešće odeljenja preduzeća.

Swimlines		
Pool	Name	Pool element predstavlja učesnika u procesu. Može biti određeni poslovni entitet (kompanija) ili se može nalaziti u nekoj poslovnoj ulozi (kupac, proizvođač). Takođe, ima ulogu grafičkog kontejnera za odvajanje skupa aktivnosti od drugih Pool elemenata, obično u kontekstu B2B situacija.
Lane	Name Name Name	Element omogućava podelu unutar Pool elementa i proširuje ga ili vertikalno ili horizontalno, sa ciljem da organizuje i kategorizuje aktivnosti.

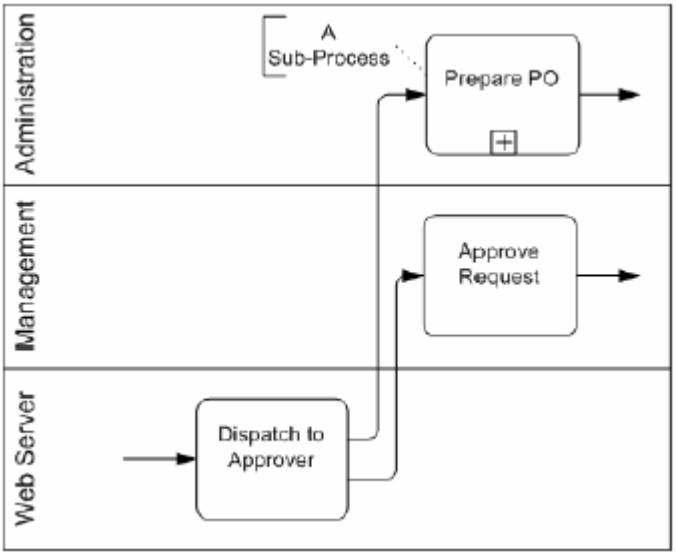
Particije se koriste za potrebe modelovanja odvojenih poslovnih entiteta koji su i fizički odvojeni na dijagramu (Slika xxx). Aktivnosti koje se nalaze u okviru jedne Particije se smatraju internim delom procesa i spojene su Sekvencom koja ne može

izlaziti van okvira Particije. Kao mehanizam komunikacije između dva učesnika tj. dve particije koriste se Poruke.



Slika 20. Primer BPD-a sa participama

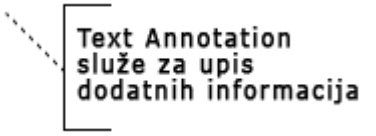
Staze su kao pojam mnogo bliže konvencionalnom shvatanju plivačkih staza kako se one koriste za razdvajanje aktivnosti procesa unutar jedne organizacije ili uloge (Slika xxx). Sekvenca može prelaziti granice Staza u okviru Particije, dok Poruke ne mogu biti korišćene između objekata toka u okviru Linija iste Particije.



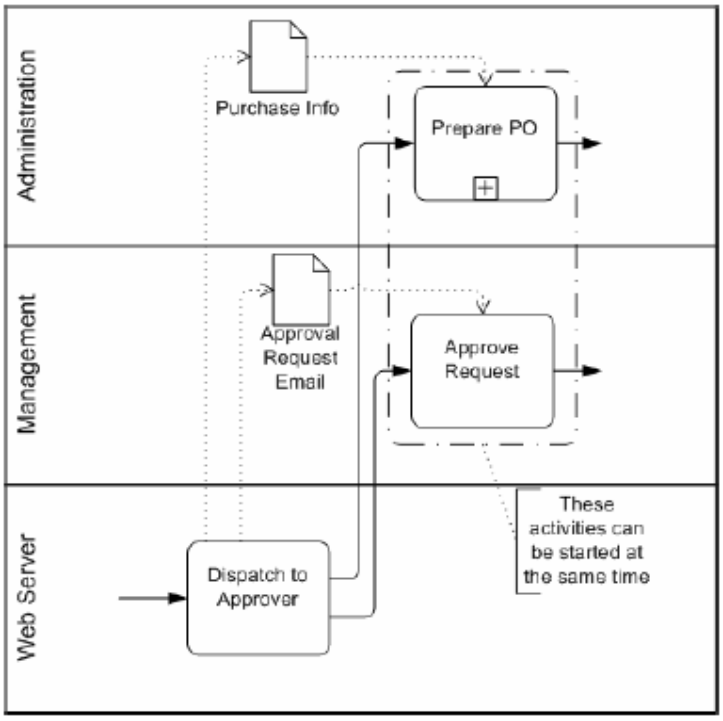
Slika 21. Primer BPD-a sa Stazama

4.2.4 Artifakti

BPMN je dizajniran da dozvoli onima koji ga koriste (modelari, alati za modeliranje) određenu fleksibilnost u proširenju osnovne notacije, kao i u obezbeđenju mogućnosti da se doda kontekst, odgovarajući za specifičnu situaciju koja se modeluje. Bilo koji broj artifakta može se dodati u dijagram.

Data Object		Predstavljaju mehanizam za prikaz podataka, koji su ulaz ili izlaz iz aktivnosti. Povezani su sa aktivnostima preko Associations elemenata.
Group		Koristi se za dokumentaciju ili u svrhu analize. Ne utiče na redosled toka aktivnosti.
Annotation		Obezbeđuje dodatne tekstualne informacije o BPMN dijagramu.

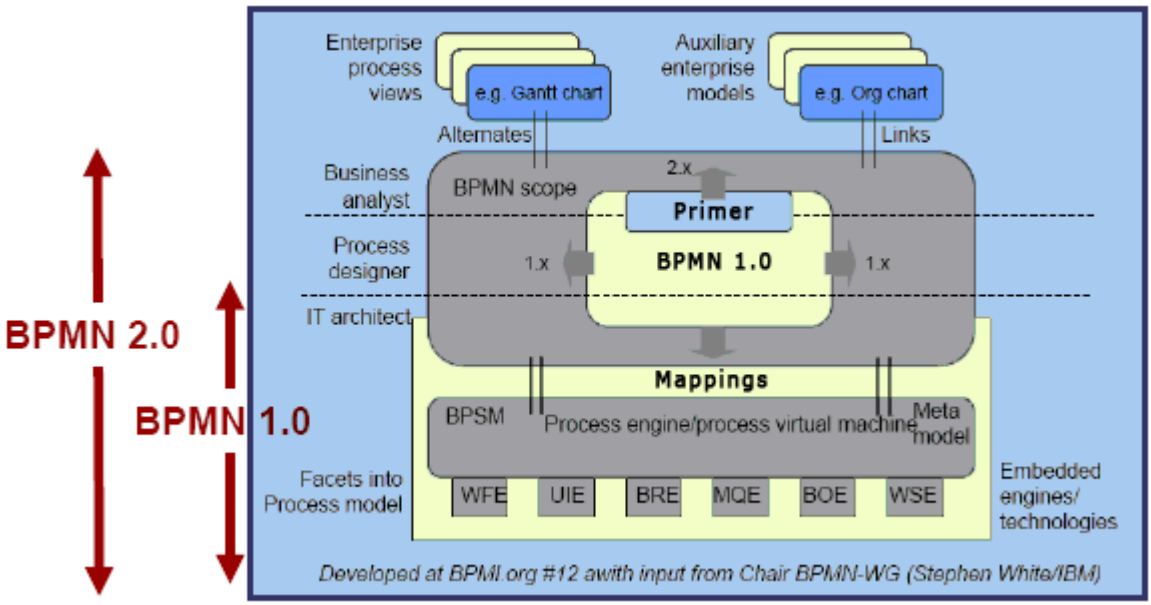
Modelari za potrebe modelovanja mogu kreirati svoje dodatke kako bi obogatili opis procesa sa više detalja. Naravno, osnovna struktura, utvrđena Događajima, Aktivnostima, Sekvencom, itd. neće biti promenjena dodacima, što se može i videti poređenjem Slike 19. sa prethodnom slikom.



Slika 22. Proces sa Grupama, Objektima i Komentarima

BPMN verzija 2.0

BPMI organizacija, odgovorna za kreiranje BPMN i drugih standarda vezanih za modeliranje poslovnih procesa, priznaje da je BPMN 1.0 samo prvi korak u razvoju ove notacije. Trenutno se razvija nova verzija BPMN 2.0, koja će sadržati poboljšanja i zadovoljiti potrebe organizacija i analiza lanaca vrednosti. Nova verzija treba da sadrži nove definicije vezane za meta model i formate za razmenu poruka.



Slika 23. Funkcionalni obim BPMN 1.0 i BPMN 2.0

Primena BPMN-a

Modelovanje poslovnih procesa pokriva izuzetno širok spektar informacija namenjen različitim korisnicima, tj. učesnicima poslovnog procesa. BPMN je dizajniran da odgovori ovakvim različitim zahtevima u modelovanju, ali su svetlu različitih ciljeva pri modelovanju razlikuju se dva osnovna tipa modela koji se mogu kreirati u BPD-u:

- Kolaborativni (javni) B2B procesi i
- Interni (privatni) poslovni procesi.

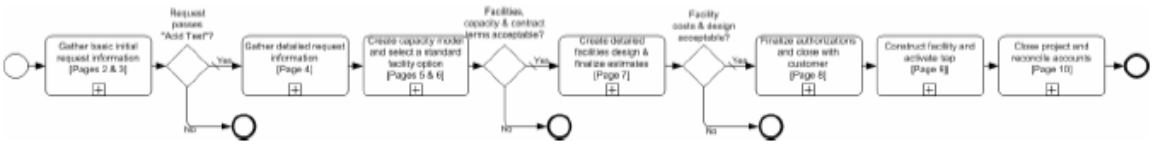
Kolaborativni procesi

Kolaborativni procesi se može shvatiti kao proces iz ugla interakcije između dva i više poslovna entiteta. Modeli ovog tipa, tj. za ove aspekte poslovnih procesa generalizovani i sa opšte tačke gledišta. Naime, oni nemaju perspektivu samo jednog od učesnika, već prikazuju interakciju između dva korisnika. Interakcija je prikazana kao sekvenca aktivnosti tj. skup razmenjenih poruka između učesnika. Aktivnosti za učesnike u kolaboraciji se mogu shvatiti kao tačke razdvajanja, odnosno spajanja. Stoga, ova vrsta dijagrama prikazuje taj javni aspekt procesa, vidljiv svakom od učenika u kolaboraciji.

Interni poslovni procesi

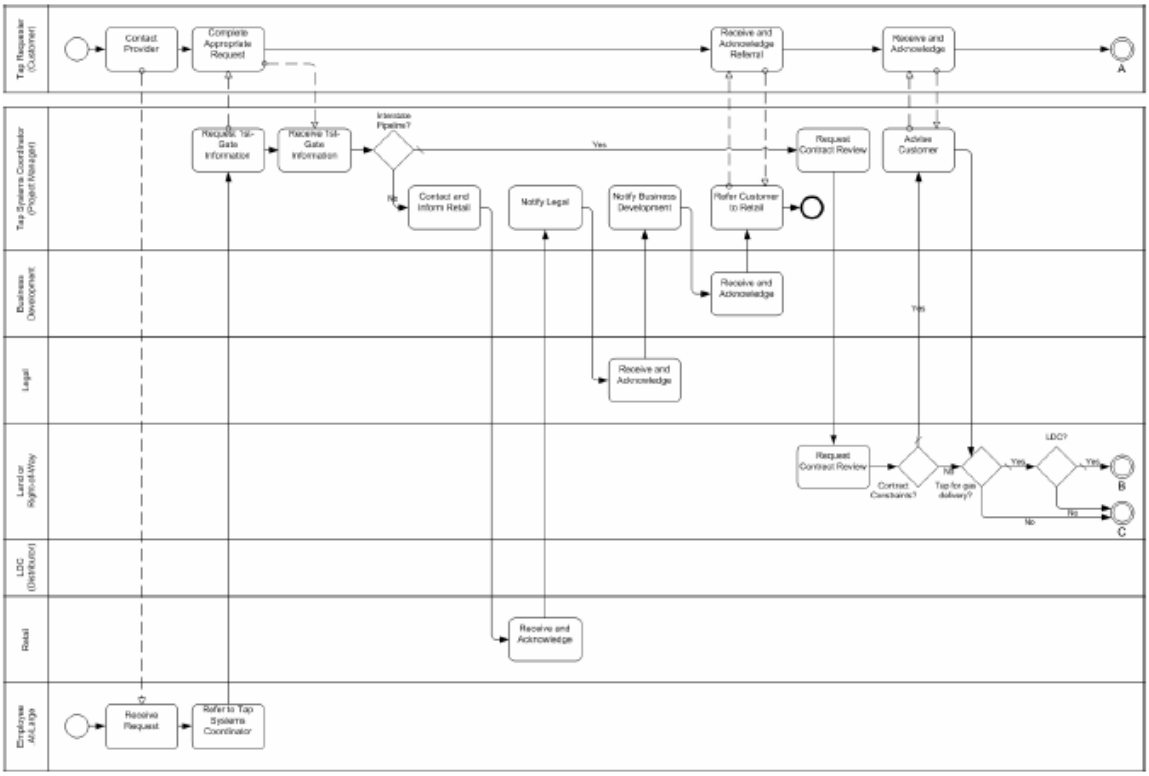
Interni poslovni procesi su procesi sagledani iz perspektive jedne organizacije, gde i ako ovi procesi ostvaruju interakciju sa eksternim učesnicima oni definišu aktivnosti koje generalno nisu javne.Ukoliko se koristi koncept plivačkih staza, sve interne aktivnosti će biti smešte u okviru jedne Particije gde Sekvenca ne može prelaziti njene granice, za razliku od Poruka kojima se eventualno prikazuje interakcija procesa sa nekim eksternim procesima. Na ovaj način se jednim dijagramom može prikazati više internih poslovnih procesa i njihova interakcija.

Najčešći pristup u modelovanju poslovnih procesa je taj da se na početku proces posmatra sa apstraktnije tačke pri čemu se prikazuju samo osnovni procesi sa aspekta manje granularnosti. Zatim se uključivanjem detalja svaki deo (aktivnost) dalje razrađuje na odvojenim dijagramima. Broj nivoa i preciznost zavise od primenjene metodologije gde je evidentno da je BPMN od toga nezavistan. Na sledećoj slici prikazan je primer jednog apstraktnog procesa, koji je u stvari skup podprocesa.



Slika 24. Primer apstraktnog poslovnog procesa

Na narednoj slici detaljnije je prikazan prvi podproces sa prethodnog dijagrama koji sadrži dve particije, jednu za kupca, a drugu za organizaciju koja pruža uslugu. Takođe, može se primetiti da dijagram prikazuje interne procese organizacije dok je prikazan proces kupca apstraktan i podrazumeva samo one aktivnosti koje su potrebne da bi se realizovala interakcija. Aktivnosti unutar organizacije su izdeljeni po stazama kako bi se prikazale odgovornosti pojedinih organizacionih jedinica ili radnih mesta.



Slika 25. Primer konkretizovanog procesa

4.3. BPEL

Kompozicija servisa u poslovne procese zahteva definisanje aktivnosti i poruka između uključenih servisa. WSDL obezbeđuje osnovni tehnički opis i specifikacije za poruke koje se razmenjuju. Međutim, opis koji obezbeđuje WSDL ne ide dalje od opisa jednostavne interakcije između klijenta i servisa. Ove interakcije mogu biti sinhrone ili asinhrone, kao i sa ili bez stanja. Ove relacije su neodgovarajuće za izražavanje, modeliranje i opis složenih kompozicija više servisa. Zbog postojećih ograničenja WSDL jezika, neophodan je mehanizam za opis kompozicije servisa u kompleksnije procese. [Juric]

Kompozicija servisa u poslovne procese može biti realizovana korišćenjem nekog od poznatih programskih jezika (Java, C#, ...). Međutim, korišćenje programskih jezika za kompoziciju servisa najčešće ima za posledicu kreiranje nefleksibilnih rešenja,

najviše zbog nejasne granice između toka procesa i poslovne logike, koji ne smeju biti čvrsto povezani. [Juric]

Opšta potreba za kreiranjem rešenja za automatizaciju izvršavanja poslovnih procesa zahteva i standarde i specijalizovani jezik za kompoziciju servisa u poslovne procese, koji će obezbediti mogućnost izražavanja poslovnih procesa na standardizovan način korišćenjem opšte prihvaćenog jezika. BPEL predstavlja takav jezik, koji je veoma brzo postao dominantan standard. Glavni cilj BPEL jezika je da standardizuje proces automatizacije između servisa. [Juric]

Business Process Execution Language (BPEL) predstavlja na XML zasnovan jezik za definiciju i opis poslovnih procesa. BPEL jezik je prvobitno kreiran od strane IBM, Microsoft i BEA. Sada je u nadležnosti neprofitnog konzorcijuma OASIS (Organization for the Advancement of Structured Information Standards), koji razvija, održava i promoviše standarde za elektronsko poslovanje, uključujući ebXML, SGML, UDDI, PKI i BPEL.

BPEL je nastao od WSFL i XLANG jezika sa ciljem da omogući "programiranje na veliko".

Korišćenjem BPEL jezika se mogu definisati jednostavni i kompleksni poslovni procesi. Uz neka proširenja, BPEL je sličan tradicionalnim programskim jezicima, jer ima elemente tipa petlje, grananja, variable, dodeljivanja, koji dozvoljavaju definisanje poslovnih procesa na algoritamski način. BPEL je specijalizovani jezik fokusiran na definisanje poslovnih procesa.

Najvažniji elementi BPEL jezika se odnose na pozivanje Web servisa. BPEL omogućava da se poziv Web servisa obavi na jednostavan način, kako sinhrono, tako i asinhrono. Operacije se mogu pozivati i sekvencijalno i paralelno. Takođe omogućava opis povratnih funkcija, upravljanje greškama, obezbeđuje podršku za procese koji se dugo izvršavaju, kao i kompenzaciju, što omogućava da se opovrgne posao koji je odradio proces, koji se nije završio uspešno. BPEL jezik omogućava [Juric]:

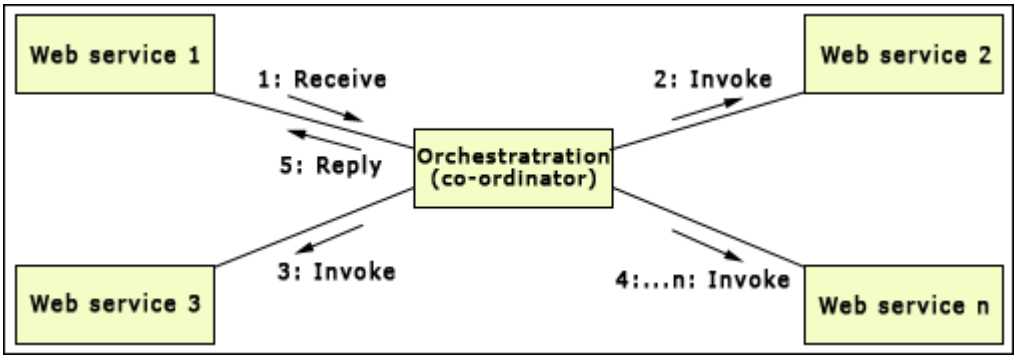
- opis logike poslovnog procesa kroz kompoziciju servisa;
- sastavljanje većih poslovnih procesa od manjih procesa i servisa;
- upravljanje sinhronim i asinhronim pozivanjem operacija servisa, kao i upravljanje povratnim funkcijama koje se izvršavaju kasnije;
- poziv operacija servisa sekvencijalno ili paralelno;
- selektivno kompenzira neuspešno završene aktivnosti;
- održavanje višestruke dugotrajne transakcione aktivnosti;
- nastavak prekinutih ili neuspešno završenih aktivnosti, što minimizuje potrebu za njihovim ponovnim izvršavanjem;
- rutiranje dolazećih poruka u odgovarajuće procese i aktivnosti;
- uzajamno povezivanje zahteva u okviru ili van poslovnog procesa;

- planiranje izvršavanja aktivnosti zasnovano na vremenu izvršavanja i definiše njihov redosled izvršavanja;
- izvršavanje aktivnosti paralelno i definisanje kako će se paralelni tokovi sinhronizovati u odnosu na uslove sinhronizacije;
- struktuiranje poslovnih procesa u nekoliko oblasti;
- upravljanje događajima zasnovanim na porukama i vremenu.

4.3.1. Orkestracija i koreografija

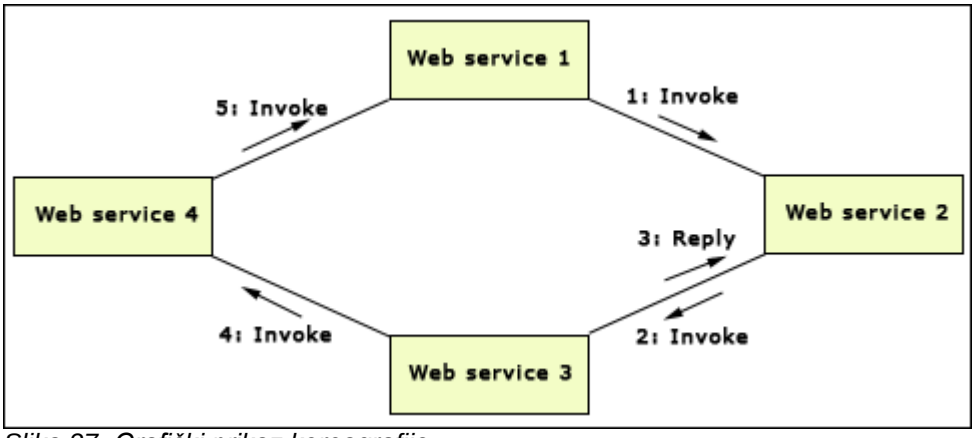
U zavisnosti od zahteva, kompozicija servisa može upućivati na privatne ili javne procese, zašta se koriste termini orkestracija i koreografija.

U orkestraciji, centralni proces ima kontrolu nad uključenim Web servisima i koordiniše izvršavanje različitih operacija Web servisa. Ovo se izvršava prema zahtevima orkestracije. Uključeni Web servisi ne znaju da su uključeni u kompoziciju i da su deo višeg poslovnog procesa. Jedino centralni koordinator orkestracije zna ovo, tako da je orkestracija centralizovana sa eksplicitnim definicijama operacija i redosledom pozivanja Web servisa. Orkestracija se uobičajeno koristi u privatnim poslovnim procesima.



Slika 26. Grafički prikaz orkestracije

Koreografija ne zavisi od centralnog koordinatora. Svaki Web servis uključen u koreografiju tačno zna kada treba da izvrši svoje operacije i sa kim treba da sarađuje. Koreografija predstavlja zajednički napor usmeren na razmenu poruka u javnim poslovnim procesima.



Slika 27. Grafički prikaz koreografije

Posmatrajući iz perspektive kompozicije Web servisa za izvršavanje poslovnog procesa, orkestracija ima prednost u odnosu na koreografiju. Orkestracija je fleksibilnija, mada granica između orkestracije i koreografije nije uvek uočljiva. Orkestracija ima sledeće prednosti [Juric]:

- tačno se zna ko je odgovoran za izvršenje celog poslovnog procesa;
- mogućnost dodavanja Web servisa, bez njihovog znanja o tome da su deo poslovnog procesa;
- obezbeđivanje alternativnih scenarija u slučaju pojave greške.

4.3.2. Izvršni i apstraktni procesi

BPEL obezbeđuje podršku za orkestraciju i koreografiju kroz izvršne i apstraktne poslovne procese.

Izvršni poslovni procesi su procesi koji su sastavljeni od skupa postojećih servisa i specificiraju tačan algoritam aktivnosti i ulaznih i izlaznih poruka. Ovi procesi se izvršavaju od strane BPEL mašina. Izvršni procesi su veoma važni, jer su oni direktan odgovor na problem automatizacije poslovnih procesa. Sa BPEL izvršnim procesima se može specificirati tačan algoritam kompozicije servisa na relativno jednostavan i otvoren način, koji se može izvršiti na odgovarajućoj BPEL mašini. Izvršni procesi popunjavaju prazninu između specifikacija poslovnih procesa i koda odgovornog za njihovo izvršavanje.

Kada se definiše BPEL izvršni proces, u stvari se definiše novi Web servis, koji je kompozicija postojećih servisa. Interfejs novog BPEL kompozitnog servisa koristi skup "port types", preko kojih obezbeđuje izvršavanje operacija kao kod bilo kog drugog servisa.

Apstraktni poslovni procesi nisu izvršni. Uglavnom se koriste da bi:

- opisali ponašanje servisa bez tačnog saznanja u kom će poslovnom procesu učestvovati;

- definisali protokole saradnje između više učesnika i precizno opisali eksterno ponašanje svakog učesnika.

Apstraktni procesi se retko koriste. Najčešći scenario je da se koriste kao šablon za definisanje izvršnih procesa. Mogu se koristiti da zamene skup pravila uobičajeno izraženih govornim jezikom, koji je često dvosmislen.

Iz BPEL ugla, poslovni proces predstavlja kolekciju koordinisanih poziva Web servisa, kao i povezanih aktivnosti koje proizvode rezultat, u okviru jedne ili više organizacija.

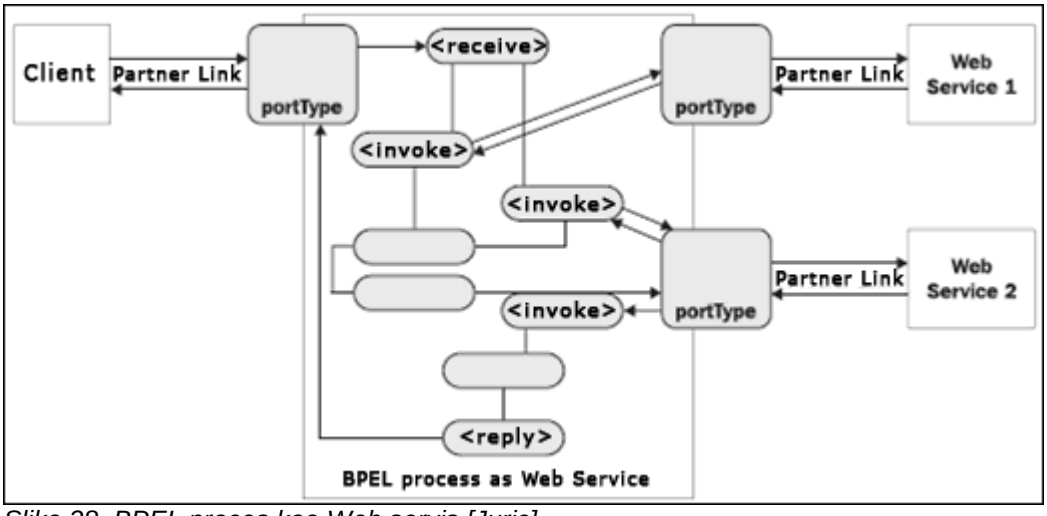
BPEL proces se sastoji od više koraka. Svaki korak se naziva aktivnost. BPEL podržava primitivne i strukturne aktivnosti.

Primitivne aktivnosti predstavljaju osnovne elemente i koriste se za zajedničke poslove, kao što su:

- pozivanje Web servisa, korišćenjem <invoke>
- čekanje na zahtev, korišćenjem <receive>
- generisanje odgovora za sinhrone operacije, korišćenjem <reply>
- manipulisanje sa varijablama podataka, korišćenjem <assign>
- ukazivanje na greške i izuzetke, korišćenjem <throw>

Primitivne aktivnosti možemo kombinovati u mnogo kompleksnije algoritme, koje definišu korake poslovnog procesa. Da bi kombinovao primitivne aktivnosti, BPEL podržava nekoliko strukturnih aktivnosti. Najvažnije su:

- <sequence> za definisanje skupa aktivnosti, koje će biti pozivane u određenom redosledu;
- <flow> za definisanje skupa aktivnosti, koje će biti pozivane paralelno;
- <switch> za implementaciju grananja;
- <while> za definisanje petlji.



Slika 28. BPEL proces kao Web servis [Juric]

4.4. Mapiranje BPMN dijagrama u BPEL izvršni jezik

Svrha kreiranja BPMN dijagrama je dvostruka, sa jedne strane treba da omogući poslovni pogled na poslovni proces, a sa druge strane da omogući generisanje izvršnog koda pomoću BPEL jezika.

Bez obzira na svoje ime - Business Process Execution Language – BPEL se realno ne bavi poslovnim procesima, bar ne na onaj način kako se njima bavi BPM zajednica. Na primer, sa BPM stanovišta, poslovni proces je sastavljen od različitih podprocesa koji mogu biti ugnježdeni ili ulančani međusobno. Svaki podproces može biti vlasništvo ili može biti izvršavan od strane različitih poslovnih jedinica. BPEL ne poznaje koncept podprocesa, koji je osnovna jedinica za mogućnost ponovnog korišćenja u BPM. To ne znači da se korišćenjem BPEL ne može kreirati proces od ugnježdenih i ulančanih komponenti. Takođe, BPEL ne podržava koncepte ljudi učesnika identifikovanih preko uloga, grupa ili korisničkih imena. U BPEL, aktivnosti procesa se dodeljuju servis endpointima, koji se razrešavaju preko URL. Ne postoji odvojeni endpoint servisa za svaku ulogu, grupu ili imenovanog korisnika. Umesto toga, postoji jedan endpoint, koji rutira zadatak prema odgovarajućem korisniku ili redu čekanja. [Silver]

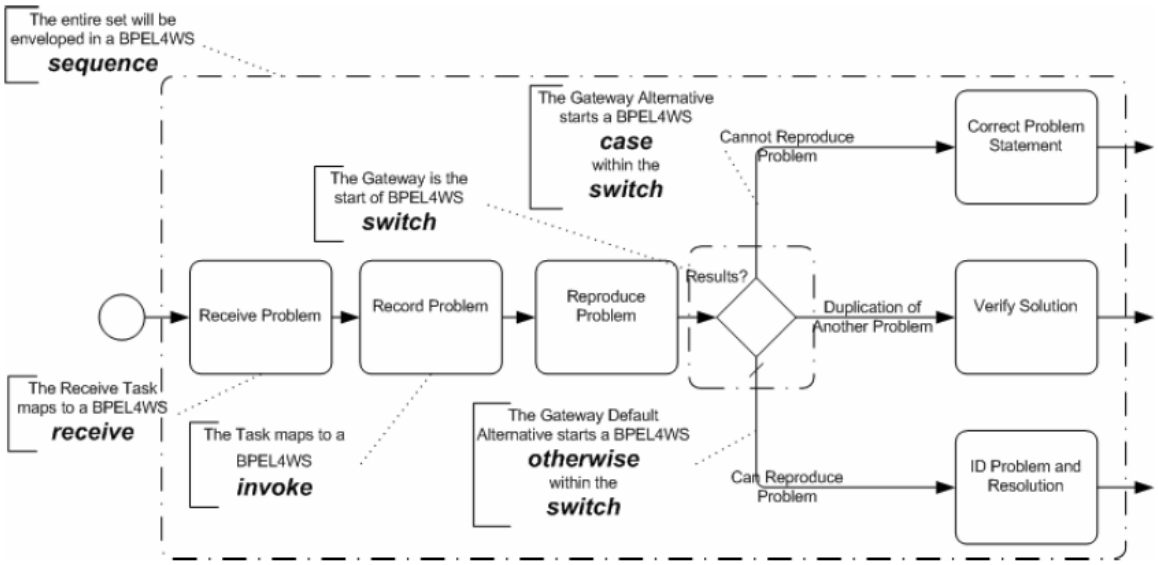
BPMN specifikacija sadrži više od 50 strana koje prikazuju mapiranje BPMN u BPEL jezik. Cilj je da se premosti jaz između grafičkog dizajna i izvršavanja procesa. Sam proces mapiranja nije jednostavan, pre svega zbog postojanja različitog broja elemenata.

Pažljivije posmatranje, pokazuje da je mapiranje samo delimično, ostavljajući po strani modele sa nestruktuiranim topologijama, kao i delove kao što su OR-join ili kompleksni Gateway. Osim toga, pošto je mapiranje samo opisano, ono predstavlja predmet interpretacija. Uopšteno govoreći, mnogo dvosmislenosti se može naći u BPMN specifikaciji vezano za mapiranje u BPEL jezik. [P4 BPMN]

4.4.1. Mapiranje BPMN dijagrama

BPMN dijagram može biti sastavljen od skupa (polu-) nezavisnih komponenata, koje se prikazuju u različitim Pool elementima. Za Pool elemente ne postoji specifično mapiranje. U slučaju da sadrži Process elemente ("white box"), može se mapirati u BPEL proces. Međutim, u slučaju mapiranja sadržaja Process elementa, može postojati jedan ili više izvedenih procesa, neophodnih za upravljanjem složenim ponašanjem, kao što su petlje. Ako je Pool element tipa "black box", njegovi atributi mogu se koristiti za definisanje PartnerLink BPEL elementa.

BPMN	**BPEL**
Start event: Basic, Message, Timer, Rule, Link	Receive element sa atributom createInstance = yes
Start event: Multiple	Pick element sa atributom createInstance = yes
End event: Message	Invoke element ili Reply element
End event: Error	Throw element
End event: Compensation	Compensate element
End event: Terminate	Terminate element
End event: Link	Invoke element
Intermediate event: Error	Upravljanje greškom ili podizanje greške.
Activities: MI	Nije moguće direktno mapiranje. Mogu se koristiti različiti elementi.
Activities: Subprocesses	Koristiti Invoke element za pokretanje.
Gateway: Exclusive data-based	Switch element
Gateway: Exclusive event-based	Pick element
Gateway: Inclusive	Više Switch elemenata u okviru Flow elementa.
Gateway: Complex	Nije moguće direktno mapiranje.
Gateway: Parallel	Flow element
Sequence Flow	Modeliranje toka kontrole od jednog elementa do drugog ili eksplicitno, korišćenjem Flow link elemenata, ili implicitno, korišćenjem kontrolnih struktura kao što su Sequence, While ili Switch.
Message Flow	Nije moguće direktno mapiranje.
Pool	Nije moguće direktno mapiranje.
Lane	Nije moguće direktno mapiranje.
Artifact	Nije moguće direktno mapiranje.
Assotiation	Nije moguće direktno mapiranje.
Exception Flow	Sve aktivnosti koje slede Intermediate Event, a sve dok se Exception Flow ne vrati u Normal Flow, mogu da se mapiraju u *faultHandlers* BPEL element.
Compensation Association	Moguće je mapiranje u *compensationHandler* element.



Slika 29. Grafički prikaz mapiranja BPMN dijagrama u BPEL kod [White]

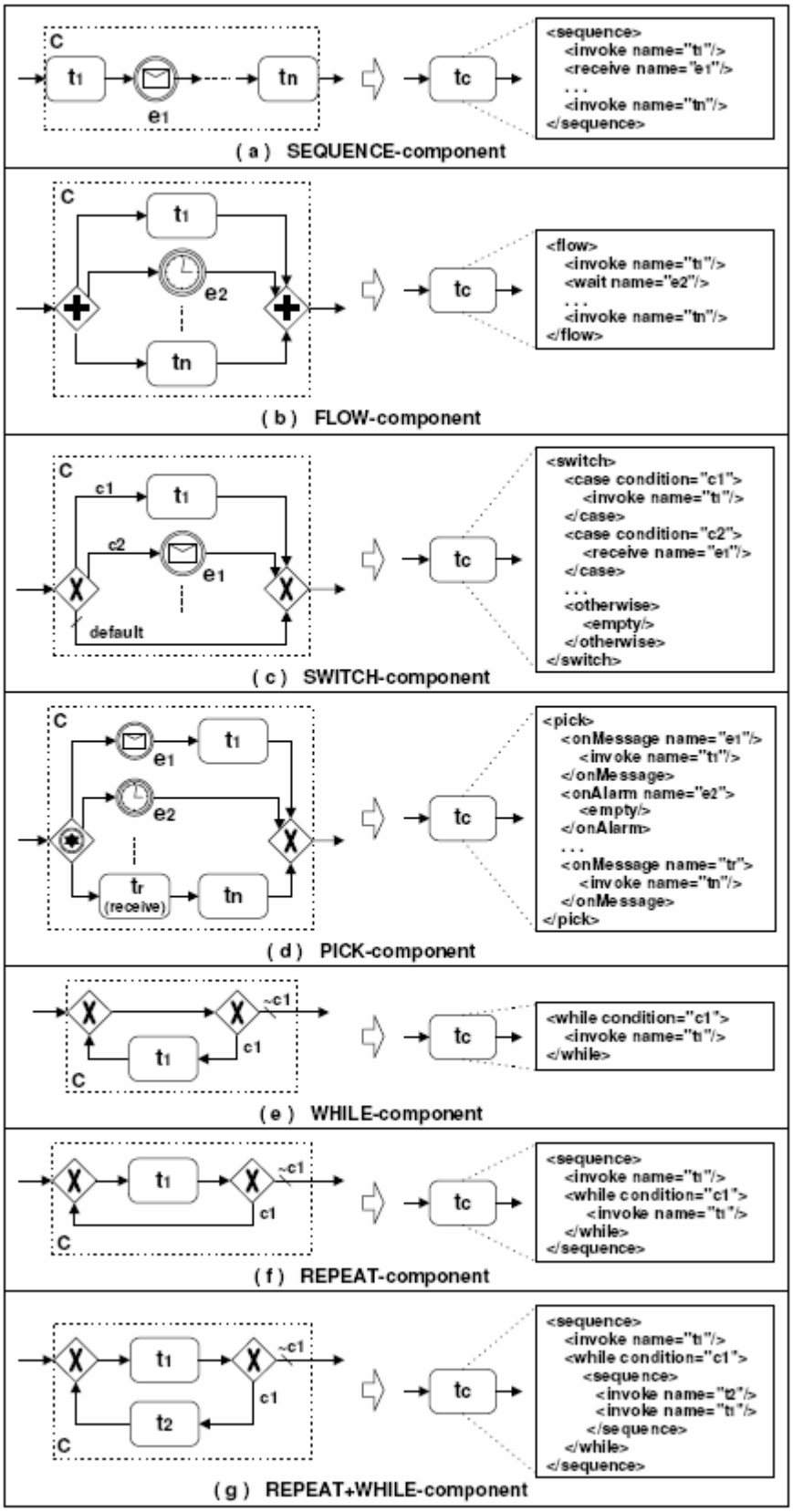
Problem nemogućnosti jednoznačnog mapiranja BPMN notacije i BPEL izvršni jezik, proizvođači softvera rešavaju neki od sledećih načina:

- Grafički predstave BPEL proces i nazivaju ga BPMN.
- Koriste jedan alat za kreiranje BPMN dijagrama, a drugi za formiranje BPEL procesa.
- Od kreiranog BPMN dijagrama kreira se BPEL izvršni kod, prilagođen određenoj procesnoj mašini, pri čemu se uobičajeno zahteva da BPMN ne sadrži elemente, koji se ne mogu direktno mapirati.

Trenutna situacija je takva, da bi se pomoću bilo kod dostupnog alata generisao BPEL od BPMN, potrebno je da se reše dva problema:

- prvi, da se dodaju svi detalji vezani za implementaciju pre nego što se BPMN izveze u BPEL, što često dovodi do uobičajenog pitanja "da li očekujete da će poslovni analitičar postati programer, ili obrnuto, da li očekujete da će programer postati poslovni analitičar",
- i drugi, zbog loše struktuiranog BPMN grafikona, ponekad izvoz nije moguć, ili za posledicu ima nevažeći BPEL kod.

Problemi koji se javljaju kod generisanja BPEL koda na osnovu BPMN dijagrama su karakteristični kod svih modela na visokom nivou. Modeli na visokom nivou moraju da se na odgovarajući način sinhronizuju sa modelima na nižem nivou, što svakako nije jednostavno uraditi, osim u retkim situacijama kada je moguće obaviti mapiranje jedan-na-jedan. Na slici je prikazano nekoliko primera BPMN dijagrama i njima odgovarajući BPEL kod.



Slika 30. Primeri elemenata BPMN dijagrama sa mapiranjem u BPEL

4.5. Uporedna analiza

Uporedna (komparativna) analiza je složen saznajni proces, tj. metod kojim se na osnovu analize strukture raznih ili sličnih predmeta ili pojava vrši upoređivanje svojstava, struktura i zakonitosti tih pojava. Komparativnom analizom se utvrđuju strukturalne, funkcionalne i genetičke istovetnosti ili različitosti više pojava [ŠEŠIĆ71].

Za potrebe realizacije uporedne analize neophodno je uspostaviti odgovarajući kriterijum. Kriterijumom se obezbeđuje uslov ili pravilo kojim se omogućava izbor, odluka ili sud o nečemu.

4.5.1. Kriterijum poređenja

Prema [List Korherr 2006] može se uspostaviti meta model kao kriterijum evaluacije jezika za modelovanje poslovnih procesa. Ovaj meta model mora zadovoljiti širok spektar koncepata prisutnih u poslovnim procesima. Predloženi meta model razmatra jezike za modelovanje poslovnih procesa sa četiri aspekta: organizacionog, funkcionalnog, informacionog i sa aspekat ponašanja (Slika 24). Ova četiri aspekta su proširena sa kontekstom poslovnog procesa da bi bilo moguće razmatrati informacije kao što su ciljevi procesa ili mere.

U osnovi, bilo je prilično teško izvršiti procenu jezika za modelovanje poslovnih procesa jer precizan opis često nedostaje, elementi nekad imaju nejasno značenje, a meta modeli nisu dostupni za neke jezike. Neki BPML-ovi imaju složene definicije, dok su druge neprecizne i ostavljaju upotrebu elemenata interpretaciji korisnika.

Generalno, funkcionalni i bihejvioristički delovi su veoma dobro predstavljeni u svim jezicima za opis procesa dok su organizacioni i informacioni delovi samo delimično podržani, a kontekst poslovnog procesa nije eksplicitno podržan.

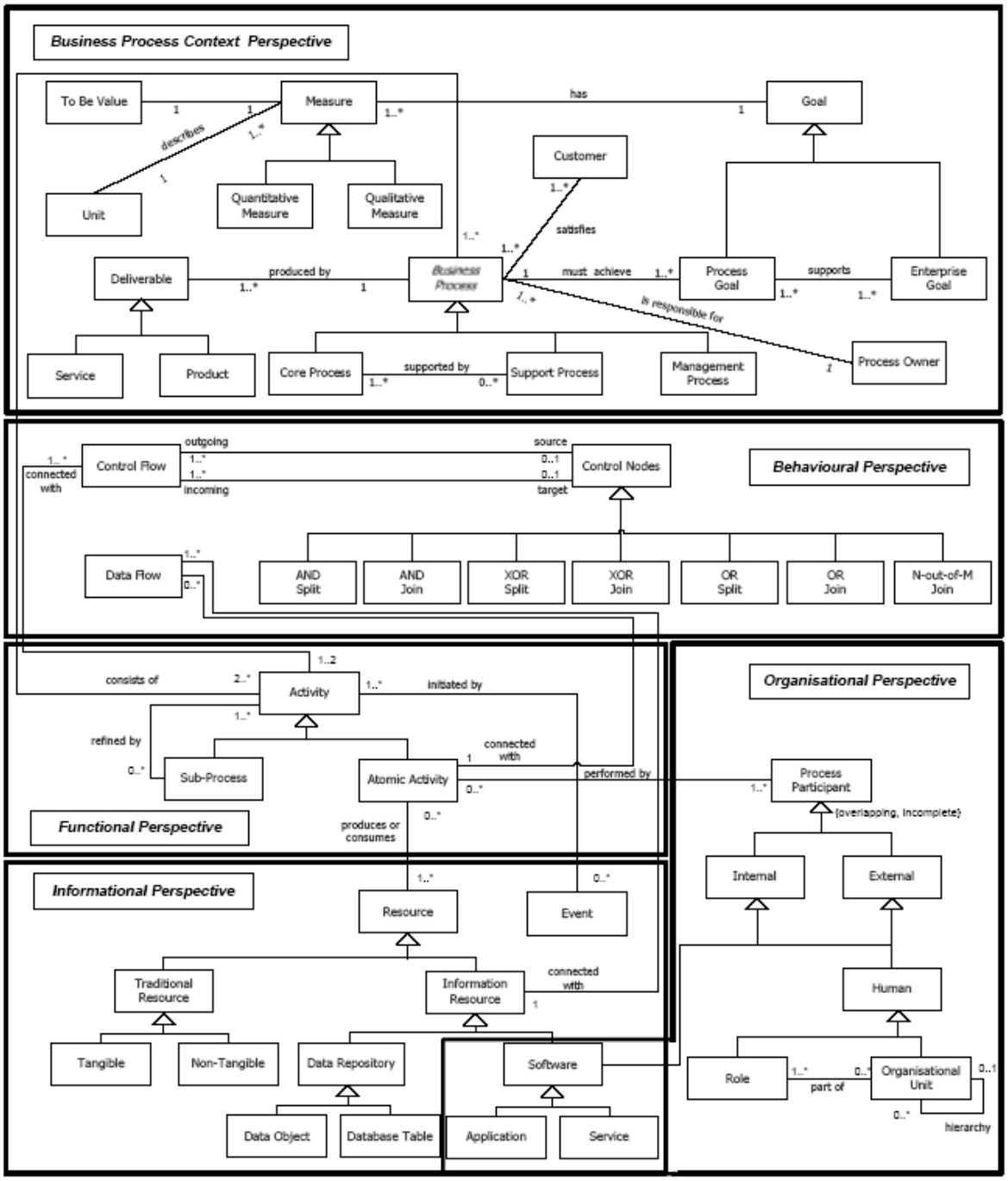
Organizaciona perspektiva predstavlja gde i od strane koga se neki element procesa izvršava. Sa tog aspekta, a prema WfMC koriste se četiri osnovna tipa učesnika procesa: organizaciona jedinica, uloga, čovek i automatizovan resurs. Na predloženom meta modelu, učesnik procesa može biti interni (unutar organizacije) ili eksterni (kupac, dobavljač ili čak drugi sistem). Identifikovana su tri tipa resursa koji mogu biti izvršioci.

Orgaznizacioni aspekt je do nekog obima podržan u skoro svim BPML-ovima. Izuzeci su IDEF 3 and Petri Nets, koji vodi poreklo iz sistemskog i softver inženjerstva. Većina drugih BPML-ova koncentrišu se mnogo više na poslovni proces i stoga uključuju koncept uloga. Nijedan jezik za opis poslovnoh procesa ne predstavlja softver kao eksplicitni koncept, a mali broj eksplicitno pokazuje da li uloga pripada organizaciji ili je eksterna. Mnogo BPML-ova koristi jedan koncept da prikaže sve tipove učesnika procesa (AD, RAD, BPMN) i ne pravi razliku između tipova.

Informacioni deo je bolje razvijen kod skorijih BPML-ova kao što su AD, BPDM, BPMN, i EPC. Njihova podrška za jezike za izvršavanje je takođe dobra. Samo EPC

podržava eksplicitnu notaciju elemenata za tradicionalne resurse i zato je pogodna za analizu procesa.

Funkcionalni aspekt predstavlja izvršne elemente procesa. Oni su predstavljeni preko aktivnosti, tj. atomske aktivnosti i podprocesa. Podprocesi dalje mogu obuhvatati više aktivnosti.



Slika 31. Meta model za ocenu jezika za modelovanje poslovnih procesa

Da bi se poboljšala fleksibilnost poslovnog procesa, postoji potreba da se smanji vreme između modeliranja procesa i transformacije u kod koji se može izvršiti. Stoga, budući jezici za modelovanje poslovnih procesa moraju da obezbede jezike za izvršavanje i za uzvrat, ponude eksplicitne elemente za notaciju za sve delove. Ali ogroman broj različitih elemenata bi bio zbunjujući za opis procesa, pa je osnovni set elemenata, kao što je BPMN takođe potreban.

4.5.2. BPMN/BPEL i UML

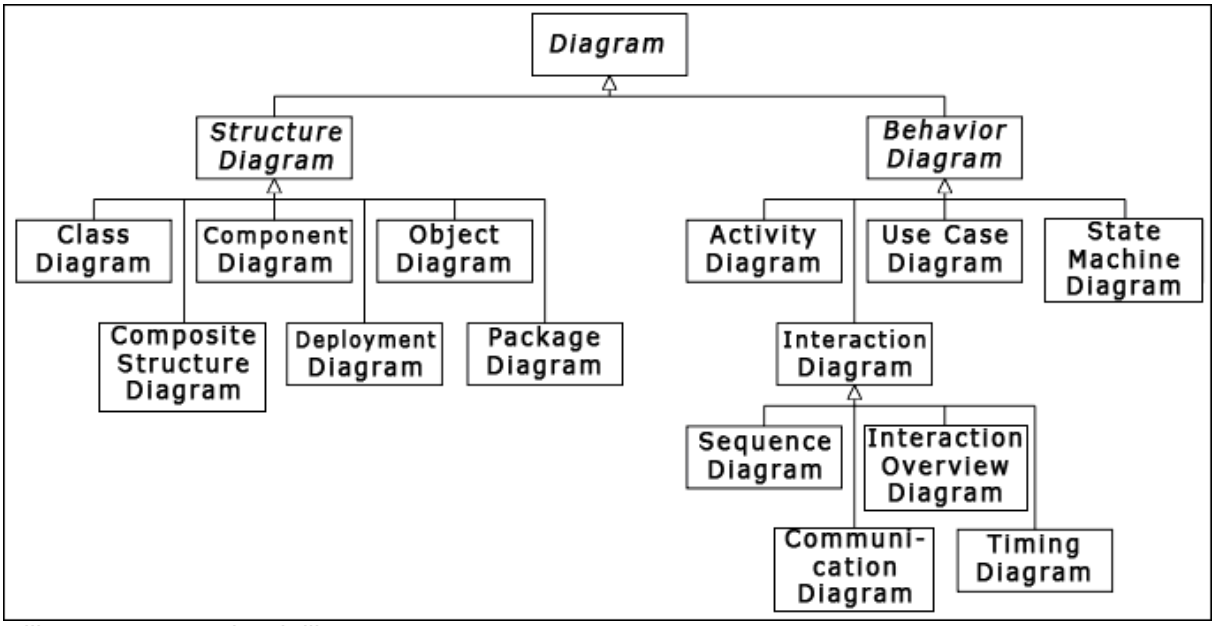
Veoma često se postavlja pitanje o svrsishodnosti uvođenja novih jezika i notacija za modeliranje poslovnih procesa, s obzirom da već određeni niz godina postoji standardizovan i veoma moćan jezik za modeliranje Unified Modeling Language (UML).

UML predstavlja jezik za specifikaciju, vizualizaciju, konstrukciju i dokumentovanje proizvoda softverskog sistema, kao i za modeliranje poslovnih procesa i drugih ne – softverskih sistema. UML predstavlja skup najboljih praktično primenjenih inženjerskih modela koji su korišćeni u razvoju velikih i složenih sistema.

Razlog što je UML postao standardni jezik za modeliranje je sadržan u činjenici da je nezavistan od programskih jezika.

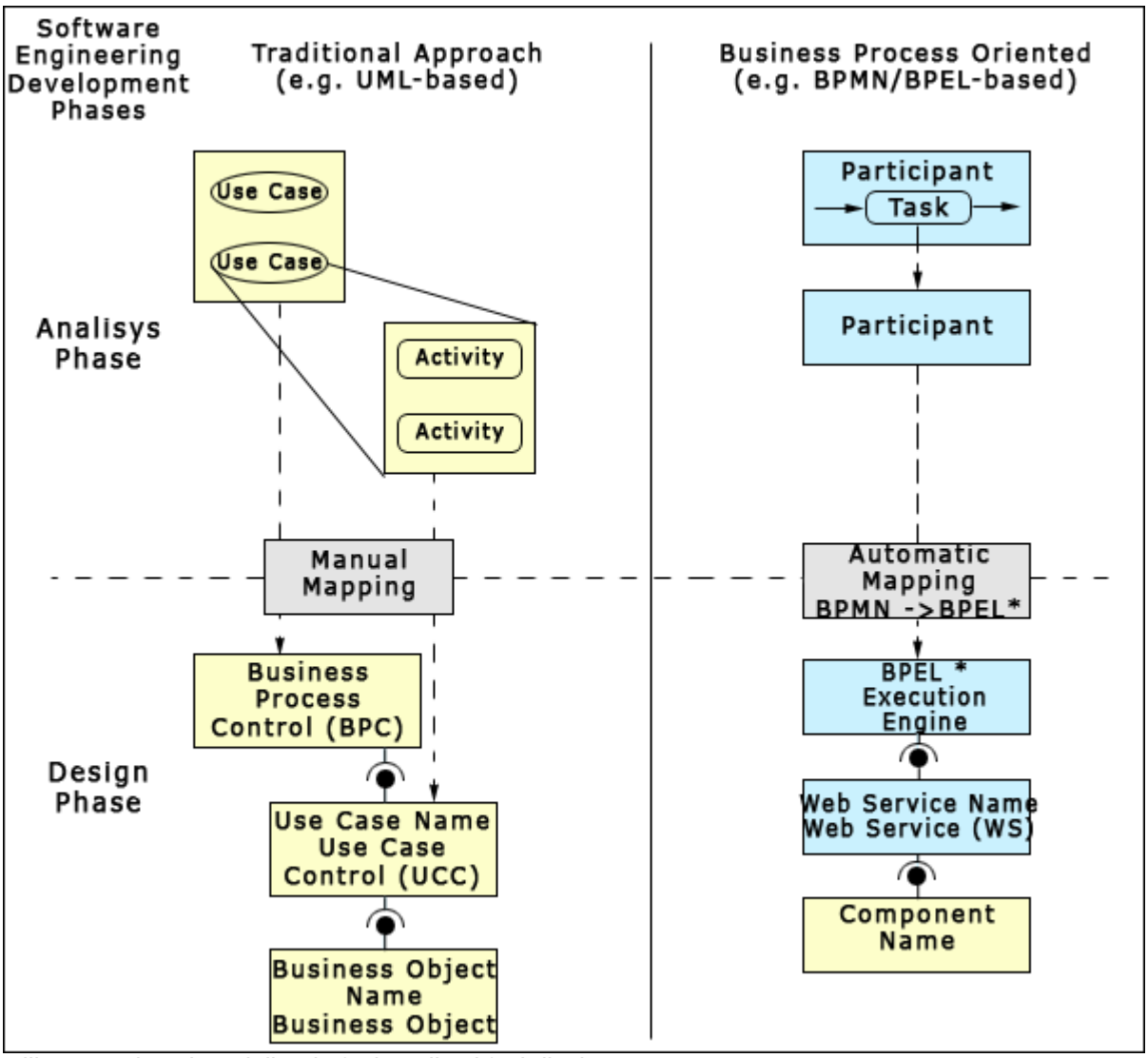
UML podržava bogat skup grafičkih elemenata, što omogućava da na razumljiv način opiše klase, komponente, čvorove, aktivnosti, dijagrame toka, slučajeve korišćenja, objekte, stanje, kao i da modeluje relacije između navedenih elemenata. UML podržava i mogućnost proširenja preko stereotipnih elemenata.

UML podržava 13 tipova dijagrama, razvrstanih u 3 kategorije: strukturni dijagrami, dijagrami ponašanja i dijagrami interakcije, koji predstavljaju podskup dijagrama ponašanja.



Slika 32. UML 2.0 tipovi dijagrama

UML jezik podržava faze analize i dizajna u okviru procesa razvoja aplikacija. Faza analize obuhvata prikupljanje zahteva korisnika, i formalizuje se pomoću UML slučaja korišćenja i dijagrama aktivnosti, koji zajedno predstavljaju poslovnu logiku softverske aplikacije. U okviru dizajn faze, poslovna logika se mapira u komponente, koje čine arhitekturu aplikacije. Ovo mapiranje se obavlja manuelno, što dovodi do neefikasnosti, nekonzistentnosti, kao i nefleksibilnosti. Sve do sada, UML nije obezbedio mogućnost da na odgovarajući način mapira slučajeve korišćenja i dijagrame aktivnosti u jezik za izvršavanje procesa. Zbog toga, neophodno je da se izabere drugi jezik za programiranje poslovnih procesa. BPMN notacija i BPEL jezik su rešenja za navedeni problem. [Dev SOA]



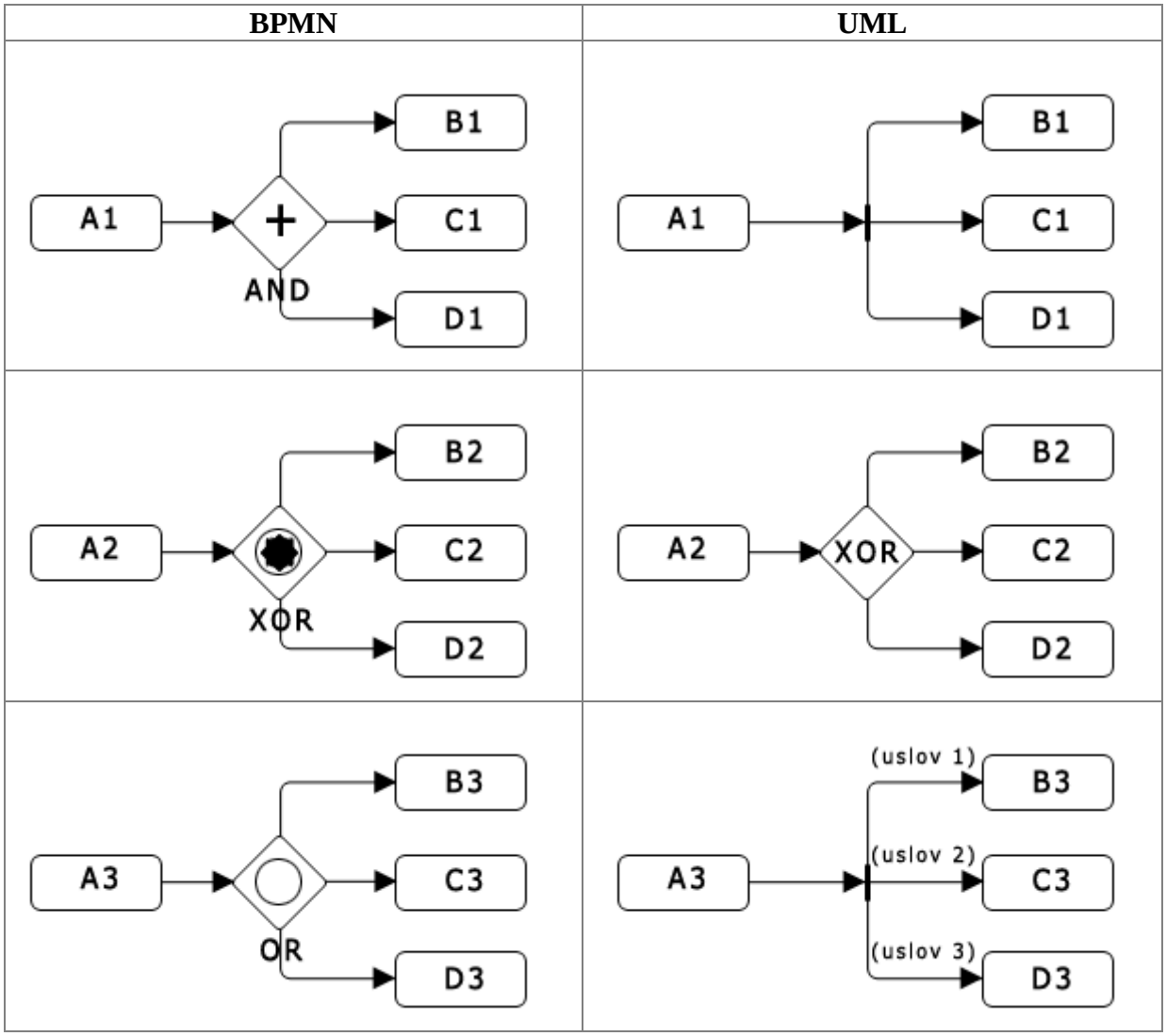
Slika 33. Pristupi modeliranju fazi analize i fazi dizajna

Ukratko, razlozi za postojanje BPMN notacije i BPEL jezika, pored UML jezika, su sledeći:

- namenjeni su različitim ciljnim grupama:
  - UML je namenjen softverskim analitičarima, dizajnerima i diveloperima, jer nudi specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta za kompjuterski zasnovane sisteme,
  - BPMN i BPEL su namenjeni poslovnim analitičarima, arhitektama sistema i softverskim inženjerima, jer nude specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta vezanih za poslovni domen,
- zasnivaju se na različitim paradigmama:
  - UML je zasnovan na objektno orijentisanom pristupu,
  - BPMN i BPEL su zasnovani na procesnom pristupu,
- različitost u nivou implementacije:

- UML jeziku nedostaje mogućnost implementacije poslovnih modela, koji su visoko konceptualni.

Bez obzira na postojanje navedenih različitosti, ipak postoji mogućnost da se za predstavljanje poslovnih procesa koriste UML dijagrami aktivnosti. Na sledećoj slici je prikazano nekoliko uzora za opis ponašanja poslovnih procesa pomoću BPMN notacije i preko UML dijagrama aktivnosti.



Slika 34. Nekoliko uzora predstavljenih i pomoću BPMN i pomoću UML jezika

Bitno je naglasiti, da je programiranje zasnovano na procesnom pristupu, sledeći korak u razvoju softverskog inženjeringa. Prema tome, procesno orijentisano programiranje ne treba da zameni, već da dopuni postojeće pristupe programiranju, kao što su strukturno, objektno i komponentno orijentisano programiranje. [Dev SOA]

4.7. BPDM

Meta model definicije poslovnih procesa (BPDM - Business Process Definiton Meta model) kreiran od OMG-a predstavlja generički meta model poslovnih procesa. On omogućava predstavljanje modela poslovnih procesa nezavisno od notacije. Ujedno, predstavlja i jedan robustan mehanizam za serijalizaciju BPMN modela.

BPDM predstavlja deljen rečnik koncepata za modelovanje procesa, tj. može se zamisliti kao univerzalna sintaksa procesa. Osnovna ideja BPDM-a je mogućnost podrške najvećem broju osnovnih tipova modela procesa, kao i razmene modela bez narušavanja sturkture i izvršne semantike.

Kako bi se postigao ovaj cilj, BPDM definiše dva fundamentalna i komplementarna pogleda na proces: Orkestraciju i koreografiju.

- Koncept orkestracije je u BPDM-u predstavljen procesom u tradicionalnom smislu, sa sekvencom aktivnosti, grananjima i sinhronizacijom tokova.
- Koreografija predstavlja apstraktniji pogled na proces, gde se opisuju interakcije entiteta u kolaboraciji od kojih svaki od njih može imati interne procese orkestracije. Ove interakcije su često struktuirane i kao Protokoli interakcije (u smislu da interakcije uglavnom imaju redosled). Protokoli uglavnom postoje i unutar organizacije između različitih uloga, kao i izvan, između eksternih učesnika procesa.

Gledano sa aspekta modelovanja poslovnih sistema, orkestracija i koreografija predstavljaju „dve strane istog novčića“, tj. dva pogleda na jedan sistem. BPDM dozvoljava da se ova dva pogleda tretiraju nezavisno, a ipak sa deljenjem osnovnih informacija.

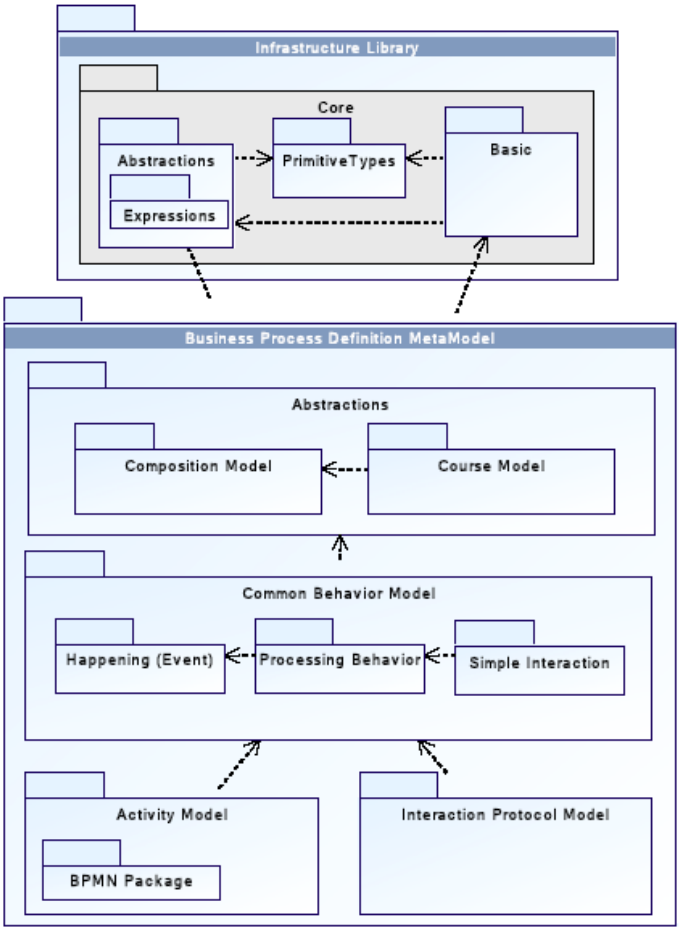
Primena BPDM-a

U svojoj osnovi, BPDM omogućava interoperabilnost između alata od kojih svaki predstavlja proces na sopstven način. Ovo takođe omogućava i portabilnost između alata u smislu da ukoliko dva alata podržavaju BPDM kao serijalizacioni mehanizam, tada model kreiran u prvom se može izvršiti u drugom. Stoga, BPDM predstavlja i tehnološku specifikaciju za proizvođače alata kako se serijalizacija procesa treba vršiti.

BPMN meta model

BPDM predstavlja eksplicitan meta model i serijalizacioni mehanizam BPMN-a. U osnovi, „minimalno slaganje“ u podršci BPDM-u je u stvari podrška onom podskupu elemenata BPDM-a koji pružaju podršku BPMN-u, tj. čine njegov meta model. Podskup elemenata koji čine podršku BPMN-u je smešten u CBM (Common Behavior Model) i čine ga događaji, interakcije i koraci procesa. Proizvođači alata koji žele da pruže podršku BPMN-u u mogu to učiniti samo na osnovu ovih elemenata dok se ostatak BPDM može ignorisati.

CBM – Common Behavior Model



Slika 35. Struktura BPDM-a

Dve, prethodno pomenute, fundamentalno različite perspektive jesu zasnovane upravo na njihovim razlikama, ali i međusobno povezane njihovom semantikom. U cilju podrške ovim momentima, BPDM mora omogućiti korišćenje ovih perspektiva nezavisno ali uz deljenje zajedničkih informacija. Konkretno, podrška se pre svega fokusira na dve osnovne perspektive poslovnih procesa: orkestraciju i koreografiju. Prvi pogled, orijentisan ka aktivnostima je ono što proces radi (Activity Model u CBM-u), dok drugi orijentisan na interakcije (protokole) predstavlja obaveze svakog od učesnika u procesu (Interaction Protocol Model).

CBM predstavlja strukturu kojom se integrišu ova dva različita aspekta procesa međusobnim deljenjem informacija. Jedan aspekt CBM-a je nazvan „Događaji“, pokrivajući pritom sve one elemente koji podrazumevaju vreme – bilo u jednom trenutku ili određenom vremenskom periodu. Drugi je orijentisan na „Ponašanje procesa“, a koji na događaje dodaje sekvence generičkih koraka (primenljivih i na orkestraciju i koreografiju). Treći aspekt je „jednostavna interakcija“ koja se odnosi na definiciju u ponovnu upotrebu protokola, takođe primenljiva i orkestracija i na koreografiju. Ova tri aspekta CBM-a pružaju podršku specifikaciji zavisnosti između Aktivnosti (u orkestraciji),

između interakcija učesnika (u kolaboraciji) i između interakcija orkestriranih procesa sa okruženjem (kombinacija orkestracije i koreografije).

Iznad CBM-a su dva nivoa apstrakcije – model toka i model kompozicije. Oni se mogu zamisliti kao funkcionalnosti koje obezbeđuju konzistentnost i validacione mehanizme koji garantuju da se transformacija izvršava tačno. Jedan od ključnih tačaka ovih apstrakcija je ispravna transformacija modela iz jedne u drugu notaciju.

Model kompozicije se može shvatiti i kao radni okvir kojim se definiše kako klase meta modela sarađuju zajedno, kao i kako se odnose prema izvršnom okruženju. On u stvari pruža principe i osnovne mehanizme ponovne upotrebe, (de)kompozicije, interkonekcije i instanciranja, kao i rekurzivne mehanizme za organizovanje bilo koje grupe elemenata modela saglasnih sa BPDM-om. Ujedno, pruža i mogućnost proširenja BPDM-a na druga izvršna okruženja.

Model kompozicije takođe definiše kompoziciju odnosa između entiteta (npr. kako je Aktivnost deo Procesa, ili tok Interakcije deo Protokola Interakcija). Takođe, opisuje i rekurzivnu strukturu kompozicije, tj. kako je npr. Proces sastavljen iz „tipiziranih“ delova koji su grupa aktivnosti ili u neki slučajevima čak i podproces. Rekurzivna dekompozicija se može odnositi i na npr. Interakcije gde se jedna može dekomponovati na više njih. Definiše i odnos između entiteta na istom nivou dekompozicije, u smislu kako npr. neka Aktivnost dolazi posle neke druge u okviru Procesa ili Interakcije u okviru Protokola Interakcija.

Takođe, model kompozicije proširuje mogućnosti BPDM-a u smislu pružanja specifične podrške za izvršnu semantiku, odnosno da se modeli kompozicija tačno i dosledno preslikaju u različita izvršna okruženja. Rezultat ovoga je tzv. semantička interoperabilnost gde dva različita izvršna okruženja verodostojno daju iste rezultate pri pozivu ugnježdenih procesa ili neke druge kompozicije.

Evidentno je dakle da model kompozicije pruža mogućnost da se korišćenjem BPDM-a na jedan uniforman način bilo koji sistem može kretati po relacijama između entiteta i kompozicionoj hijerarhiji kako u vreme „dizajna“ tako i izvršenja.

Model toka pruža BPDM-u mogućnost podrške širokog spektra modelovanja dinamike procesa. Zasnovan je na apstraktnom koordinacioni mehanizam koraka koji je proširiv na najveći broj oblika ponašanja sistema. Pruža podršku za osnovne elemente kao što su npr. koraci konvergencije ili divergencije u BPMN-u, kao i npr. koraci sekvence. Ono što je ovde važno napomenuti je da su ovo koraci u vreme dizajna, dok se u izvršnom vremenu oni pretvaraju u ograničenja.

BPDM pruža pouzdanu semantičku interoperabilnost, robusno izvršenje i efikasno praćenje i upravljanje procesom. Rezultat ovakve sofisticiranosti je mogućnost podrške širokom spektru poslovnih scenarija, od onih apstraktnih, korišćenih od strana višeg rukovodstva, pa do onih u izvršnom obliku.

5. UPRAVLJANJE RADNIM TOKOM

Upravljanje radnim tokom (Work Flow Management) je tehnologija u ubrzom razvoju koja se koristi od strane različitih industrija. Njena primarna karakteristika je automatizacija procesa koja obuhvata kombinaciju aktivnosti ljudi i mašina, posebno onih koje podrazumevaju interakciju sa IT aplikacijama i alatima. Iako se primena ove tehnologije prvenstveno odnosi na kancelarijske poslove kao što su osiguranje, bankarstvo, pravna i opšta administracija takođe se može primeniti i na neke vrste industrijskih i proizvodnih aplikacija. Postojanje širokog obima proizvoda na tržištu koji su zasnovani na ovoj tehnologiji daje mogućnost proizvođačima softvera da se usmere na posebne funkcionalnosti i specifične potrebe aplikacije. Međutim, još uvek ne postoje definisani standardi koji bi omogućili da ovi proizvodi funkcionišu zajedno, što za rezultat ima takozvana nekompatibilna „ostrva“ izolovanih automatizovanih procesa.

WFM koalicija je grupa kompanija koje su se udružile sa ciljem da zajedno daju predlog rešenja ovog problema. Uočeno je da svi proizvodi zasnovani na ovoj tehnologiji imaju opšte karakteristike što omogućava da se dostigne nivo interoperabilnosti korišćenjem opštih standarda za različite funkcije. WFM koalicija je pokušala da identifikuje ove funkcionalne oblasti i da razvije specifikaciju za implementaciju aplikacija. Namera je da ova specifikacija omogući interoperabilnost između heterogenih aplikacija radnog toka i poboljša njihovu integraciju sa drugim IT servisima kao što su elektronska pošta i upravljanje dokumentima. Time bi se poboljšale šanse za efektivno korišćenje tehnologije upravljanja radnog toka na IT tržištu u korist i proizvođača softvera i samih korisnika.

4.1. Šta je workflow?

Tok rada (workflow) se odnosi na automatizaciju procedura u okviru kojih se prosleđuju dokumenta, informacije ili zadaci između učesnika prema definisanom setu pravila, da bi se doprinelo ili postigao opšti poslovni cilj.

Dok se radni tok može i manuelno organizovati, u praksi većina radnih tokova je organizovana u kontekstu IT sistema koji omogućava kompjuterizovanu podršku za proceduralnu automatizaciju.

Radni tok je kompjuterizovana realizacija ili automatizacija poslovnog procesa u celini ili delovima.

Radni tok se često vezuje za reinženjering poslovnih procesa (BPR) koji se odnosi na analizu, modeliranje, definiciju i redosled implementacije operacija srži poslovnih procesa organizacije ili drugog poslovnog entiteta. Iako sve aktivnosti reinženjeringa poslovnih procesa ne rezultuju implementacijom, tehnologija radnog toka je često odgovarajuće rešenje pošto razdvaja logiku poslovne procedure od njene IT

operativne podrške, omogućavajući da promene po odgovarajućem redosledu budu uključene u proceduralna pravila koja definišu poslovni proces. Sa druge strane, nije neophodno da sve implementacije radnog toka budu deo reinženjeringa poslovnog procesa.

4.2. Sistem za upravljanje radnim tokom

Sistem za upravljanje radnim tokom je deo koji omogućava proceduralnu automatizaciju poslovnog procesa upravljanjem redosledom radnih aktivnosti i pozivanjem odgovarajućih ljudskih i/ili IT resursa koji su u vezi sa različitim koracima u okviru aktivnosti.

Sistem za upravljanje radnim tokom je sistem koji u potpunosti definiše, upravlja i izvršava radne tokove kroz izvršavanje softvera čiji je redosled izvršavanja vođen kompjuterskom reprezentacijom logike radnog toka.

Pojedinačni poslovni proces može imati životni ciklus od nekoliko minuta do nekoliko dana ili čak meseci u zavisnosti od kompleksnosti i trajanja različitih sastavnih aktivnosti.

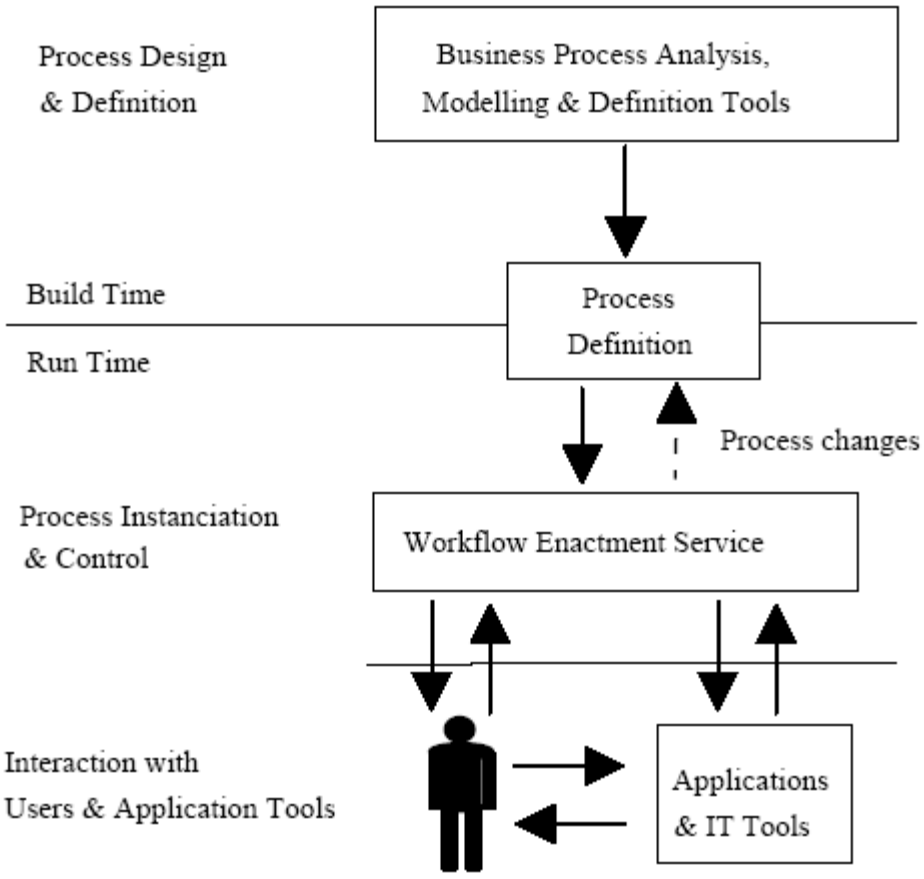
Ovakvi sistemi mogu biti implementirani na razne načine, mogu koristiti razne IT i komunikacione infrastrukture kao i da se izvršavaju u malom lokalnoj radnoj grupi ali i između preduzeća. Model koji je dala WFC grupa tako obuhvata širok pogled na upravljanje radnim tokovima, što ima za cilj da omogući razne implementacione tehnike i operativna okruženja koji su karakteristični za ovu tehnologiju.

Uprkos ovoj raznovrsnosti svi sistemi za upravljanje radnim tokovima pokazuju opšte karakteristike što obezbeđuje osnovu za razvoj integracije i sposobnosti interoperabilnosti između različitih proizvoda. Predložen model opisuje opšti model za izgradnju sistema za upravljanje radnim tokovima i pokazuje kako se može dovesti u vezu sa raznim alternativnim pristupima implementaciji.

Na najvišem nivou svi sistemi za upravljanje radnim tokovima se mogu okarakterisati kao podrška tri funkcionalna područja:

- **Funkcije u vreme izgradnje (bildovanja)** koje se odnose na definisanje, a često i modelovanje procesa radnog toka i aktivnosti koje ga čine.
- **Funkcije kontrole u vreme izvršavanja** koje se odnose na upravljanje procesom radnog toka u operativnom okruženju i redosled različitih aktivnosti koje je potrebno obraditi kao deo svakog procesa.
- **Interakcije u vreme izvršavanja** između ljudi kao korisnika i IT aplikativnih alata za procesiranje različitih koraka aktivnosti.

Slika 10. prikazuje osnovne karakteristike sistema za upravljanje radnim tokom i veze između glavnih funkcija.



Slika 36. Karakteristike sistema radnog toka

4.2.1. Funkcije u vreme bildovanja

Funkcije u vreme izgradnje (bildovanja) su one funkcije koje za rezultat imaju kompjuterizovanu definiciju poslovnog procesa. Tokom ove faze poslovni proces se prevodi iz realnog sveta u formalnu definiciju koju je moguće kompjuterski obraditi korišćenjem jedne ili više tehnika analize, modeliranja i definisanja sistema. Rezultujuća definicija se često zove model procesa, templejt procesa, metapodaci procesa ili samo definicija procesa. U ovom radu se koristi termin definicija procesa.

Definicija procesa je kompjuterizovana predstava procesa koja obuhvata manuelnu definiciju i definiciju radnog toka.

Definicija procesa obično sadrži mnoštvo diskretnih koraka aktivnosti koje su u vezi sa kompjuterom i/ili ljudskim operacijama i pravila koje kontrolišu odvijanje procesa kroz različite korake aktivnosti.

Definicija procesa se može izraziti u tekstualnoj ili grafičkoj formi ili notacijom nekog formalnog jezika. Neki sistemi radnog toka dozvoljavaju dinamička odstupanja od definicije procesa iz operativnog okruženja u vreme izvršavanja, kao što je naznačeno povratnom strelicom na prošloj slici.

Članovi WFC koalicije smatraju da inicijalno kreiranje definicije procesa nije polje za standardizaciju. Upravo se to smatra velikim poljem razdvajanja između proizvoda na tržištu.

Međutim, rezultat bildovanja, definicija procesa je identifikovana kao potencijalno polje standardizacije da bi omogućila razmenu podataka definicije procesa između različitih alata za bildovanje i proizvoda za izvršavaje.

4.2.2. Funkcije kontrole procesa u vreme izvršavanja

U vreme izvršavanja definicija procesa se interpretira softverom koji je odgovoran za kreiranje i kontrolu operativnih instanci procesa, kao i raspoređivanje različitih koraka aktivnosti u okviru procesa i pozivanje odgovarajućih ljudskih i IT resursa. Ove funkcije kontrole procesa u vreme izvršavanja predstavljaju vezu između procesa izmodeliranog u definiciji procesa i procesa u realnom svetu, koja se odražava u interakciji između korisnika i IT aplikativnih alata. Centralna komponenta je softver za kontrolu upravljanja radnim tokom („engine“) koji je odgovoran za kreiranje i brisanje procesa, kontrolu raspoređivanja aktivnosti u okviru operativnog procesa i interakciju sa aplikativnim alatima i ljudskim resursima.

4.2.3. Interakcije aktivnosti u vreme izvršavanja

Pojedine aktivnosti u okviru procesa radnog toka se obično odnose na ljudske operacije često realizovane zajedno sa posebnim IT alata (npr. popunjavanje forme) ili na operacije za obradu informacija koje zahtevaju poseban aplikativni program koji radi nad definisanim informacijama (npr. ažuriranje baze). Interakcija sa softverom za kontrolu procesa je neophodna za transfer kontrole između aktivnosti, za određivanje operativnog statusa procesa, za pozivanje aplikativnih alata i prosleđivanje odgovarajućih podataka. Postoji nekoliko koristi koje donosi standardizovani frejmvork za podršku ovog tipa interakcije, uključujući upotrebu konzistentnog interfejsa u brojnim sistemima radnog toka i sposobnost za razvoj opštih aplikativnih alata za rad sa različitim proizvodima.

4.3. Model za implementaciju

Uprkos raznovrsnosti proizvoda na tržištu pokazalo se mogućim konstrukcija opšteg modela implementacije sistema radnog toka. Glavne funkcionalne komponente opšteg modela prikazane su na narednoj slici.

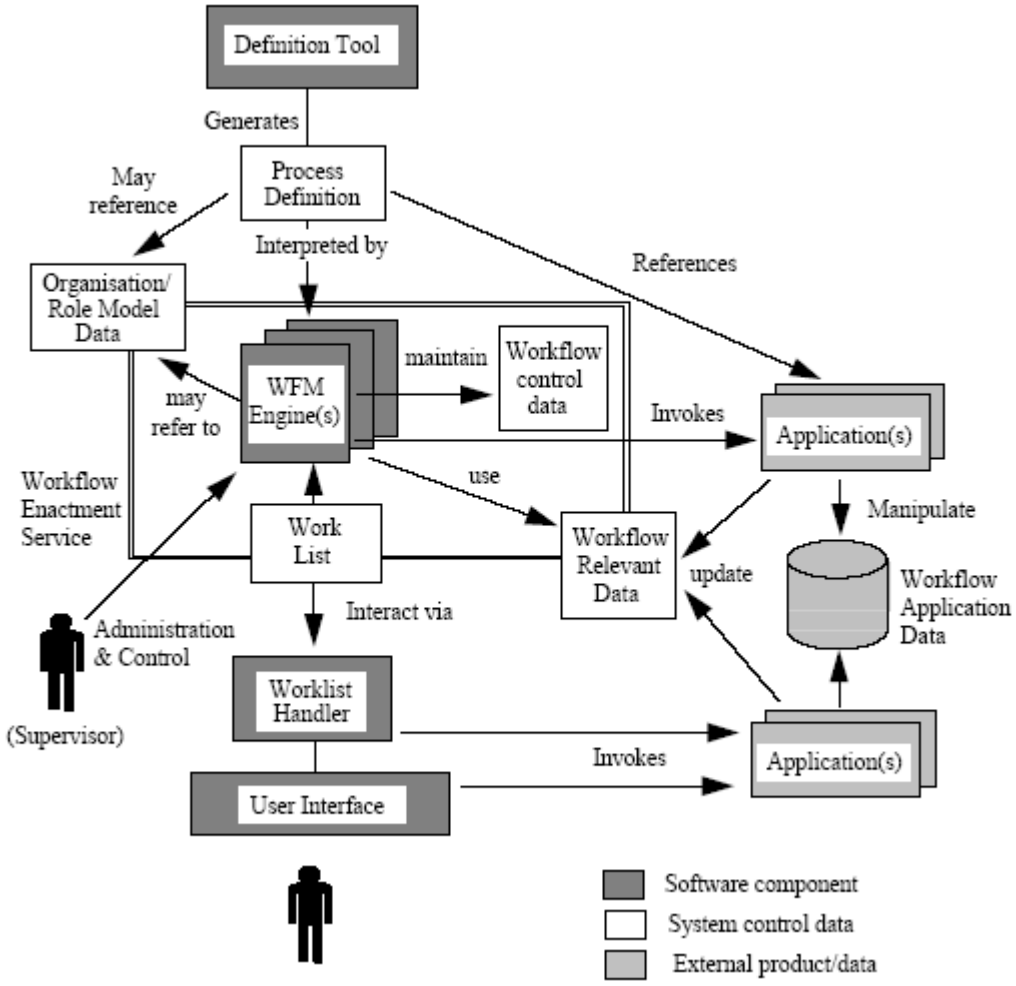
Postoje tri tipa komponenti:

- Softverske komponente koje pružaju podršku za razne funkcije u sistemu
- Različiti tipovi definicija sistema i kontrolni podaci koji se koriste od strane jedne ili više softverskih komponenti
- Aplikacije i baze podataka koje nisu deo proizvoda radnog toka, ali se mogu koristiti kao deo celokupnog sistema radnog toka

4.3.1. Definicija procesa

Alat za definiciju procesa se koristi za kreiranje opisa procesa u formi koju kompjuter može da obradi. Može se zasnivati na formalnom jeziku za definiciju procesa, modelu objekti veze ili u jednostavnim sistemima, na skriptama ili skupu komandi za transfer informacija između učesnika.

Definicija procesa sadrži sve neophodne informacije o procesu koje omogućavaju da proces može da se izvrši od strane softvera. Ona uključuje informacije o početnim i krajnjim uslovima, sastavne aktivnosti i pravila za navigaciju između njih, korisničke zadatke koje potrebno izvršiti, reference na aplikacije koje se mogu pozvati, definiciju relevantnih podataka koji se mogu koristiti.



Slika 37. Struktura opšteg modela

Definicija procesa se može odnositi na model organizacija odnosno uloga koji sadrži informacije o organizacionoj strukturi i ulogama u okviru organizacije. Time se omogućava da definicija procesa bude specifična u zavisnosti od organizacionih entiteta i funkcija uloga povezanih sa pojedinim aktivnostima ili objektima podataka, pre nego specifičnim učesnicima. Servisi za realizaciju radnog toka imaju odgovornost da povežu organizacione entitete ili uloge sa specifičnim učesnicima u toku izvršavanja radnog toka.

4.3.2. Servis za realizaciju radnog toka

Servis za realizaciju radnog toka interpretira opis procesa i kontroliše instanciranje procesa i redosled aktivnosti, dodajući radne stavke u radne liste korisnika i pozivajući aplikativne alate po potrebi. To se vrši preko jednog ili više kooperativnih endžina za upravljanje radnim tokom koji upravlja izvršavanjem pojedinačnih instanci različitih procesa. Servis za realizaciju radnog toka održava interne kontrolne podatke

bilo da su oni centralizovani ili distribuirani na više endžina; ovi kontrolni podaci uključuju informacije o internom stanju u vezi sa različitim instancama procesa i aktivnosti u toku izvršenja, a mogu takođe uključiti i kontrolne tačke i informacije za oporavak i ponovan početak koji koriste endžini da bi se oporavili od neuspelih provera uslova.

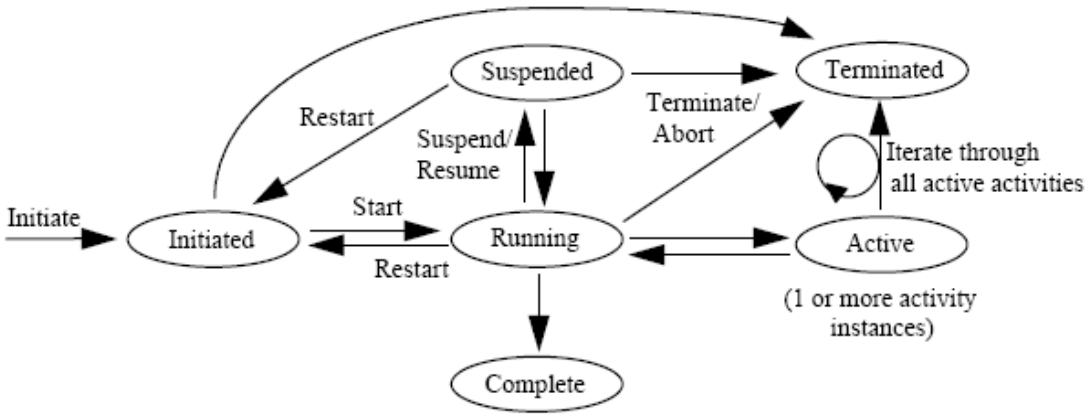
Definicija procesa zajedno sa relevantnim podacima za izvršenje se koriste za kontrolu navigacije kroz različite korake aktivnosti u okviru procesa, pružajući informacije o kriterijumu za ulazak odnosno izlazak iz pojedinog koraka aktivnosti, paralelnom ili sekvencijalnom izvršavanju različitih aktivnosti, zadataka korisnika ili IT aplikacija povezanih sa svakom od aktivnosti. To može zahtevati pristup modelu organizacija i uloga, ukoliko definicija procesa obuhvata koncepte u vezi sa ovim tipovima entiteta.

Endžin za radni tok takođe ima sposobnost aktiviranja aplikacionih alata koji su neophodni za izvršenje pojedinih aktivnosti. Pregled ovih mehanizama se u mnogome razlikuje; neki jednostavni sistemi podržavaju samo jedan alat kao što je npr. editor dokumenata, dok drugi mogu sadržati metode za pozivanje alata šireg opsega, i lokalnih i udaljenih u odnosu na endžin.

4.3.2.1 Prelaz stanja procesa i aktivnosti

Servis za izvršavanje toka rada može se posmatrati kao mašina prelaza stanja gde individualne instance procesa ili aktivnosti menjaju stanje kao odgovor na eksterne događaje ili na specifične kontrolne odluke od strane endžina toka rada (npr. navigacija ka sledećem koraku u aktivnosti u okviru procesa).

Na dijagramima koji slede prikazani su osnovni prelazi stanja instance procesa i aktivnosti.

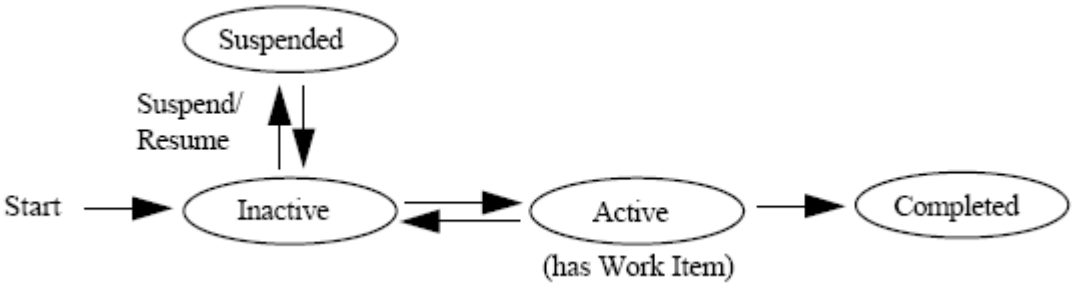


Slika 38. Primer prelaza stanja za instancu procesa

Osnovna stanja su:

Iniciran – instanca procesa je kreirana uključujući relevantne podatke, ali proces još uvek nije ispunio uslove za početak izvršavanja

Izvršava se – instanca procesa je započela izvršavanje i bilo koja od aktivnosti može biti startovana (kada se ispune odgovarajući uslovi za start aktivnost)
Aktivan – jedna ili više aktivnosti su startovane (npr. stavka rada je kreirana i dodeljena odgovarajućoj instanci aktivnosti)
Suspendovan – instanca procesa je u stanju mirovanja i nijedna aktivnost se ne startuje dok se proces ne vrati u stanje izvršavanja
Završen – instanca procesa je ispunila uslove za završetak i uništava se
Prekinut – izvršavanje instance procesa je zaustavljeno pre nego što je normalno završena



Slika 39. Primer prelaza stanja za instancu aktivnosti

Osnovna stanja instance aktivnosti su:
Neaktivna – aktivnost u okviru procesa je kreirana ali nije još pokrenuta (nisu ispunjeni uslovi) i ne postoje stavke rada za obradu
Aktivna – stavka rada je kreirana i dodeljena instanci aktivnosti na obradu
Suspendovana – instanca aktivnosti je u stanju mirovanja
Završena – izvršavanje instance aktivnosti je završeno

4.3.3. Workflow engine

Endžin toka rada (workflow engine) je odgovoran za deo ili celokupno okruženje za kontrolu u toku izvršavanja.

Endžin radnog toka je softverski servis ili „endžin“ koji obezbeđuje izvršno okruženje za instancu radnog toka.

Obično ovakav softver pruža mogućnosti za:

- Interpretaciju definicije procesa
- Kontrolu instance procesa – kreiranje, aktivaciju, suspenziju, završetak
- Navigaciju između aktivnosti procesa, koja se može odnositi na sekvencijalne ili paralelne operacije, interpretaciju relevantnih podataka
- Uključivanje i isključivanje specifičnih učesnika

- Održavanje kontrolnih podataka i relevantnih podataka, prosleđivanje relevantnih podataka od aplikacija do korisnika i obrnuto
- Interfejs za aktiviranje eksternih aplikacija i povezivanje relevantnih podataka

Endžin toka rada može da kontroliše izvršavanje skupa procesa ili potprocesa, kaoi instanci sa definisanim opsegom određenim sa opsegom tipova objekata i njihovih atributa, koje može da interpretira u okviru definicije procesa.

Kako se servis za izvršavanje toka rada sastoji od više endžina, i sam proces izvršavanja se deli između postojećih endžina. Ova podela se može vršiti prema tipu procesa, sa posebnim endžinom koji kontroliše pojedini tip procesa u celini, zatim funkcionalna podela sa posebnim endžinom koji kontroliše one delove procesa koji zahtevaju alokaciju korisnika ili resursa u okviru sopstvenog domena kontrole ili neki drugi kriterijum za podelu.

4.4. BPM i Workflow

Koje su razlike između ”BPM” i ”Workflow”?

Iako su često tretirani kao sinonimi, BPM i tok rada su, u suštini, dva različita i odvojena entiteta čije su razlike više nego akademske.

Tok rada se bavi utvrđivanjem redosleda aktivnosti za specifične aplikacije preko unapred definisanih setova instrukcija, koji uključuju jednu ili obe automatizovane procedure (softverski orijentisane) i ručne aktivnosti (ljudski rad).

BPM se bavi definicijom, izvršavanjem i upravljanjem poslovnim procesima definisanim nezavisno od bilo koje pojedinačne aplikacije. BPM je set radnih tokova, još izdvojen i mogućnošću da koordinira aktivnostima preko više aplikacija.

Integracija između sistema sa radnim tokom i eksternih sistema je ograničena, često samo dozvoljavajući povlačenje dokumenata ili podataka, samo kao transfer bez stvarne svesti o samom sadržaju.

BPM sistemi dozvoljavaju oboje: i čuvanje i introspekciju eksternih dokumenta i podataka, prezentujući zatvorenu petlju procesa za validaciju integriteta transakcija, podataka i sadržaja, kao i inicijativu kompenzujućih aktivnosti ako je to potrebno. BPM procesi odvajaju instrukcije za izvršavnja od toka procesa; tako da, rutiranje može biti povezano sa rezultatima procesa i ključnim tačkama procesa. Kako su procesi radnog toka povezani sa konkretnom aplikacijom, tok procesa obezbeđuje alternativne načine za postizanje istog cilja.

Za tok rada je bitno ponavljanje a za BPM je to koordinacija (takođe i automatizacija i orkestracija, respektivno).

BPM sadrži tok posla (logički tok rada), pored ostalih karakteristika. Tok rada podrazumeva specifično tržište koje je nastalo sredinom 90-tih koje je takođe sadržalo tok posla; međutim uglavnom su praćene elektronske forme dok su one propagirale kroz organizaciju. Proizvodi toka rada te ere su imali slab tok funkcionalnosti dizajna, lošu skalabilnost i bili su izolovani. Uprkos ovim ograničenjima, ova era je bila uspešna. Ovo je bilo fizičko tržište radnog toka, i ne treba ga mešati sa potrebom za tokom posla.

BPM je radni tok koji ima sofisticirani tok dizajna kroz modelovanje i analizu procesa. BPM podržava veliku kolicinu posla i brojne korisnike sa sofisticiranim mašinama stanja za dugoročne poslovne događaje i transakcije. On je lak za korišćenje, otvoren, ima pametne opcije podržane od strane mašina pravila i koristi pogodnosti integracione tehnologije. Konačno, BPM se povezuje sa orkestriranim web servisima, meri poslovnu aktivnost i optimizira procese za bolji rezultat i obim rada. Kao što se može videti, BPM je mnogo drugačiji od toka rada, ali ipak pomaže toku posla kroz poslove na globalnom i lokalnom nivou.

6. WEB SERVISI

6.1. Definicija Web servisa

Web servisi predstavljaju najnoviju distribuiranu tehnologiju. Web servisi se postali tehnologija koja se uobičajeno koristi za interoperabilnost i integraciju aplikacija i informacionih sistema. Web servisi obezbeđuju tehnološku osnovu za postizanje interoperabilnosti između aplikacija koje koriste različite softverske platforme, operativne sisteme i programske jezike. Izgrađeni su na XML jeziku. Dok je XML u osnovi standard za integraciju na nivou podataka, Web servisi su postali standard za integraciju na nivou servisa.

Web servisi su kao distribuirana tehnologija podržani od strane svih glavnih proizvođača softvera. Oni su prva tehnologija koja ispunjava obećanje o univerzalnoj interoperabilnosti između aplikacija koje rade na različitim platformama. Osnovne specifikacije na kojima su zasnovani Web servisi su SOAP, WSDL i UDDI. SOAP, WSDL i UDDI su zasnovani na XML jeziku, što čini da poruke koje stižu do Web servisa, odgovori koji se vraćaju, kao i opis Web servisa, budu ljudski čitljive.

Web servis predstavlja softverski sistem dizajniran da podrži interoperabilnu mašina-mašina interakciju preko mreže. On poseduje interfejs opisan u mašinski čitljivom formatu (WSDL). Drugi sistemi komuniciraju sa Web servisom korišćenjem SOAP poruka, koje se obično prenose preko HTTP, XML serializovanim u saradnji sa drugim Web standardima. [Web Services Arch]

Jednostavna definicija bi glasila: Web servis je aplikacija koja obezbeđuje Web API. Za API se može reći da podržava komunikaciju aplikacija-aplikacija. Web API predstavlja API koji dozvoljava aplikacijama da komuniciraju koristeći XML i Web. [Manes]

Web servisi pojednostavljuju proces komunikacije između aplikacija. Pojednostavljenje utiče na smanjenje troškova i vremena razvoja, lako održavanje, kao i smanjenje ukupnih troškova.

Dva sistema koriste Web servise da komuniciraju međusobno kada aplikacija kreira XML dokument u obliku poruke i pošalje ga preko mreže do Web servisa. Opciono, Web servis šalje odgovor na zahtev u formi XML dokumenta. Web servis standardi definišu interfejs poruke, format, mapiranje sadržaja poruke u implementaciju servisa, opciona zaglavlja, kao i način za publikovanje servisa i otkrivanje drugih Web servisa.

Neke od ključnih osobina Web servisa su [IBM SOA]:

- **Web servisi su samostalni.**
  Na klijentskoj strani nije potreban dodatni softver. Dovoljan je samo programski jezik sa XML i HTTP podrškom. Na serverskoj strani, potreban je Web server.
- **Web servisi su samoopisujući.**
  Definicija formata poruke se nalazi u samoj poruci. Nisu potrebna nikakvi dodatna skladišta za meta podatke, niti alati za njihovo kreiranje.
- **Web servisi su modularni.**
  Web servisi predstavljaju tehnologiju za isporučivanje i obezbeđivanje pristupa poslovnim funkcijama preko Web-a; J2EE, CORBA, kao i drugi standarda koji predstavljaju tehnologije za primenu Web servisa.
- **Web servisi mogu biti publikovani, locirani i pokrenuti preko Weba.** Za to su neophodni sledeći standardi:
  - Simple Object Access Protocol (SOAP)
  - Web Service Description Language (WSDL)
  - Universal Description, Discovery, and Integration (UDDI)
- **Web servisi su nezavisni od jezika i interoperabilni.**
  Interakcija Web servisa i njegovog klijenta je dizajnirana tako da bude nezavisna od platforme i jezika. Njihova interakcija zahteva WSDL dokument za definiciju interfejsa i opis servisa, zajedno sa mrežnim protokolom. Pošto ni Web servis, ni njegov klijent ne znaju međusobno na kojim se platformama nalaze i pomoću kojeg jezika su kreirani, interoperabilnost je obezbeđena.
- **Web servisi su bazirani na otvorenim standardima.**
  XML i HTTP predstavljaju tehničku osnovu Web servisa. Veliki deo Web servis tehnologije je kreiran koristeći Open Source projekte.
- **Web servisi su dinamički.**
  Opis i pronalaženje Web servisa se mogu automatizovati preko UDDI i WSDL.

Web servise možemo shvatiti kao udaljene procedure koje su „izložene“ od strane serverskih aplikacija. Oni predstavljaju nov način za izvršavanje poziva udaljenih metoda preko HTTP protokola, pri čemu mogu koristiti protokol SOAP (Simple Object Access Protokol). Ovaj protokol značajno pojednostavljuje stvari u odnosu na DCOM (Distributed COM). On predstavlja standard koji je zasnovan na jeziku XML i koji sadrži detalje o tome kako se mogu izvršiti pozivi metoda putem HTTP-a na predvidiv način. Web servisi se opisuju pomoću posebnog jezika koji je nazvan WSDL (Web Service Description Language), koji je potpuno zasnovan na XML sintaksi, i koji omogućuje dinamičko otkrivanje web servisa u vreme izvršavanja. Jezik WDSL uz pomoć XML šema pruža opis svih metoda, kao i neophodnih tipova za njihovo pozivanje, koje su javno izložene u web servisu.

Ono što je dalje posebno značajno istaći kod web servisa je da oni:

- izlažu različite funkcionalnosti web korisnicima preko standardnih web protokola, najčešće SOAP-a,
- obezbeđuju način za opis njihovog interfejsa u onoliko detalja koliko je potrebno korisnicima da izgrade klijentsku aplikaciju koja komunicira sa njima (ovo je XML WSDL dokument);
- su registrovani tako da ih potencijalni korisnici mogu lako pronaći. Ovo se postiže uz pomoć **UDDI** (***Universal Discovery Description and Integration***).

Pored distribuiranosti web servisi omogućavaju i prevazilaženje nekih od oblika heterogenosti, što je jedan veliki korak napred u odnosu na prethodne tehnologije. Neke od prednosti su:

standardizovan trasnportni mehanizam (standardizacija komunikacionog protokola: XML, Soap);

- nezavisnost od operativnog sistema na kojem radi web servis;
- nezavisnost od hardverske platforme;
- nezavisnost od jezika u kojem je web servis implementiran;
- heterogeni izvori podataka (baza podataka, sistem datoteka...)

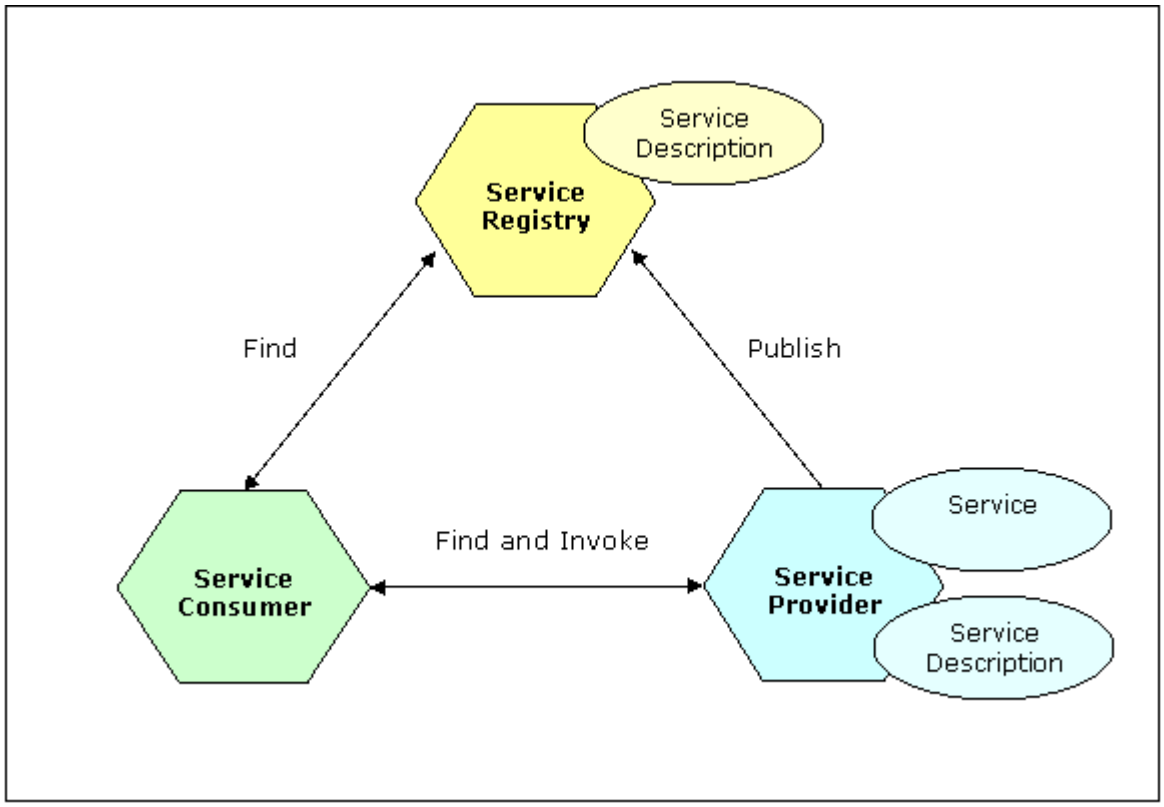
Primenom web servisa se evidentno odmah prevazilaze ove tehničke, odnosno sistemske razlike. Međutim, strukturna i semantička heterogenost i dalje ostaje značajan problem.

Pojava web servisa je stvorila uslove i za razvoje nove arhitekture orijentisane ka servisu (service oriented architecture). Ona predstavlja evoluciju distribuiranih sistema gde se poslovna logika aplikacija ili neke druge funkcionalnosti grupišu u module i prezentuju kao servisi za klijentske aplikacije.

6.2. Arhitektura Web servisa

Popularna interpretacija Web servisa se zasniva na IBM-ovoj arhitekturi, koja se sastoji od tri elementa:

- **Service Consumer**: Predstavlja aplikaciju, softverski modul ili drugi Web servis, koji inicira traženje Web servisa u registru, povezuje se sa Web servisom i izvršava Web servis operaciju.
- **Service Provider**: Predstavlja mrežno adresni entitet, koji prihvata i izvršava zahteve od Service Consumer. Takođe publikuje opis svojih servisa u Service Registry, tako da ga potencijalni korisnici mogu pronaći.
- **UDDI (Service Registry)**: Obezbeđuje otkrivanje i pronalaženje Web servisa. Sadrži skladište dostupnih Web servisa i dozvoljava pretraživanje informacija koje sadrži od strane zainteresovanih potencijalnih korisnika.



Slika 40. Arhitektura Web servisa

Operacije kod arhitekture Web servisa:

- **Publish**: Da bi bio dostupan, Web servis mora biti objavljen, tako da ga zainteresovani korisnici mogu otkriti i pozvati na izvršavanje.
- **Find**: Zainteresovani potencijalni korisnik pronalazi Web servis ispitivanjem registra servisa po nekom kriterijumu.
- **Bind and Invoke**: Posle pronalaženja opisa Web servisa, korisnik se povezuje sa Web servisom i poziva ga na izvršavanje u skladu sa informacijama koje se nalaze u opisu Web servisa.

Artifakti kod arhitekture Web servisa:

- **Service**: Servis stavljen na raspolaganje za korišćenje preko objavljenog interfejsa.
- **Service Description:** Predstavlja opis Web servisa, koji specificira način interakcije sa Servis provajderom. Opisuje se format zahteva i odgovor od strane servisa.

6.2. Kada treba koristiti Web servise?

Web servisi su korisni i prilagodljivi, ali svakako imaju svoja ograničenja. Preporuka je da se koriste u sledećim situacijama:

- Web servisi su odlični za povezivanje različitih sistema. Na primer, Web servisi se mogu koristiti da povežu više platformi, kao što su Windows, Macintosh, Linux, UNIX, AS/400, kao i mainframe sistemi. Web servisi se takođe mogu koristiti za povezivanje različitih programskih jezika, kao što su Visual Basic, Java, C++, RPG, COBOL, Perl.
- Web servisi se dobro ponašaju u nekontrolisanim uslovima. Na primer, ako se komunicira sa različitim korisnicima, i naročito ako ne postoji kontrola nad korisničkim sistemima, Web servisi se mogu koristiti kao fleksibilno i prilagodljivo rešenje.
- Web servisi su nenadmašni u podršci višekanalnim klijentima. Na primer, Web servis se može koristiti za isporučivanje istih podataka bogatim klijentima, Web čitačima, mobilnim uređajima.
- Web servisi se dobro ponašaju u dinamičkim uslovima. Ukoliko sistem treba da se prilagodi promenljivim uslovima, da odgovori na potrebu da se doda ili promeni deo sistema, Web servisi predstavljaju odličan izbor.
- Web servisi su dobri u prikupljanju informacija. Web servisi se mogu koristiti da povuku informacije zajedno sa različitih izvora za portal, aplikacije vezane za kupce, i druge aplikacije za različite vrste integracije.
- Web servisi smanjuju ponavljanje podataka. Ako postoji više aplikacija koje u osnovi obavljaju isti posao, Web servisi se mogu koristiti za centralizaciju te funkcionalnosti u jedan skup deljivih servisa.

6.3. Kada ne treba koristiti Web servise?

Bez obzira što su Web servisi korisni i prilagodljivi, ne treba da se koriste kao rešenje za sve situacije.

- XML je veoma prilagodljiv, ali ne predstavlja najkompaktniji i najefikasniji mehanizam za transfer podataka. SOAP poruke su mnogo veće u odnosu na binarne poruke kreirane od strane RPC, RMI, COBRA ili DCOM. Takođe, obrada XML poruke traje duže u odnosu na obradu binarne poruke. Razlika u performansama postaje uočljivija sa povećanjem veličine i složenosti poruke.

- SOAP ne treba da bude potpuna zamena za tradicionalne tehnologije, kao što su RPC, RMI, COBRA ili DCOM. Ove tehnologije još uvek imaju veoma važnu ulogu u razvoju aplikacija. One su dizajnirane da obezbede mehanizam sa veoma visokim performansama za komunikaciju sa različitim komponentama u okviru homogenog aplikacionog sistema ili servisa.

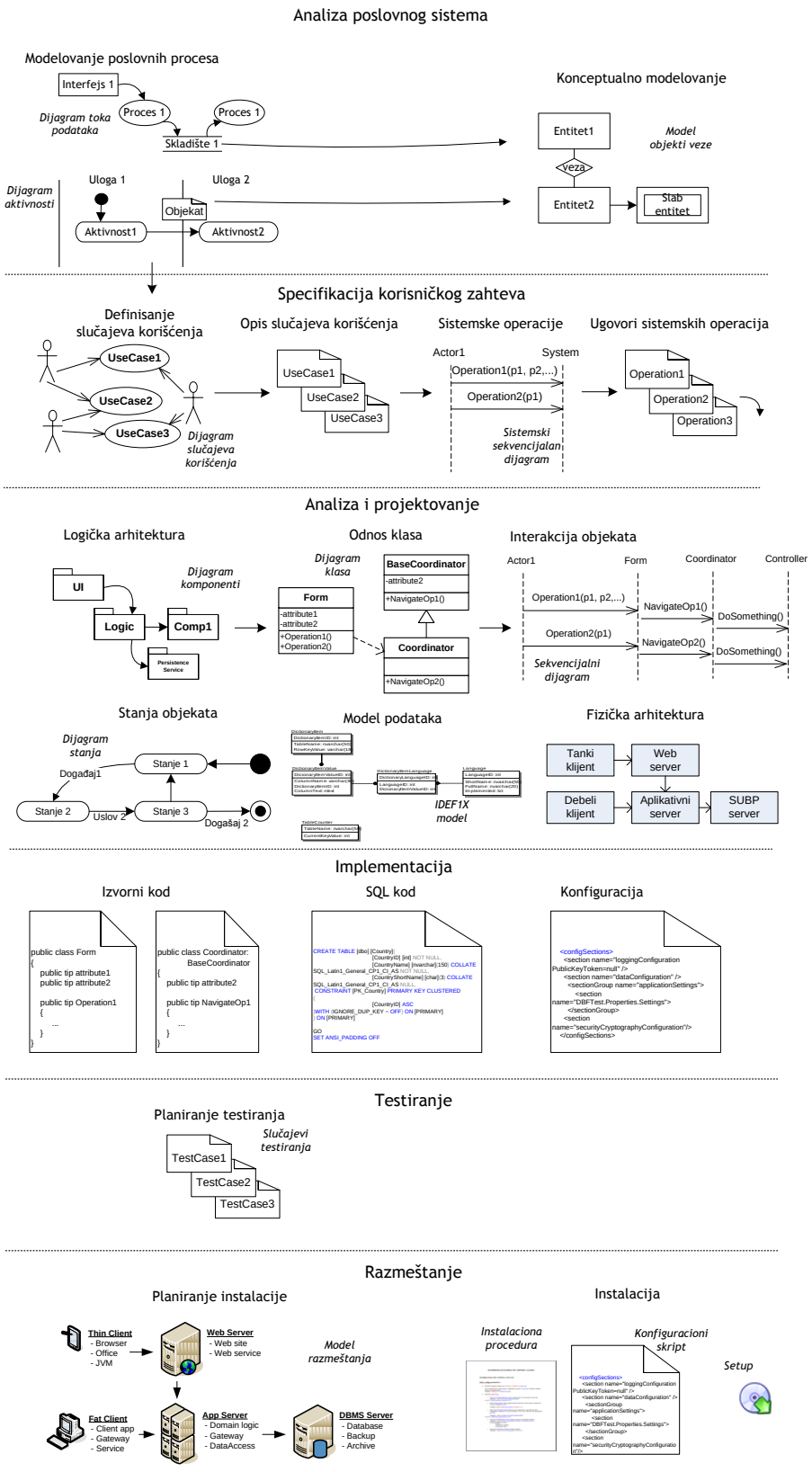
- Web servis ne obezbeđuje razvojni komponentni model, tako da se aplikacije ne mogu praviti korišćenjem Web servisa. Aplikacije se prave korišćenjem komponent tehnologije zasnovane na određenom programskom jeziku, kao što je Java ili Visual Basic.

7. PREDLOŽENO REŠENJE PROBLEMA

7.1. Metodologija

Primenjena metodologija predstavlja detaljnije razrađen Jedinstveni proces u smeru ka RUP-u (Rational Unified Process) sa jasno definisanim fazama životnog ciklusa softvera podrazumevajući pritom aktivnosti svake od faza kao i modele kao rezultat

Proces razvoja koji se ovde razmatra je proces u užem smislu koji pretpostavlja da su ulazni podaci prikupljeni i na određeni način dokumentovani, tako da svaka od faza ovog procesa se oslanja na dokumentovane prikupljene podatke. Ovo se prevashodno odnosi na početne faze projekta u kojima (Specifikaciju korisničkog zahteva i Analizu i Projektovanje).



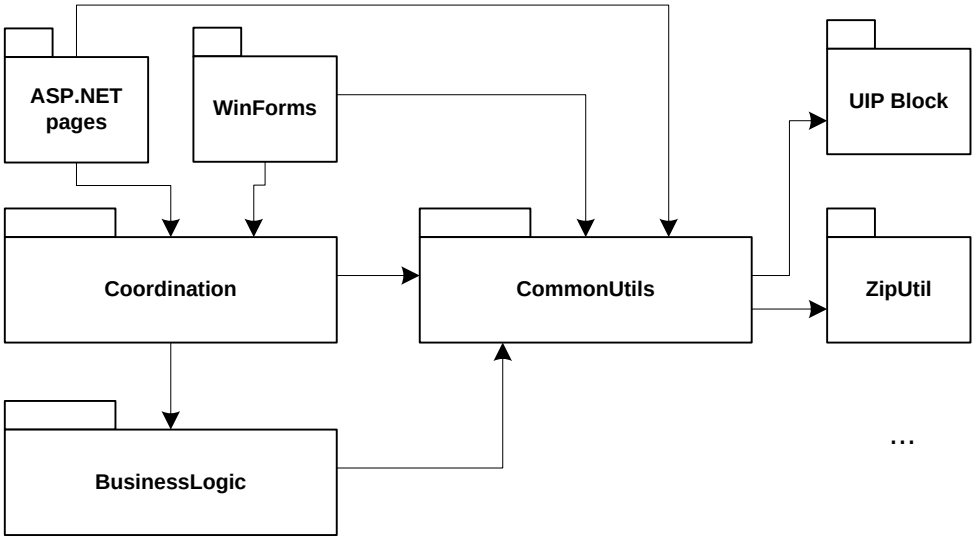
Slika 41. Primenjena metodologija

7.2. Arhitektura

Arhitektura softverskih rešenja je razvijena troslojna, u smislu da sadrži slojeve (elemente) odgovorne za:

- Pristup podacima
- Poslovnu logiku
- Koordinaciju (navigaciju)
- Korisnički i aplikativni interfejs

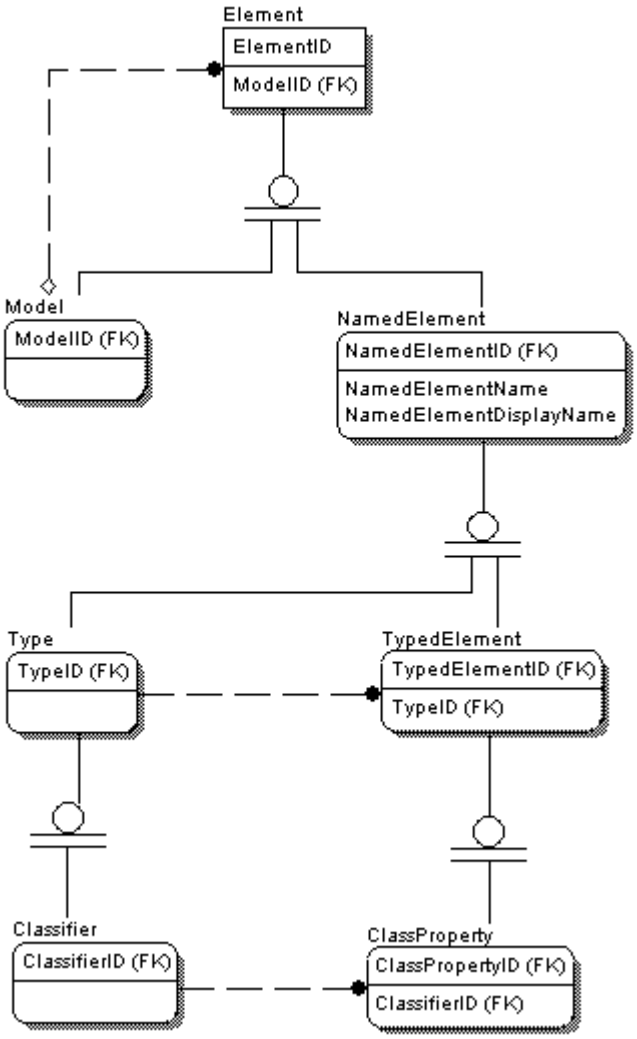
Navedeni elementi arhitekture se koriste po potrebi, tj. u zavisnosti od toga da li postoji potreba za element sa tom ulogom. Npr. ako aplikacija nema interfejs slojevi korisničkog interfejsa i koordinacije se neće koristiti.



Slika 42. Arhitektura sistema

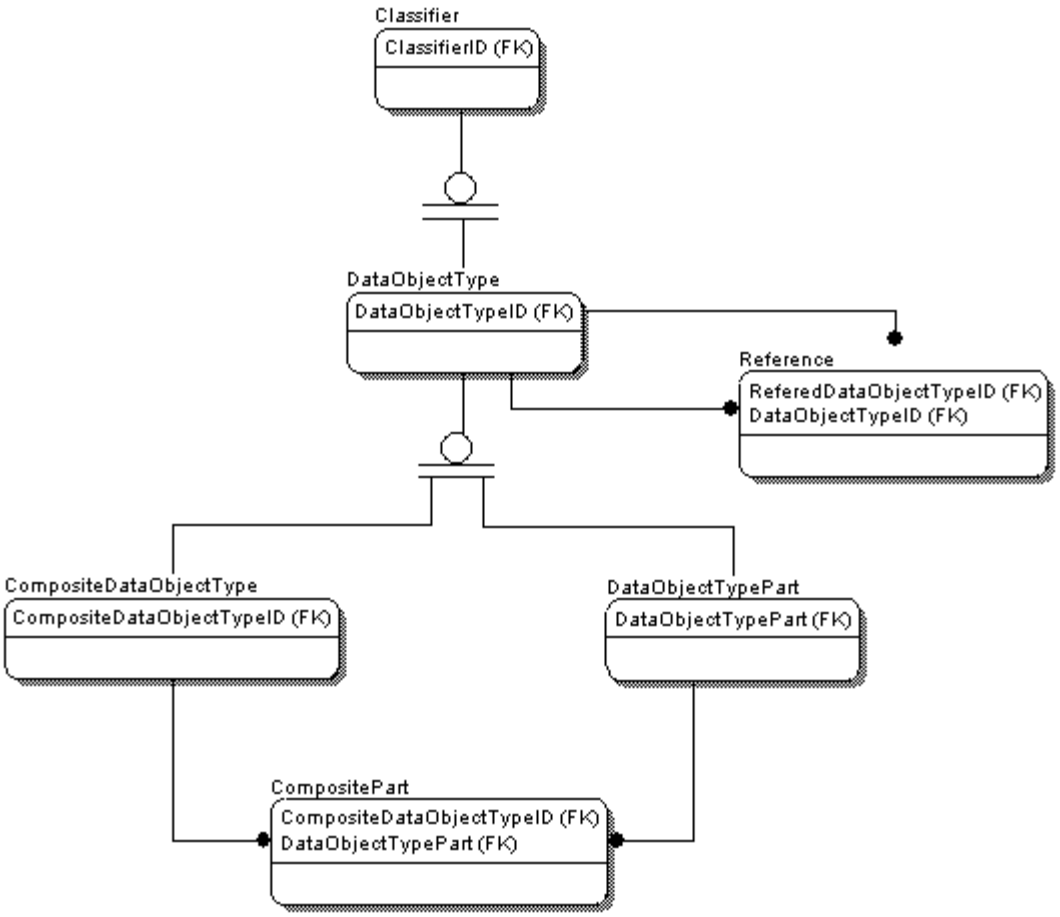
7.3. Model

7.3.1. Osnovni elementi – Core::Basic elements



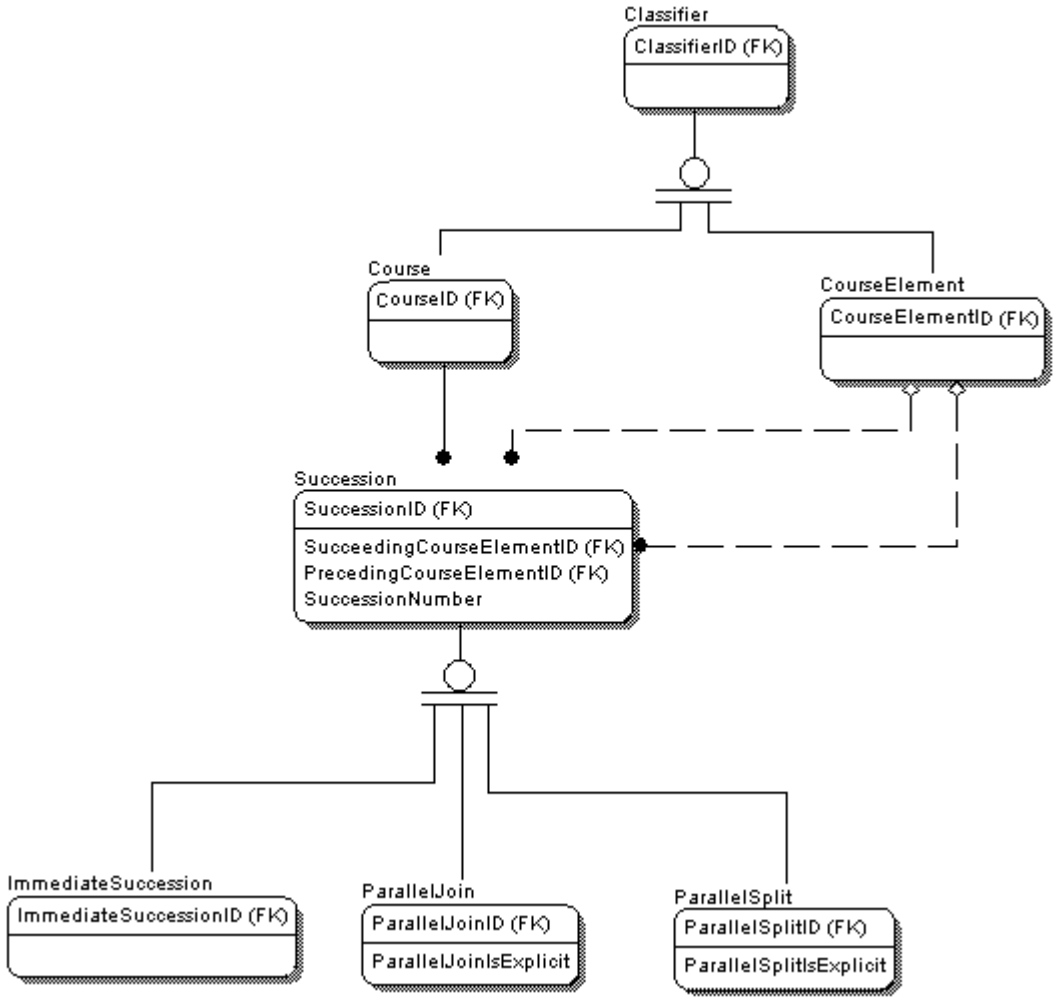
Ovim podmodelom su predstavljeni osnovni elementi koji predstavljaju srž modela odnosno osnovu iz koje se dalje sve izvodi. Mogu biti element sa imenom ili model koji opet može sadržati element odnosno imenovani element ili drugi model. Dalje imenovani element može postati tip ili tipizirani element koji je nekog tipa. Tip koji sadrži svojstva postaje klasifikator.

7.3.2. Kompozicija – Core::Composition



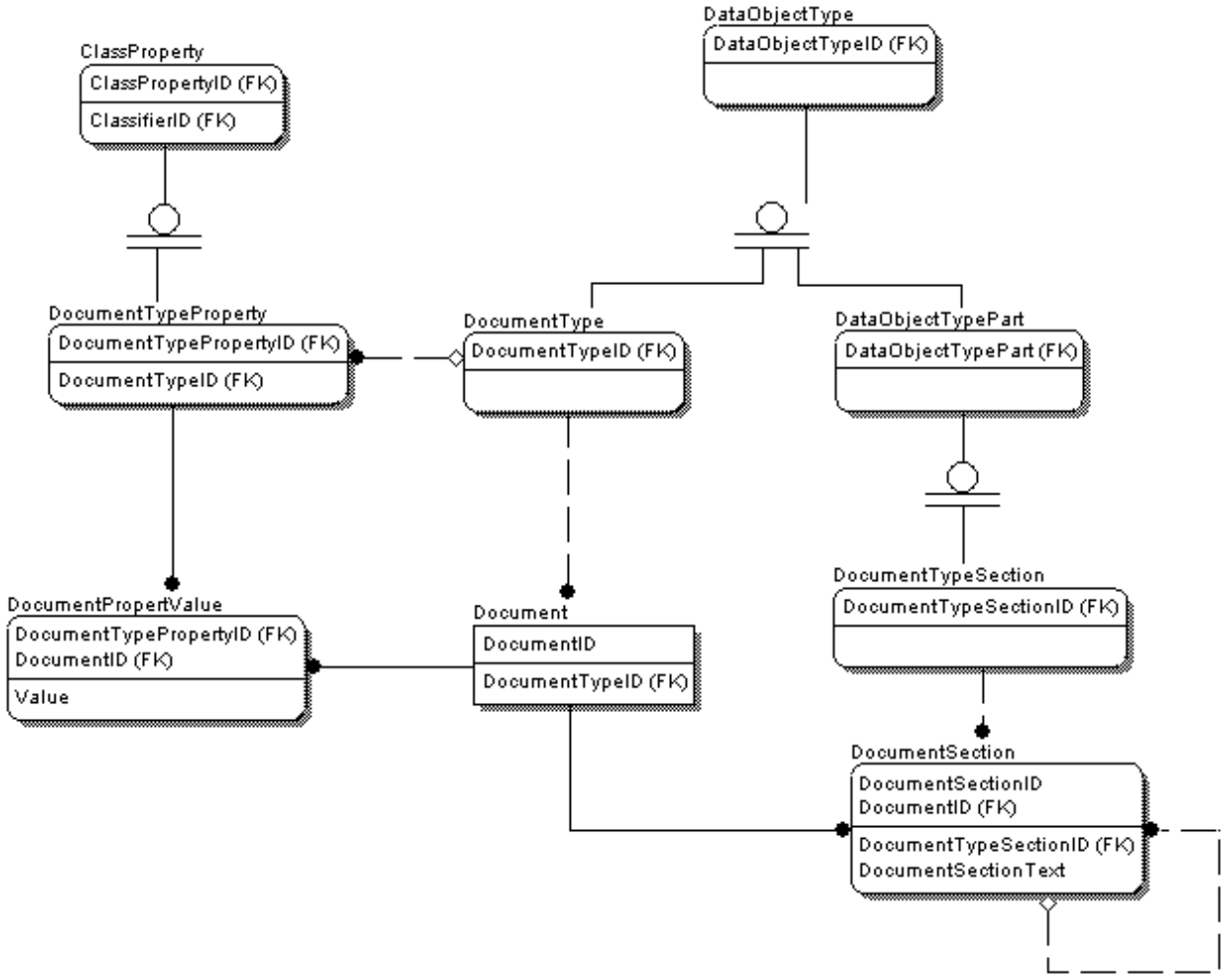
Podmodel Kompozicije pokazuje kako se elementi modela između sebe mogu komponovati odnosno kakve se strukture mogu predstavliti. Klasifikator može postati neki tip objekta koji se odnosi na podatke, a koji može nastati na osnovu nekih drugih podataka ili na osnovu tih podataka nastaje novi tip objekta. Na to ukazuje entitet Referenca. Objekat tipa podataka može biti kompozitna struktura ili samo deo koji sam može biti nezavistan, ali i deo nekog drugog kompozitnog objekta.

7.3.3.Tok – Core::Course



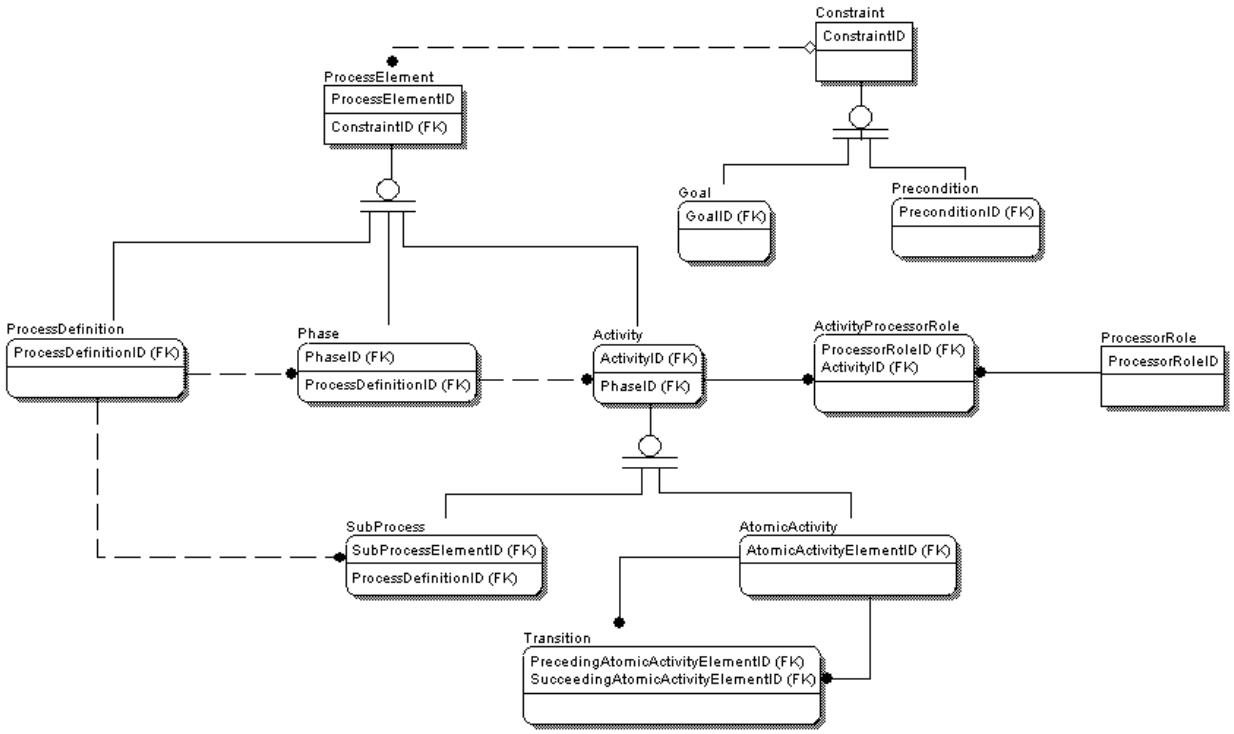
Podmodel Toka ukazuje na redosled elemenata koji su deo toka. Klasifikator može biti tok ili element toka. Entitet Redosled pripada nekom toku i čuva podatke o prethodnim odnosno narednim elementima toka. Sam prelazak toka sa jednog elementa toka na drugi se može izvršiti odmah bezuslovno, može se izvršavati paralelno po granama toka i može predstavljati tačku spajanja više grana toka.

7.3.4. Objekat podataka – Core::Data Object



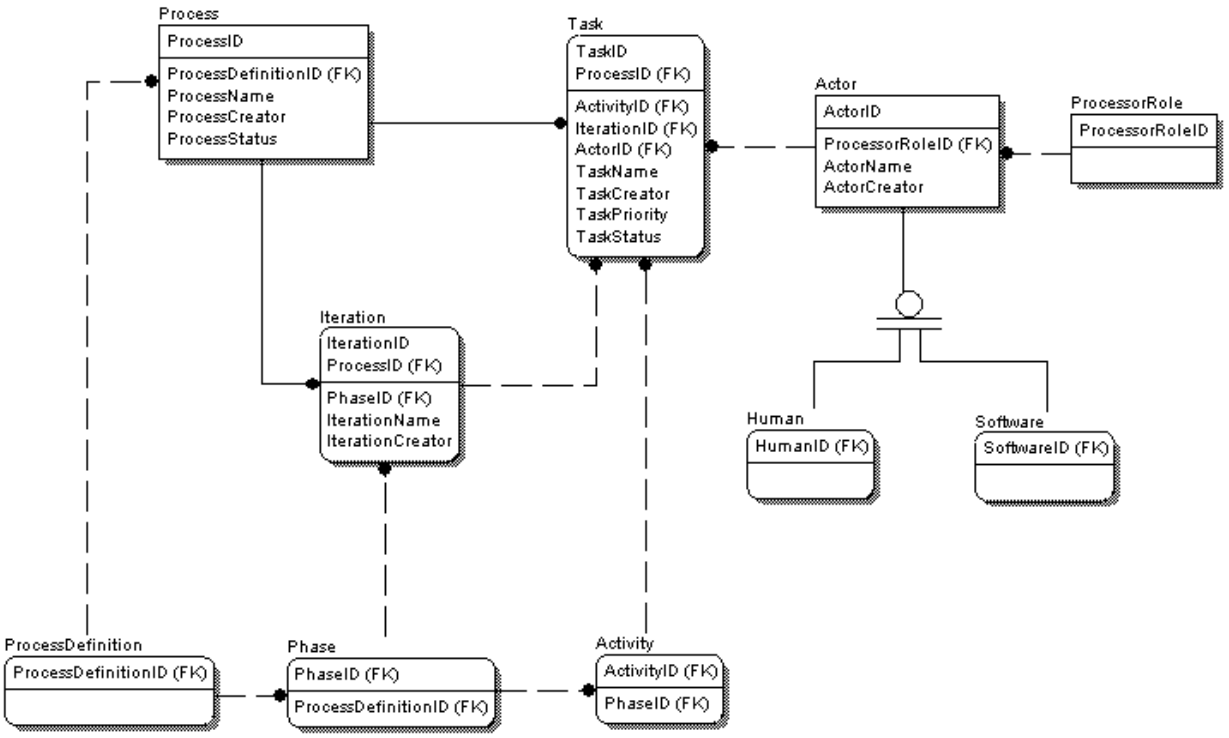
Tip objekta podataka može se posmatrati kao tip dokumenta. Sama instanca dokumenta je nekog tipa dokumenta. Tip objekta podataka koji predstavlja deo može postati tip sekcije u okviru dokumenta, samim tim konkretna instanca sekcije dokumenta je nekog tipa sekcije dokumenta. Sekcija dokumenta može sadržati druge sekcije odnosno može se kreirati na osnovu druge sekcije. Na nivou tipa dokumenta moguće je definisati tipove svojstava koji su karakteristični za neki tip dokumenta, a čije se konkretne vrednosti upisuju u agregirani objekat koji nastaje sizmeđu instance dokumenta i tipa svojstva za taj dokument.

7.3.5. Model procesa – Process definition



U modelu procesa na najvišem nivou nalazi se element procesa koji može biti definicija procesa, faza ili neka aktivnost. Za element procesa mogu se definisati i ograničenja. Ograničenja definišu kriterijume za ulazak i izlazak iz nekog elementa procesa. Definicija procesa može imati više faza, a u okviru faze može postojati više aktivnosti. Same aktivnosti mogu biti neki podproces ili atomska aktivnost na nivou koje se definiše transformaciija. U okviru modela postoje uloge koje su zadužene za obavljanje odgovarajućih aktivnosti.

7.3.6. Model izvršavanja procesa – Process instance



Model instance procesa predstavlja model izvršavanja definicije procesa. Instanca procesa ima informaciju o konkretnoj definiciji procesa, kao što iteracija i zadatak znaju kojoj instanci procesa pripadaju. Iteracija se odnosi i na fazu u definiciji procesa, a zadatak na aktivnost. Zadatak se izvršava od strane aktora koji obavlja predefinisanu rolu. Sam aktor, osim ljudi može biti i neki softver.

8. ZAKLJUČAK

Za potrebe Sistema za automatizaciju i praćenje razvojnih projekata, a u skladu sa postojećim frejmvorkom preduzeća „Soprex“, napravljen je model koji osim toga što podržava različite procese razvoja softvera, podržava i druge tipove procesa pa samim tim predstavlja metamodel procesa.

Model koji je predložen u okviru ovog rada, napravljan je po ugledu na Metamodel definicije poslovnih procesa (BPDM) i Metamodel procesa razvoja softvera (SPEM) razvijenih od strane OMG-a.

Metamodel procesa razvoja softvera se koristi za opis konkretnog procesa razvoja ili familije povezanih procesa razvoja softvera. U okviru ove specifikacije postoji ograničenje u pogledu definisanja minimalnog skupa elemenata modeliranja procesa koji su neophodni da se opiše bilo koji proces razvoja softvera.

Sa druge strane Metamodel definicije poslovnih procesa integriše više različitih pristupa i notacija procesa. U svojoj osnovi, BPDM omogućava interoperabilnost između alata od kojih svaki predstavlja proces na sopstven način.

Rešenje prezentovano u ovom diplomskom radu može da zadovolji potrebe velikog broja poslovnih procesa, bez ograničavanja na procese navedenog preduzeća, tako da se model može koristiti nezavisno od frejmworka za razvoj softvera koji su specifični za pojedina preduzeća.

Naravno, prostora za izmene prezentovanog modela ima, a trenutno se radi i na implementaciji rešenja kao svojevrsna validacija vrednosti samog modela.

Ovaj diplomski rad ukazuje na mogućnost standardizacije opisa poslovnih procesa, što približava stvaranju kompatibilnih složenih softverskih komponenti, čime bi se znatno smanjili troškovi softverskih sistema, a njihove mogućnosti znatno uvećale.

9. INDEX SLIKA

10. LITERATURA

1. Booch, G., J. Rambaugh, I. Jacobson: *The Unified Software Development Process*, Addison -Weseley, 1999
2. Beate List, Birgit Korherr: An Evaluation of Conceptual Business Process Modelling Languages, 2006
3. OMG Object Management Group: Software Process Engineering Metamodel Specification, Version 1.1, 2005
4. OMG Object Management Group: Business Process Modeling Notation Specification, 2006
5. Stephen A. White: Introduction to BPMN, IBM Corporation, 2006
6. Stephen A. White: BPMN tutorial, IBM Corporation, 2006
7. David Hollingsworth: The Workflow Reference Model, Workflow Management Coalition, 1995
8. David Chappell, Chappell & Associates, Microsoft and BPM: A Technology Overview, 2007
9. Marija Bjeković i Ivan Bojičić: Pristup razvoju informacionog sistema automatskom transformacijom modela, Beograd, 2006
10. *Microsoft MSDN Online Library*, Microsoft Corporation, http://msdn.microsoft.com/library/
11. http://www.omg.org
12. http://www.modeldriven.org/web/bpdm
13. http://www.bpmn.org/
14. http://www.omg.org/mda
15. http://en.wikipedia.org/wiki/Metamodeling

061/ 11-00-105

Seminarski, diplomski, maturski radovi, prevodi na engleski i eseji…