SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

SUSTAV ZA RAZMJENU I UPRAVLJANJE INFORMACIJAMA O PROIZVODU

MAGISTARSKI RAD

Mentor:

Dr. sc. Dorian MARJANOVIĆ, izv. prof.

Mario ŠTORGA, dipl.inž.

ZAGREB, 2002.

PODACI ZA BIBLIOGRAFSKU KARTICU:

UDK: 658.512.2

Ključne riječi: proces konstruiranja, upravljanje informacijama o proizvodu, strukturiranje proizvoda, modeliranje i prikaz informacija o proizvodu, STEP, XML, web servis

Znanstveno područje: TEHNIČKE ZNANOSTI

Znanstveno polje: STROJARSTVO

Institucija u kojoj je rad izrađen:
Fakultet strojarstva i brodogradnje, Sveučilište u Zagrebu

Mentor rada: Dr.sc. Dorian MARJANOVIĆ, izv. prof.

Broj stranica: 130

Broj slika: 65

Broj tablica: 2

Broj korištenih bibliografskih jedinica: 81

Datum obrane:

Povjerenstvo:
Dr.sc. Bojan JERBIĆ, izv. prof. – predsjednik povjerenstva

Dr.sc. Dorian MARJANOVIĆ, izv. prof. – voditelj

Dr.sc. Jože DUHOVNIK, red. prof. Fakulteta za strojništvo, Ljubljana – član povjerenstva

Institucija u kojoj je rad pohranjen:
Fakultet strojarstva i brodogradnje, Zagreb

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

Poslijediplomski znanstveni studij

Zagreb, 2002-02-06

Zadatak za magistarski rad

Kandidat: Mario Štorga, dipl. inž. strojarstva

Zadatak: **SUSTAV ZA RAZMJENU I UPRAVLJANJE INFORMACIJAMA O PROIZVODU**

Sadržaj:

Distribuirani razvoj proizvoda u uvjetima globalne suradnje implicira potrebu za razvojem novih računalnih sustava za podršku konstruktorima pri razmjeni i upravljaju informacijama o proizvodu. Obim i različitost informacija te fleksibilnost i otvorenost koja se zahtijeva u njihovoj razmjeni i upravljanju, otežavaju njihovu kontrolu i održavanje.

Definirati informacijsku infrastrukturu za razmjenu i upravljanje informacijama o proizvodu u procesu konstruiranja. Istražiti mogućnost uporabe dijelova ISO 10303-STEP standarda kao semantičke i strukturne osnove sustava. Kreirati računalnu platformu za upravljanje i korištenje informacijama tijekom razvoja proizvoda. U razvoju sustava uzeti u obzir mogućnosti web tehnologija i komunikacije između korisnika u mrežnom okruženju. Posebnu pažnju posvetiti XML tehnologiji u smislu definiranja specifikacije XML strukture za razmjenu i upravljanje informacijama o proizvodu u skladu sa semantičkom i strukturnom osnovom.

U radu je potrebno:

- istražiti teoretske modele prikaza i opisivanja informacija o proizvodu,
- analizirati razmjenu i upravljanje informacijama tijekom razvoja proizvoda,
- koncipirati semantičku i strukturnu osnovu sustava u skladu s teoretskim osnovama i međunarodnim ISO 10303 standardom,
- razraditi i predložiti metodologiju implementacije sustava u realnu radnu okolinu,
- koncipirati i realizirati arhitekturu sustava korištenjem web tehnologije za razmjenu informacija tijekom razvoja proizvoda.

Zadatak zadan:

Rad predan:

Mentor: Voditelj smjera:

Predsjednik Odbora

za poslijediplomske studije:

Dr.sc. Đorian Marjanović, izv. prof. Dr.sc. Dragutin Ščap, red. prof.

Dr.sc. Božo Vranješ, red. prof.

ZAHVALA

Zahvaljujem mentoru, prof. dr. sc. Dorianu Marjanoviću na savjetima, korisnim raspravama i potpori tijekom izrade ovog rada.

Djelatnicima Katedre za osnove konstruiranja zahvaljujem na pomoći i vremenu koje smo zajednički utrošili na brojne analize i rasprave koje su rezultirale većom kvalitetom prikazanog istraživanja.

Na kraju, posebno zahvaljujem svojoj obitelji na razumijevanju i potpori.

SADRŽAJ

POPIS SLIKA VII
POPIS TABLICA IX
PREDGOVOR X
SAŽETAK RADA XI
SUMMARY XII

1. UVODNA RAZMATRANJA 1-1

1.1 MOTIVACIJA ZA RAD 1-1
1.2 CILJEVI ISTRAŽIVANJA I PODRUČJE 1-3
1.3 PRISTUP – ISTRAŽIVAČKE METODE 1-4
1.4 STRUKTURA RADA 1-5

2. PREGLED STANJA I TEORETSKIH OSNOVA ISTRAŽIVANJA 2-1

2.1 PROCES KONSTRUIRANJA 2-1
2.2 TEORETSKE OSNOVE 2-3
2.2.1 Teorija tehničkih sustava 2-4
2.2.2 Teorije procesa konstruiranja 2-5
2.2.3 Teorija domena i njezina proširenja 2-6
2.2.4 Teorije informacijskih modela 2-8
2.2.5 Računalne tehnologije i jezici 2-9
2.3 STRUKTURA I STRUKTURIRANJE PROIZVODA 2-9
2.3.1 Struktura proizvoda s gledišta procesa konstruiranja 2-11
2.3.2 Struktura proizvoda s gledišta upravljanja informacijama o proizvodu 2-12
2.4 UTJECAJ NA RAD 2-14

3. RAZMJENA I UPRAVLJANJE INFORMACIJAMA O PROIZVODU 3-1

3.1 VARIJANTNOST INFORMACIJA O PROIZVODU 3-1
3.2 DIJELOVI METAMODELA INFORMACIJA U PROCESU RAZVOJA PROIZVODA 3-3
3.2.1 Nositelji informacija o proizvodu 3-5
3.2.2 Subjekti 3-9
3.2.3 Aktivnosti 3-10
3.3 POTREBNE FUNKCIJE SUSTAVA ZA PODRŠKU RAZMJENI I UPRAVLJANJU S PODACIMA O PROIZVODU 3-12
3.4 UTJECAJ NA RAD 3-14

4. REALIZACIJA INFORMACIJSKOG MODELA SUSTAVA PREMA ISO 10303-STEP STANDARDU 4-1

4.1 OPĆENITO O RAZVOJU MEĐUNARODNIH STANDARDA ZA PRIKAZ I RAZMJENU INFORMACIJA O PROIZVODU 4-1
4.2 ISO 10303-STEP STANDARD 4-3
4.2.1 Struktura ISO 10303-STEP standarda 4-4

4.2.2 EXPRESS 4-6
4.3 OSNOVNI ENTITETI STEP PDM SHEME 4-7
4.3.1 Identifikacija i klasifikacija komponenti 4-8
4.3.2 Svojstva komponenti 4-10
4.3.3 Hijerarhijska struktura komponenti i relacije među njima 4-11
4.3.4 Identifikacija i klasifikacija dokumenta 4-13
4.3.5 Vanjske reference i njihova veza s dokumentima 4-13
4.3.6 Svojstva dokumenata i vanjskih referenci 4-14
4.3.7 Veza strukturiranih dokumenata i vanjskih referenci s fizičkim komponentama 4-15
4.3.8 Autorizacija 4-16
4.3.9 Proces konfiguriranja 4-17
4.3.10 Upravljanje poslom i projektima 4-18
4.3.11 Jedinice i mjere 4-19

5. XML TEHNOLOGIJA 5-1

5.1 OPĆENITO O RAZVOJU XML-A 5-1
5.1.1 Povijest XML-a 5-2
5.1.2 Arhitektura web aplikacija: prošlost i budućnost 5-3
5.1.3 Uloge XML-a 5-5
5.2 ARHITEKTURA I NAČIN KORIŠTENJA XML-A 5-8
5.2.1 Osnovni rječnik XML-a 5-8
5.2.2 Glavne značajke i korištenje XML arhitekture 5-11
5.3 XML I STEP – ULOGA XML TEHNOLOGIJE U RADU 5-13

6. RAČUNALNA IMPLEMENTACIJA SUSTAVA 6-1

6.1 OBJEKTNO ORIJENTIRANO PROGRAMIRANJE 6-1
6.2 ARHITEKTURA RAČUNALNE IMPLEMENTACIJE SUSTAVA 6-2
6.3 RAZINA TRAJNOG ZAPISA PODATAKA WEB SERVISA 6-5
6.4 POSLOVNA RAZINA WEB SERVISA 6-8
6.4.1 Dijagram slučajeva korištenja web servisa 6-8
6.4.2 Dijagrami klasa poslovne razine web servisa 6-10
6.4.3 XML operacije 6-25

7. PROTOTIPNA REALIZACIJA WEB SERVISA 7-1

7.1 RJEŠENJE ZASNOVANO NA J2EE© RAZVOJNOJ PLATFORMI 7-1

8. ZAKLJUČAK 8-1

8.1 GLAVNI REZULTATI RADA 8-1
8.2 SMJEROVI DALJNJEG ISTRAŽIVANJA 8-4

9. LITERATURA 9-1

KRATKI ŽIVOTOPIS XIII
SHORT BIOGRAPHY XIII

POPIS SLIKA

Slika 1.1: Distribuirani pristup informacijama o proizvodu 1-2
Slika 1.2: Metodologija rada primijenjenih istraživanja prema *Jorgensenu* [3]..... 1-4
Slika 2.1: Principijelne relacije između procesa konstruiranja i proizvoda [7] 2-2
Slika 2.2: Iz realnosti do računalnog modela, prema [9] 2-3
Slika 2.3: Općeniti model tehničkog procesa 2-4
Slika 2.4: Aktivnosti u procesu konstruiranja kroz četiri razine [20]..... 2-5
Slika 2.5: Nova Teorija domena, adaptirano prema [19] 2-7
Slika 2.6: Revidirani kromosomski model proizvoda [4] 2-8
Slika 2.7: Apstraktna struktura proizvoda [36].....2-10
Slika 2.8: Pogledi na proizvod s različitih perspektiva.....2-13
Slika 3.1: Dio različitih informacija o proizvodu 3-2
Slika 3.2: Veza metamodela i implementacijskih modela 3-4
Slika 3.3: Dijelovi metamodela informacija kojima se upravlja u procesu razvoja proizvoda..... 3-4
Slika 3.4: Funkcionalna struktura sustava 3-14
Slika 4.1: Povijesni razvoj standarda za prikaz informacija o proizvodu 4-2
Slika 4.2: Arhitektura STEP standarda 4-5
Slika 4.3: Pozicija i uloga STEP PDM sheme 4-8
Slika 4.4: Struktura STEP PDM sheme 4-8
Slika 4.5: Dijagram entiteta za identifikaciju i klasifikaciju fizičkih komponenti..... 4-9
Slika 4.6: Dijagram entiteta za prikaz osnovnih svojstva komponenti 4-10
Slika 4.7: Dijagram entiteta za prikaz hijerarhijske strukture komponenti 4-12
Slika 4.8: Dijagram entiteta za prikaz alternativnih i zamjenskih komponenti 4-12
Slika 4.9: : Dijagram entiteta za identifikaciju vanjskih datoteka.....4-13
Slika 4.10: Entiteti za prikaz veza dokumenta sa konstituirajućim datotekama.....4-14
Slika 4.11: Entiteti za prikaz veza dokumenata i vanjskih datoteka sa fizičkim komponentama .4-15
Slika 4.12: Ilustracija veze između dokumenata, datoteka i fizičkih komponenti.....4-16
Slika 4.13: Dijagram entiteta za prikaz entiteta organizacije i osoba.....4-16
Slika 4.14: Dijagram entiteta za definiranje odobrenja.....4-17
Slika 4.15: Dijagram entiteta za definiranje konfiguracijskih koncepata.....4-18

Slika 4.16: Dijagram entiteta upravljanje poslovima 4-18
Slika 4.17: Dijagram entiteta za identifikaciju projekata 4-19
Slika 4.18: Dijagram entiteta za identifikaciju jedinica mjere 4-19
Slika 5.1: Klasična arhitektura web aplikacija 5-3
Slika 5.2: Arhitektura web aplikacija korištenjem XML-a 5-4
Slika 5.3: Primjer dobro strukturiranog XML dokumenta 5-9
Slika 5.4: Arhitektura i korištenje XML tehnologije 5-11
Slika 5.5: Integracija STEP-a i XML-a 5-14
Slika 6.1: Korištenje web servisa 6-4
Slika 6.2: Dijagram slučajeva korištenja (USE CASE) web servisa 6-9
Slika 6.3: Dijagram klasa koje modeliraju upravljanje trajnom pohranom podataka 6-11
Slika 6.4: Dijagram klasa koje modeliraju poslovnu logiku i pravila manipuliranja podacima 6-12
Slika 6.5: Dijagram klasa *OrgPersonController*, *Organization*, *Person* i *OrgPerson* 6-14
Slika 6.6: Dijagram klasa *ActionRequestController*, *ActionRequest*, *RequestStatus*, i *ActionRequestStatus* 6-15
Slika 6.7: Dijagram klasa *ApprovalAssignementController*, *ApprovalStatus* i *AppliedApprovalAssignement* 6-16
Slika 6.8: Dijagram klasa *ConfigurationController*, *ConfigurationItem* i *ConfigurationDesign* 6-17
Slika 6.9: Dijagram klasa *PersonAssignmentController*, *PersonRole*, *AppliedPersonAssignment* 6-18
Slika 6.10: Dijagram klasa *ProjectController*, *OrganizationalProject* i *ProductConcept* 6-19
Slika 6.11: Dijagram klasa *DocumentController*, *AppliedDocumentReference*, *DocumentType* i *DocumentFile* 6-20
Slika 6.12:Dijagram klasa *ProductController*, *Product*, *ProductContext* i *ProductCategory* 6-22
Slika 6.13: Dijagram klasa PDFController, QuantifiedAssemblyComponentUsage, ProductDefinitionFormation, PropertyDefinition, PropertyType 6-24
Slika 6.14: Uloga XML operacija u radu web servisa 6-26
Slika 6.15: Preslikavanje metoda poslovnih komponenti u XML operacije 6-27
Slika 7.1: Java 2 razvojna platforma 7-2
Slika 7.2: Koraci u realizaciji i testiranju predložene arhitekture web servisa 7-3
Slika 7.3: Kreiranje tablica i relacija u relacijskoj bazi 7-4
Slika 7.4: Elementi "*eng. Entity EJB*" komponente 7-5
Slika 7.5: Sučelje web klijenta za testiranje komponenti koje upravljaju trajnim zapisom podataka u relacijsku bazu 7-6

Slika 7.6: Kreiranje nove instance klase *PersonEJB*..... 7-7
Slika 7.7: Prikaz kreirane instance i vrijednosti koje su pohranjene u relacijsku bazu..... 7-7
Slika 7.8: Rezultati pozivanja metode za kreiranje novog korisnika..... 7-7
Slika 7.9: Metode dostupne za svaku instancu klase *PersonEJB*..... 7-8
Slika 7.10: Sučelje web klijenta za testiranje komponenti *PDFController* 7-9
Slika 7.11: Sučelje web klijenta za testiranje XML operacije 7-9
Slika 7.12: XML izlazni dokument sa podacima koji su rezultat obrade.....7-10
Slika 7.13: Prikaz rezultata transformiran u HTML oblik.....7-11

POPIS TABLICA

Tablica 5-1: Osnovna sintaksa XML-a 5-9
Tablica 6-1: SQL opis strukture tablica i relacija..... 6-5

PREDGOVOR

Ovaj rad dio je istraživačkog rada u području unapređenja računalne podrške procesu konstruiranja strojarskih proizvoda, te je prikaz iskustva autora u primjeni i razvoju sustava za upravljanje i razmjenu informacija o proizvodu.

Rad je dio cjelokupnih istraživanja unutar projekta "Model inteligentnog CAD sustava", financiranog od strane Ministarstva znanosti i tehnologije Republike Hrvatske, u kojem je predviđeno informacijsko modeliranje proizvoda kao objekta konstrukcijskog procesa. Očekuje se da realizacija sustava posluži kao osnova za kreiranje naprednijih alata za pomoć konstruktorima u procesu konstruiranja.

SAŽETAK RADA

Istraživanje predstavljeno u ovom radu je usmjereno k razvoju sustava za podršku pri razmjeni i upravljanju informacijama o proizvodu tijekom razvojnog vijeka proizvoda, korištenjem nove generacije mrežnih tehnologija.

U radu je dan pregled teorije tehničkih sustava, teorija procesa konstruiranja, teorije domena, informacijskih teorija i računalnih tehnologija. Predložena je informacijska infrastruktura za opis inženjerskog znanja i modeliranje podataka o inženjerskim proizvodima, te izgradnju računalne platforme za kreiranje, spremanje, pristup i upravljanje tim informacijama. Razvijena je infrastruktura web servisa, zasnovana na XML tehnologiji, koja omogućuje razmjenu informacija i koordinirani razvoj proizvoda u prostorno dislociranim tvrtkama.

Temeljni elementi informacijskog metamodela proizvoda, kojima se gradi predloženi web servis su informacije vezane uz:

- nositelje podataka o proizvodu -fizičke komponente proizvoda i strukturirani dokumenti definirani hijerarhijom, revizijama, varijantama i statusom,
- subjekte koji koriste, sadrže, posjeduju i upravljaju podacima o proizvodu - korisnici, projekti, radne grupe, konstrukcijski zadaci,
- aktivnosti i procese -upravljanje hijerarhijskom strukturom komponenti proizvoda, upravljanje inženjerskim promjenama, procedure odobrenja i autorizacije, dokumentiranje procesa konstruiranja, proces konfiguriranja proizvoda.

Postavljenim metamodelom analizirana je i interpretirana STEP PDM shema te je razvijen implementacijski model sustava za upravljanje i razmjenu informacija o proizvodu sukladno ISO 10303 standardu.

U radu su opisane glavne grupe implementiranih entiteta STEP PDM sheme. Prikazane su prednosti STEP/XML integracije kao osnovnog koncepta sustava koji je predložen. Opisana arhitektura web servisa je implementirana i testirana korištenjem *Java 2 Enterprise Edition* (J2EE™) platforme.

Ključne riječi:

proces konstruiranja, upravljanje informacijama o proizvodu, strukturiranje proizvoda, modeliranje i prikaz informacija o proizvodu, STEP, XML, web servis

SUMMARY

The aim of proposed MS thesis is to research the strategy for application of a new generation web technology tools to support product data management in product development process. All the aspects of the product development process - design, development, manufacturing and product support - are performed more or less collaboratively and concurrently. The strategy presented in this paper intends to bring order to this "apparent chaos" by allowing a digital product development environment to capture the product data, within a digital model that can be viewed, modified and managed simultaneously. In order to support inter-organizational requirements in the new industries of the 21st century, an infrastructure of XML-based web service for collaborative product data management is proposed.

The proposed web service is founded on the informational meta-model that is based on:

- Information binding product data carriers -products physical components and structured documents defined by hierarchy, revisions, variants and status;
- Information binding subjects that create, own and manipulate product data (users, projects, organization, design tasks);
- Information binding activities that use the product data -products' and documents' hierarchical structure management, engineering change management, approval and authorization procedures, design process documentation and history, product configuration processes.

Previously described metamodel elements are used for understanding, interpretation and correct implementation of informational model for the product data exchange, defined by ISO 10303 STEP standard. The main groups of the implemented STEP PDM Schema entities are described. Discussion about benefits of STEP/XML integration as a base concept of the proposed web service is given. The architecture of the web service built by using *Java 2 Enterprise Edition* (J2EE™) platform is described.

Keywords:

design process, product data management, product structuring, product information modeling and representation, STEP, XML, web service

1

UVODNA RAZMATRANJA

U prvoj su glavi definirane značajke istraživanja koje je provedeno tijekom izrade ovog magistarskog rada. Glava započinje objašnjenjem motivacije koja je autora potaknula na istraživanje. Nakon toga je definirano područje istraživanja i glavni ciljevi. Slijedi prikaz metodologije koja je korištena u istraživanju, te je objašnjena struktura rada.

1.1 MOTIVACIJA ZA RAD

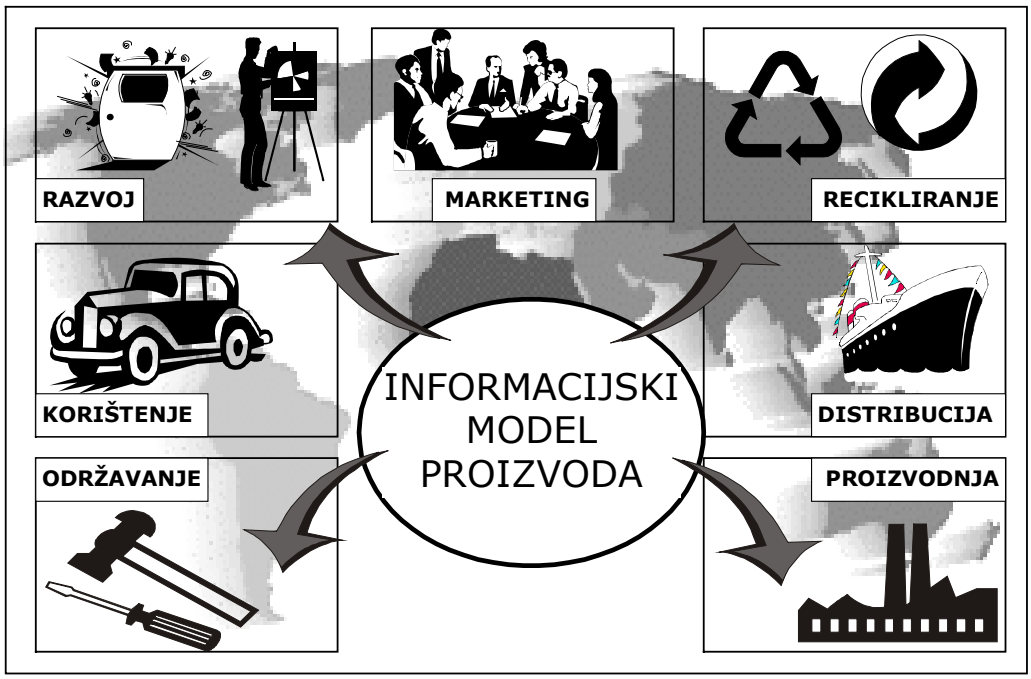
Prikaz informacija o proizvodu i način njihove razmjene mijenjao se relativno sporo u proteklih dvjestotinjak godina. Prije doba industrijske revolucije, inženjerski se zadatak definirao fizičkim modelom proizvoda koji je bilo potrebno reproducirati. Na primjer, čovjek koji je proizvodio cijevi za puške morao je biti siguran da dimenzije cijevi odgovaraju dimenzijama modela cijevi koristeći mjerne uređaje tog doba za prenošenje duljine s modela na cijev koju je izrađivao. Godine 1801., *Gespard Monge* je napisao knjigu *"La Geometrie Descriptive"*, koju možemo smatrati prvom raspravom o modernoj inženjerskoj dokumentaciji [1]. Ona je između ostalog uključivala teoriju projiciranja pogleda na predmet u tri različite ravnine te dodavanje opisa veličine i oblika predmeta.

Korištenje tehničke dokumentacije koja je uključivala projekcije predmeta sa točnim dimenzijama, u praksi je unaprijedilo razvoj proizvoda. Na temelju tako kreirane dokumentacije sastavne su dijelove mogli proizvoditi različiti proizvođači, te su takvi sastavni dijelovi mogli biti bez problema naknadno međusobno sklopljeni. Kao krajnji rezultat, proizvođači su počeli izrađivati kvalitetnije proizvode u većim količinama. Tehnička dokumentacija je postala izlazni produkt konstrukcijskog procesa, ali

istovremeno i ulaz u proces izrade proizvoda. Informacije iz tehničke dokumentacije počele su se koristiti kao podloga za izradu planova proizvodnje, iz kojih su dalje definirane operacije za izradu. Možemo reći da su s pojavom tehničke dokumentacije, različiti ljudi u procesu razvoja proizvoda počeli gledati na iste podatke o proizvodu na drugačiji način, sa stajališta vlastite uloge u tom procesu.

Slijedeći važan iskorak u evoluciji prikaza informacija o proizvodu dogodio se u posljednjih dvadesetak godina XX. stoljeća, kao posljedica pojave i uporabe računala u procesu razvoja proizvoda. "Elektronska" tehnička dokumentacija kreirana pomoću računalnih alata za podršku procesu konstruiranja, omogućila je veću produktivnost u odnosu na klasičnu - "papirnatu" dokumentaciju. Potrebne izmjene mogle su biti napravljene u vrlo kratkom vremenu, a i način pohrane tehničke dokumentacije je postao bitno jednostavniji. Istovremeno su razvijani računalni alati za podršku i ostalim fazama razvoja proizvoda te je omogućena razmjena informacija o geometriji proizvoda između takvih sustava.

Danas na početku novog milenija, zahtjevi su tržišta veći nego ikad. Proces globalizacije, smanjenje životnog vijeka proizvoda, rastuća složenost i potreba za većom varijantnošću proizvoda traže od proizvođača veću produktivnost i konkurentnost na tržištu. Razvoj današnjih kompleksnih proizvoda kao što su automobili, avioni ili brodovi, nije djelo samo jednog čovjeka ili manje grupe ljudi. Ovaj proces se danas najčešće dijeli među cijelom mrežom različitih proizvođača koji uglavnom nisu geografski smješteni na jednom mjestu, već mogu biti raspodijeljeni po cijelom svijetu. Osnovni preduvjet za ostvarivanje ovakve suradnje je da u svakom trenutku moraju svim sudionicima koji su uključeni u razvoj, biti na raspolaganju sve potrebne informacije o trenutnom stanju i statusu projekta.



Slika 1.1: Distribuirani pristup informacijama o proizvodu

Količina tih informacija te fleksibilnost i otvorenost koja se zahtijeva u njihovoj razmjeni i upravljanju, otežavaju njihovu kontrolu i održavanje. Upravo je problematika distribuiranog pristupa informacijama o proizvodu [Slika 1.1], istaknula potrebu za korištenjem računalnih sustava u tu svrhu. Takvi sustavi daju podršku pri razmjeni i upravljaju informacijama o proizvodu, osiguravajući da je prava informacija dostupna pravom korisniku u pravo vrijeme i u pravom obliku. Koristeći digitalni model proizvoda kao osnovu za prikupljanje, pregledavanje, analiziranje i mijenjanje podataka, primjena navedenih računalnih sustava omogućuje efikasno praćenje i kontroliranje informacija o proizvodu tijekom cijelog životnog vijeka proizvoda.

Kao dodatan poticaj distribuiranom razvoju proizvoda pridonio je i istovremeni razvoj globalne računalne mreže (Interneta), koji je zadnjih godina donio nagli prijelaz sa statičkih k dinamičkim sadržajima dostupnim na mreži. Dinamičke značajke dale su dodatnu snagu i interaktivnost mrežnim aplikacijama kroz mogućnost dinamičkog manipuliranja podacima. Preko dinamičkih Internet stranica članovima razvojnog tima je omogućeno simultano praćenje podataka o razvoju proizvoda, uključivanje s vlastitim radom i suradnja s ostalima na dovršenju projekata.

Završavajući ovaj kratki pregled možemo zaključiti da smo u dvjestotinjak godina evolucije napredovali od prvih tehničkih crteža do pojma "elektronskog inženjerstva" (*eng. e-engineering*). E-inženjerstvo je promijenilo način na koji shvaćamo komunikaciju, suradnju i razmjenu znanja između pojedinaca, grupa, odjela i poduzeća kroz cijeli životni vijek proizvoda. To je razlog zašto su sustavi koji nam nude rješenja u području e-inženjerstva, danas podjednako glavni predmet interesa kako u istraživačkim tako i u industrijsko-razvojnim zajednicama.

1.2 CILJEVI ISTRAŽIVANJA I PODRUČJE

Potaknuto problematikom distribuiranog razvoja proizvoda i iz toga proizašle potrebe za razmjenom informacija o proizvodu, istraživanje predstavljeno u ovom radu je usmjereno k razvoju sustava za podršku pri razmjeni i upravljanju informacijama o proizvodu tijekom razvojnog vijeka proizvoda. Realizacija takvog sustava zahtjeva definiranje informacijske infrastrukture za formaliziranje inženjerskog znanja i modeliranje podataka o inženjerskim proizvodima, te računalnu platformu za kreiranje, spremanje, pristup i upravljanje tim informacijama. Istraživanje prikazano u ovom radu temelji se na najvažnijim teoretskim konceptima i industrijskim standardima u upravljanju i razmjeni informacija o proizvodu, te dokumentira predloženu arhitekturu sustava. Zbog prethodno navedenih tvrdnji ovo istraživanje možemo svrstati u područje primijenjenih istraživanja.

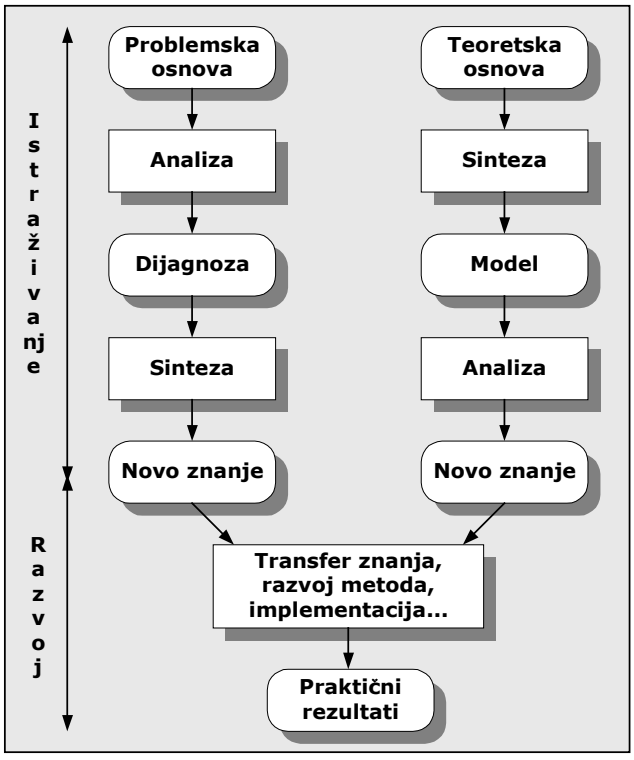
Prema *Ropoholu* [2], primijenjena istraživanja imaju dvije vrste ciljeva: teoretske i praktične. Vezano uz tu činjenicu možemo izdvojiti dva osnovna cilja predstavljenog istraživanja: definiranje fundamentalnih koncepata i značajki sustava za podršku razmjeni i upravljanju s informacijama o proizvodu, te prijedlog načina

implementacije sustava u realnu radnu okolinu. Ovdje je potrebno naglasiti da u okviru rješavanja prvog zadatka nije moguće obuhvatiti sve aspekte koji se mogu pojaviti u uporabi takvih sustava, već je u radu prikazan uravnoteženi skup najvažnijih zahtjeva i funkcija prema ocjeni autora. Posebna pozornost u radu je posvećena ISO 10303 – STEP (*eng.* ***ST****andard for* ***E****xchange of* ***P****roduct Data*) internacionalnom standardu za razmjenu podataka o proizvodu, te istraživanju mogućnosti uporabe tog standarda kao semantičke i strukturne osnove modela podataka o proizvodu, koja čini "jezgru" sustava. Obzirom na kompleksnost prethodno navedenog standarda u radu će se obuhvatiti samo pregled dijelova standarda značajnih za opisivanje proizvoda i upravljanje podacima o proizvodu.

Drugi dio postavljenih ciljeva obuhvaća opis predložene implementacije sustava. Ovdje je bitno naglasiti da je sustav namijenjen za uporabu među korisnicima koji pristupaju i spremaju podatke u mrežnom okruženju, pa je i predložena implementacija prilagođena tehnologiji koja je dostupna u ovom okruženju. Posebna pažnja je posvećena primjeni standarda za razmjenu elektronskih informacija– XML tehnologije. U radu će se definirati način implementacije sustava za razmjenu i upravljanje informacijama o proizvodu, na temelju prethodno predložene STEP semantičke i strukturne osnove, korištenjem XML tehnologije.

1.3 PRISTUP – ISTRAŽIVAČKE METODE

Metodologija istraživanja opisanih u ovom radu temelji se na kritičkom racionalizmu i induktivizmu, prema *Jorgensenu* [3].



Slika 1.2: Metodologija rada primijenjenih istraživanja prema *Jorgensenu* [3]

Prema toj metodologiji se postojeće teorije, standardi, modeli i metode dalje razvijaju na temelju proučavanja relevantne literature, logičkog strukturiranja i praktičnih razmatranja.

Početna točka istraživanja prikazanih u radu je realna problemska osnova koja je analizirana te su dijagnosticirani glavni zahtjevi za njezino rješavanje [Slika 1.2]. Paralelno ovoj aktivnosti, definirano problemsko područje je istraženo u kontekstu teoretskih osnova, što je opisano potrebnim dijelovima teoretskog metamodela koji opisuje područje postavljenog problema. Da bi se potvrdili rezultati istraživanja, realizirana je i testirana prototipna izvedba sustava za podršku razmjeni i upravljanju informacijama o proizvodu.

Ovdje se mora naglasiti da je kod primijenjenih istraživanja vrlo teško dati pravilnu ocjenu vrijednosti dobivenih rješenja, jer rezultati ne mogu biti dokazani formalnim znanjem, a vrlo je teško dobiti i empirijske vrijednosti. Kao jedini kriterij preostaje uspješnost višestruke implementacije za rješavanje realnih problema, te usporedba sa sličnim projektima.

1.4 STRUKTURA RADA

U ovom se radu analiziraju koncepti upravljanja informacijama o proizvodu, te opisuje razvoj i implementacija sustava zasnovanog na tim konceptima. Sukladno tome rad je podijeljen u dvije logičke cjeline. Prva cjelina započinje uvodnom glavom, koja uključuje pregled motivacije za istraživanje, te definira glavne ciljeve i metodologiju istraživanja, a nastavlja se kroz drugu, treću i četvrtu glavu.

U drugoj su glavi prikazane teoretske osnove koje čine polazišnu točku istraživanja. Ukratko su opisane Teorija tehničkih sustava, Teorija procesa konstruiranja, Teorija domena, informacijske teorije te računalne tehnologije i programski jezici koje su primijenjeni u istraživanju. Osim navedenog, glava donosi pregled područja strukturiranja proizvoda i hijerarhijske strukture proizvoda koja se promatra kao poveznica nositelja informacija o proizvodu.

U trećoj se glavi analizira problematika razmjene i upravljanja informacijama o proizvodu, u svrhu razumijevanje i pravilne implementacije informacijskog modela definiranog ISO STEP 10303 standardom. Nakon uvodnog dijela u kojem se općenito govori o pojmu informacija o proizvodu i njihovoj varijantnosti, veći dio glave posvećen je analizi elementa koje sadrži metamodel informacija u razvojnoj fazi proizvoda. Na kraju trećeg glave definirane su potrebne osnovne funkcije sustava.

U četvrtoj glavi je ukratko objašnjen povijesni razvoj međunarodnih standarda za razmjenu podataka o proizvodu. Opširnije je opisana struktura ISO 10303-STEP standarda koji se kroz svoj dugogodišnji razvoj profilirao kao standard za razmjenu podataka o proizvodu. Poseban je naglasak stavljen na entitete STEP PDM sheme, odnosno njezine dijelove čijom semantikom je realiziran informacijski model proizvoda koji je definiran kao osnova sustava koji se istražuje.

Druga logička cjelina rada proteže se kroz petu, šestu i sedmu glavu te opisuje metodologiju implementacije. U petoj je glavi analizirana XML tehnologija. Uvodni dio glave donosi pregled povijesti razvoja XML standarda te definiciju osnovnih pojmova. Glavni dio glave je posvećen opisivanju uloga u kojima se XML tehnologija danas primjenjuje, te osnovnih značajki XML-a. Na kraju je glave objašnjena veza XML-a i STEP standarda, te uloga XML tehnologije u istraživanju koje je prikazano u ovome radu.

U šestoj je glavi predložena metodologija implementacije sustava. Uvodni dio glave donosi pregled i definiciju osnovnih pojmova objektno orijentiranih tehnika modeliranja računalnih sustava i programiranja. Glavni dio glave je posvećen definiranju troslojne arhitekture sustava koji se kako je predloženo realizira u obliku web servisa. Za svaku od tri razine arhitekture, (klijentsku razinu, razinu poslovne logike i razinu trajnog zapisa podataka), u šestoj su glavi definirane i opisane osnovne značajke.

Sedma glava opisuje prototipnu realizaciju sustava korištenjem Java 2 razvojne platforme, odnosno J2EE© (*eng. Java 2 Enterprise Edition*) specifikaciju tehnologija tvrtke Sun Microsystems©. Navedena razvojna platforma je izabrana zbog njezinih značajki koje ju definiraju kao platformski neutralnu, portabilnu, višekorisničku i sigurnu osnovu za razvoj distribuiranih programskih aplikacija, što u potpunosti odgovara zahtjevima preložene implementacijske metodologije.

Ovaj se rad završava s zaključnom glavom u kojoj se analiziraju rezultati rada te predlažu pravci daljnjih istraživanja.

2

PREGLED STANJA I TEORETSKIH OSNOVA ISTRAŽIVANJA

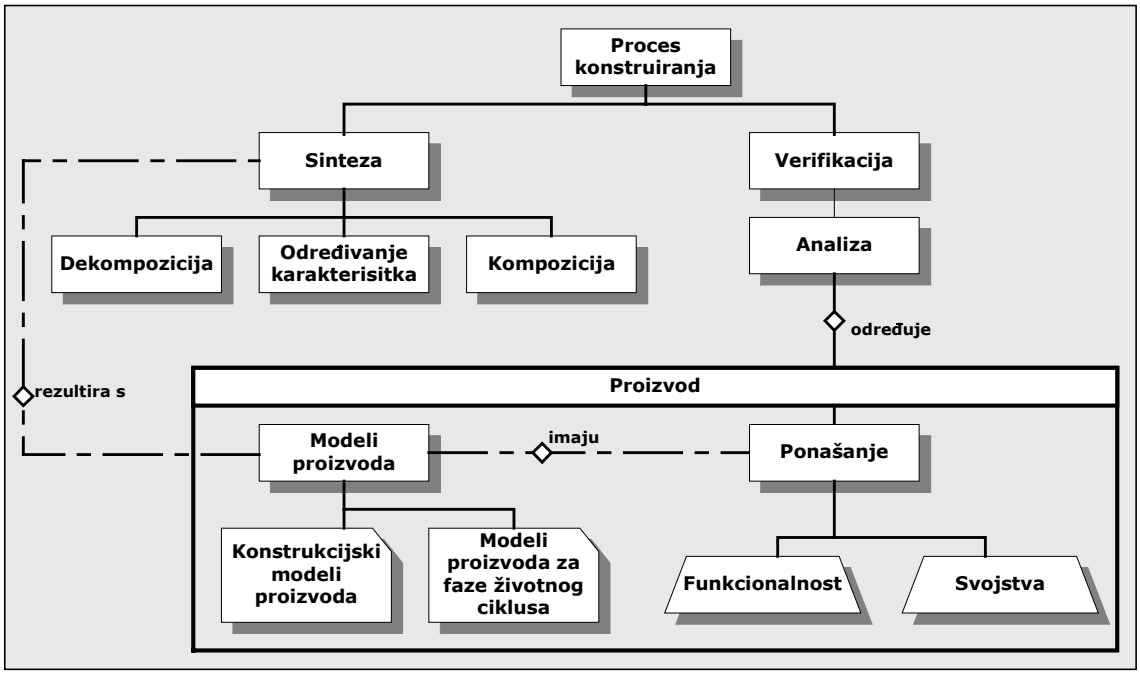
U drugoj su glavi prikazane teoretske osnove koje čine polazišnu točku istraživanja. Ukratko su opisane Teorija tehničkih sustava, Teorija procesa konstruiranja, Teorija domena, informacijske teorije te računalne tehnologije i programski jezici koji su primijenjeni u istraživanju. U drugom dijelu glave dan je pregled područja strukturiranja proizvoda i hijerarhijske strukture proizvoda promatrane kao glavne poveznice nositelja informacija o proizvodu.

2.1 Proces konstruiranja

Proces konstruiranja može se definirati kao intelektualni proces koji rezultira proizvodom određenim s zahtijevanom funkcionalnošću i svojstvima [4]. Funkcionalnost i svojstva definiraju određeno ponašanje proizvoda koje će on manifestirati tijekom svojeg uporabnog vijeka. Pri tome je bitno naglasiti da ponašanje ne može biti ukonstruirano direktno u proizvod, nego ga definiraju gradivni elementi proizvoda, relacije između njih, te utjecaji iz okoline. Konstruirajući proizvod, inženjer određuje ne samo gradivnu strukturu komponenti koje čine proizvod, nego definira čitav niz različitih struktura koje su nositelji informacija o ponašanju i izvedbi proizvoda [5] (npr. struktura hidrauličke ili pneumatske regulacije, struktura mehanizama itd.). Različite strukture zajedno čine definiciju proizvoda, koja je prilagođena potrebama pojedinih faza životnog vijeka, kao što su na primjer proizvodnja, prodaja, korištenje, održavanje [6]. Koordiniranje i upravljanje različitim strukturama važan je čimbenik uspješnosti procesa razvoja proizvoda.

Prema *Andreasenu* [7], konceptualna faza procesa konstruiranja sastoji se od dvije potpuno različite aktivnosti: sinteze i verifikacije. Sinteza je aktivnost tijekom koje se proizvod određuje značenjem pojedinih atributa. Prilikom sinteze proizvoda inženjer za modeliranje proizvoda primjenjuje različite modele koji se sastoje od elementa modela i relacija između tih elemenata. Aktivnost sinteza se dalje dijeli u dekompoziciju, karakterizaciju i kompoziciju. Dekompozicija znači raščlambu proizvoda koji se konstruira u jednostavnije podsustave. Karakterizacija je proces u kojem inženjeri određuju karakteristike podsustava proizvoda i njihove vrijednosti. Tijekom kompozicije se određuju sučelja između pojedinih podsustava proizvoda zbog njihove integracije u cjelovito rješenje. Za razliku od sinteze, verifikacija je aktivnost u kojoj se kontroliraju svojstva proizvoda i njegovih podsustava prema zahtjevima konstruktorskog zadatka. Aktivnost analize se u konceptualnoj fazi procesa konstruiranja, može promatrati kao podskup aktivnosti verifikacije, kojom se određuje ponašanje proizvoda kroz analizu funkcionalnosti i svojstava podsustava proizvoda.

Principijelne relacije između procesa inženjerskog konstruiranja i proizvoda mogu se definirati [7] kako slijedi [Slika 2.1].



Slika 2.1: Principijelne relacije između procesa konstruiranja i proizvoda [7]

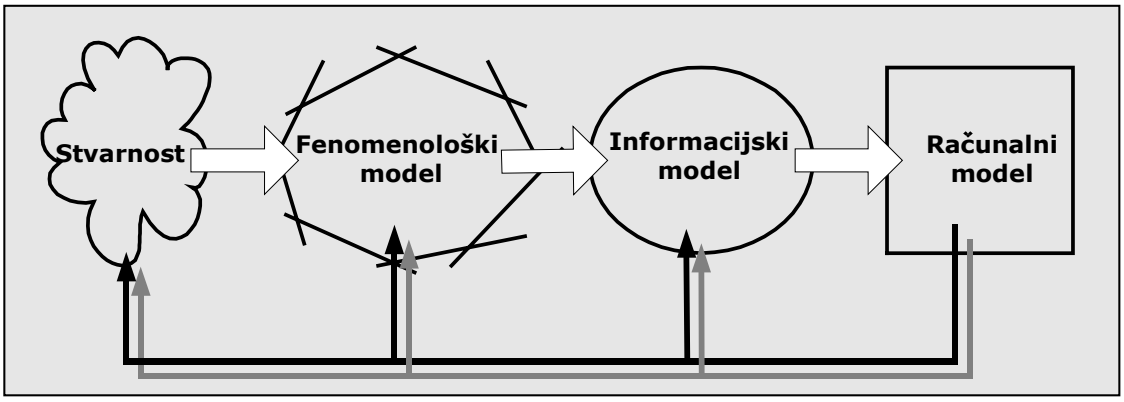
Modeli proizvoda nastali tijekom njegove sinteze se konstituiraju od sastavnih podsustava proizvoda definiranih svojim karakteristikama, te međusobnim vezama tih podsustava [8]. Na taj način aktivnost sinteze rezultira opisom strukture proizvoda, koji će se tijekom korištenja ponašati u skladu sa zahtjevima.

Za efikasno odvijanje procesa konstruiranja potrebno je definirati informacijsku infrastrukturu koja je u mogućnosti spremiti i administrirati informacije o proizvodu koji se konstruira. Specifikacija podataka koji čine informacijsku infrastrukturu zove

se informacijski model proizvoda [8], te je on prikaz realnih svojstava proizvoda. Potreba za informacijskim modelom proizvoda je indicirana svojstvom konstruiranja kao dinamičkog procesa donošenja odluka, što rezultira stalnom nastajanjem novih podataka u procesu konstruiranja te višestrukim promjenama njihovih vrijednosti.

2.2 Teoretske osnove

Ukupni cilj znanosti o konstruiranju je doprinos poboljšanju razumijevanja fenomena konstruiranja, koji je izražen kroz poboljšane modele proizvoda i procesa konstruiranja [4]. Proces konstruiranja i proizvodi kao njegov fizički rezultat, modeliraju se brojnim, u literaturi opisanim teorijama [9], [10], [11], [13], [14], [15]. Opisane metode modeliranja i modeli procesa konstruiranja na međusobno slične načine, prikazuju odnose stvarnog i apstraktnog svijeta. Prema [9], tijekom preslikavanja stvarnog svijeta u računalne modele, razlikujemo međukorake, odnosno zasebne modele (fenomenološke i informacijske), koji se temelje na različitim teoretskim osnovama [Slika 2.2].



Slika 2.2: Iz realnosti do računalnog modela, prema [9]

Fenomenološki modeli se osnivaju na teorijama konstruiranja, kao što je npr. Teorija tehničkih sustava [10], te definiraju osnovu za modeliranje strukture i ponašanja proizvoda [12],[13],[14]. Informacijske teorije čine osnovu za formalno modeliranje proizvoda i procesa konstruiranja usmjereno k računalnoj implementaciji. Informacijski modeli se osnivaju na informacijskim teorijama, npr. *eng. entity-relationship* modeliranju ili objektno-orijentiranom modeliranju [16]. Računalni modeli koriste informatičku znanost za uspostavu računalnih modela inženjerskih procesa i proizvoda te se zasnivaju na računalnim tehnologijama i programskim jezicima, kao npr. C++, Java, Visual Basic, SQL, XML,...

Preslikavanje između navedenih modela s lijeva u desno [Slika 2.2], objašnjava tijek istraživanja u kojem se fenomenološki modeli formaliziraju u informacijske i računalne modele. Relacije u suprotnom smjeru predstavljaju potvrđivanje prilikom kojeg se svaka od tri klase modela sučeljava sa realnošću koju modelira, te se na taj način provjerava njihova uporabljivost. Čvrsta granica između prikazane tri klase

modela znači da se teoretske osnove koje postoje u pozadini istraživanja mijenjaju kroz preslikavanje, iz teorija konstruiranja i proizvoda do računalnih teorija. Kroz istraživački dio ovog rada uglavnom su proučavani fenomenološki i informacijski modeli podataka o proizvodu, a prilikom implementacije je kreiran računalni model sustava za razmjenu i upravljanje informacijama o proizvodu.

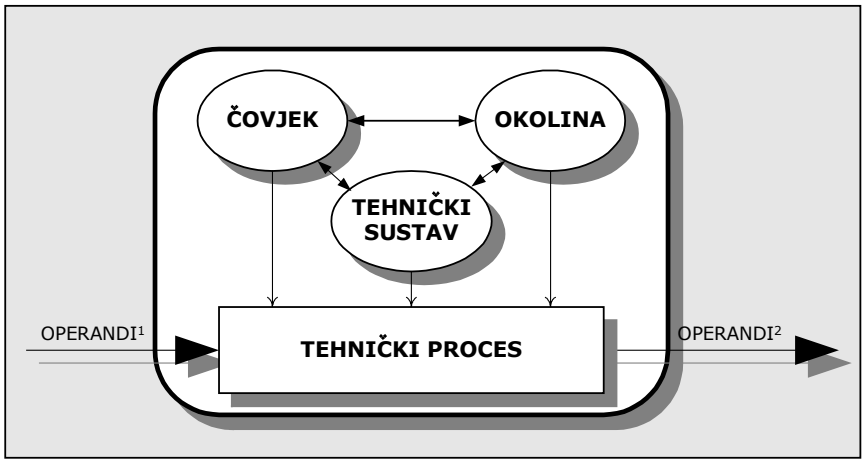
Osnovu rada čini pet glavnih teoretskih područja koje će biti ukratko objašnjene u idućim poglavljima:

- teorija tehničkih sustava,
- teorije procesa konstruiranja,
- teorija domena,
- teorije informacijskih modela,
- računalne tehnologije i jezici.

2.2.1 Teorija tehničkih sustava

Primarni cilj Teorije tehničkih sustava je klasificiranje i kategorizacija znanja o tehničkim sustavima u uređeni skup zaključaka vezanih uz njihovu prirodu, reguliranje kontrole, i razvoj [17]. Teorija tehničkih sustava modelira fenomenološki koristeći pravila i metode za modeliranje tehničkih sustava, te je kao takva primjenjiva za opisivanje strukture i ponašanja proizvoda.

Hubka i *Eder* [17] su definirali da su tehnički sustav (proizvod), čovjek i okolina potrebni za izvođenje tehničkog procesa u kojem se operandi transformiraju iz ulaznog stanja u izlazno stanje [Slika 2.3]. U toj transformaciji operanda mijenjaju se njihovi atributi. Da bi se izvršio tehnički proces potrebni su efekti. Tehnički sustav, čovjek i okolina kreiraju te efekte, npr. svjetlost, silu, toplinu. Tehnički proces može se podijeliti u podprocese. Svaki od tih podprocesa također zahtjeva efekte za svoje izvršenje. Potpunost svih podprocesa, odnosno uspješno transformiranje njihovih operanda od ulaznog stanja prema izlaznom, određuje svrhu tehničkog sustava. U strukturi komponenti koje čine tehnički sustava mogu se razlikovati skup konstitucijskih elementa i skup relacija između njih.



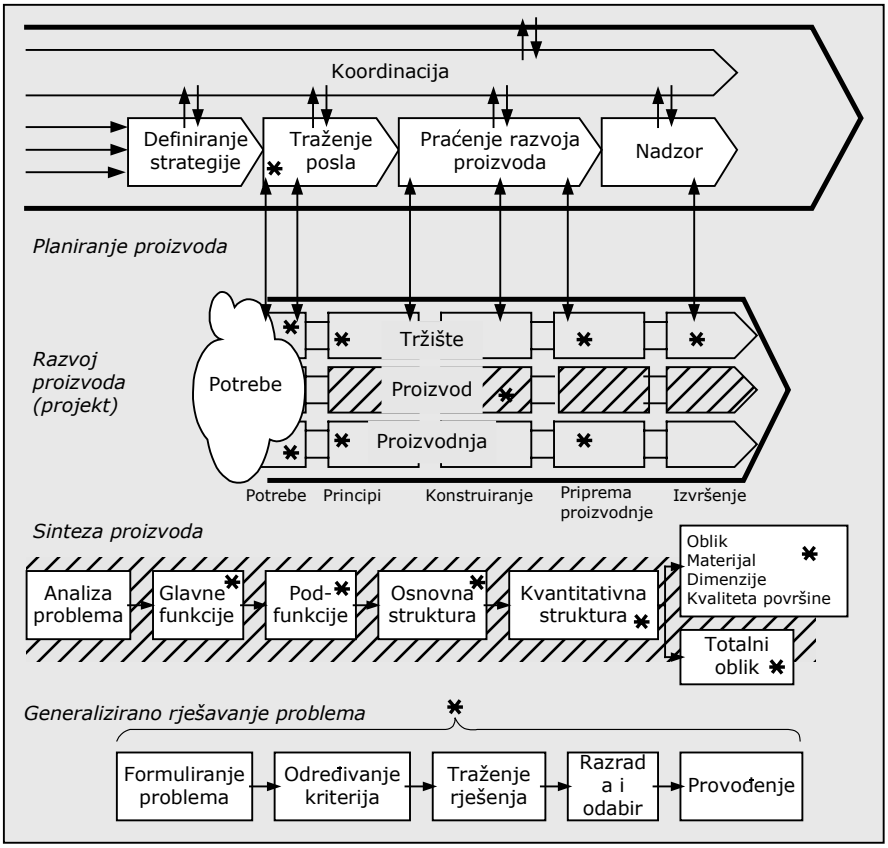
Slika 2.3: Općeniti model tehničkog procesa

Govoreći terminologijom biologije, struktura komponenti se može nazvati anatomskom strukturom tehničkog sustava [18], [19]. Kompozicija komponenti čini složenije komponente, koje dalje čine strojeve itd. Teorija tehničkih sustava je relevantna za ovaj rad iz razloga što objašnjava kako i zašto modelirati proizvod (tehnički sustav), proučavanje kojeg je predmet ovog rada.

2.2.2 Teorije procesa konstruiranja

Teorije koje modeliraju proces konstruiranja su općenite i sveobuhvatne fenomenološke teorije, deskriptivne i preskriptivne, koje racionalno objašnjavaju prirodu konstrukcijskog procesa [10], [11], [12], [13], [14], [15]. Proces konstruiranja može se opisati s različitih gledišta, kao npr. planiranja, organizacije, metoda, aktivnosti. U smislu istraživanja koja su obuhvaćena ovim radom, posebno je interesantan segment aktivnosti u procesa konstruiranja. Sa stanovišta aktivnosti, proces konstruiranja može se modelirati kroz četiri razine, [20] [Slika 2.4]:

1. planiranje proizvoda,
2. razvoj proizvoda,
3. sinteza proizvoda,
4. generalizirano rješavanje problema.



Slika 2.4: Aktivnosti u procesu konstruiranja kroz četiri razine [20]

Planiranje proizvoda sastoji se od aktivnosti kao što su određivanje poslovne strategije, traženje mogućih poslova, praćenje tehnološkog razvoja te odabir i

koordinacija mogućih projekta. O aktivnostima planiranja proizvoda ovisi proizvodna strategija, strategija ponašanja na tržištu te tehnološka strategija. Razvoj proizvoda uključuje sve aktivnosti nužne za ostvarenje proizvoda od definiranje zahtjeva, preko određivanja principa rada pojedinih rješenja, detaljne razrade do priprema za proizvodnju. Razvoj proizvoda za cilj ima uspostavu simultanosti, integraciju i uključivanje značajki životnog vijeka proizvoda.

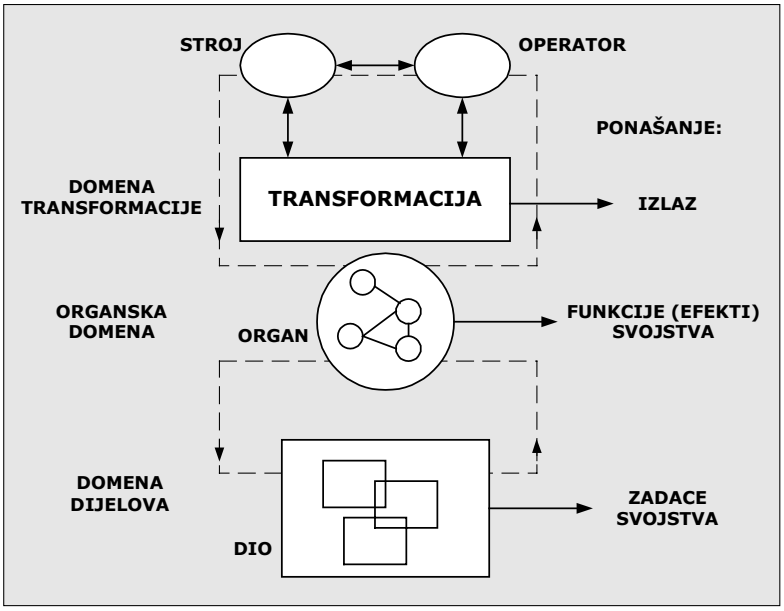
Sinteze proizvoda kao objekt interesa ima mehanički proizvod, te se njome određuju aktivnosti potrebne za definiranje karakteristika proizvoda. Generalizirano rješavanje problema definirano je aktivnostima koje se temelje na promatranjima i teorijama za opisivanje načina čovjekovog razmišljanja i rješavanja postavljenih probleme. Važni elementi su kreativnost, razrada, metode za sistematsku sintezu, upravljanje informacijama i dokumentima. Metodologija generaliziranog rješavanja problema vrijedi za rješavanje različite tipove problema, te je kao takva uključena u pojedine aktivnosti ostalih razina kroz koje se promatra proces konstruiranja. Razmatranje aktivnosti u procesu konstruiranja važno je za ovaj rad zbog određivanja granica unutar kojih je potrebno definirati informacije koje je potrebno razmjenjivati u procesu konstruiranja te na kakav način njima upravljati.

2.2.3 Teorija domena i njezina proširenja

U teoriji domena [19], [21], sinteza proizvoda je objašnjena određivanjem četiri različita modela sustava. Proizvod može biti promatran kao sustav gledajući ga u četiri različite domene ili područja: proizvod kao proces, kao funkcija, kao organ, i kao dio (*eng. part*). Kad se proizvod modelira kao procesni sustav, njegovi elementi su procesi, a relacije između njih su operandi: materijal, energija i informacije. Glavni razlog za modeliranje proizvoda kao procesa je određivanje što bi proizvod trebao biti, te kako su čovjek i okolina vezani uz njega. Kad se proizvod modelira kao funkcionalni sustav, njegovi elementi su funkcije koje proizvod treba ostvariti, a relacije mogu biti vrijeme, prostor ili logika. Glavni razlog za modeliranje proizvoda kao funkcionalnog sustava je određivanje osnovnih gradivnih blokova sustava prema zahtjevima te kontrola da li je proizvod u stanju ispuniti sve postavljene zahtjeve.

Aktivni elementi u konstrukciji, koji kreiraju potrebne efekte za izvršenje tehničkog procesa se u Teoriji domena nazivaju organi. Organski sustav, u suprotnosti od funkcionalnog je strukturalni opis konstrukcije. Kad se proizvod modelira s organskim sustavom, elementi sustava su organi ili organizmi (kompleksni organi), koji su vezani u parove i poretke. Glavni razlog za modeliranje proizvoda kao organskog sustava je objašnjavanje funkcionalnosti. Dio (*eng. part*) je fizički materijalizirani element proizvoda. On se definira kao osnovni element izrađen od jednog materijala, bez operacija spajanja. Kad se proizvod modelira u ovoj domeni elementi su sklopovi i njihovi jednostavniji dijelovi sa fizičkim vezama između njih. Razlog za ovakvo modeliranje proizvoda je specificiranje što sustav mora isporučiti kao konačno rješenje.

Potrebno je naglasiti da je u zadnjih nekoliko godina ova teorija doživjela najveće promjene [Slika 2.5]. Današnja polazišta nove Teorije domena su takva da striktno govoreći, ne postoji funkcionalna struktura mehaničkih sustava, nego zajednička struktura koja definira međusobne veze organa koji su konstituirani prema njihovoj funkcionalnosti [19]. Može se reći da je funkcionalna domena izostavljena iz nove Teorije domena, te se funkcije razmatraju kao klasa ponašanja organa zajedno sa klasom svojstava organa.



Slika 2.5: Nova Teorija domena, adaptirano prema [19]

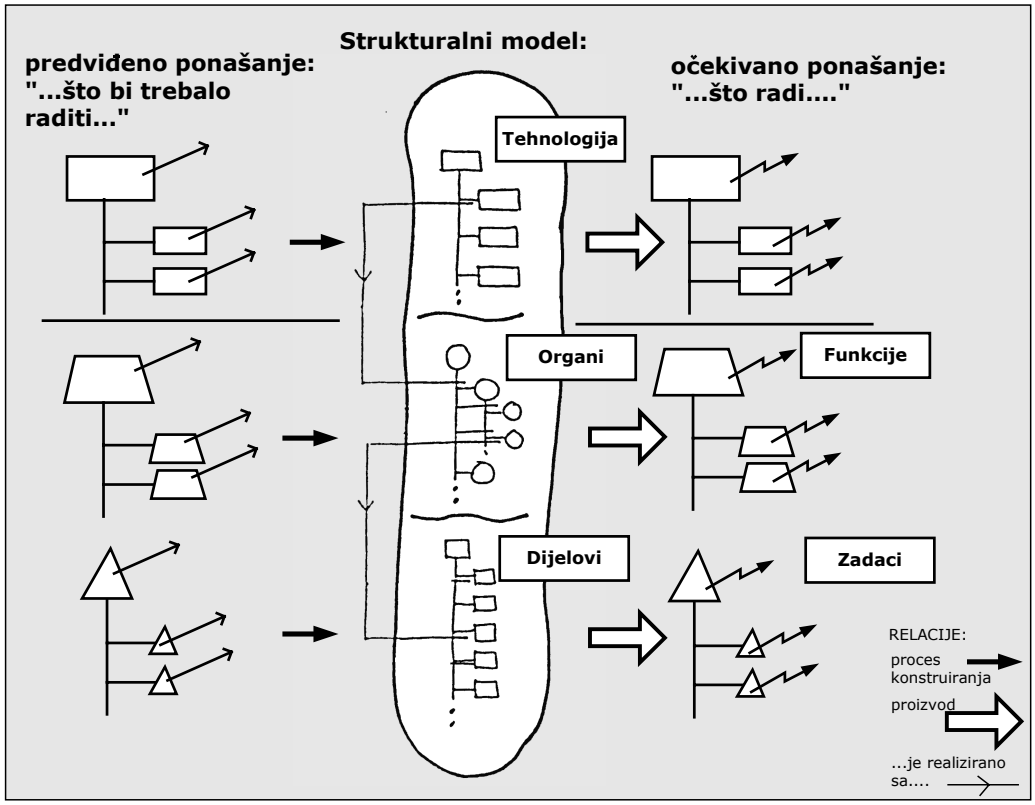
Kao jedna od primjena Teorije domena u modeliranju proizvoda razvijen je kromosomski model [22], koji je kasnije u skladu s promjenama koje su nastale u novoj Teoriji domena revidiran [4] [Slika 2.6]. Temeljna ideja kromosomskog modela je da u idealnom slučaju ovakav model treba moći, baš kao ljudski kromosom, objediniti potpunu definiciju proizvoda. Kromosomski model proizvoda omogućuje opisivanje izvođenja tehničkog procesa, ali i opisuje detalje pojedinog dijela proizvoda te na taj način povezuje ponašanje i funkciju proizvoda sa njegovim komponentama.

Iz perspektive ovog rada posebno je interesantno proširenje Teorije domena na Teoriju strukturiranja, prema kojoj tehnički sustav može biti strukturiran sa četiri različite gledišta [23]:

- vezano uz četiri domene različite domene (proces, funkcije, organi, dijelovi),
- orijentirano prema akciji, ovisno o različitim tipovima odnosa koje nalazimo u području razvoja proizvoda (kinematika, termodinamika, odnos čovjek-stroj…),
- gledano iz perspektive životnog vijeka proizvoda (proizvodnja, prodaja, korištenje, održavanje…).

Na ovaj način, Teorijom strukturiranja su definirani različiti strukturalni pogledi na

proizvod, što je od velike važnosti za analiziranje problematike koja se pojavljuje pri razmjeni i upravljanju informacijama o proizvodu.



Slika 2.6: Revidirani kromosomski model proizvoda [4]

2.2.4 Teorije informacijskih modela

Informacijski se model može shvatiti kao formalni opis ideja, činjenica i procesa koji zajedno čine model dijela stvarnog svijeta i osiguravaju eksplicitni skup interpretacijskih pravila [24]. U idealnom slučaju, informacijski model treba osigurati potpunu, točnu i jedinstvenu sliku pojave koja se modelira. Prema [24] informacija je definirana kao znanje o idejama, činjenicama i/ili procesima. S druge strane, informacije potrebne u različitim slučajevima primjene, predstavljaju podatke, tj. simbole ili funkcije definirane u skladu s implicitnim ili eksplicitnim pravilima. Može se stoga reći da je svrha korištenja informacijskog modela, formuliranje opisa stvarnog svijeta, na način da se informacije mogu koristiti i razmjenjivati bez znanja o izvoru informacija.

Informacijske modele možemo podijeliti u modele specifične namjene i općenite modele. Modeli specifične namjene određuju konceptualni model koji je usmjeren na određeno područje primjene. Primjer takvih modela su TAXIS [25], SDM [26], SAM [27] i O_2 [28]. Za razliku od modela specifične namjene, pristup općenito primjenjivih modela se realizira preko skupa specifičnih konceptualnih modela. Prvi općeniti modeli bili su hijerarhijski i mrežni modeli [29], nakon čega su slijedili relacijski [30] modeli NIAM [31] i familija IDEF [32] modela. Predstavnici modela koji

dijele karakteristike obiju koncepcija su EDM [33] i EXPRESS [24]. Budući da je EXPRESS definiran ISO 10303 standardom i kao takav se koristi kao osnova za modeliranje podataka o proizvodu u kontekstu njihove razmjene i upravljanja, njegove značajke opisane su u četvrtoj glavi.

Za modeliranje arhitekture te realizaciju sustava, u istraživanju su korišteni principi objektno orijentiranog modeliranja i programiranja: modelirati aplikaciju kao skup objekata koji međusobno komuniciraju da bi postigli zajednički cilj. Važno je naglasiti da se objektno orijentirani pristup ne bavi programiranjem u smislu razvoja algoritama i struktura podataka, već ga treba promatrati kao skup metoda (postupaka i alata) za organiziranje aplikacija. Općenitije možemo reći da je objektno orijentirani pristup programiranju ustvari predstavlja tehniku za koncipiranje programskih sustava [34]. Osnovni elementi strukturiranja programa su objekti. Objekti modeliraju entitete iz stvarnog svijeta, mogu obuhvaćati apstrakcije kompleksnih fenomena ili mogu predstavljati elemente programskog sustava (npr. stogove ili upravljanje grafičkim prikazom). Operacijski gledano, objekti kontroliraju računalni proces. Iz perspektive razvoja programskog sustava, najvažnija karakteristika objekata nije njihovo ponašanje, nego činjenica da se ponašanje objekta može opisati apstraktnom karakterizacijom njegova sučelja. Takva apstraktna karakterizacija dovoljna je za početno koncipiranje sustava. Stvarno ponašanje objekta može se implementirati i doraditi kasnije, prema specifičnim potrebama implementacije. Detaljniji prikaz objektno orijentiranih tehnika opisan je šestoj glavi gdje se govori o računalnoj implementacija sustava.

2.2.5 Računalne tehnologije i jezici

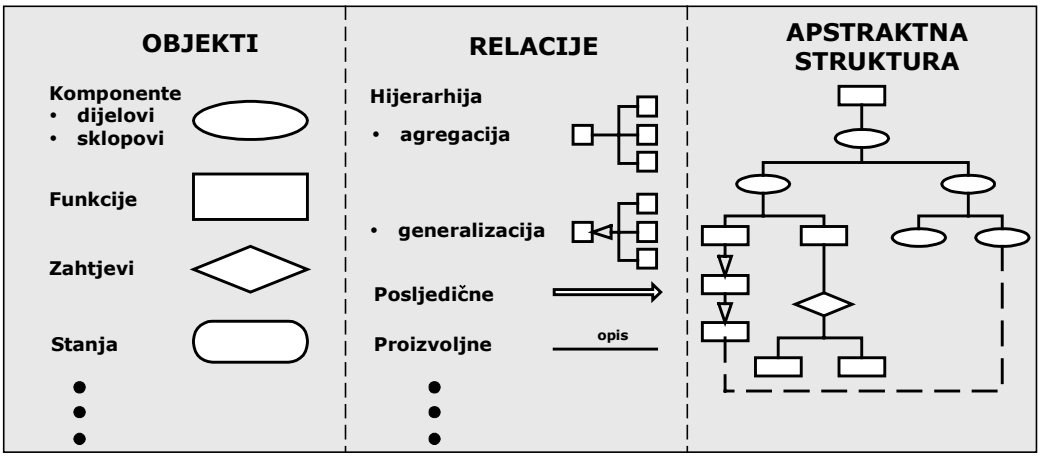
Razvoj sustava za podršku razmjeni i upravljanju informacijama o proizvodu temelji se na postojećim računalnim tehnologijama i konceptima, odnosno dostupnom softveru koji se može koristiti u evaluacijske svrhe. Tako je implementacijsko rješenje koje je ponuđeno u potpunosti zasnovano na Java 2 razvojnoj platformi, odnosno J2EE© (*eng. Java 2 Enterprise Edition*) specifikaciji tehnologija tvrtke Sun Microsystems© [35], kojoj je cilj pružiti platformski neutralnu, portabilnu, višekorisničku i sigurnu osnovu za razvoj distribuirani programskih aplikacija koja se zasniva na Java programskom jeziku.

2.3 Struktura i strukturiranje proizvoda

Iz teoretskih osnova prikazanih ukratko u prethodnim poglavljima, može se zaključiti da se hijerarhijska struktura komponenti proizvoda treba razmatrati kao jedno od najvažnijih svojstva proizvoda te poveznice većine nositelja informacija koje se razmjenjuju u procesu razvoja proizvoda. Pojam "hijerarhijska struktura" dolazi iz Teorije tehničkih sustava [17], te se definira kao karakteristika koja opisuje ukupnost sustava, odnosno elemente koji čine sustav i relacije između njih [4]. Već je rečeno u prethodnim poglavljima da pri razvoju proizvoda možemo razmatrati

različite tipove hijerarhijskih struktura koje opisuju s jedne strane komponente i relacije među njima, a s druge strane funkcije pojedinih komponenti, strukturu organa i sl.

U literaturi se također može pronaći stajalište prema kojem se proizvod može prikazati apstraktnom strukturom [Slika 2.7] koju čine objekti strukture i relacije među njima [36]. Glavni objekti takve apstraktne strukture su komponente proizvoda (dijelovi i sklopovi). Uz fizičke komponente, objekti mogu biti i funkcije, zahtjevi i stanja. Glavninu relacija čine agregacija i generalizacija. Osim njih se u apstraktnoj strukturi mogu pojaviti različite posljedične relacije (npr. tijek sile, ograničenja), te proizvoljne i zasebno definirane relacije. Jasno je da je gore navedeni skup objekata i relacija samo prvi korak za kombiniranje osnovnih elemenata u sveobuhvatnu apstraktnu strukturu proizvoda. Za potpunu definiciju proizvoda je potrebno proširenje tog skupa. Način na koji bi se takva apstraktna struktura mogla kreirati je definiranje pojedinih različitih hijerarhijskih struktura za funkcije, fizičke komponente i sl. Potpuni opis proizvoda dobio bi se integracijom elemenata pojedinih struktura u sveobuhvatnu hijerarhijsku strukturu. Ovakav sveobuhvatni opis proizvoda, još nije realiziran, jer postoji mnogo otvorenih pitanja o vrstama veza i pravilima prema kojim se pojedine strukture i njihovi elementi mogu međusobno kombinirati.



Slika 2.7: Apstraktna struktura proizvoda [36]

Inženjer tijekom sinteze proizvoda primjenjuje različite poglede na njegovu strukturu. Iz te tvrdnje proizlazi činjenica se proces konstruiranja može promatrati kao aktivnost strukturiranja proizvoda [37]. Istraživanja koja se danas provode u području strukturiranja proizvoda donose nam dvije različite interpretacije pojma strukture proizvoda [38]. Prva interpretacija promatra proces konstruiranja kao proces kreiranja podataka o proizvodu, tj. strukture proizvoda, koja je poveznica nositelja informacija. Ova interpretacija polazi od definiranja strukture kao skupa elemenata i skupa relacija od kojih se sustav sastoji, odnosno karakteristika sustava. Strukturiranje proizvoda je prema toj interpretaciji aktivnost kreiranja strukture, odnosno proizvoda koji ima strukturu. Druga interpretacija strukture proizvoda bavi se upravljanjem informacijama, a ne na procesima kreiranja informacija. Informacije

o proizvodu su prema toj interpretaciji zajedničko ime za sve podatke koji su vezani uz proizvod i podatke vezane uz proces konstruiranja. Druga interpretacija je usko vezana uz definiranje korisnika informacija i njihovih potreba za informacijama. U poglavljima koja slijede, dati će se osvrt na obje prethodno navedene interpretacije, uz napomenu da se one međusobno isprepliću.

2.3.1 Struktura proizvoda s gledišta procesa konstruiranja

U ovoj interpretaciji, struktura proizvoda se definira kroz elemente i relacije od kojih se proizvod sastoji, a strukturiranje se definira kao aktivnost generiranja tih elementa i relacija koji čine proizvod. Osnova ove interpretacije su razmatranja vezana uz životni vijek proizvoda, s posebnim osvrtom na razvojnu fazu. "Jezik" kojim se konstruktor koristi u modeliranju proizvoda, sastoji se od principa rješenja, funkcija, dijelova, hijerarhijskih relacija itd. U pregledu istraživanja iz ovog područja mogu se definirati slijedeći pravci:

- *Definiranje modela strukture proizvoda* – definiranje "jezika" koji konstruktor koristi u za opisivanje proizvoda procesu konstruiranja.
- *Modeliranje procesa strukturiranja* – definiranje modela konstrukcijskog procesa odnosno strukturiranja proizvoda.
- *Razvoj podrške konstrukcijskom procesu* – podrška konstruktoru u kreiranju proizvoda i donošenju ključnih odluka u tom procesu.

Definiranje modela strukture proizvoda

Model strukture proizvoda može se definirati kao opis proizvoda ovisan o točki gledanja, a koji pokazuje elemente od kojih se proizvod sastoji te njihove međusobne relacije. U Teoriji domena [21] *Andreasen* definira četiri različita područja koja koristi inženjer za modeliranje proizvoda: proces, funkcija, organi i dijelovi. *Erens* [39] razlikuje: područje funkcija, tehnološko područje i područje komponenti. U svakom od tih područja, struktura proizvoda se definira na različitim nivoima apstrakcije proizvoda te svaki sudionik u razvoju proizvoda može imati svoj pogled na strukturu istog proizvoda. Možemo izdvojiti dva tipa strukture, hijerarhijska i arhitekturna [38]. Hijerarhijska opisuje vezu između elemenata, bilo da je riječ o hijerarhiji funkcija ili komponenti. Arhitekturna struktura opisuje sučelja između elemenata, ulazno/izlazne relacije između funkcija ili veza između komponenata. Hansen [40] u svojem radu opisuje ostale tipove relacija vezane uz: sintezu proizvoda, vremenske relacije, kontrolne relacije itd.

Modeliranje procesa strukturiranja

Strukturiranje proizvoda je koncentrirano na kreiranje strukture proizvoda kao integralnog dijela konstrukcijskog procesa. U definiranju strukture sinteza proizvoda je ekvivalentna kreiranju struktura u različitim područjima [41]. Aspekti koji se najčešće razmatraju u istraživanjima mogu se podijeliti u četiri grupe:

- *Funkcijski orijentirana dekompozicija mehaničkih sustava* – počevši s

funkcijom, sintetiziraju se rješenja, te izabire najbolje. Rješenja pojedinih podfunkcija moraju se uklopiti u konačno rješenje [40].

- *Konfiguracijsko konstruiranje* – do rješenja konstrukcijskog problema se dolazi kombiniranjem predefiniranih elemenata na predefinirane načine. Naročito može doći do značaja u prodaji i procesiranju narudžbi, gdje se na temelju zahtjeva naručitelja sintetizira varijanta proizvoda specifična za pojedinog naručitelja [42].
- *Strukturiranje orijentirano životnom vijeku proizvoda* – pri kreiranju strukture komponenti proizvoda, zahtjevi različitih faza životnog vijeka moraju biti uzeti u obzir. Životnom ciklusu orijentirano strukturiranje zahtijeva integrirani razvoj proizvoda i procesa proizvodnje [41].

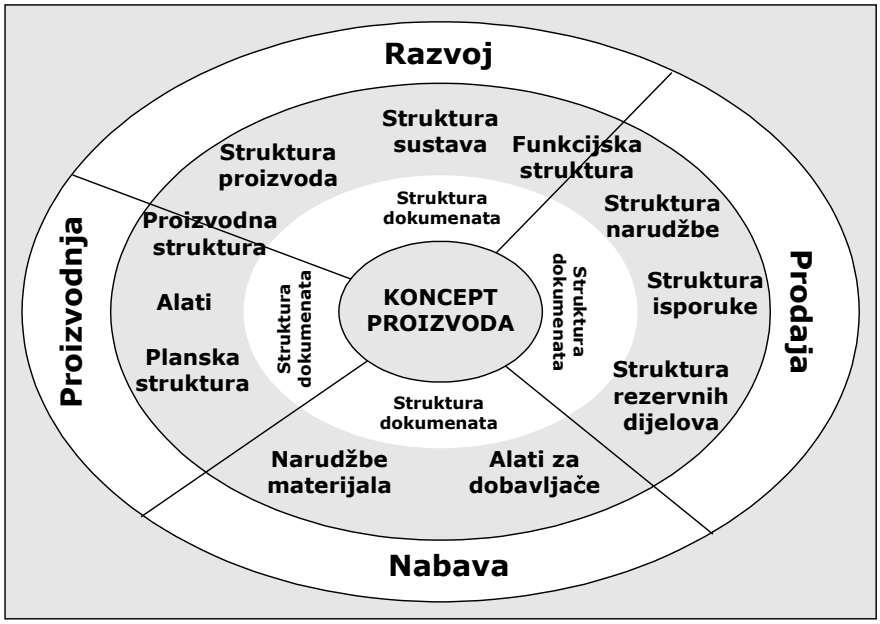
Razvoj podrške konstrukcijskom procesu (strukturiranju)

Fokusira se uglavnom na podršci u donošenju odluka opisivanjem metoda za kreiranje komponentne arhitekture, razmatrajući i tehnički i ekonomski aspekt. Veliku ulogu u razvoju podrške imaju računalni alati za podršku konstrukcijskom procesu [43]. *Lanner* je tako predstavio pristup kreiranju arhitekture proizvoda nazvan, PAD (*eng. Product Architecture Design*) metodologija. Prema toj definiciji, arhitektura proizvoda definira preslikavanje između fizičkih i funkcionalnih elemenata proizvoda. Najprije je potrebno kreirati strukturu organa, nakon čega se korištenjem matričnog zapisa definiraju relacije između organa i vrednuju tehničkim i ekonomskim kriterijima. Rezultati tog vrednovanja definiraju koji organi moraju biti zajedno kombinirani u fizičke elemente.

Erixon predlaže metodu za definiranje modula MFD (*eng. Modular Function Deployment*). MFD započinje sa prevođenjem zahtjeva naručitelja u listu tehničkih zahtjeva, nakon čega slijedi funkcijska dekompozicija i određivanje najboljeg koncepta rješenja. Idući korak je kreiranje matrice indikacije modula, koja pokazuje za svaki tehnički princip kolika je potreba za kreiranjem modula. Na kraju, kad su evaluirani prijedlozi o modulima, za svaki od predloženih modula provodi se DFX (*eng. Design for X*) analiza. *Hansen* definira zahtjeve za računalne sustave koji pružaju podršku procesu konfiguriranja.

2.3.2 Struktura proizvoda s gledišta upravljanja informacijama o proizvodu

U interpretacijama s gledišta upravljanja informacijama, struktura proizvoda se promatra kao poveznica ili kostur nositelja podataka o proizvodu. Većina informacija se kreira tijekom konstrukcijskog procesa. Korisnici tih informacija su sve aktivnosti u životnom ciklusu proizvoda: prodaja, planiranje proizvodnje i kontrola, održavanje, servisi itd. [Slika 2.8]. Jedan od glavnih zadataka koji se pojavljuje prilikom istraživanja problematike upravljanja podacima je dokumentiranje inženjerske aktivnosti sa aspekta promjena koje nastaju na strukturi proizvoda.



Slika 2.8: Pogledi na proizvod s različitih perspektiva

Istraživanja u ovom području se razlikuju od istraživanja vezanih iz prvu interpretaciju zbog:

- naglasak je na upravljanju informacijama, a ne na procesima u kojim se kreiraju i koriste informacije,
- obuhvaćene su informacije o proizvodu koje se koriste kroz sve faze životnog vijeka proizvoda, a ne samo u razvojnoj fazi,
- korisnici informacija nisu samo konstruktori već i ljudi iz prodaje, management poduzeća, planiranje proizvodnje, proizvodnja.

U ovoj interpretaciji mogu se razlikovati dva temeljna područja istraživanja:

- *Istraživanje nositelja informacija o proizvodu* – koncentrira se na definiranje informacija koji se trebaju bilježiti u promatranom sustavu. Također se definira i način korištenja tih informacija.
- *Razvijanje podrške za procesiranje informacija* – bilježenje, ponovo korištenje, prikaz, transformiranje itd.

Definiranje nositelja podatka o proizvodu

Osnovno pitanje koje se postavlja pri definiranju granica unutar kojih se promatraju informacije o proizvodu je o klasama podataka koje se treba pratiti. Klasificiranje podataka o proizvodu mora se obavljati u sprezi sa svrhom u kojoj će se oni upotrebljavati. *Blessing* [43] smatra da u tehničkom smislu nije teško pokrivati široki raspon podataka, već je limitirajući faktor praktične prirode. Za definiranje okvira potrebno je dobro razumijevanje potrebe za podacima u procesima koji ih koriste.

Pri tome se u relevantnoj literaturi razmatraju dvije osnovne kategorije podataka:

- *Podaci vezani uz proizvod* – uključuje opis proizvoda za različite poglede, verzije i proizvode [45], standardizirana rješenja [42], podatke vezane uz

životni ciklus proizvoda, specifikacije, proizvodne procese….

- *Podaci vezani uz proces konstruiranja* – uključuje aktivnosti, argumente, ključne odluke, povijest proizvoda [43], status proizvoda [45], općenite podatke o procesu, alate planiranja konstrukcijskog procesa, status projekta itd.

Jedan od opisa koji se susreće u literaturi, prema van *den Hameru* [45], osniva se na identifikaciji pet glavnih karakteristika koje su vezane istovremeno uz upravljanje podacima i upravljanje procesom konstruiranja. To su podaci vezeni uz verzije i varijante komponenti, hijerarhiju komponenti, različite poglede na hijerarhijsku strukturu i status pojedinih komponenti. Verzije nastaju kao rezultat modifikacija, te su vezane uz iterativnost procesa konstruiranja. Pogledi odgovaraju različitim aspektima istog proizvoda, te pomažu u redukciji kompleksnosti konstruktorskih zadataka. Hijerarhija objašnjava gradivnu strukturu. Status odgovara verifikaciji, potvrđivanju i odobravanju u procesu konstruiranja. Varijante nastaju prilikom razvoja familija proizvoda. Također važan čimbenik u području definiranje nositelja podataka o proizvodu su i istraživanja informacijskog toka u smislu optimiranja razvojnog procesa proizvoda, te pronalaženja pretpostavki za implementaciju alata za podršku pri upravljanju podacima [43].

Razvijanje podrške za procesiranje podataka o proizvodu

Pojam procesiranja podataka vezan je i uz bilježenje podataka i prezentiranje podataka korisnicima. Zahtijevana funkcionalnost od sustava za podršku uključuje [46]:

- lako bilježenje podataka, indeksiranje i spremanje,
- brzo pretraživanje i ponovna upotreba podataka,
- prezentiranje podataka korisnicima,
- transformacija podataka između različitih modela/pogleda,
- upravljanje konzistentnošću.

2.4 Utjecaj na rad

Ovim glavom je definirano područje istraživanja, odnosno teoretske osnove koje su poslužile kao osnova istraživanja. Uočena je uloga hijerarhijske strukture komponenti proizvoda kao glavne poveznice nositelja informacija o proizvodu, te je objašnjen pojam strukturiranja i strukture proizvoda u kontekstu današnjih istraživanja. Saznanja iz područja navedenih u poglavljima ove glave, iskorištena su kao osnova za analizu problematike razmjene i upravljanja informacijama o proizvodu koja je opisana u trećoj glavi, te za definiranje metodologije implementiranja informacijskog modela podataka o proizvodu definiranog prema međunarodnom ISO 10303 standardom.

3

RAZMJENA I UPRAVLJANJE INFORMACIJAMA O PROIZVODU

U ovoj je glavi analizirana problematika razmjene i upravljanja informacijama o proizvodu, sa svrhom razumijevanja, interpretacije i pravilne implementacije informacijskog modela definiranog prema ISO STEP 10303 standardu. Nakon uvodnog dijela glave u kojem se općenito govori o pojmu varijantnosti informacija o proizvodu, veći dio teksta je posvećen analizi dijelova koje bi trebao sadržavati metamodel informacija koje se razmjenjuju u razvojnoj fazi proizvoda. Na kraju treće glave definirane su potrebne funkcije sustava.

3.1 Varijantnost informacija o proizvodu

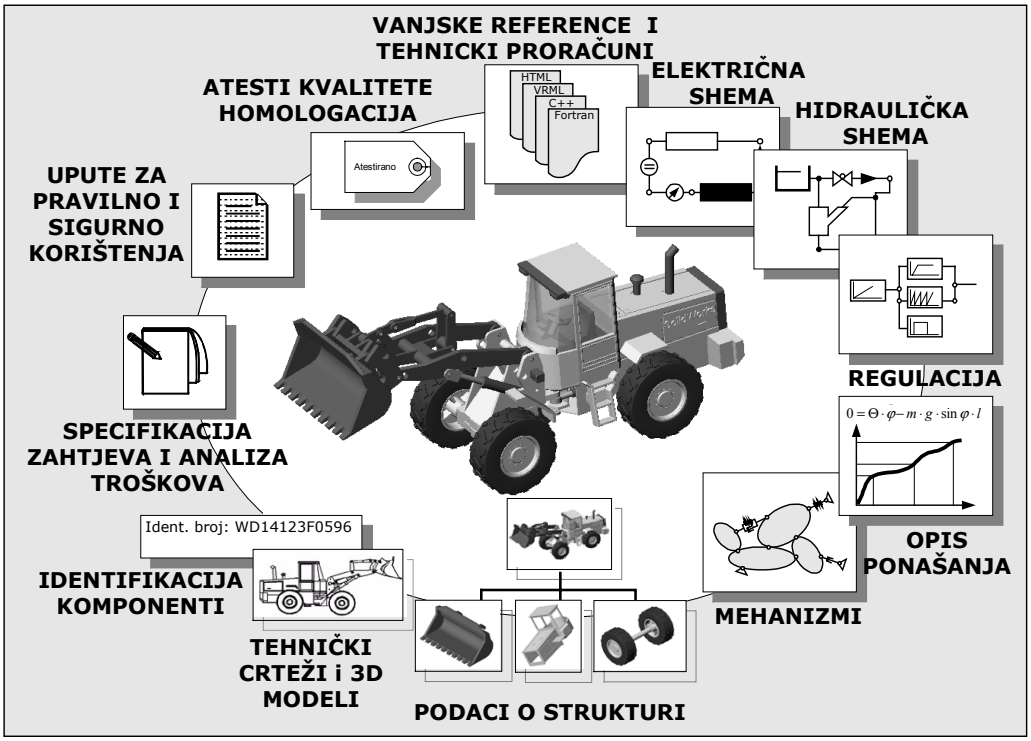
Na početku ove glave razmatranja ćemo nastaviti analizom raznolikosti informacija o proizvodu koje se razmjenjuju u fazi razvoja proizvoda. Raznolikost informacija najbolje je ilustrirati pomoću liste koja definira samo mali dio različitih informacija vezanih uz proizvod [47], [Slika 3.1]:

- Konstrukcija proizvoda glavni je izvor informacija o proizvodu. Za svaki proizvod to su podaci o zahtjevima koji dolaze iz analize tržišta ili od naručitelja, 3D modeli, crteži, sastavnice, proračuni, rezultati analiza, video zapisi itd.
- Odjel marketinga i prodaje priprema kataloge proizvoda, prodajne brošure, i ostali propagandni materijal vezan uz proizvod.
- Odjel prodaje koristi informacije o proizvodu prilikom kreiranja ponuda, te za to često treba podatke o rasporedu proizvodnje i o raspoloživim kapacitetima.

U slučaju konfigurabilnih proizvoda, prodaja također može kreirati inicijalne konfiguracije proizvoda prema zahtjevima naručitelja.

- Priprema, planiranje proizvodnje te proizvodnja definiraju i koriste informacije vezane uz izradu pojedinog proizvoda. To su radionički crteži, NC programi, planovi proizvodnje, tehnološke liste, pravilnici o načinu kontrole.
- Održavanje koristi informacije o proizvodu za zamjenu i popravak oštećenih dijelova proizvoda.

Iz gore navedene liste možemo zaključiti da informacije o proizvodu uključuju podatke vezane uz definiranje proizvoda, ali i podatke vezane uz procese koji podatke o proizvodu specificiraju, kreiraju i koriste. Podaci o proizvodu mogu se pronaći na različitim mjestima unutar organizacije poduzeća, te mogu biti pohranjeni na različitim medijima. U najvećem broju poduzeća, većina podataka još je uvijek pohranjena na tradicionalnom mediju, a to je papir. Sve više podataka pohranjeno je u digitalnom obliku, te su dostupni u lokalnoj ili globalnoj mreži poduzeća, dok se dio može pronaći pohranjen na mikrofilmovima. Prisutna šarolikost medija određuje i različito upravljanje podacima, a poseban problem za razmjenu informacija su mnogostruki formati podatka (numerički, grafički, alfanumerički) pri čijoj razmjeni često dolazi do gubitaka podataka. Također je važno naglasiti da je područje korištenja podataka unutar poduzeća, ali i izvan njega (razmjena podataka s dobavljačima ili kupcima).



Slika 3.1: Dio različitih informacija o proizvodu

Mnoge informacije prikazane su na različite načine u različitim dijelovima poduzeća što često može dovesti do konfliktnih situacija. Korisnici mogu imati kopije istih

podataka koje je ponekad teško otkriti obzirom na moguće korištenje različite terminologije za njihovo opisivanje, te se iz tog razloga javlja mogućnost redundancije. Vijek trajanja podatka zahtjeva zasebnu pozornost. Uobičajeno je čuvanje podataka na tradicionalni način, čuvanjem dokumenata u arhivama, ali također je nezaobilazno prisutna pohrana i u elektronskom obliku. Posebno je važno stoga predvidjeti mogućnost podrške za manipuliranje takvim, starim, arhiviranim podacima.

Prema [48] možemo zaključiti ovoj kratki pregled raznolikosti informacija o proizvodu definiranjem tipova izvora podatka o proizvodu, prema načinu na koji se njima upravlja u procesu razvoja proizvoda:

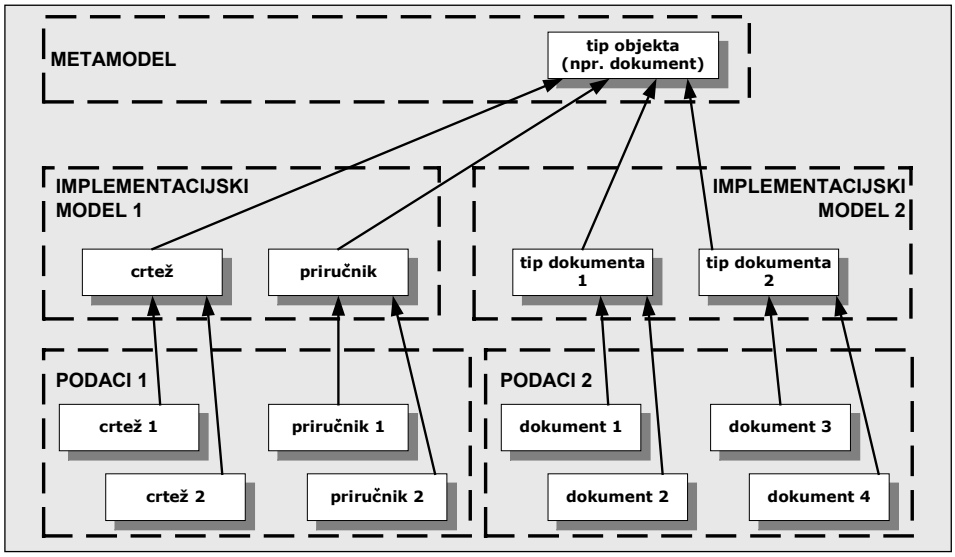
- *tip1 - 'design rationale'*: podaci koji definiraju argumente i razloge za pojedine konstruktorske odluke,
- *tip 2 - 'design dossier'*: podaci koji opisuju ili definiraju proizvod i/ili proces,
- *tip 3 - 'library information'*: podaci koji uključuju znanja eksperata, kataloge ili baze znanja,
- *tip 4 - 'project management information'*: podaci relevantni za voditelje projekata i uključuju informacije o troškovima, potrošenom vremenu, podjeli poslova, stanju projekta.

3.2 Dijelovi metamodela informacija u procesu razvoja proizvoda

U ovom je radu informacijski model podataka definiran kao osnova sustava za razmjenu i upravljanje informacijama o proizvodu, odnosno predstavlja jezgru oko koje će sustav biti izgrađen. Na osnovu pregleda literature i teoretskih osnova (druga glava rada) te dosadašnjeg iskustva autora dokumentiranog u [49], [50], [51], [52], provedena je analiza čiji je cilj bio bolje razumijevanje problema upravljanja i razmjene informacija u procesu razvoja proizvoda. Na temelju analize definirani su dijelovi 'metamodela' informacija koji je u radu poslužio kao osnova za razumijevanje i pravilno implementiranje informacijskog modela definiranog prema ISO STEP 10303 međunarodnom standardu za razmjenu podataka o proizvodu (detaljnije opisan u četvrtoj glavi).

Metamodel informacija koji se promatra u ovom radu je statički, iz razloga što opisuje kojim vrstama informacija se upravlja u sustavu, bez detaljnog opisa operacija i procesa koji manipuliraju tim informacijama. To konkretno znači da se metamodelom informacija ne definira npr. kojim se konkretnim tipovima dokumenata upravlja u predloženom ustavu, već se općenito definira potreba za postojanjem klasifikacija različitih tipova dokumenta. Implementacijski modeli koji su specijalizacija metamodela primijenjena na realnu okolinu modeliraju specifične potrebe prilikom stvarne implementacije. [Slika 3.2] ilustrira vezu između metamodela i implementacijskih modela. Prema njoj, metamodel definira općeniti tip

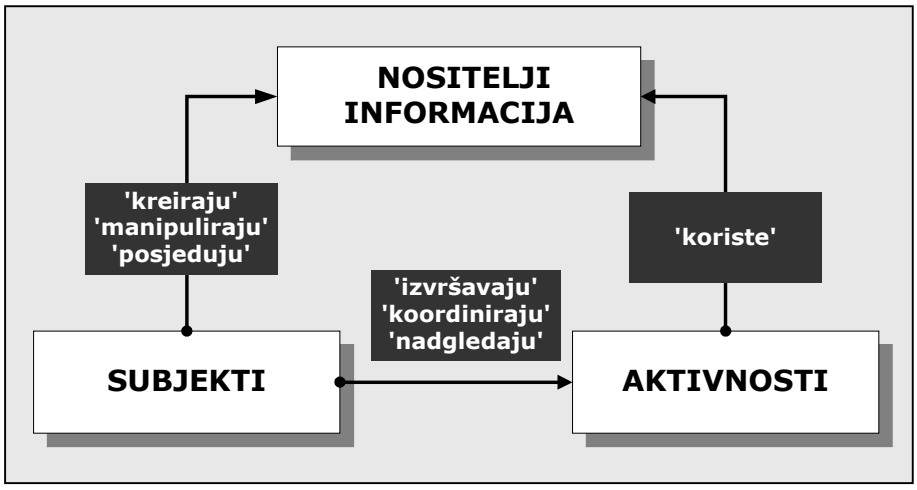
objekta kojeg opisuje, u konkretnom slučaju može se kao primjer uzeti dokument. Na temelju tako definiranog tipa, prilikom implementacije se implementacijskim modelom definiraju specifični tipovi dokumenta koji postoje u realnom slučaju (npr. crteži i uputstva za upotrebu). To znači da sustav koji je realiziran na temelju tako definiranog implementacijskog modela manipulira crtežima i priručnicima kao instancama objekta definiranog u metamodelu. Neki drugi implementacijski model može definirati drukčije tipove dokumenta po kriteriju druge realne implementacije.



Slika 3.2: Veza metamodela i implementacijskih modela

U okviru razmatranja metamodela informacija kojima se upravlja u razvojnoj fazi proizvoda definirani su njegovi osnovni dijelovi [48], [Slika 3.3]:

- informacije vezane uz nositelje podataka o proizvodu,
- informacije vezane uz subjekte koji upravljaju podacima o proizvodu,
- informacije vezane uz aktivnosti u kojima subjekti koje koriste podatke o proizvodu.



Slika 3.3: Dijelovi metamodela informacija kojima se upravlja u procesu razvoja proizvoda

U idućim poglavljima detaljnije će se analizirati svaki od navedenih elementa, te relacije koje postoje među njima, ali i unutar svakog od njih.

3.2.1 Nositelji informacija o proizvodu

U okvirima metamodela informacija kojima se upravlja u procesu razvoja proizvoda, informacije koje su vezane uz nositelje podataka o proizvodu u razvojnoj fazi proizvoda definirane su fizičkim komponentama proizvoda i strukturiranim dokumentima. Uloga ovih informacija je slijedeća:

- određivanje referentne terminologije,
- identifikacija fizičkih komponenti proizvoda i dokumenata,
- opisivanje svojstava fizičkih komponenti i dokumenta,
- definiranje osnove za katalogizaciju i klasifikaciju informacija o proizvodu.

Sa svrhom daljnje analize nositelja informacija o proizvodu s gledišta upravljanja i razmjene mogu se izdvojiti njihove glavne karakteristike: hijerarhija nositelja, verzije nositelja i njihov status. U daljnjem tekstu će se detaljnije analizirati svaka od tih karakteristika.

Hijerarhija

U ovom se radu pod izrazom "proizvod" promatra hijerarhijska struktura sklopova, podsklopova i jednostavnih dijelova, koja opisuje dekompoziciju složene cjeline u manje složene sastavne komponente[1]. Općenito govoreći, hijerarhiju proizvoda je moguće definirati relacijom koja kaže da komponenta sadrži drugu komponentu 'kao sastavni dio' [47]. Taj proces dekompozicije moguće je ponavljati do razine do koje je potrebno omogućiti upravljanje podacima. Npr. za komponente koje se naručuju od vanjskih dobavljača nije potrebno raditi dekompoziciju, nego se njima upravlja kao nerastavljivim. Hijerarhijska struktura proizvoda je kao što je spomenuto u prethodnom poglavlju, poveznica nositelja potrebnih informacija. Dva su osnovna pristupa kojim korisnici dolaze do traženih podataka: direktan, preko pretraživanja komponenti kroz vrijednosti njihovih atributa, ili indirektan preko hijerarhijske strukture proizvoda [54]. Također je među komponentama moguće definirati i ostale tipove relacija što je objašnjeno u drugoj glavi, a što omogućuje da pojedini specijalisti vide strukturu komponenti iz vlastite perspektive gledanja.

Najjednostavniji pristup upravljanju različitim pogledima na strukturu proizvoda je onaj koji definira postojanje jedinstvene hijerarhijske strukture, a svaka komponenta može biti vidljiva u više različitih pogleda na strukturu [45]. Prethodno navedeni

[1] Komponenta će se u nastavku glave koristiti kao pojam koji općenito opisuje element proizvoda bilo koje složenosti, sklop, podsklop ili nerastavljivi dio.

pristup nije najsretnije rješenje, jer se svi pogledi u takvom slučaju osnivaju na zajedničkoj osnovnoj strukturi, te se filtriranjem određuju komponente ili dijelovi podataka koji pripadaju određenom pogledu. Općenitije gledano, može se reći da proizvodi mogu imati mnogostruke različite strukture, koje se ne mogu kreirati filtriranjem iz jedne osnovne, ali tada je potrebno definirati mehanizme koji će povezivati te različite strukture.

Za svaku pojedinu komponentu, postoji čitav niz atributa koji je opisuju. Da bi se tim mnoštvom podataka moglo lako i brzo upravljati potrebno je pri implementaciji osigurati identifikaciju komponenti i klasifikaciju podataka. Podaci koji nose komponente proizvoda klasificiraju se na razne načine ovisno o potrebama implementacije. Vrlo važna činjenica je saznanje da pojedina komponenta definirana s jasnom ulogom i potrebnim atributima, može biti iskorištena kao sastavni dio u više različitih komponenti ili u različitim projektima. Atributi, koji predstavljaju podatke koje svaka komponenta nosi, variraju od opisa svojstava komponente, kao što je npr. materijal od kojeg je napravljena ili kvalitete površinske obrade, do kvantitavnih informacija, kao npr. broja istih komponenti koji se nalazi u nekoj nadređenoj komponenti.

Osim fizičkih komponenti kao nositelji podataka o proizvodu definirane su komponente-dokumenti koje se kreiraju te se njima upravlja pomoću različitih alata u procesu razvoja proizvoda. Potrebno je reći da je velika većina dokumenta vezana uz hijerarhijsku strukturu proizvoda. Kao tipični primjeri dokumenata možemo spomenuti tehničku dokumentaciju kreiranu CAD alatima, sastavnice proizvoda ili priručnike i uputstva za konstruiranje koji su napravljeni u aplikacijama za obradu teksta. Dokumenti ne moraju biti u papirnatom obliku. Npr. 3D model može se također promatrati kao dokument vezan uz proizvod. Relacije koje se pojavljuju između komponenti proizvoda i dokumenata su mnogostruke. To znači da npr. čitav niz sličnih proizvoda, može imati zajednički priručnik za uporabu, jedna komponenta može imati vezu prema više dokumenta ili mogu postojati dokumenti koji nisu vezani ni uz jedan proizvod ili njegovu komponentu. Dok u svakodnevnom radu dokument obično znači datoteku, koja može biti pregledavana ili mijenjana s odgovarajućim alatom (aplikacijom), u sustavu za podršku razmjenu i upravljanju s podacima o proizvodu dokument se može promatrati i kao apstraktni objekt. Bitno je naglasiti da se tradicionalni sustavi za podršku upravljanju dokumentima uglavnom upravljaju samo verzijama, a ne i sadržajem svakog pojedinog dokumenta [47]. Uobičajeno je da je datoteka najmanja jedinica kojom se upravlja u takvim sustavu. Međutim i takva datoteka ima svoju internu strukturu. Dokument dakle može kao i proizvod biti hijerarhijska struktura različitih drugih dokumenta i ostalih entiteta kao što su slike, linkovi i sl.

Činjenica je da tradicionalni sustavi "ne razumiju" strukturu dokumenta, što nije dovoljno, ako su dokumenti napravljeni na način da su njihove komponente kreirane u drugim dokumentima (npr. jedna slika može biti prisutna u više dokumenta koji sadrže vezu na njezinu originalnu datoteku). Takve veze bi trebala biti poznate

sustavu te na taj način onemogućiti korisniku brisanje nečeg što je uključeno u drugi dokument. Problem možemo riješiti tako da se i dokumentima pristupa kao strukturiranoj formi, kao i komponentama fizičke hijerarhije proizvoda. Općenito govoreći strukturalne relacije među dokumentima moraju se koristiti kada je važno upravljati strukturom dokumenata. U svim ostalim slučajevima, komponente proizvoda moraju imati relaciju asocijativnosti prema datotekama, odnosno relaciju koja govori koja datoteka s podacima pripada određenoj komponenti proizvoda.

Odgovarajućim pretraživanjem hijerarhijskog stabla komponenti i strukturiranih dokumenata proizvoda omogućeno je dobivanje podataka na različitim razinama, koji su vezani za pojedine dijelove, za pojedini sklop ili cijeli proizvod. Potrebno je također naglasiti da se u ovom radu ne razmatraju podaci koji su vezani uz geometriju pojedine komponente ili sklopa, što znači da se informacije kojima se upravlja opisuju značajke koji vrijede za cijelu komponentu.

Verzije (revizije i varijante)

Konstruktori mijenjaju podatke o proizvodu u više koraka [45]. Svaki korak rezultira novom verzijom nositelja podataka o proizvodu te se tako kreira informacija o evoluciji proizvoda. Dvije su vrste verzija koje su promatrane u okviru metamodela. *Revizije* odgovaraju koracima koje konstruktori poduzimaju da bi ispravili nedostatke i pogreške ili optimizirali konstrukciju. S druge strane komponenta ili dokumenti mogu imati nekoliko alternativa, koje se tada zovu *varijante*. Npr. priručnik za uporabu nekog proizvoda mora biti dostupan na više jezika. Razliku između revizija i varijanti može se sažeti u slijedećem: nova revizija uvijek zamjenjuje staru, dok su sve varijante dostupne u isto vrijeme. Pri tome ne smijemo zanemariti varijante koje se mogu pojaviti u više razina. Npr. varijantni proizvod s tri različite varijante, te za svaku varijantu proizvoda postoje uputstva na četiri jezika.

Bilo da se radi o reviziji ili varijanti, nova verzija nastaje kad konstruktori žele napraviti neku promjenu na fizičkim komponentama proizvoda ili dokumentima. Pogledamo li analogiju sa računalnim sustavima, nova verzija datoteke uvijek jednostavno "prepiše" postojeću. "Prepisivanje" postojeće verzije u konstrukcijskom procesu može biti problem, jer prethodna verzija ponekad može značiti korak unatrag, koji je uvijek prisutan u iterativnosti konstrukcijskog procesa. Obzirom da konstruktor mora imati mogućnost povratka na prethodnu vrijednost podatka, potrebno je da metamodelom jasno razlikovati vrste verzija.

Tehnički dokument koji opisuje fizičku komponentu proizvoda, često je proizveden od strane jednog čovjeka ili male grupe ljudi, te kad je spreman za korištenje, objavljuje se i koristi od strane ostalih korisnika. Obično, dokument doživljava promjene veliki broj puta, uz neprestane kvalitativne pomake. Objavljivanje novih revizija je obično povezano sa procedurama koje osiguravaju kvalitetu podataka (provjera, kontrola, odobravanje). Stare verzije je također potrebno čuvati, za slučaj da je potrebno znati kako su podaci izgledali prije, kao referencu na temelju koje se derivira nova revizija.

Kad se počnu pratiti verzije komponenti u hijerarhijskoj strukturi, potrebno je znati da nove verzije komponenti mogu imati drukčiju kompoziciju od svojeg prethodnika. To se događa zbog mogućnosti da komponente ili dokumenti koje nam inicijalno nisu trebale, budu aktivirane u novoj verziji ili se inicijalno odabrane komponente i dokumenti mogu zamijeniti drugim. Praćenje takvih promjena verzija u hijerarhijskoj strukturi može se ostvariti na dva načina: statički i dinamički. Prednost statičkog načina je u tome što je struktura pojedine verzije komponente ili dokumenta 100% fiksna, bez potrebe da se posebno bilježi da li pojedina komponenta pripada određenoj verziji ili ne. Nedostatak ovog pristupa je u tome što i najmanja promjena uzrokuje potrebu za kreiranjem nove verzije cjelokupnog proizvoda (komponenta ima referencu na određenu verziju druge komponente). Dinamički pristup različit je u tome što pojedina verzija komponente nema referencu na neku određenu verziju druge komponente koja joj je u hijerarhiji podređena, nego na komponentu. Kod ovog pristupa potrebno je definirati pravila ili algoritme koji omogućuju određivanje koja od postojećih verzija je relevantna. Najčešće je to definirano određivanjem zadnje kreirane verzije kao referentne, ali može biti riješeno i korištenjem nekog filtra (prema datumu ili opisu). Prednost je dinamičkog pristupa u tome što se minorne promjene na najnižoj razini mogu obaviti bez utjecaja na gornje razine. Nedostatak dinamičkog pristupa je taj što se ne može točno znati kako je izgledala struktura neka stare verzije, o čemu je potrebno voditi računa.

Status:

U konstruktorskoj okolini, informacija koja je stabilna, točna i provjerena tretira se drugačije od informacije koja je privremena, nepotvrđena i podložna promjenama. Jedan od primarnih ciljeva konstrukcijskog procesa je pružiti sve potrebne informacije koja opisuju proizvod - potpunost, ali isto tako osigurati informacije koje odgovaraju upotrebi – kvalitetu. Promjena u statusu informacije ne odgovara promjeni informacije same po sebi, nego samo prikazuje činjenicu da je netko u organizacijskoj strukturi odlučio da informacija zadovoljava ili ne zadovoljava određene kriterije. Praćenje statusa se koristi da bi se odredilo što se može ili treba učiniti s komponentama proizvoda koji se konstruira. Na primjer: određeno konstrukcijsko rješenje može dobiti drukčiji status, ako razvojni tim odluči da je dovoljno dobro da postane dio internih standarda gotovih rješenja.

U okviru statusa potrebno je definirati način prikazivanja informacije, odnosno odrediti da li je informacija javna, vezana uz specifični projekt, ili se tiče samo osobe koja je njezin vlasnik, odnosno kreator informacije. To nas u ovoj analizi dovodi do potrebe za direktnom vezom između statusa informacija i korisnika tih informacija. Potrebno je voditi računa o tome da podaci koji se nalaze u konstruktorskom privatnom radnom prostoru, nisu vidljivi za ostale, tako dugo dok to on ne odredi. Upravljanje podacima uključuje istovremeno upravljanje svim prethodno navedenim karakteristikama nositelja podataka. Sve su karakteristike relevantne za praktičnu primjenu uz potrebu za definiranjem prioriteta upravljanja, što ovisi o pojedinoj implementaciji.

3.2.2 Subjekti

U okvirima metamodela informacija kojima se upravlja u procesu razvoja proizvoda, informacije vezane uz subjekte koji upravljaju podacima o proizvodu definiraju organizaciju radnih zadataka i korisnika unutar razvojnih timova. Uloga ovih informacija je slijedeća:

- definiranje osnove za identifikaciju korisnika i projekata u razvojnoj fazi proizvoda,
- definiranje uloga i ovlasti svakog korisnika unutar organizacije i projekta.

Sa svrhom daljnje analize subjekata s gledišta upravljanja i razmjene informacijama može se promatrati upravljanje korisnicima i projektima. U daljnjem tekstu će se detaljnije analizirati svaka od tih grupa.

Korisnici

Korisnici kreiraju, mijenjaju, koriste i brišu podatke tijekom razvoja proizvoda. Dva su pogleda s kojih možemo promatrati i analizirati korisnike. Prvi pogled je iz perspektive organizacijske strukture razvojnog tima. Unutar razvojnog tima, svaki pojedini korisnik ima određenu ulogu, odnosno točno određeni položaj u hijerarhiji obavljanja poslova. Ovisno o ulozi koja mu je dodijeljena, variraju njegova prava i obveze, odnosno definirana je točna pozicija korisnika unutar toka informacija u procesu konstruiranja. Podaci koji proizlaze iz ovog aspekta, omogućuju nam točnu identifikaciju svakog korisnika unutar radne grupe što uključuje definiranje njegove uloge u razvojnom procesu.

Drugi pogled je iz perspektive podatka o proizvodu koji se razmjenjuju u procesu razvoja proizvoda. Svaki od tih podatka ima odgovornog korisnika, odnosno poznato je tko je kreirao ili promijenio podatak i točan trenutak kad je podatak kreiran odnosno promijenjen. Vezano uz status podataka, njime je definirano kakav pristup pojedina grupa korisnika ima do podatka, odnosno da li su podatak ili grupa podataka dostupni korisniku samo za pregledavanje ili mu je dozvoljeno izvoditi promjene i brisanje podatka. Na taj se način definiraju ovlasti svakog pojedinog korisnika unutar razvojnog tima. Obzirom na različite uloge i ovlasti korisnika u razvojnom timu moguće je u grubo klasificirati glavne grupe korisnika:

- voditelji projekata,
- konstruktori,
- administratori sustava,
- ostali.

Specifičnu i detaljniju klasifikaciju korisnika potrebno je provesti prilikom realne implementacije ovisno o posebnostima pojedinog razvojnog tima, te je moguće detaljnije grupirati korisnike prema tako definiranim posebnostima unutar ove četiri glavne grupe.

Projekti

Zbog nastojanja za unapređivanjem organizacije rada unutar razvojnog tima potrebno je promatrati i konstrukcijski zadatak (projekt) u informatičkom kontekstu [55]. Svaki projekt integrira slijedeće podatke vezane uz konstrukcijski zadatak: ulazne podatke (zahtjeve), izlazne podatke (karakteristike - dokumentaciju gotovog rješenja), rokove za dovršenje pojedinog dijela ili ukupnog zadatka, troškove i organizacijske podatke koji se prije svega odnose na ljudske resurse odnosno podjelu radnih zadatka unutar razvojne grupe.

Potreba za definiranjem projekta kao zasebnog elementa proizlazi iz potrebe organiziranja rada na razvoju kompleksnih proizvoda. Upravljanje takvim projektima temelji se na raščlambi projekata u manje složene cjeline, koje točno određuju koja grupa ili pojedinac radi na pojedinim komponentama kompleksnog proizvoda. Pripadnost grupe ili pojedinca određenom projektu ili njegovom pojedinom dijelu, može poslužiti za kontrola pristupa do podataka i dokumenta koji su vezani uz pojedini projekt [54].

Ovakva organizacija projekata omogućuje voditeljima projekata ili cijelog razvojnog tima kontroliranje statusa i napretka pojedinih projekta, upravljanje raspoloživim ljudskim resursima i planiranje budućih projekata.

3.2.3 Aktivnosti

U okvirima metamodela informacija kojima se upravlja u procesu razvoja proizvoda, informacije vezane uz aktivnosti u kojima subjekti koje koriste podatke o proizvodu definirane su: upravljanjem hijerarhijskom strukturom proizvoda i dokumenata, upravljanjem promjenama i kontrolom podataka, osiguravanjem konzistentnosti podataka, procedurama odobravanja i autorizacije, dokumentiranjem i upravljanjem tokom procesa konstruiranja te podrškom razvoju konfigurabilnih proizvoda. Uloga ovih informacija je slijedeća:

- kreiranje osnove za određivanje i definiranje u procesa koji koriste podatke u razvojnoj fazi proizvoda,
- definiranje atributa koji opisuju takve procese.

Upravljanje informacijama

Procesi upravljanja informacijama mogu se opisati sa nekoliko fundamentalnih koncepata. Prvi od njih je koncept *transakcije.* Transakcije su općenito gledajući, pojam preuzet iz teorija baza podataka. Pogledamo li ih iz konstruktorske perspektive, npr. kreiranja revizije dokumenta, pretpostavka je da transakcija u kojoj nastaje nova revizija komponente počinje u trenutku kad je kreiran zahtjev za novom revizijom i da završava kad je revizija odobrena i objavljena za korištenje ostalim sudionicima. U postupku kreiranja revizije, ključno je pitanja tko može vidjeti podatke o komponenti i kada. Kod tradicionalnih transakcija u bazama podataka, njihovo izolacijsko svojstvo onemogućava da itko vidi promjenu dok transakcija nad

podacima nije izvršena. Za razliku od toga, u konstruktorskoj okolini je potrebno omogućiti pristup podacima unutar transakcije i prije nego je ona službeno potvrđena odobrenjem nadležnog korisnika. Tri su faze koje pri tome razlikujemo unutar transakcije u konstruktorskoj okolini: prva, transakcija je započela, revizija nije potvrđena što znači da kreator može slobodno modificirati reviziju prema svojim željama; drugo, revizija je potvrđena od strane kreatora, ona se ne može više mijenjati, ali treba biti potvrđena od odgovorne osobe koja mora moći vidjeti promjene i odobriti ih; i treća, revizija je odobrena što znači da je transakcija završena te rezultat postaje dostupan svim ostalim korisnicima.

Drugi osnovni koncept je c*heck-in/out* mehanizam koji se koristi da bi se osiguralo da korisnici uvijek imaju zadnje potvrđeno stanje podataka u svojoj radnoj okolini. Prilikom *check out*-a, korisniku se u prostor njegove radne okoline prebacuje zadnje potvrđeno stanje podataka koji su relevantni za njegov rad. Isto tako nakon što se promjene izvrše i budu potvrđene potrebno je *check in* mehanizmom novo odobreno stanje vratiti iz konstruktorske privatne radne okoline u zajednički prostor.

Koncept *zaključavanja* se koristi da bi se spriječilo mijenjanje istog podatka od strane više korisnika istovremeno. Korisnik mora biti u mogućnosti zaključati ono na čemu radi, te eventualno definirati kakav utjecaj ima promjena koju je izvršio na više nivoe u hijerarhiji.

Tri navedena koncepta trebaju se implementirati u procedurama kojima se kreiraju, koriste, mijenjaju i brišu informacije. Pri tome je potrebno bilježiti odabrani skup aktivnosti, te taj zapis tada predstavlja povijest određenog projekta. Dokumentiranje konstrukcijskog procesa na taj način, omogućuje pohranjivanje informacija o tome kako se proces konstruiranja odvijao, koje su ključne odluke donesene, koje alternative u konstrukciji su razmatrane, koje su odbijene, tko je napravio promjene, kada i na čiji zahtjev.

Proces konfiguriranja

Proizvodnja konfigurabilnih proizvoda, sve je više zanimljiva proizvođačima, zbog jednostavne mogućnosti prilagođavanja proizvoda specifičnim zahtjevima kupaca. Konfigurabilni proizvodi predstavljaju skup različitih, ali vrlo bliskih proizvodnih varijanti, o čemu je već bilo govora na početku glave. Sve moguće varijante opisuju se općenitom hijerarhijskom strukturom. Varijante konfigurabilnih proizvoda se modificiraju prema svakoj pojedinoj narudžbi, na osnovu predefiniranog modela konfigurabilnog proizvoda te pravila za sklapanje konfiguracija [47]. Ulazne informacije u proces konfiguriranja sadržane su listom zahtjeva, koja je formalni prikaz zahtjeva kupaca. Obrađivanje tih zahtjeva na jednom mjestu (obično već u odjelu prodaje) omogućuje početak i inicijalno planiranje proizvoda koji se osnivaju na istoj strukturnoj osnovi. Iza svakog zahtjeva u strukturiranoj listi zahtjeva stoji pojedina varijanta konfigurabilnog proizvoda. Proces konfiguriranja potrebno je kontrolirati konfiguracijskim modelom, koji opisuje sve moguće varijante proizvoda i koji specificira kako pojedina varijanta odgovara pojedinim zahtjevima. Isto tako

nakon konfiguriranja, konfiguracija može biti predmet rekonfiguracije, što se prakticira da bi se ispunili novi zahtjevi kupaca. Prilikom pružanja podrške za proces konfiguriranja postoje različiti pristupi: najjednostavnije je uključiti hijerarhijsku strukturu kao osnovu za opis proizvoda te joj pridružiti odgovarajuća konfiguracijska ograničenja. Da bi se ta ograničenja prikazala, struktura mora biti proširena konceptima kao što su opcionalne, alternativne i parametarske komponente. Opcionalne komponente su one koje ne trebaju biti realizirane u svakoj varijanti. Alternativne imaju nekoliko različitih fizičkih mogućnosti koje ih realiziraju dok parametarske imaju parametre koji se mijenjaju tijekom konstrukcijskog procesa.

Problematika podrške procesu konfiguriranja posebno je područje kojim se bavi veliki broj istraživačkih timova, te izlazi izvan okvira ovog rada. Ono što je interesantno s gledišta ovog istraživanja je evidentiranje strukture različitih konfiguracija kroz koncept varijanti, bez podrške za implementaciju pravila za njihovo sklapanje.

Upravljanje projektima i poslovima

Upravljanje projektima i poslovima osniva se na konceptu podjele radnih zadataka. Već je prije rečeno da se kompleksni projekti dijele na podprojekte u kojima sudjeluje jedna osoba ili grupa ljudi. Informacije vezane uz projekte, a koje je potrebno bilježiti, čine podaci o strukturi projekata, odnosno način na koji su projekti vezani u hijerarhiji nekog nadređenog, kompleksnijeg projekta. Za svaki projekt potrebno je ostvariti točnu identifikaciju projekta. Upravljanje ovim segmentom uključuje koordinaciju među članovima tima, definiranje koraka i redoslijeda aktivnosti u konstrukcijskom procesu, dodjeljivanje ljudi za rad na pojedinim segmentima projekata, zadavanje rokova. Isto tako poželjno je automatizirati komunikaciju među članovima razvojnog tima u najvećoj mogućoj mjeri, odnosno definirati standardne poruke koje se razmjenjuju između članova tima, a govore o statusu podatka, zahtjevima za akcijama nad podacima, stanjima koja se aktiviraju ovisno o procesu koji se provodi nad podacima.

3.3 Potrebne funkcije sustava za podršku razmjeni i upravljanju s podacima o proizvodu

Na temelju provedene analize informacija kojima se upravlja u procesu razvoja proizvoda, definiran je popis potrebnih funkcija koje bi sustav za podršku pri razmjeni i upravljanju s podacima o proizvodu trebao podržavati. Pri tome se može naglasiti da izraz entitet u ovom popisu referencira komponentu proizvoda ili dokument kao nositelje podataka kojima se upravlja pomoću sustava:

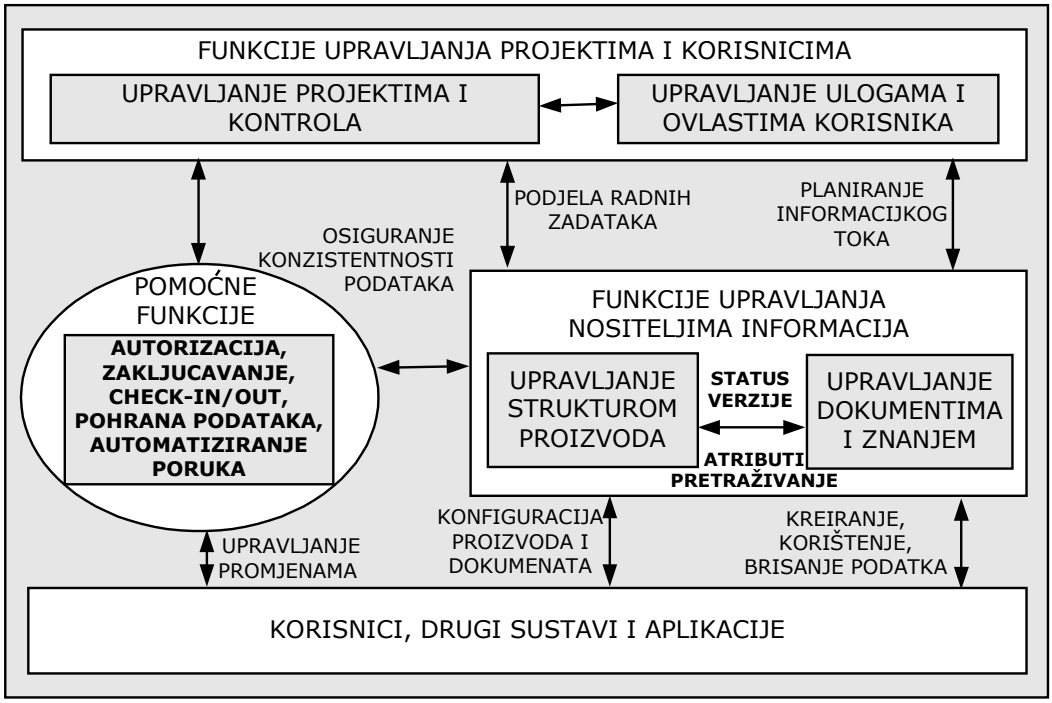
- *Sigurno spremanje podatka u središnje spremište podataka s kontroliranim pristupom.* Ova funkcija obuhvaća kontroliranje autorizacije za pristupanje podacima, te statusa podataka u smislu zaključavanja podataka u trenutku u kojem se vrše promjene. Ključni zahtjev je definiranje metodologije identifikacije i klasifikacije podatka, na način da se osigura jedinstvenost

svakog podatka, a u skladu sa zahtjevima pojedine realne implementacije.

- *Definiranje mehanizama za povezivanje entiteta i njihovih atributa.* Svojstva dokumenata i komponenti proizvoda na svim razinama su opisana značenjem atributa. Atributi pružaju nužne informacije o entitetu, te mogu biti korišteni za identifikaciju i pronalaženje entiteta.
- *Upravljanje hijerarhijskom podjelom entiteta u jednostavnije komponente.* Unutar ove funkcije potrebno je uključiti podršku za razvoj, održavanje i pretraživanje strukture dokumenata i komponenti proizvoda, na način koji će omogućiti brz i jednostavan pristup podacima. Potrebno je podržati definirane relacije među komponentama proizvoda i dokumenata, te međusobne relacije komponenti i dokumenta.
- *Upravljanje razvojem entiteta praćenjem verzija.* U konstrukcijskim okruženjima korisnici uobičajeno provode mnogo više vremena modificirajući postojeća rješenja nego stvarajući nova rješenja kroz revizije entiteta. Isto tako od sustava se zahtjeva upravljanje alternativnim varijantama entiteta, odnosno konfigurabilnim objektima, uz podršku alternativnim i opcionalnim komponentama. Naravno podršku je potrebno pružiti i relacijama između komponenti i njihovih mnogostrukih verzija.
- *Upravljanje procedurama za kreiranje, mijenjanje i brisanje entiteta.* Isto tako potrebno je osigurati procedure za izolaciju podataka za vrijeme editiranja u konstruktorskoj privatnoj radnoj okolini. Podaci također moraju biti provjereni i odobreni u točno definiranom redoslijedu operacija, prije nego što krenu u upotrebu, pa je potrebno podržati i procedure odobrenja.
- *Upravljanje projektima i korisnicima koji su dio razvojnog tima.* Potrebno je podržati strukturiranje projekata u manje radne cjeline, definiranje veza između projekta, registriranje korisnika, dodjeljivanje poslova određenim korisnicima, definiranje uloga i ovlasti za svakog pojedinog korisnika.
- *Upravljanje automatskim tokom podatka unutar razvojnog procesa.* To obuhvaća definiranje informacijskog toka unutar razvojnog tima s točno određenim redoslijedom procesa te automatiziranje poruka kojima članovi tima međusobno komuniciraju što se posebno odnosi na poruke o dodjeli radnih zadataka te upozorenja o načinjenim ili zahtijevanim promjenama.
- *Upravljanje procedurama trajne pohrane podataka (arhiviranja).* Potrebno je definirati načine pretraživanja završenih projekata i korištenja podataka koji su vezani uz takve projekte.
- *Integracija s ostalim alatima.* S točke gledanja krajnjih korisnika, iskoristivost sustava ovisi o tome koliko je taj sustav integriran s ostalim alatima koje koriste u svakodnevnom radu, pri čemu je posebnu pozornost potrebno posvetiti integraciji s preglednicima dokumenata i alatima za rad sa pojedinim tipovima dokumenata.

Prethodno analizirane funkcije mogu se grupirati u tri osnovne grupe [Slika 3.4]:

- funkcije upravljanja nositeljima informacija,
- funkcije upravljanja projektima i korisnicima,
- pomoćne (sistemske) funkcije.



Slika 3.4: Funkcionalna struktura sustava

Navedene grupe funkcija detaljnije su analizirane u okviru šeste glave koja govori o računalnoj implementaciji sustava.

3.4 Utjecaj na rad

Metamodelom informacija kojima se upravlja i koje se razmjenjuju u razvojnoj fazi proizvoda, definirani su podaci koje je potrebno uključiti u informacijski model kao jezgru sustava za razmjenu i upravljanje informacijama o proizvodu. Potreba za time je bila uvjetovana činjenicom da STEP standard pokriva vrlo široko područje razmjene podataka o proizvodu uključujući podatke iz različitih životnih faza proizvoda, te različitih područja, od brodogradnje do proizvodnje elektroničkih komponenti. Provedena analiza pomogla je u pravilnoj interpretaciji predloženog STEP informacijskog modela za razmjenu i upravljanje informacijama u konkretnom slučaju primjene za strojarske proizvode u njihovoj razvojnoj fazi.

4

REALIZACIJA INFORMACIJSKOG MODELA SUSTAVA PREMA ISO 10303-STEP STANDARDU

U ovoj glavi ukratko je objašnjen povijesni razvoj međunarodnih standarda za razmjenu podataka o proizvodu. Opširnije je opisana struktura ISO 10303-STEP standarda koji se kroz svoj dugogodišnji razvoj profilirao kao vodeći standard za razmjenu podataka o proizvodu. Poseban je naglasak stavljen na entitete STEP PDM sheme, odnosno njezinih dijelova čijom semantikom je realiziran informacijski model sustava.

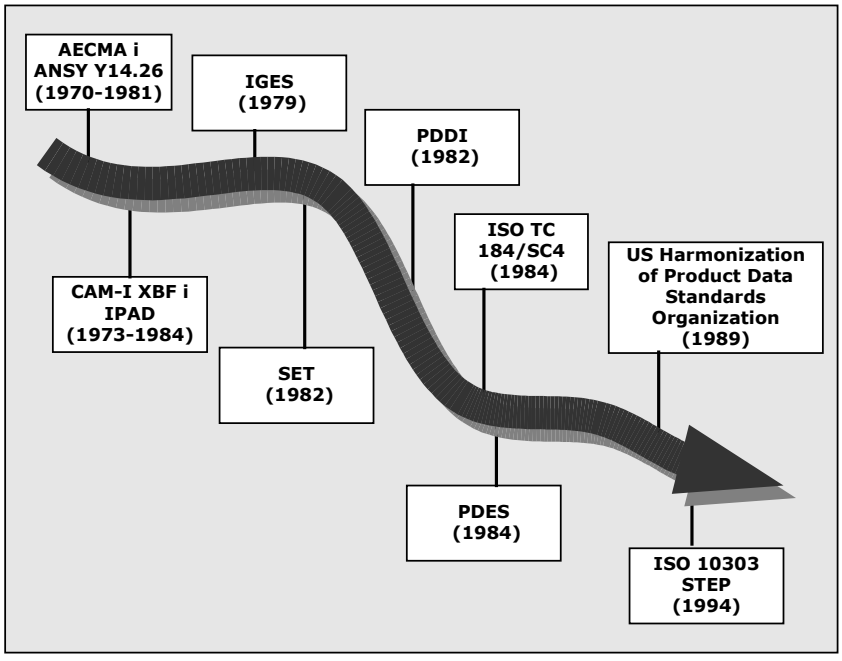
4.1 Općenito o razvoju međunarodnih standarda za prikaz i razmjenu informacija o proizvodu

Prema razmatranjima koja su prikazana u prethodnim glavama rada, može se zaključiti da su u istraživanju koje je prikazano ovim radom, analizirani različiti pristupi modeliranju problematike upravljanja i razmjene informacija o proizvodu. Raznolikost pristupa proizlazi iz činjenice da ne postoje općenito prihvaćeni standardi koji bi više ili manje opisali cjelokupno promatrano područje. Ipak, projekti kojima je cilj bila standardizacija ovog područja nisu novi, te će se na početku ove glave dati kratki pregled najvažnijih od njih, odnosno onih koji su dali značajniji doprinos ovom području.

Problem razmjene informacija je prepoznat u praksi vrlo rano, tako da prvi pokušaji standardizacije u ovom području datiraju iz 1970. godine [1] kad je Europsko udruženje zrakoplovne industrije AECMA, razvilo unificirani format koji je omogućio

suradnju kompanija u razmjeni jednostavnih geometrijskih informacija. Format je dorađivan u nekoliko navrata, ali s pojavom kompleksnijih tipova geometrijskog prikaza proizvoda, pojavili su se mnogi problemi u njegovom daljnjem razvoju. Nekako usporedo tome, NASA je pokrenula IPAD projekt koji je bio također orijentiran na razmjenu geometrije te je omogućio korištenje modeliranja informacija za integraciju različitih sustava. Jedan od najznačajnijih koraka u razvoju uslijedio je početkom 80-tih pojavom IGES-a (*eng. Initial Graphics Exschange Specification*) [56] kao univerzalnog translatora za prijenos geometrijskih informacija između različitih CAD paketa koji je aktualan i danas. IGES je u praksi omogućio prvu razmjenu podataka između CAD paketa pomoću datoteke. Također je poslužio i kao osnova za daljnja usavršavanja standarda. Prvi korak u tome je bilo definiranje SET (*franc. Standard D'Exchange et de Transfer*) standarda koji je startao u francuskoj zrakoplovnoj-industriji 1983., te VDA_FS (*njem. Verband der Automobilindustrie-Flachen-Schnittstelle*) u Njemačkoj. Nova istraživanja su donijela proširenje IGES-a u smislu definiranja detaljnijih specifikacija s posebnim osvrtom na strojarsko područje, podršku informacijama o strukturi podataka o proizvodu te pravila i preporuke za zadržavanje koherentnosti u budućem razvoju.

Nadogradnja koja je uslijedila istraživanjima u američkoj zračnoj-industriji kroz PDDI projekt, kretala se u smjeru definiranja skupa informacijskih modela, jezika i formata datoteka, te na taj način odvojila podatke od njihove definicije i mehanizama uporabe.



Slika 4.1: Povijesni razvoj standarda za prikaz informacija o proizvodu

Slijedeći vremensku liniju dolazimo do najvažnijeg koraka u pokušajima standardiziranja područja, internacionalnog standarda za razmjenu podataka o proizvodu, službeno poznatog i kao ISO 10303–STEP standard (*eng. **St**andard for **E**xchanging **P**roduct Data*). Glavni cilj projekta je definiranje standarda koji bi

obuhvaćao opis svih podataka vezanih uz proizvod. Prvi dijelovi standarda su prihvaćeni još 1994. godine, a novi se neprestano i kontinuirano razvijaju te usvajaju kao dijelovi međunarodnog standarda.

Najveći dio STEP-a bavi se modeliranjem geometrijskih podataka o proizvodu i načinima njihove razmjene. To je ujedno i najuspješniji dio standarda. STEP standard sastoji se mnogo logičkih cjelina podijeljenih u klase, od kojih će najvažnije biti pregledno opisane u poglavlju koje slijedi.

4.2 ISO 10303-STEP standard

ISO 10303 je internacionalni standard za računalno-primjenjiv prikaz i razmjenu podataka o proizvodu. Cilj standarda je osigurati neutralni mehanizam za opisivanje i razmjenu podataka o proizvodu tijekom životnog vijeka proizvoda, neovisno o sustavu koji ga koristi. Priroda tog opisa čini STEP pogodnim ne samo za razmjenu podataka u neutralnom formatu nego i kao osnovu za implementiranje baza podataka o proizvodu koje mogu poslužiti za trajno spremanje i dijeljenje istih između različitih korisnika i aplikacija[57]. Početno zamišljeni kontekst korištenja standarda možemo podijeliti u [58]:

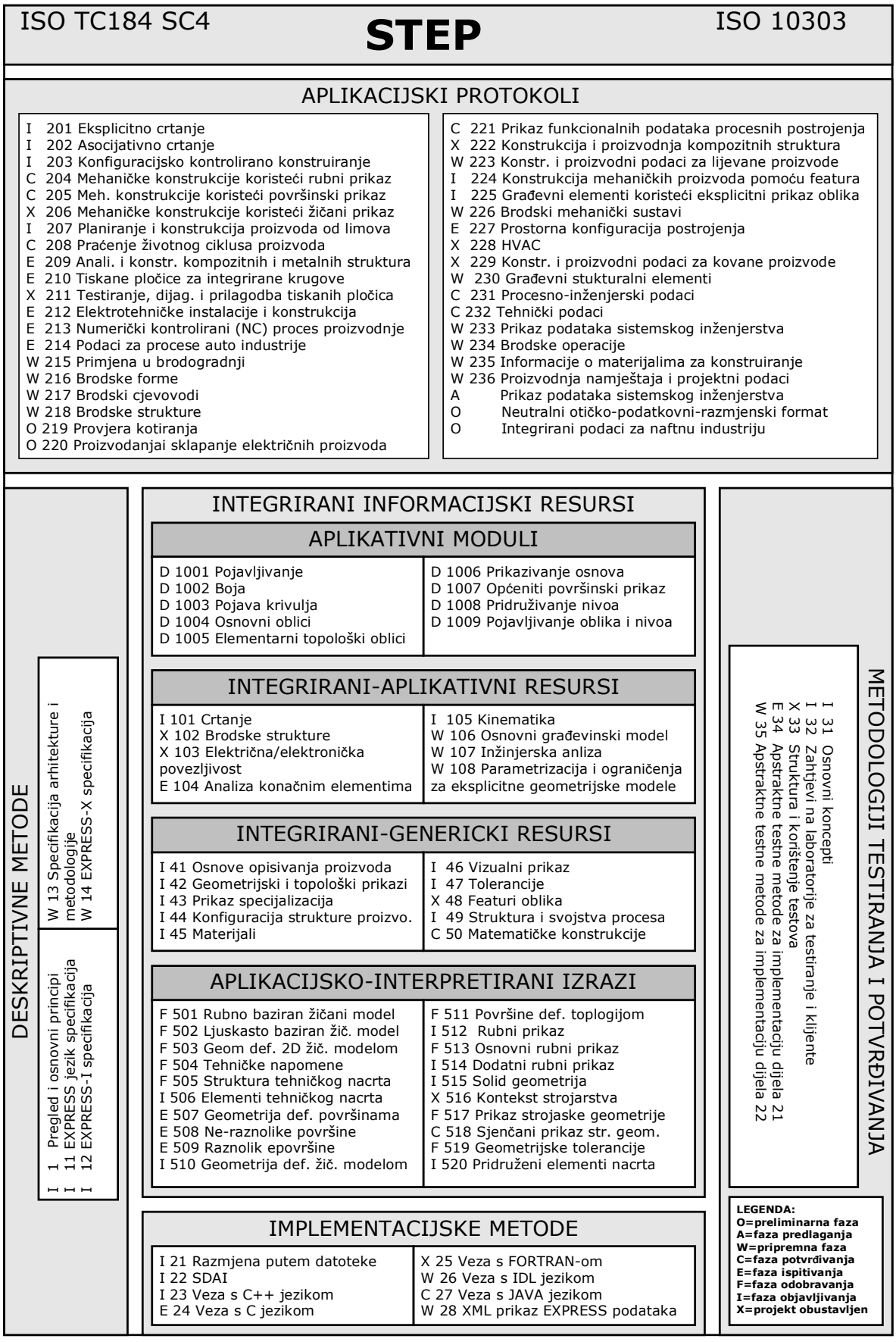
- *Razmjena podataka o proizvodu* – opisuje transfer podataka o proizvodu između različitih aplikacija. STEP definira oblik podataka koji se razmjenjuju između aplikacija. Svaka aplikacija pritom zadržava kopiju podatka u vlastitom obliku. Razmjenu podataka čini transfer informacija između jednog softverskog sustava k drugom kroz medij koji predstavlja stanje informacije u jednom vremenskom trenutku. Te informacije se spremaju digitalno, tipično u ASCII ili binarnom zapisu. Glavne karakteristike konteksta korištenja: iniciran od strane stvaratelja informacije, transformacija u neutralni format, sadržaj određen diskretnom vrijednošću u vremenu, stvara se kopija podatka.
- *Dijeljenje podataka o proizvodu* – definira pristup i operiranje nad jednom kopijom podataka između više aplikacija simultano. STEP je projektiran za podršku sučeljima između pojedine kopije i aplikacija koje ju dijele. Aplikacije ne čuvaju podatke u vlastitom obliku. Dijeljenje omogućuje jedan logički izvor informacija kojem različiti softverski sustavi imaju pristup. Kontrola pristupa, ispravljanje informacija i vlasništvo nad podacima su tipično kontrolirani implementacijom i administracijom izvora informacije. Glavne karakteristike konteksta: iniciran primateljem podatka, podacima se manipulira na zahtjev, postoje nivoi pristupa podacima, jedan izvor podatka bez kopija. Može se reći da je ovaj kontekst primjene tipičan za sustav koji se razmatra kao tema istraživanja prikazanog u ovom radu.
- *Arhiviranje podataka o proizvodu* – definira načine i procedure dugovječnog spremanja podatka, te je kao takav iskoristiv za podršku procesima arhiviranja podataka.

4.2.1 Struktura ISO 10303-STEP standarda

Arhitekturne komponente STEP-a reflektiraju se u dekompoziciji standarda u seriju dijelova. Dijelovi STEP-a mogu se kategorizirati u četiri glavne grupe [Slika 4.2]:

- *Deskriptivne metode* – podupiru STEP standard. Predstavljaju mehanizme za definiranje informacijskih konstrukcija standarda. Uključuju opis EXPRESS-a, formalnog jezika za modeliranje podatka. EXPRESS se koristi kao jezik modeliranja problematike u ostalim dijelovima STEP standarda [59].
- *Integrirani informacijski resursi* – informacijski modeli koji se koriste u više specifičnih dijelova standarda. Oni su konceptualni modeli podataka o proizvodu te osnovni semantički elementi koji se koriste za opis proizvoda u bilo kojoj fazi životnog vijeka. Najveći dio podrške STEP-a razmjeni geometrije definiran je unutar ovih resursa, uključujući podršku za žičane modele, geometriju površina i geometriju punih tijela. Osim toga sadrže podršku za konfiguriranje proizvoda, inženjerske analize, proces konstruiranja, materijale, vizualno predočenje. Općenito se može reći da ako neki koncept mora biti uključen na nekoliko mjesta u STEP-u, biti će definiran unutar integriranih informacijskih resursa.
- *Aplikacijski protokoli* –informacijski modeli koji opisuju specifičnu individualnu primjenu standarda za razmjenu i upravljanje s podacima o proizvodu Aplikacijski protokoli sadrže informacijske modele zapisane u EXPRESS-u koji zadovoljavaju specifične zahtjeve, za opis proizvoda, definirane aplikacijskim kontekstom, te predstavljaju dijelove STEP-a koji se mogu implementirati. Aplikacijski protokoli predstavljaju najveći i najznačajniji dio standarda. Većina dijelova je u razvoju dok je manji dio usvojen. Za razliku od integriranih resursa, koji definiraju načine prikaza geometrije, aplikacijski protokoli definiraju uporabu tih prikaza. Aplikacijski protokoli mogu se implementirati uporabom jedne od implementacijskih metoda.
- *Implementacijske metode* – opisuju precizno preslikavanje iz informacijskih specifikacija pojedinih dijelova STEP-a u prikaz informacija (npr. tekstualna datoteka) i pristup informacijama (npr. veza prema programskim jezicima). Svaka implementacijska metoda definira način na koji se informacijske konstrukcije preslikavaju. Uključuju razmjenu pomoću fizičke datoteke, standardno sučelje za pristup podacima (SDAI), te sučelja prema programskim jezicima C, C++, FORTRAN, JAVA, XML.

Dijelovi infrastrukture standarda, koju čine deskriptivne i implementacijske metode, odvojeni su od dijelova ovisnih od industrijskoj grani (aplikacijskih protokola). Trenutno su aplikacijski protokoli definirani za strojarske i električne proizvode, a u razvoju su aplikacijski protokoli za definiranje kompozitnih materijala, savijanja limova, automobilsku industriju, proizvodne procese, brodograđevnu industriju, itd. Očekuje se tijekom vremena definiranje aplikacijskih protokola za sve grane industrije.



Slika 4.2: Arhitektura STEP standarda

4.2.2 EXPRESS

EXPRESS je razvijen kao jezik za formalno opisivanje informacijskih modela. Počeci razvoja jezika datiraju iz 1982. godine kada se u okviru PDDI projekta razvio DSL jezik, na osnovu kojega se 1986. godine EXPRESS razvio. EXPRESS je formalni jezik za opisivanje informacijskih modela, a ne implementacijski jezik kao npr. C, te se na osnovu EXPRESS zapisa ne može kreirati izvršni kod. Ovdje će se izdvojiti neke karakteristike jezika:

- EXPRESS se može uporabiti za opisivanje ograničenja te strukture i relacija između podataka. Ograničenja predstavljaju eksplicitan oblik zadovoljenja ispravnosti informacijskog modela.
- Modeli podataka opisani EXPRESS-om mogu se obrađivati uporabom računala tj. uporabom određenih programskih rješenja. Na ovaj način je izbjegnuta potreba za korisničkom interpretacijom ili transkripcijom zapisa.
- EXPRESS je međunarodno priznati standard za formalno opisivanje modela podataka što predstavlja veliku prednost pri uporabi.

Jezik je do danas prošao kroz nekoliko verzija te se razvijao simultano sa razvojem STEP-a. Tijekom izrade pojedinih verzija EXPRESS je poprimio neke mogućnosti drugih programskih jezika (C, C++, ADA, Algol, Euler, Modula-2, Pascal, PL/I, SQL).

Osnovni element EXPRESS-a je shema. Shema sadrži definiciju modela i služi za određivanje granica definicije modela te kao mehanizam podjele velikih informacijskih modela. U svakoj shemi postoje tri kategorije definicija:

- *Definicije entiteta* opisuju klase objekata stvarnog svijeta uz pripadajuće osobine. Osobine objekata nazivaju se atributi. Atributi mogu predstavljati jednostavne vrijednosti (naziv, težina, itd.) ili relacije između instanci (vlasnik, dio od). Entiteti se mogu organizirati u hijerarhijske strukture te na taj način nasljeđivati atribute od nadređenih entiteta. Mehanizam nasljeđivanja podržava jednostruko i višestruko nasljeđivanje tzv. AND/OR nasljeđivanje.
- *Definicije tipova* opisuju područja mogućih vrijednosti podataka. Jezik podržava nekoliko ugrađenih tipova podataka, na osnovu kojih se mogu kreirati novi tipovi, uporabom mehanizama generalizacije ili agregacije tipova podataka.
- *Definicije algoritama* predstavljaju definicije funkcija ili procedura koje služe za definiranje ograničenja.

Atributi se mogu prikazati pomoću jednostavnog tipa podataka (npr. *integer*, *string*…) ili preko složenog tipa podataka tj. drugog entiteta. Tipovi podataka određuju područje vrijednosti podataka. Podržani tipovi podataka dijele se na: jednostavne, složene, imenovane i konstrukcijske tipove podataka.

Prilikom opisa EXPRESS-a bitno je naglasiti dodirne točke sa objektnim programskim jezicima. U EXPRESS-u entiteti su definirani kao klase. Iz definiranih klasa mogu se

derivirati druge klase koje nasljeđuju osobine nadređenih klasa. Iako EXPRESS posjeduje osobine objektnih programskih jezika on to zapravo nije jer ne dozvoljava definiranje metoda. U EXPRESS-u je mehanizam za određivanje klasifikacije ostvaren strukturom *nadređeni entiteti*/*derivirani entiteti*. *Derivirani entiteti* mogu biti međusobno povezani na različite načine ovisno o zadanom operatoru. Dozvoljeni operatori za određivanje odnosa između *deriviranih entiteta* su:

- *ONEOF* – instance *deriviranih entit*eta se međusobno isključuju,
- *ANDOR* – ukoliko se instance *deriviranih entiteta* međusobno ne isključuju ili uključuju tada se odnos između njih definira kao *ANDOR*,
- *AND* – omogućava definiranje višestrukih međusobno uključivih relacija.

Derivirani entiteti nasljeđuju atribute *nadređenih* te ukoliko *derivirani entitet* ima više od jednog *nadređenog* tada nasljeđuje atribute svih *nadređenih*. No, prilikom višestrukog nasljeđivanja može doći do konflikta između atributa različitih *nadređenih entiteta*. Navedena situacija se rješava na taj način što svaki atribut može imati prefiks koji sadrži ime *nadređenog entiteta* iz kojega se atribut nasljeđuje.

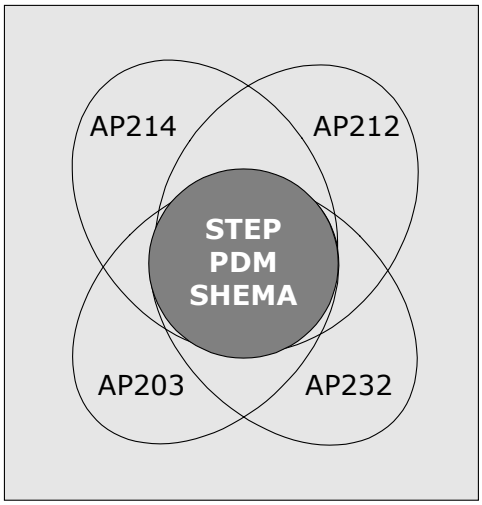
Informacijski model EXPRESS-a definiran je shemama koje sadrže definiciju etniteta, tipova, funkcija i procedura, te pravila na entitetima i pravila koja definiraju relacije između entiteta ili relacija. U EXPRESS-u definirana pravila se ne mogu izvršavati niti se varijablama mogu pridružiti vrijednosti, ali je uključena potpuna proceduralno-jezična sintaksa za određivanje pravila. EXPRESS je potpuni proceduralni jezik za definiciju strukturalnih i varijabilnih relacija, omogućuje definiciju apstraktnih tipova podataka, koristeći ograničenja za opis ponašanja objekta. Bitno je naglasiti da je jezik samo specifikacija i nije izvršan. Zbog gore navedenih svojstava EXPRESS je izabran za modeliranje metamodela podataka o proizvodu kojima je potrebno upravljati u razvojnoj fazi proizvoda.

4.3 Osnovni entiteti STEP PDM sheme

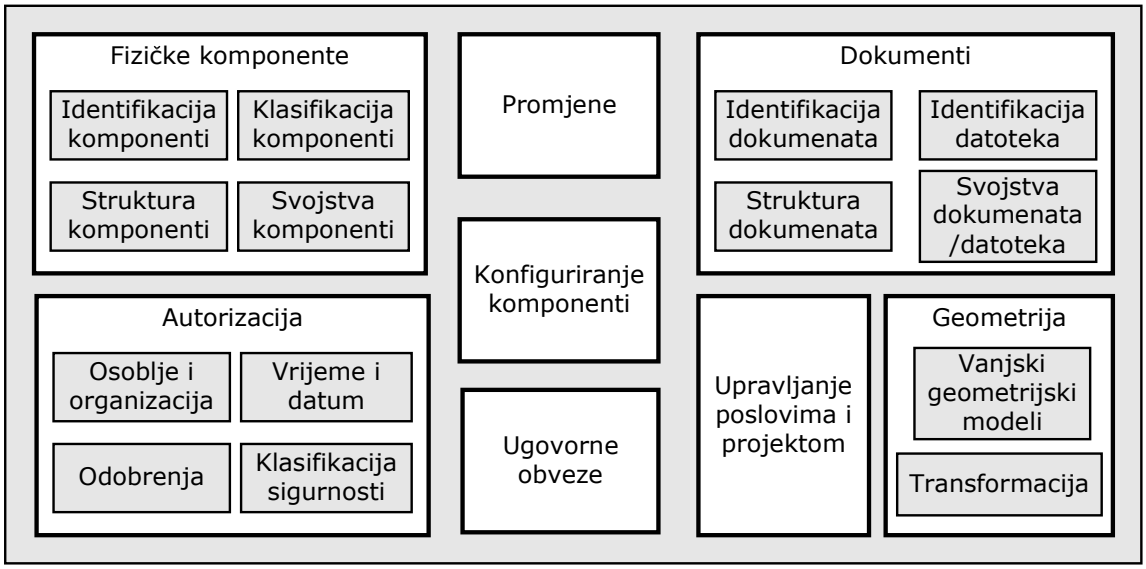
STEP PDM shema čini osnovni skup entiteta koji podržavaju koncept upravljanja i razmjene podataka o proizvodu unutar STEP standarda. Sheme je nastala integracijom pojedinih aplikacijskih protokola definiranih STEP standardom [Slika 4.3], a koji zadiru u područje upravljanja informacijama o proizvodu [60]. Potrebno je naglasiti da STEP PDM shema još nije službeno potvrđena kao dio standard nego se nalazi u fazi odobravanja, te je zbog toga potpuno otvorena za nadogradnju i prilagođavanja.

Entiteti sheme za upravljanje s podacima o proizvodu organizirani u grupe prema kriteriju sudjelovanja entiteta u realiziranju funkcionalnosti sheme. Podjela u grupe sugerira modularnu strukturu sheme. Detaljan opis svih semantičkih jedinice [Slika 4.4] koje čine funkcionalnost sheme, rezultirao bi prevelikim brojem stranica, pa su u daljnjem tekstu opisani najvažniji entiteti koji čine shemu i relacije između njih, a

koji su blisko vezani uz elemente metamodela analiziranog i prikazanog u trećoj glavi ovog rada. Entiteti i relacije su prikazani korištenjem EXPRESS-G specifikacije.



Slika 4.3: Pozicija i uloga STEP PDM sheme



Slika 4.4: Struktura STEP PDM sheme

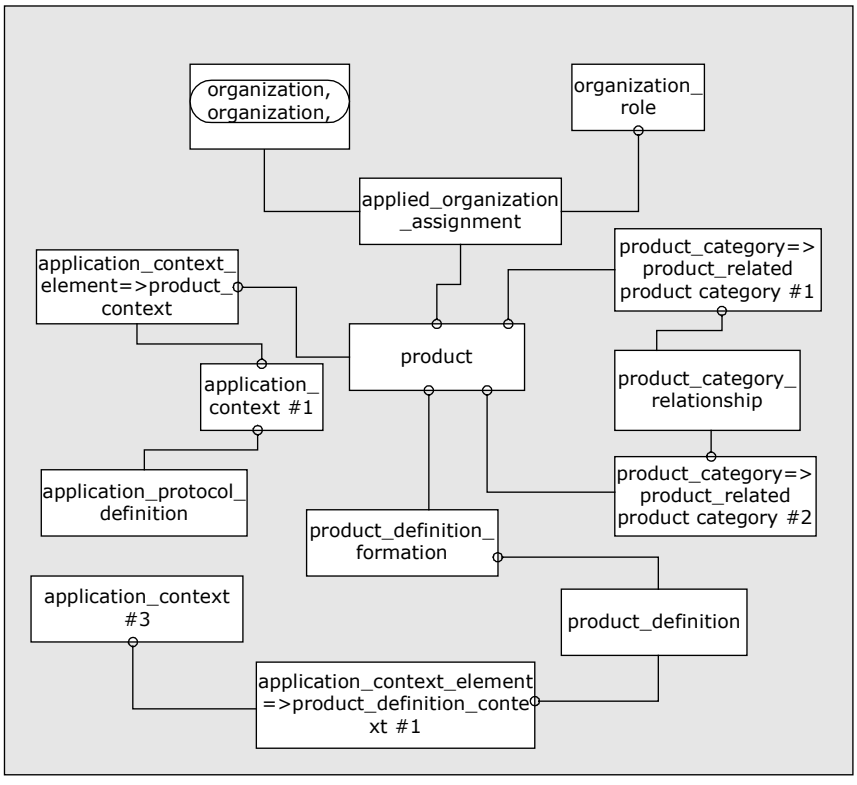
4.3.1 Identifikacija i klasifikacija komponenti

Počevši od fizičkih komponenti koje čine proizvod, STEP PDM shema definira način upravljanja svim podacima vezanim uz njih. Jedinstvenu identifikaciju svake pojedine komponente proizvoda definiraju tri odvojena koncepta [Slika 4.5]:

- *Glavna identifikacija komponenti* –razgraničuje identifikaciju različitih verzija, te definiciju različitih pogleda na komponente proizvoda. Pored toga definira pripadnost pojedine komponente sa svim njezinim karakteristikama pojedinom projektu (proizvodu), odnosno korisnicima u razvojnoj fazi proizvoda. Tako definirana daje osnovu za evidentiranje koraka u razvoju komponenti proizvoda kroz revizije. Osnovni entitet *product* predstavlja

osnovu identifikacije svake pojedine komponente. Shemom je preporučena veza sa entitetima organizacijske sheme koji definiraju korisnike i projekte te definiraju odgovornost korisnika za pojedinu komponentu. Entitet *product_definition_formation* predstavlja osnovu za identifikaciju specifičnih revizija komponenti definiranih entitetom p*roduct.*. Skup takvih entiteta koji su vezani uz jedan *product* entitet predstavlja proces razvoja komponenti proizvoda opisan kroz revizije. Entitet *product_definition* predstavlja identifikaciju specifičnog pogleda na pojedinu reviziju komponente, vezano uz specifičnu fazu životnog vijeka proizvoda ili područje primjene.

- *Informacije o kontekstu* – daju nužne pretpostavke za pravilnu interpretaciju informacije o identifikaciji proizvoda, a sastoji se od dva različita, ali međusobno povezana područja: identifikacije aplikacijskih protokola i informacija o aplikacijskom kontekstu. Osnovni element *application_protocol_definition* definira STEP specifikaciju koja uključuje ili koristi PDM shemu te na taj način definira općeniti kontekst razmjeni podataka. Entitet *application_context* definira područje primjene koje definira podatke (DMU, konstrukcija, planiranje procesa proizvodnje….). Entitet *product_context* i njegovi podtipovi definiraju pogled na podatke iz perspektive inženjerske primjene.



Slika 4.5: Dijagram entiteta za identifikaciju i klasifikaciju fizičkih komponenti

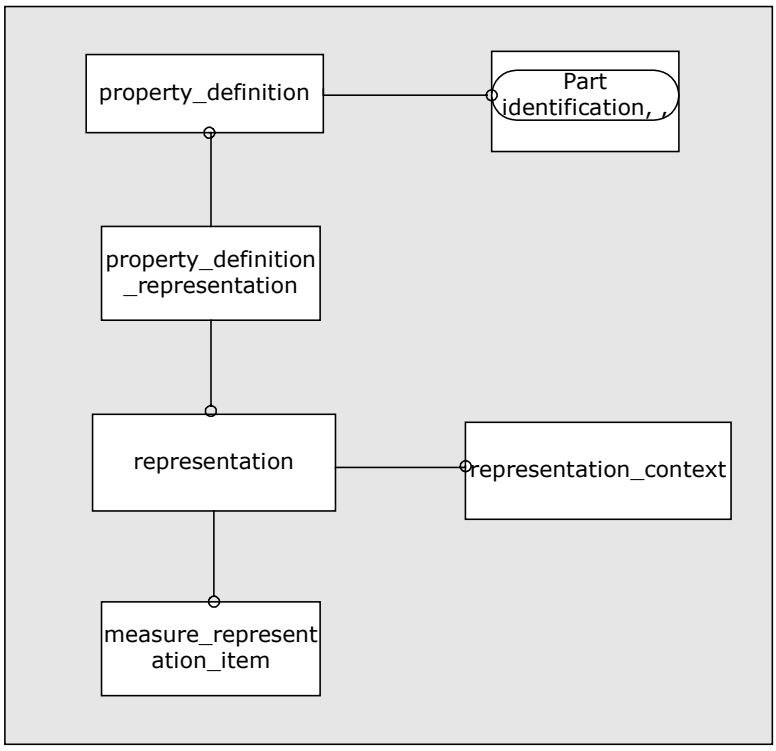
- *Klasifikacija tipova* – definira osnovne pretpostavke za razvrstavanje komponenti proizvoda koje se interpretiraju kao fizički dijelovi i komponenti koje se interpretiraju kao strukturirani dokumenti. Također daje podršku

razlikovanju različitih vrsta komponenti, npr. sklop, dio. Osnovni entitet *product_related_product_category* predstavlja specifičnu klasifikaciju vezanu direktno uz komponentu, a *product_category_relationship* definira hijerarhijsku vezu među kategorijama.

4.3.2 Svojstva komponenti

STEP PDM Shema omogućuje specificiranje svojstava koja su vezana uz proizvod i njegove komponente. Svojstvo je definicija fizičkih ili proizvoljnih karakteristika definiranih od strane korisnika. Jedno od specijalnih svojstava je i oblik komponente proizvoda – prikaz geometrije. Svojstva podržana shemom se mogu podijeliti u:

- *Općenita svojstva komponenti* [Slika 4.6] – svojstva vezana uz nositelje informacija o proizvodu, nezavisna svojstva ili predefinirana svojstva kao npr. masa, kvaliteta površine, cijena, trajnost, materijal. Osnovni entitet *property_definition* predstavlja svojstvo koje karakterizira komponentu. Vezu između tako definiranog svojstva i prikaza tog svojstva opisuje entitet *property_definition_representation*. Entiteti *representation*, *representation_context* i *measure_representation_item* definiraju način prikaza svojstva i njegovu kvantitetu.
- *Svojstva koja definiraju vanjski oblik komponenti* – svojstva vezana uz geometriju odnosno oblik, datoteke koje čuvaju geometriju. Ova svojstva se neće razmatrati u radu, jer se nalaze izvan područja u kojem je definiran predmet istraživanja u ovom radu.



Slika 4.6: Dijagram entiteta za prikaz osnovnih svojstva komponenti

- *Svojstva koja definiraju strukturu geometrijskih modela* – topološke veze sa ostalim komponentama, orijentacija i pozicioniranje komponenti u sklopu Ova svojstva se također neće razmatrati u radu.

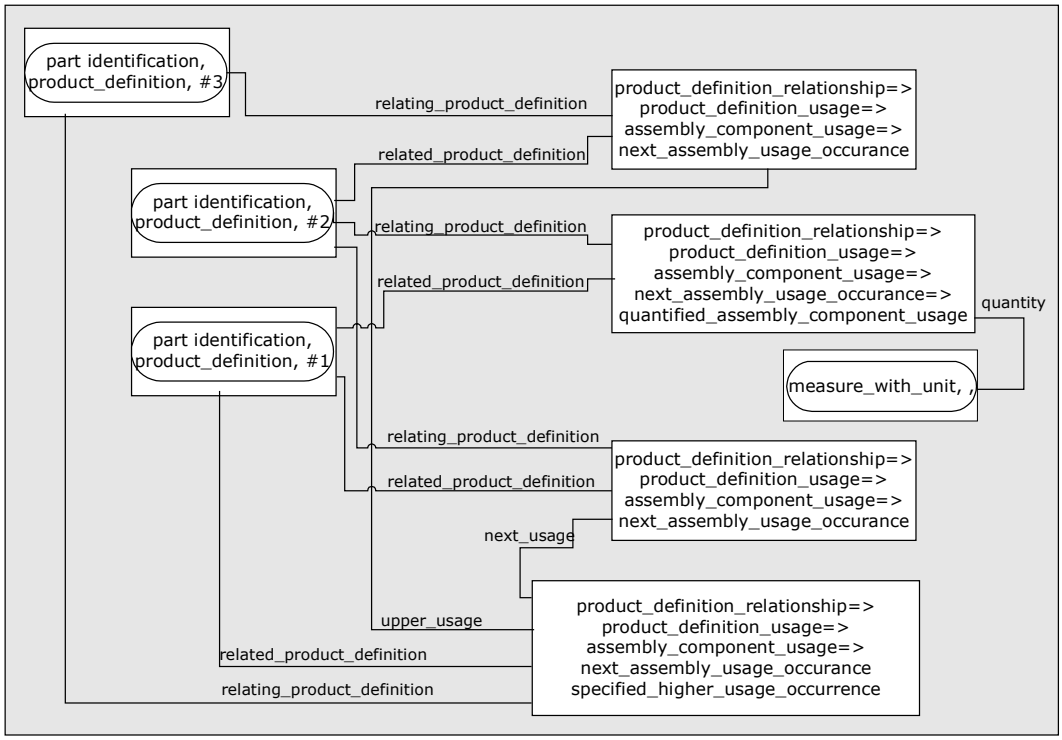
Za prikazivanje svojstava komponenti definirani su brojni tipovi. Shemom se u tu svrhu preporuča korištenje vrijednost atributa u *property_definition* entitetu. Navedimo neka od njih. Reciklabilnost definira sposobnost komponente da bude ponovo iskorištena nakon svoje primarne upotrebe. Masa predstavlja kvantitetu koja govori o količini materijala od kojeg se komponenta sastoji. Kvaliteta površine je svojstvo koje govori o potrebnom načinu izrade komponenti proizvoda. Cijena specificira troškove proizvodnje pojedine komponente. Trajnost određuje predviđeni vremenski period za sigurnu upotrebu komponente.

4.3.3 Hijerarhijska struktura komponenti i relacije među njima

STEP PDM shema podržava eksplicitnu hijerarhijsku strukturu komponenti koje čine proizvod. Ovakva eksplicitna struktura odgovara tradicionalnim konstruktorskim sastavnicama. Relacije koje postoje između komponenti su višeznačne. Najprije, jednostruko pojavljivanje konstitutivne komponente u definiciji nadređene komponente opisuje entitet *next_assembly_component_usage.* Višestruko pojavljivanje komponenti kao konstituanta neke druge komponente moguće je opisati na dva načina: višestrukom upotrebom entiteta *next_assembly_component_usage* ili korištenjem entiteta *quantified_assembly_component_usage.* Koji entitet će se koristiti ovisi o tome da li se pojedinoj komponenti mora moći pristupiti individualno ili ne. Ponekad, bez obzira što su komponente fizički jednake, moraju imati drukčije ime, posebnu identifikacijsku karakteristiku, ili neko svojstvo kao na primjer različitu orijentaciju u nadređenoj komponenti.

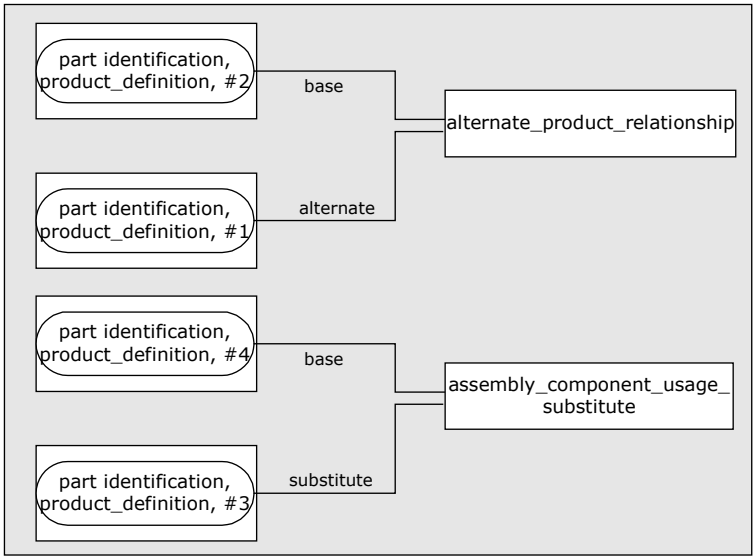
PDM shema putem entiteta *promissory_usage_occurance* pruža podršku za identifikaciju pojavljivanja određene komponente kao sastavnog dijela komponente koja se nalazi na višem nivou hijerarhije nego što joj je neposredno nadređena komponenta. Ovakva mogućnost može se iskoristiti u slučajevima kad struktura nije kompletno definirana, odnosno u fazi koncipiranja za kreiranje preliminarne sastavnice. Omogućeno je i opisivanje pojavljivanja komponente u više hijerarhijskih razina unutar proizvoda pomoću entitet *specified_higher_usage_occurrence* koji za svako pojavljivanje komponente omogućuje definiranje različite identifikacije, pozicije u sklopu, geometrijskog prikaza ili drugih svojstava koja mogu biti različita od onih koja su inicijalno određena za komponentu.

Osim toga shema omogućuje postavljanje relacija između komponenti kojima se definiraju alternativne i zamjenske komponente [Slika 4.8], odnosno relacije za podršku i identifikaciju komponenti koje se naručuju od vanjskih dobavljača. Alternativne komponente mogu se zamijeniti u svakom pojavljivanju komponente, dok zamjenske mogu zamijeniti komponentu samo u posebno definiranim situacijama.



Slika 4.7: Dijagram entiteta za prikaz hijerarhijske strukture komponenti

Osnovni entitet je *alternat_product_relationship* koji definira da je alternativna komponenta po obliku i funkciji ekvivalentna originalnoj u svakom slučaju korištenja. Što se tiče zamjenskih komponenti, entitet *assembly_component_usage_substitute* definira kontekst korištenja u kojem komponenta može biti zamijenjena drugom.



Slika 4.8: Dijagram entiteta za prikaz alternativnih i zamjenskih komponenti

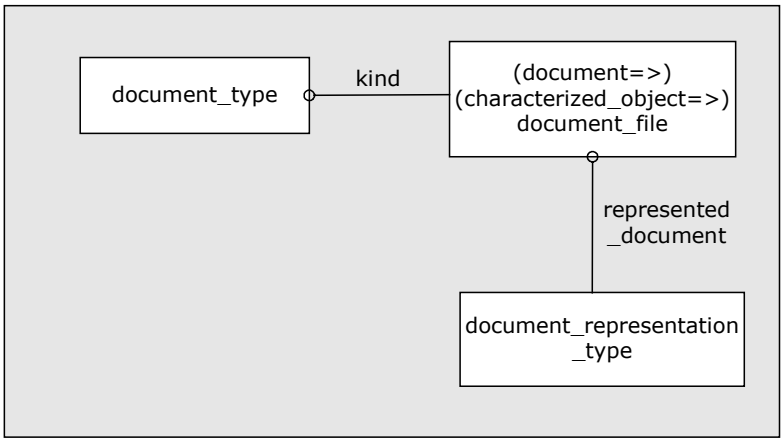
Relacije između revizija određene komponente definiraju se entitetom *product_definition_formation_relationship* koji opisuje hijerarhijsku vezu između prethodne i trenutne revizije. Na taj način je osigurana pohrana "povijesti nastajanja" revizija. Obje revizije vezane na taj način moraju imati referencu na osnovnu komponentu koju definiraju.

4.3.4 Identifikacija i klasifikacija dokumenta

Osnovna koncepcija STEP standarda pod nazivom "*eng. part as product*" koja je definirana u prethodnim poglavljima kroz identifikaciju, klasifikaciju i definiciju strukturalnih odnosa između komponenti proizvoda, primjenjuje se i u konceptu "*eng. document as product*". Drugim riječima entiteti koji su prethodno opisani koriste se i za definiranje strukturalne organizacije dokumenata koji su organizirani kao hijerarhijske strukture komponenti zajedno s pripadajućim im svojstvima. Bitno je samo u identifikaciji kategorije entiteta definirati da se ne radi o fizičkoj komponenti, nego o strukturiranom dokumentu. Ista pravila vrijede svugdje osim na mjestima gdje nisu posebno specificirane razlike. Jedan od primjera razlike je definiranje konteksta primjene, koji se ne definira za dokument posebno, nego za dokument vrijedi kontekst koji je određen komponentom kojoj je dokument pridružen.

4.3.5 Vanjske reference i njihova veza s dokumentima

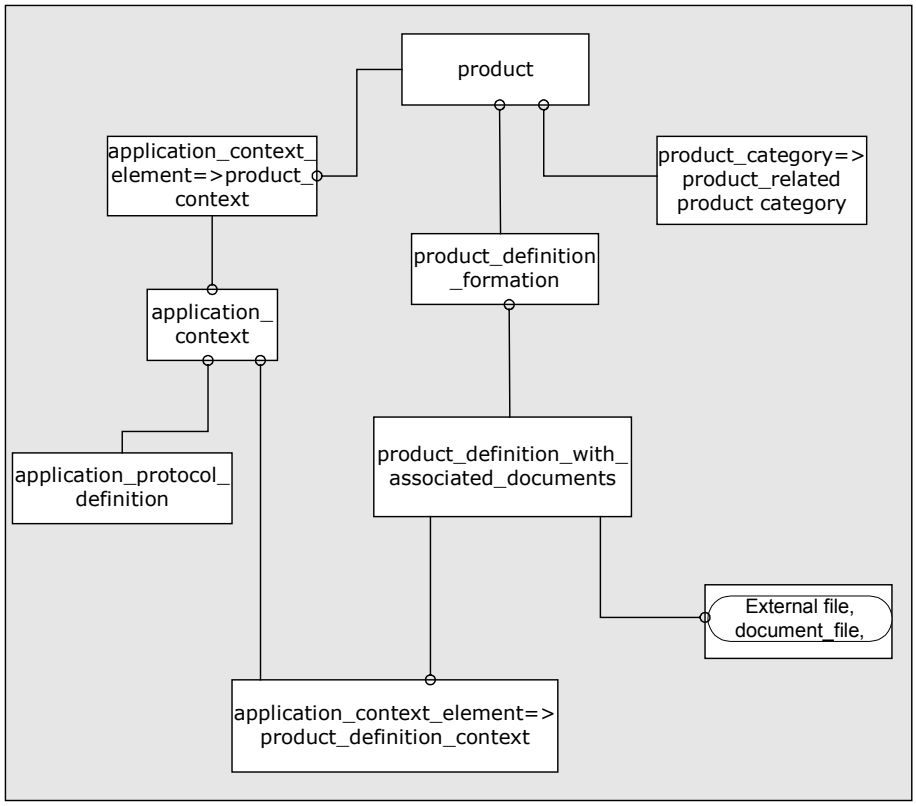
Ono što je posebno interesantno u opisivanju dokumenta su entiteti koji predstavljaju vanjske reference na nositelje koji sadrže informacije i kojima se pristupa bez zadiranja u organizaciju njihovog sadržaja. Vanjske reference mogu predstavljati bilo digitalne ili "papirnate" dokumente. Vanjska reference asocijacijom je vezana uz ostale podatke o proizvodu, odnosno njihove nositelje. Identifikacija vanjskih referenci se izvodi korištenjem entiteta *document_file* [Slika 4.9]. Vanjske reference se klasificiraju pomoću entiteta *documet_type*, a način prikaza odnosno određivanje da li se radi o digitalnom ili papirnatom obliku reference definira entitet *document_representation_type*.



Slika 4.9: : Dijagram entiteta za identifikaciju vanjskih datoteka

Entitet *product_definition_with_associated_documents* specifičan je za kontekst "*eng. document as product*" i samo se u tom slučaju može koristiti [Slika 4.10]. On predstavlja vezu prema vanjskim (konstituirajućim) datotekama koje čine aktualni sadržaj definicije prikaza dokumenta. Ovakav način definiranja je potreban da bi se omogućila kontrola i praćenje promjena, odnosno da bi se promjena konstituirajuće

datoteke reflektirala kroz definiciju nove verzije dokumenta.



Slika 4.10: Entiteti za prikaz veza dokumenta sa konstituirajućim datotekama

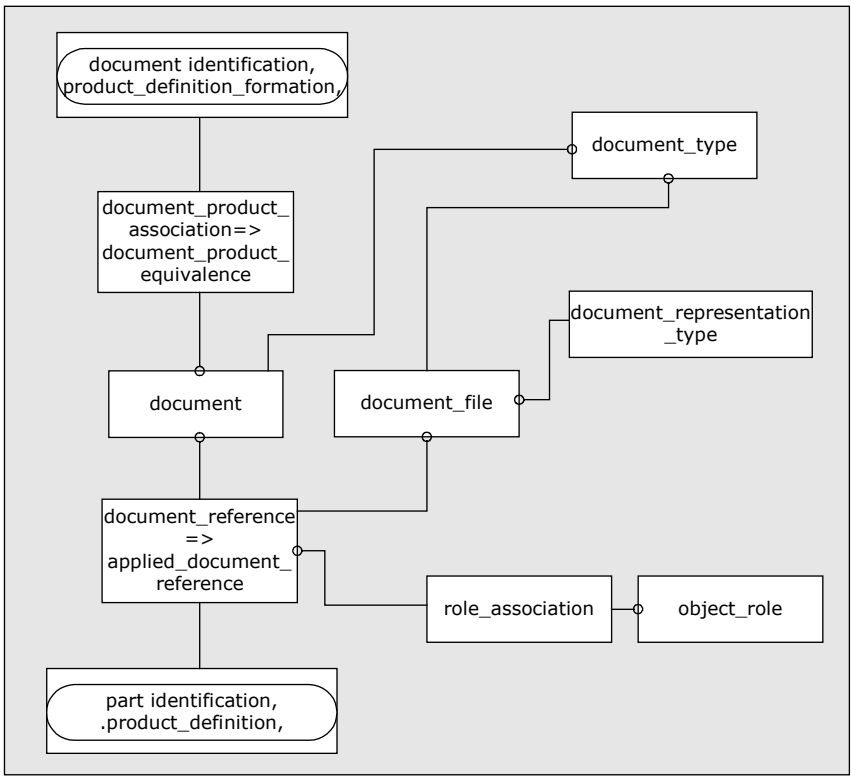
4.3.6 Svojstva dokumenata i vanjskih referenci

Svojstva dokumenata su u PDM shemi vezana kako uz prikaz dokumenta tako i na pojedine vanjske reference. Shema entiteta koji opisuju svojstva dokumenata i vanjskih referenci odgovara shemi za definiranje svojstva fizičkih komponenti [Slika 4.6]. Da bi se izbjegla redundancija preporuča se da svojstva koja se dijele između svih konstituanata budu vezana direktno uz entitet *representation* dokumenta nego da se repliciraju za sve vanjske reference. Moguće je posebno izdvojiti svojstva koja su specifična za vanjske reference:

- *Document_content_property* – opisuje referencu (npr. razina prikazanih detalja, tip geometrije, jezik, mjerilo i sl.).
- *Document_creation_property* – opisuje softver kojim je referenca kreirana (npr. verzija, operativni sustav, putanju do izvršne datoteke i sl.).
- *Document_format_property* – opisuje format reference (npr. ekstenzija datoteke, znakovni kod, format dokumenta i sl.)
- *Document_size_property* – definira veličinu vanjske reference (npr. veličina datoteke, broj stranica u dokumentu i sl.).
- *Document source_property* – definira lokaciju (putanju) gdje se vanjska reference nalazi bilo da se radi o digitalnom ili 'papirnatom' obliku.

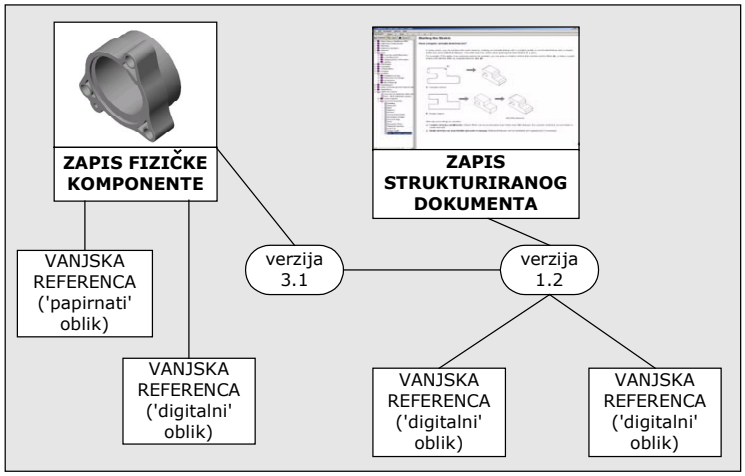
4.3.7 Veza strukturiranih dokumenata i vanjskih referenci s fizičkim komponentama

Konceptom "*eng. document as product*" strukturirani dokumenti i vanjske reference mogu se povezati sa fizičkim komponentama [Slika 4.11] relacijom asocijacija koja je realizirana korištenjem entiteta *document* i *applied_document_reference*. Kada se dokument koji promatramo kao strukturu veže za fizičke komponente, osnovni dokument se veže na referentnu komponentu korištenjem dodatnog entiteta *document_produt_equivalence* koji specificira vezu neke od komponenti dokumenta na fizičku komponentu proizvoda. Veza komponente proizvoda i komponente složenog dokumenta može se ostvariti na razini osnovne identifikacije, revizije komponenti, ili definicije prikaza komponenti. Preporučena razina je veze preko revizija komponenti. Vanjske reference mogu se vezati s fizičkim komponentama na isti način. Obzirom da su vanjske reference prikazane entitetom *document_file* potreban nam je samo entitet *applied_document_reference* da bi ih vezali uz fizičke komponente. *Object_role* entitet definira ulogu dokumenata i referenci koji su pridruženi fizičkim komponentama i to na dvije razine: *mandatorna* uloga što znači da je dokument integralni dio potpune definicije proizvoda i *informativna* koja znači da dokumenti sadrže samo neke dodatne informacije koje nisu presudne za razumijevanje proizvoda.



Slika 4.11: Entiteti za prikaz veza dokumenata i vanjskih datoteka sa fizičkim komponentama

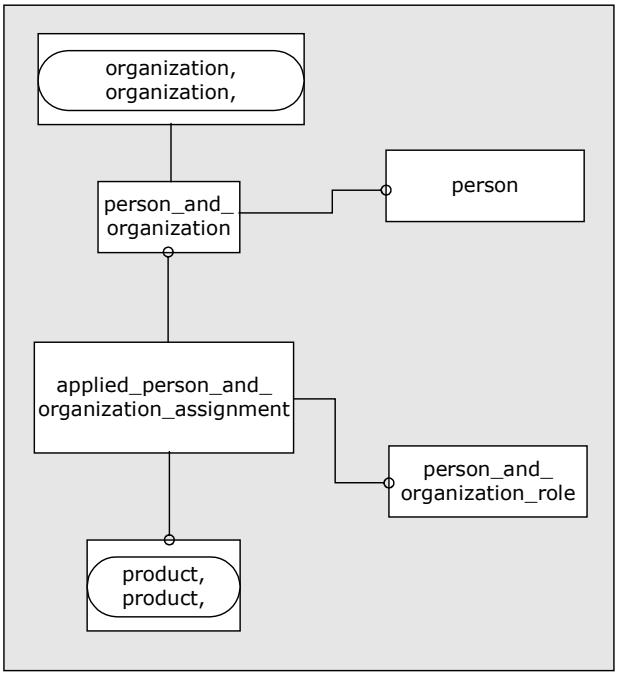
Značenje gore navedenog koncepta prikazano je na [Slika 4.12], gdje je prikazana veza fizičkih komponenti sa strukturiranim dokumentima i datotekama.



Slika 4.12: Ilustracija veze između dokumenata, datoteka i fizičkih komponenti

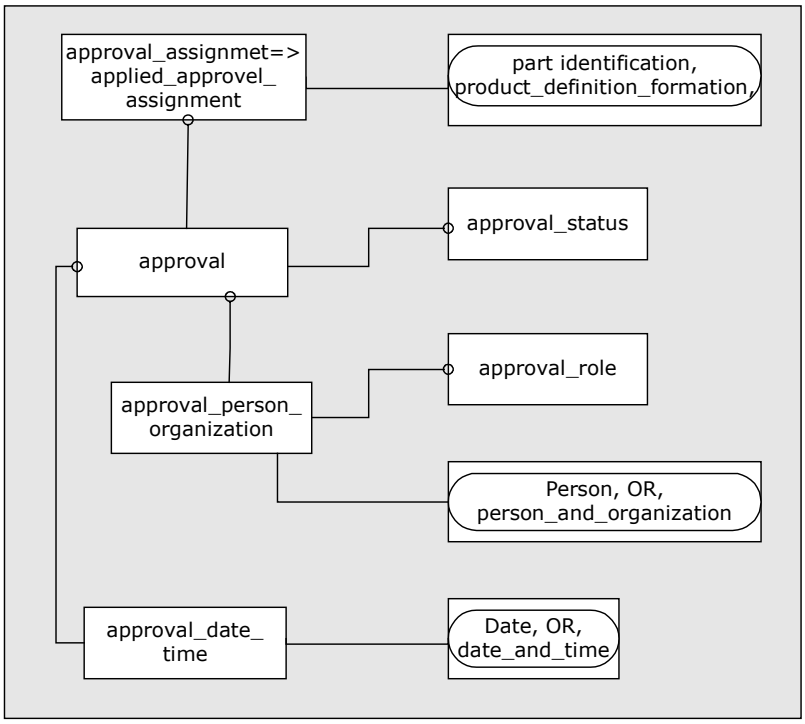
4.3.8 Autorizacija

STEP PDM shema prikazuje organizaciju razvojnog tima i osobe koje su dio organizacije u smislu funkcija koje oni provode vezano uz podatke o proizvodu i njihove veze [Slika 4.13]. Osoba je uvijek definirana u kontekstu neke organizacije entitetom *person.* Organizacija predstavlja grupu korisnika i definirana je entitetom *organization*. Pripadnost osoba organizaciji definirana je entitetom *person_and_organization*. Proizvod i njegove komponente te dokumenti i datoteke su uvijek vezani uz organizaciju, definirajući na taj način odgovornosti za pojedine proizvode i dokumente pomoću entiteta *applied_person_and_organization_ assignment.* Uloga organizacije ili osoba koju imaju u upravljanju s podacima o pojedinim proizvodima definirana je entitetom *person_and_organization_role*.



Slika 4.13: Dijagram entiteta za prikaz entiteta organizacije i osoba

Procedure odobravanja i definiranja statusa podataka podržane su definiranjem entiteta koji su vezani uz odobrenja podataka o komponentama i dokumentima [Slika 4.14].



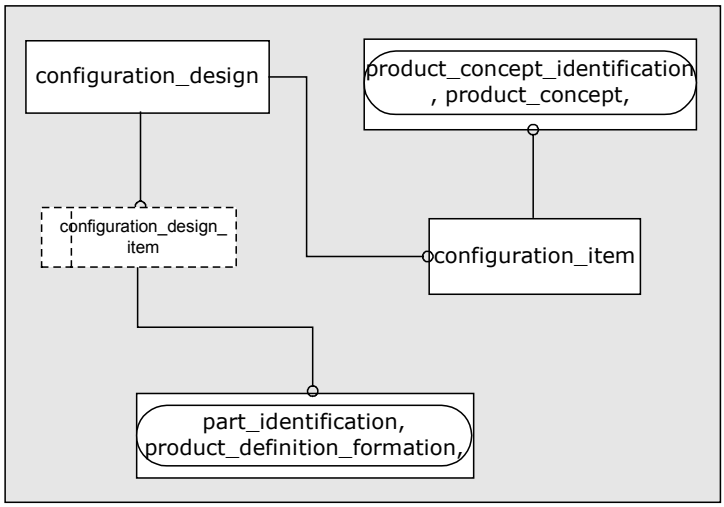
Slika 4.14: Dijagram entiteta za definiranje odobrenja

Osnovni entitet ove grupe entiteta je *approval* koji opisuje stanje ili status ispravnosti podataka o proizvodu. Entitetom *applied_approval_status* se pojedinom podatka pridružuje određeni stupanj odobrenja, a *approval_person_organization* definira tko je odgovoran za pojedino odobrenje. Važna karakteristika je i entitet *approval_date_and_time* koji specificira trenutak kad je odobrenje definirano.

4.3.9 Proces konfiguriranja

PDM shemom pružena je podrška i konceptualnoj definiciji strukture konfigurabilnih proizvoda, koja definira konfigurabilni proizvod iz točke gledanja kupca ili odjela prodaje. Obično takav pogled zahtijeva konfigurabilni proizvod koji se ovisno o zahtjevima i željama kupaca isporučuje u različitim varijantama. Identifikacija varijanti podržana je s dva osnovna koncepta: identifikacijom fiksnih konfiguracija proizvoda koje se kao takve nude kupcima i identifikacijom koja podržava definiranje proizvoljnih konfiguracija i njima pripadajućih fizičkih komponenti prema specifičnim zahtjevima kupaca. U konkretnom slučaju zanimljivija je druga mogućnost koja eksplicitno definira pojedinu varijantu s prikazom njezine kompozicije [Slika 4.15].

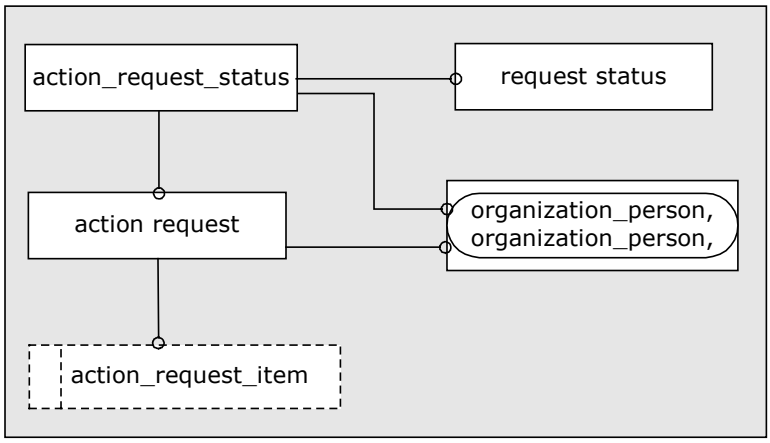
Pojedina konfiguracija definirana je entitetom *configuration_item* koji identificira pojedinu konfiguracijsku varijantu. Veza između pojedine konfiguracije i određene revizije komponente definirana je entitetom *configuration_design*.



Slika 4.15: Dijagram entiteta za definiranje konfiguracijskih koncepata

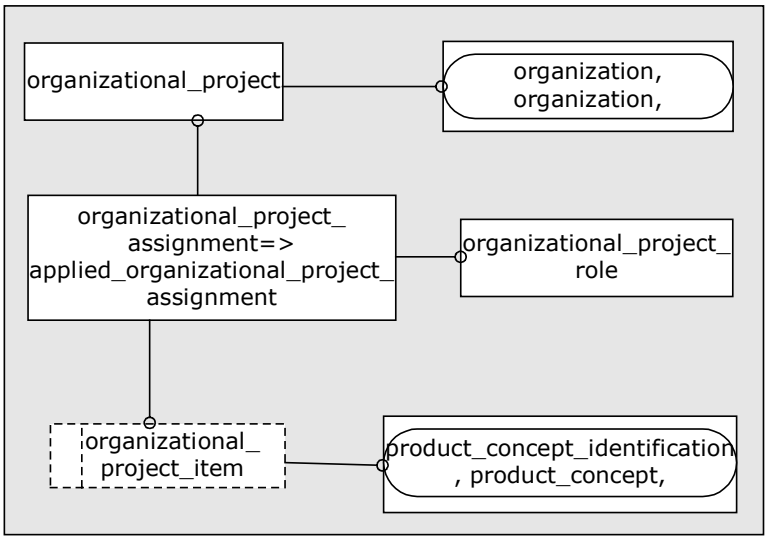
4.3.10 Upravljanje poslom i projektima

Područje upravljanja poslom unutar razvojnog tima uključuje opisivanje inicijalnih zahtjeva za proizvod, zahtjeva za promjenama kao i zahtjeve za određenim akcijama koje treba provesti s podacima u razvojnoj fazi. Prethodno opisani segmenti podržani su s tri koncepta: radni zahtjev, redoslijed rada i aktivnosti. Radni zahtjev u shemi je prikazan entitetom *action_request* kojim određeni korisnik traži od drugog korisnika određenu akciju. Pri tome se status izvršenja pojedina akcije prati entitetom *action_request_status* pomoću kojeg je moguće pratiti faze izvođenja akcije.



Slika 4.16: Dijagram entiteta upravljanje poslovima

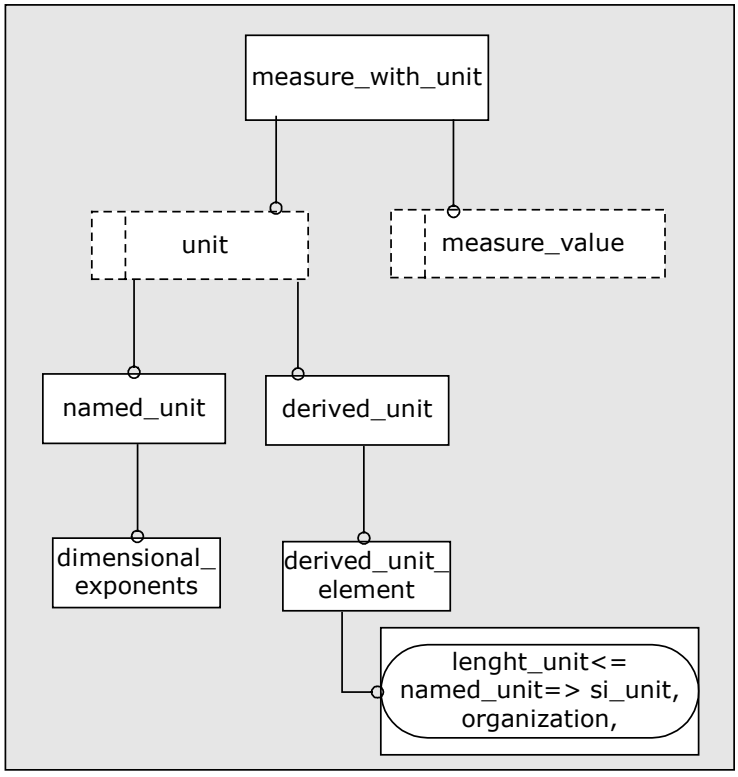
U dodatku ovom segmentu upravljanja je i definiranje struktura za prikaz podataka vezanih uz projekte kao nositelje posla, definiranje odgovornosti za projekte i veze između projekata [Slika 4.17]. Osnovni entitet upravljanja projektima je *organizational_project* koji je odgovoran za identifikaciju projekta za kojeg je odgovorna osoba ili grupa osoba (organizacija). Vezu projekta i podataka o proizvodu definira entitet *applied_organizational_project_assignment*. Veza između dva projekta definirana je entitetom *organizational_project_relationship*. Ova relacija služi nam za podjelu nekog kompleksnog projekta u podprojekte.



Slika 4.17: Dijagram entiteta za identifikaciju projekata

4.3.11 Jedinice i mjere

Mjere izražene u jedinicama su specifikacija fizičke kvantitete definirane ISO31 standardom. PDM shema dozvoljava specificiranje različitih mjera, odnosno izražavanja kvantitete kao na primjer u definiranju svojstva mase komponente ili u definiranju relacija među komponentama (broj istih komponenti u nadređenoj komponenti). Definiciju mjere nam daju dvije komponente: vrijednost i jedinica [Slika 4.18].



Slika 4.18: Dijagram entiteta za identifikaciju jedinica mjere

PDM shema pri tome podržava primjenu:

- Osnovnih jedinica definiranih SI sustavom prema ISO1000;
- Konvertiranih jedinica koje se dobivaju iz osnovnih pomoću faktora konverzije;
- Izvedenih jedinica koje se dobivaju izrazima;

Osnovni entitet sheme je entitet *measure_with_unit* koji je specifikacija fizičke kvantitete. Definicija jedinica je višestruka, i osniva se na sedam osnovnih jedinica definiranih SI sustavom preko entiteta *dimensional_exponent*. Pomoću tog entiteta se definira *named_unit* koji može predstavljati jednu od gore navedenih: osnovu jedinicu prema SI sustavu, konvertiranu jedinicu ili jedinicu mjere ovisnu o kontekstu. Izvedene jedinice definiraju se entitetom *derived_unit*, na primjer [N/mm^2]. Entitet *derived_unit_element* predstavlja asocijaciju jedinice s eksponentom. Najbolje ćemo to objasniti pogledamo li primjer N/mm^2 koja je kao što smo rekli izvedena jedinica. Ona se sastoji od dva elementa, jedinice *Newton*-a koji ima eksponent 1.0 i milimetra s eksponentom -2.0.

5

XML TEHNOLOGIJA

U petoj je glavi analizirano područje XML tehnologije. Uvodni dio glave donosi pregled povijesti razvoja XML standarda te definiciju osnovnih pojmova. Glavni dio glave je posvećen opisivanju uloga u kojima se XML tehnologija danas primjenjuje, te osnovnih značajki XML-a. Na kraju je objašnjena veza XML-a i STEP standarda, te primjena XML tehnologije u istraživanju koje je prikazano u ovome radu.

5.1 Općenito o razvoju XML-a

Usporedo s rastom i razvojem računalnih mreža, količina informacija koje su nastale kao proizvod rada u takvoj mrežno-orijentiranoj okolini se također povećavala, do točke kad je ljudima postalo nemoguće efektivno upravljati tim informacijama. Podaci govore da je negdje oko 1997. godine, količina podataka sadržanih u elektronskom obliku prerasla ukupni broj riječi svih do tada napisanih i objavljenih radova na tradicionalnom mediju (papir). Još je zanimljiviji podatak da taj broj teži k udvostručenju svakih 18 mjeseci. Posebno problematična iskustva s prevelikim brojem informacija zabilježena su na Internetu kao danas najvećoj računalnoj mreži. Jedan od najvažnijih dijelova Interneta je www (*eng. World Wide Web)*, koji predstavlja sustav međusobno povezanih sadržaja koji mogu biti HTML stranice, serverske stranice sa programskim skriptama i sličnim resursima, koje su ujedno i glavni nositelji informacija koje cirkuliraju globalnom mrežom [61], [62]. To je glavni razlog težnji k razvoju standardiziranih mehanizama za kreiranje i pretraživanje hijerarhijski strukturiranih podataka i dokumenta, te načina za njihovo prikazivanje i kontrolu što je glavna svrha grupe tehnologija koje su standardizirane pod

zajedničkim nazivom XML specifikacije. U nastavku glave dati će se kratki pregled razvoja i glavnih karakteristika koje definiraju tu grupu tehnologija.

5.1.1 Povijest XML-a

Pogledamo li na trenutak u prošlost Internet je izum ili zamisao jednog čovjeka, *Tima Barnes-Leea*, koji je u trenutku nastajanja Interneta radio kao programer u CERN-u, laboratoriju za visoko-energetska istraživanja u Ženevi. Njegova zasluga u nastajanju Interneta je u kreiranju osnovnog protokola za razmjenu informacija putem globalne mreže HTTP-a (*eng.* ***H****yper****T****ext* ***T****ransfer* ***P****rotocol*) i HTML-a (*eng.* ***H****yper****T****ext* ***M****arkup* ***L****anguage*) osnovnog jezika koji se koristi za definiranje oblika tih informacija. Cijela ideja razmjene informacija putem globalne mreže, potekla je od grupe znanstvenika u istom laboratoriju, koji su imali potrebu spremiti i prenositi izvještaje, sažetke, gotove znanstvene radove koje su producirali, na način koji bi omogućio njihovo lako referenciranje i pregledavanje od strane drugih znanstvenika i zainteresiranih organizacija. Na taj bi se način omogućilo različitim ljudima međusobno povezivanje pojedinih dijelova sadržaja stranica tih dokumenta, što je ostvareno prije navedenim protokolom i jezikom.

Osnovni doprinos *Barnes-Lee*-eve ideje protokola i jezika je u tome što je to omogućilo definiranje prvih preglednika takvih dokumenta, kao jednostavnih aplikacija koje su se zajedno sa specifikacijom protokola, jezika besplatno distribuirale zainteresiranim istraživačima. To je naravno ubrzalo razvoj aplikacija i alata za podršku. Znanstvenici u CERN laboratoriju su do tada u tu svrhu uglavnom koristili SGML (*eng.* ***S****tandard* ***G****eneralized* ***M****arkup* ***L****anguage*) jezik za opisivanje i spremanje svojih radova, izvještaja, disertacija i ostale dokumentacije. Pogledamo li u korijene SGML-a moramo se vratiti u '60-te godine prošlog stoljeća, kad je u laboratorijima IBM-a nastao GML (*eng.* ***G****eneralized* ***M****arkup* ***L****anguage*), kao pomoć u organizaciji ogromne količine dokumentacije koju je ta kompanija proizvodila. 1978, je Američki nacionalni institut za standarde (ANSI), na osnovi GML-a stvorio standard nazvan GCA 101-1983. Šest godina kasnije, nakon što se GCA pokazao uspješnim u demonstriranju mogućnosti manipuliranja dokumentima, internacionalna organizacija za standarde (ISO) je započela rad na globalnoj verziji , produkt čega je nastala specifikacija SGML-a, kao ISO 8879:1986 standard, 1986. godine.

Bartners-Lee je kreirao HTML kao instancu SGML-a na način da je iskoristio njegova pravila za kreiranje tagova (*eng. tags*), odnosno delimitera između kojih podaci moraju biti upisani, a oni određuju na koji se način interpretiraju. No pravu revoluciju HTML je doživio uključivanjem grafičkog prikaza, odnosno definiranjem tagova za prikaz slika i razvijanjem preglednika, što je omogućilo uključivanje i pregledavanje slika kao sastavnih dijelova dokumenata.

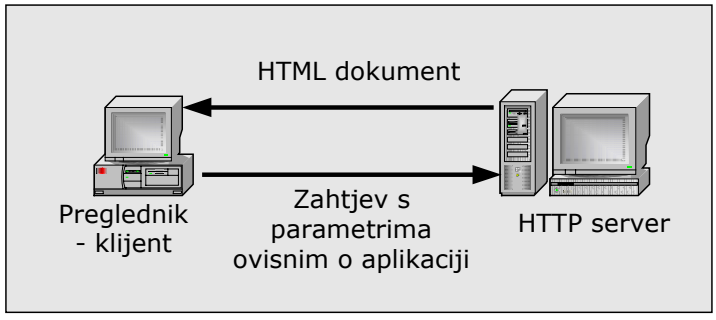
Naglim razvojem Interneta, međunarodni konzorcij koji se brine za njegovu standardizaciju W3C *(eng.* ***W****orld* ***W****ide* ***W****eb* ***C****onsortium*, http://www.w3.org), je započeo proces razvoja proširenog jezika za definiranje sadržaja dokumenta koji bi

kombinirao fleksibilnost i snagu SGML-a sa sve raširenijom prihvaćenošću HTML-a. Tako je nastao novi jezik, XML (*eng. e**X**tensible **M**arkup **L**anguage*) kao podskup SGML-a. Korištenje SGML kao početne točku, omogućilo je razvojnim timovima jednostavnije prihvaćanje i prilagodbu na novopredloženi standard. Namjera u razvoju XML-a je bila pojednostavljenje SGML-a u području čitanja i zapisivanja strukturnih dijelova dokumenta korištenjem jednostavnih i lako dostupnih alata, ali isto tako i jednostavno programsko procesiranje dokumenta i razmjena podataka, što je proširilo mogućnosti HTML-a. U dodatku ovome se mora naglasiti da se XML oslanja na postojeće Internet protokole i programske aplikacije za lako procesiranje i prijenos podataka tim medijem. Bitno je također naglasiti da kao podskup SGML-a, XML omogućava i povratnu kompatibilnost sa sustavima koji koriste taj jezik, tako da podaci definirani u XML-u mogu biti korišteni i u takvim sustavima, štedeći na taj način novac i resurse potrebne za konverziju. Glavna prednost XML-a je u mogućnosti formalnog opisivanja sintakse strukture podataka koju definiramo i razmjenjujemo s drugim korisnicima.

XML 1.0 specifikacija je postala službenom preporukom W3 konzorcija 1998. godine i od tada je mehanizmi za opis podataka implementirani u XML omogućuju novi način dijeljenja informacija putem Interneta. Najveći doprinos tome je svojstvo javnog standarda koji se neprestano nadograđuje te omogućuje razmjenu podataka između različitih platformi. Jednostavno rečeno možemo završiti ovaj pregled tvrdnjom da je XML danas standardni format za prikaz i prijenos tekstualnih podataka koji se često opisuju i kao ”ASCII za web”.

5.1.2 Arhitektura web aplikacija: prošlost i budućnost

Prije nego se detaljnije posvetimo osnovnim značajkama XML-a, posvetimo nekoliko rečenica uklapanju XML-a u arhitekturu danas rastuće domene web aplikacija. Prva generacija web aplikacija slijedila je tradicionalni klijent-server model arhitekture softvera. Klijentske aplikacije u tom modelu su mrežni preglednici koji djeluju na zahtjev korisnika. Objekti korisničkog sučelja preglednika prosljeđuju zahtjev za određenim sadržajem HTTP serveru [Slika 5.1]. Takav zahtjev može sadržavati mnogobrojne parametre i njihove vrijednosti koji ga dodatno definiraju. Mogući parametri moraju unaprijed biti određeni aplikacijom i poznati klijentu da bi mogli biti uključeni u zahtjev za određenim sadržajem.

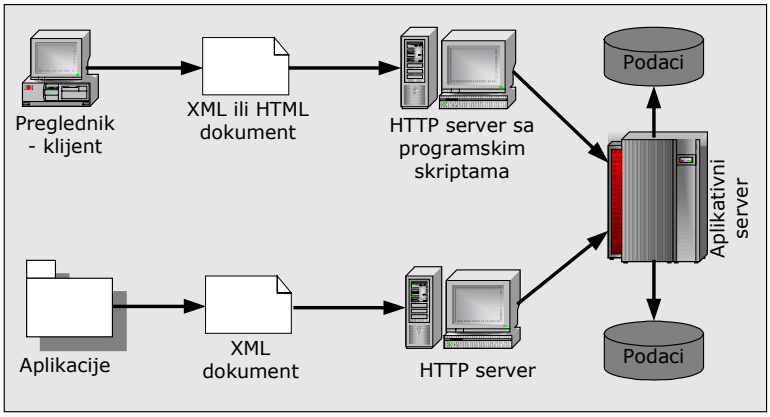


Slika 5.1: Klasična arhitektura web aplikacija

U ovakvom okruženju, server ispunjava zahtjeve korisnika kreiranjem dinamičkih HTML dokumenta korištenjem odgovarajućih serverskih skripti. Skripte služe za pristup bazi podataka i/ili procesiranju tih podataka. Ovakva arhitektura međutim je u praksi pokazala određena ograničenja:

- ograničenje na web preglednike kao klijente,
- ne postoje skripte za komunikaciju s drugim aplikacijama,
- sav sadržaj se prikazuje u HTML formatu, što onemogućava klijentu bilo kakvo naknadno procesiranje te ograničava korisnika na pogled koji je isporučio server u trenutku ispunjavanja zahtjeva,
- ako korisnik želi drugi tip prezentacije potrebno je pokrenuti novi zahtjev serveru.

Obzirom na navedena ograničenja, dodavanje funkcionalnosti XML tehnologije web aplikacijama, donosi višestruke prednosti u njihovom korištenju [Slika 5.2]. Kao prvo, klijenti, koji mogu biti preglednici ili druge aplikacije, mogu poslati XML dokument kao zahtjev serveru. Parametri koje XML dokument može sadržavati nadmašuju mogućnosti klasičnog pristupa. Struktura zahtjeva može biti formalno specificirana korištenjem standardnih mehanizama koje podržava server, te omogućava klijentu verifikaciju zahtjeva prije njegovog slanja. Kad je zahtjev zaprimljen, server ga može procesirati na način sličan klasičnim web aplikacijama ili uz proširenje tog seta sposobnosti. Podaci kojima server manipulira također su prikazani u XML obliku te je vrlo lako povezati dokumente koji se nalaze na nekoliko različitih servera ili transformirati podatke iz jednog oblika u drugi ako je postavljen takav zahtjev.



Slika 5.2: Arhitektura web aplikacija korištenjem XML-a

Koje su prednosti koje nam donosi ovakav pristup? Kao prvo, nema više ograničenja na web preglednike kao klijente. XML definira strukturirane podatke kojima se lako manipulira. Isti podaci mogu biti iskorišteni za prezentaciju ili kao subjekt naknadnog procesiranja na klijentskoj strani. To dovodi do buduće vizije u kojoj bi svaki element procesiranja na mreži servera, klijenta i aplikacija koristio iste mehanizme za razmjenu podataka – XML tehnologiju. Ovdje je potrebno naglasiti još jednom da su

ti isti mehanizmi podržani na svim računalnim platformama. Razvojni timovi će biti u mogućnosti koristiti podatke sa netradicionalnih medija kao i podatke s drugih servera za zadovoljavanje zahtjeva korisnika, što web aplikacije pomiče iz tradicionalne klijent-server orijentirane tehnologije u potpunu višeslojnu (*eng. multi-tier*) aplikativnu okolinu.

5.1.3 Uloge XML-a

Kao što je već rečeno u prethodnim poglavljima XML predstavlja specifikaciju za označavanje i definiranje pojedinih dijelova sadržaja dokumenta pomoću tzv. tagova *(eng. tags)*. Tagovi definiraju kontekst sadržaja kojeg omeđuju korištenjem XML sintakse, omogućujući na taj način definiranje kompleksnih podatkovnih struktura. Gdje i kako XML može biti iskorišten? Ovog trenutka može se reći da se XML koristi na mnogo različitih načina te mnogi proizvođači razvijaju alate za njegovo korištenje. Osnovne prednosti XML-a mogu se definirati kroz slijedeće navode:

- XML funkcionira kao format za razmjenu strukturiranih podataka.
- XML je fleksibilan te je jednostavno i brzo moguće dodati i maknuti elemente u strukturu dokumenta.
- XML dokument predstavlja hijerarhijsku strukturu podataka na način s kojim su računalni korisnici dobro upoznati iz osnovnog koncepta manipuliranja računalnim podacima kroz direktorije i datoteke.
- XML je jezik transformacije: moguće je filtrirati podatke koje sadrži XML dokument, promijeniti ih u drugi format ili ih poredati korištenjem različitih kriterija.
- XML je međunarodni standard, što znači da se može koristiti kao medij komunikacije između raznih operacijski sustava odnosno računalnih platformi.
- XML podržava osnovne postavke objektno-orijentiranog programiranja.

Osnovni nedostaci XML tehnologije su:

- XML nije namijenjen upotrebi kao baza podataka s velikim kapacitetom. Nije iskoristiv za pretraživanje i korištenje više tisuća zapisa (od jednom).
- XML je ipak kompleksniji od HTML-a, što znači da je i njegovo korištenje kompliciranije.
- XML nije proceduralni jezik.
- XML je još u fazi razvoja, te kao i sa svim tehnologijama koje se razvijaju, postoji određeni rizik u njegovom korištenju.

U daljnjem tekstu dat će se kratki pregled osnovnih načina korištenja XML-a.

XML kao format dokumenata

SGML-a, koji je prethodnica XML-a, se koristi kao meta-jezik za definiranje sadržaja

dokumenta, sposoban za kreiranje vokabulara za opisivanje strukture gotovo svih tipova dokumenta. Jasno je da je proces ljudskog razmišljanja uglavnom nelinearan, ali dokumenti koje kreiramo uglavnom imaju linearnu strukturu. (Na primjer, knjiga je u biti linearna pojava, bez obzira na nelinearni tok misli koji je doveo do njenog stvaranja). XML je inicijalno nastao kao sofisticiraniji jezik za opisivanje sadržaja dokumenata prije svega tekstualnih dokumenta, prilagođen podršci za HTML odnosno prikazu na Internetu, međutim taj njegov aspekt se sporo razvija zbog slijedećih problema:

- Proizvođači pretraživača nisu još ugradili kompletnu podršku za korištenje XML- u svoje preglednike prema za to predviđenom standardu.
- Za kreiranje kompletnog jezika za opis strukturiranih dokumenta, potrebno je dokument podijeliti u mnogostruke logičke cjeline, što je često mnogo teže učiniti s dokumentima nego s dobro definiranim strukturama podataka.
- Uvijek postoji više od jednog načina za prikaz strukture dokumenta koje je prikladno određenom kontekstu upotrebe dokumenta, a određivanje tog najboljeg načina može biti vremenski zahtjevno s mnogostrukim iteracijama.
- Prikazivanje kompleksnih dokumenta uglavnom sadrži mnogostruke zahtjeve, što znači razvijanje kompleksnih mehanizama.

Opisana problematika je dovela do toga da je u počecima razvoja standarda fokus istraživača bio na razvoju XML-a kao zamjene za HTML, dok je danas fokus na različitim načinima korištenja XML-a. Ono što karakterizira dokumente koji se osnivaju na XML-u je da su oni strukturirani: elementi dokumenta mogu biti poredani u bilo kojem redoslijedu, a tekst se može slobodno isprepletati s entitetima koji imaju kompleksni set atributa s različito definiranim vrijednostima. U SGML-u, veze između pojedinih elemenata dokumenta definiraju se u posebnom dokumentu DTD (*eng. **D**ocument **T**ype **D**efinition)*, a što je također naslijeđeno u XML-u.

XML kao format za upravljanje dokumentima

Da bismo jasno razlučili razliku između formata dokumenta i formata za upravljanje dokumentima poslužit ćemo se usporedbom s dobro poznatom sintagmom datoteka i direktorija (mapa). U računalnom smislu direktorij u osnovi predstavlja hijerarhijsku strukturu, kojom se korisnici služe za smisleno i organizirano pohranjivanje datoteka. Strukturu direktorija korisnici pretražuju u svrhu dobivanja informacije o računalnom sustavu, strukturi datoteka koje sadrži, informacijama o korisnicima itd. Teoretski gledano, hijerarhijska struktura direktorija može se prikazati korištenjem XML dokumenta. XML se može iskoristiti za definiranje jedne ili više ″tematske mape″ u prostoru datoteka kojima se želi upravljati. Prednost ovakvog pristupa je u organiziranju datoteka odnosno dokumenta koje one predstavljaju, prema njihovom sadržaju, a ne fizičkoj lokaciji. Osnovna svrha te strukture u ovom slučaju je kreiranje mape relacija između dokumenta (ili njihovih poddokumenta) u virtualnom prostoru. Ovaj pristup uklanja potrebu za redundancijom dokumenta u slučajevima

kada je potrebno imati fizički dokument u više tematskih mapa, jednostavnim prikazivanjem tih relacije u različitim XML dokumentima.

XML kao format za razmjenu podataka

HTML govori vrlo malo o strukturi podataka koje prikazuje, odnosno u najboljem slučaju daje nam mogućnost zapisa strukturiranih podataka u regularne tablice koje imaju zaglavlja - retke sa nazivima pojedinih kolona i retke u koje se upisuju podaci. Pri tome je bitno naglasiti da nam HTML ne daje nikakvu vezu između naziva kolona i podataka u ovakvoj organizaciji. Strukturiranjem podataka korištenjem XML-a dajemo svrhu klasičnoj tablici koja odjednom postaje vrlo jasan objektni model. Ovakav pristup uvelike olakšava posao programerima. Naime većina programskih jezika ima osnovne strukture podataka koji su uključeni u specifikaciju jezika – vezane liste, vektore, kolekcije, ali svi oni uglavnom predstavljaju linearne strukture. Problem prikaza strukture stabla je u ne postojanju konzistentne metodologije za kreiranje takvih struktura, te je potrebno programirati vlastite tehnike za popunjavanje stabla, navigaciju, pretraživanje elemenata itd.

XML je na ovom području ponudio dvije stvari: mehanizme za kreiranje i pretraživanje hijerarhijskih podatkovnih stabala (XML parseri) i način prikaza hijerarhije na konzistentan način i način pogodan za kontroliranje. Jedna od karakteristika najvećeg broja baza podataka je mogućnost spremanja podatka u tablice, koje sadrže informacije o tipovima podataka, ograničenja o mogućim vrijednostima svakog polja, mogućnosti da pojedino polje bude prazno itd. XML omogućuje programerima postavljanja karakteristika pojedinih elemenata u njihovim XML dokumentima na način koji je sličan bazi podataka.

XML kao jezik za transformaciju podataka – XSL

Da bi se XML dokument mogao transformirati u oblik koji je pogodan za prikaz podataka koje sadrži, kao podskup XML tehnologije postoje mehanizmi koji omogućavaju takvu konverziju, a definiraju se korištenjem XSL-a (*eng. e**X**tensible **S**tyle **L**anguage*). Taj skup mehanizama je nastao iz potrebe za mogućnošću pregrupiranja podataka i njihovog filtriranja. Jezik podržava transformaciju XML dokumenta u druge oblike koji mogu biti i drugog tipa, npr. HTML, SQL skripte, tekstualni dokumenti ili RTF dokumenti. Obzirom da postoje još neke klase dokumenta za koje bi bilo teško napisati XML filtre, možemo reći, da teoretski gledano, XSL bi trebao biti u mogućnosti transformirati podatke koje sadrži XML dokument u bilo koji oblik.

XSL skripte mnogu biti vrlo kompleksne te je to jedan od razloga zašto su one jedan od najzahtjevnijih aspekata XML standarda. XSL omogućuje ugradnju logike u elemente XML dokumenta, što znači da je moguće ugraditi jednostavne operacije poput sumiranja vrijednosti ili provjeru ispravnosti XML strukture. Zajedno sa odgovarajućim serverom, XML proširuje potencijal XML-a iz jednostavnog jezika za prikaz podataka do događajima progonjenog programskog jezika.

XML kao programski jezik

Pomoću XML specifikacije moguće je kreirati objekte implicirane organizacijom podatkovne strukture ili je moguće koristiti neke od dodataka za eksplicitno definiranje objekata, tipova podataka, kolekcija i ostalih entiteta koji su vezani uz objektno orijentirani način programiranja. Kao što je već rečeno u prošlom poglavlju, pomoću XSL-a je XML napravio korak od transformiranje podataka iz jednog oblika u drugi, što može značiti i vezanje podataka na elemente grafičkog sučelje što se i radi kod većine aplikacija s grafičkim sučeljem.

Važno je ipak naglasiti razliku između XML-a i proceduralnih jezika, XML ipak nije stvarno pokretan događajima. Iako se događaji mogu deklarirati u XML strukturi ipak postoji potreba za mehanizmima koji će povezati te predviđene događaje i stvarne događaje koji ih aktiviraju prilikom korištenja.

5.2 Arhitektura i način korištenja XML-a

Nakon pregleda povijesnog razvoja i uloga XML-a, ovaj pregled će se nastaviti definiranjem osnovnog vokabulara standarda, te pregledom glavnih značajki uključenih u pružanje osnovne funkcionalnosti.

5.2.1 Osnovni rječnik XML-a

Jedna od najznačajnijih karakteristika XML je njegovo svojstvo proširljivosti. HTML-om je vrlo jednostavan jezik za opisivanje sadržaja dokumenata i njihovu razmjenu putem Interneta, pomoću unaprijed definiranih tagova za formatiranje. U suprotnosti s tim, svojstvo proširljivosti XML-a očituje se u mogućnosti kreiranja vlastitih setova tagova, koji mogu biti specifični za pojedinu kompaniju, znanstvenu disciplinu ili neku drugu domenu. Svakoj individui koja koristi XML, omogućeno je definiranje vlastitog seta tagova na temelju sintakse standarda. Sve što je potrebno za razmjenu informacija između različitih korisnika i aplikacija je dijeljenje tog specifičnog vokabulara s ostalim korisnicima.

Tako je do sada razvijeno nekoliko specifičnih vokabulara. U znanstvenom području tu su npr. CML (*eng. **C**hemical **M**arkup **L**anguage* – http://www.xml-oml.org), specijalno kreiran za kemijske znanosti, zatim MathML (*eng. **M**athematical **M**arkup **L**anguage* - http://www.w3.org/Math/) za jednostavniju razmjenu matematičkih izraza, za biokemijske znanosti BSML (*eng. **B**ioinformatic **S**equence **M**arkup Language*), itd.

U području poslovne primjene, najvažniji predstavnici su iz sfere elektronskog trgovanje kao što je cXML (*eng. **C**omerce**XML*** – http://www.cxml.org/home) i IFX (*eng .**I**nteractive **F**inancial **E**xchange* – http://www.ifxforum.org). Razvoj naravno nije mogao zaobići ni područje računalnih tehnologija, te je tu zabilježen najveći napredak na polju opisivanja kompleksnih web site-ova kao SGF (*eng. **S**tructured **G**raph **F**ormat*SGF – http://www.isl.hiroshima-u.ac.jp/project/SGF/indeks.html), te

baza podataka XMI (*eng.* ***X****ML* ***M****etadata* ***I****nterface* – http://www-4.ibm.com/software/ad/features/xmi.html).

Vokabular kojim se opisuje struktura dokumenta i podatka svih navedenih područja, opisuje se korištenjem sintakse XML-a [Tablica 5-1].

Tablica 5-1: Osnovna sintaksa XML-a

Sintaksa	Upotreba
<ime_oznake>	Početni tag elementa
<ime_oznake atribut=”vrijednost”>	Početni tag s atributom
</ime_oznake>	Završni tag elementa
<ime_oznake/>	Prazan element
<?xml version=”1.0”?>	XML deklaracija
<?ime_instrukcije?>	Instrukcija za procesiranje
<!--string-->	Komentar
<![CDATA[string…]]>	Odjeljak s podacima koji se ne parsiraju
<!DOCTYPE string…>	Deklaracija tipova elemenata koji su uključeni u dokumentu – referenca na vanjski DTD, koji definira vokabular dokumenta

Prikaz dobro strukturiranog XML dokumenta prikazan je slikom [Slika 5.3].

```
<? xml version="1.0"?>
<! DOCTYPE data SYSTEM "product.dtd">
<! -- Ovo je primjer XML dokumenta -->
<Product>
        <Id>#00012</Id>
        [...]
        <Documents>
                <DocReference>#00101</DocReference>
                <DocReference>#00102</DocReference>
                [...]
        </Documents>
        [...]
</Product>
[...]
<Document>
        <Id>#00101</Id>
        [...]
</Document>
<Document>
        <Id>#00102</Id>
        [...]
</Document>
```

Slika 5.3: Primjer dobro strukturiranog XML dokumenta

Pomoću sintakse kreiraju se osnovni gradivni blokovi XML dokumenta: instrukcije važne za procesiranje dokumenata, elementi, atributi, i komentari.

Instrukcije važne za procesiranje dokumenta

Instrukcije vezane uz procesiranje XML dokumenta, nalaze se uvijek na početku XML dokumenta te se definiraju između šiljatih zagrada i znak "?" kako je definirano slijedećim primjerom:

<? Ovo je primjer instrukcije za procesiranje ?>

Uloga ovih instrukcija je definiranje informacija koje izlazi izvan uobičajenih okvira XML strukture, a koje koriste alati koji procesiraju XML dokument. Na primjer, instrukcije za procesiranje možemo upotrijebiti za specificiranje XML seta koji se koristi prilikom definiranja XML podataka. Formalna XML struktura uvijek započinje s instrukcijom *<? xml version="1.0" ?>*, koja indicira da se radi o XML dokumentu, iako većina aplikacija za procesiranje neće generirati grešku ako ovaj redak nije uključen.

Elementi

Elementi su osnovni gradivni blokovi XML dokumenta te ih možemo usporediti s tagovima iz HTML-a, samo što ih ovdje definiraju korisnici ovisno o kontekstu u kojem koriste XML dokument. Elementi mogu sadržavati ostale jedinice, kao što su podaci, reference, instrukcije za procesiranje, itd. Elementi se definiraju između šiljatih zagrada unutar kojih najprije slijedi jedna riječ koja definira tip elementa, te nijedna ili više kombinacija "ime=vrijednost" koji se nazivaju atributi, što je definirano slijedećim primjerom:

<productContext type="1125" processed="false">

Bitno je naglasiti da svaki element mora imati završetak koji je sličan formatu HTML tagova, na način koji je ilustriran slijedećim primjerom:

</productContext>

Bitno je naglasiti razliku u odnosu na klasični HTML koji je općenito poznat, te je u njemu dozvoljeno da pojedini elementi ne moraju imati završni dio. U XML-u bez obzira da li element sadrži tekst ili druge elemente mora imati dio koji ga zatvara. Unutar početnog i završnog taga elementa, nalazi se podatak ili drugi elementi što definira informacija koju element sadrži i daje joj kontekst unutar strukture dokumenta.

Atributi

Atribut je par "ime=vrijednost" koji je pridružen određenom elementu. Ako pogledamo primjer iz prethodnog odlomka, *type* i *processed* su imena atributa, dok su vrijednosti između znakova navoda vrijednosti tih atributa. Atributi služe kao nositelji dodatnih informacija o elementima za razliku od informacija koje elementi sadrže.

Komentari

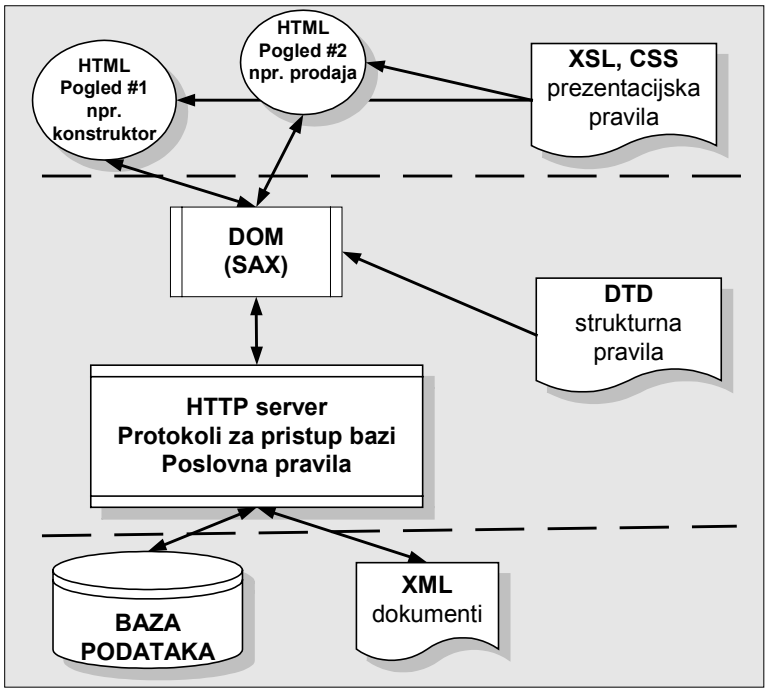
Komentari u XML strukturi su jedinice koje se dodaju da bi se dodatno razjasnila struktura, te se definira između šiljatih zagrada i znakova ”!-- ” ”-- ” što je prikazano na slijedećem primjeru:

<! -- Ovo je jednostavan komentar -->

Bitno je naglasiti da komentari ne mogu uključivati druge komentare unutar sebe.

5.2.2 Glavne značajke i korištenje XML arhitekture

Nakon što je prikazan vokabular XML sintakse, pogledajmo pobliže značajke i specifikacije koje čine funkcionalnost XML-a. Korištenje XML-a se temelji na pretpostavkama troslojne arhitekture [Slika 5.4] pomoću koje se strukturirani podaci čuvaju odvojeno od pravila za njihovo manipuliranje i definiranje načina prikaza tih podataka korisnicima. Pojedini elementi XML arhitekture objašnjeni su u odlomcima koji slijede.



Slika 5.4: Arhitektura i korištenje XML tehnologije

DTD i XML Shema

Da bi se omogućio univerzalni način za komuniciranje između različitih aplikacija koje koriste XML dokumente i razumijevanje vokabulara u smislu *ovaj* element slijedi *taj* element, što znači *to*, XML specifikacija definira mehanizam pod nazivom DTD (*eng. **D**ocument **T**ype **D**efinition*). DTD koristi gramatiku jezika za specificiranje strukture elemenata pojedinog XML dokumenta te mogućih vrijednosti koje elementi mogu poprimiti. DTD formalno i precizno dokumentira vokabular dokumenta, te se najčešće koristi za provjeru sintaktičke ispravnosti XML dokumenta. Obzirom da je

DTD ostavština SGML-a donosi i određene nedostatke. Zbog toga je XML zajednica proširila standardne mogućnosti DTD-a sa mogućnošću definiranja i tipova podataka koji se koriste u dokumentu, približivši na taj način XML format načinu na koji klasični sustavi baza podataka pohranjuju informacije, definiranjem XML sheme. XML shema je u potpunosti napisana u XML-u te između ostalog omogućava eksplicitno definiranje ukupnog broja pojedinih elemenata koji se može pojaviti u strukturi te unaprijed vrijednosti za razliku od DTD-a.

DOM

Naziv DOM (*eng. **D**ocument **O**bject **M**odel*) dolazi iz konteksta web preglednika. Objekti kao što su prozori, dokumenti koji se prikazuju u njima, povijest prikazivanja pojedinih dokumenta, u preglednicima se razmatraju kao dijelovi preglednikog objektnog modela. Prilikom razvoja aplikacija za web koriste se standardizirani načini za pristup pojedinim elementima preglednika definirani W3C. DOM predstavlja definiciju neovisnu o programskom jeziku i platformi, te opisuje sučelje za pristup pojedinim objektima. U konkretnom slučaju to je način za navigaciju i manipuliranje sadržajem i strukturom XML dokumenata. Koncept DOM-a omogućuje naime tretiranje XML dokumenta kao kolekcije individualnih objekata, za koje je točno definirano što oni mogu raditi, kao komuniciraju međusobno, kako ih je moguće adresirati. Bitno je naglasiti da DOM omogućuje modificiranje XML strukture i na strani klijenta i na strani servera. Osnovni nedostatak DOM pristupa je u vrlo velikim memorijskim zahtjevima, obzirom da odjednom kreira sve objekte koji su definirani XML strukturom, te kao takav i nije najbolja platforma za manipuliranje velikim XML dokumentima.

SAX

SAX (*eng. **S**imple **A**PI for **X**ML*) je alternativni način za procesiranje XML dokumenata. Za razliku od DOM-a koji čuva cjelokupnu sliku objekata i njihovih relacija u memoriji, SAX omogućuje selektivno čitanje dokumenata na način da se pozivaju određeni događaji definirani u pojedinim tagovima XML dokumenta. Ovaj pristup posebno može biti koristan kad je potrebno izvući potreban skup informacija iz XML strukture.

Transformacija XML-a

Već je rečeno da se XML može promatrati kao format dokumenta, hijerarhijski model za pohranu podataka ili medij za prijenos podataka preko mreže. Kako god gledamo XML dokument, on je kolekcija elemenata organiziranih prema određenoj shemi te je kao takav hijerarhijska struktura podataka. Međutim korištenje XML zahtjeva mogućnost prikaza različitih formata tih podataka, što se ostvaruje jezikom za transformaciju *eng. e**X**ensible **S**tylesheet **L**anguage* (XSL) koji je specificiran XML standardom. Transformacije XML dokumenta možemo uglavnom svrstati u tri kategorije: (i)strukturalne transformacije, gdje se jedan vokabular transformira u drugi; (ii)kreiranje dinamičkih dokumenta, što omogućuje korisnicima filtriranje,

pregrupiranje i sortiranje dijelova XML dokumenta; (iii)transformacija u drugi prikaz, kao što je npr. HTML, WAP.

5.3 XML i STEP – uloga XML tehnologije u radu

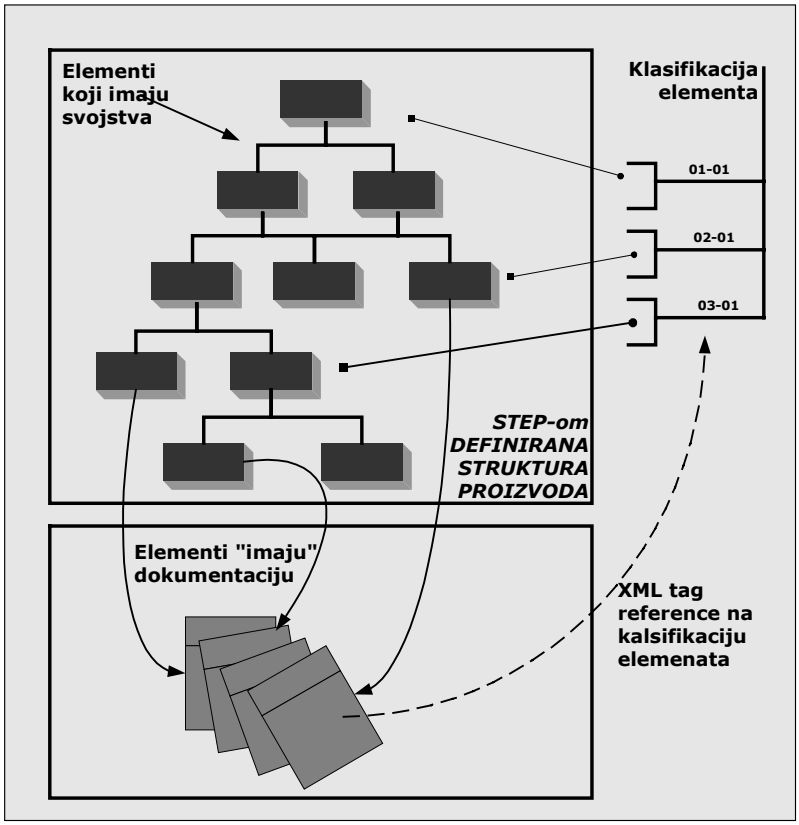
Iako je razvoj STEP i SGML (kao prethodnice XML-a) standarda započeo odvojeno, s različitim vizijama i različitim ciljevima, oni dijele zajedničko nastojanje za omogućavanjem razmjene informacija korištenjem platformski nezavisnih standardnih formata podataka. STEP je, kao što je opisano u četvrtoj glavi, u osnovi koncentriran na razmjenu podataka o proizvodu, te je njegov razvoj potaknut prije svega potrebom za razmjenom CAD modela između različitih CAD sustava. Za razliku od STEPA, kao što je rečeno u prethodnim odlomcima, razvoj SGML je započeo s ciljem razmjene strukturiranih tekstualnih podataka, kao što su npr. tehnički izvještaji. Ipak, već davno se primijetilo da su ova dva područja vrlo bliska i zajednički upotrebljiva. To je dokumentirano početkom 80-tih u nekim pokušajima uključivanja crteža spremljenih u IGES formatu i strukturiranog teksta u GML formatu u integrirana tehnička uputstva, koja su između ostalog uključivala i tablice s točnim dimenzijama i identifikacijskim brojevima pojedinih komponenti koje su dobivene iz baze podataka o proizvodu.

U zadnjem desetljeću, veza STEP-a i SGML-a je prerasla okvire samog uključivanja podataka o proizvodu u uputstva koja se tiskaju kao priručnici, zbog činjenice da se ne može zanemarivati veza proizvoda i dokumentacije koja je njegov ravnopravni dio [63]. Prednosti razmjene podataka koja se osniva na postavkama XML-a znače veliki korak k standardiziranju načina razmjene svih informacija o proizvodu definiranih pojedinim dijelovima STEP standarda, kroz medij XML-a [64], [65], [66]. Takva standardizacija doprinosi osnovnom cilju, a to je kao što je već naglašeno integracija svih podatka o proizvodu [Slika 5.5].

Timovi istraživača iz oba područja rade na potpunoj harmonizaciji ova dva standarda [67]. Harmonizacija bi trebala potvrditi novo značenje informacijskih objekata u kojem je struktura dokumenta slična ili istovjetna strukturi proizvoda, sa naglaskom na korištenje svih prednosti nove infrastrukture za njihovu razmjenu koju sa sobom nosi XML. Potpunom harmonizacijom ova dva standarda, očekuje se ostvarenje slijedećih prednosti:

- potpuna integracija dokumentacije o proizvodu u sustave za upravljanje podacima o proizvodu, npr. kreiranje konfigurabilne dokumentacije ovisno o specifičnoj konfiguraciji proizvoda,
- kreiranje repozitorija strukturiranih informacija svih vrsta, koji će biti lako dostupni za korištenje preko web preglednika. Informacije koje će postati dostupne na ovaj način, mogu uključiti korporacijske podatke, kreiranje kolekcija fizički razmještenih dokumenta i podataka, te na taj način olakšati održavanje dokumentacije s jednog mjesta.

- informacijski tok, tehnički i administrativni, će postati lakši za opisivanje, jednako kao i proces upravljanja svim podacima o proizvodu i njihovim relacijama.



Slika 5.5: Integracija STEP-a i XML-a

Važan korak k tome je napravljen definiranjem dijela STEP standarda sa oznakom ISO 10303-28:*"Product data representation and exchange: Implementation methods: XML representation of EXPRESS-driven data"* [68] kojim se definiraju preporuke za pretvaranje strukture podatka o proizvodu definirane STEP-om u definiciju elemenata strukture XML dokumenata.

Gledano iz perspektive ovog rada, odnosno razvoja sustava za razmjenu i upravljanje informacijama o proizvodu, integracija STEP specifikacije podataka o proizvodu i XML-a zajednički definira osnovnu semantiku metamodela podataka o proizvodu koji je opisan u prethodnoj glavi i tehnologiju za razmjenu tako strukturiranih podatka putem web aplikacija, što će biti opisano u slijedećoj glavi. Time se omogućava istovremeni rad različitih korisnika na istom projektu, fleksibilnija organizacijska struktura razvojnih timova, strukturiranje sadržaja tehničke dokumentacije kao dijela proizvoda i kreiranje distribuiranog virtualnog razvojnog okruženja za podršku razvoju proizvoda.

6

RAČUNALNA IMPLEMENTACIJA SUSTAVA

U šestoj je glavi predložena metodologija implementacije sustava. Uvodni dio glave donosi pregled i definiciju osnovnih pojmova objektno orijentiranih tehnika modeliranja računalnih sustava i programiranja. Glavni dio glave je posvećen definiranju troslojne arhitekture sustava koji se kako je predloženo realizira u obliku web servisa. Za svaku od tri razine arhitekture, (klijentsku razinu, razinu poslovne logike i razinu trajnog zapisa podataka) definirane su i opisane osnovne značajke.

6.1 Objektno orijentirano programiranje

Strojarski, građevinski i elektrotehnički inženjeri prilikom osmišljavanja sustava koje konstruiraju koriste različite tipove modela. Najčešće su to modeli strukture koji stručnjacima pomažu u vizualizaciji i specifikaciji osnovnih dijelova ciljanog sustava i definiranju veza između pojedinih elemenata sustava. Ovisno o području primjene, modeli mogu modelirati dinamički te omogućiti proučavanje ponašanja dijelova sustava. U modeliranju i izgradnji arhitekture računalnih aplikacija postoje dvije osnovne perspektive: proceduralna perspektiva i objektno orijentirana perspektiva.

Proceduralna perspektivu koristi tradicionalni način razvoja softvera. Prema ovom pristupu gradivni blokovi računalnih aplikacija su procedure ili funkcije. Ova perspektiva zahtijeva od razvojnih timova fokusiranje na kontrolu i dekompoziciju velikih i zahtjevnih algoritama koji se koriste u aplikacijama u jednostavnije. Osnovni nedostatak ovakvog pristupa je otežana implementacija naknadnih zahtjeva ili promjena. Objektno orijentirana perspektiva za razliku od proceduralne, kao temeljne gradivne blokove programskog sustava koristi objekte ili klase. Jednostavno govoreći, objekt je pojam, općenito definiran rječnikom prostora rješavanja

postavljenog problema, dok je klasa opis skupa zajedničkih objekata [69]. Svaki objekt određuju identitet (razlikuje ga od ostalih objekata), stanje (u osnovi definiraju ga podaci koji su vezani s objektom) i ponašanje (definira što i kako objekt radi). Razvoj objektno orijentiranih tehnika isprva je bio potaknut potrebom razumijevanja kompleksnih programskih sustava. U zadnjih nekoliko godina pozornost je usmjerena na proučavanje primjene objektnog pristupa na probleme analize i modeliranja pojava i elemenata iz stvarnog svijeta u raznim područjima pa tako i u području kojim se bavi znanost o konstruiranju.

U svrhu vizualizacije, specifikacije, konstrukcije i dokumentiranja objektno orijentiranih sustava razvijen je UML (*eng.* ***U****nified* ***M****odeling* ***L****anguage*) [69], [70], [71]. UML omogućuje pohranu, razmjenu i primjenu znanja u procesima rješavanja problema objektno orijentiranim metodama. UML ne propisuje nikakav određeni pristup rješavanju problema, nego se može prilagoditi svakom pristupu. U modeliranju računalne implementacije sustava za razmjenu i upravljanje s podacima o proizvodu koje je opisano u idućim poglavljima koristit će se rječnik i pravila UML kao osnova objektno-orijentirane pristupa.

6.2 Arhitektura računalne implementacije sustava

U okviru zadatka ovog magistarskog rada, potrebno je realizirati sustav, na način da se informacijama o proizvodu upravlja te se ono razmjenjuju, korištenjem globalne računalne mreže, Interneta. Kao što je već rečeno u prethodnoj glavi, XML tehnologija je u zadnjih nekoliko godina unaprijedila kreiranje heterogenih računalnih okruženja koja su namijenjena dijeljenju informacija iz različitih područja primjene računala putem globalne računalne mreže. Gledano iz tehničke perspektive, nove mogućnosti koje je donijelo doba Internetskih aplikacija nisu revolucija u distribuiranom korištenju računala, već predstavljaju novi korak u evoluciji mrežnih aplikacija. Napredak je realiziran korištenjem XML tehnologija, uz područje strukturiranog prikaza informacija, u području strukturiranog prikaza razmjene informacija među različitim tipovima klijenata. Zadnje dostignuće u tom području naziva se *"*web servis*"*. Web servisi se definiraju [72], [73], [74] kao distribuirane funkcionalne komponente koje su sposobne za međusobnu komunikaciju preko standardnih Internet protokola, te su kao takve ponuđene za korištenje u razvoju drugih aplikacija. Drugim riječima, web servisi su zamišljeni kao gradivni blokovi za kreiranje otvorenih, distribuiranih računalnih sustava čije mogućnosti razvojni timovi brzo i jednostavno implementiraju u vlastita rješenja.

Pogledajmo glavne značajke korištenja ovakvog pristupa. Prije pojave web servisa postojao je naglašen problem integracija različitih aplikacija nastajao uglavnom zbog različitosti u programskim jezicima i tehnologijama koje su korištene. Vjerojatnost da su dva poslovna sustava razvijena korištenjem istog programskog jezika je vrlo mala, a često je potrebno zajednički koristiti njihove komponente integrirajući ih na taj način u jedno poslovno rješenje. Da bi se web servisi primijenili za integraciju

različitih aplikacija, one trebaju biti prilagođene uporabi putem Interneta. Korištenjem standardnih protokola za razmjenu podataka putem Interneta, koji su neovisni o računalnoj platformi, web servisi u svojem radu prihvaćaju i realiziraju zahtjeve klijenata. Komunikacija klijenata i servisa odvija se razmjenom XML dokumenata. Primjena XML standarda omogućuje da razmjena podatka klijenata i web servisa ne ovisi o tome koja tehnologija je nositelj izvora podataka, te koja se računalna platforma nalazi u pozadini. Pri tome klijenti koji koriste web servise mogu biti različitih vrsta: uređaji za mobilnu komunikaciju, ručna računala, web preglednici, druge aplikacije, poslovni sustavi i drugi web servisi.

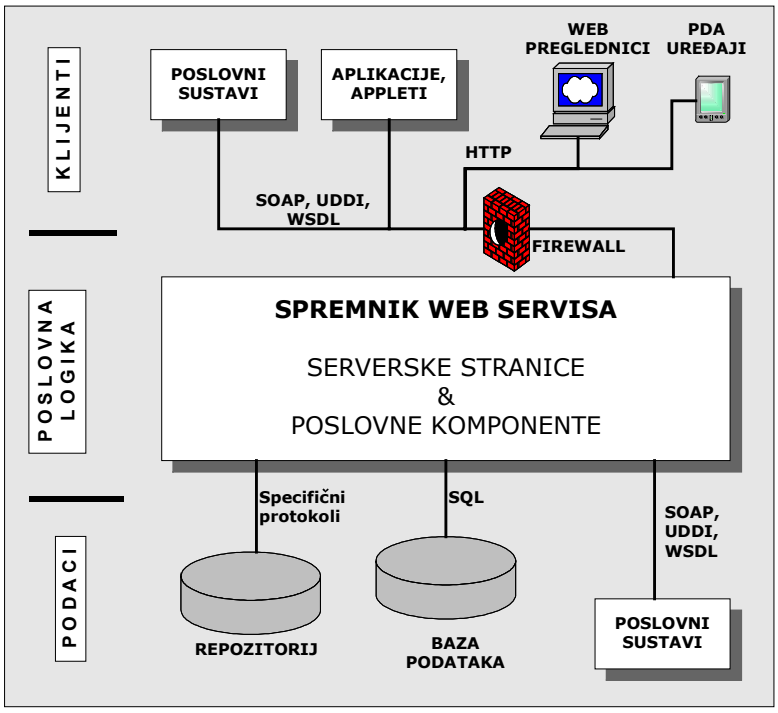
Vizija uporabe web servisa predviđa njihovo registriranje u privatnim ili javnim, poslovnim registrima, tako da kreatori servisa opišu komponente servisa, definiraju njihova sučelja, poslovnu logiku te uvjete korištenja. Mogući scenarij korištenja web servisa mogu se prikazati slijedećim koracima:

1. Kreator web servisa:
   - kreira osnovne komponente servisa, sklapa ih međusobno te ih priprema za korištenje, koristeći se programskim jezikom i platformom po vlastitom izboru.
   - definira i opisuje web servis korištenjem WSDL-a (*eng .**W**eb **S**ervice **D**escription **L**anguage-a*), te na taj način omogućuje dostupnost servisa drugim korisnicima.
   - registrira servis u UDDI (*eng. **U**niversal **D**escription, **D**iscovery and **I**ntegration*) registre. Ti registri omogućuju razvojnim timovima objavljivanje njihovih servisa te pretraživanje i korištenje drugih objavljenih servisa.
2. Potencijalni korisnik pronalazi web servis pretraživanjem UDDI registara.
3. Korisnička aplikacija, web preglednik ili uređaj za mobilnu komunikaciju se povezuje na odabrani web servis korištenjem SOAP-a (*eng. **S**imple **O**bject **A**ccess **P**rotocol*), HTTP (*eng. **H**yper**T**ext **T**ransfer **P**rotocol*) ili WAP protokola koji nude mogućnost korištenja XML formata za slanje parametara komponentama servisa, te prihvaćanje rezultata obrade podataka.

U strukturi web servisa tri su osnovne razine čije elemente je potrebno kreirati prilikom razvoja integriranih distribuiranih aplikacija [Slika 6.1]:

- *Klijentska razina* – omogućuje samostalnim aplikacijama, poslovnim sustavima, web pretraživačima, PDA uređajima i mobilnim telefonima korištenje web servisa slanjem zahtjeva za podacima i primanjem rezultata.
- *Poslovna razina* – spremnik web servisa s komponentama koje implementiraju: poslovnu logiku, mehanizme za transformaciju podatka i procesiranje klijentskih zahtjeva, logiku i mehanizme za pristup trajno zapisanim podacima.

- *Razina trajnog zapisa podataka* – uključuje jednu ili više baza podataka, postojeće poslovne sustave, druge web servise, te repozitorije za trajno spremanje podataka ovisno o kontekstu primjene.



Slika 6.1: Korištenje web servisa

Osnovna prednost korištenja web servisa je dinamička karakteristika njihovog djelovanja. Razdvajanje poslovne logike za upravljanje podacima od načina prikaza podataka, omogućuje kreiranje dinamičkog odziva koji ovisi o specifičnoj situaciji primjene. Web servisi procesiraju zahtjev ovisno o identitetu korisnika, postavkama sustava, lokaciji i razlogu za zahtjev koji je pred njih postavljen. Više servisa moguće je međusobno kombinirati u radu, te na taj način suradnjom ostvariti njihovo zajedničko djelovanje.

Princip rada servisa je slijedeći. U prvom koraku web servis prima zahtjeve klijenata u obliku XML dokumenta. XML dokument koji je zaprimljen se analizira, te se ovisno o elementima koje sadrži, određenim redoslijedom pozivaju jedna ili više metoda komponenti poslovne razine. Metode poslovnih komponenti servisa izvrše zahtijevane operacije nad podacima, a vrijednosti koje te metode vraćaju zapisuju se ponovo u XML dokument koji se vraća klijentu. Takav XML dokument moguće je prije prikaza klijentu transformirati u željeni oblik. Posebno je to naglašeno kod klijenata poput web preglednika, PDA ili mobilnih telefona, za koje se kreira prezentacijski server koji transformira XML dokumente u oblik koji je prikladan za pojedini tip klijenta: HTML ili WML oblik, koristeći se pri tome XSL tehnologijom za transformaciju XML-a.

Autor ovog rada smatra da bi se kreiranjem web servisa za razmjenu i upravljanje informacijama o proizvodu na temelju informacijskog modela i zahtijevane

funkcionalnosti, što je opisano kroz prethodne glave rada, omogućila računalna implementacija koja bi ispunila zahtjeve primjene u heterogenim i distribuiranim inženjerskim razvojnim okruženjima. Time bi se također olakšalo povezivanje različitih aplikacija koje razvojni timovi već koriste u radu, a koje generiraju i koriste podatke o proizvodu. Stoga će u nastavku ove glave, biti opisani glavni dijelovi od kojih je izgrađen web servis za razmjenu i upravljanje s podacima o proizvodu. Računalna implementacija definirana na taj način predstavlja zadnji korak u preslikavanju iz realne problemske osnove u domenu računalnog rješavanja problema, koje je definirano na početku ovog rada [Slika 2.2]. U tu svrhu, najprije će se opisati razina trajnog zapisa podataka, nakon toga razina poslovne logike te XML operacije za pristupanje poslovnim komponentama servisa.

6.3 Razina trajnog zapisa podataka web servisa

Na osnovi predloženog metamodela podataka o proizvodu koji je detaljno analiziran i obrazložen u trećoj i četvrtoj glavi, za potrebe računalne implementacije web servisa definirana je struktura za trajni zapisa podataka. Vrijednosti atributa entiteta koji čine metamodel podataka o proizvodu trajno će biti pohranjene korištenjem entiteta relacijske baze podataka. U skladu sa STEP strukturom zapisa podataka o proizvodu kreirane su tablice i relacije među njima. Svaki entitet STEP modela može imati više instanci, te će se atributi svake pojedine instance zapisivati u relacijsku bazu kao jedan redak u tablicama, pri čemu će svaka kolona tablice predstavljati pojedini atribut entiteta. Osim osnovnih tablica (tablice zapisa STEP entiteta), u bazi podataka mogu se nalaziti i dodatne tablice koje služe za zapis podataka potrebnih za rad servisa. Relacije koje su definirane između tablica su relacije asocijativnosti, a govore o vezama između atributa entiteta. Struktura tablica i relacija opisana je SQL (*eng. Structured Querying Language* [16]) izrazom koji omogućuje njihovu implementaciju u bilo koju komercijalnu relacijsku bazu [Tablica 6-1].

Tablica 6-1: SQL opis strukture tablica i relacija

SQL opis	Odgovarajući entiteti STEP PDM sheme

create table ProductContext(
productContextId varchar(80) not null primary key,
application varchar(80) not null);
create table ProductCategory(
productCategoryId varchar(80) not null primary key,
description varchar(80) null);
create table Product(
productId varchar(80) not null primary key,
name varchar(80) not null,
description varchar(80) null,
productContextId varchar(80) not null foreign key references ProductContext(productContextId),
productCatgoryId varchar(80) not null foreign key references ProductCategory(productCategoryId));
create table ProductDefinitionFormation(
prodDefFormId varchar(80) not null primary key,
description varchar(80) null,
	Entiteti STEP PDM sheme opisani u poglavlju 4.3.1

productId varchar(80) not null foreign key references Product(productId));	

create table PropertyType(
propertyTypeId varchar(80) not null primary key,
description varchar(80) not null,
unit varchar(80) not null);
create table PropertyDefinition(
propertyTypeId varchar(80) not null foreign key references PropertyType(propertyTypeId),
prodDefFormId varchar(80) not null foreign key references ProductDefinitionFormation(prodDefFormId),
items varchar(80) not null,
constraint pkPropertyDefinition primary key(propertyTypeId, prodDefFormId));
	Entiteti STEP PDM sheme opisani u poglavlju 4.3.2

create table QuantifiedAssemblyComponentUsage(quaAssCompUsageId varchar(80) not null primary key, name varchar(80) null, description varchar(80) null, relatingProdDefFormId varchar(80) not null foreign key references ProductDefinitionFormation(prodDefFormId), relatedProdDefFormId varchar(80) not null foreign key references ProductDefinitionFormation(prodDefFormId), quantity varchar(80) not null);	Entiteti STEP PDM sheme opisani u poglavlju 4.3.3

create table DocumentType(
documentTypeId varchar(80) not null primary key,
description varchar(80) not null);
create table DocumentFile(
documentFileId varchar(80) not null primary key,
name varchar(80) not null,
description varchar(80) not null,
documentTypeId varchar(80) not null foreign key references DocumentType(documentTypeId));
create table AppliedDocumentReference(
documentFileId varchar(80) not null foreign key references DocumentFile(documentFileId),
prodDefFormId varchar(80) not null foreign key references ProductDefinitionFormation(prodDefFormId),
constraint pkAppliedDocumentReference primary key(documentFileId, prodDefFormId));
	Entiteti STEP PDM sheme opisani u poglavljima 4.3.4 - 4.3.7

create table Person(
personId varchar(80) not null primary key,
lastName varchar(80) not null,
firstName varchar(80) not null,
prefixTitles varchar(80) null,
suffixTitles varchar(80) null);
create table Organization(
organizationId varchar(80) not null primary key,
description varchar(80) null);
create table OrgPerson(
orgPersonId varchar(80) not null primary key,
description varchar(80) null,
personId varchar(80) not null foreign key references Person(personId),
organizationId varchar(80) not null foreign key references Organization(organizationId));
create table ApprovalStatus(
approvalStatusId varchar(80) not null primary key,
description varchar(80) not null);
create table AppliedApprovalAssignment(
approvalStatusId varchar(80) not null foreign key references ApprovalStatus(approvalStatusId),
orgPersonId varchar(80) not null foreign key references OrgPerson(orgPersonId),
	Entiteti STEP PDM sheme opisani u poglavlju 4.3.8

approvalDateTime varchar(80) not null, approvalItemId varchar(80) not null, constraint pkAppliedApprovalAssignment primary key(approvalStatusId, orgPersonId, approvalDateTime, approvalItemId));	
create table OrganizationalProject(orgProjectId varchar(80) not null primary key, name varchar(80) not null, description varchar(80) not null, organizationId varchar(80) not null foreign key references Organization(organizationId)); create table ProductConcept(productConceptId varchar(80) not null primary key, name varchar(80) not null, description varchar(80) null, orgProjectId varchar(80) not null foreign key references OrganizationalProject(orgProjectId)); create table ConfigurationItem(configurationItemId varchar(80) not null primary key, name varchar(80) not null, description varchar(80) null, purpose varchar(80) null, productConceptId varchar(80) not null foreign key references ProductConcept(productConceptId)); create table ConfigurationDesign(configurationItemId varchar(80) not null foreign key references ConfigurationItem-(configurationItemId), prodDefFormId varchar(80) not null foreign key references productDefinitionFormation(prodDefFormId), name varchar(80) not null, description varchar(80) null, constraint pkConfigurationDesign primary key(configurationItemId, prodDefFormId));	Entiteti STEP PDM sheme opisani u poglavlju 4.3.9

create table PersonRole(
roleId varchar(80) not null primary key,
description varchar(80) null);
create table AppliedPersonAssignment(
orgPersonId varchar(80) not null foreign key references OrgPerson(orgPersonId),
assignmentItemId varchar(80) not null,
roleId varchar(80) not null foreign key references PersonRole(roleId),
constraint pkAppliedPersonAssignment primary key(orgPersonId, assignmentItemId, roleId));
create table RequestStatus(
requestStatusId varchar(80) not null primary key,
description varchar(80) not null);
create table ActionRequest(
actionRequestId varchar(80) not null primary key,
purpose varchar(80) not null,
description varchar(80) not null,
orgPersonId varchar(80) not null foreign key references OrgPerson(orgPersonId));
create table ActionRequestStatus(
reqStatusId varchar(80) not null foreign key references requestStatus(requestStatusId),
actionRequestId varchar(80) not null foreign key references ActionRequest(actionRequestId),
orgPersonId varchar(80) not null foreign key references OrgPerson(orgPersonId),
dateTime varchar(80) not null,
constraint pkActionRequestStatus primary key(reqStatusId, actionRequestId, orgPersonId));
	Entiteti STEP PDM sheme opisani u poglavlju 4.3.10

Relacijska baza je u prikazanom istraživanju izabrana za trajnu pohranu podataka prije svega zbog općenite jednostavnosti korištenja relacijskih baza i upravljanja

pohranjenim podacima, mogućnosti provođenja kompleksnih upita, te dodatnih alata za kreiranje izvještaja i upravljanje bazom.

6.4 Poslovna razina web servisa

Definiranje komponenti poslovne razine web servisa zahtjeva analizu arhitekture servisa s različitih točki gledanja. Različiti pogledi na arhitekturu komponenti koje svojim djelovanjem ostvaruju poslovne logiku servisa može se iskoristiti za kontrolu iterativnog i inkremantalnog razvoja servisa kroz njegov životni vijek. Kreiranjem arhitekture komponenti web servisa je u ovom radu ostvareno:

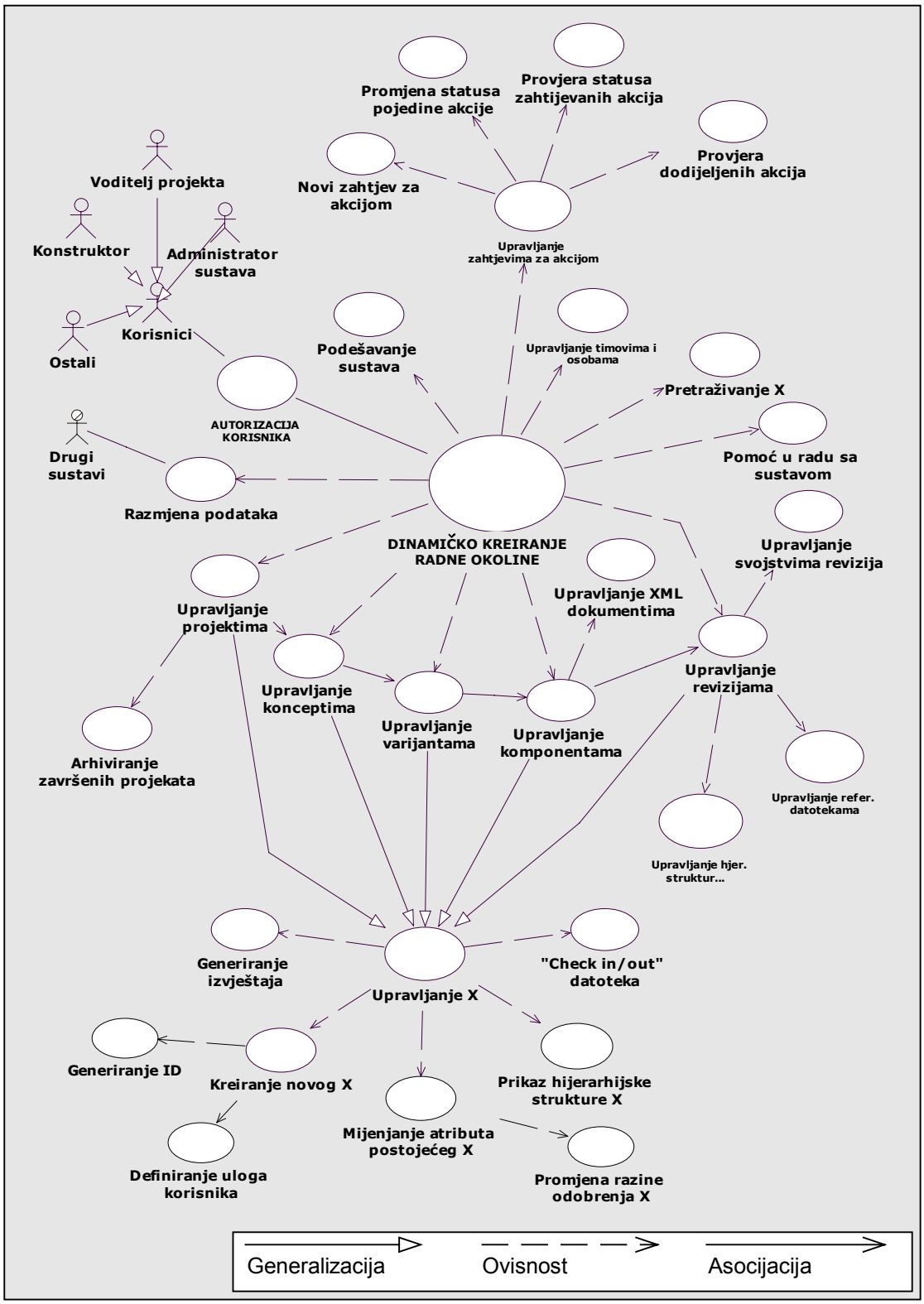
- organizacija razine poslovne logike web servisa,
- selekcija potrebnih komponenti i njihovih sučelja,
- ponašanje koje je specificirano međusobnim interakcijama između komponenti poslovne razine servisa, kao i interakcijama između poslovnih komponenti i entiteta trajne fizičke pohrane podataka,
- kompozicija komponenti u složenije podsustave.

6.4.1 Dijagram slučajeva korištenja web servisa

Kao početna točka za izgradnju razine poslovne logike web servisa za upravljanje i razmjenu s podacima o proizvodu, poslužila je analiza zahtijevanog ponašanja servisa, na temelju funkcija definiranih u poglavlju 3.3., te entiteta STEP PDM sheme definiranih u četvrtoj glavi. U tu svrhu je upotrebom objektnog pristupa, te grafičkim prikazom prema UML standardu [69], kreiran dijagram mogućih slučajeva korištenja (*eng. use case*) koji govori o željenom kontekstu korištenja servisa [Slika 6.2].

Analiza konteksta korištenja servisa započela je definiranjem mogućih korisnika te njihovih zahtjeva, odnosno akcija koje servis mora omogućiti korisnicima. Za bolje razumijevanje dijagrama potrebno je naglasiti da su pojedini slučajevi korištenja označeni elipsom prema UML specifikaciji. Ovaj tip dijagrama objašnjava što sustav radi, ali ne i kako. Organizacija i međusobna interakcija slučajeva korištenja definirana je relacijama koje su postavljene između njih. U ovom slučaju su to generalizacija, relacija ovisnosti i asocijacija.

Gledano iz objektno orijentirane perspektive, generalizacija slučajeva korištenja može se usporediti sa generalizacijom klasa. To znači da slučajevi korištenja koji su djeca nasljeđuju ponašanje i značenje slučajeva korištenja koji predstavljaju roditelja u relaciji generalizacije. Generalizacija je prikazana punom linijom na čijem kraju se nalazi neispunjena strelica. U konkretnom slučaju generalizacija je definirana između slučaja korištenja nazvanog ***Upravljanje X*** koji definira općenito upravljanje entitetima kreiranim prema STEP PDM shemi. Pet specijaliziranih slučajeva korištenja se iz njega definiraju za svaki pojedini entitet uz specifična proširenja.



Slika 6.2: Dijagram slučajeva korištenja (USE CASE) web servisa

Relacije ovisnosti između slučajeva korištenja predstavljaju semantičku relaciju kojom je definirano da osnovni slučaj korištenja uključuje ponašanje drugog slučaja korištenja koji sudjeluje u toj relaciji. Relacija uključivanja prikazana je na dijagramu

isprekidanom linijom s otvorenom strelicom na kraju. U konkretnom slučaju relacija uključivanja je definirana za osnovu ***Dinamičko kreiranje radne okoline*** koja ovisno o pravima i ulozi korisnika omogućuje korištenje pojedinih komponenti servisa: pojedinih instanci slučaja ***Upravljanje X***, zatim ***Pomoć u radu sa sustavom***, ***Podešavanje sustava***, ***Upravljanje timovima i osobama***, ***Upravljanje zahtjevima za akcijom***, ***Pretraživanje X, Arhiviranje završenih projakata.*** Jednako tako osnova ***Upravljanje X*** uključuje slijedeće moguće slučajeve korištenja ***Kreiranje novog X*** kojim se kreira potpuno novi entitet kojim se upravlja, zatim ***Mijenjanje atributa postojećeg X*** kojime se mijenja neki od atributa već postojećeg entiteta, ***Generiranje izvještaja*** za pojedini entitet, ***Checkin/out datoteka*** pojedinog entiteta, te ***Generiranje prikaza hijerarhijske strukture X*** za entitet kojim se upravlja.

Relacije asocijacija na dijagramu slučajeva korištenja, prema UML-u, označuju komunikaciju između korisnika i pojedinih slučajeva korištenja, opisujući na taj način komuniciranje razmjenom poruka, odnosno upita koji se šalju web servisu i odgovora na te upite koje web servis isporučuje korisnicima. U konkretnom slučaju predviđeno je da korisnici komuniciraju sa pojedinim slučajevima korištenja preko dinamički kreirane radne okoline, koja ovisi o pravima i ulozi svakog korisnika koji pristupa web servisu. Relacije su prikazane punom tankom linijom, sa ili bez otvorene strelice na kraju. Relacije asocijacije definirane između upravljanja pojedinim entitetima naznačuju strukturalnu veze između entiteta kojima se upravlja. Te veze su definirane relacijama asocijacije iz metamodela te su uključene i u fizički zapis podataka opisan u prethodnom poglavlju.

6.4.2 Dijagrami klasa poslovne razine web servisa

Prema dijagramima korištenja koji su definirani u prethodnom poglavlju, kreirana je struktura klasa koje se implementiraju kao glavne poslovne komponente web servisa. Klase su kreirane na način da svaka od njih predstavlja skup uloga koji je definiran atributima i metodama klase. Da bi se olakšalo modeliranje raspodjele odgovornosti pojedinih komponenti web servisa, klase koje međusobno surađuju su grupirane prilikom objašnjavanja atributa i metoda. Pri modeliranju strukture klasa poslovne razine web servisa razlikujemo tri grupe (moduli web servisa):

- klase koje modeliraju operacije trajnog zapisa podatka u relacijsku bazu,
- klase koje modeliraju poslovnu logiku i pravila manipuliranja podacima,
- dodatni moduli web servisa.

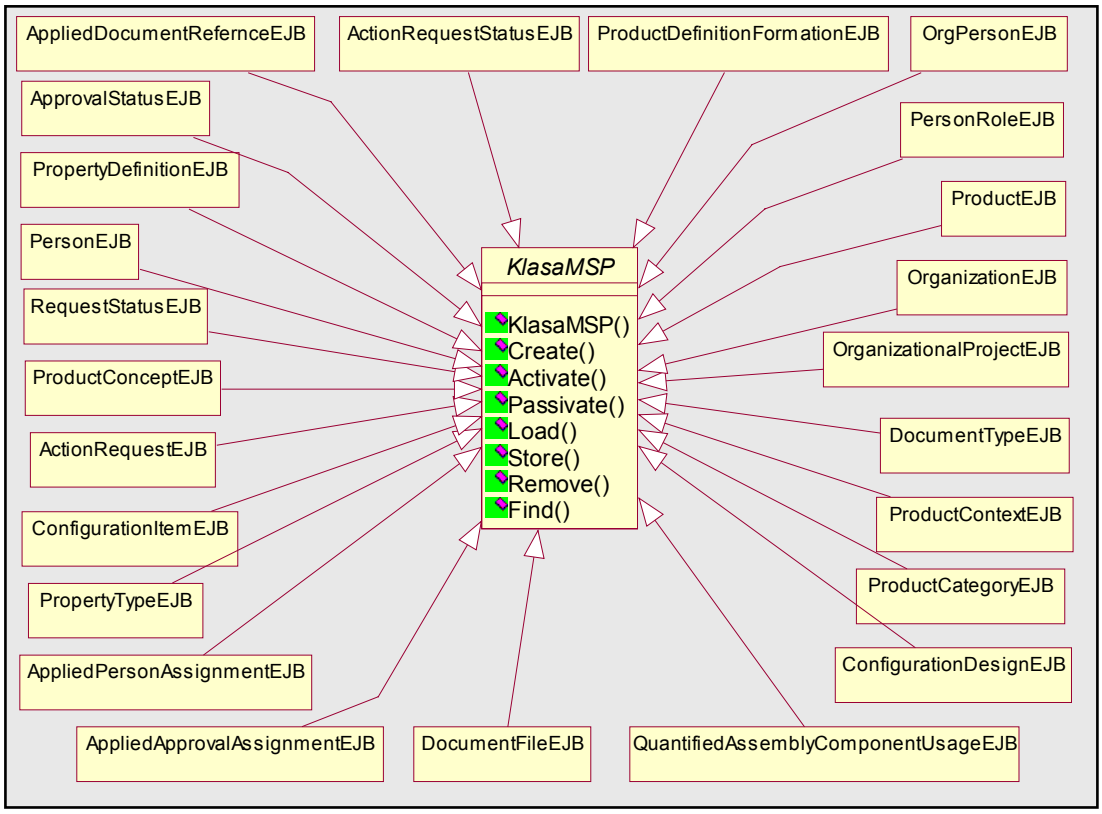
Klase koje modeliraju upravljanje trajnim zapisom podataka u relacijsku bazu

Ovu grupu čine klase koje modeliraju operacije za upravljanje trajnom pohranom podataka u razini fizičkog zapisa podataka, odnosno tablicama i relacijama baze podataka. Uloga ove grupe klasa je prikaz fizičkih podataka kao realnih objekata, što

omogućuje manipuliranje podacima pozivanjem metoda takvih objekta. Na taj se način proširuju mogućnosti relacijskih baza podataka objektno orijentiranim metodama programiranja, odnosno ostvaruje trajno zapisivanje vrijednosti atributa pojedinih objekta. Ovakav pristup omogućuje jednostavno rješenje za provođenje kompleksnih operacija i pretraživanja vrijednosti atributa objekta.

Sve klase ove grupe koriste metode za pristup entitetima relacijske baze podataka, koje su definirane u apstraktnoj klasi ***KlasaMSP*** [Slika 6.3]:

- *Create()* – metoda služi za istanciranje novog objekta klase, odnosno kreiranje novih zapisa u bazi podataka,
- *Load()* – metoda služi za učitavanja podataka iz baze u vrijednosti atributa objekta, omogućujući na taj način manipuliranje tim podacima kao atributima objekta
- *Store()* – metoda služi za zapisivanje vrijednosti atributa objekta u bazu podataka, sinkronizirajući na taj način vrijednost između trajnog zapisa i vrijednosti atributa.
- *Remove()* – metoda služi za brisanje zapisa iz relacijske baze, nakon te metode objekt se pasivizira te čeka ponovno korištenje



Slika 6.3: Dijagram klasa koje modeliraju upravljanje trajnom pohranom podataka

Osim metoda i atributa koje nasljeđuju od apstraktne klase, klase za modeliranje upravljanja zapisom podataka sadrže atribute koji odgovaraju atributima pojedinog

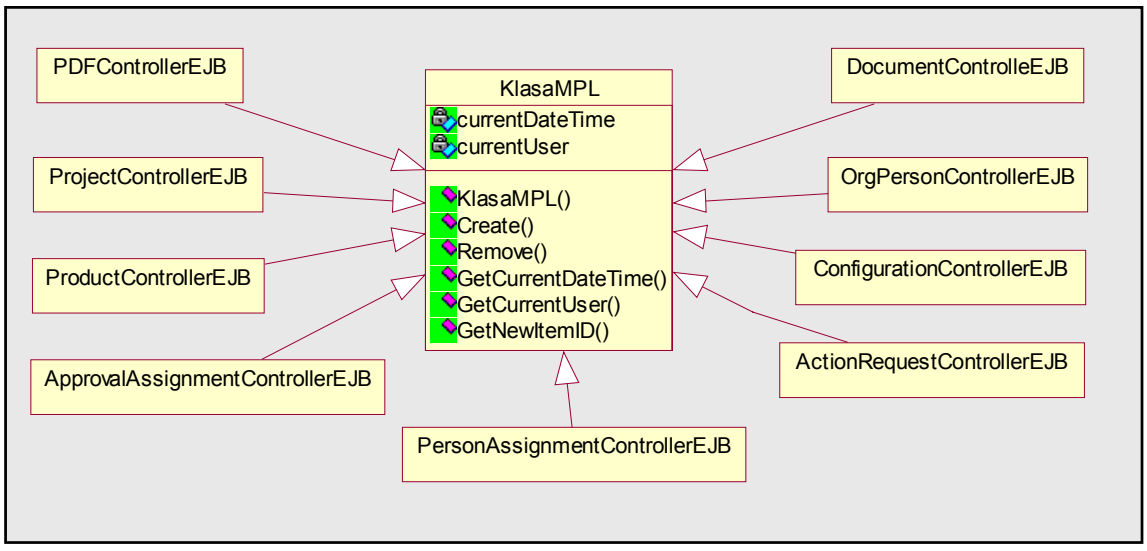
entiteta iz područja informacijskog modela kojeg klasa modelira, te dodatne metode za pretraživanje podataka koje kao rezultat vraćaju kolekciju objekta koji ispunjavaju zahtjeve pretraživanja.

Klase koje modeliraju poslovnu logiku i pravila manipuliranja podacima

Klase ove grupe predstavljaju procese za upravljanje podacima o proizvodu. Bitno je naglasiti da se objekti tih klasa istanciraju samo za jednog korisnika istovremeno, te se instance ne dijele između klijenata. To povlači za sobom da objekti ovih klasa nisu perzistentni, odnosno njihovi atributi se trajno ne pohranjuju. Na temelju metoda ovih klasa, definiraju se XML operacije koje omogućavaju komunikaciju korisnika i web servisa putem XML dokumenta.

Sve klase ove grupe za svoj rad koriste metode koje su definirane u apstraktnoj klasi ***KlasaMPL*** [Slika 6.4]:

- *create()* – metoda služi za istanciranje novog objekta klase, pri čemu je moguće definirati nekoliko ovakvih metoda sa različitim argumentima koje instanciraju objekte na različite načine ovisno o zahtjevima,
- *remove()* – metoda služi za brisanje instance objekta, nakon što je završen njegov životni vijek, odnosno kad je objekt završio sa obradom zahtjeva klijenta,
- *getCurrentDateTime()* – metoda se poziva prilikom istanciranja novog objekta, promjene atributa, ili pri uklanjanju objekta iz skupa instanci neke klase, a služi za bilježenje vremena događaja,



Slika 6.4: Dijagram klasa koje modeliraju poslovnu logiku i pravila manipuliranja podacima

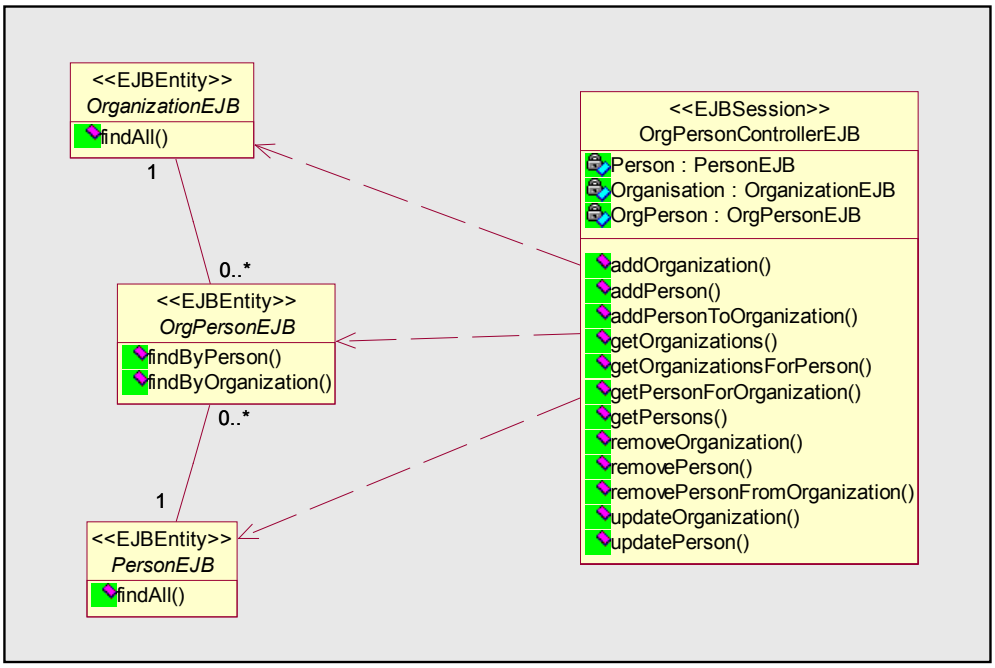
- *getCurrentUser()* - metoda se poziva prilikom istanciranja novog objekta, promjeni atributa, ili uklanjanju objekta iz grupe klasa za trajnu pohranu podataka, a služi za evidentiranje korisnika,

- *getNewItemId()* – metoda se aktivira istanciranjem novog objekta iz grupe klasa za trajnu pohranu podataka, a dodjeljuje jedinstvenu identifikacijsku oznaku.

U daljnjem tekstu slijedi detaljniji prikaz struktura klasa koje modeliraju poslovnu logiku i pravila manipuliranja podacima, te njihove reference na klase koje modeliraju fizički zapis podataka.

Objekti klase ***OrgPersonController*** istanciraju se sa svrhom upravljanja i organizacije korisnika i grupa korisnika. U klasi ***OrgPersonController*** [Slika 6.5] definirani su slijedeći atributi i metode:

- *Person* – atribut tipa ***Person*** koji sadrži informacije o osobi koja ima neku od definiranih uloga za upravljanje i razmjenu s podacima proizvodu,
- *Organisation* – atribut tipa ***Organisation*** koji sadrži informacije o timu u koji su raspodijeljeni korisnici radi lakšeg upravljanja projektima za vrijeme razvojnog ciklusa proizvoda,
- *OrgPerson* – atribut tipa ***OrgPerson*** koji sadrži informacije o tome koji je korisnik pridružen pojedinom timu,
- *addOrganisation()* – metoda služi za istanciranje novog objekta klase ***Organisation***, odnosno dodavanje novog tima,
- *addPerson()* - metoda služi za istanciranje novog objekta klase ***Person***, odnosno dodavanje novog korisnika sustava,
- *addPersonToOrganisation()* - metoda služi za istanciranje novog objekta klase ***OrgPerson***, odnosno dodjeljivanje korisnika određenom timu,
- *getOrganisations()* - metoda služi za pretraživanje polja svih timova sa svrhom odabira jednog od postojećih,
- *getOrganisationsForPersons()* - metoda služi za pretraživanje polja timova kojima je pridružen pojedini korisnik,
- *getPersonsForOrganisation()* - metoda služi za pretraživanje polja korisnika koji su pridruženi pojedinom timu,
- *getPersons()* - metoda služi za pretraživanje polja svih korisnika sa svrhom odabira jednog od postojećih,
- *removeOrganisation()* – metoda služi za brisanje zadane instance iz polja timova,
- *removePerson()* – metoda služi za brisanje zadane instance iz polja korisnika
- *removePersonFromOrganisation()* – metoda služi za brisanje zadane instance iz polja dodijeljenih korisnika pojedinom timu,
- *updateOrganisation()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog tima,

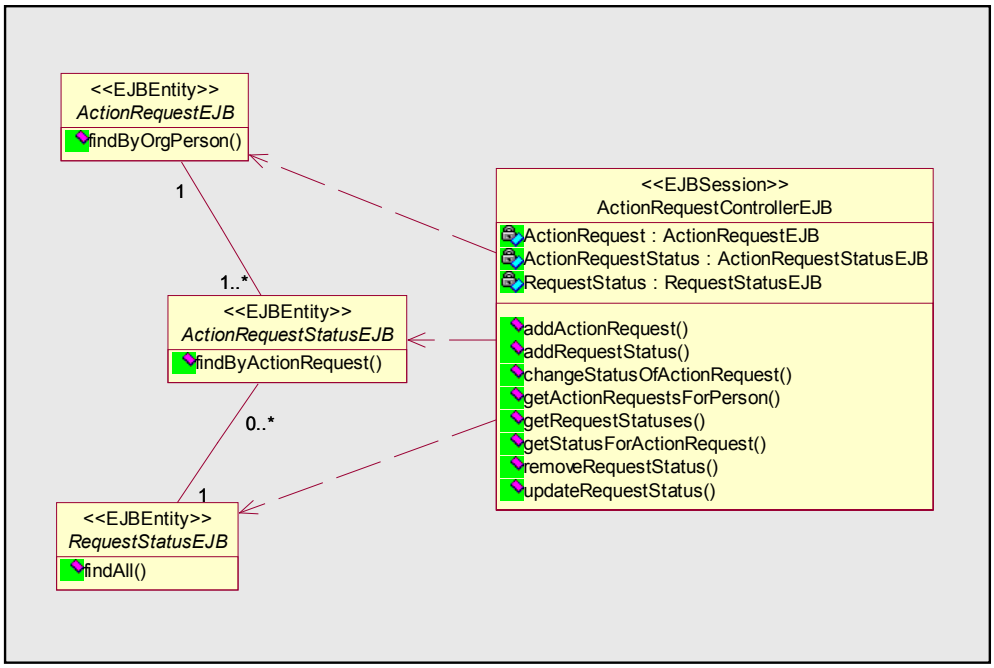


Slika 6.5: Dijagram klasa *OrgPersonController*, *Organization*, *Person* i *OrgPerson*

- *updatePerson()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog korisnika.

Objekti klase ***ActionRequestController*** istanciraju se u svrhu upravljanja zahtjevima za pojedinim akcijama koje definiraju korisnici, te kontrolu trenutnog statusa pojedine akcije. U klasi ***ActionRequestController*** definirani su slijedeći atributi i metode [Slika 6.6]:

- *ActionRequest* – atribut tipa ***ActionRequest*** koji sadrži informacije o pojedinom zahtjevu za akcijom koji se razmjenjuje između korisnika,
- *ActionRequestStatus* – atribut tipa ***ActionRequestStatus*** koji sadrži informacije o pojedinom statusu neke od zahtijevanih akcija,
- *RequestStatus* – atribut tipa ***RequestStatus*** koji sadrži informacije o statusu koji pojedina zahtijevana akcija može poprimiti,
- *addActionRequest()* - metoda služi za istanciranje novog objekta klase ***ActionRequest***, odnosno kreiranje novog zahtjeva za akcijom nekog od korisnika,
- *addRequestStatus()* - metoda služi za istanciranje novog objekta klase ***RequestStatus***, odnosno kreiranje novog statusa koji može poprimiti zahtjev za akcijom,
- *changeStatusOfActionRequest()* – metoda služi za istanciranje novog objekta klase ***ActionRequestStatus***, odnosno definiranje novog statusa za neku od zahtijevanih akcija,
- *getActionRequestsForPerson()* - metoda služi za pretraživanje polja akcija koje se zahtijevaju od pojedinog korisnika,

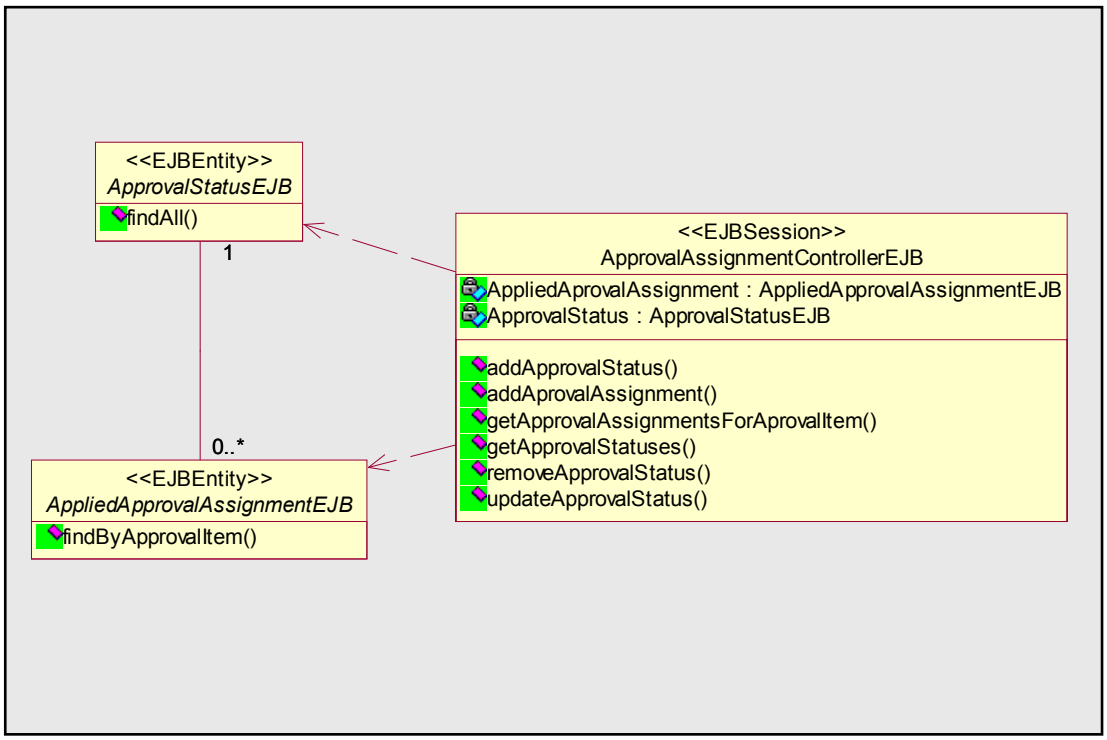


Slika 6.6: Dijagram klasa *ActionRequestController*, *ActionRequest*, *RequestStatus*, i *ActionRequestStatus*

- *getRequestStatuses()* - metoda služi za pretraživanje polja statusa koji su dozvoljeni za akcije,
- *getStatusForActionRequest()* - metoda služi za pretraživanje polja statusa koji su dodijeljeni pojedinom zahtjevu za akcijom,
- *removeRequestStatus()* – metoda služi za brisanje zadane instance iz polja dozvoljenih statusa koje može poprimiti zahtjeva za akcijom,
- *updateRequestStatus()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog statusa.

Objekti klase ***ApprovalAssignmentController*** istanciraju se u svrhu upravljanja odobrenjima, odnosno statusa pojedinih entiteta. U klasi ***ApprovalAssignment-Contoller*** definirani su slijedeći atributi i metode [Slika 6.7]:

- *AppliedAprovalAssignment* – atribut tipa ***AppliedApprovalAssignment*** koji sadrži informacije o stupnju dodijeljenog odobrenja za pojedini entitet za koji se to zahtijeva,
- *ApprovalStatus* – atribut tipa ***ApprovalStatus*** koji sadrži informacije o stupnju odobrenja koji se može dodijeliti pojedinom entitetu,
- *addApprovalStatus()* - metoda služi za istanciranje novog objekta klase ***ApprovalStatus***, odnosno definiranje novog stupnja odobrenja koji se može dodijeliti pojedinom entitetu,
- *addApprovalAssignment()* - metoda služi za istanciranje novog objekta klase ***AppliedApprovalAssignment***, odnosno dodjeljivanje novog stupnja odobrenja nekom od entiteta,



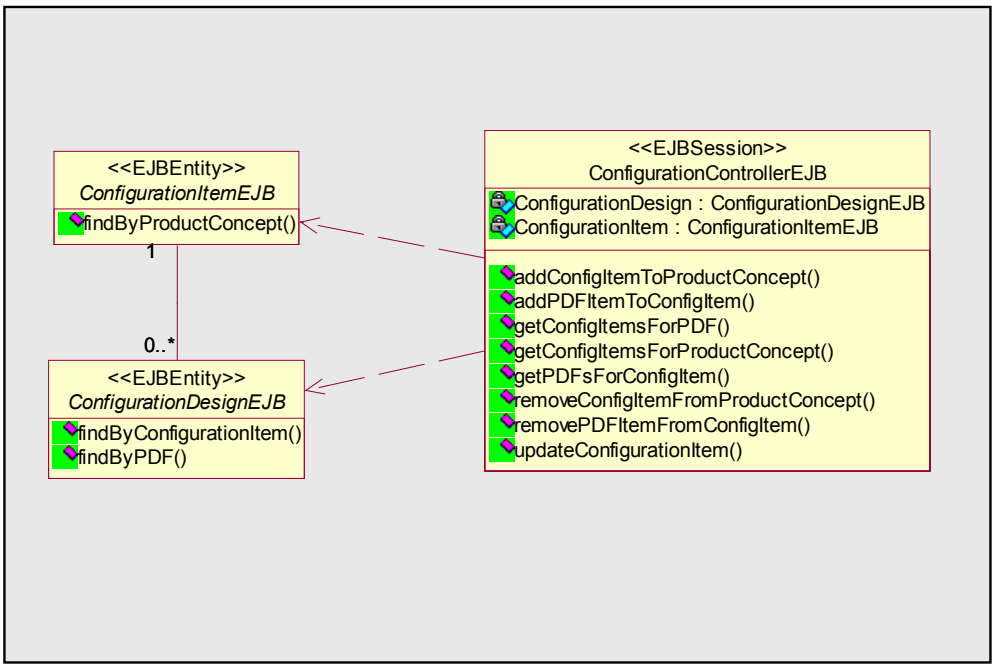
Slika 6.7: Dijagram klasa *ApprovalAssignementController*, *ApprovalStatus* i *AppliedApprovalAssignement*

- *getApprovalAssignmentsForApprovalItem()* - metoda služi za pretraživanje polja odobrenja koji su dodijeljeni pojedinom entitetu,
- *getApprovalStatuses()* - metoda služi za pretraživanje polja stupnjeva odobrenja koje je moguće dodijeliti entitetu,
- *removeApprovalStatus()* – metoda služi za brisanje zadane instance iz polja dozvoljenih stupnjeva odobrenja koji se mogu dodijeliti entitetu,
- *updateApprovalStatus()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog mogućeg stupnja odobrenja.

Objekti klase ***ConfigurationContoller*** istanciraju se u svrhu upravljanja konfiguracijama. U klasi ***ConfigurationContoller*** definirani su slijedeći atributi i metode [Slika 6.8]:

- *ConfigurationDesign* – atribut tipa ***ConfigurationDesign*** koji sadrži informacije o komponenti koja je dodijeljena određenoj konfiguraciji (varijanti) proizvoda,
- *ConfigurationItem* – atribut tipa ***ConfigurationItem*** koji sadrži informacije o pojedinoj konfiguraciji koja je definirana za neki konceptualni dio proizvoda,
- *addConfigurationItemToProductConcept()* - metoda služi za istanciranje novog objekta klase ***ConfigurationItem***, odnosno definiranje nove konfiguracije za određeni konceptualni dio proizvoda,
- *addPDFItemToConfigItem()* - metoda služi za istanciranje novog objekta klase

ConfigurationDesign, odnosno pridruživanje revizije komponente pojedinoj konfiguraciji,

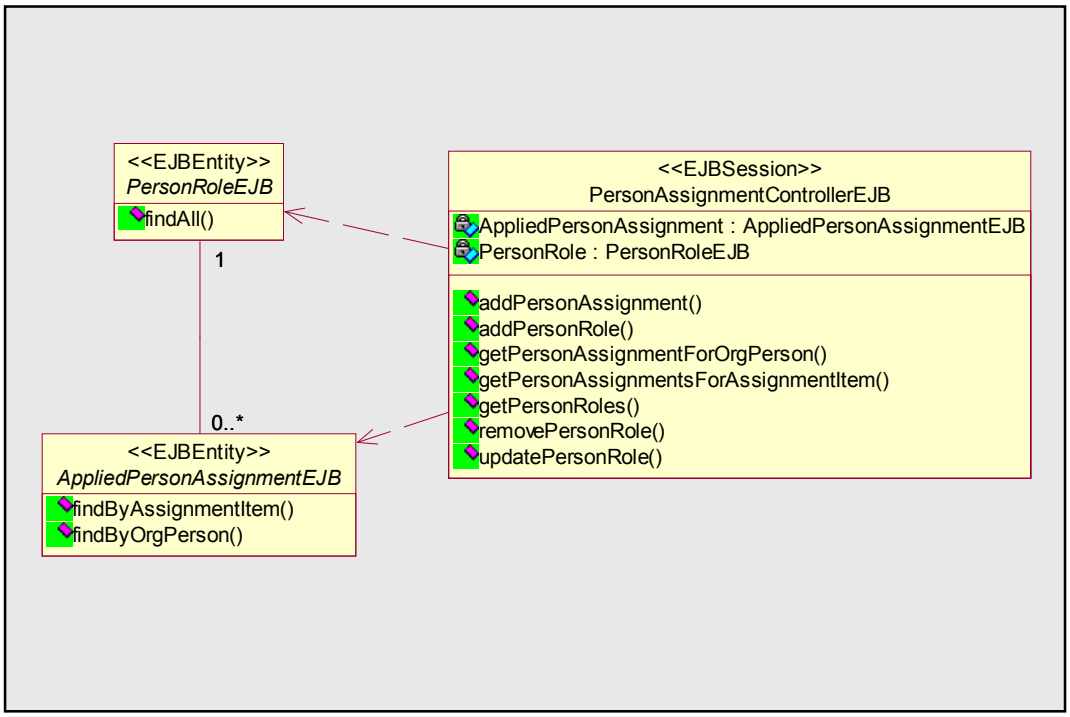


Slika 6.8: Dijagram klasa *ConfigurationController*, *ConfigurationItem* i *ConfigurationDesign*

- *getConfigItemsForPDF()* - metoda služi za pretraživanje polja konfiguracija kojima je pridružena pojedina revizija komponente,
- *getConfigItemsForProductConcept()* - metoda služi za pretraživanje polja konfiguracija koje su definirane za pojedini konceptualni dio proizvoda,
- *getPDFsForConfigItem()* - metoda služi za pretraživanje polja revizija komponenti koje su pridružene pojedinoj konfiguraciji,
- *removeConfigItemFromProductConcept()* – metoda služi za brisanje zadane instance iz polja konfiguracija koje su dodijeljene pojedinom konceptualnom dijelu proizvoda,
- *removePDFItemFromConfigItem()* – metoda služi za brisanje zadane instance iz polja revizija komponenti koje su pridružene pojedinoj konfiguraciji,
- *updateConfigurationItem()* – metoda služi za postavljanje vrijednosti atributa instance pojedine konfiguracije.

Objekti klase ***PersonAssignmentContoller*** istanciraju se u svrhu dodjeljivanja korisnika sa određenim ulogama pojedinom entitetu. U klasi ***PersonAssignmentController*** definirani su slijedeći atributi i metode [Slika 6.9]:

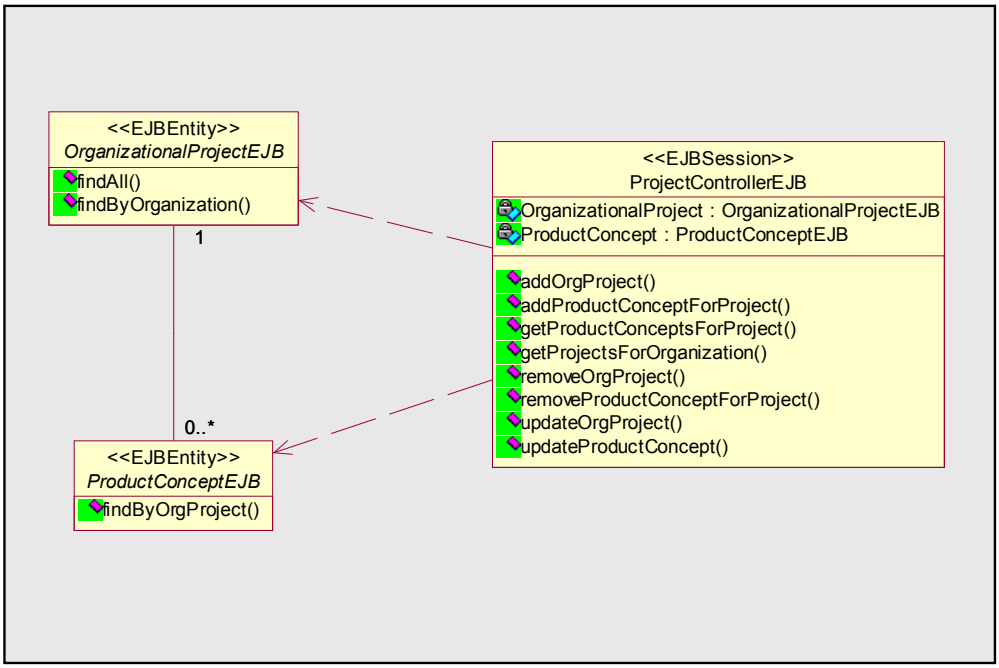
- *AppliedPersonAssignment* – atribut tipa ***AppliedPersonAssignment*** koji sadrži informacije o korisniku koji ima neku od uloga pri manipuliranju pojedinim entitetom,



Slika 6.9: Dijagram klasa *PersonAssignmentController*, *PersonRole*, *AppliedPersonAssignment*

- *PersonRole* – atribut tipa ***PersonRole*** koji sadrži informacije o ulozi koju pojedini korisnik može poprimiti nad pojedinim entitetom,
- *addPersonAssignment()* - metoda služi za istanciranje novog objekta klase ***AppliedPersonAssignment***, odnosno definiranje uloge pojedinog korisnika u upravljanju entitetom,
- *addPersonRole()* - metoda služi za istanciranje novog objekta klase ***PersonRole***, odnosno definiranje tipa uloge koju korisnik može poprimiti nad entitetom,
- *getPersonAssignmentForOrgPerson()* - metoda služi za pretraživanje polja uloga koje su dodijeljene pojedinom korisniku nad različitim entitetima,
- *getPersonAssignmensForAssignmentItem()* - metoda služi za pretraživanje instanci korisnika koji su dodijeljeni pojedinom entitetu,
- *getPersonRoles()* – metoda služi za pretraživanje polje mogućih uloga korisnika,
- *removePersonRole()* – metoda služi za brisanje zadane instance iz polja mogućih uloga korisnika,
- *updatePersonRole()* – metoda služi za postavljanje vrijednosti atributa instance pojedine moguće uloge korisnika.

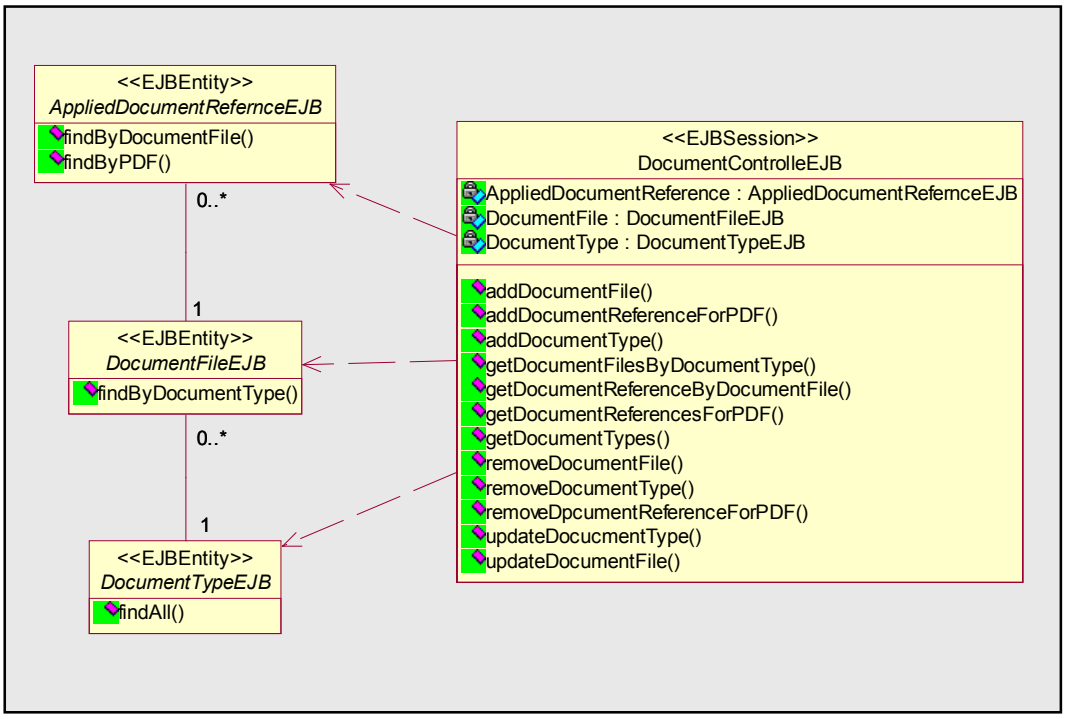
Objekti klase ***ProjectContoller*** istanciraju se u svrhu upravljanja projektima. U klasi ***ProjectController*** definirani su slijedeći atributi i metode [Slika 6.10]:



Slika 6.10: Dijagram klasa *ProjectController*, *OrganizationalProject* i *ProductConcept*

- *OrganizationalProject* – atribut tipa ***OrganizationalProject*** koji sadrži informacije o projektu, odnosno proizvodu čijim podacima se upravlja sustavom,
- *ProductConcept* – atribut tipa ***ProductConcept*** koji sadrži informacije o pojedinom konceptualnom dijelu proizvoda,
- *addPProductConceptForProject()* - metoda služi za istanciranje novog objekta klase ***ProductConcept***, odnosno definiranje novog konceptualnog dijela proizvoda,
- *addOrgProject()* - metoda služi za istanciranje novog objekta klase ***OrganizationalProject***, odnosno novog projekta (proizvoda),
- *getProductConceptsForProjects()* - metoda služi za pretraživanje polja konceptualnih dijelova koji su definirani za pojedini proizvod,
- *getProjectsForOrganisation()* - metoda služi za pretraživanje polja projekata koji su dodijeljeni pojedinom timu,
- *removeProductConceptForProject()* – metoda služi za brisanje zadane instance iz polja konceptualnih dijelova pojedinog proizvoda,
- *removeOrgProject()* – metoda služi za brisanje zadane instance iz polja projekata,
- *updateOrgProject()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog projekta,
- *updateProductConcept()* – metoda služi za postavljanje vrijednosti atributa instance konceptualnog dijela proizvoda.

Objekti klase ***DocumentContoller*** istanciraju se u svrhu upravljanja vanjskim datotekama koje sadrže informacije o proizvodu. U klasi ***DocumentController*** definirani su slijedeći atributi i metode [Slika 6.11]:



Slika 6.11: Dijagram klasa *DocumentController*, *AppliedDocumentReference*, *DocumentType* i *DocumentFile*

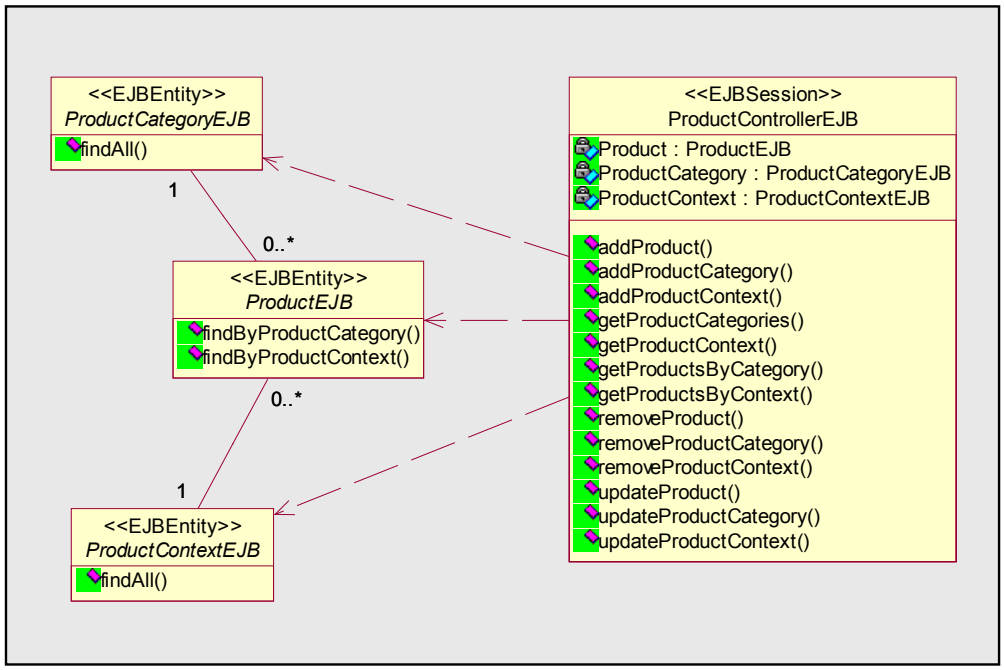
- *AppliedDocumentReference* – atribut tipa ***AppliedDocumentReference*** koji sadrži informaciju koja je vanjska datoteka pridružena pojedinoj reviziji komponente,
- *DocumentFile* – atribut tipa ***DocumentFile*** koji sadrži informacije o vanjskoj datoteci koja je nositelj informacija o proizvodu,
- *DocumentType* – atribut tipa ***DocumentType*** koji sadrži informacije o tipu vanjske datoteke
- *addDocumentFile()* - metoda služi za istanciranje novog objekta klase ***DocumentFile***, odnosno definiranje nove vanjske datoteke s informacijama o proizvodu,
- *AddDocumentReferenceForPDF()* - metoda služi za istanciranje novog objekta klase ***AppliedDocumentReference***, odnosno dodjeljivanje nove vanjske datoteke pojedinoj reviziji komponente,
- *addDocumentType()* - metoda služi za istanciranje novog objekta klase ***DocumentType***, odnosno definiranje novog tipa kojeg može biti vanjska datoteka,
- *getDocumentFilesByDocumentType()* - metoda služi za pretraživanje polja vanjskih datoteka prema njihovom tipu,

- *getDocumentReferenceByDocumentFile()* - metoda služi za pretraživanje polja revizija komponenti kojima je pridružen određena vanjska datoteka,
- *getDocumentReferenceForPDF()* - metoda služi za pretraživanje polja vanjskih datoteka koje su pridružene određenoj reviziji komponente,
- *getDocumentTypes()* - metoda služi za pretraživanje polja mogućih tipova vanjskih datoteka,
- *removeDocumentFile()* – metoda služi za brisanje zadane instance iz polja vanjskih datoteka,
- *removeDocumentType()* – metoda služi za brisanje zadane instance iz polja mogućih tipova vanjskih datoteka,
- *removeDocumentReferenceForPDF()* – metoda služi za brisanje zadane instance iz polja vanjskih datoteka koje su dodijeljene određenoj reviziji komponente,
- *updateDocumentType()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog mogućeg tipa vanjskih datoteka,
- *updateDocumentFile()* – metoda služi za postavljanje vrijednosti atributa instance pojedine vanjske datoteke.

Objekti klase ***ProductContoller*** istanciraju se u svrhu upravljanja komponentama proizvoda. U klasi ***ProductController*** definirani su slijedeći atributi i metode [Slika 6.12]:

- *Product* – atribut tipa ***Product*** koji sadrži informacije o pojedinoj komponenti proizvoda (fizički dio ili strukturirani dokument),
- *ProductCategory* – atribut tipa ***ProductCategory*** koji sadrži informacije o kategoriji kojoj pripada pojedina komponenta,
- *ProductContext* – atribut tipa ***ProductContext*** koji sadrži informacije o kontekstu u kojem se definiraju pojedine komponente,
- *addProduct()* - metoda služi za istanciranje novog objekta klase ***Product***, odnosno definiranje komponente proizvoda,
- *addProductCategory()* - metoda služi za istanciranje novog objekta klase ***ProductCategory***, odnosno definiranje nove kategorije komponenti,
- *addProductContext()* - metoda služi za istanciranje novog objekta klase ***ProductContext***, odnosno definiranje novog konteksta za komponente,
- *getProductCategories()* - metoda služi za pretraživanje polja mogućih kategorija komponenti,
- *getProductContext()* - metoda služi za pretraživanje polja mogućih konteksta komponenti,
- *getProductsByCategory()* - metoda služi za pretraživanje polja komponenti

prema kategoriji kojoj su dodijeljeni,



Slika 6.12:Dijagram klasa *ProductController*, *Product*, *ProductContext* i *ProductCategory*

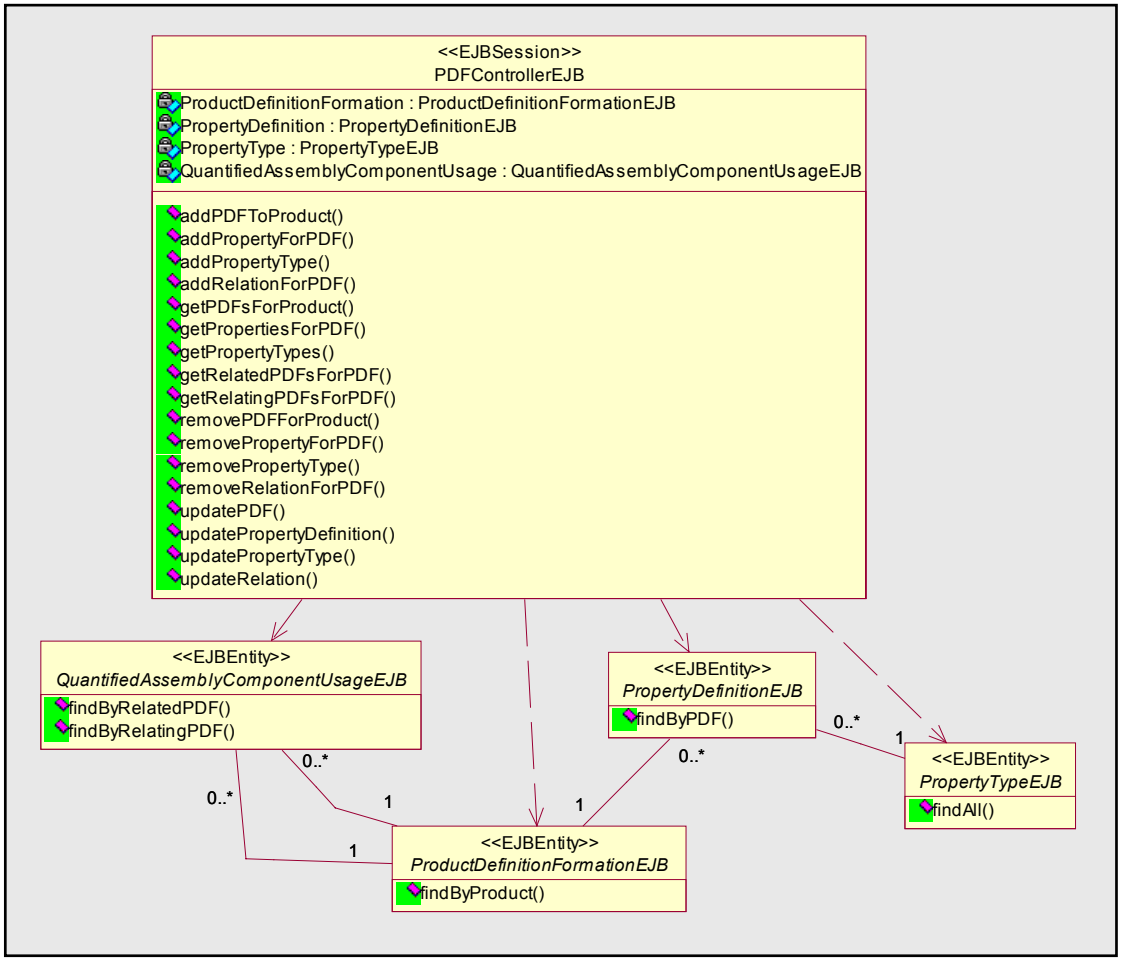
- *getProductsByContext()* - metoda služi za pretraživanje polja komponenti prema kontekstu u kojem su definirane,
- *removeProduct()* – metoda služi za brisanje zadane instance iz polja komponenti,
- *removeProductCategory()* – metoda služi za brisanje zadane instance iz polja mogućih kategorija komponenti,
- *removeProductContext()* – metoda služi za brisanje zadane instance iz polja mogućih konteksta u kojima se definiraju komponente,
- *updateProduct()* – metoda služi za postavljanje vrijednosti atributa instance pojedine komponente,
- *updateProductCategory()* – metoda služi za postavljanje vrijednosti atributa instance pojedine moguće kategorije,
- *updateProductContext()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog mogućeg konteksta.

Objekti klase ***PDFController*** istanciraju se u svrhu upravljanja revizijama pojedinih komponenti, hijerarhijskih relacija između revizija i svojstvima revizija. U klasi ***PDFController*** definirani su slijedeći atributi i metode [Slika 6.13]:

- *ProductDefinitionFormation* – atribut tipa ***ProductDefinitionFormation*** koji sadrži informacije o pojedinoj reviziji komponente proizvoda,
- *PropertyDefinition* – atribut tipa ***PropertyDefinition*** koji sadrži informacije o svojstvu pojedine revizije,

- *PropertyType* – atribut tipa ***PropertyType*** koji sadrži informacije o tipu mogućeg svojstva koje se može definirati za pojedinu reviziju,
- *QuantifiedAssemblyComponentUsage* - atribut tipa ***QuantifiedAssembly-ComponentUsage*** koji sadrži informacije o relaciji koja je definiran između dvije revizije različitih komponenti
- *addPDFToProduct()* - metoda služi za istanciranje novog objekta klase ***ProductDefinitionFormation***, odnosno definiranje revizije pojedine komponente proizvoda,
- *addPropertyForPDF()* - metoda služi za istanciranje novog objekta klase ***PropertyDefinition***, odnosno definiranje svojstva pojedine revizije,
- *addRelationForPDFs()* - metoda služi za istanciranje novog objekta klase ***QuantifiedAssembly-ComponentUsage***, odnosno definiranje strukturne relacije roditelj-dijete između dvije revizije,
- *getPDFsForProduct()* - metoda služi za pretraživanje polja revizija koje su definirane za komponentu,
- *getPropertiesForPDF()* - metoda služi za pretraživanje polja dodijeljenih svojstava za pojedinu reviziju,
- *getPropertyTypes()* - metoda služi za pretraživanje polja mogućih tipova svojstava koja se mogu dodijeliti reviziji,
- *getRelatedPDFsForPDF()* - metoda služi za pretraživanje polja nadređenih revizija za pojedinu reviziju,
- *getRelatingPDFsForPDF()* - metoda služi za pretraživanje polja podređenih revizija za pojedinu reviziju,
- *removePDFForProduct()* – metoda služi za brisanje zadane instance iz polja revizija pojedine komponente,
- *removePropertyForPDF()* – metoda služi za brisanje zadane instance iz polja svojstava za jednu reviziju,
- *removePropertyType()* – metoda služi za brisanje zadane instance iz polja mogućih tipova svojstava koja se mogu dodijeliti revizijama,
- *removeRelationForPDF()* – metoda služi za brisanje zadane instance iz polja relacija za pojedinu reviziju,
- *updatePDF()* – metoda služi za postavljanje vrijednosti atributa instance pojedine revizije,
- *updatePropertyDefinition()* – metoda služi za postavljanje vrijednosti atributa instance svojstva koje je dodijeljeno reviziji,
- *updatePropertyType()* – metoda služi za postavljanje vrijednosti atributa instance pojedinog mogućeg tipa svojstava,

- *updateRelation()* – metoda služi za postavljanje vrijednosti atributa instance pojedine relacije koja je definirana između revizija.



Slika 6.13: Dijagram klasa PDFController, QuantifiedAssemblyComponentUsage, ProductDefinitionFormation, PropertyDefinition, PropertyType

Dodatni moduli web servisa

Za potpunu funkcionalnost u strukturi web servisa je potrebno predviđeni su pomoćni module kojima se web servis može nadograditi. Glavni predstavnici te grupe čine moduli za:

- *Autorizaciju korisnika* – prilikom korištenja web servisa korisnik se mora prijavit za rad, te proći proces autorizacije. Na temelju identiteta korisnika, određuju se njegova prava i uloge, te ovisno o tome dinamički prilagođava radna okolina, odnosno mogućnost korištenja pojedinih funkcija web servisa.
- *Generiranje hijerarhije* – obzirom da su entiteti iz domene metamodela, predstavljani kao niz objekata, potrebno je za lakše snalaženje korisnika u njihovoj hijerarhijskoj strukturi, kreirati prikaz te strukture. To znači da je potrebno iz objekata i njihovih relacija generirati prikaz ovisno o razini hijerarhije koju korisnik promatra.

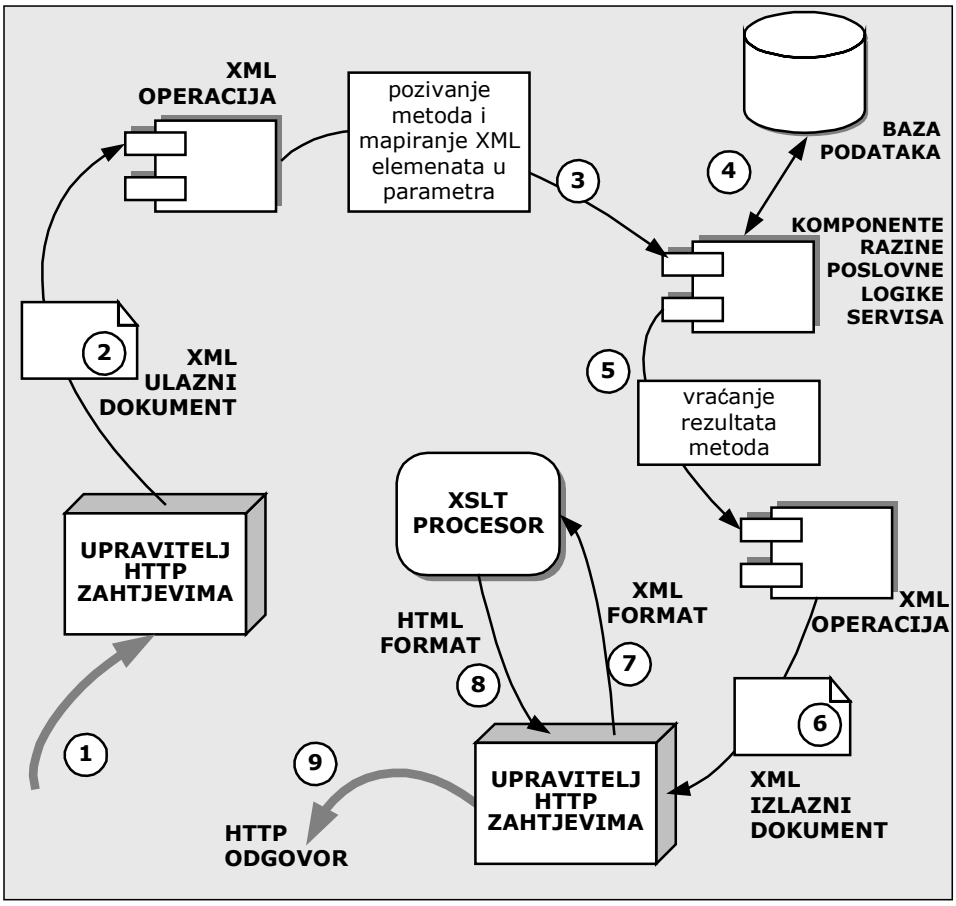
- *Generiranje identifikacijske oznake* - specifično za svaku pojedinu vrstu entiteta iz domene metamodela. Zbog zahtjeva identifikacije, svaki entitet mora imati unificiranu oznaku, koja se može generirati ovisno o vrsti entiteta. (npr. projekti mogu imati jedan sustav označavanja, komponente drugi, revizije treći itd.).
- *Upload/download datoteka* – tijekom rada, korisnici fizičke datoteke na kojima rade prebacuju u svoju radnu okolinu, te se mora osigurati kontrola tog prebacivanja, odnosno povratka promijenjenih datoteka, uz definiranje pravila za osiguranje konzistentnosti datoteka.
- *Podršku kreiranju i upravljanju sadržaja strukturiranih dokumenata* – ukoliko se od sustava zahtjeva mogućnost kreiranje i manipuliranja sadržajem strukturiranih dokumenata potrebno je kreirati modul koji će tu funkcionalnost implementirati.
- *Generiranje izvještaja* – kao izlaz iz sustava potrebno je definirati specifične izvještaje ovisno o potrebama i zahtjevima svake realne implementacije.
- *Arhiviranje završenih projekta* – potrebno je definirati postupke i procedure za trajnu pohranu gotovih projekata.
- *Podešavanje radnih postavki*- za ispravo funkcioniranje servisa, potrebno je omogućiti administratorima kontrolu svih postavki i mogućnost njihovih promjena.
- *Pružanje pomoći korisnicima* – potrebno je kreirati sustav pomoći koji će omogućiti korisnicima pružanje pravodobnih informacija o korištenju i svojstvima servisa.

Bitno je naglasiti da svaki od prethodno spomenutih modula zahtjeva detaljnu analizu i kreiranje zasebne strukture klasa koje bi implementirale tražena djelovanja. Stoga analiza i implementacija svakog pojedinog modula predstavlja otvorena vrata i poziv za suradnju u smislu novih spoznaja i nadogradnje rezultata istraživanja opisanog u ovom radu.

6.4.3 XML operacije

XML operacija je naziv za logičke entitete koji opisuju i implementiraju mehanizme komunikacije između klijenata i web servisa. Bitno je naglasiti da svaka XML operacija predstavlja odgovor na točno određeni zahtjev klijenta web servisu. XML operacije se programiraju pozivanjem određenog broja metoda na objektima koji se istanciraju iz klasa u poslovnoj razine web servisa. XML operacije primaju zahtjev klijenta o obliku XML dokumenta (naziva se XML ulazni dokument). Elementi XML ulaznog dokumenta se zatim preslikavaju u parametre koje XML operacija prosljeđuje metodama koje poziva. Nakon što metode završe sa obradom podataka vraćaju rezultate obrade XML operaciji koja ih formatira u izlazni XML dokument redoslijed događaja prilikom obrade zahtjeva klijenta prikazuje [Slika 6.14].

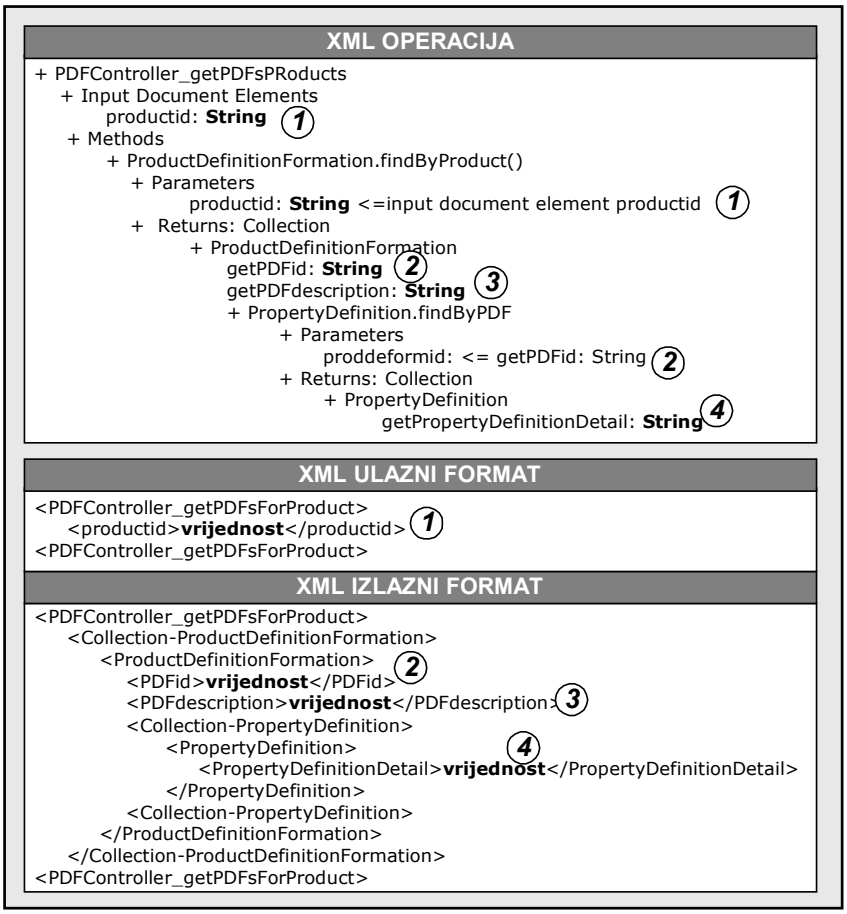
Ključni korak u definiranju XML operacija je određivanje metoda koje će operacija pozivati te redoslijeda njihovog pozivanja. U jednoj XML operaciji je moguće pozivati više metoda, uključujući i pozivanje metode različitih objekata. Agregacijom više metoda različitih objekata u jednu XML operaciju omogućuje se prezentiranje podataka koje one vraćaju kao jedne logičke cjeline.



Slika 6.14: Uloga XML operacija u radu web servisa

Kao primjer mogu se razmotriti značajke XML operacije za dobivanje podataka o svim svojstvima svih revizija neke komponente [Slika 6.15] definirane u web servisu. Na slici su prikazana tri ključna segmenta koji čine XML operaciju. U gornjem dijelu slike su prikazane metode pojedinih objekata koje je XML operacija poziva točno redoslijedom kojim su tu navedene. U tu svrhu konkretna XML operacija najprije šalje ulazni parametar sa vrijednošću identifikacijske oznake komponente, metodi ***findbyProduct()*** definiranoj u klasi ***ProductDefinitionFormation***. Poziv te metode vraća kolekciju objekata koji predstavljaju revizije definirane za traženu komponentu. Nakon toga, se za svaku reviziju koja je definirana svojim identifikacijskim parametrom poziva metoda koja vraća kolekciju svojstava definiranih za tu reviziju. Korištenje ovako definirane XML operacije zahtijeva kreiranje XML ulaznog dokumenta koji nosi vrijednost parametra koji govori o identifikaciji komponente za koju se traže podaci o revizijama. Isto tako XML operacija formatira XML izlazni dokument koji strukturirano prikazuje vraćene

podataka. Na slici je brojevima u kružićima naznačeno kako se pojedine vrijednosti, koje vraćaju metode, preslikavaju u odgovarajuće elemente XML ulaznog i izlaznog dokumenta.



Slika 6.15: Preslikavanje metoda poslovnih komponenti u XML operacije

Osnovna karakteristika XML izlaznog dokumenta je upravo njegova prilagodljivost. Naime, imena tagova kojima se definira kontekst podataka koje tag sadrži, mogu se definirati proizvoljno, a moguće je i točno određivanje koji od podataka koje vraćaju metode objekata će biti uključeni u XML izlaznom dokumentu, onemogućujući na taj način, pojedinim korisnicima dostupnost svim podacima.

XML operacije se implementiraju kao izvršne klase web servisa, te se opisuju u registrima, koji sadrže informacije o njima i strukturi podataka s kojima upravljaju. Na temelju zapisa u registrima, kao integralni dio web servisa kreiraju se komponente koje predstavljaju most između zahtjeva klijenata i poziva odgovarajućih XML operacija opisanim u registrima web servisa. Te komponente također implementiraju logiku za prikaz podataka koji se vraćaju klijentima, odnosno transformaciju XML izlaznih dokumenta u oblik prilagođen klijentu. XML operacije web servisa za upravljanje i razmjenu informacijama o proizvodu, definiraju se u svrhu klijentskih poziva metodama komponenti iz razine poslovne logike opisanih u poglavlju 6.4, a po opisanom i prikazanom [Slika 6.15] principu.

7

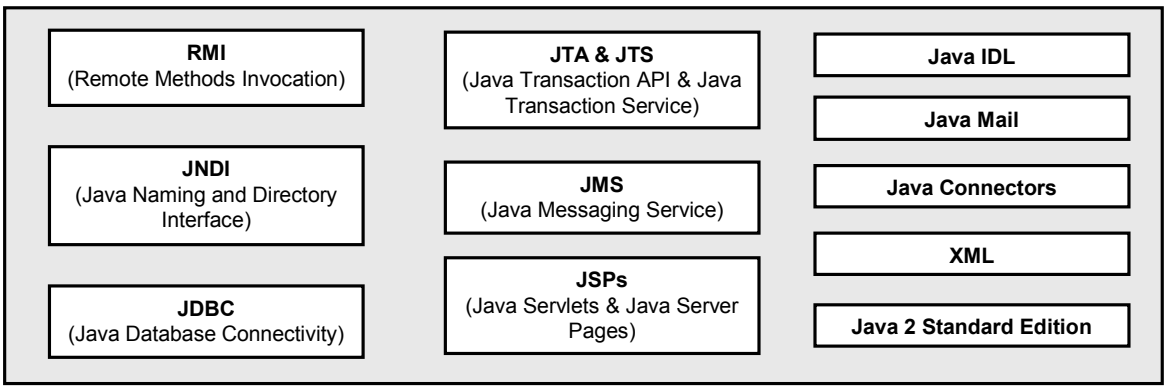
PROTOTIPNA REALIZACIJA WEB SERVISA

U sedmoj je glavi opisana prototipna realizacija web servisa za razmjenu i upravljanje informacijama o proizvodu korištenjem Java 2 razvojne platforme, odnosno J2EE© (eng. Java2 Enterprise Edition) specifikaciju tehnologija tvrtke Sun Microsystems©. Navedena razvojna platforma je izabrana zbog njezinih karakteristika koje ju definiraju kao platformski neutralnu, portabilnu, višekorisničku i sigurnu osnovu za razvoj distribuirani programskih aplikacija, što u potpunosti odgovara zahtjevima preložene implementacijske metodologije.

7.1 Rješenje zasnovano na J2EE© razvojnoj platformi

Za verifikaciju dosadašnjih izlaganja te provjeru prethodno opisane koncepcije, web servis za razmjenu i upravljanje informacijama o proizvodu je programski realiziran korištenjem Java2© razvojne platforme, odnosno J2EE© (*eng. Java2 Enterprise Edition*) specifikacije tehnologija tvrtke *Sun Microsystems*© [75], [76], [77], [78], [79], [81]. Navedena platforma i razvojna okolina, odabrani su zbog otvorenosti i pristupačnosti programskih alata i gotovih serverskih komponenti potrebnih za programsku realizaciju predloženog servisa. Pored toga je važan kriterij u odabiru realizacijske platforme bilo i svojstvo hardversko-softverske nezavisnosti J2EE© specifikacije, što omogućuje razvoj aplikacija koje je moguće koristiti na različitim računalnim platformama sa različitim operativnim sustavima. Upravo ta raznolikost je i osnovna značajka radne okoline konstruktora, prisutne u području primjene raznovrsnih aplikacija prilikom distribuiranog razvoja proizvoda.

Tehnologije koje su korištene prilikom realizacije sustava, a koje su uključene u J2EE© specifikaciju jesu [Slika 7.1]:



Slika 7.1: Java 2 razvojna platforma

- **Enterprise JavaBeans (EJB)** – koriste se za kreiranje serverskih komponenti odgovornih za poslovnu logiku servisa te definiraju način upravljanja poslovnim komponentama u procesu njihovog korištenja. EJB su osnova J2EE© platforme, te omogućuju neovisan razvoj, razmjenu i korištenje istih komponenti u različitim aplikacijama.
- **Java Remote Method Invocation (RMI)** – servis odgovoran za komunikaciju između komponenti korištenjem Internet komunikacijskih protokola.
- **Java Naming and Directory Interface (JNDI)** – služi za identifikaciju lokacije komponente ili ostalih potrebnih resursa širom mreže. Ovo je ključna tehnologija koja omogućuje klijentima pristup točno određenim EJB komponentama.
- **Java Database Connectivity (JDBC)** – predstavlja standardno sučelje za pristup relacijskim bazama podataka i izvođenje operacija nad podacima koji su tamo pohranjeni.
- **Java Transaction API (JTA) & Java Transaction Service (JTS)** – ove specifikacije pružaju podršku izvođenju transakcija prilikom izvođenja aplikacija, osiguravajući na taj način mnogostrukim korisnicima mogućnost rada sa istim podacima i sinkronizaciju podataka između klijenata i servisa.
- **Java Messaging Service (JMS)** – servis koji omogućuje asinkronu komunikaciju između distribuiranih komponenti aplikacija.
- **Java Servlets and Java Server Pages (JSPs)** – komponente koje služe za kreiranje klijentskih aplikacija koje servisu pristupaju putem HTTP protokola korištenjem specifičnih zahtjev/odgovor (*eng. request/response*) mehanizma.
- **Java IDL** – tehnologija koja omogućuje integraciju J2EE© rješenja s ostalim programskim jezicima, implementacijom CORBA specifikacije u Javi.
- **Java Mail** – servis koji omogućuje slanje email poruka na platformski neutralan i protokolski neovisan način iz Java aplikacija.
- **Java Connectors** – služe za integraciju J2EE platforme s postojećim

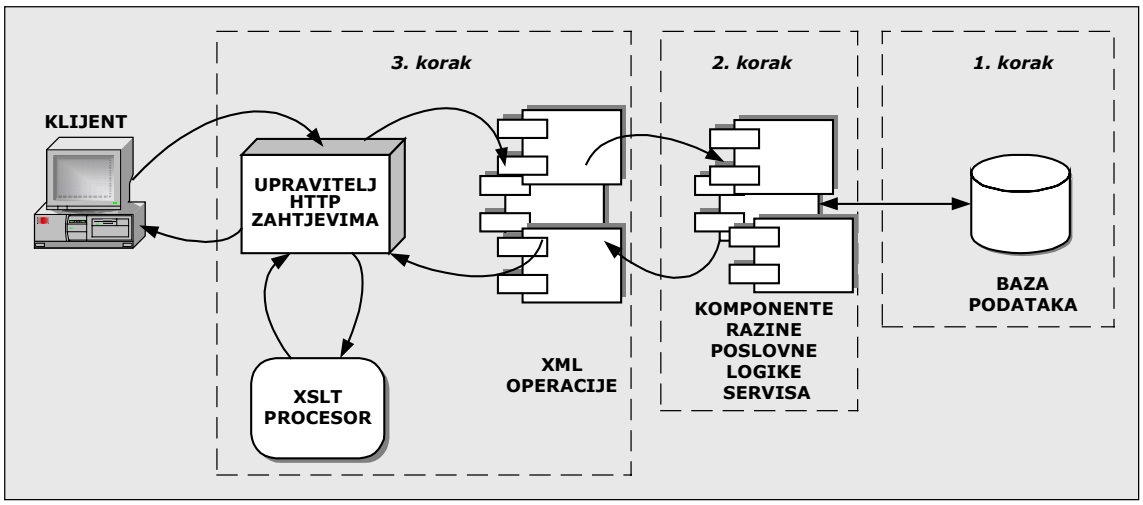
informatičkim sustavima (ERP, MOM, BEA's Tuxedo, IBM's CICS & IMS).

- **The Extensible Markup Language (XML)** – služi kao format za razmjenu podataka i opisivanje komponenti prilikom njihovog objavljivanja.
- **Java2 Standard Edition (J2SE)** – specifikacija osnova Java jezika i servisa definirana kroz mnogostruke biblioteke (.awt, .net, .io).

Potrebno je još naglasiti da je web servis kreiran tijekom istraživanja prikazanog u ovom radu, realiziran korištenjem *Forte for Java©*, *Enterprise Edition, 3.0* [81] razvojne okoline za kreiranje rješenja koja se zasnivaju na J2EE© platformi.

Programska realizacija i testiranje web servisa ostvarena je u tri koraka [Slika 7.2]:

- kreiranje infrastrukture relacijske baze,
- kreiranje i testiranje komponenti poslovne razine web servisa,
- kreiranje i testiranje XML operacija.



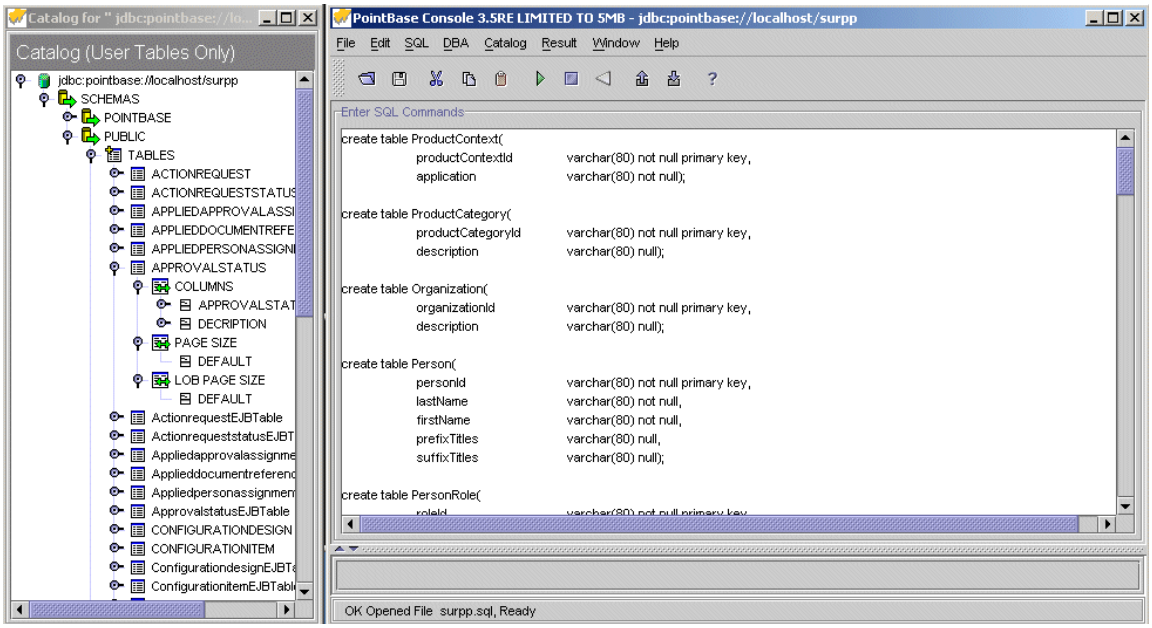
Slika 7.2: Koraci u realizaciji i testiranju predložene arhitekture web servisa

Kreiranje infrastrukture relacijske baze

Početni korak bio je kreiranje infrastrukture relacijske baze podatka u koju će se trajno spremati vrijednosti atributa pojedinih entiteta definiranih informacijskim modelom opisanim u poglavlju 4.3. Korištenjem SQL izraza definiranog u poglavlju 6.3, kreirane su potrebne tablice i relacije asocijacije između polja pojedinih tablica, u mrežno orijentiranoj relacijskoj bazi *Pointbase Network Server*, koja je integralni dio korištenog razvojnog okoline [Slika 7.3].

Kreiranje i testiranje komponenti poslovne razine web servisa

Slijedeći korak u realizaciji web servisa bio je kreiranje i testiranje komponenti koje su odgovorne za izvršavanje poslovne logike servisa. Komponente su izgrađene na osnovi strukture klasa koja je definirana u poglavlju 6.4. Prema analizi koja je opisana u tom poglavlju, kreirane su tri osnovne grupe klasa poslovne razine.



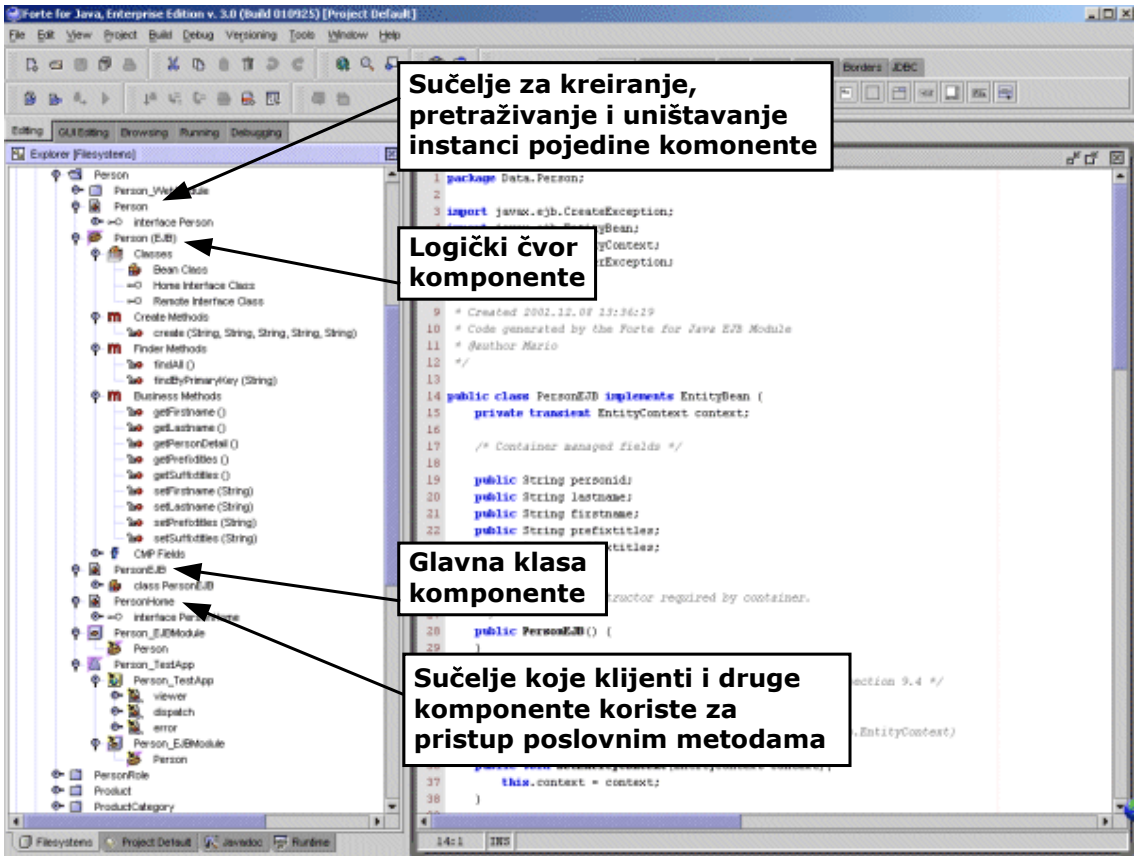
Slika 7.3: Kreiranje tablica i relacija u relacijskoj bazi

Prva od njih, grupa klasa koje modeliraju operacije trajnog zapisa podataka u relacijsku bazu, realizirana je korištenjem " *eng. entity EJB*" komponenti. Uloga tih komponenti (kao dijela J2EE© rješenja) je objektno-orijentirano manipuliranje podacima koji se trajno zapisuju (kao npr. ime i prezime korisnika, ID sklopa, naziv pojedine revizije, putanja do eksterne reference koja sadrži dodatne informacije o proizvodu, informacija o odgovornom korisniku, stupanj odobrenja itd.). Ovako kreirane komponente preslikavaju vrijednosti atributa pojedinih objekata u odgovarajuće tablice relacijske baze. Za vrijeme korištenja servisa svaki slog pojedine tablice je u memoriji računala predstavljen zasebnom instancom odgovarajuće klase. Pohranjivanjem vrijednosti atributa instanci u polja odgovarajućeg sloga, trajno se čuva zadnje stanje promjena podataka i nakon što korisnik završi s izvršavanjem operacija koje su mu dozvoljene nad podacima.

Komponente ovog tipa sadrže programsku logiku koja je odgovorna za učitavanje vrijednosti atributa prilikom istanciranja objekata i spremanje svih promjena vrijednosti njihovih atributa. Jedna od prednosti ovakvog pristupa je u naprednijim mogućnostima manipuliranja podacima kojima se u konkretnom slučaju realizacije manipulira kao objektima. Moguće je između ostalog, pozivati odgovarajuće metode implementirane u instancama, kao npr. metode za kreiranje odgovarajućeg ID-a prilikom istanciranja novog projekta, koncepta, konfiguracije ili komponente proizvoda. Za svaki od entiteta informacijskog modela prikazanog u poglavlju 4.3. kreirane su "*eng. entity EJB*" komponente. Za svaku od tih komponenti kreirana je glavna klasa koja modelira podatke, sučelje koje koriste klijenti i druge komponente za pristup poslovnim metodama definiranim u glavnoj klasi, sučelje koje klijenti koriste za kreiranje, pretraživanje i uništavanje pojedinih objekata glavne klase, klasa primarnog ključa koja definira jedinstvenost svakog objekta [Slika 7.4].

Za testiranje komponenti definiranih na ovaj način, kreirani su web klijenti sa

svrhom potvrđivanja ispravnog rada komponente, odnosno ispravnosti kreiranja novih instanci, manipuliranja trajnom pohranom vrijednosti njezinih atributa te rada poslovnih metoda [Slika 7.5]. Korisničko sučelje klijenata za testiranje, podijeljeno je u tri područja. Gornji dio korisničkog sučelja namijenjen je praćenju instanci koje se kreiraju tijekom korištenja komponente. U donjem dijelu su prikazane metode koje je moguće pozvati ovisno o kreiranoj instanci, sa poljima u koja se unose potrebni parametri prilikom pozivanja metode. Srednji dio sučelja namijenjen je praćenju rezultata koje vraća zadnja pozvana metoda.



Slika 7.4: Elementi "*eng. Entity EJB*" komponente

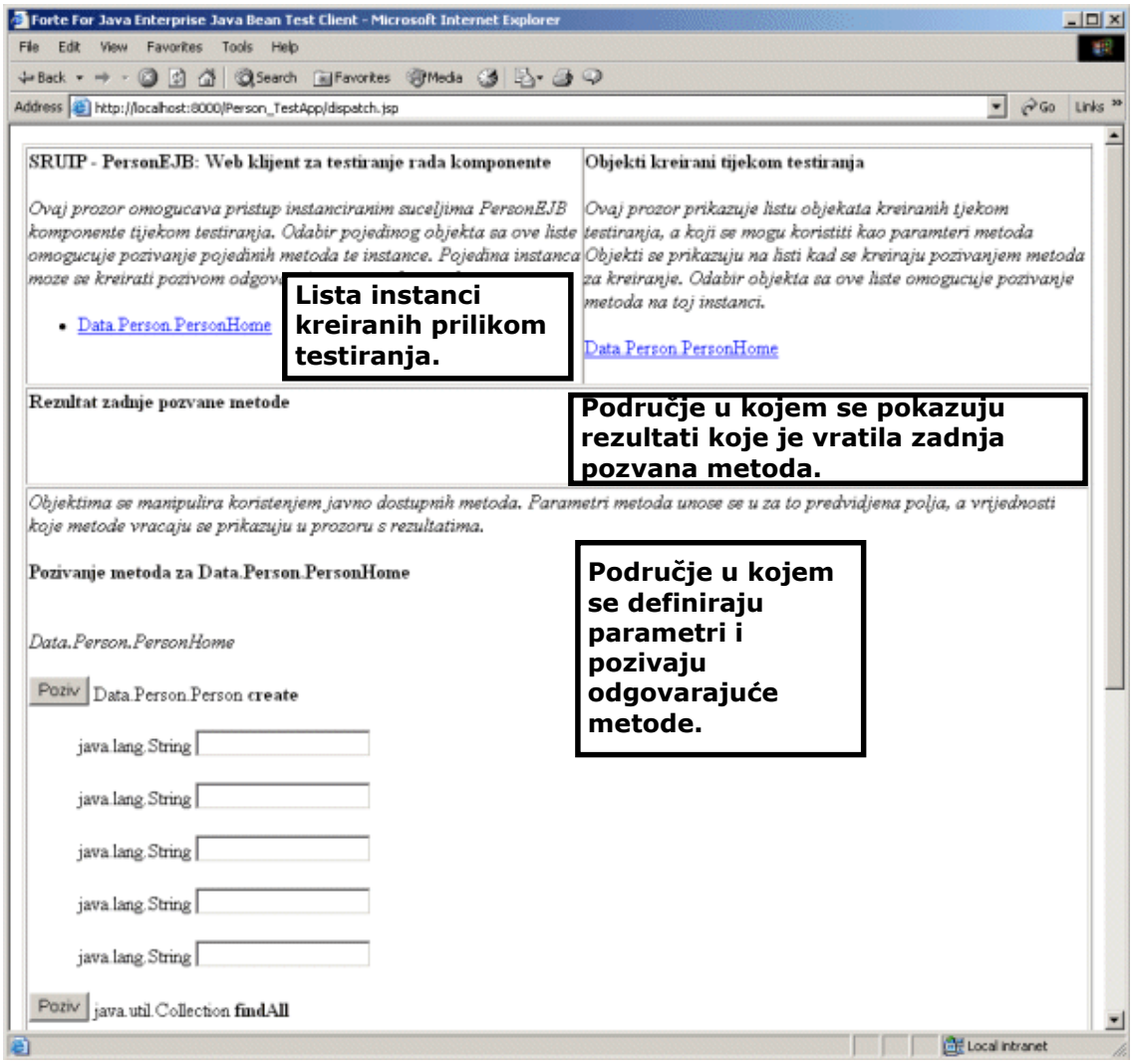
Korištenje ovako kreiranih klijenata ilustrirat će se na primjeru komponente koja je odgovorna za upravljanje trajnim zapisom podataka o korisnicima sustava - ***PersonEJB***. Novi korisnik sustava kreira se pozivanjem metode ***create()***, koja prilikom poziva zahtijeva parametre sa vrijednostima identifikacijske oznake korisnika, imena, prezimena, te titule [Slika 7.6]. Nova instanca korisnika u memoriji može se kreirati i kao rezultat pozivanja metoda za pretraživanja postojećih pohranjenih podataka.

Vrijednosti parametara prilikom kreiranja novog korisnika upisuju se u za to predviđena tekstualna polja, koja su na sučelju poredana prema redoslijedu kolona u tablici u koju se vrijednosti trajno spremaju. Pored svakog polja za unos parametara ispisan je i očekivani tip podatka za pojedini parametar – u konkretnom slučaju to je tekstualni tip podataka. Nakon pozivanja metode u odgovarajućoj tablici relacijske

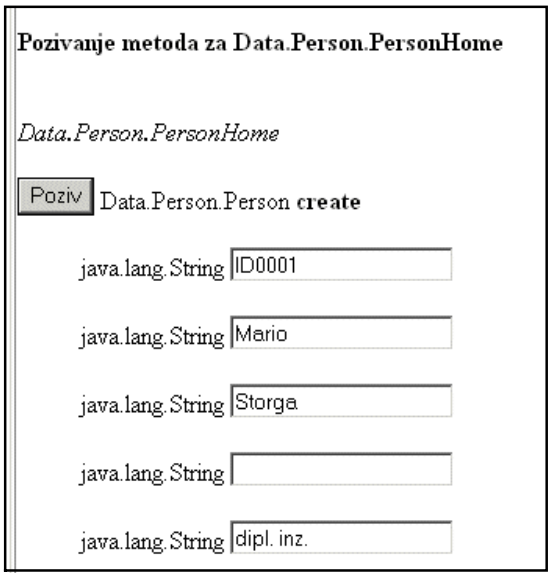
baze kreira se novi redak, a u računalnoj memoriji nova instanca klase ***PersonEJB***. Tako novokreiranoj instanci može se pristupiti preko stabla kreiranih instanci u gornjem dijelu klijentskog sučelja [Slika 7.7]. Rezultati koje vraća metoda za kreiranje novog korisnika mogu se pratiti u za to predviđenom dijelu klijentskog sučelja [Slika 7.8].

Odabirom kreirane instance iz gornjeg dijela klijentskog sučelja, korisniku postaju dostupne metode poslovne metode definirane za tu instancu. U konkretnom slučaju, to su metode za čitanje vrijednosti atributa pojedine instance korisnika, mijenjanje vrijednosti atributa instanci korisnika i sinkronizacija vrijednosti trajno pohranjenih u relacijskoj bazi, te brisanje instance korisnika što znači i fizičko brisanje odgovarajućeg retka iz relacijske baze [Slika 7.9]. Rezultati pozivanja svake od tih metoda također se mogu pratiti u za to predviđenom dijelu korisničkog sučelja.

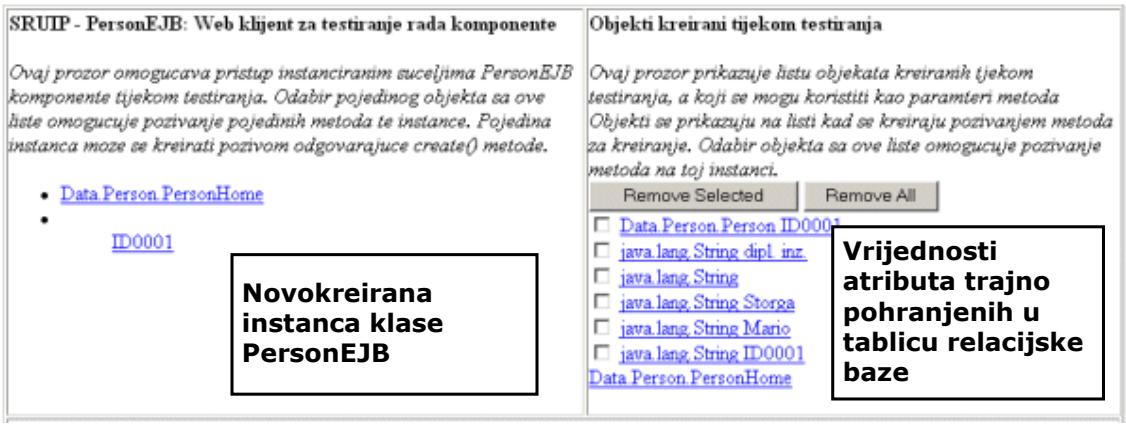
Opisani princip rada klijenata za testiranje, jednak i za ostale komponente koje modeliraju operacije trajnog zapisa ostalih podataka u relacijsku bazu.



Slika 7.5: Sučelje web klijenta za testiranje komponenti koje upravljaju trajnim zapisom podataka u relacijsku bazu



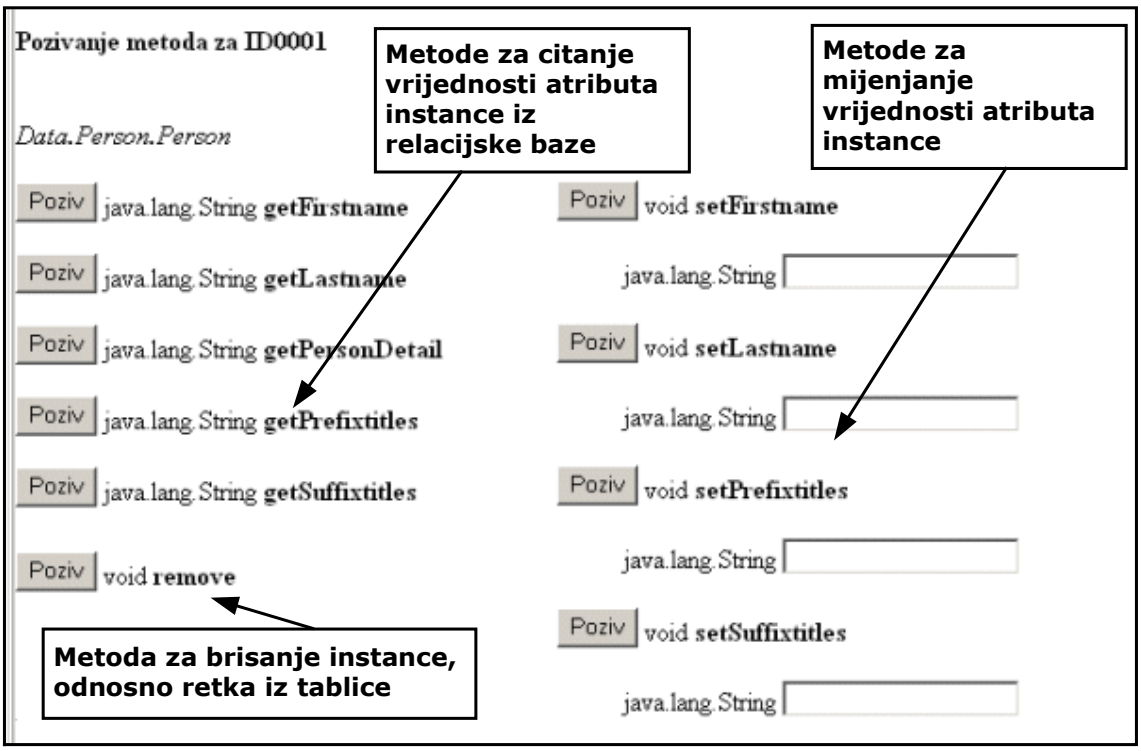
Slika 7.6: Kreiranje nove instance klase *PersonEJB*



Slika 7.7: Prikaz kreirane instance i vrijednosti koje su pohranjene u relacijsku bazu

Slika 7.8: Rezultati pozivanja metode za kreiranje novog korisnika

Klase druge grupe, koje modeliraju poslovnu logiku i pravila manipuliranja podacima, realizirane su korištenjem *eng. session EJB* komponenti. Svrha tih komponenti kao dijela J2EE© rješenja je objektno-orijentirano manipuliranje poslovnim procesima web servisa. Pod pojmom poslovnih procesa možemo definirati bilo koju zadaću koja uključuje logiku, algoritme ili tok podataka (kontrola i upravljanje korisnicima i radnim grupama, provjera i dodjeljivanje odobrenja, konfiguriranje proizvoda, kreiranje i mijenjanje koncepta proizvoda, upravljanje revizijama komponenti itd.).

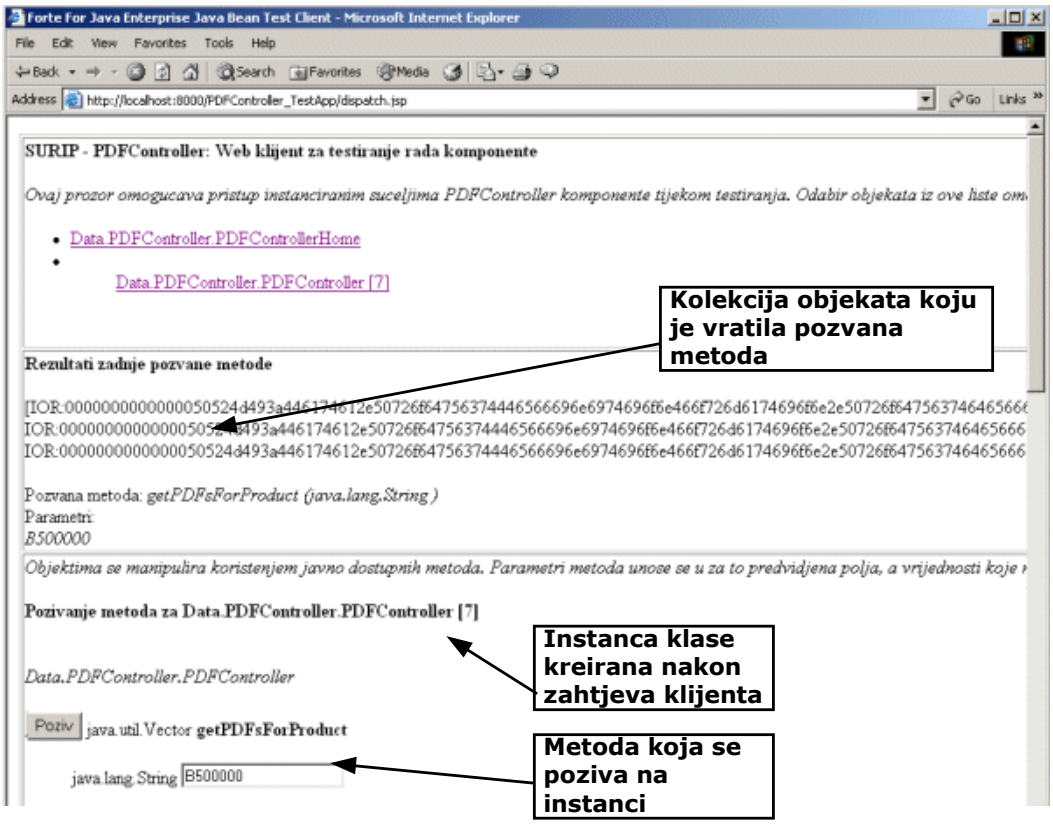


Slika 7.9: Metode dostupne za svaku instancu klase *PersonEJB*

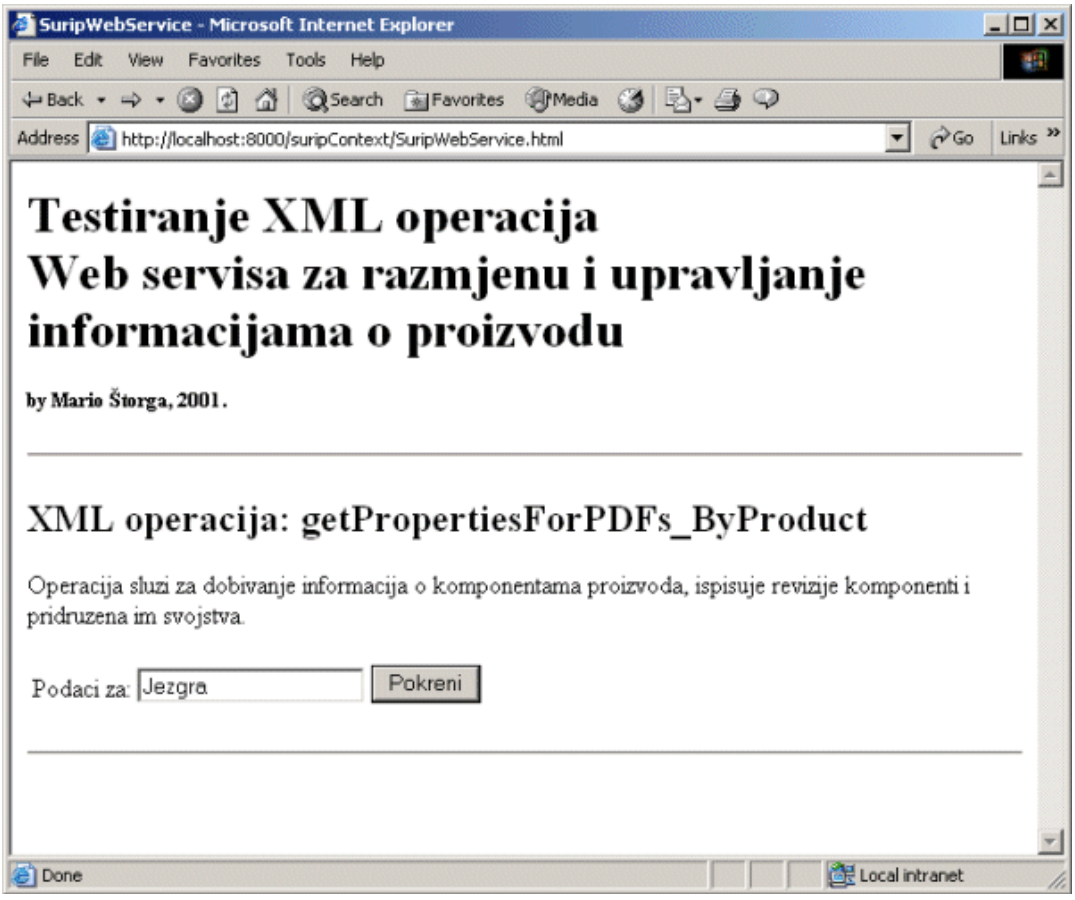
Ovakve komponente kreirane su za rad s jednim korisnikom istovremeno, te njihov životni vijek traje koliko i pojedini proces koji korisnik izvršava nad podacima. Logika implementirana u ovim komponentama štiti krajnjeg korisnika od kompleksnosti upravljanja podacima. To možemo ilustrirati na primjeru komponente odgovorne za kreiranja nove pozicije u proizvodu, kojom je definirano da je najprije potrebno pozvati metodu koja će kreirati novi ID, zatim zapisati vrijeme kreiranja, odgovornog korisnika, kreirati prvu reviziju pozicije te pridružiti tu reviziju nekom nadređenom sklopu, dodijeliti početni status, jer revizija u tom trenutku još nije odobrena. Komponenta sa ovakvom logikom se istancira svaki puta kad neki od krajnjih korisnika želi kreirati novu poziciju.

Struktura ovih komponenti slična je prethodno opisanim *eng. entity EJB* komponentama, s razlikom da u njima nije implementirana logika za trajno spremanje vrijednosti atributa pojedinih instanci. Ideja ovakvog pristupa je da se ovim komponentama, za vrijeme njihovog korištenja, pozivaju pojedine metode komponenti koje su odgovorne za trajno spremanje podatka u relacijsku bazu.

Za drugu grupu komponenti, također su kreirani testni web klijenti sa ciljem potvrđivanje ispravnog rada. To možemo ilustrirati na primjeru koji ilustrira samo dio rada komponente kreirane na osnovi klase ***PDFController***. Navedena komponenta implementira logiku za upravljanje revizijama pojedinih komponenti hijerarhijske strukture proizvoda, uključujući i svojstva revizija, te vanjske reference, prema informacijskom modelu [Slika 7.10].



Slika 7.10: Sučelje web klijenta za testiranje komponenti *PDFController*

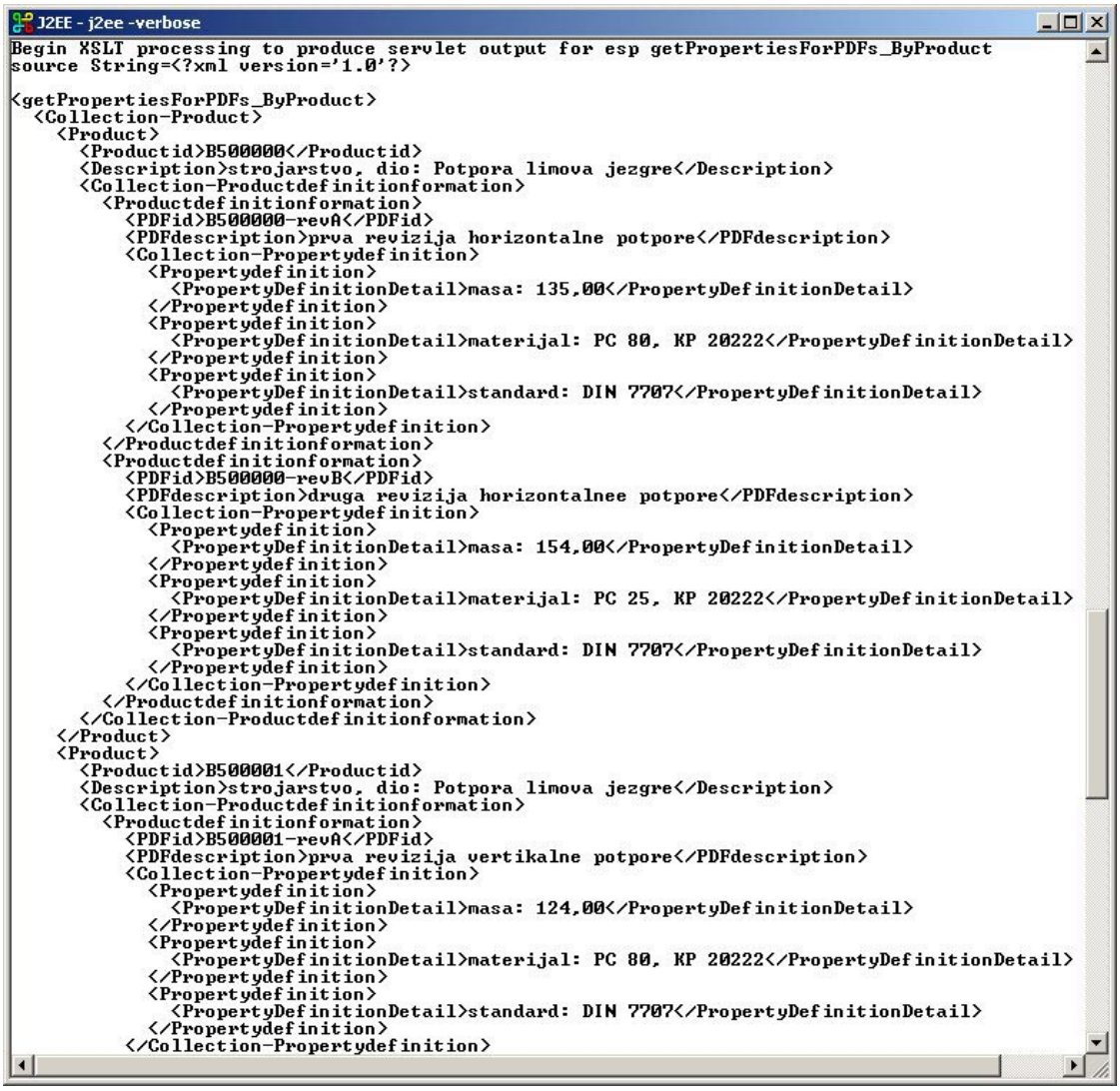


Slika 7.11: Sučelje web klijenta za testiranje XML operacije

Da bi krajnji korisnik mogao upravljati revizijama, potrebno je najprije kreirati novu instancu komponente ***PDFController***, koja je odgovorna samo za taj zahtjev korisnika, te će nakon što ga izvrši jednostavno biti uništena. Nakon što je instanca kreirana moguće je pristupiti njenim metodama. Jedna od tih metoda, koja se poziva u sklopu dobivanja informacije o pohranjenim revizijama određene pozicije proizvoda, zahtjeva kao ulazni parametar ID pozicije, te kao rezultat vraća kolekciju objekata klase ***ProductEJB*** koji predstavljaju podatke trajno pohranjene u relacijskoj bazi, što je prikazano na gornjoj slici. Nakon što su takvi objekti istancirani, moguće je nad podacima koje ti objekti predstavljaju izvoditi potrebne procese obrade. Ovo je samo jednostavna ilustracija korištenja testnog klijenta, koji su po istom principu kreirani za svih devet upravljačkih klasa opisanih u poglavlju 6.4.2.

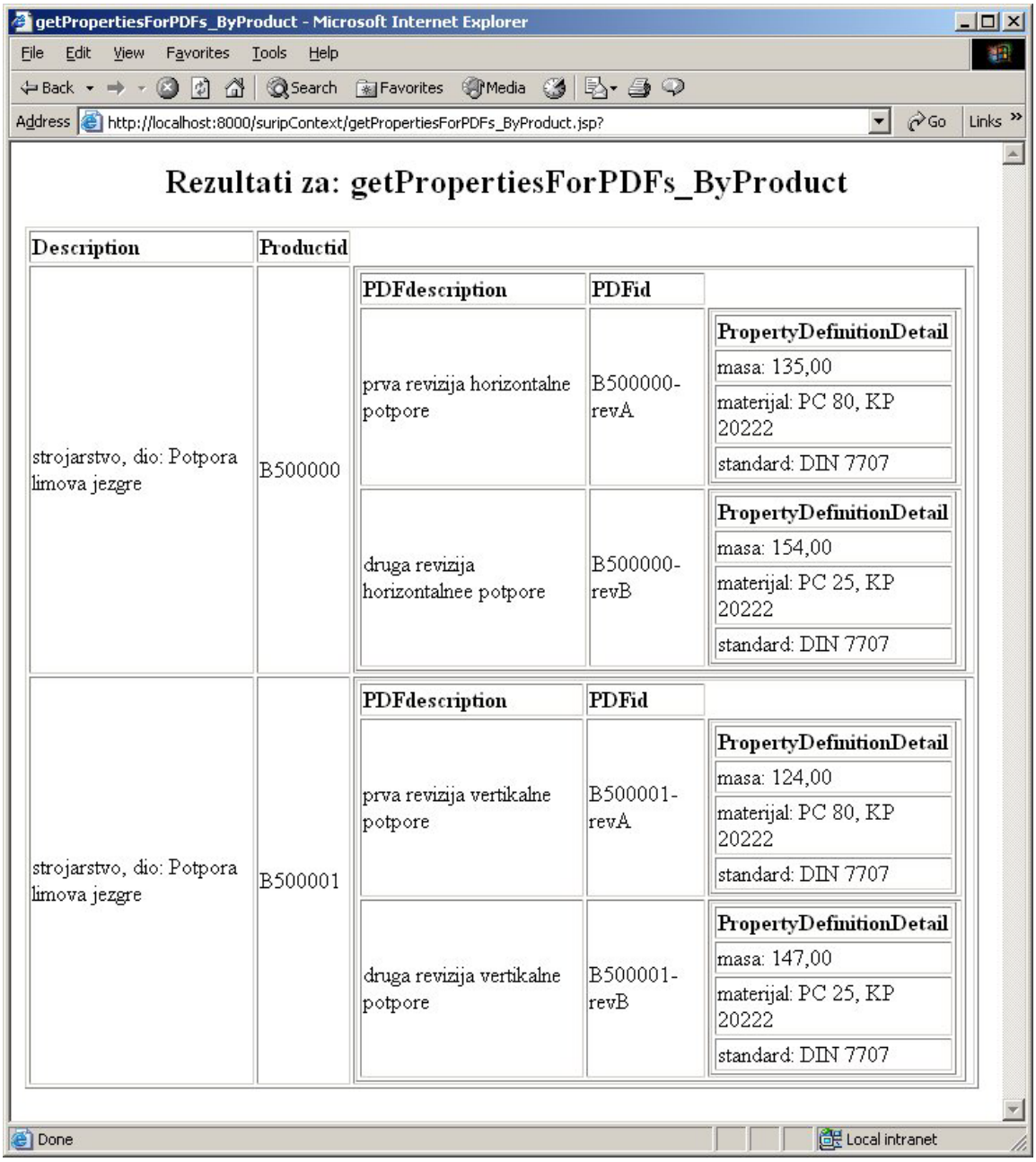
Kreiranje i testiranje XML operacija

Posljednja faza u realizaciji i testiranju web servisa korištenjem tehnologija dostupnih u J2EE© platformi, čini kreiranje XML operacije.



Slika 7.12: XML izlazni dokument sa podacima koji su rezultat obrade

Prema principu za kreiranje XML operacija opisanom u poglavlju 6.4.3, programski su realizirane izvršne klase XML operacija i web klijenti za potvrđivanje ispravnosti njihovog rada. Kao primjer ovdje će se spomenuti XML operacija čija uloga je vraćanje informacija o kreiranim fizičkim komponentama proizvoda, njihovim revizijama i svojstvima [Slika 7.11].



Slika 7.13: Prikaz rezultata transformiran u HTML oblik

Nakon pokretanja izvođenja XML operacije, prema principu opisanom u šestoj glavi, redom se pozivaju metode komponenti u poslovnoj razini servisa te se kreira XML izlazni dokument sa traženim podacima koji se vraćaju klijentu [Slika 7.12]. Obzirom da je klijent u ovom slučaju web preglednik, potrebno je i rezultate XML operacije, transformirati iz XML oblika u oblik pogodan za prikaz u HTML formatu [Slika 7.13], što je također implementirano u izvršnim klasama XML operacije.

Zbog ograničenja prostora u ovoj glavi je opisan samo mali dio programski realiziranih komponenti web servisa i klijenta koji su iskorišteni za njegovo testiranje, ali i to prema mišljenju autora predstavlja opseg dovoljan za potvrdu predložene koncepcije i arhitekture sustava. Podaci koji su korišteni u testiranju web servisa realni su proizašli iz procesa konstruiranja dijelova energetskog transformatora.

8

ZAKLJUČAK

Završna glava donosi diskusiju glavnih rezultata istraživanja prikazanog u radu. U glavi su također definirane smjernice za nastavak istraživanja.

8.1 Glavni rezultati rada

Prema postavljenom cilju istraživanja, određeno je da razvoj sustava za razmjenu i upravljanje informacijama o proizvodu zahtijeva: (i) definiranje informacijske infrastrukture za formaliziranje inženjerskog znanja i modeliranje informacija o inženjerskim proizvodima, (ii) kreiranje računalne platforme za stvaranje, spremanje, pristup i upravljanje tim informacijama u uvjetima globalne suradnje putem mrežnog okruženja.

U uvodnom dijelu rada analizirane su značajke proizvoda, uz promatranje procesa konstruiranja u kontekstu strukturiranja proizvoda. Razlog ovakvom pristupu leži u činjenici da je hijerarhijska struktura komponenti proizvoda, analizom teoretskih osnova, prepoznata kao poveznica nositelja informacija o proizvodu u procesu razvoja proizvoda. Ukratko su prikazana glavna teoretska područja: teorija tehničkih sustava, metodologije procesa konstruiranja, teorija domena, te teorije informacijskih modela i računalnih tehnologija koje su iskorištene u radu. U istraživanju je korištenjem prethodno navedenih teoretskih područja, realna problemska osnova modelirana preslikavanjem iz područja fenomenoloških i informacijskih modela podataka o proizvodu, u računalni model sustava. Ovo preslikavanje je rezultiralo predloženom implementacijom sustava za upravljanje i razmjenu informacija o proizvodu.

Kao osnova sustava koji je predmet istraživanja, definiran je informacijski model podataka o proizvodu, te on predstavlja jezgru oko koje je sustav izgrađen. Pregled teoretskog područja poslužio je za upoznavanje s nizom osnovnih koncepata za definiranje informacijskog modela podataka o proizvodu. Tim konceptima prikazan je osnovni skup potreba koje je bilo potrebno razmatrati prilikom definiranja informacijske infrastrukture sustava, te su oni opisani elementima 'metamodela' informacija o proizvodu. Kao glavni elementi metamodela izdvojeni su: nositelji informacija o proizvodu, subjekti koji upravljaju informacijama te aktivnosti koje koriste informacije.

U kontekstu nositelja informacija, analizirani su podaci vezani uz hijerarhijsku strukturu nositelja, verzije (uključujući revizije i varijante) nositelja i njihov status. Pri tome je bitno naglasiti da su kao glavni nositelji informacija u radu definirane fizičke komponente proizvoda te hijerarhijski strukturirani dokumenti. U kontekstu subjekata koji upravljaju informacijama analizirani su podaci vezani uz glavne korisnike informacija u razvojnoj fazi proizvoda, podaci vezani uz organizaciju radnih zadataka te podaci vezani uz definiranje uloga i prava korisnika unutar razvojnog tima. U kontekstu aktivnosti analizirani su podaci vezani uz: upravljanje hijerarhijskom strukturom proizvoda i strukturiranih dokumenata, upravljanje promjenama i kontrola podataka, osiguravanje konzistentnosti podataka, procedure odobravanja i autorizacije, dokumentiranje i upravljanje tokom procesa konstruiranja te podrška razvoju konfigurabilnih proizvoda. Na osnovu provedenih analiza i dosadašnjeg iskustva autora u području istraživanja, definirane su funkcije koje je bilo potrebno implementirati u sustav za upravljanje i razmjenu informacija o proizvodu.

Prethodno opisana raščlamba problematike razmjene i upravljanja informacijama o proizvodu učinjena je sa svrhom razumijevanja, interpretacije i pravilne implementacije informacijskog modela za upravljanje podacima o proizvodu koji je definiran ISO STEP 10303 standardom. Potreba za time je bila uvjetovana činjenicom da STEP standard pokriva vrlo široko područje razmjene podataka o proizvodu uključujući podatke iz različitih životnih faza proizvoda, te različitih područja, od brodogradnje do proizvodnje elektroničkih komponenti. Provedena analiza pomogla je u pravilnoj interpretaciji predloženog STEP informacijskog modela u konkretnom slučaju razmjene i upravljanja informacijama strojarskih proizvoda u njihovoj razvojnoj fazi.

Interpretacijom STEP PDM sheme u navedenom kontekstu, opisani su osnovni entiteti informacijske infrastrukture sustava te njihove relacije. Entiteti su grupirani kako slijedi:

- entiteti za identifikaciju i klasifikaciju komponenti proizvoda,
- entiteti koji definiraju svojstva komponenti,
- entiteti za definiranje hijerarhijske strukture komponenti i relacija među njima,

- entiteti za identifikaciju i klasifikaciju vanjskih referenci (datoteke i informacije pohranjene na tradicionalnim medijima),
- entiteti za definiranje veza vanjskih referenci s komponentama,
- entiteti potrebni za identifikaciju korisnika i dodjelu odgovornosti,
- entitet za opisivanje organizacije projekata i radnih grupa,
- entiteti za upravljanje tokom posla unutar razvojnog tima,
- entiteti za podršku konfigurabilnim proizvodima.

Drugi dio postavljenih ciljeva istraživanja bio je implementiranje sustava na osnovu prethodno definiranog informacijskog modela. Bitno je naglasiti da je sustav zamišljen za uporabu među korisnicima koji pristupaju i spremaju informacije putem mrežnog okruženja (Interneta/Intraneta), što je uvelike utjecalo na predloženu metodologiju implementacije. S tom svrhom posebna pozornost je posvećena XML tehnologiji koju čini grupa standardiziranih mehanizama za kreiranje i pretraživanje hijerarhijski strukturiranih podataka i dokumenta, te mehanizama za prikazivanje i kontrolu takvih podataka putem Interneta. Gledano iz perspektive ovog rada, odnosno razvoja sustava za razmjenu i upravljanje informacijama o proizvodu, zaključeno je da integracija STEP informacijskog modela i XML tehnologije omogućuje korištenje mrežne tehnologije za istovremeni rad različitih korisnika na istome projektu, razmjenu informacija, strukturiranje sadržaja tehničke dokumentacije kao komponente u hijerarhijskoj strukturi proizvoda te kreiranje distribuiranog virtualnog razvojnog okruženja za upravljanje informacijama o proizvodu.

Integracija informacijskog modela i XML ostvarena je implementacijom sustava kao web servisa. Web servis je u literaturi definiran kao kolekcija povezanih funkcionalnih komponenti kojima se pristupa putem Interneta, te su kao takve ponuđene za korištenje drugim aplikacijama. Autor ovog rada smatra da se kreiranjem web servisa za razmjenu i upravljanje informacijama o proizvodu na temelju informacijskog modela i zahtijevane funkcionalnosti, ostvarila računalna implementacija koja ispunjava zahtjeve primjene u heterogenim i distribuiranim inženjerskim razvojnim okruženjima. Time se također olakšalo povezivanje različitih aplikacija koje razvojni timovi već koriste u svojem radu, a koje generiraju i koriste podatke o proizvodu.

U tu svrhu, definirana je troslojna arhitektura web servisa, te su opisane značajke pojedinih razina arhitekture: razine trajnog zapisa podataka, razine poslovne logike web servisa, te XML operacije za pristup klijenata poslovnim komponentama servisa. Prilikom definiranja korištene su objektno orijentirane tehnike modeliranja računalnih sustava i programiranja. Za svaku od razina je korištenjem UML-a, modelirana struktura klasa sa pripadajućim atributima i metodama, te relacijama između klasa. Na kraju je opisan način kreiranja XML operacija koje omogućuju komunikaciju klijenata i servisa razmjenom XML dokumenta.

Web servis, odnosno njegove komponente su realizirane te je njihov rad testiran korištenjem J2EE© razvojne platforme što je omogućilo kreiranje platformski neutralnog, portabilnog višekorisničkog rješenja.

8.2 Smjerovi daljnjeg istraživanja

Nastavak istraživanja, autor rada vidi prije svega u integriranju razmjene informacija o proizvodu u sve faze životnog vijeka proizvoda. S tom svrhom, potrebno je u informacijsku infrastrukturu uključiti podatke iz ostalih faza životnog vijeka proizvoda. Isto tako potrebno je opisani informacijski model podataka o proizvodu proširiti za upotrebu u početnim fazama konstrukcijskog procesa. Posebno se to odnosi na opisivanje zahtjeva i funkcionalne strukture, odnosno definiranja veze između podataka koji opisuju proizvod gledan na različitim nivoima apstrakcije. Ono što nije podržano ovim modelom je također podrška procesu konstruiranja konfigurabilnih proizvodima, posebno izražena implementiranjem konfiguracijskih pravila kao dodatku sada opisanoj strukturi konfigurabilnih proizvoda. Tako definirana infrastruktura mogla bi poslužiti kao osnova za kreiranje XML rječnika za područje znanosti o konstruiranju po uzoru na slične rječnike već definirane za ostale znanstvene discipline.

Isto tako ostaju otvorena pitanja integracije između predloženog sustava te ostalih računalnih alata koje konstruktori koriste u svakodnevnom radu. Iako je predloženom implementacijom sustava omogućena razmjena informacija sa bilo kojom aplikacijom putem XML dokumenta, potrebno je pričekati ugradnju podrške za XML tehnologiju u specijalizirane alate poput CAE sustava, te tada vidjeti koji će se novi zahtjevi mogu postaviti na predloženi sustav. Ono što se može u ovom trenutku predvidjeti je svakako potreba za uključivanjem podatka vezanih uz geometriju komponenti proizvoda što nije razmatrano u okviru ovog rada.

9

LITERATURA

[1] NIST Special Publication 939: "STEP The Grand Experience"; Manufacturing Engineering Laboratory; ed. Sharon J. Kemmerer; 1999.

[2] Ropohol, G.: "Flexibile Fertigungssysteme"; Krausskopf Verlag; Mainz; 1971.

[3] Jorgensen, K. A.: "Paradigms for research work"; Department of Production; Alborg University; 1992.

[4] Mortensen, N. H.: "Design modelling in a Designer's Workbench"; Ph.D Thesis, Department of Control and Engineering Design; Technical University of Denmark; 1997.

[5] Hansen, C. T.: "Towards a Tool for Computer Supported Configuring of Machine Systems"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[6] McKay A.: "A framework for the characterization of product structure"; Proceedings of the 3rd WDK Workshop on Product Structuring; Delft University of Technology; ed. M. Tichem, T. Storm, M.M. Andreasen, A.H.B. Duffy; 1997.

[7] Andreasen, M. M.: "Conceptual Design Capture"; Proceedings of Engineering Design Conference '98; Brunel University; 1998.

[8] Zhang, Y., MacCallum, K.J., Duffy, A.: "Product knowledge modelling and management"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[9] Duffy, A., Andreasen. M.M.: "Enhancing the Evolution of Design Science"; Proceedings of ICED'95 Praha; Heurista Zurich; 1995.

[10] Hubka V.: "Theorie der Konstruktionsprozesse"; Springer-Verlag; Berlin; 1976.

[11] VDI 2221:"Methodik zum EntWickeln und Konstruiren technischer Systeme und Produkte"; Berlin; Beuth Verlag; 1993.

[12] Roth, K.: "Konstruiren mit Konstuktionskatalogen"; Springer; 1980.

[13] Yoshikava, H.: "General Design Theory as a Formal Theory of Design"; Intelligent CAD I; ed. Yoshikawa; Gossard; Procc. of IFIP TC5/WG5.2; North Holland; Boston; 1987.

[14] Pahl, G., Beitz, W.: "Engineering Design"; The Design Coucil; London; 1988.

[15] Ullman, D.G.:"The Mechanical Design Process"; McGraw-Hill; 1992.

[16] Ramakrishnan, R.: "Database Management Systems"; McGraw Hill, Singapore 1998.

[17] Hubka, V., Eder, W.E.: "Theory of Technical System"; Springer-Verlag; Berlin 1988.

[18] Hubka, V., Eder, W.E.:"A scinetific approach to engineering design"; Design Studies; Vol. 8, No. 3, pp. 123-137, 1987.

[19] Andreasen, M.M.:"Conceptual Design Capture"; Proceedings of EDC'98-Design Reuse; Brunel Universit; pp. 21-30; 1998.

[20] Andreasen, M.M, Hein, L.: "Integrated Product Development"; IFS Publications Ltd; Springer-Verlag; London; 1987.

[21] Andreasen, M.M.: "Machine Design Methods Based on Systematic Approach – Contribution to a Design Theory"; Department of Control and Engineering Design; Lund Institute of Technology, Sweden;1980.

[22] Ferreirinha, P. et al: "TEKLA, A language for Developing Knowledge Based Design Systems"; Proceedings of ICED 1990; Dubrovnik; Heurista; Zurich; 1990.

[23] Andreasen, M.M., Hansen, C.T., Mortensen, N.H.:"On the identification of product structure laws"; 3rd WDK Workshop on Product Structuring; Delft University of Technology; 1997.

[24] Schenk, D., Wilson, P.:"Information moddeling the EXPRESS Way"; Oxford University Press, New York; 1994.

[25] Bernstein, J. M., Wong, H.:"A language facilitz for designing database intensive aplications"; ACM Trans. Database Systems; Vol 5.; No 2.; pp. 185–207; 1980.

[26] Hammer, M., McLeod, D.:"Database description with SDM: a semantic database model"; ACM Trans. Databases; Vol 6.; No 3.; pp. 351–386; 1981.

[27] Su, W. Z. S.:"Modeling integrated manufacturing with SAM"; IEEE Computing; Vol 19.; No 1.; pp. 34–39; 1986.*

[28] Deux:"The Storz of O2"; IEEE Trans. Knowledge and Data Engineering; Vol 2.; No 1.; 1990.

[29] Tsichritizis, D.:"Hierarchical database management"; ACM Computer Surveys; Vol 8.; No 1.; pp. 105–124; 1976.

[30] Chen, P:"The entity-relationship model: towards a unified view of data"; ACM Trans. Database Systems; Vol 1.; No 1.; pp. 9-36; 1976.

[31] Halpin, A. T., Nijssen, M. G.:"Conceptual Schema and Relational Database Design"; Prentice-Hall; USA; 1989.

[32] "IDEF1X Manual USAF Intefrated Computer-Aided Manufacturing Program"; D'Appleton Company; USA; 1986.

[33] Eastman, M. C., Bond, H. A., Chase, C. S.:"A data model for design databases"; Artificial Intelligence in Design '91; Butterworth-Heineman Ltd; Oxford; 339–366; 1991.

[34] Eliëns A.,"Principles of Object-Oriented Software Development"; Reading; Addison –Wesley; 1995.

[35] http://java.sun.com/j2ee/sdk_1.3/index.html

[36] Pulm, U., Lindemann, U.: "Enhanced Systematics for functional Product Structuring"; Proceedings of International Conference on Engineering Design ICED 01; Glasgow; Scotland; 2001.

[37] Hansen, C.T.: "Towards a tool for computer supported structuring of products"; Proceedings of International Conference on Engineering Design ICED 97; Tampere; Finland; 1997.

[38] Tichem, M., Storm, T., Andreasen, M.M., MacCallum, K.J.:"Product structuring, an overview"; Proceedings of International Conference on Engineering Design ICED 97; Tampere; Finland; 1997.

[39] Erens F., Verhulst, K.:"Architecture for product families"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[40] Hansen C.T.:"An approach towards considering technical and economic aspect in product architecture design"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[41] Andreasen, M.M.:"The structuring of products and product programmes"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[42] Yu, B., MacCallum, K.:"Product structuring in reality"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[43] Tichem, M., Storm, T.:"Issues in Product Structuring"; Proceedings of the WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1995.

[44] Blessing L.T.M.:"Design process capture and support"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[45] Van den Hamer, P., Lepoeter, K.:"The Philips 5-dimensional framework for modelling design data and design processes"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[46] Murdoch, T.:"A layered framework for structuring produc data"; Proceedings of the WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1995.

[47] Peltonen, H.:"Concepts and an Implementation for Product Data Management"; Acta Polytechnica Scandinavica; Mathematics and Computer Series No. 105; Finish Academies of Technology, Espoo 2000.

[48] Vroom, R.:"Information generated and/or used during product and proces development"; Proceedings of the 2nd WDK Workshop on Product Structuring; Delft University of Technology; ed. M.Tichem, T. Storm, M.M. Andreasen, K.J. MacCallum; 1996.

[49] Pavlić D., Marjanović D., Štorga M.:"The Implementation of WEB-Based Technologies in Engineering Data Management"; Proceedings of the 6th International Design Conference DESIGN 2000/Dubrovnik; Hrvatska; CTT, FSB, WDK; pp. 333-338; 2000.

[50] Štorga M., Pavković N., Marjanović D.:"Computer Aided Product Structure Design"; Proceedings of the 6th International Design Conference DESIGN 2000/Dubrovnik; Hrvatska; CTT, FSB, WDK; pp. 359-364; 2000.

[51] Marjanović D., Bojčetić N., Deković D., Pavković N., Pavlić D., Štorga M., Žezelj D.: "Design Department Data Flow Integration"; Proceedings 3rd International Workshop IPD 2000/Magdeburg; Germany; Unversity of Magdeburg; pp. 135-150; 2000.

[52] Štorga M., Pavlić D., Marjanović D.: "Reducing Design Development Cycle by Data Management within Design Office"; Proceedings 13th International Conference on Engineering Design ICED 01/Glasgow; Scotland; IMechE, London UK; Vol 1; pp. 429-436; 2001.

[53] Pavković, N.:"Objektno orjentirani pristup modeliranju procesa konstruiranja";

doktorska disertacija; Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2000.

[54] Duhovnik, J., Tavčar, J.:"Elektronsko poslovanje in tehnični informacijski sistemi"; LECAD, Univerza v Ljubljani, Fakulteta za strojništvo; Slovenija; 2000.

[55] VDI 2221:"Methodik zum Entwickeln und Konstruiren technischer Systeme und Produkte"; Berlin; Beuth Verlag; 1993.

[56] Smith, B., Nagel, R., Wellington, J.:"IGES-Initial Graphics Exchange Specification"; Autofact; 1981.

[57] ISO 10303-1: "Industrial data systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles"; ISO; Geneva; 1994.

[58] ISO 10303: "Descriptive methods: architecture and development methodology reference manual"; ISO TC 184/SC4/WG10 N62; 1996

[59] ISO 10303-11: "Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: EXPRESS language reference manual"; 1994.

[60] PDM Implementator Forum: "Usage Guide for the STEP PDM Schema"; ProSTEP GmbH & PDES Inc.; 2000.

[61] Cagle, K.: "XML – Developer's Handbook"; SYBEX Inc.; 2000.

[62] Anderson, R.: "Professional XML"; Wrox Press Ltd.; Birmingham; UK; 2000.

[63] Karden, H.: "STEP and XML Interoperability"; PDMI2, 1999.

[64] Schreiber A.:"STEP and Internet Standards – Distributed Product Data Availability via XML"; PDMI2, 1999.

[65] Hemio, T.:"XML product model server"; VTT; Finland; 2000.

[66] Ehrler, A., Iselborn, B.:" A framework fro XML based Collaboration and Integration"; ProSTEP Science Days 2000.

[67] Wrightson, A.M.:"XML and STEP"; XML study notes; http://helios.hud.ac.uk /staff/scomaw/xmlstudy/xmlstep.htm; 1999.

[68] ISO/WD 10303-28:1999(E): "Product Data Representation and Exchange – Part 28: Implementation Methods: XML representation of EXPRESS-driven data"; ISO TC 184/SC4/WG11 N101; Geneva; 1999.

[69] Booch, J., Rumbaugh, J., Jacobson, I.:"The Unified Modeling Language User Guide"; Addison-Wesley; USA; 1999.

[70] Alhir, S.S.:"UML in a nutshell"; O'Reilly; USA; ed. Oram, A.; 1998.

[71] Tracy, K.W., Bouthoorn, W.F.:"Object-Oriented Artificial Intelligence Using

C++"; Computer Science Press; New York; 1996.

[72] Glass, G.: "The Web Services (R)evolution – Applaying Web Services To Applications"; http://www-106.ibm.com/developerworks/webservices/library/-ws-peer1.html?dwzone=ws; 2000.

[73] "Building Web Services"; Sun Microsystems, Inc.; USA; 2001.

[74] Vawter, C., Roman, E.:"J2EE vs. Microsoft.NET – a comparison of building XML-based web services"; The Middleware Company; http://www.middleware-company.com/; 2001.

[75] Bodoff, S., Green, D., Jendrock, E., Pawlan, M., Stearns, B.:"The J2EE™ Tutorial"; Sun Microsystems, Inc.; USA; 2001.

[76] "Building Web Components"; Sun Microsystems, Inc.; USA; 2001.

[77] "Programming Persistence"; Sun Microsystems, Inc.; USA; 2001.

[78] "Building Enterprise JavaBeans Components"; Sun Microsystems, Inc.; USA; 2001.

[79] "Building JSP™ Pages That Use XML Data Services"; Sun Microsystems, Inc.; USA; 2001.

[80] Roman, E.:"Mastering Enteprise JavaBeans™ and the Java™2 Platform, Enterprise Edition"; Wiley Computer Publishing; ed. Elliott R.; USA; 1999.

[81] "Forte™ for Java™, Enterprise Edition Tutorial"; Sun Microsystems, Inc.; USA; 2001.

KRATKI ŽIVOTOPIS

Mario Štorga rođen je 24.02.1974. godine u Varaždinu gdje je završio srednju strojarsku školu. Fakultet strojarstva i brodogradnje u Zagrebu je upisao 1992. godine. Diplomirao je 1997. godine na usmjerenju "Strojarske konstrukcije".

Od 1997. godine zaposlen je pri Katedri za osnove konstruiranja Fakulteta strojarstva i brodogradnje u Zagrebu u svojstvu znanstvenog novaka pri projektu 120-015 "Model inteligentnog CAD sustava". Tijekom rada na fakultetu aktivno sudjeluje u nastavi iz sljedećih kolegija: Uvod u računala, Primjena računala-K, Konstruiranje pomoću računala, Osnove konstruiranja. Kao honorarni asistent sudjeluje u izvođenju vježbi pri Studiju Dizajna (Arhitektonski fakultet u Zagrebu) te strojarskom i informatičkom odjelu Tehničkog veleučilišta u Zagrebu. Osim u nastavi, aktivno je sudjelovao u organizaciji međunarodnih znanstvenih skupova DESIGN '98, DESIGN 2000 i DESIGN 2002 koji se održavaju u Dubrovniku.

Tijekom ljeta 2000. godine sudjelovao je na međunarodnom seminaru o metodologiji procesa konstruiranja u organizaciji Danskog tehničkog sveučilišta. Kao autor ili koautor objavio je 9 znanstvenih i 8 stručnih radova u Hrvatskoj i inozemstvu. Član je Hrvatskog društva za elemente strojeva i konstrukcije te međunarodne udruge *The Design Society*. Govori i piše engleskim jezikom, a pasivno se služi njemačkim.

SHORT BIOGRAPHY

Mario Štorga born on February 24th 1974 in Varaždin where he completed his secondary school education (technical high school). In 1997, he graduated in mechanical design from the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb.

Since 1997 he has been a science novice at the Design Department of the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb, at the research project No. 120-015 "Model of intelligent CAD system". Besides teaching activities he took part in the organization of the international design conferences DESIGN ’98, DESIGN 2000 and DESIGN 2002 held in Dubrovnik.

Mario Štorga has participated and passed the “Ph.D. Course - Design Methodology” during the summer 2000, organized by Technical University of Denmark. As the author or coauthor he has published 9 scientific papers and 8 technical reports in Croatia and abroad. He is member of the Croatian society for machine elements and design as well as an international association "The Design Society". He has a good command of English and can read German well.