Računalni model konstrukcijskog znanja

Bojčetić, Nenad

Doctoral thesis / Disertacija

2001

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:353183

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-31**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

RAČUNALNI MODEL KONSTRUKCIJSKOG ZNANJA

DOKTORSKA DISERTACIJA

Mentor:
Prof. dr. sc. Dorian Marjanović, dipl. inž

Mr. sc. Nenad Bojčetić, dipl. inž.

.

Zagreb, 2001.

PODACI ZA BIBLIOGRAFSKU KARTICU

UDK:	658.512.2:681.3:621
Ključne riječi:	teorija konstruiranja, konstruiranje pomoću računala, objektno orijentirano modeliranje, objektno orijentirano programiranje, konstrukcijsko znanje, STEP, modeliranje uporabom značajki
Znanstveno područje:	Tehničke znanosti
Znanstveno polje:	Strojarstvo
Institucija u kojoj je rad izrađen:	Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje
Mentor:	Prof. dr. sc. Dorian Marjanović
Broj stranica:	109
Broj slika:	73
Broj tablica:	1
Broj korištenih bibliografskih jedinica:	142
Datum obrane:	
Povjerenstvo:	Prof. dr. sc. Izvor Grubišić Prof. dr. sc. Anton Jezernik Prof. dr. sc. Dorian Marjanović
Institucije na kojima je rad pohranjen:	Fakultet strojarstva i brodogradnje Zagreb Nacionalna i sveučilišna knjižnica Zagreb

Zahvaljujem mentoru, prof. dr. sc. Dorianu Marjanoviću na savjetima, korisnim raspravama i podršci u tijeku izrade ovog rada.

Primjedbe i savjeti članova povjerenstva također su puno pomogli pri izradi i povećanju kvalitete rada.

Djelatnicima Laboratorija za osnove konstruiranja i djelatnicima Katedre za konstruiranje te dipl. ing. Draganu Žeželju zahvaljujem na doprinosu ovom radu kroz brojne analize i rasprave tijekom zajedničkih istraživanja na projektu kojega je dio i ova disertacija.

Ministarstvo znanosti i tehnologije Republike Hrvatske također je potpomoglo izradu ovog rada financiranjem projekta 120-015 "Razvoj modela ICAD sustava".

Predgovor

Izrada ovog rada potaknuta je višegodišnjim istraživačkim radom u području unapređenja računalne podrške procesu konstruiranja i modeliranja strojarskih proizvoda, te iskustvom autora u primjeni i razvoju dijelova CAD sustava.

Postojećim CAD sustavima nedostaje podrška upravljanju i manipulaciji konstrukcijskim znanjem tijekom procesa konstruiranja. Također nedostaje veza između geometrijskog prikaza proizvoda i konstrukcijskog znanja uporabljenog za njegovo konstruiranje. Osnovni cilj provedenog istraživanja je objektno orijentiranim metodama modelirati konstrukcijsko znanje o proizvodu te ga spregnuti sa CAD modelom. Uporaba takovog modela, u sustavima za pomoć konstruktoru u procesu konstruiranja, pridonijela bi povećanju fleksibilnosti i efikasnosti primjene CAD sustava te povećanju produktivnosti procesa konstruiranja.

Istraživačkim projektom broj 120-015 "Model inteligentnog CAE sustava" predviđeno je istraživanje modeliranja konstrukcijskog znanja tijekom procesa konstruiranja te je ova disertacija dio cjelokupnog istraživanja unutar navedenog projekta. Očekuje se da realizacija proširenog modela proizvoda posluži kao osnova za kreiranje naprednijih alata za pomoć konstruktoru u procesu konstruiranja.

Sažetak

Tema ove disertacije je razvoj strukture proširenog CAD modela. Prošireni CAD model čini sprega konstrukcijskog znanja i parametarskog CAD modela. Kao osnova za kreiranje strukture konstrukcijskog znanja uporabljena je semantika STEP standarda. Realizacija strukture konstrukcijskog znanja ostvarena je uporabom objektno orijentiranog pristupa.

Parametarski CAD model kreiran je tehnikom modeliranja pomoću značajki. Prošireni CAD model, prilikom uporabe, manifestira *kvazi* inteligentno ponašanje realizirano pomoću "rule based" metode umjetne inteligencije. Dijelovi proširenog CAD modela spregnuti pravilima zaključivanja tvore jedinstvenu cjelinu i na taj se način ponašaju kao objekt u procesu konstruiranja proizvoda.

Glavni rezultat istraživanja u ovom radu je prijedlog modela konstrukcijskog znanja. Predloženi model konstrukcijskog znanja podijeljen je u logičke grupe:

- elementi – značajke i dijelovi koji tvore pojedine verzije proizvoda,
- pravila – po kojima se aktiviraju pojedina znanja ovisno o odabranoj verziji,
- dokumenti – digitalni ili fizički dokumenti koji tvore ili su vezani za određena znanja o proizvodu,
- aplikacije – vanjski računalni programi koji se aktiviraju ovisno o verziji proizvoda,
- postavke – predložene vrijednosti i dozvoljene granice u kojima se moraju nalaziti vrijednosti parametara te sustavi jedinica i postavke svojstava materijala,
- ograničenja – definiranje dijelova koji moraju postojati u sklopu, ukoliko se želi uporabiti odabrana verzija proizvoda, te geometrijska ograničenja prilikom ubacivanja CAD modela u sklop,
- relacije – načini postavljanja vrijednosti parametara geometrijskog modela proizvoda.

Navedeni dijelovi konstrukcijskog znanja preslikani su u klase objektno orijentiranog računalnog modela konstrukcijskog znanja. Predloženi model je implementiran u CAD sustav Pro/ENGINEER® uporabom JAVA® programskog jezika i J-Link biblioteka.

Ključne riječi:

teorija konstruiranja, konstruiranje pomoću računala, objektno orijentirano modeliranje, objektno orijentirano programiranje, konstrukcijsko znanje, STEP, modeliranje uporabom značajki

UDK 658.512.2:681.3:621

COMPUTER MODEL OF DESIGN KNOWLEDGE

SUMMARY

The subject of this thesis is the development of the extended CAD model structure. Extended CAD model consists of the design knowledge and parametric CAD model coupled together. As the basis for the development of the design knowledge structure STEP standard was used. Realization of the design model structure is achieved through usage of the object oriented programming approach.

Parametric CAD model is created in feature based parametric CAD modeler. Extended CAD model exhibits *kvazi* intelligent behavior realized through usage of the rule based inference engine. Parts of the extended CAD model coupled with inference rules make the unique whole and in this way behaves as an object in the design process.

The main *result* of this research is the proposition of the design knowledge model. The proposed model of the design knowledge is divided into logical groups as follows:

- elements – features and parts that constitute the product version,
- rules – set of definitions that describe the product version,
- documents – digital and physical documents that makes or are connected to certain design knowledge,
- applications – external programs related to the product version,
- settings – definition of allowable limits or range of values for parameters, definition of units and material properties,
- constraints – definition of required existing parts in assembly and geometric constraints for part assembling,
- relations – definition of the initialization of the model parameters.

The elements of the design knowledge model are mapped to the classes of the object oriented model. The proposed model is implemented using the Pro/ENGINEER, JAVA programming language and the J-Link libraries.

Keywords:

engineering design, design theory, computer-based design, object-oriented technology, feature-based modeling, STEP, design knowledge

UDC 658.512.2:681.3:621

Sadržaj:

Popis slika:

Popis tablica:

1 Uvod

Računalnu sustavi za podršku procesu konstruiranja jedan su od najznačajnijih alata za pomoć inženjeru pri radu. Činjenica je da se od svih raspoloživih sustava najviše koriste CAD[1] sustavi tj. sustavi za kreiranje računalnih modela proizvoda i/ili tehničke dokumentacije. U radu s većinom CAD sustava inženjer konstruktor ograničen je samo na manipulaciju geometrijskim prikazom proizvoda, dodajući osnovne (linije, kružnice, točke, ...) ili složene geometrijske elemente (površine, kruta tijela, značajke, ...) uz ograničenu mogućnost pretraživanja baza podataka ili provođenja proračuna.

Naglasak pri uporabi CAD sustava, a i pri razvoju istih, je većinom na rješavanju mogućnosti unapređenja uporabe i manipulacije geometrijskim prikazom proizvoda. U većini inženjerskih područja, pa tako i u domeni strojarskih konstrukcija, geometrijski prikaz tj. računalni prikaz fizičkog oblika proizvoda nije jedini oblik informacija koje su potrebne inženjeru. Konstruktor mora voditi računa ne samo o geometrijskim podacima već i o osobinama uporabljenih materijala, mogućnostima dostupne tehnologije, ograničenjima pri transportu, uvjetima eksploatacije, sklopivosti, itd.

U današnje vrijeme svjedoci smo sve većeg povećanja složenosti proizvoda kao i sve bržeg razvoja novih tehnologija [1]. Rezultat toga je povećanje potrebnog znanja konstruktora, skoro do razine kada to prelazi mogućnosti jednog čovjeka. Da bi prevladale ove probleme tvrtke moraju razmišljati o:

- unapređenju uporabe unutrašnjeg[2] i vanjskog[3] konstrukcijskog znanja,
- maksimalnoj ponovnoj uporabi postojećih komponenti i postojećih konstrukcijskih rješenja,
- unapređenju timskog rada,
- uporabi metoda umjetne inteligencije.

Navedeno još više ističe potrebu za uporabom računala i programa za računalnu podršku u svim područjima rada inženjera. U posljednjih desetak godina u istraživanjima se konstruiranje

[1] Computer Aided Design
[2] Pod pojmom unutarnje znanje smatra se znanje unutar tvrtke ili konstrukcijskog ureda tj. znanje koje je razvijeno ili usvojeno unutar tvrtke (preporuke, standardi, upute)
[3] Pod pojmom vanjsko znanje smatra se znanje koje nije vezano za tvrtku, naprimjer znanje iz nekog područja, međunarodni standardi, znanja drugih tvrtki

tretira kao proces obrade i generiranja informacija o proizvodu [1], [2], [3], [4]. Uvriježeni način rada i pristup konstruiranju uvelike se promijenio pod imperativom povećanja efikasnosti, razine kooperativnosti, fleksibilnosti i skraćenja vremena potrebnog za izradu proizvoda.

Da bi konstruktori zadovoljili zahtjeve tijekom konstrukcijskog procesa obrađuju značajan dio informacija o proizvodu. Količina, vrsta, značaj i priroda informacija koje tvore konstrukcijsko znanje ovisi o različitim kontekstima, među kojima su i faze razvoja proizvoda pa tako i faza procesa konstruiranja. Izvori i struktura informacija koje konstruktori obrađuju različiti su. Konstruktor mora imati pristup informacijama i znanju u pravo vrijeme, tj. kada konstruira.

1.1 Opis zadatka (definicija problema)

Ovisno o konstrukcijskom zadatku, konstrukcijski proces uključuje aktivnosti više konstruktora i programskih sustava koji su smješteni u različitim dijelovima tvrtke ili u drugim tvrtkama. U proces konstruiranja su uključene i različite programske aplikacije, što rezultira složenim informatičkim okruženjem. Prilikom uporabe različitih programskih aplikacija pojavljuje se problem razmjene podataka među njima i problem interpretacije dobivenih rezultata. Iz navedenog proizlazi zahtjev za što većom standardizacijom oblika informacija i zapisa znanja, kako pri konstruiranju, odnosno opisu proizvoda, tako i pri razmjeni između različitih programskih aplikacija, korisnika ili tvrtki.

Trendovi globalne ekonomije uvjetuju komunikaciju između tvrtki, u smislu kooperacije pri razvoju proizvoda. Pored razmjene geometrijskih informacija nužna je i razmjena drugih informacija o proizvodu tj. razmjena skupa znanja o proizvodu. U tom se slučaju izrazito naglašava kompatibilnost računalnih sustava i uporabljenih aplikacija u raznim tvrtkama. Rješenje kompatibilnosti klasifikacije i zapisa podataka je moguće provesti pomoću specifičnih rješenja ukoliko se radi o malom broju tvrtki, no problem raste kada se radi o većem broju tvrtki. U tom se slučaju postavlja pitanje o standardima zapisa informacija o proizvodu i njihovoj razmjeni.

Današnji standardi (IGES[4], SET[5], VDAFS[6]), naročito u CAD domeni, ne pružaju mogućnost razmjene informacija koje nisu geometrijske ili topološke tj. u najboljem slučaju može se razmijeniti ograničena količina i tip informacija o tehnologiji ili materijalu. Dakako da postoji mogućnost integracije ostalih znanja u postojeće standarde, što bi dovelo do proširenja uporabe standarda i otvorilo mogućnosti kvalitetnije komunikacije između različitih računalnih sustava. Vodeći se potrebama zapisivanja i razmjene znanja Kneebone [5] opisuje mogućnost integracije inženjerskog znanja u postojeće standarde za razmjenu podataka o proizvodu.

Prijenos svih informacija o proizvodu drugoj tvrtci nije u skladu sa razvojnom ili proizvodnom politikom pojedinih, ako ne i svih tvrtki. Dostupnost informacijama o proizvodu od strane drugih tvrtki implicira rješavanje problem prava pristupa i uporabe pojedinih znanja i informacija o proizvodu. Idealno bi bilo da druge tvrtke ili kooperanti mogu pristupiti dijelovima znanja tj. informacijama o proizvodu koji se tiču samo njih, a da su sve informacije i dalje pohranjene kod matične tvrtke. Ovaj zahtjev uključuje provjeru korisnika te definiranje razina i prava pristupa pojedinim znanjima. Salminen i grupa autora [6] pristupa ovom problemu putem

[4] Interim Graphics Exchange Specification
[5] Standard D'Exchange et de Transfer
[6] Verband der Automobilindustrie-Flachen-Schnittstelle

osnivanja globalne inženjerske mreže nazvane GEN koja bi omogućila razmjenu informacija među konstruktorima i time omogućila smanjenje vremena konstruiranja proizvoda, povećala efikasnost i omogućila pristup znanjima eksperta. Ključni dio GEN projekta je projekt pod imenom GENIAL [7]. Gui opisuje model za definiranje semantike modela proizvoda (GVE[7]) kao osnovu za razmjenu informacije o proizvodu u globalnom radnom okruženju [8].

Količina informacija koje konstruktor prilikom konstruiranja mora razmijeniti s računalom prelazi njegove mogućnosti. U isto vrijeme povećava se broj konstrukcijskih i proizvodnih procesa koji se oslanjaju na računalnu podršku. Budući da većina novih konstruktora nema znanja i iskustva potrebnih za trenutačno uključenje u konstrukcijski proces, neophodno je utrošiti dio vremena na njihovo obrazovanje [9]. To vrijeme mora biti što je moguće kraće jer se troši ne samo vrijeme konstrukcijskog procesa, već i vrijeme iskusnog konstruktora koji obrazuje novog i time ne može potpuno učestvovati u konstrukcijskom procesu. Uz navedeni problem često se javlja i problem uporabe "starih" konstrukcijskih rješenja (naročito je izraženo pri ponovljenoj i varijantnoj konstrukciji), za koje se često ne zna kako su nastala te koja su znanja, standardi, proračuni i dokumenti uporabljeni.

Prethodni navodi naglašavaju potrebu za upravljanjem informatičkim okružjem, u svrhu unapređenja efikasnosti i pouzdanosti konstrukcijskog procesa. Iako bi ekspertni sustavi mogli riješiti dio navedenih problema, problem je što je većina ekspertnih sustava usko specijalizirana [10], nefleksibilna i prezahtjevna za održavanje. Rješenje ovog problema je moguće kreiranjem modela proizvoda, koji se sastoji od geometrijskog prikaza i znanja potrebnog za njegovu konstrukciju [11], [12].

Istraživanja u ovom radu usmjeriti će se na mogućnost računalnog modeliranja konstrukcijskog znanja. Konstrukcijsko znanje će se modelirati kao objekt[8] procesa konstruiranja proizvoda, zapisat će se semantikom STEP standarda u relacijsku bazu podataka, te će se pravilima zaključivanja spregnuti sa grafičkim CAD prikazom. Na taj će način postojeći CAD model manifestirati *kvazi* inteligentno ponašanje.

1.2 Cilj istraživanja i hipoteza

Osnovni cilj ovog rada je objektno orijentiranim metodama modelirati konstrukcijsko znanje o proizvodu te ga spregnuti sa CAD modelom.

Hipoteza je rada da prošireni CAD model, kao sprega zapisa konstrukcijskog znanja i CAD modela, manifestira *kvazi* inteligentno ponašanje.

Prošireni CAD model sastoji se od dva međusobno spregnuta dijela: parametarskog CAD modela i objektno modeliranog konstrukcijskog znanja. Navedeni dijelovi spregnuti pravilima zaključivanja tvore jedinstvenu cjelinu i ponašaju se kao objekt u procesu konstruiranja proizvoda.

Pretpostavlja se da se značajan dio znanja potrebnog pri konstruiranju proizvoda može prikazati kao objekt u procesu konstruiranja i pridružiti CAD modelu. Atributi objekata znanja zapisivat će se u relacijsku bazu. Kao osnova za kreiranje strukture zapisa informacija o znanju i geometrijskih informacija o proizvodu koristit će se semantika STEP standarda. Sprega između

[7] Generic Virtual Enterprise

[8] Problematika objekata i objektnog pristupa može se sagledati sa stanovišta programiranja, ali i sa stanovišta modeliranja. Tako da se objekti mogu okarakterizirati i opisati na različite načine. Pristup koje će se koristiti u radu bazira se na dvije definicije objekta: objekt posjeduje identitet, stanje i ponašanje [13] , objekt je element modularnosti strukture i ponašanja koji posjeduje osobine [14].

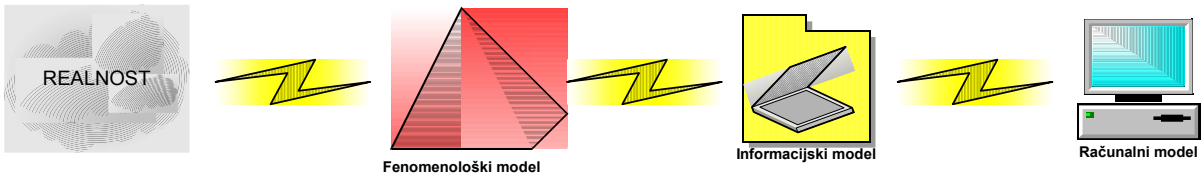
korisnika, CAD modela i baze znanja ostvarit će se mehanizmom poruka. Predloženi model omogućit će vezivanje znanja o konstrukciji za računalni CAD prikaz te razmjenu i ponovnu uporabu pohranjenog znanja. Primjena predloženog proširenog CAD modela odnosi se pretežno na ponovljene i varijante konstrukcije.

1.3 Metodologija istraživanja

Ova disertacija dio je ukupnih istraživanja na projektu razvoja modela CAD sustava čije se karakteristike trebaju približiti ideji inteligentnog CAD sustava. Rad se jednim dijelom oslanja na istraživanja prikazana u [15], [16], [17], [18] i [19]. U nedostatku prikladnijeg termina za karakterizaciju CAD sustava, koji bi sadržavao okruženje za kreiranje i uporabu konstrukcijskog znanja, koristit će se pridjev inteligentni ili prigodniji izraz *kvazi* inteligentni sustav. U literaturi se pridjev *inteligentan* često koristi [20], [21] ali inteligentni sustav prije je cilj kojem treba težiti nego praktično ostvariva mogućnost.

Razlozi za kreiranje modela su različiti. Ukoliko se radi o većim sustavima, kreiramo modele zbog nemogućnosti potpunog poimanja realnog sustava ili pomoću modela pokušavamo bolje razumjeti sustav koji razvijamo. Model se može koristiti i kao sredstvo za spremanje, procesiranje i razmjenu znanja [22], [23]. No, koji god razlozi nas motivirali za kreiranje modela moramo se voditi idejom da, iako su modeli zapravo pojednostavljenje stvarnog, kvalitetni modeli moraju biti usko vezani sa realnim sustavom koji modeliramo. Prema [23] izjave o nečemu (što se modelira) uvijek se odnose na određenu svrhu (zašto se model kreira), prikazuju original sa različitih aspekata, potječu od nekoga (tko kreira model), usmjerene su nekome, i dio su specifičnog konteksta.

Sa stanovišta modeliranja, metodologija istraživanja može se prikazati kao niz preslikavanja prema slici 1-1 [24]. Realnost se modelira teorijama znanosti o konstruiranju, temeljem kojih se preslikava u fenomenološke modele. Informacijski modeli temelje se na informacijskim teorijama npr. "entity-relationship" modeliranju ili objektno orijentiranom modeliranju. Računalni model temelji se na programskom jeziku, odnosno odabranom programskom okruženju realizacije. Preslikavanja s lijeva u desno prikazuju proces istraživanja u kojem se fenomenološki model postepeno formalizira sredstvima informacijskih i računalnih modela. Putanja s desna u lijevo označava proces verifikacije u kojem se model uspoređuje s realnosti.



Slika 1-1 Preslikavanja od realnosti do računalnog modela

U uvodnom dijelu rada analizirat će se značajke proizvoda u procesu konstruiranja i sagledat će se sadašnje stanje istraživanja u teoretskom kao i u računalnom modeliranju proizvoda, posebice modeliranje proizvoda uporabom značajki, i znanja o proizvodu u procesu konstruiranja. Model konstrukcijskog znanja koncipirat će se na temelju metodičke analize i klasifikacije konstrukcijskog znanje. Nakon čega će se analizirat mogućnost implementacije predloženog modela objektnim metodama. Kao uvod u predloženi pristup ukratko će se

razmotriti osnovni principi i dostupne spoznaje objektno orijentirane tehnike programiranja i objektno orijentiranog pristupa modeliranju.

Slijedit će opis STEP informacijskog modela i načini prijenosa podataka između različitih CAD sustava. U okviru ovog dijela rada predložit će se struktura zapisa modela konstrukcijskog znanja u skladu sa semantikom STEP-a.

Razvit će se programske procedure za kreiranje objekata konstrukcijskog znanja i inicijalizaciju atributa objekata na osnovu zapisa u relacijskoj bazi.

U slijedećem dijelu rada predložit će se i opisati struktura sustava za kreiranje, promjenu i uporabu konstrukcijskog znanja pri modeliranju proizvoda. Analizom mogućnosti dostupnih CAD sustava, i na osnovu predloženog koncepta zapisa znanja, koncipirat će se sustav za povezivanje znanja s CAD modelom te metode za uporabu znanja kroz mehanizme CAD sustava. Pri koncipiranju sustava za manipulaciju konstrukcijskim znanjem o proizvodu, koji će se temeljiti na predloženoj strukturi zapisa konstrukcijskog znanja, uzet će se u obzir mogućnosti mrežne tehnologije.

Predloženi model testirat će se na odabranom primjeru realne konstrukcije. Na kraju će se analizirati primjenjivost i mogućnosti razvijenog modela, i predložiti smjernice daljnjeg razvoja.

2 Pregled stanja istraživanja i teoretske osnove

Prilikom konstruiranja inženjer se koristi iskustvenim znanjem, formalnim znanjem stečenim obrazovanjem te intuitivnim znanjem i talentom. Znanje potrebno za generiranje i prosuđivanje konstrukcijskog rješenja ovisno je o domeni konstrukcijskog zadatka, dok je znanje o strukturiranju procesa konstruiranja većinom neovisno o domeni zadatka.

Konstrukcijski zadatak može se riješiti na više različitih načina, odnosno može rezultirati s različitim strojnim sustavima ili sklopovima. Ta karakteristična mnogostrukost mogućih rješenja uvjetovana je količinom svojstava proizvoda koja se trebaju odrediti u postupku konstruiranja. Proces konstruiranja može se razložiti na manje cjeline (faze, dijelove procesa, etape, operacije) koji čine strukturu procesa.

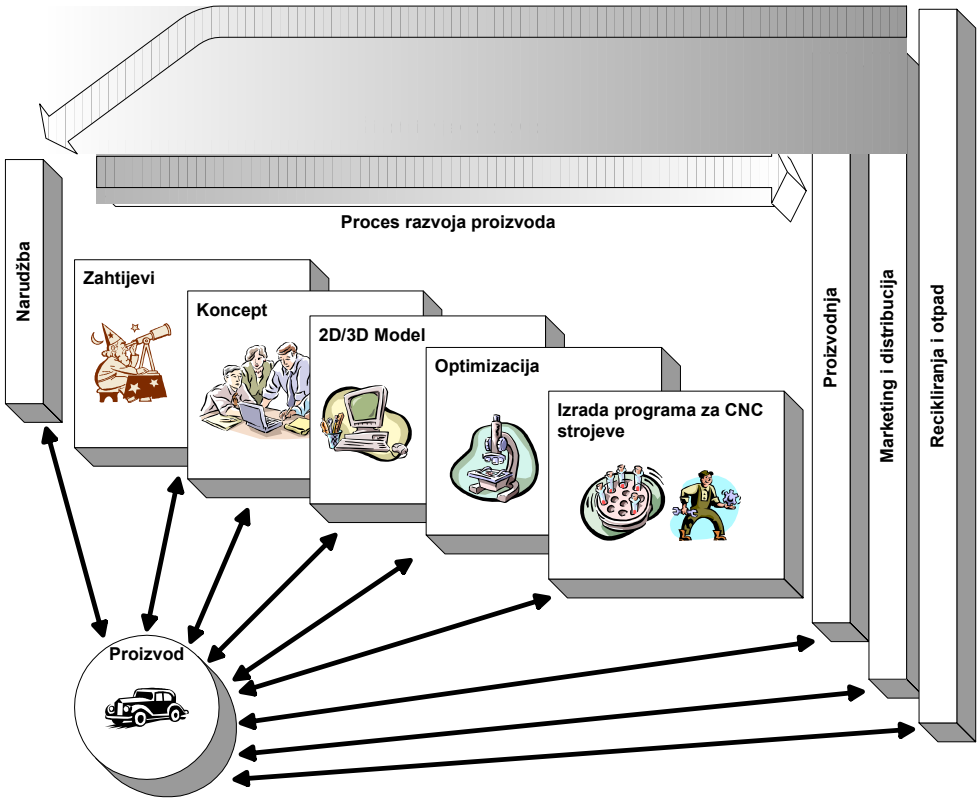
Kompleksnost, često međusobno proturječnih zahtjeva, rezultira najčešće višestrukim ponavljanjem određenih faza konstruiranja nakon raščišćavanja liste zahtjeva i postavljanja pretpostavki. Iterativnost je stoga jedan od značajnih karakteristika konstruiranja.

Određeni misaoni postupci (intuicija, nastajanje ideje) koji se ne mogu racionalno objasniti imaju obilježja umjetničke kreativnosti. Ti postupci se ne mogu formalizirati u svrhu kreiranja cjelovitog teoretskog modela prikaza konstrukcijskog procesa. Sa stanovišta teorije proizvoda mogu se navesti faktori koje utiču na proces konstruiranja prema [25]:

- nagli porast potreba uvjetovan tržišnim zakonitostima,
- period razvoja proizvoda je sve kraći,
- vijek trajanja proizvoda je sve kraći,
- javlja se pojam kritične brzine konstruiranja,
- raste količina proizvoda u serijskoj i masovnoj proizvodnji,
- zahtjevi za kvalitetom također rastu,
- troškovi se moraju svoditi na minimum,
- najveći utjecaj na strukturu troškova ima konstrukcijsko rješenje proizvoda,
- produktivnost tehnologije raste daleko brže od produktivnosti konstruiranja.

Iz navedenog može se uočiti mnogostrukost upliva na proces konstruiranja, kompleksnost procesa, a i širina potrebnih znanja. Naprimjer, razmatranja utjecaja okoliša na proizvodni sustav s implikacijama na CIM[9], odnosno na proces konstruiranja i CAD mogu se naći u [26].

Rezultati istraživanja, kao i iskustva iz industrije naglašavaju potrebu integracije informacijskih tokova u procesu izrade proizvoda, čiji je dio proces konstruiranja. Poduzeća koja su krenula tim smjerom, znatno su unaprijedila produktivnost i smanjila troškove [27]. Ako promatramo informacijske tokove cijelog procesa proizvodnje (Slika 2-1), sigurno je da je proces konstruiranja najkompleksniji njegov dio. U procesu konstruiranja obrađuje se i stvara najveći dio informacija o proizvodu. Bitno je napomenuti da proces konstruiranja ima najveći utjecaj na troškove proizvodnje [25], [26].



Slika 2-1 Proces izrade proizvoda

Postizanje skraćenja vremena konstruiranja uz zadržavanje kvalitete proizvoda poticaj su velikog broja istraživanja procesa konstruiranja, pogotovo metoda i modela računalom podržanog konstruiranja. Jedan dio istraživanja u području CAD-a, bavi se integracijom CAD alata i implementacijom CAD alata u sve faze procesa konstruiranja. Razvijen je značajan broj različitih pristupa i pravaca istraživanja. Za ilustraciju sadašnjih trendova istraživanja dan je kratak pregled značajnijih radova, uz grubu klasifikaciju.

Teoretski okvir[10] za inteligentne CAD sustave možemo pronaći u radovima Akmana, Hagena i Tomiyame [28] koji su predložili "jezik za opis konstruiranja". Jezik za opis konstruiranja sagrađen je na principima logičkog i objektno orijentiranog programiranja. Jezik se

[9] Computer Integrated Manufacturing
[10] frameworks

temelji na općoj teoriji konstruiranja Yoshikawe i Tomiyame. Proučavajući različite metode i znanje te klasificirajući podzadatke neovisno o pridruženom znanju Chandrasekaran [29] je razvio strukturu zadataka u procesu rješavanja konstrukcijskih problema.

Problemu konstruiranja uobičajenih dijelova i kataloškog izbora, Colton i Dascanio [30] pristupili su koncipiranjem i implementiranjem integrirane radne okoline. U većini istraživačkih projekata posebna pozornost usmjerava se na razvoj "prijateljskih" korisničkih sučelja, integriranim s modulima za provjeru informacija. Istraživanja u području modeliranja modela podataka proizvoda mogu se podijeliti na rješavanje problema:

- semantike modela [31],
- integracije više računalnih procesa [32],
- minimalizacije troškova razvoja proizvoda [33],
- obrade baza podataka [34].

Konvencionalni CAD sustavi nisu koncipirani za asistiranje konstruktoru u rješavanju kompleksnih problema što potiče istraživanja integriranih programskih okolina [35]. Gauseimer i Vajna bave se aspektima kvalitete i efikasnosti razvoja proizvoda [36], [33]. Modeli radnih okolina[11] razvijaju se sa više različitih pristupa, no u ovom području tek predstoji usaglašavanje ciljeva i metoda [37], [38].

Snažan poticaj istraživanjima dao je razvoj modela procesa konstruiranja temeljen na paradigmi umjetne inteligencije [39]. Blount i Clarke tretiraju konstruiranje kao aktivnost rješavanja problema koja se može automatizirati [40]. Mostow [41] istražuje tehniku "kompilacije znanja" kao transformaciju eksplicitno prikazanog znanja iz domene u efikasni algoritam za izvođenje određenih zadataka u domeni. Uloge ekspertnih sustava i sustava za upravljanje bazama podataka u okolini "konstruiranja za izradu" razmotrene su u radu [42].

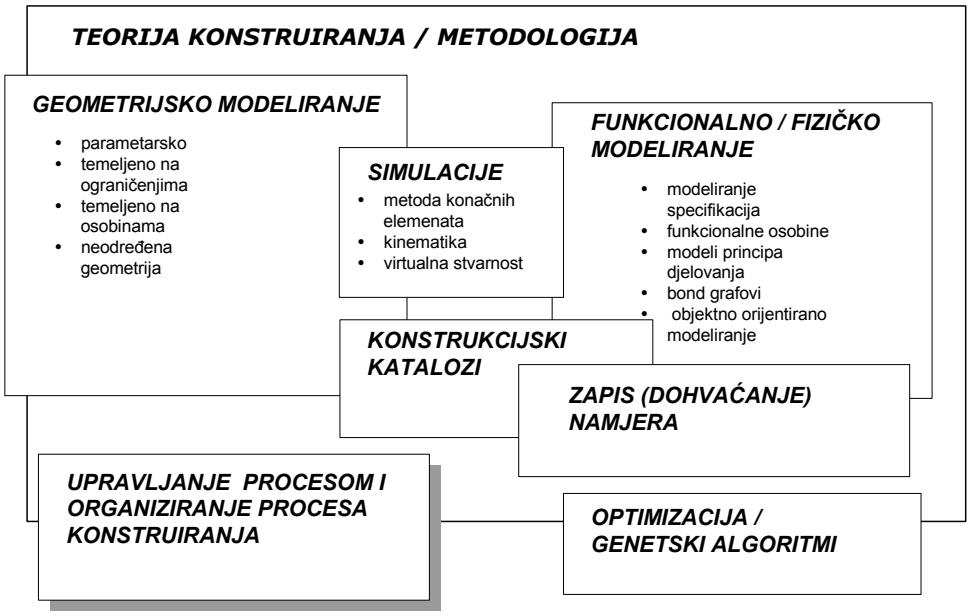
U razvoju "inteligentnih agenata" često se upotrebljavaju metode rješavanja problema analogijom, primijenjene na inženjersko konstruiranje [43]. Značajan utjecaj na formalizaciju i rješenja računalnih metoda u inženjerskom konstruiranju pokazala su istraživanja koja kombiniraju umjetnu inteligenciju i metode optimizacije [44]. Istraživanja primjene ekspertnih sustava sugeriraju da standardni pristupi nisu pogodni za konstruiranje i planiranje. Stoga u ovom području treba razviti nove pristupe [45]. Ullman i D'Ambrosio predložili su taksonomiju računalne podrške konstruiranju [46]. Općenitiji pristupi bave se razvojem "okvira" znanja[12]. Forde i koautori uvode strategiju koncipiranja objektno orijentiranog "okvira" [47]. Smithers [48] svoju teoriju konstruiranja (teorija razina znanja) upotrebljava kao praktičnu alternativu spoznajnoj teoriji konstruiranja.

Razliku između modeliranja procesa konstruiranja [49] i modeliranja određenog zadatka naglašena je činjenicom da su računalni modeli, ovisno o domeni, dovedeni gotovo do savršenstva, u suženim prostorima interesa. Računalni modeli zadataka konstruiranja općenito su kreirani za dobro definiranu klasu problema kao u [50], [51] i u mnogim drugim radovima.

Općenitiji pristupi modeliranju procesa konstruiranja pokušaji su koji podržavaju izvođenje i označavanje alternativnih koncepata upotrebljavajući negeometrijske entitete za podršku kompleksnim konstrukcijskim projektima. Pohranjeni koncepti omogućuju implicitno zapisivanje konstrukcijskog znanja [52], [53]. Proces konstruiranja i potreba za automatiziranim

[11] designers workbench
[12] knowledge frameworks

sustavima objašnjavanja, kao i njihova interakcija s konstruktorom, predmet su kontinuiranih istraživanja [54].



Slika 2-2 Područja istraživanja primjene računala u znanosti o konstruiranju prema [55]

2.1 Konstruiranje kao rješavanje zadatka

Istraživanja koja proučavaju proces konstruiranja, teorije konstruiranja i metodologije konstruiranja pokušavaju pronaći odgovore na sljedeća pitanja [56]:

- što je proces konstruiranja,
- kako se odvija proces konstruiranja,
- na koji način se kreiraju i ocjenjuju konstrukcijske mogućnosti (alternative),
- na koji način se cilj konstruiranja transformira u fizički oblik,
- na koji način konstruktori pretražuju i proučavaju konstrukcijske alternative,
- kako se treba održavati povijest nastanka konstrukcije,
- na koji način se treba prikazati tijek procesa konstruiranja.

Zakonitosti koje vladaju u procesu konstruiranja kao kreativnoj ljudskoj djelatnosti, nastoje se opisati teorijama čija su polazišta uvjetovana okolinom i svim njenim uplivima.

Različitost poimanja proces konstruiranja možemo opisati nizom definicija:

- "Konstruiranje je primjena znanstvenih principa, tehničkih informacija i mašte radi definiranja strukture stroja ili sustava predviđenog za izvršavanje prethodno zadanih funkcija s najvećom ekonomičnošću i efektivnošću." (G. B. R. Fielden)
- "Konstruiranje je proces pretraživanja u kojem se zadovoljavajuće konstrukcijsko rješenje pronalazi iz skupa alternativnih." (S. J. Gero)
- "Konstruiranje je konstantan proces između onog što želimo postići i na koji način to želimo postići. " (N. P. Suh)

- "Konstruiranja je skup aktivnosti koje vode od utvrđenog zahtjeva na proizvod do generiranja skupa informacija, potrebnih za proizvodno sačinjavanje proizvoda." (A. Kostelić)

Proces konstruiranja je proces u kojem se provode određene aktivnosti da bi se kreirao model koji zadovoljava zadane specifikacije [12]. Prema [25] mogu se navesti sljedeće značajne karakteristike procesa konstruiranja:

- Proces konstruiranja je sinteza relativno dobro poznatih elemenata u jednu jedinstvenu, otprije poznatu cjelinu sa zahtijevanim svojstvima. Ta sinteza iziskuje kreativan i stvaralački rad. Iz toga proizlazi važna karakteristika procesa konstruiranja da čovjek mora kontrolirati proces, odnosno imati pretežan udio u donošenju potrebnih odluka.
- Konstruiranje se može promatrati i kao proces učenja.
- Svaki konstrukcijski zadatak može se riješiti na mnogo različitih načina, odnosno može rezultirati različitim strojnim sustavima ili sklopovima. Ta karakteristična mnogostrukost mogućih rješenja uvjetovana je količinom svojstava proizvoda koje treba odrediti u postupku konstruiranja.
- S filozofskog gledišta proces konstruiranja je također i spoznajni proces: sustav na početku nepoznat spoznaje se, odnosno postaje poznat. Na temelju toga može se reći da je spoznajna teorija također jedan izvor općih zakonitosti za proces konstruiranja.
- Proces konstruiranja je vrlo zahtjevan kreativan rad, ali ne smije se promatrati kao umjetnost, nego kao znanstveni rad. Određeni misaoni postupci (intuicija, nastajanje ideje) koji se ne mogu racionalno objasniti imaju obilježja umjetničke kreativnosti. Ti postupci ne mogu se formalizirati u svrhu stvaranja cjelovitog teoretskog prikaza konstrukcijskog procesa.
- Svaki proces konstruiranja može se razložiti u manje cjeline (faze, dijelove procesa, etape ili operacije) koje čine strukturu procesa.
- Konstrukcijski proces se može promatrati i kao proces prerade informacija, gdje se na temelju ulaznih zahtjeva generira skup informacija koje opisuju proizvod.
- Do sada pretežno samostalna djelatnost (u okviru zadatka), sve više se pretvara u timski rad u kojem se koriste prednost većeg informacijskog kapaciteta i međusobne razmjene ideja i postupaka.
- Velika kompleksnost međusobno proturječnih zahtjeva dovodi do potrebe za višestrukim ponavljanjem određenih faza nakon početnog apstrahiranja i postavljanja pretpostavki, dok se ne odrede potrebne vrijednosti. Iterativni postupak je jedna od tipičnih karakteristika procesa konstruiranja.,

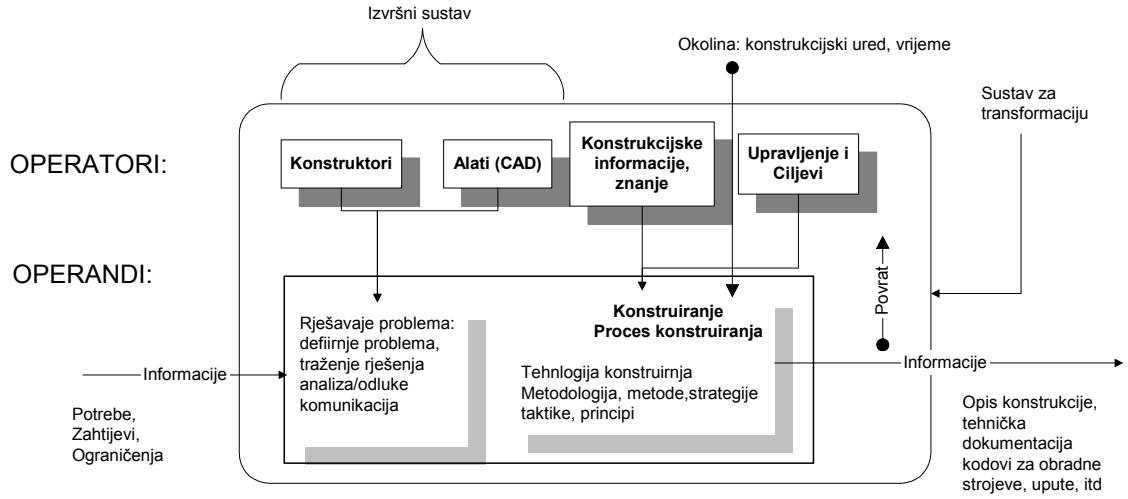
Opći model konstruiranja koji je postavio Hubka 1973. godine daje pregled skupa aktivnosti, vrsta procesa i upliva na proces konstruiranja. Razvojem općeg modela konstruiranja Hubka je tijekom vremena postavio teoriju tehničkih sustava, te unutar nje noviji opći proceduralni model konstruiranja.

Opći model procesa konstruiranja prikazan na slici 2-3 može se interpretirati na slijedeći način:

- konstruiranje je proces transformiranja informacija od zahtjeva kupaca do potpunog opisa predloženog tehničkog sustava,

- prikazuje se osnovna struktura procesa, uključujući regulacijske, kontrolne i pomoćne procese,
- kao direktni operatori, konstruktori i njihova sredstva za rad izvode akcije (efekte) na skupu informacija (operanda konstrukcijskog procesa),
- prikazuje se utjecaj ostalih različitih faktora na proces konstruiranja (metode rada, radna okolina, upravljanje).

Prema [58] i [59] zadatak procesa konstruiranja je pretvorba postavljenih zahtjeva u opis željenog strojnog sustava. Drugim riječima potrebno je definirati i opisati uređaj (proizvod) koji će proizvoditi željene učinke i pri tome zadovoljavati zahtijevana svojstva. Osim postavljenih zahtjeva potrebno je voditi računa i o ostalim aspektima tehničke i ekonomske naravi - npr. tehnologičnosti, troškovima, itd. - što vrijedi i za sam proizvod, i za njegovo funkcioniranje u životnom vijeku.



Slika 2-3 Opći model procesa konstruiranja [57]

U praksi se često javljaju situacije u kojima konstrukcijski zadatak, odnosno lista zahtjeva na proizvod nisu točno ili potpuno definirani što može rezultirati nesporazumima i posljedično povećanim troškovima i prekoračenjem rokova. Za ilustraciju dana je struktura elemenata definicije zadatka prema [60].

- definiranje ciljeva - ima prije svega strateški smisao, a daljnja razrada ima taktički smisao,
- opis problema na slobodniji način,
- navesti što su nužni zahtjevi, a što željeni,
- definiranje uvjeta za realizaciju zadatka uz provjeru da li su predviđeni uvjeti točni,
- razmatranje mogućnosti daljnjeg razvoja rješenja (da li se zadatak izvršava kao nešto konačno ili u prijelaznom obliku),
- definirati koja svojstva rješenje mora imati,
- definirati koja svojstva rješenje ne smije imati,
- raščišćavanje zadatka - provjera da li je sve jasno i potpuno, te da li je nešto inkompatibilno.

Ako se zadatak definira u fazi nuđenja proizvoda, najčešće treba još uključiti i predprojekt i kalkulaciju.

Prema polazištu zahtjeva koji definiraju zadatak, zadaci se mogu razlučiti na:

- interne razvojne zadatke koji su potaknuti unutar samog proizvodnog okruženja - najčešće temeljem povratnih sprega u informacijskim tokovima,
- zadatke potaknute indirektnim zahtjevima tržišta - temeljem praćenja stanja tržišta i konkurencije,
- zadatke potaknute direktnim zahtjevima tržišta odnosno narudžbom.

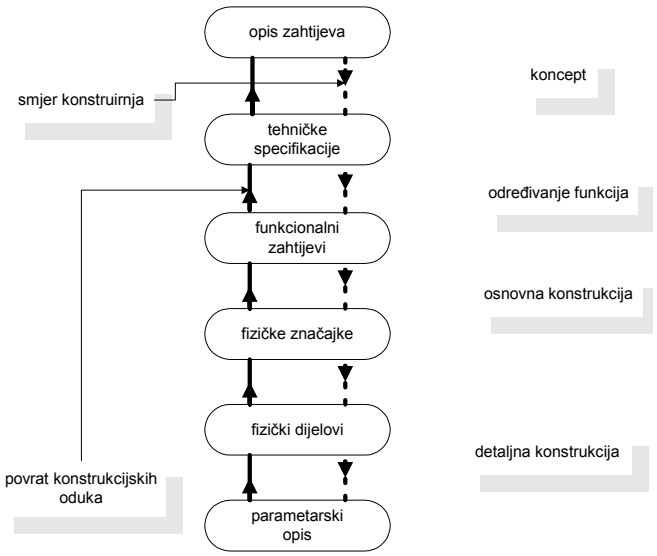
2.1.1 Faze rješavanja konstrukcijskog zadatka

Zadatak koji se prenosi konstruktoru, je pored ostalog, rezultat izbora koji tim planera (rukovodioci, prodaja, razvoj, proizvodnja) vrši prema raznim kriterijima (tržište, želje kupaca, troškovi, kapaciteti, rokovi, rizik, itd.) [60]. Prema [61] proces konstruiranja može se podijeliti na faze, nakon definiranja zadatka raščišćavanjem liste zahtijeva, slijedi:

- **Koncipiranje** predstavlja onaj dio procesa konstruiranja pri kojem se nakon raščišćavanja svih zahtjeva vezanih za zadatak utvrđuje principijelno rješenje zadatka, traženjem odgovarajućih metoda i principa djelovanja. Dobivena rješenja vrednuju se prema kriterijima danim u listi zahtjeva.
- **Projektiranje** je faza procesa konstruiranja u kojoj se utvrđuje funkcionalno, strukturalno i ekonomsko rješenje zadatka u takvom opsegu da je moguća daljnja konstruktivna razrada. Izrađenost projekta treba biti takva da se može izvršiti kalkulacija i nuđenje. Projekt treba obuhvatiti sve što sustav sadrži, ali ne daje odgovor kako to treba izraditi. Pošto se iz projektnog rješenja odstrane slaba mjesta, pristupa se tehničkom i ekonomskom vrednovanju i nakon toga optimiranju projektnih detalja.
- **Konstrukcijska razrada** ima zadatak definirati informacije kako treba konstrukciju izraditi odnosno kako treba fizički izgledati kao gotov proizvod. U toj fazi detaljno se razrađuje sva potrebna tehnička i tehnološka dokumentacija za odabranu varijantu rješenja.

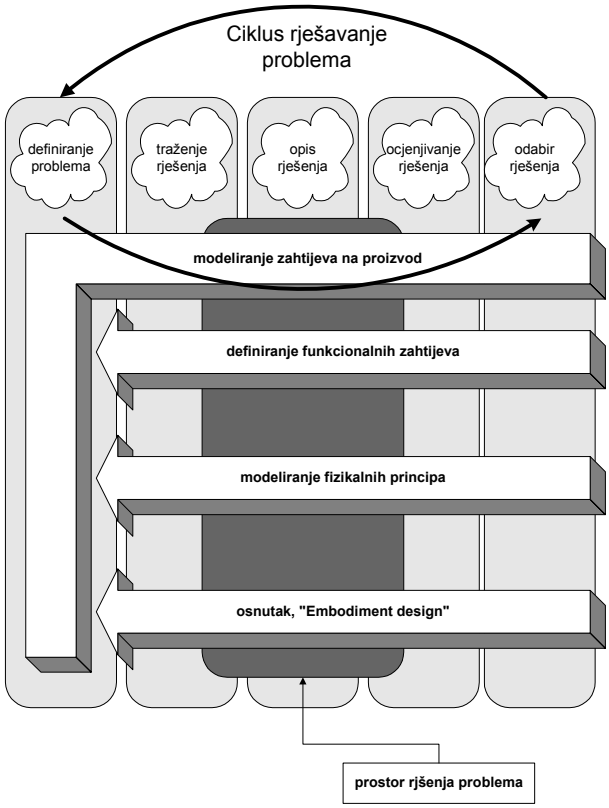
U procesu rješavanja zadatka postavljene granice između navedenih faza nisu uvijek jasne, a isto tako često se može i naizmjence obavljati aktivnosti iz različitih faza. Određene vrste operacija odnosno aktivnosti mogu se na isti način odvijati u svim fazama. Iz tih razloga sustav za podršku konstrukcijskom procesu treba pokušati modelirati na taj način da se može primijeniti u svim fazama procesa konstruiranja.

Murtagh [62] opisuje razine kroz koje prolazi proizvod prilikom procesa konstruiranja (Slika 2-4). Pri tome ističe da je nemoguće postići sekvencijalno aktiviranje pojedinih razina (započinjanje i završavanje pojedinih zadataka), već da se slijed aktivnosti prebacuje između razina. Učestalost skokova između susjednih razina ovisi o poteškoćama u zadovoljenju konstrukcijskih zahtijeva. Iz čega se može zaključiti da se proces konstruiranja ne može ograničiti samo na promišljanje kroz "horizontalne" razine, već da je potrebno uzeti u obzir i "vertikalne" razine.



Slika 2-4 Razine u procesu konstruiranja prema [62]

Zajednička osobina svih pristupa proučavanju procesa konstruiranja je kretanje prema "naprijed" (prema završetku procesa konstruiranja tj. proizvodu) raščišćavanjem zahtijeva na proizvod, napredujući preko opisivanja funkcija proizvoda da bi se pronašlo principijelno rješenje i kreirao opis proizvoda [63], [64]. Na osnovu navedenog Grabovski [65] izvodi četiri osnovne razine u procesu konstruiranja (Slika 2-5).



Slika 2-5 Prikaz konstrukcijskog procesa u strojarstvu prema [65]

Proces konstruiranja počinje definiranjem zahtijeva na proizvod. Ova razina služi kao referenca za određivanje kvalitete rezultata dostignutog procesom raščišćavanja zahtijeva. Može se reći da zahtjevi na proizvod određuju željene osobine budućeg proizvoda. Razina definiranja

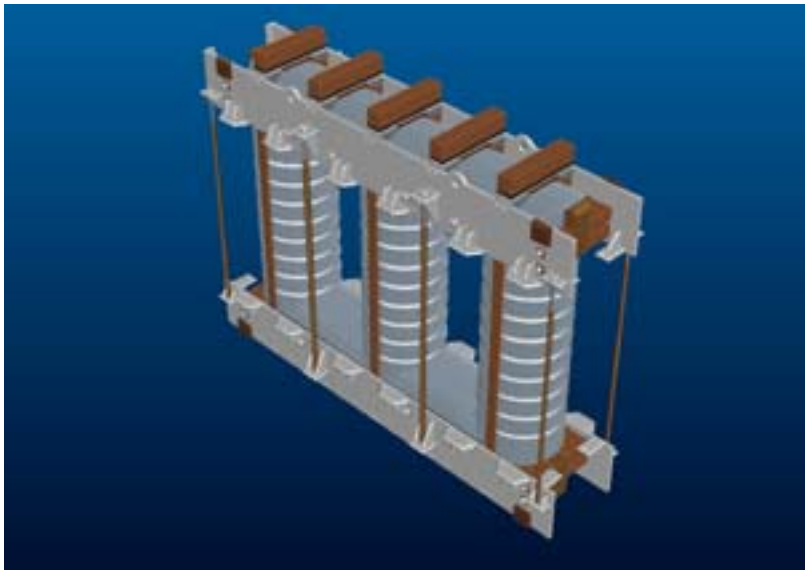
funkcionalnih zahtijeva koristi se za opisivanje funkcije proizvoda i pronalaženje konstrukcijskog rješenja. Definiranje funkcionalnih zahtijeva uključuje određivanje jedne funkcije proizvoda tj. međurelacije između funkcija (funkcionalne strukture) i skupa parcijalnih rješenja. Nakon određivanja funkcije proizvoda određuje se principijelno rješenje. Modeliranje fizikalnih principa uključuje sve informacije koje opisuju fizikalne efekte i njihovu strukturu, osnovne razloge uporabe određenih materijala i potrebne matematičke jednadžbe. Nakon završetka ove razine jedno fizikalno pravilo pridruženo je jednoj funkciji proizvoda. Osnutak ("Embodiment design") proizvoda je najkonkretnija razina, koja završava kreiranjem geometrijskog prikaza modela proizvoda. U svakoj razini konstrukcijskog procesa mora se završiti potpuni ciklus rješavanja problema

2.2 CAD

2.2.1 Modeliranje krutim tijelima

Razvoj aplikacija za kreiranje modela kao pomoć pri konstruiranju započeo je 50-tih i 60-tih godinama prošlog stoljeća [66]. Prvi CAD sustavi bili su orijentirani samo na dvodimenzionalno crtanje tj. kreiranje tehničke dokumentacije [67]. Tek tijekom 60-tih i početkom 1970. godine pojavile su se prvi zahtjevi za proširenjem 2D CAD sustava uključenjem treće dimenzije. No, prelazak na 3D pokazao se kao znatno kompleksniji zadatak od samo dodavanja treće koordinate. Smatralo se da je potrebno imati središnji trodimenzionalni model na osnovu kojeg bi se mogli automatski kreirati dvodimenzionalni crteži.

Tijekom 1970. godine počeli su radovi na implementiranju već poznatih algoritama za prikaz skrivenih linija i površina. U isto vrijeme pojavili su se i prvi CAD sustavi koji koriste kruta tijela kao osnovu za kreiranje modela. Uskoro su se izdvojila dva pristupa: "Boundary representation"[13] na Sveučilištu Cambridge i CSG[14] (Slika 2-6) model na Sveučilištu Rochester.



Slika 2-6 CSG 3D model

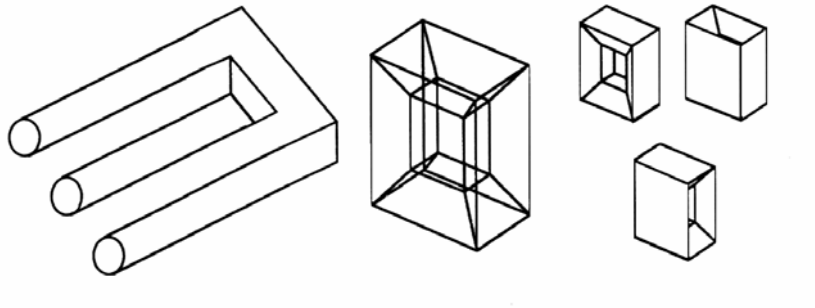
[13] Model se sastoji od faceta koji su podskup ravninskih, kvadratnih i toroidoalnih površina

[14] CSG – Constructive Solid Geometry – sastoji se od konačnog broja Boolean operacija primijenjenih na poluprostorima određenim matematičkim nejednadžbama

Osnovna prednost CAD sustava koju koriste kruta tijela je ta što je praktički nemoguće kreirati geometriju tj. model koji nije ispravan [68] (Slika 2-7)[15]. Korisnik u radu koristi primitive punih tijela, a ne geometrijske objekte niže razina kao što su točke, linije, kružnice, itd. Dugo se mislilo da će CAD sustavi koji koriste kruta tijela istisnuti "crtaće" sustave, no do toga još nije došlo. Čak i danas većina CAD sustava temelji se na dvodimenzionalnom crtanju. Umjesto u području razvoja proizvoda uporaba CAD sustava koji koriste kruta tijela očituje se u izradi modela, izradi dokumentacije, proučavanju pakiranja, robotici, definiciji geometrije te FEM[16] analizi i izradi CNC[17] programa. Tijekom godina razvile su se nove metode koje su doprinijele kvaliteti CAD sustava [69]. Neke od njih su: NURBS površine, slobodne površine, elastični objekti, "physical-based modeling" i "volume-based modeling".

Geometrijske modele, koji se koriste u CAD aplikacijama možemo podijeliti u tri glavne skupine [68]:

- grafički modeli – pretežno namijenjeni kreiranju tehničke dokumentacije i ilustracija,
- površinski modeli – namjena im je podrška procesu kreiranja i izradi slobodnih površina. U ovo područje mogu se svrstati i napredne krivulje (splajn i NURBS), iz razloga uske povezanosti sa površinama, iako bi bolje odgovarale grafičkim modelima,
- kruta tijela – namjena im je da potpuno obuhvate trodimenzionalnu geometriju fizičkog tijela.



Slika 2-7 Problemi u radu sa žičanim modelom

CAD sustavi ne podržavaju sveobuhvatan proces konstruiranja, odnosno sve faze tog procesa. Između ostalog nedostaju i kvalitativni aspekti (nepotpuna provjera grešaka, strukturiranje, estetika, inovacija, ...). Iz navedenog proizlazi da je u biti nedostatak inteligencije najveći nedostatak tih sustava, a što je posljedica:

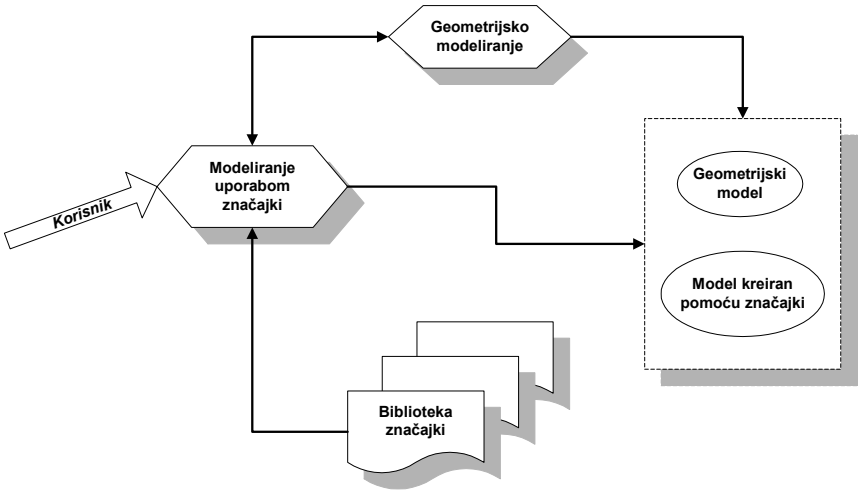
- nemogućnost obrade simboličkog prikaza konstrukcije,
- nemogućnost reprezentacije znanja o konstruiranju, te nepostojanje mehanizma zaključivanja,
- nepoznavanje formalnih zakona procesa konstruiranja, odnosno relacija između zahtjeva i konstrukcijskih rješenja.

[15] Na slici 2-7 lijevo ilustriran je neispravan model, dok je na istoj slici desno ilustriran informacijski nepotpun model.
[16] FEM – Finite Element Method – metoda konačnih elemenata
[17] CNC – Computer Numeric Control – izrada programa za numerički upravljane strojeve

Budući da se u novije vrijeme od CAD sustava zahtijeva znatno veća funkcionalnost, nego što je osnovna karakteristika "klasičnih" CAD sustava, razvoj je krenuo prema uporabi značajki i parametara.

2.2.2 Modeliranje značajkama

Modeliranje pomoću značajki[18] (Slika 2-8) je tehnika modeliranja koja omogućuje integriranje geometrijskog modeliranja i konstruiranja [70]. Jedna od osobina ove tehnike modeliranja je obogaćenje podataka vezanim za prikaz proizvoda semantičkim informacijama. Dodatne semantičke informacije omogućuju napredniju komunikaciju u procesu konstruiranja [71].



Slika 2-8 Modeliranje pomoću značajki

Jedna od osnovnih motivacija za uporabu značajki je proizašla iz problematike vezane za planiranje i modeliranje procesa proizvodnje (CAPP[19]) uporabom CAD sustava. Tijekom 1984. godine istraživanja u području planiranja proizvodnje i geometrijskog modeliranja počela su se orijentirati na problematiku uporabe značajki. Kao rezultati u tvrtci John Deere kreiran je elaborat na temu sistematizacije i parametara značajki te njihov značaj u području proizvodnje. Pratt i Wilson su 1987 prvi predložili koncept modeliranja pomoću značajki ("Design by Feature"). Prototipni sustavi temeljeni na modeliranju pomoću značajki kreirani su 80-tih godina pretežito na sveučilištima. Komercijalno dostupni sustavi pojavili su se u kasnim 80tim. Prvi paketi koji su se pojavili bili su ProENGINEER® tvrtke PTC[20] i Cimplex® tvrtke Cimplex Inc.

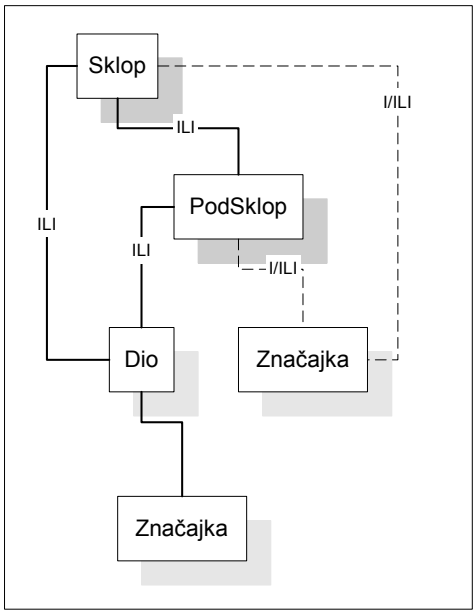
Pojam značajke nije lako objasniti, jer sama riječ može imati više značenja. No, možemo reći da značajka prikazuje inženjersko značenje [68]. Značajka se može shvatiti i kao gradivni dio definicije proizvoda ili geometrijskog poimanja proizvoda. Shah [72], [73] karakterizira značajku na sljedeći način:

- značajka je fizički dio dijela[21],
- značajka se može dodijeliti generičkom obliku,
- značajka ima inženjersko značenje (važnost),

[18] Modeliranje pomoću značajki se odnosi na Feature Based Modeling tehniku modeliranja
[19] CAPP – Computer Aided Process Planning
[20] PTC – Parametric Technology Corporation
[21] Pod dijelom se smatra part tj model

- značajka ima predvidljive osobine.

Martino [70] definira koncept značajki kao skup elemenata oblika, koji imaju funkcionalno značenje u zadanom kontekstu, i omogućuju vezu između funkcionalnosti i forme. Uporabom značajki moguće je kreirati prikaz proizvoda više razine, koji se može koristiti u sustavima za analizu i simulaciju. Budući da su značajke vezane za zadanu domenu omogućuju prikaz proizvoda sa različitih točaka gledanja i vezu prema sustavima u drugim fazama procesa kreiranja proizvoda. Značajke predstavljaju inženjersko značenja geometrije, dijela ili sklopa. Inženjersko značenje može sadržavati formalizaciju funkcija značajke, objašnjenje na koji način se značajka može napraviti ili ponašanje u određenim situacijama [74].



Slika 2-9 Odnos sklop, podsklop, dio, značajka

Značajka je fizički entitet koji tvori fizički dio dok je atribut karakteristika ili mjerilo kvalitete proizvoda. Tako se može tvrditi da je atribut karakteristika ili osobina značajke, dok je, značajka element dijela, a dio je element sklopa (Slika 2-9). Atributi se mogu koristiti na bilo kojoj razini od karakterizacije značajke do karakterizacije sklopa. Neki atributi značajke mogu biti:

- pozicija,
- orijentacija,
- dimenzije oblika,
- vrijednosti tolerancija,
- geometrijska ograničenja,
- kvaliteta obrade površine.

Budući da se pod pojmom značajke može smatrati široki spektar fizičkih osobina ili karakteristika dijela neophodno je tipizirati značajke. Značajke se mogu podijeliti u tipove prema sljedećim kategorijama značajki prema [68]:

- značajke oblika ("Form feature") – predstavlja dio nominalne geometrije, stereotipni oblici,

- značajke tolerancije ("Tolerance feature") – odmak od nominalnog oblika, veličine ili položaja,
- značajke sklopa ("Assembly feature") – grupiranje različitih značajki u svrhu definiranja relacija u sklopu, kao što su uvjeti sklapanja, relativan položaj ili orijentacija dijela, različiti oblici spojeva, kinematičke relacije,
- značajke funkcije ("Functional feature") – skup značajki vezanih za određenu funkciju, može uključivati "design intent"[22], negeometrijske parametre vezane za funkciju ili učinak,
- značajke materijala ("Material features") – sastav materijala, tretman, uvjeti, itd.

Značajke oblika, tolerancije i sklopa usko su povezane sa geometrijom dijela te se iz tog razloga skupno nazivaju geometrijske značajke. U postojećim CAD sustavim, temeljenim na značajkama, u većini slučajeva koriste se geometrijske značajke i to posebice značajke oblika i pojedine značajke sklopa .

Jedna od osnovnih uloga značajki je kreiranje veze između entiteta definicije proizvoda. Veza entiteta definicije proizvoda omogućuje obuhvaćanje konstrukcije ili izradbenih ograničenja. Iz tog razloga oblik, ponašanje i inženjerska važnost značajke mora se ugraditi u njen prikaz. Informacije koje mogu biti uključene u značajki su sljedeće:

- generički oblik (topologija i/ili geometrija),
- parametri izmjera,
- ograničenja parametara i ograničenja relacija,
- podrazumijevajuće vrijednosti parametara,
- metoda vezivanja za lokaciju,
- parametri lokacije,
- metode orijentacije,
- parametri orijentacije,
- ograničenja dimenzija, lokacije ili orijentacije,
- tolerancije,
- procedure za kreiranje geometrije,
- algoritmi za prepoznavanje,
- parametri proračunati na osnovu drugih značajki,
- pravila i procedure nasljeđivanja,
- pravila i procedure valorizacije,
- negeometrijski atributi (broj dijela, funkcija, itd.).

Na osnovu navedenog navest će se poboljšanja koja donosi uporaba značajki u CAD sustavima [75]:

- Jedna od najvažnijih prednosti značajki je u mogućnosti opisa proizvoda primitivama više razine[23] određenih kao generičke apstrakcije oblika koji se ponavljaju. Otvorena struktura značajki omogućuje uključivanje samo onih primitiva koji su relevantni za određenu domenu.

[22] u tekstu se pod design intent smatra konstruktorova nakana ili ideja
[23] u tekstu se pod primitvama više razine misli na kruta tijela (kocka, cilindar, kugla, stožac, itd)

- Dopuštajući opisivanje modeliranja preko entiteta iz zadane domene, model kreiran uporabom značajki u većoj mjeri sadržava konstruktorovu (korisnikovu) namjeru nego klasični geometrijski modeli. No, trenutno prevladavajuća uporaba samo geometrijskih značajki dozvoljava samo sadržavanje geometrijskih aspekata modela.
- Uporaba značajki dovela je do kreiranja naprednijih korisničkih sučelja za manipulaciju geometrijom. Naprimjer, kod "klasičnih" sustava korisnik ukoliko želi smanjiti promjer rupe, mora prvo "zatvoriti" postojeću te zatim kreirati novu. Kao nus pojava javljaju se dodatni parametri u bazi crteža (podaci o staroj rupi) koji nemaju nikakvu svrhu. Pri modeliranju uporabom značajki korisnik samo odabere parametar koji želi promijeniti i unese novu vrijednost. Na osnovu vrijednosti, promijenjenog atributa, provedu se novi proračuni modala (ugrađena provjera ispravnosti modela) te se model regenerira.

3 Znanje u procesu konstruiranja

Proces konstruiranja je jedno od najzahtjevnijih područja uporabe CAD tehnologije. Konstruiranje uključuje kreiranje, upravljanje i manipulaciju kompleksnim entitetima koji su međusobno povezani različitim semantičkim relacijama. Opisi elemenata konstrukcije i relacija među njima mijenjaju se tijekom procesa konstruiranja. Rezultat procesa konstruiranja, realiziran uporabom CAD aplikacije, treba sadržavati zapis procesa nastajanja komponenti i proizvoda te pripadajućih znanja značajnih za životni ciklus proizvoda [76].

Konstruiranje je između ostalog i proces upravljanja, manipulacije i obrade informacija [4]. Konstruktor tijekom procesa konstruiranja koristi velike količine informacija u cilju ispunjenja zadanog zadatka. Količina i oblik tih informacija, koja su sastavni dio konstrukcijskog znanja, ovisi o više parametara koji direktno ovise o fazi procesa konstruiranja.

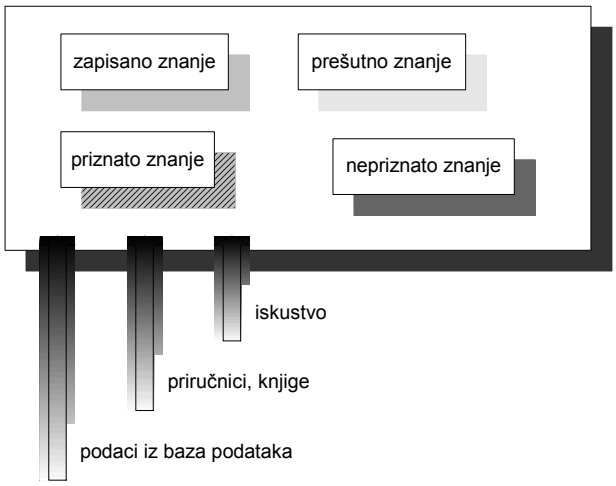
Koncipiranje programskog sustava za rad s konstrukcijskim znanjem zahtijeva ispravno definiran model konstrukcijskog znanja. Formalna struktura modela, koji će se koristiti za definiranje konstrukcijskog znanja, mora biti u mogućnosti opisati elemente znanja i relacije između njih. U cilju kreiranja koncepta modela konstrukcijskog znanja neophodno je proučiti strukturu znanja u procesu konstruiranja.

Postoji više načina na koji bi se mogla prikazati struktura konstrukcijskog znanja. Prema [77] predložene su sljedeće kategorije:

- formalno i neformalno znanje,
- izvori znanja,
- generičko i specifično znanje,
- znanje koje se koristi u više faza procesa konstruiranja,
- strukturno, deskriptivno i proceduralno znanje.

Tomiyama [77] generalno dijeli znanje (pa tako i konstrukcijsko znanje) u dva oblika: priznato i nepriznato znanje. Gledano iz druge dimenzije znanje dijeli na prešutno i zapisano. Prešutno znanje je oblik znanja koji je implicitno ili eksplicitno prepoznatljiv ljudima i koje se koristi prilikom donošenja odluka, ali ga je teško formalno opisati. Zapisano znanje je znanje opisano u bilo kakvom obliku koji je prepoznatljiv ljudima. Na osnovu navedenog može se reći da se znanje može podijeliti, kombiniranjem prethodno navedenih oblika znanja, u tri kategorije (Slika 3-1):

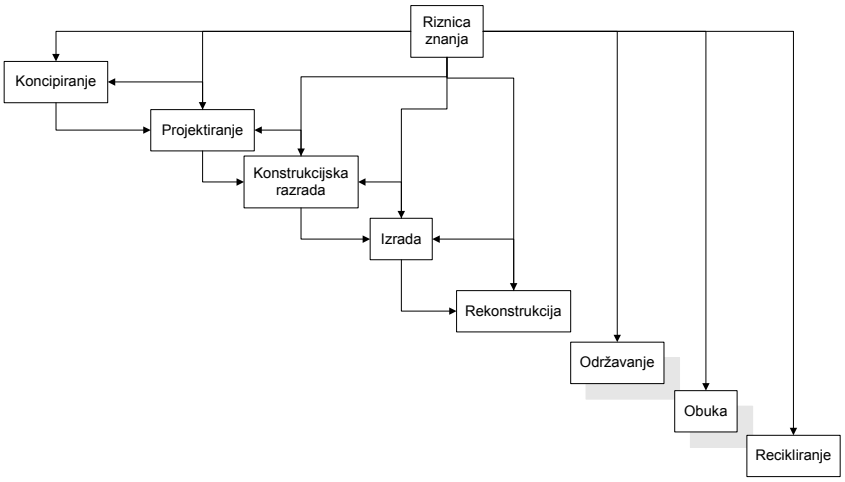
- priznato i zapisano,
- priznato i prešutno,
- nepriznato i prešutno.



Slika 3-1 Klasifikacija znanja

Kombinacija nepriznatog i zapisanog znanja sama po sebi nema smisla. Osnovni cilj sistematizacije znanja prema [77] je pretvorba priznatog i prešutnog znanja u priznato i zapisano.

Barhes [78] dijeli znanje prema prirodi i opsegu na eksplicitno i implicitno te na meta znanje. Pod eksplicitnim znanjem smatra priručnike, crteže, neformalne zabilješke, baze podataka, ekspertne sustave, video zapise itd. Iz nabrojenog se može zaključiti da se eksplicitno znanje ne može strukturirati u jedan jedinstven informacijski sustav. Implicitno (smatra se znanje stečeno iskustvom) znanje je potrebno identificirati, rekonstruirati pomoću eksperta, pohraniti i održavati, a vrlo teško ga je formalizirati. Pod meta znanjem podrazumijevaju se tehničke natuknice koje uključuju različite odluke i razloge njihovog donošenja, nastale greške te naprečac donesene odluke i razloge njihovog donošenja. Ovo znanje, koje nije zapisano kao dio tehničke dokumentacije, je ključno za razlikovanje uspješnog od neuspješnog projekta. Barhes predlaže kreiranje riznice znanja (Slika 3-2) tvrtke iz koje bi svaki korisnik moga crpiti informacije i potrebna znanja.



Slika 3-2 Riznica znanja prema [78]

Reddy i grupa autora proučava problematiku razmjene znanja između zemljopisno razmještenih timova [79]. Proučavanje se temelji na tvrdnji da je konstruiranje aktivnost u kojoj se intenzivno koriste informacije i znanje, te da, u procesu konstruiranja u kojem je uključeno više konstruktora, razmjena informacija i znanja postaje središnji problem. Pri tome je znanje kategorizirano na sljedeći način:

- osobno znanje – znanje stečeno obrazovanjem i iskustvom,
- postojeća konstrukcijska rješenja – znanje o postojećim konstrukcijskim rješenjima uporabljenim prilikom konstruiranja sličnog proizvoda,
- lista zahtjeva – znanje dobiveno od naručitelja,
- rezultati analize – znanje dobiveno na osnovu rezultata zadovoljenja zahtjeva od strane predloženog konstrukcijskog rješenja,
- znanje o izradi – znanje o tome na koje će način proizvod biti izrađen (postupci, materijali, obrade, ...),
- znanje o sklopivosti – znanje o tome na koji će se način proizvod sklopiti,
- znanje o dobavljačima,
- razna znanja – troškovi, raspoloživi budžet, rokovi, ...

Batanov [80] opisuje mogućnost integracije konstrukcijskog znanja i modeliranja uporabom značajki. Integracija se realizira uvođenjem značajki koje definiraju dva tipa konstrukcijskog znanja: znanje vezano za podatke ("fact-features") i znanje vezano za pravila ("rule-features"). Za različite potrebe različiti skupovi značajki se mogu uporabiti ovisno o potrebama. Zanimljiva podjela znanja o proizvodu opisana je u [81]. Znanje koje se koristi u procesu konstruiranja proizvoda podijeljeno je na: znanje o proizvodu, znanje o konstrukcijskim akcijama i znanje o procesu konstruiranja. Pod znanjem o proizvodu podrazumijeva se znanje o funkciji, ponašanju i konstrukciji. Znanje o konstrukcijskim akcijama sadrži znanja o tome na koji način se moraju provesti određene akcije prilikom konstruiranja. Znanje o procesu konstruiranja sadrži znanja o načinima organizacije konstrukcijskih akcija. Znanje o proizvodu dijeli se na: trenutno znanje o proizvodu – znanje o proizvodu koji se trenutno konstruira i znanje o proizvodima iz domene – znanje o proizvodima iz domene proizvoda koji se konstruira. Nadalje, znanje o proizvodima iz domene dijeli se na: znanje o općenitim pojmovima iz domene i znanje o postojećim proizvodima. Drugačiji načini klasifikacije i pogleda na konstrukcijsko znanje mogu se naći u [82], [83], [84], [85], [86].

Zadataka konstruktora je kreirati proizvod koji posjeduje određenu funkciju [87]. Prilikom konstruiranja konstruktor koristi komponente koje zadovoljavaju određene funkcije koje tvore funkciju proizvoda. U kontekstu manipulacije i upravljanja konstrukcijskim znanjem o proizvodu, znanje o funkciji proizvoda ima važnu ulogu [77]. Problematika vezana za prepoznavanje, definiranje i uporabu znanja o funkciji kao i znanja i ponašanju, koje je blisko povezano sa znanjem o funkciji, proizvoda tematika je više znanstvenih radova [88], [89], [90], [91].

Predodžba o mogućoj funkciji proizvoda i ponašanju proizvoda može se pomiješati što može dovesti do problema prilikom definiranja proizvoda. Funkcija proizvoda je definirana kao subjektivan opis ponašanja u danom kontekstu, tj. funkcija apstrahira ponašanje, i ovisi o korisniku. Ponašanje se s druge strane može derivirati na osnovu strukture proizvoda [92]. Odnos između funkcije, ponašanja, strukture proizvoda i cilja tj. konstrukcijskog zadatka može se

sažeto opisati prema [93]: ispunjenje konstrukcijskog zadatka je omogućeno funkcijom postignuto ponašanjem i prikazano strukturom proizvoda.

Ukoliko bi se u programski sustav za pomoć konstruktoru u procesu konstruiranja implementirala znanja o funkciji i ponašanju proizvoda omogućilo bi se lakše praćenje i bolje razumijevanje pojedinih konstrukcijskih odluka i rješenja. No, za kvalitetnu implementaciju neophodno je sistematizirati i klasificirati znanja o funkciji i ponašanju, u danoj domeni, te shvatiti, zapisati i analizirati relacije između njih. Kao dobra osnova za proučavanje znanja o funkciji i ponašanju proizvoda potrebno je stvoriti kvalitetne ontološke pretpostavke [94], [95].

U radu će se znanje o funkciji i ponašanju proizvoda uporabiti isključivo u svrhu odabira komponente proizvoda te u svrhu proučavanja kreiranog proizvoda. Znanje o funkciji i ponašanju se neće dodatno kategorizirati. Vrijednosti znanja o funkciji i ponašanju proizvoda odabirat će ili definirati konstruktor tijekom rada.

4 Informacijski modeli i metode razmjene podataka između CAD aplikacija

4.1 Informacijski modeli

Način prikaza informacija o proizvodu konstantno napreduje. Prije 1980 godine fizički prikaz proizvoda činio je i opis proizvoda. Razvoj računala te tehnika i tehnologija vezanih za računala dovelo je do pojave programa za pomoć u izradi tehničkih crteža tzv. CAD aplikacija. Kreiranje tehničkih crteža uporabom CAD aplikacija otvorilo je široke mogućnosti u domeni opisa proizvoda pomoću računala u odnosu na crtež na papiru. No, pojava računalnih informacija implicira detaljniju analizu modela informacija. U ovom dijelu rada pozornost će se usmjeriti na opis informacijskih modela i opis ISO 10303 standarda za oblikovanje i razmjenu informacija o proizvodu.

Razvoj informacijskih modela s pripadajućim aplikacijama većinom je vezan uz ne tehničku problematiku. Primjena informacijskih tehnologija u inženjerskim područjima zahtijeva drugačiji pristup zbog različite prirode informacija [96] koje se obrađuju. Priroda inženjerskih podataka uveliko je različita od podataka u poslovnim aplikacijama. Podatke u inženjerskim aplikacijama karakterizira potencijalno velika količina i vrlo složena povezanost između struktura podataka.

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. U idealnom slučaju, informacijski model bi trebao osigurati potpunu, točnu i jedinstvenu sliku pojave koja se modelira. Budući da osnovu informacijskih modela čine informacije i podaci potrebno ih je definirati. Prema [97] informacija je definirana kao znanje o idejama, činjenicama i/ili procesima. S druge strane podaci su simboli ili funkcije koje predstavljaju informacije za potrebe primjene, u skladu s implicitnim ili eksplicitnim pravilima. Može se reći da je svrha korištenja informacija, odnosno podataka, formulacija opisa stvarnog svijeta, na način da se te informacije mogu koristiti i razmjenjivati bez znanja o izvoru informacija, tj. bez potrebe za kreiranjem informacijskog modela.

Općenito gledano, model nastaje preslikavanjem prostora promatranja u prostor prikaza. Ukoliko se navedeno primijeni na proizvod može se reći da informacijski model nastaje

preslikavanjem elemenata proizvoda u odgovarajuće elemente informacijskog modela. Tada se informacijski model proizvoda sastoji od skupa opisa i pravila koja prikazuju elemente proizvoda.

Najvažniji element informacijskog modela je podatak. Podatak predstavlja simbolički prikaz elemenata stvarnog svijeta, kojim se opisuje atribut, koji je bitan za informacijski sustav, objekte ili događaje u stvarnom svijetu. Polazište za definiranje strukture svih informacijskih sustava može se pronaći u ANSI/SPARC[24] arhitekturi [97], [98], [99]. Prema [100] ANSI/SPARC arhitektura može se podijeliti na tri razine te pripadajuće modele podataka:

- konceptualna ili logička razina (konceptualni model podataka),
- vanjska ili korisnička razina (logički ili implementacijski model podataka,
- unutarnja ili fizička razina (fizički model podataka).

Konceptualna razina je potpuni prikaz konceptualnog ili logičkog modela podataka. Ovakav prikaz se može smatrati stvarnim stanjem bez programskih ograničenja no istovremeno može bitno odstupati od onoga što vidi korisnik. Konceptualni model odražava inženjersku stranu percepcije informacija i predstavlja rješenje definirano razumljivim terminima na razini inženjerske primjene. Konceptualni model podataka je cjelovit, konzistentan i neredundantan opis podataka informacijskog sustava koji se modelira. Integrirajući pojedinačne poglede korisnika na podatke informacijskog sustava, konceptualni model opisuje strukturu podataka čitavog informacijskog sustava. Ovaj model podataka neovisan je o implementaciji, kako logičkoj (sustavu za upravljanje) tako i fizičkoj (bazi podataka).

Prilagodbom konceptualnog modela određenom sustavu za upravljanje i manipulaciju bazom podataka nastaje logički model. Više logičkih modela može podržavati jedan konceptualni model. Logički model sastoji se od struktura i operacija nužnih za definiranje organizacije i pristupa podacima. Za opis logičkog modela koristi se više modela podataka: relacijski, mrežni, hijerarhijski, objektni ili datotečni model.

Korisnička razina predstavlja opis dijela logičkog modela podataka, na način primjeren korisniku. Na ovoj razini se opis konceptualnog modela preslikava u formu prilagođenu uporabi odnosno aplikaciji. Za svakog korisnika, ovisno o potrebi, moguće je definirati korisničku razinu.

Unutarnja ili fizička razina sukladna je s konceptualnom razinom i predstavlja opis fizičkih podataka. Na ovoj razini definira se fizički model zapisa podataka. Fizički model nastaje implementacijom logičkog modela odnosno strukture baze podataka. Ovo je najniža razina i predstavlja opis stvarne fizičke organizacije baze podataka koji mora biti sukladan opisu na konceptualnoj razini. Fizička zapis ovisi o korištenom sustavu za upravljanje i manipulaciju bazom podatka.

Informacijske modele možemo podijeliti u modele specifične namjene i općenite modele namjene. Modeli specifične namjene određuju konceptualni model koji je usmjeren na određeno područje primjene. Primjer takvih modela su TAXIS [101], SDM [102], SAM* [103] i O_2 [104]. Za razliku od modela specifične namjene, pristup općenito primjenjivih modela realizira se preko skupa specifičnih konceptualnih modela. Prvi općeniti model bio je hijerarhijski i mrežni model [105], a nakon njega slijedili su relacijski [106] modeli NIAM [107] i familija IDEF[108] modela. Predstavnici modela koji dijele karakteristike obiju koncepcija su EDM [109] i

[24] ANSI/SPARC – American National Standard Institute/Standard Planning and Requirements Comitee

EXPRESS [97]. Budući da je za razumijevanje STEP zapisa neophodno poznavanje EXPRESS-a, od navedenih informacijskih modela pozornost će se usmjeriti na ovaj jezik.

EXPRESS je razvijen kao jezik za formalno opisivanje informacijskih modela. Počeci razvoja jezika datiraju iz 1982. godine kada se u okviru PDDI[25] projekta razvio DSL[26] jezik. Na osnovu kojega se 1986. godine razvio EXPRESS. EXPRESS je dio ISO standarda (ISO 10303-11) u okviru kojega se nalaze i specifikacije potrebne za uporabu jezika. EXPRESS je formalni jezik za opisivanje informacijskih modela, a ne implementacijski jezik kao npr. C, te se na osnovu zapisa ne može kreirati izvršni kod. Jezik karakterizira nekoliko osobina koje ga izdvajaju od drugih sličnih jezika (UML[27], OMT[28], ...):

- 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 postao 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 što ga čini vrlo pragmatičnim. Tijekom izrade pojedinih verzija EXPRESS je poprimio neke mogućnosti drugih programskih jezika (C, C++, ADA, Algol, Euler, Modula-2, Pascal, PL/I, SQL) kao što su objektno orijentirani pristup i bogata kolekcija tipova podataka.

Osnovi element EXPRESS-a je shema. Shema sadržava 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** koje 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** koje 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.
- **Pravila ispravnosti** su važna komponenta prilikom definiranja entiteta ili tipova podataka. Pravila ispravnosti ograničavaju relacije između instanci ili određuju područje vrijednosti unutar kojega se mora nalaziti vrijednost definiranog tipa. Pravila mogu biti lokalna ili globalna. Lokalna pravila primjenjuju se na sve instance određenog tipa entiteta a predstavljaju tvrdnje, definirane u kontekstu deklaracije entiteta, o ispravnosti pojave entiteta. Za neku pojavu deklariranog tipa podataka, pravilo ima jedno od tri moguća stanja: potvrđeno, prekršeno ili nepoznato. U skladu

[25] Product Data Definition Interface
[26] DSL je jezika za definiranje podataka.
[27] UML – Unified Modeling Language [110]
[28] OMT – Object Modeling Technique [111]

s pravilima jezika prekršena stanja nisu dozvoljena tako da stanja lokalnih pravila mogu poprimiti vrijednosti potvrđena ili nepoznata. Pravila mogu pripadati jednom od entiteta u relaciji ili entitet u relaciji može sadržavati pravilo. Postoje dodatne dvije vrste lokalnih pravila:

- ograničenje jedinstvenosti – kontrolira jedinstvenost vrijednosti atributa među instancama danog tipa entiteta, i
- ograničenje domene – opisuje druge restrikcije među vrijednostima atributa za svaku instancu danog tipa entiteta.

Globalna pravila važe za čitavu informacijsku bazu.

- **Definicije algoritama** predstavljaju definicije funkcija ili procedura koje služe za definiranje ograničenja.

Atributi se mogu prikazati pomoću jednostavnog tipa podataka (npr. integer) ili preko drugog 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.

Jednostavni tipovi podataka su:

- *number* – vrijednost koja može biti cjelobrojna ili realna,
- *real* – realna vrijednost,
- *integer* – cjelobrojna vrijednost,
- *string* – niz znakova,
- *Boolean* – logička vrijednost (TRUE, FALSE),
- *logical* – logičke vrijednosti (TRUE, UNKNOWN, FALSE),
- *binary* – binarne vrijednosti (0,1).

Složeni tipovi podataka predstavljaju uređene i neuređene skupine elemenata nekog osnovnog tipa. U ovu skupinu spadaju:

- *array* – predstavlja polje vrijednosti stog tipa,
- *bag* – čini skupina elemenata, proizvoljno poredanih, kojima je dozvoljeno ponavljanje,
- *list* – predstavlja slijed pojmova,
- *set* – čini skup elemenata, proizvoljno poredanih, kojima nije dozvoljeno ponavljanje.

Imenovani tipovi podataka mogu se dodatno podijeliti na: entitete (predstavljaju objekte kojima se barata u jeziku i pomoću kojih se opisuje domena), svojstva entiteta (opisuju se atributima uz mogućnost da i sam entitet može postati atribut drugog entiteta), i na definirane tipove (koji predstavljaju proširenje skupa standardnih tipova podataka).

Konstrukcijski tipovi podataka predstavlja skupinu podataka koju čine:

- *enumeration* – enumeracijski tip koji čine uređeni skupovi vrijednosti prikazani imenima,
- *select* – odabrani tipovi koje čine imenovane skupine drugih tipova.

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.

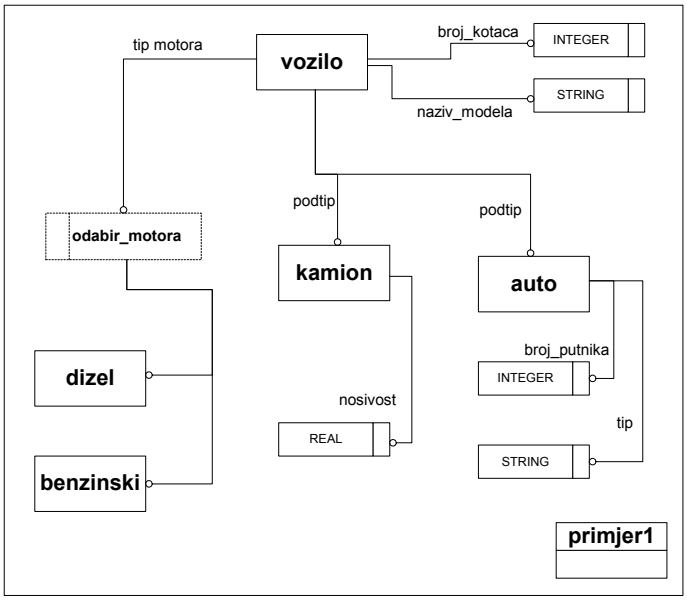
U EXPRESS-u mehanizam za određivanje klasifikacije je ostvaren strukturom *podređeni*/*nadređeni*[29]. Podređeni mogu biti povezani na različite načine ovisno o zadanom operatoru. Dozvoljeni operatori za određivanje međuodnosa između podređenih su:

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

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

Kao što je navedeno informacijski model u EXPRESS-u sastoji se od shema koje čine 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. Jezik je samo specifikacije i nije izvršan. Na slici 4-1 dan je primjer zapisa entiteta u EXPRESS-u i grafički prikaz u EXPRESS-G[30]. EXPRESS se koristi za definiranje modela podataka u STEP-u.

```
SCHEMA primjer1;

ENTITY vozilo
SUPERTYPE OF (ONEOF (auto, kamion));
  broj_kotaca : INTEGER;
  motor: odabir_motora;
END_ENTITY;

ENTITY auto
SUBTYPE OF (vozilo);
  naziv_modela : STRING;
END_ENTITY;

ENTITY kamion
SUBTYPE OF (vozilo);
  nosivost : REAL;
END_ENTITY;

TYPE odabir_motora = SELECT
(dizel, benzinski);
END_TYPE;

END_SCHEMA;
```



Slika 4-1 Primjer zapisa u EXPRESS-u i EXPRESS-G

[29] subtype/supertype struktura
[30] EXPRESS-G je grafički jezik odnosno način grafičke prezentacije EXPRESS strukture

4.2 STEP

Standard za razmjenu informacija o proizvodu STEP[31] definira specifikacije za prikazivanje i razmjenu informacija o proizvodu. Pod nazivom STEP krije se neformalno ime za ISO 10303[32] industrijski standard kojim je opisana problematika prikazivanja i razmjene informacija o proizvodu. STEP je međunarodno priznati standard koji omogućuje jednoznačan prikaz i mehanizam razmjene informacija, između računalnih aplikacija, tijekom životnog vijeka proizvoda. Za razmjenu informacija između različitih računalnih sustava osim STEP-a koriste se i sljedeći standardi [112], [113], [114]:

- IGES[33] – je prvo razvijeno rješenje problema razmijene podataka između različitih CAD sustava, ovaj standard omogućuje razmjenu samo geometrijskih informacija,
- DXF[34] – omogućuje razmjenu opisa geometrije dijela modeliranog jednostavnim značajkama (DXF nije "pravi" standard, ali je u širokoj uporabi, pa je postao de facto standard),
- VDA-FS[35] – ovaj standard je orijentiran na razmjenu informacije o slobodnim površinama ("free form surfaces") i slobodnim krivuljama ("free form curves") specifičnim za automobilsku industriju, razvijen je na temeljima IGES standarda,
- SET[36] – standard je razvijen kao zamjena za IGES u avio i automobilskoj industriji u Francuskoj, standard sadrži detaljnu specifikaciju podataka za domenu strojarstva i dodatne informacije o strukturi podataka i uporabljenim konceptima.

Razvoj STEP standarda započeo je 1983. od strane međunarodne organizacije za standardizaciju ISO (TC184 "Industrial Automation Systems and Integration", SC4 "Industrial Data") sa sljedećim ciljevima [112]:

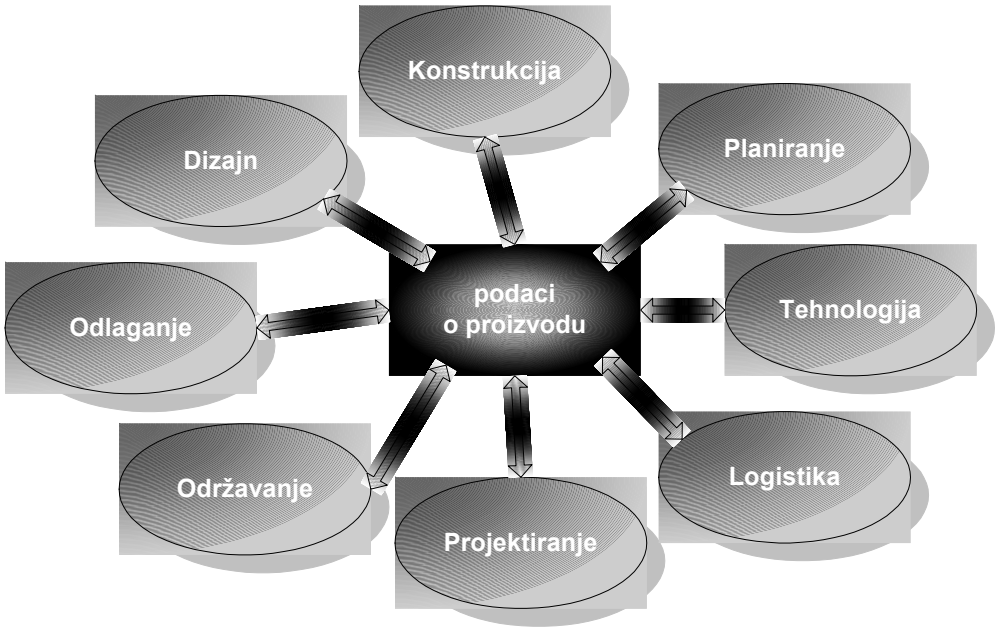
- standardizacija mehanizama za modeliranje informacija o proizvodu, neovisno o sustavu koji se primjenjuje, kroz cijeli životni vijek,
- stvaranje jedinstvenog međunarodnog standarda koji će pokriti sve aspekte razmjene podataka u CAD/CAM domeni,
- odjeljivanje opisa proizvoda od primjene stvara osnovu za kreiranje baze podataka i kreiranje neutralnog formata zapisa,
- uporaba STEP standarda u industriji umjesto nacionalnih.

STEP je standard koji se konstantno razvija i dograđuje, a sastoji se od niza standarda za specifikaciju i definiciju informacija o proizvodu. U sadašnjem stupnju razvoja STEP omogućuje razmjenu podataka između CAD/CAM sustava pomoću datoteka i pomoću standardnih aplikacijskih sučelja (API). Cilj prema kome se teži u razvoju STEP-a je podrška razmjeni informacija o proizvodu tijekom cijelog životnog vijeka (Slika 4-2).

Budući da se želi ostvariti praćenje podataka o proizvodu tijekom njegovog čitavog životnog vijeka, STEP more obuhvaćati podatke o geometriji, topologiji, tolerancijama, relacijama, atributima, sklopovima te konfiguracijama proizvoda. Ostvarenje navedenoga omogućeno je podjelom STEP-a u više dijelova. Dijelovi STEP-a obuhvaćaju područja od

[31] STEP – **ST**andard for the **E**xchange of **P**roduct Model Data
[32] ISO 10303 je standard definiran kao "Industrial automation system – Product data representation and exchange"
[33] Interim Graphics Exchange Specification
[34] Data Exchange Format
[35] Verband der Automobilindustrie-Flachen-Schnittstelle
[36] Standard d'Echange et de Transfert

zajedničkog interesa (kao što su procedure za testiranje), formate datoteka, sučelja programa, kao i dijelove specifične za pojedine industrijske grane.



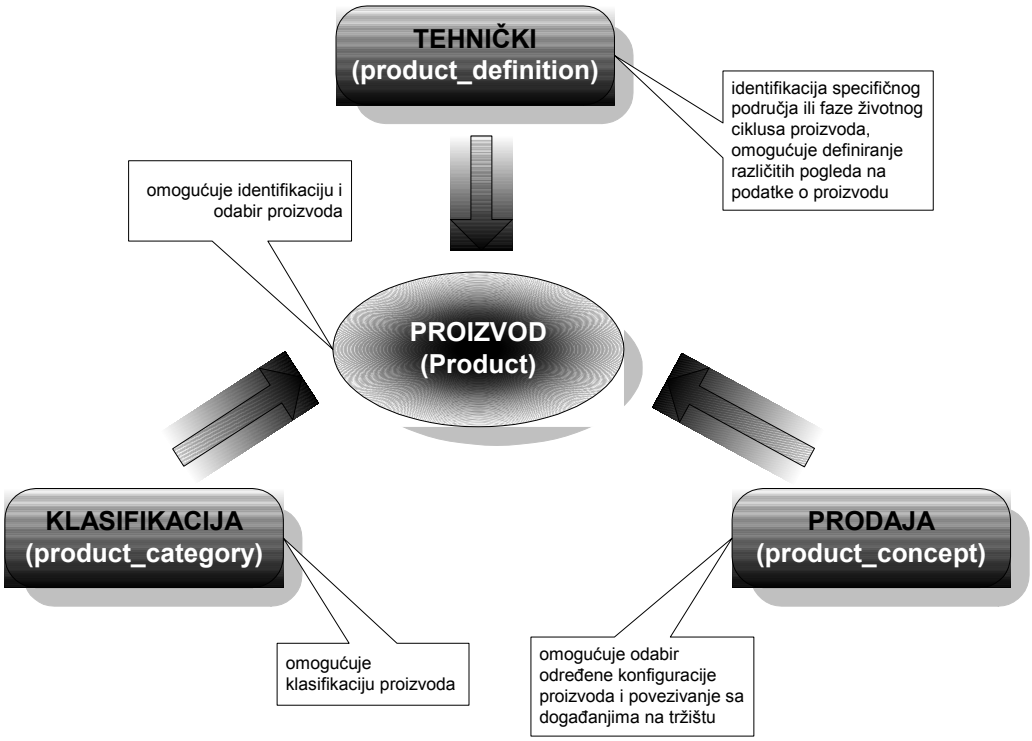
Slika 4-2 Podaci o proizvodu

Eksperti iz pojedinih industrijskih grana pomoću EXPRESS-a opisuju skupove informacija potrebne za opisivanje proizvoda iz određene industrijske grane ili područja. Ove definicije tvore aplikacijske protokole, koji čine pretežni dio STEP standarda, a koriste se kao osnova za razmjenu informacija o proizvodu.

Standard je podijeljen u nekoliko područja (Slika 4-4) odnosno kategorija:

- **Metode opisa** (obuhvaćaju dijelove: #1, #11, #12 i dijelove: #13 i #14 koji su u pripremnoj fazi). Metode opisa su prva značajna komponenta strukture STEP-a [115]. Metode opisa određuju zajednički mehanizmi za definiranje strukture podataka STEP-a te definiraju jezik za specifikaciju formalnog opisa podataka (EXPRESS).
- I**ntegrirani resursi** (uključuje dijelove: #41 do #50 uz iznimku #48 – značajke oblika, čiji razvoj je obustavljen). Integrirani resursi su osnovni semantički elementi koji se koriste za opis proizvoda te osiguravaju seriju općenitih EXPRESS modela od kojih se grade aplikacijski protokoli. Iako se koriste kao osnova za kreiranje aplikacijskih protokola nisu namijenjeni direktnoj implementaciji. Detaljna struktura i semantika integriranih resursa temeljena je na ugrađenoj strukturi podataka tj. strukturi na višoj razini na osnovu koje se kreira i održava sintaktička i semantička konzistentnost modela. Integrirani resursi su podijeljeni na dvije skupine, koje imaju sličan oblik i formu:
  - integrirani generički resursi (dijelovi 41 do 49) definiraju komponente konceptualnog modela podataka koje su neovisne o primjeni, najbitnije generičke resurse predstavljaju dijelovi 41 i 42 koji definiraju geometrijski i topološki prikaz te okvir za opis proizvoda,
  - integrirani aplikacijski resursi (dijelovi 101 do 199) proširuju generičke resurse zbog ostvarivanja podrške zahtjevima specifičnih aplikacija.

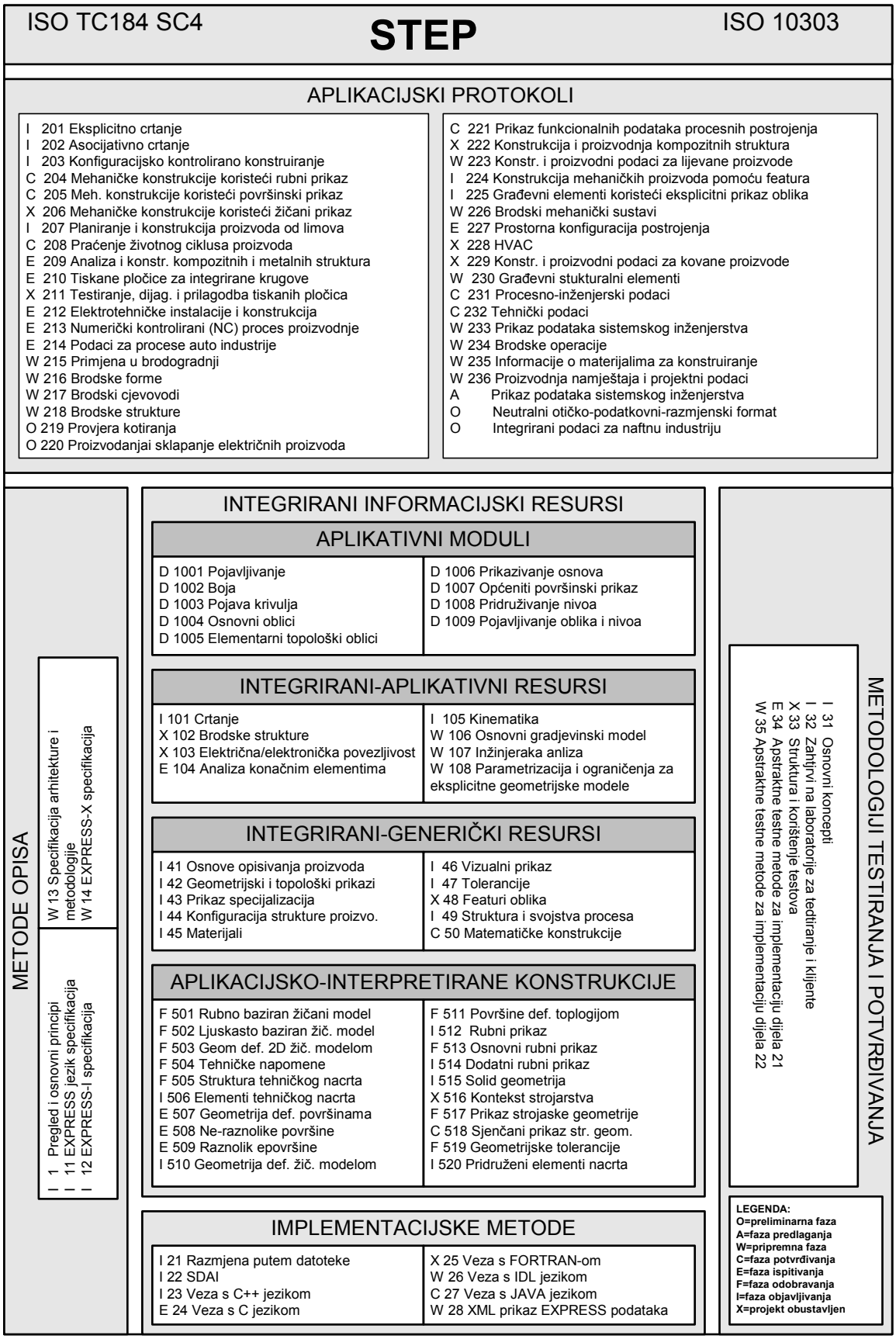
- **Aplikacijski protokoli** (uključuje dijelove: #201, #202, #203, ...). Aplikacijski protokoli sadrže informacijske modele zapisane u EXPRESS-u koji zadovoljavaju specifične zahtjeve, za opis proizvoda, definirane aplikacijskim kontekstom, te predstavljaju dijelove STEP-a koji se mogu implementirati. 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 standarde prikaza geometrije, aplikacijski protokoli definiraju uporabu tih prikaza. Aplikacijski protokoli mogu se implementirati uporabom jedne od implementacijskih metoda.
- **Metode implementacije** (obuhvaćaju dijelove od #21 – razmjena putem datoteka, do #28 – XML prikaz EXPRESS podataka). Metode implementacije predstavljaju standardne tehnike implementacije aplikacijskih protokola. Svaka metoda implementacije određuje način na koji se struktura podataka definirana STEP-om može preslikati na određenu metodu implementacije. Ovo područje uključuje strukturu razmjene podataka uporabom datoteka (dio #21), standard sučelja za pristup podacima "SDAI"[37] (dio 22) te veze prema programskim jezicima (dijelovi #23 (C++), #24 (C), #25 (Fortran), #26 (IDL), #27 (JAVA) i #28 (XML)).
- **Metode za testiranje i potvrđivanje** (obuhvaćaju dijelove: #31 do #35 uz iznimku #33 – struktura i korištenje testova, čiji razvoj je obustavljen) opisuju metodologiju testiranja i implementacije različitih dijelova STEP-a.



Slika 4-3 Pogledi na proizvodu sa različitih stajališta

Osnovni aspekt STEP strukture podataka je podjela na tri različita pogleda ili potencijalne uporabe podataka o proizvodu (Slika 4-3). Ovi pogledi opisuju na koji način je proizvod: klasificiran ili kategoriziran, prikazan na tržištu, te tehnički opisan u svrhu dizajna, konstrukcije, izrade, ili održavanja.

[37] SDAI - Standard Data Access Interface



Slika 4-4 Struktura STEP standarda

Dijelovi infrastrukture standarda, koju čine metode opisa i metode implementacije, odvojeni su od dijelova ovisnih od industrijskoj grani (aplikacijskih protokola). Veći dio infrastrukture standarda je završen dok su aplikacijski protokoli potpuno otvorena za daljnju dogradnju. Trenutno su aplikacijski protokoli definirani za strojarske i električne proizvode, a u razvoju su aplikacijski protokoli za definiranje kompozitnih materijala, savijanja limova, automobilsku industriju, proces izrade ("manufacturing"), brodograđevnu industriju, itd. Očekuje se tijekom vremena definiranje aplikacijskih protokola za sve grane industrije.

Svaki aplikacijski protokol podijeljen je u četiri glavna povezana dijela:

- AAM[38] opsuje područje aplikacijskog protokola koji određuje procese i podatke koje protokol podržava, AAM se dokumentira uporabom IDEF0 dijagrama,
- ARM[39] opisuje definiciju informacijskih zahtjeva protokola, definira aplikacijske objekte te definira odnose i ograničenja koja se primjenjuju na podacima, ovaj dio protokola opisuje se uporabom EXPRESS-G, NIAM ili IDEF1X dijagrama,
- AIM[40] je konceptualna shema koja opisuje ARM u terminima biblioteka i postojećih definicija; AIM se uvijek opisuje uporabom EXPRESS-a; u okviru AIM-a dozvoljeno je dodatno definiranje entiteta, koji postoje u integriranim resursima, ali nije dozvoljeno definiranje novih.

Razvoj parametarskih tehnika u domeni izrade CAD modela proizvoda potakao je početak proučavanja implementacije parametara u STEP. Grupa stručnjaka unutar NIST-a[41] prihvatila je dva glavna pravca razvoja. Prvi pravac je kratkoročan, orijentiran na premošćivanje razlika između STEP standarda i parametarskih tehnika dostupnih u današnjim CAD aplikacijama. Pristup koji se koristiti u ovom pravcu određuje pogled na parametarske mogućnosti STEP-a kao dodatne, a ne kao sastavni dio samog standarda ili zamjena za isti [116]. Drugi pravac dugoročno je orijentiran na proučavanje i predviđanje razvoja u domeni CAD tehnologija. Interes u ovom pravcu fokusirati će se na uključivanje parametrike u područje modeliranja ne-geometrijskim oblicima, snimanju procesa razvoja proizvoda, te mogućnostima uključivanja konstrukcijskog znanja. U dokumentu ISO 14959-1 naglašava se potreba za definiranjem dinamičkih entiteta, akcija i metoda. Početci razvoja koji idu u navedenom smjeru vide se u kreiranju EXPRESS-2 jezika čije mogućnosti su opisane u dokumentu ISO 10303-11 iz 1998 godine.

U radu će se koristiti aplikacijski protokol broj 203, koji je u potpunosti usvojen, i koji definira shemu informacijskog modela kao osnove za implementaciju u PDM[42] sustavima. AP 203 ("configuration – controlled designs") sadrži 36 aplikacijskih objekata podijeljenih u devet jedinica funkcionalnosti ("UOF"[43]): "Authorization", "Bill of Material", "Design Information", "Design Activity Control", "Effectivity", "End Item Identification", "Part Identification", " Shape" i "Source Control". U okviru svake jedinice funkcionalnosti definirani su aplikacijski objekti koji određuju informacije potrebne za opisivanje danog proizvoda.

[38] AAM – Application Activity Model
[39] ARM – Application Reference Model
[40] AIM – Application Interpreted Model
[41] NIST – National Institute of Standards and Technology
[42] PDM – Product Data Management
[43] UOF – Unit Of Functionality – opisuju, logički kompletan, podskup informacija o danom proizvodu.

5 Objektno modeliranje

Razvoj objektno orijentiranih tehnika motiviran je potrebom razumijevanja kompleksnih programskih sustava. U početcima razvoj je bio usmjeren prvenstveno na uporabu pri razvoju računalnih programa [117]. U zadnjih nekoliko godina pozornost je usmjerena i na proučavanje primjene objektnog pristupa i na probleme analize i modeliranja pojava i elemenata iz stvarnog svijeta, rada sa bazama podataka [118], te na problematiku vezanu za proces konstruiranja [119]. Objektno orijentirani pristup procesu konstruiranja i prikazu proizvoda [11] omogućio je prirodnu dekompoziciju i hijerarhijski prikaz proizvoda i dijelova proizvoda [120].

Booch [13] definira objekt na sljedeći način: objekt posjeduje stanje, ponašanje i identitet; struktura i ponašanje sličnih objekata definirana je u zajedničkoj klasi; termini instanca i objekt su međusobno izmjenjivi. Klasa se može definirati kao:

- opis koji se može primijeniti na svaki objekt iz skupine objekata,
- način grupiranja objekata baziran na osnovu zajedničkih karakteristika,
- generalizacija objekata,
- skup objekata koji dijele zajedničke konceptualne osnove,
- opis ponašanja podređenih apstraktnih tipova podataka pomoću definicije sučelja prema operacijama koje se mogu provesti nad danim tipom podataka.

Objekt je jedinstvena instanca određene klase. Npr. *ovalni kotao*[44] se može smatrati objektom klase *kotao*. Naziv *ovalni kotao* odnosi se na određeni objekt dok se naziv *kotao* odnosi na klasu objekta. Standardni *vijak M8x1.5* može se smatrati objektom iz klase *vijaka*. Osobine koje karakteriziraju objekt *vijak M8x1.5* i objekt *ovalni kotao* se razlikuju.

Razlikovanje objekta ovalni kotao od drugih objekta iste klase nije dostatno samo na razini naziva objekta, već objekt mora posjedovati identitet na osnovu kojega se može razlikovati od drugih objekata iste klase. Objekt *ovalni kotao* posjeduje stanja u kojima se može nalaziti (koncept, projekt, razrada, izrada, ...). Iako objekt *ovalni kotao* može biti samo u jednom stanju, drugi objekti iste klase mogu biti u nekom drugom stanju. Što znači da stanje objekta uključuje ne samo statičke već i dinamičke osobine objekta. Ponašanje objekta se odnosi na

[44] Kotao je dio energetskog transformatora u koji se smješta jezgra transformatora.

moguće aktivnosti objekta. Npr. kako se *ovalni kotao* ponaša ukoliko se promjeni debljina lima ili svjetla širina otvora.

Objektno orijentirana paradigma uključuje nekoliko osnovnih koncepata koji se koriste za definiranje i razvoj uporabom objektno orijentiranog pristupa [121]:

- Objekti:

Objekti modeliraju entitete iz stvarnog svijeta, mogu obuhvaćati apstakcije kompleksnih fenomena ili mogu prikazivati elemente programskog sustava. Operacijski gledano, objekti kontroliraju računalni proces. Pravilan odabir i definiranje objekta predstavlja osnovu ispravnog pristupa uporabi objektno orijentiranog modeliranja. Objekte možemo definirati na sljedeći način:

- objekt predstavlja skupinu podataka i operacija koje se mogu izvršiti na danom tipu podataka,
- objekt je deklarirana varijabla određene klase,
- objekt karakteriziraju vlastiti skupovi atributa i operacija koje može izvršiti.

Osnovi elementi objekta su atributi i metode. Atributi su informacijski detalji svojstveni objektu. Ovisno o konkretnom programskom jeziku atributi se nazivaju i varijablama ili svojstvenim poljima. Metoda je funkcionalni detalj svojstven objektu i pohranjen kao dio objekta. Za metode se koriste i nazivi svojstvena funkcija ili operacija.

- Klase i hijerarhija klasa:

Klasa služi kao predložak za kreiranje objekata. Primarna funkcija klase je opisivanje objekata koji dijele isto ponašanje i strukturu. Klase se mogu podijeliti u podklase pri čemu je svaka podklase derivirana iz nadređene klase ili superklase, a razlikuje se po definiciji dodatnih vlastitih atributa i operacija (Slika 5-1). Podjela klase u podklase određuje hijerarhijsku strukturu klasa.

- Apstrakcija:

Apstrakcija je način fokusiranja na bitne karakteristike objekta. Definiranjem objekata na osnovu interakcije prema drugim objektima, ne ulazeći u unutarnje detalje objekta, možemo jednostavnije kreirati sučelja objekta. Apstrakcijom se određuju bitne karakteristike objekta, koje ga razlikuju od drugih objekata, i na taj način se omogućuje definiranje konceptualnih granica objekta relativno u odnosu na promatrača.

- Čahurenje:

Čahurenje je proces koji je usko povezan sa problematikom apstrakcije objekta. Za razliku od apstrakcije, koja omogućava kreiranje različitih okvira objekta, čahurenje je tehnika skrivanje implementacije objekta. Ova tehnika omogućuje odvajanje implementacije od sučelja objekta. Razdvajanje implementacije od sučelja objekta omogućava promjenu implementacije bez upliva na sučelje te osigurava zaštićenost implementacije od uporabe objekta. Čahurenje promovira modularnost tj. objekti se moraju promatrati kao blokovi za gradnju kompleksnog sustava.

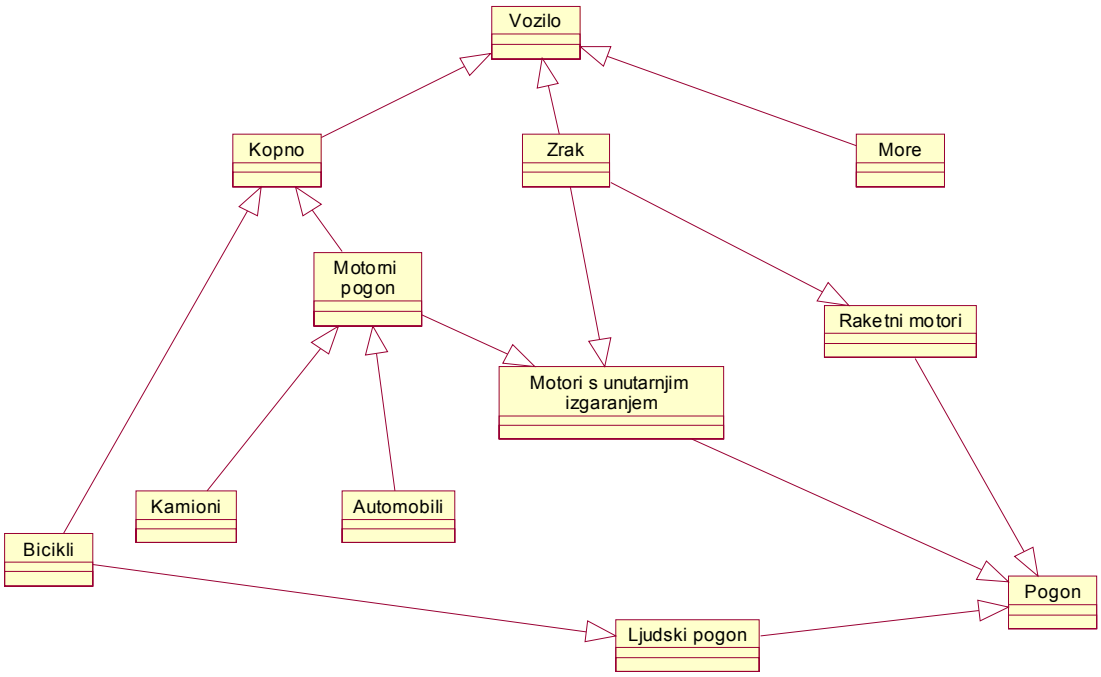
- Nasljeđivanje:

Nasljeđivanje je mehanizam stjecanja karakteristika klasa kojima objekt pripada. Usporedba klasa sa konvencionalnim načinima kategorizacije dovodi do izražaja pojam hijerarhije klasa. Podklase su zapravo specijalizacija i/ili proširenje postojećih klasa uz nasljeđivanje njihovih svojstava. Pojedine klase mogu naslijediti osobine više klasa, kada

do izražaja dolazi kvaliteta uporabljenog koncepta prilikom kreiranja hijerarhijske strukture klasa. Naime, prilikom nasljeđivanja osobina iz različitih klasa mogu se pojaviti kontradiktorni uvjeti npr. klasa nasljeđuje iz jedne superklase metodu **zapisi ()** koja vraća cjelobrojne vrijednosti, dok istovremeno iz druge superklase nasljeđuje metodu istog naziva, ali u ovom slučaju metoda vraća znakovni tip podataka.

- Modularnost:

Modularnost je osobina više komponentnih sustava u kojima su komponente neovisne ali međusobno kompatibilne. Cilj modularnosti je dekompozicija sustava u manje module koji su slabo spregnuti tako da su unutarnje promjene u modulu lokalizirana samo na dotični modul. Moduli moraju međusobno komunicirati u cilju postizanja rješenja zadanog problema.



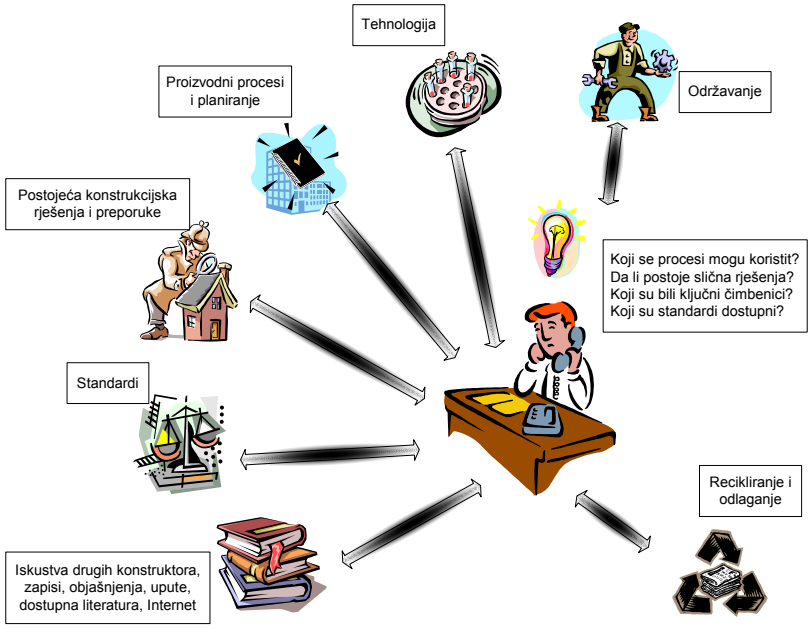
Slika 5-1 Primjer hijerarhijske strukture i višestrukog nasljeđivanja
primjer je kreiran uporabom UML-a[45] [110]

[45] UML – Unified Modeling Language je jezika za vizualiziranje, specificiranje, konstruiranje i dokumentiranje rezultata procesa razvoja računalnih programa kao i za modeliranje poslovnih sustava [110].

6 Konceptualni model konstrukcijskog znanja

6.1 Struktura znanja u procesu konstruiranja

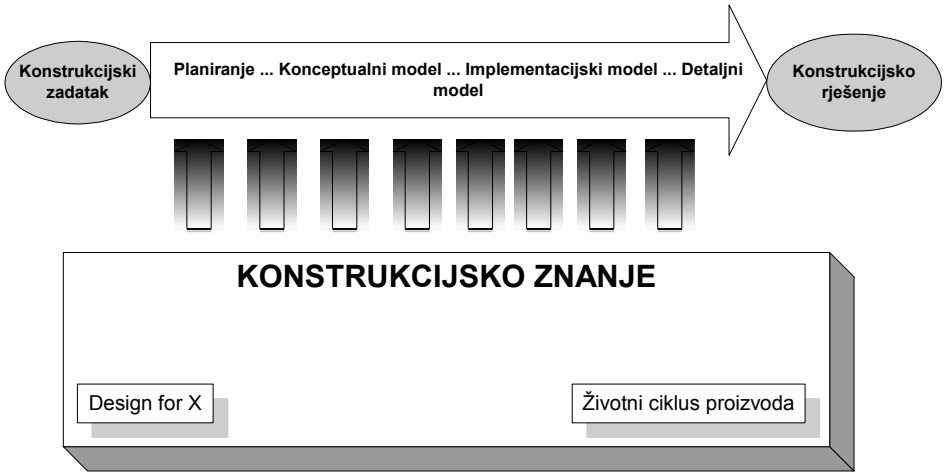
Priroda procesa konstruiranja te složenost i raznolikost znanja (Slika 6-1) potrebnog prilikom konstruiranja proizvoda zahtijeva sustav prikazivanja i obrade znanja koji će biti što je moguće fleksibilniji i robusniji [122]. Prikupljanje i obrada znanja potrebnog u procesu konstruiranja proizvoda (Slika 6-2) jedan je od važnijih čimbenika procesa konstruiranja. Odluke koje konstruktor donosi tijekom procesa konstruiranja i uptrebljena konstrukcijska rješenja ovise o kvaliteti znanja dostupnog konstruktoru. Prilikom donošenja odluka od vitalne je važnosti da su konstruktoru informacije (znanje) dostupne: u pravo vrijeme, u potrebnoj količini i u ispravnom obliku.



Slika 6-1 Kompleksnost i raznolikost konstrukcijskog znanja

Tehnička dokumentacija, koja je, još uvijek, najrašireniji oblik geometrijskog prikaza proizvoda, nije pogodna kao osnova za razmjenu informacija i znanja o proizvodu između CAD

programskih aplikacija. Razlog je i što većina CAD programskih aplikacija ne posjeduje mogućnosti vezivanja kompleksnijih oblika znanja za prikaz proizvoda niti mogućnost praćenja nastanka proizvoda (praćenje konstrukcijskih odluka, razlozi, objašnjenja, itd.).



Slika 6-2 Prikaz konstrukcijskog procesa prema [4]

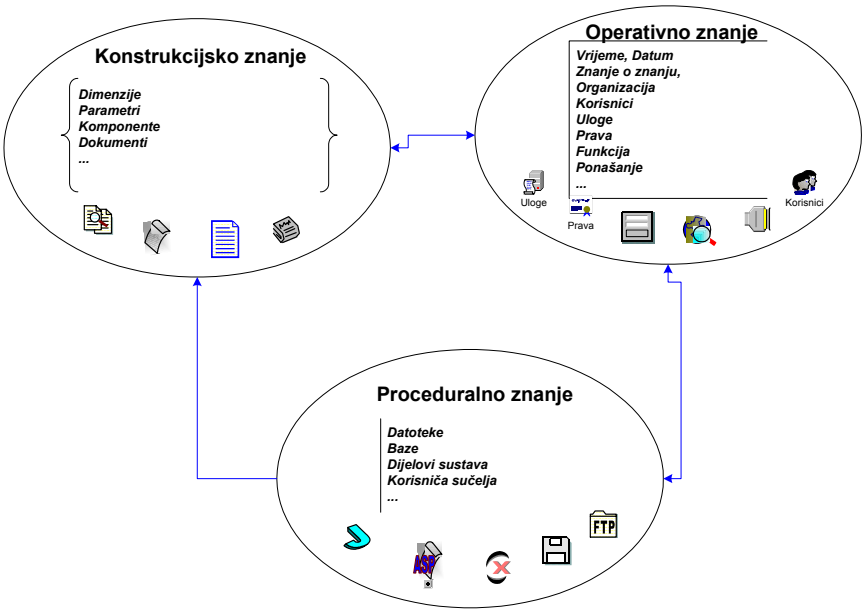
Složeniji tehnički crteži kreirani po zakonitostima tehničkog crtanja i u skladu sa usvojenim standardima mogu biti neformalni, nerazumljivi i nepotpuni tako da njihovo razumijevanje ovisi o vještini i iskustvu inženjera.

Jedan od ciljeva sustava za pomoć konstruktoru u procesu konstruiranja je i formalno prikazivanje i integracija znanja uporabljenog i generiranog tijekom procesa konstruiranja proizvoda. Glavna razlika između sustava za pomoć u procesu konstruiranja temeljenih na geometrijskom prikazu i sustava temeljenih na uporabi znanja je u načinu realizacije pomoći. Sustavi za pomoć procesu konstruiranja temeljeni na geometrijskom prikazu omogućuju samo pasivnu potporu konstrukcijskom procesu, dok sustavi temeljeni na znanju, uz uporabu metoda umjetne inteligencije[46], aktivno podupiru proces konstruiranja. Aktivna potpora se očituje u činjenici da se metode umjetne inteligencije mogu ugraditi u sve faze procesa konstruiranja čime se omogućuju nadzor i pružanje pomoći (uporaba znanja) u većini konstruktorovim akcijama.

Proširenje mogućnosti trenutno dostupnih CAD sustava u svrhu pomoći inženjeru konstruktoru u procesu konstruiranja proizvoda bio bi značajan odmak u smjeru kreiranja kvalitetnijih i razumijevanja nastanka postojećih konstrukcijskih rješenja te unapređenja komunikacije između konstruktora. Navedeno proširenje zahtijeva kreiranje mehanizama za nadogradnju informacija o konstrukciji, koje su trenutno na razini prikaza geometrije, vezivanjem konstrukcijskog znanja kreiranog i uporabljenog u procesu konstruiranja proizvoda za geometrijski prikaz.

Temeljem analize znanja u procesu konstruiranja, opisanih u trećoj glavi rada, te istraživanja dokumentiranih u [15], [16], [17] i [18] predložena je podjela znanja u procesu konstruiranja (Slika 6-3).

[46] Pod metodama umjetne inteligencije ovdje se smatraju: metode zapisa znanja (Blackboard [123], [124], Rule Based[125]) i metode zaključivanja (Backward i Forward chaining [125])



Slika 6-3 Struktura znanja u fazi procesa konstruiranja

Znanje je podijeljeno prema ulozi u sustavu[47], za pomoć inženjeru konstruktoru u procesu konstruiranja proizvoda, u tri kategorije:

- konstrukcijsko znanje,
- operativno znanje,
- proceduralno znanje.

Konstrukcijsko znanje je najvažnija kategorija navedenih znanja. Ovo znanje je vezano za konstrukciju tj. sadrži informacije relevantne za proces konstruiranja proizvoda. Informacije relevantne za proces konstruiranja mogu se podijeliti ovisno o strukturi i obliku zapisa na geometrijske informacije (geometrijski prikaza proizvoda, CAD model), informacije o dokumentima koji se koriste pri konstruiranju (upute, standardi, preporuke) i informacije o programskim procedurama vezanih za pravila zaključivanja i vanjske programske aplikacije (proračune, simulacije, kontrole).

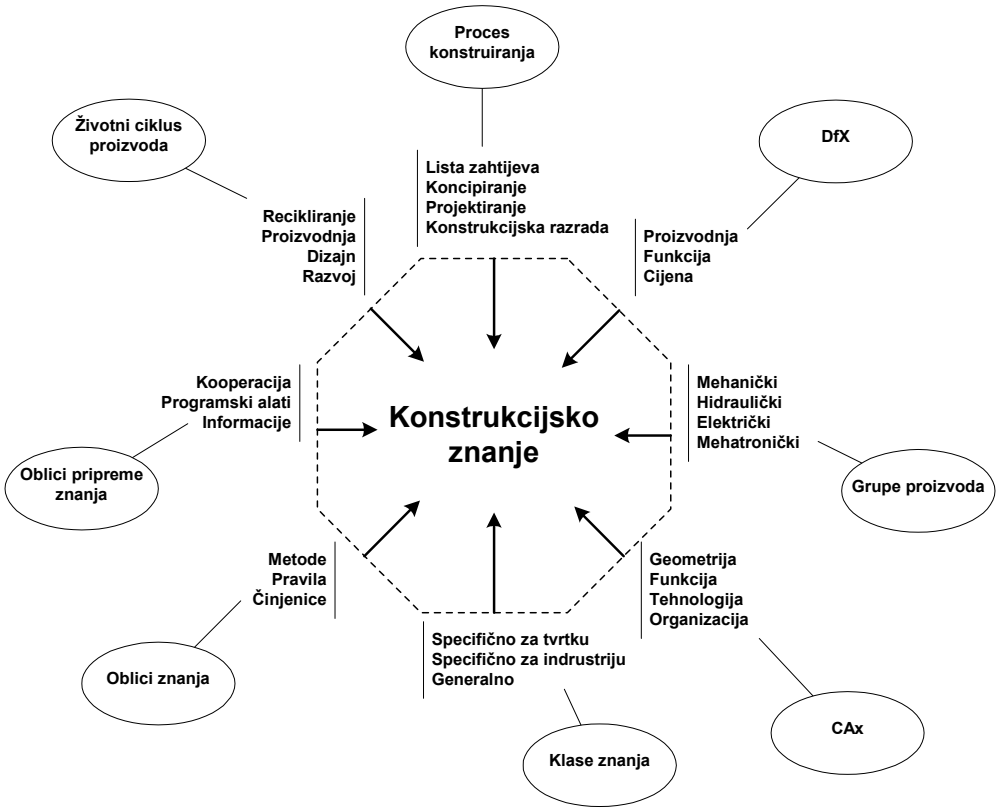
Operativno znanje vezano je s konstrukcijskim znanjem na taj način što dodaje informacije vezane za vrijeme i mjesto nastanka konstrukcijskog znanja. Također sadrži informacije o korisniku koji je kreirao konstrukcijsko znanje te informacije o organizaciji u kojoj se korisnik nalazi. Operativno znanje sadrži i informacije o ovlastima i ulogama pojedinih subjekata[48] u sustavu. Možda jedna od najvažnijih uloga operativnog znanja su informacije o znanju koje se nalazi u sustavu. Ovaj dio informacija, operativnog znanja, naziva se znanje o znanju[49] [126], [127] i sadrži, osim informacija o znanju kreiranom u sustavu, informacije o načinu uporabe pojedinog znanja, na koji način se mogu pretražiti postojeći zapisi znanja, na koji način se isti mogu promijeniti te što treba učiniti ukoliko se kreirano znanje želi obrisati.

[47] Pod sustavom se smatra programska aplikacija, koja je kreirana, i koja služi za manipulaciju, upravljanje i uporabu proširenog CAD modela u CAD aplikacijama.
[48] Pod subjektom se smatraju svi korisnici sustava, administrator sustava i administrator znanja
[49] U nekim radovima ovo znanje se naziva i meta-znanje

Proceduralno znanje je vezano za sustav upravljanja i manipulacije konstrukcijskim znanjem. Ovo znanje sadrži informacije o mjestu pohrane i načinu pristupa pojedinim datotekama neophodnim za inicijalizaciju ili ispravan rad sustava, zatim informacije o načinu interpretacija zapisa entiteta znanja te sve neophodne postupke i mehanizme za pohranu, promjenu i uporabu konstrukcijskog i operativnog znanja.

6.1.1 Model konstrukcijskog znanje

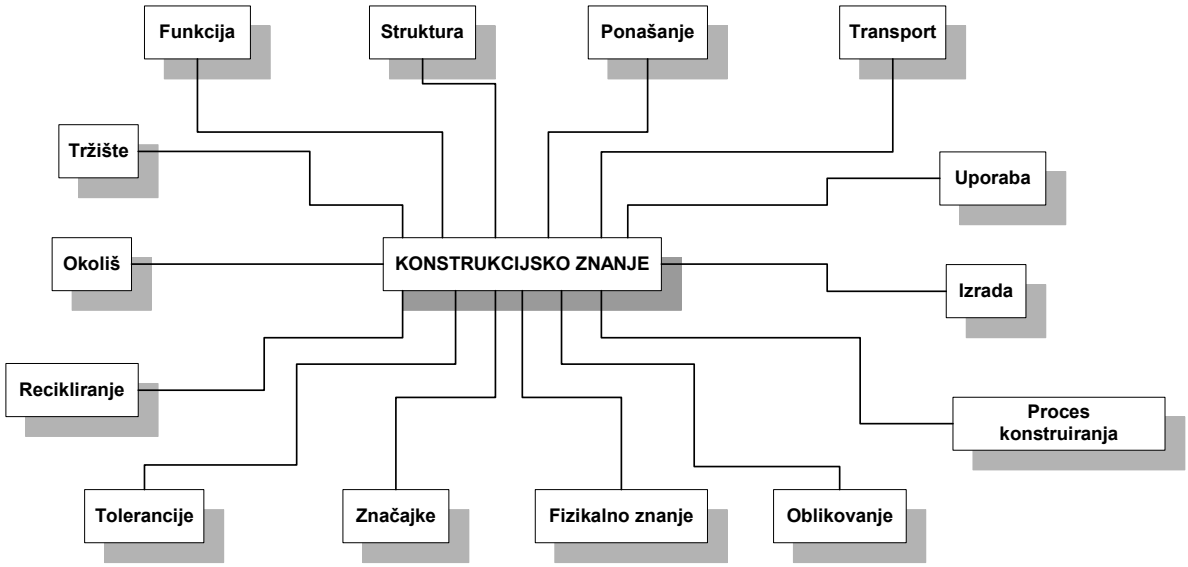
Tijekom procesa konstruiranja konstruktor koristi znanja iz različitih područja vezanih za životni ciklus proizvoda. Utjecaj znanja iz drugih područja vezanih za životni ciklus proizvoda, kao na primjer vizualni identitet, može biti zanemariv, ali isto tako može utjecati na ključne odluke u procesu konstruiranja i na smjer konstruiranja proizvoda (npr. utjecaj na okolinu). Konstrukcijsko znanje se može promatrati iz različitih područja ili faza vezanih za životni vijek proizvoda (Slika 6-4). Ovisno o fazi izrade proizvoda znanje o proizvodu pa tako i konstrukcijsko znanje može se strukturirati na različite načine. Detaljniji pregled i analiza različitih pristupa strukturiranju znanja dan je u trećem poglavlju.



Slika 6-4 Različiti pogledi na konstrukcijsko znanje [4]

Konstrukcijsko znanje, tj. znanje koje konstruktor koristi prilikom kreiranja konstrukcije, teško je odrediti već i zbog same činjenice što je proces konstruiranja multidisciplinarna djelatnost. Tijekom procesa konstruiranja proizvoda konstruktor mora voditi računa o raznim faktorima koji utječu na rezultat procesa konstruiranja. S jedne strane to su informacije koje se odnose na zadovoljenje liste zahtijeva, internih i vanjskih standarda, pravila i preporuke konstruiranja. S druge strane to su znanja koja se ne odnose direktno na proces konstruiranja, ali ga mogu ograničiti, npr. pakiranje, transport, vizualni identitet, okoliš, kulturni i religijski utjecaji itd.

Da bi konstruktor mogao relevantno promišljati i donositi odluke mora posjedovati ili imati pristup znanjima koja imaju upliva na konstrukciju prilikom konstruiranja proizvoda. Pod znanjima se smatra, ne samo znanje usko vezanom za proces konstruiranja, već i znanje o vanjskim utjecajnim faktorima (gdje su nastali, kako utiču na konstrukciju, itd.). Na osnovu promišljanja o procesu konstruiranja i istraživanjima provedenim u tom području predložena je struktura konstrukcijskog znanja prikazana na slici 6-5.



Slika 6-5 Struktura konstrukcijskog znanja

Konstrukcijsko znanje je podijeljeno, na osnovu vrste informacija koje sadrže i njihov utjecaj na proizvod, na elemente. Elementi konstrukcijskog znanja predstavljaju logičke cjeline. Informacije iz pojedinih elemenata konstrukcijskog znanja mogu se koristiti u drugim elementima. Konstrukcijsko znanje sastoji se iz sljedećih elemenata:

- **Znanje o funkciji** opisuje funkciju proizvoda ili dijela koji se konstruira. Vezano je za konstruktorovu percepciju proizvoda ili dijela u određenom kontekstu (npr. sklopu ili ciljanom radnom okruženju). Ovo znanje opisuje ulogu komponente[50] u sklopu.
- **Znanje o strukturi** opisuje strukturu proizvoda i veze između komponenti proizvoda (značajki ili dijelova) te znanje potrebno za kreiranje različitih verzija proizvoda, bilo da se radi o proizvodu kao sklopu ili proizvodu kao samostalnoj cjelini.
- **Znanje o ponašanju** opisuje ponašanje proizvoda i usko je vezano za strukturu proizvoda. Ponašanje se opisuje putem parametara koji određuju stanje komponente i zakonitosti koje određuju principe.
- **Znanje o transportu** opisuje način transporta proizvoda te ograničenja i dozvoljena odstupanja pojedinih parametara proizvoda bitnih za transport (npr. gabariti, težina).
- **Znanje o uporabi** ovo znanje nije direktno vezano za proces konstruiranja već se odnosi na problematiku uporabe gotovog proizvoda u eksploataciji. Povrat

[50] Pod komponentom se smatra element sklopa tj. dio ("part") ili element dijela tj. značajka ("feature")

informacija vezanih za uporabu proizvoda može se iskoristiti u konstrukciji sličnih, prilagodbi i poboljšanju istih te konstrukciji novih proizvoda.

- **Znanje o izradi** opisuje znanja vezana za tehnologiju izrade proizvoda. Područje koje pokriva navedeno znanje je široko i kompleksno. Ovo područje se sastoji od dijelova znanja vezanih za dostupne i raspoložive tehnološke procese, obradne strojeve, postupke završne obrade, montaže itd..
- **Znanje o procesu konstruiranja** opisuje proces konstruiranja u cjelini, veze i odnose između pojedinih faza procesa konstruiranja te utjecaj ostalih faza procesa konstruiranja na konstrukciju. U okviru znanja o procesu konstruiranja uključeno je znanje vezano za listu zahtijeva te koncipiranje i projektiranje.
- **Fizikalno znanje** opisuje fizikalne zakone, materijale, strukturu i osobine materijala te specifična svojstva istih (kao što su podatnost itd.).
- **Znanje o značajkama** usko je vezano za mogućnosti CAD sustava. Navedeno znanje opisuje pojedine značajke tj. opisuje, inženjersko značenje značajke[51], atribute, parametre i dimenzije te relacije između značajki u dijelu i ili sklopu.
- **Znanje o tolerancijama** predstavlja kompleksno područje vezano za tolerancije oblika i položaja, mjera, kvalitetu obrade površine, vezane tolerancije, tolerancijska polja te utjecaj pojedinih tolerancija na značajku tj. proizvod (preklopi, spojevi).
- **Znanje o recikliranju** opisuje materijale koji se moraju koristiti i postupke koji se moraju poštivati da bi se kreirani proizvod mogao reciklirati.
- **Znanje o okolišu** opisuje utjecaj proizvoda na okolinu, te određuje ograničenja na proizvod koja se moraju poštivati ukoliko se proizvod želi koristiti u određenom okolišu.
- **Znanje o tržištu** opisuje utjecaj tržišta na proizvod, cijene materijala, cijene transporta i pakiranja, trenutno aktualne trendove u dizajnu, te cijene istog proizvoda konkurentnih proizvođača.
- **Znanje o oblikovanju** opisuje znanje neophodno da bi se ispravno oblikovao proizvod. Oblikovanje se može promatrati sa npr. tehnološkog ili dizajnerskog aspekta.

Budući da je konstrukcijsko znanje u predloženom obliku preopsežno, u radu će se istraživanje ograničiti na znanje o strukturi uključujući i utjecajne faktore drugih znanja u okviru u kojem utječu na znanje o strukturi. Tako definirano znanje tvorit će konstrukcijsko znanje. Iako su postavljenja određena ograničenja na obim proučavanja konstrukcijskog znanja, koncept modela strukture konstrukcijskog znanja, koji je realiziran, i koncept sustava za obradu i manipulaciju konstrukcijskim znanjem, omogućuje ograničeno kreiranje i uporabu znanja iz svih područja prikazanih na slici 6-5.

U sljedećim poglavljima detaljnije je dan opis strukture konstrukcijskog znanja te uloge znanja o funkciji i ponašanju u sustavu . Ostala znanja, koja čine konstrukcijsko znanje, svako za sebe predstavlja veliko i kompleksno područje i zahtijevaju posebna istraživanja koja nadilaze cilj ovoga rada.

[51] Pod inženjerskim značenjem značajke smatra se opis značajke u inženjerskim terminima (utor, rukavac, hrapavost površine, ...)

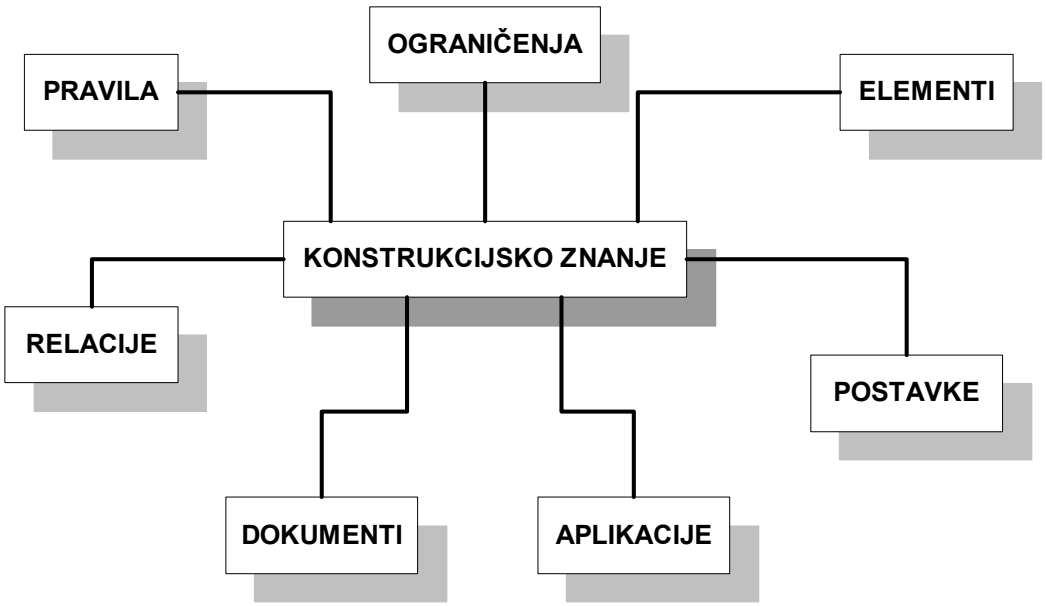
6.1.1.1 Konstrukcijsko znanje

Model konstrukcijskog znanja podijeljen je u logičke grupe:

- elementi – značajke i dijelovi koji tvore pojedine verzije proizvoda,
- pravila – po kojima se aktiviraju pojedina znanja ovisno o odabranoj verziji,
- dokumenti – digitalni ili fizički dokumenti koji tvore ili su vezani za određena znanja o proizvodu,
- aplikacije – vanjski računalni programi koji se aktiviraju ovisno o verziji proizvoda,
- postavke – predložene vrijednosti i dozvoljene granice u kojima se moraju nalaziti vrijednosti parametara te sustav jedinica i postavke svojstava materijala,
- ograničenja – dijelovi koji moraju postojati u sklopu ukoliko se želi uporabiti odabrana verzija proizvoda te ograničenja prilikom uporabe geometrijskog modela dijela u sklopu,
- relacije – načini postavljanja vrijednosti parametara geometrijskog modela proizvoda.

Logičke grupe čine gradbeni elementi znanja tj. entiteti znanja[52]. Logičke grupe imaju određene uloge u proširenom CAD modelu kako slijedi (Slika 6-6).:

- **Elementi** služe za određivanje značajki koje tvore verziju konstrukcije ili dijelova koji tvore sklop konstrukcije. Za svaku značajku ili dio sklopa zapisuje se identifikacijska oznaka te informacija o tome da li je značajka aktivna u geometrijskom CAD modelu konstrukcije. Prilikom određivanja dijelova koji tvore određenu verziju konstrukcije velika pozornost se poklanja kontroli "regeneracije"[53] sklopa ili dijela. Ovaj dio konstrukcijskog znanja je nositelj geometrijske konfiguracije CAD modela konstrukcije.



Slika 6-6 Struktura modela konstrukcijskog znanja

[52] Entiteti znanja su detaljnije objašnjeni u poglavlju 6.1.1.3.

[53] Ukoliko se želi osigurati ispravnost konstrukcije tj. verzije mora se voditi računa o odnosu otac – dijete u modelu, tj. prilikom određivanja značajki ili komponenti koji tvore određenu verziju konstrukcije ne smije se dozvoliti da se prilikom isključivanja "suppress" neke značajke ili komponente naruši ispravnost i regenerativnost modela jer je dijete ostalo bez reference tj. bez oca

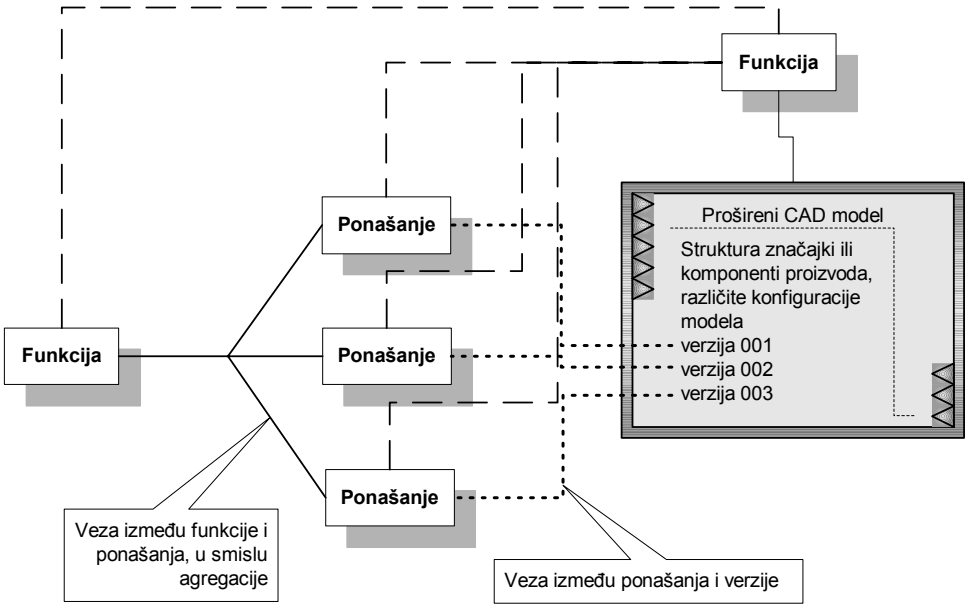
- **Pravila** određuju slijed kontrole ograničenja za odabranu verziju konstrukcije te aktiviranje pojedinih prikaza dokumentacije bitne za odvijanje tijeka konstruiranja. Pravila predstavljaju središnji dio modela konstrukcijskog znanja i realizirana su preko skupa ključnih riječi koje, uz definiranu sintaksu i semantiku, tvore programski kod. Programski se kod prilikom aktiviranja odabrane verzije proizvoda mora raščlaniti[54] te izvršiti.
- **Dokumenti** služe za određivanje dokumenata vezanih za konstrukciju i datoteka pomoći vezanih za pojedina znanja. Dokumenti mogu biti tekstualne datoteke ili datoteke u HTML formatu zapisa Datoteke dokumenata, neovisno o formatu zapisa, mogu biti smještene na lokalnom ili na udaljenom računalu. Pristup do dokumenata na udaljenom računalu omogućen je mrežnim protokolima neovisno da li se radi o lokalnoj ili globalnoj mreži.
- **Aplikacije** opisuju računalne programe koje se koriste tijekom procesa konstruiranja. Aktiviranje programskih aplikacija upravlja se programskim kodom znanjem – pravila. Znanje o aplikacijama osim naziva računalnog programska koji se aktivira sadrži i informaciju o putanji do programske aplikacije te način prijenosa argumenata aplikaciji i povrata rezultata u sustav.
- **Postavke** opisuju postavke koje se koriste u drugim dijelovima sustava te početne i preporučene vrijednosti parametara prilikom kreiranja znanja – relacije. U okviru ovog znanja postavljaju se početni parametri za: zadavanje minimalne i maksimalne vrijednosti granica vrijednosti parametara, zadavanje skupa diskretnih vrijednosti iz kojeg se odabire vrijednost parametra, zadavanje sustava jedinica, te zadavanje svojstava materijala.
- **Ograničenja** opisuju uvjete koji se moraju poštivati i kontrolirati prilikom aktiviranja ili odabiranja pojedine verzije konstrukcije. Pojedina ograničenje su određena uvjetom postojanja pojedinih komponenti u sklopu. Osnovna namjena ovog znanja je određivanje komponenti s kojima i na koji način odabrana verzija konstrukcije se može uklopiti u višoj instanci tj. sklopu.
- **Relacije** opisuje načine određivanja vrijednosti pojedinih parametara ili dimenzija konstrukcije. Vrijednost dimenzije ili parametra značajke može se zadati na sljedeće načine:
  - direktnim zadavanjem vrijednosti,
  - zadavanjem programske aplikacije (čiji rezultat će postati nova vrijednost parametra ili dimenzije, nakon aktiviranja),
  - zadavanjem područja unutar kojeg se može kretati vrijednost parametra ili dimenzije uz zadavanje preporučene vrijednosti,
  - odabirom iz diskretnog skupa vrijednosti uz zadavanje preporučene vrijednosti.

6.1.1.2 Znanje o funkciji i ponašanju

Znanje o funkciji i znanje o ponašanju proizvoda u radu je samo dotaknuto, naime problematika proučavanja, interpretacije i klasificiranja znanja o funkciji i znanja o ponašanju prelazi okvire ovog rada. Oba znanja će se u sustavu koristiti isključivo za pojašnjenje verzija konstrukcije proizvoda i uloge proizvoda u širem kontekstu. Ujedno znanje o funkciji i znanje o

[54] Potrebna je programska aplikacija ili programski kod "parser" koji će provjeriti sintaksu zapisa te ovisno o zapisu pokrenuti pojedine akcije

ponašanju će se koristiti i za klasificiranje i kasniji kriterij odabira pojedinih verzija proizvoda. Dodatni očekivani doprinos uključivanja znanja o funkciji i znanja o ponašanju je u boljem razumijevanju proizvoda, naročito ukoliko se radi o složenijim proizvodima.



Slika 6-7 Veza između funkcije, ponašanja i verzije

Znanje o funkciji veže se za konstruktorovu percepciju uloge proizvoda u kontekstu u kojem se proizvod koristi. Konstruktor prilikom kreiranja proširenog CAD modela tj. dodavanjem dijelova konstrukcijskog znanja mora odrediti funkciju proizvoda ili parcijalnu funkciju komponente proizvoda. Funkcija se može izabrati iz skupa predodređenih funkcija ili kreiranjem nove funkcije. Funkcija se definira na razini definiranja znanja o znanju.

Ponašanje je za razliku od funkcije vezano za strukturu proizvoda. Uloga znanja o ponašanju u kontekstu konstrukcijskog znanja je u svrhu pojednostavljenja klasifikacije i kasnije uporabe proširenih CAD modela od strane konstruktora. Budući da svaki konstruktor neće kreirati proširene CAD modele (iako je to moguće) potreban je dodatni način za razumijevanje konstrukcije proizvoda i uporabe željene verzije u sklopu. Ponašanje bi se moglo razbiti u pod dijelove, no u radu je umjesto toga omogućeno povezivanje ponašanja i verzije proizvoda. Na taj način je kreirana mogućnost vezivanja strukture i funkcije ovisno o ponašanju tj. svaka verzija konstrukcije za koju je vezana tj. definirana funkcija vezano je i ponašanje koje ovisi o strukturi proizvoda (Slika 6-7).

6.1.1.3 Entiteti znanja

Model konstrukcijskog znanje prema slici 6-6 sastoji se od logičkih grupa. Logičke grupe čine entiteti znanja kao gradbeni elementi. Entiteti znanja kreiraju se kombinacijom operanda i operatora u izrazu. Osnovni oblik zapisa entiteta znanja je "izraz". Izraz je analogan, u razmatranom kontekstu, "izrazu" kao dijelu naredbe programskog jezika. Izraz je određen kao niz sintaktičkih elemenata koji se sastoje od lokucije[55], operanda i operatora. Ako se izraz

[55] Riječ ili skup riječi kojima se nešto izriče.

interpretira kao ograničenje ili kao uvjet u pravilu, tada se određuje njegova vrijednost kao istina ili laž. Izraz se može definirati i kao predikatna formula nad danim jezikom.

Operandi su atributi izraza i mogu biti znakovne konstante, numeričke konstante ili znakovne varijable, a ovise o konstrukcijskom znanju koje se gradi. Operand je znakovna konstanta, i može mu se pridružiti samo jedan znak, kao i odjelni simbol čija uloga je razdvajanje izraza u logičke cjeline. Lokucija je specifičan oblik zapisa i u osnovi je identičan izrazu uz razliku uporabe liste. Lista je skup numeričkih konstanti. Struktura entiteta znanja opisana je sljedećom shemom (prema EBNF[56] notaciji [128]):

```
izraz =  < id > ',' (< si > | < nk >) ',' (< nr > | < no > | < np > | < td > |
         < sz > | < iok > | < nvpa > | < pv > | < sqlcs >) ',' < lokucija > ','
         (< vk > | < od > | < vo > | < dop >) ',' < docid >
lokucija =    < ndp > | < '=' > (< nk > | < to >) ';' < ioto > ';'
         (< pv > | < '?' > | < path > | < sqlu >) | < '(' >
         (< minv >, < maxv >) | (< vn > | {< vn > ','}) |
         (< sj >) |
         (< ro > ',' < E > ',' < ni > ',' < G >) | (< zkus > ',' < zkup >) < ')' >

id = < numerička konstanta >   - identifikacijska oznaka
si = < znakovna konstanta >    - simboličko ime
nk = < znakovna konstanta >    - naziv komponente
nr = < znakovna konstanta >    - naziv tipa relacija:
                                 map – postavljanje vrijednosti parametra na
                                 osnovu relacije određene token-om
                                 intrel — određuje da se radi o internoj relaciji CAD
                                 sustava
no = < znakovna konstanta >    - naziv operacije:
                                 mate – poravnavanje zadanih značajki lice na lice,
                                 smjerovi normala na pozitivnoj strani gledaju u
                                 suprotnom smjeru
                                 align – poravnavanje zadanih značajki, smjerovi
                                 normala na pozitivnoj strani gledaju u istom smjeru
                                 insert – ubacivanje komponente u provrt (komponenta i
                                 provrt moraju biti kružnog presjeka)
                                 orient – poravnavanje orijentacije zadanih značajki
                                 postoji – opisuje postojanja komponente
np = < znakovna konstanta >    - naziv postavke:
                                 minmaks – određuje zadavanje minimalne i maksimalne
                                 vrijednosti parametra
                                 diskretne – određuje zadavanje skupa diskretnih
                                 vrijednosti parametra
                                 materijal – zadavanje svojstava materijala
                                 jedinice – zadavanje radnih jedinica
td = < znakovna konstanta >    - tip dokumenta:
                                 html – opisuje web dokument u html obliku
                                 doc – opisuje datoteku u formatu zapisa programa za
                                 obradu teksta
                                 txt - opisuje tekstualnu datoteku
                                 asci - opisuje tekstualnu datoteku
                                 xls – opisuje datoteku tabličnog programa
```

[56] Extended Backus-Naur Form

sz = < znakovna konstanta > - status komponente:
aktivna – komponenta je uključena
neaktivna – komponenta je isključena
iok = < numerička konstanta > - identifikacijska oznaka komponente u CAD sustavu
nvpa = < znakovna konstanta > - naziv vanjske programske aplikacije
pv = < numerička konstanta > - pretpostavljena vrijednost
sqlcs = < znakovna konstanta > - niz znakova[57] za ostvarivanje komunikacije sa bazom
vk = < znakovna konstanta > - veza komponente:
slobodna – komponenta nema djece
vezana – komponenta posjeduje djecu
od = < znakovna konstanta > - oblik dokumentacije:
digitalni – oblik zapisa u elektronskom obliku na nekom računalu
fizicki – datoteka se nalazi u fizičkom obliku (papirnati, film, folije, ...)
vo = < numerička konstanta > - vrijednost odmaka, numerička vrijednost koja je vezana za ograničenja **mate** i **align**, a određuje odmak značajki prilikom sučeljavanja (mate offset i align offset)
dop = < znakovna konstanta > - dodatni opis entiteta znanja
docid = < numerička konstanta >- identifikacijska oznaka dokumenta vezanog za određeno znanje
ndp = < znakovna konstanta > - naziv dimenzije ili parametra
nk = < znakovna konstanta > - naziv komponente
to = < znakovna konstanta > - tip operacije:
izvrsi – aktiviranje vanjske programske aplikacije
minmaks – provjera zadavanja vrijednosti između minimalne i maksimalne
diskretne – provjera odabira jedne između ponuđenih diskretnih vrijednosti
ioto = < znakovna konstanta > - identifikacijska oznaka tipa operacije
pv = < znakovna konstanta > - pretpostavljena vrijednost
? = < znakovna konstanta > - određivanje unosa vrijednosti od strane korisnika
path = < znakovna konstanta > - putanja do datoteke koja će se učitati nakon aktiviranja vanjske programske aplikacije
sqlu = < znakovna konstanta > - SQl upit
minv = < numerička konstanta > - minimalna vrijednost parametra
maxv = < numerička konstanta > - maksimalna vrijednost parametra
vn = < numerička konstanta > - {<vn>} niz diskretnih vrijednosti
sj = < znakovna konstanta > - sustav jedinica
ro = < numerička konstanta > - gustoća
E = < numerička konstanta > - modul elastičnosti
ni = < numerička konstanta > - Poissonov koeficijent
G = < numerička konstanta > - modul smicanja
zkus = < znakovna konstanta > - značajka komponente u sklopu
zkup = < znakovna konstanta > - značajka komponente u dijelu

[57] SQL connection string

Tablica 1. Prikaz ispravnih oblika entiteta znanja

1	Provrt	**aktivna, neaktivna**	7	=;;cut id 7	**slobodna, vezana**	2
1		**map**	2	d0 = part3;;d48		3
				d0 = part3 ; ; d48		
				d0 = **izvrsi** ; 1 ; 8, 30, 40		
				d0 = **minmaks** ; 2 ; 20		
				d0 = **diskretne** ; 3 ; 5		
				d0 = ; ; 34		
				d0 = ; ; ?		
		intrel		d28 = d13 / 2		
1	Excel proračun	**cp, sql, run**	excel.exe	=;;D:\home\one.xls		4
1	Standard A	**html, doc, txt**	ie.exe	=;;D:\home\two.html	**digitalni, fizički**	
1	granice A	**minmaks**	10	=;;(20, 30)		6
	vrijednosti B	**dikretne**	5	=;;(2, 5, 8, 10)		
	C.0361	**materijal**		=;;(ro, E, ni, G)		
	ANSI	**jedinice**		=;;(ANSI)		
1	part11	**mate, align, insert, orient, postoji**		=;;(DTM1, DTM3)		10

Izraz se zapisuje kao niz znakova tijekom kreiranja pojedinog znanja. Ključne riječi koje se mogu pridružiti pojedinim entitetima znanja ovise o konstrukcijskom znanju koje se gradi tj. kreira. Vrijednost izraza određuje se raščlanjivanjem[58] i interpretacijom u tijeku rada sustava. Postupkom raščlanjivanja varijablama se dodjeljuju vrijednosti atributa objekata. Budući da se dio konstrukcijsko znanja – pravila razlikuje od ostalog konstrukcijskog znanja, ovo znanje će se tretirati kao iznimka i neće se graditi uporabom entiteta znanja.

U sljedećem dijelu rada prikazani su pojedini entiteti znanja, opis atributa entiteta i moguće vrijednosti atributa. Za dio entiteta znanja koji je vezan za relacije prikazane su mogućnosti inicijalizacije vrijednosti parametara CAD modela. Sažeti prikaz ispravnih oblika entiteta znanja prikazan je u tablici 1. .

Entitet **ELEMENTI:**

Izraz:: <id>, <si>, <sz>**,** <iok>,<nk>, <vk>, <docid>

- <id> – identifikacijska oznaka
- <si> – simbolički naziv
- <sz> – opisuje trenutno stanje komponente u CAD modelu, vrijednosti mogu biti:
  - **aktivna** – komponenta je uključena
  - **neaktivna** – komponenta je isključena
- <iok> – identifikacijska oznaka komponente u CAD modelu
- <nk> – naziv komponente u CAD modelu
- <vk> – ključna riječ koja određuje da li je komponenta vezana ili nije, vrijednosti mogu biti:

[58] Postupak se obavlja programski uporabom parser-a

- **`slobodna`** – opisuje komponentu kao slobodnu tj. komponenta nema djece[59], na osnovu čega slijedi da se komponenta može isključiti[60] ili uključiti[61] bez implikacija na konzistentnost CAD modela
- **`vezana`** – opisuje komponentu kao vezanu tj. komponenta posjeduje djecu, na osnovu čega slijedi da će isključenje ili uključenje komponente, isključiti ili uključiti najmanje jednu dodatnu komponentu, što može utjecati na konzistentnost CAD modela

- <docid> – pokazivač na identifikacijsku oznaku dokumenta koji je vezan za ovaj entitet znanja, dokument opisuje problematiku vezanu za uporabu tj. aktiviranje ovog entiteta znanja, dokument kreira korisnik, a prikazuje se na zahtjev

Primjer:: #1, Provrt, **`aktivna,`** 7,= cut id 7, **`slobodna`**, #2

Entitet **RELACIJE:**

Izraz:: <id>, <si>, <lokucija>;;d48, **`map`**, , #3

- <id> – identifikacijska oznaka
- <si> – simboličko ime u sustavu
- <lokucija> – izraz znanja – relacija koji određuje način inicijalizacije parametra CAD modela, izraz može imati sljedeće oblike:
  - <ndp> = <nk>;;<pn> – direktno postavljanje vrijednosti parametra na vrijednost parametra iz drugog dijela ili komponente :
    - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja,
    - <nk> – naziv komponente iz koje se očitava vrijednost drugog parametra,
    - <pn> – naziv parametra čija vrijednost će se uporabiti za inicijalizaciju prvog parametra,
    - Primjer:: d0 = part3 ; ; d48
  - <ndp> = <to>;<ioto>;{<vn>} – vrijednost parametra ovisi o povratnoj vrijednosti iz vanjske programske aplikacije:
    - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja,
    - <to> – ključna riječ koja određuje aktiviranje vanjskog programa ili aplikacije u svrhu dobivanja vrijednosti parametra,
    - <ioto> – identifikacijska oznaka znanja program koja određuje vanjsku programsku aplikaciju koja će se aktivirati,
    - {<vn>} – lista argumenata koja će se proslijediti programskoj aplikaciji,
    - Primjer:: d0 = **`izvrsi`** ; #1 ; 8, 30, 40
  - <ndp> = <to>;<ioto>;<pv> – vrijednost parametra će se postaviti na zadnju zadanu vrijednost u zapisu entiteta (predložena vrijednost), prilikom zadavanja predložene vrijednosti kontroliraju se granice unutar kojih se vrijednost mora nalaziti
    - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja
    - <to> – ključna riječ koja određuje provjeru gornje i donje granice zadavanja predložene vrijednosti,
    - <ioto> – identifikacijska oznaka znanja postavke koje će se uporabiti za provjeru granica,
    - <pv> – predložena vrijednost parametra ,

[59] Child – u kontekstu odnosi se na Parent-Child relationship model CAD sustava
[60] Suppress – naredba u okviru CAD sustava za isključivanje značajke ili komponente
[61] Resume – naredba u okviru CAD sustava za uključivanje značajke ili komponente

        - Primjer:: d0 = **`minmaks`** ; #2 ; 20
    - <ndp> = <to>;<ioto>;<pv> – vrijednost parametra će se postaviti na zadnju zadanu vrijednost u zapisu entiteta (predložena vrijednost), prilikom zadavanja predložene vrijednosti moguće je odabrati samo jednu od ponuđenih vrijednosti :
        - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja,
        - <to> – ključna riječ koja određuje provjeru odabira predložene vrijednosti između ponuđenih određenih znanjem postavke,
        - <ioto> – identifikacijska oznaka znanja postavke koje će se uporabiti za provjeru odabrane vrijednosti,
        - <pv> – predložena vrijednost parametra,
        - Primjer:: d0 = **`diskretne`** ; #4 ; 5
    - <ndp> = ;;<pv> – direktno postavljanje vrijednosti parametra na zadanu vrijednost:
        - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja,
        - <pv> – zadana vrijednost parametra,
        - Primjer:: d0 = ; ; 34
    - <ndp> = ;;<?> – određuje obavezno zadavanje vrijednosti parametra od strane korisnika:
        - <ndp> – naziv parametra ili dimenzije čija se vrijednost postavlja,
        - <?> – ključna riječ koja određuje obavezno zadavanje vrijednosti, parametra od strane korisnika,
        - Primjer:: d0 = ; ; ?
- <nr> – ključna riječ koja određuje da li se radi o internim relacijama CAD modela ili relacijama kreiranim u sustavu, vrijednosti mogu biti:
    - **`map`** – određuje relacije kreirane u sustavu,
    - **`intrel`** – određuje relacije kreiran u CAD modelu, relacije kreirane u CAD modelu ne mogu se koristiti u pravilima već samo unutar CAD modela,
- <docid> – pokazivač na identifikacijsku oznaku dokumenta koji je vezan za ovaj entitet znanja, dokument opisuje problematiku vezanu za uporabu tj. aktiviranje ovog entiteta znanja, dokument kreira korisnik, a prikazuje se na zahtjev,

Primjer:: #1, veza 1, d0 = part3;;d48, **`map`**, , #3

Entitet **APLIKACIJE:**

Izraz:: <id>, <si>, <ni>, <nvpa>, <sqlcs>, <docid>

- <id> – identifikacijska oznaka,
- <si> – simboličko ime,
- <ni> – opcija određuje način aktiviranja vanjske programske aplikacije te način prijenosa argumenata, vrijednosti mogu biti:
    - **`run`** – vanjska programska aplikacija aktivira se direktno, a argumenti se prenose programski,
    - **`sql`** – određuje uspostavljanje veze sa bazom podataka preko SQL veze, te uporabu SQL upita za prijenos podataka,
    - **`cp -`** vanjska programska aplikacija aktivira se direktno, a argumenti se prenose pomoću copy/paste paradigme,
- <nvpa> – naziv vanjske programske aplikacije koja se aktivira ako je vrijednost opcije načina aktiviranja **`run`** ili string za vezu s bazom tj. "connection string" ukoliko je vrijednost opcije načina aktiviranja **`sql`**,

- <sqlcs> – ova opcija može imati dvojaku funkciju: ukoliko je vrijednost opcije načina aktiviranja **`run`** tada određuje naziv vanjske datoteke koja će se koristiti prilikom aktiviranja vanjske programske aplikacije, ukoliko je vrijednost opcije načina aktiviranja **`sql`** tada određuje SQL upit koji će se proslijediti bazi,
- <docid> – pokazivač na identifikacijsku oznaku dokumenta koji je vezan za ovaj entitet znanja, dokument opisuje problematiku vezanu za uporabu tj. aktiviranje ovog entiteta znanja, dokument kreira korisnik, a prikazuje se na zahtjev,

Primjer:: #1, Excel proračun, **`run`**, excel.exe, =D:\home\one.xls, #4

Entitet **DOKUMENTI:**

Izraz:: <id>, <si>, <td>, <nvpa>, <sqlcs>, <od>

- <id> – identifikacijska oznaka,
- <si> – simboličko ime,
- <td> – opcija određuje tip dokumenta koji se prikazuje, vrijednosti mogu biti:
  - **`html`** – web dokument u HTML obliku zapisa,
  - **`doc`** – opisuje datoteku u formatu zapisa programa za obradu teksta,
  - **`txt`** – opisuje tekstualnu datoteku,
  - **`asci`** – opisuje tekstualnu datoteku,
  - **`xls`** – opisuje datoteku tabličnog programa,
- <nvpa> – vanjska programska aplikacija,
- <sqlcs> – putanja do dokumenta koji se želi prikazati; putanja može biti prema datoteci na lokalnom računalu, na računalu u mreži ili prema datoteci na računalu na globalnoj mreži; ukoliko se radi o fizičkom dokumentu sadrži opis smještaja dokumenta; ukoliko se ispred niza znakova nalazi znak ! tada će se dokument obavezno prikazati svaki put kada se aktivira kreirani entitet znanja,
- <od> – oblik dokumenta, vrijednosti mogu biti:
  - **`digitalni`** – oblik zapisa u elektronskom obliku na nekom računalu,
  - **`fizicki`** – datoteka se nalazi u fizičkom obliku,

Primjer:: #1, Standard B, **`html`**, iexplorer.exe, =http:\\www.cadlab.fsb.hr\stb.html, **`digitalni`**

Entitet **POSTAVKE**:

Izraz:: <id>, <si>, <to>, , =(<sj>), , <docid>

- <id> – identifikacijska oznaka,
- <si> – simboličko ime entiteta znanja,
- <to> – ključna riječ koja određuje postavljanje sustava jedinica, ostale vrijednosti mogu biti:
  - **`minmaks`** – određuje zadavanje minimalne i maksimalne vrijednosti parametra,
  - **`diskretne`** – određuje zadavanje skupa diskretnih vrijednosti parametra,
  - **`materijal`** – zadavanje svojstava materijala,
- =(<sj>) – zadani sustav jedinica,
- <docid> – pokazivač na identifikacijsku oznaku dokumenta koji je vezan za ovaj entitet znanja; dokument opisuje problematiku vezanu za uporabu tj. aktiviranje ovog entiteta znanja; dokument kreira korisnik, a prikazuje se na zahtjev,

Primjer:: #1, ANSI, **jedinice**, , =(ANSI), , #6

Entitet **OGRANIČENJA:**

Izraz:: <id>, <nk>, <no>, ,= (<zkus>, <zkup>), , docid

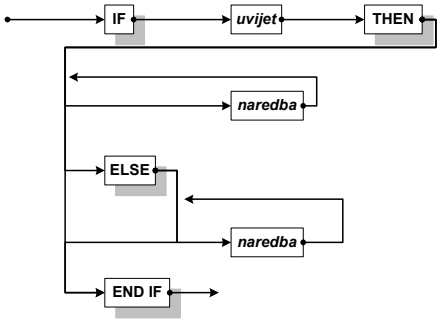
- <id> – identifikacijska oznaka,
- <nk> – naziv komponente u CAD sustavu prema kojoj se određuju ograničenja,
- <no> – zadano ograničenje, određuje poravnavanje zadanih značajki licem na lice, smjerovi normala na pozitivnoj strani moraju gledati u suprotnom smjeru, ostale vrijednosti mogu biti:
  - **mate** – poravnavanje zadanih značajki, smjerovi normala na pozitivnoj strani gledaju u suprotnom smjeru,
  - **align** – poravnavanje zadanih značajki, smjerovi normala na pozitivnoj strani gledaju u istom smjeru,
  - **insert** – ubacivanje komponente u provrt (komponenta i provrt moraju biti kružnog presjeka),
  - **orient** – poravnavanje orijentacije zadanih značajki,
  - **postoji** – određuje samo provjeru postojanja komponente,
- =(<zkus>, <zkup>) – značajke komponenti koje se sučeljavaju:
  - <zkus> - naziv značajke u sklopu,
  - <zkup> - naziv značajke u dijelu,
- <docid> – pokazivač na identifikacijsku oznaku dokumenta koji je vezan za ovaj entitet znanja; dokument opisuje problematiku vezanu za uporabu tj. aktiviranje ovog entiteta znanja; dokument kreira korisnik, a prikazuje se na zahtjev,

Primjer:: #1, part34, **mate**, ,= (adtm1, dtm1), , #7

Zapis konstrukcijskog znanja, pravila, razlikuje se od zapisa ostalih oblika znanja u tome što je zapis pravila sličniji strukturi programskog jezika. U okviru dijela konstrukcijskog znanja – pravila definirane su ključne riječi (naredbe) i sintaksa jezika pravila, prema programskim cjelinama, na sljedeći način:

- Programski blokovi:
  - **GLOBAL**, **END GLOBAL** – blok koji određuje skup naredbi koje se izvršavaju svaki put prilikom uporabe pojedine verzije konstrukcije,
  - **FOR *verzija*, END FOR** – blok naredbi koje su specifične za pojedinu verziju, naredbe koje se nalaze unutar bloka izvode se svaki put kada se pojedina verzija odabere ili ubaci u sklop, naziv verzije je znakovna konstanta,
- Uvjetno izvršavanje:
  - **IF *uvjet* THEN, ELSE, END IF** – struktura logičke if-blok naredbe je istovjetna strukturi identične naredbe u programskom jeziku Fortran [129], blokovi if naredbe mogu biti ugniježđeni te mogu ili ne moraju sadržavati opciju **ELSE**, definicija blok if naredbe prikazana je na slici 6-9. Kao operatori logičkog uvjeta ne koristi se sintaksa Fortran-a nego standardni matematički operatori uspoređivanja ("<" - manje od, ">" - veće od, "=" - jednako , "**!**=" - različito), prilikom kreiranja logičkih uvjeta, u trenutnoj implementaciji, nije predviđena uporaba logičkih operatora (negacije, konjunkcije i disjunkcije) u svrhu kreiranja složenijih uvjeta.

- **OGRANIČENJE –** ključna riječ koja određuje aktiviranje kontrole zadovoljenja zadanog ograničenja, ukoliko ograničenje nije zadovoljeno traži se intervencija korisnika, zadana postavka mora prethodno biti zadana u okviru znanja ograničenja,



Slika 6-8 Struktura blok IF naredbe

- Izvršne naredbe:
  - **MATERIJAL** – određivanje aktivnih postavki materijala za odabranu verziju, zadana postavka mora prethodno biti zadana u okviru znanja postavke,
  - **DOKUMENT** – aktiviranje prikaza sadržaja zadanog dokumenta prilikom uporabe (ubacivanja u sklop) određene verzije, zadana postavka mora prethodno biti zadana u okviru znanja dokument,
  - **JEDINICE** – određivanje aktivnih jedinica za odabranu verziju, zadana postavka **unit** mora prethodno biti zadana u okviru znanja dokument,
  - **RELACIJA** – aktiviranje preslikavanja vrijednosti parametara i dimenzija u modelu, zadana postavka mora prethodno biti zadana u okviru znanja relacije,
  - **ELEMENT** – određivanje značajki koje tvore verziju.

Primjer uporabe naredbi pravila prikazan je na slici 6-9. Implementacija tako postavljanje strukture biti će detaljnije razrađena u sljedećem poglavlju.

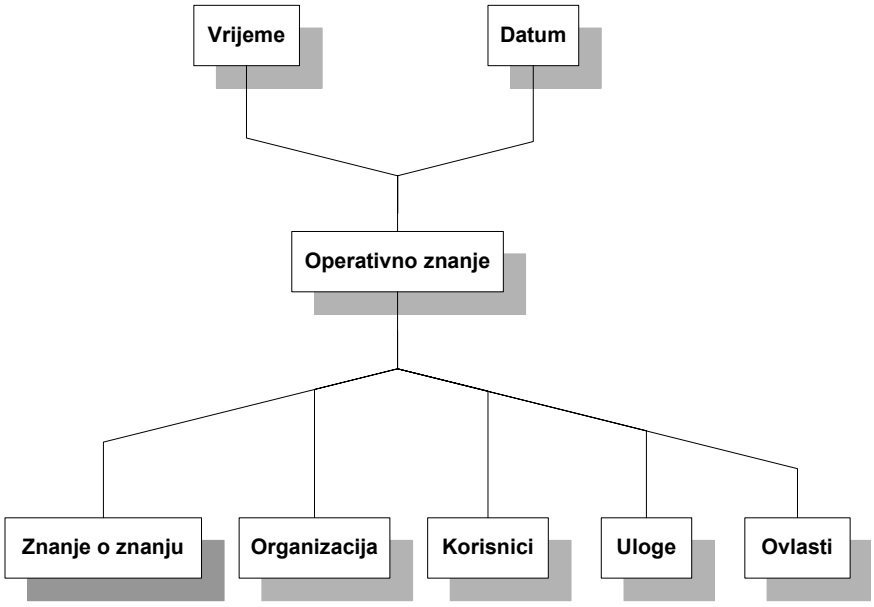
```
GLOBAL
   JEDINICE 1
   DOKUMENT 3
END GLOBAL
FOR verzija 1
   PROVJERI 1
   ELEMENT 2
   IF OGRANICENJE 3  THEN
         RELACIJA 1
         DOKUMENT 1
         DOKUMENT 2
         MATERIJAL 1
   ELSE
         RELACIJA 2
         DOKUMENT 1
   END IF
END FOR

FOR verzija2
   OGRANICENJE 2
   ELEMENT 3
   IF d20 > 9.0 THEN
         RELACIJA 2
         DOKUMENT 1
   END IF
   IF OGRANICENJE 9 THEN
         IF part18:d34 == 35 THEN
               RELACIJA 1
               MATERIJAL 2
         ELSE
               RELACIJA 2
         END IF
   END IF
END FOR
```

Slika 6-9 Primjer uporabe naredbi pravila

6.1.2 Operativno znanje

Operativno znanje sadrži dodatne informacije o konstrukcijskom znanju. Dodatna znanja su neophodna prilikom manipulacije i upravljanja znanja i predstavljaju nadogradnju konstrukcijskog znanja. Osim što je nadogradnja konstrukcijskog znanja, operativno znanje, služi i kao veza između proceduralnog i konstrukcijskog znanja. Operativno znanje se sastoji iz sljedećih dijelova (Slika 6-10):

- **znanje o znanju** – opisuje znanja koja su uporabljena pri konstruiranju proizvoda, na osnovu ovog znanja kreiraju se upiti koji se mogu uporabiti za pregledavanje i proučavanje znanja uporabljenog pri konstruiranju proizvoda,
- **datum i vrijeme** – opisuje ulogu datuma i vremena u različitim kontekstima, u kontekstu nastanka znanja tj. kreiranja, uporabe, promijene ili u kontekstu korisnika prijava za rad, promjena uloga ili prava,
- **organizacija** – opisuje informacije vezane za tvrtku ili radnu jedinicu u tvrtci u kojoj korisnik tj. konstruktor radi,
- **korisnici** – opisuje korisnike sustava tj. konstruktore; podatke oblika: ime, prezime, radno mjesto, itd.,
- **uloge** – opisuje uloge koje se mogu pridružiti pojedinim korisnicima,
- **ovlasti** – opisuje ovlasti korisnika u radu sa pojedinim znanjima i sa sustavom.

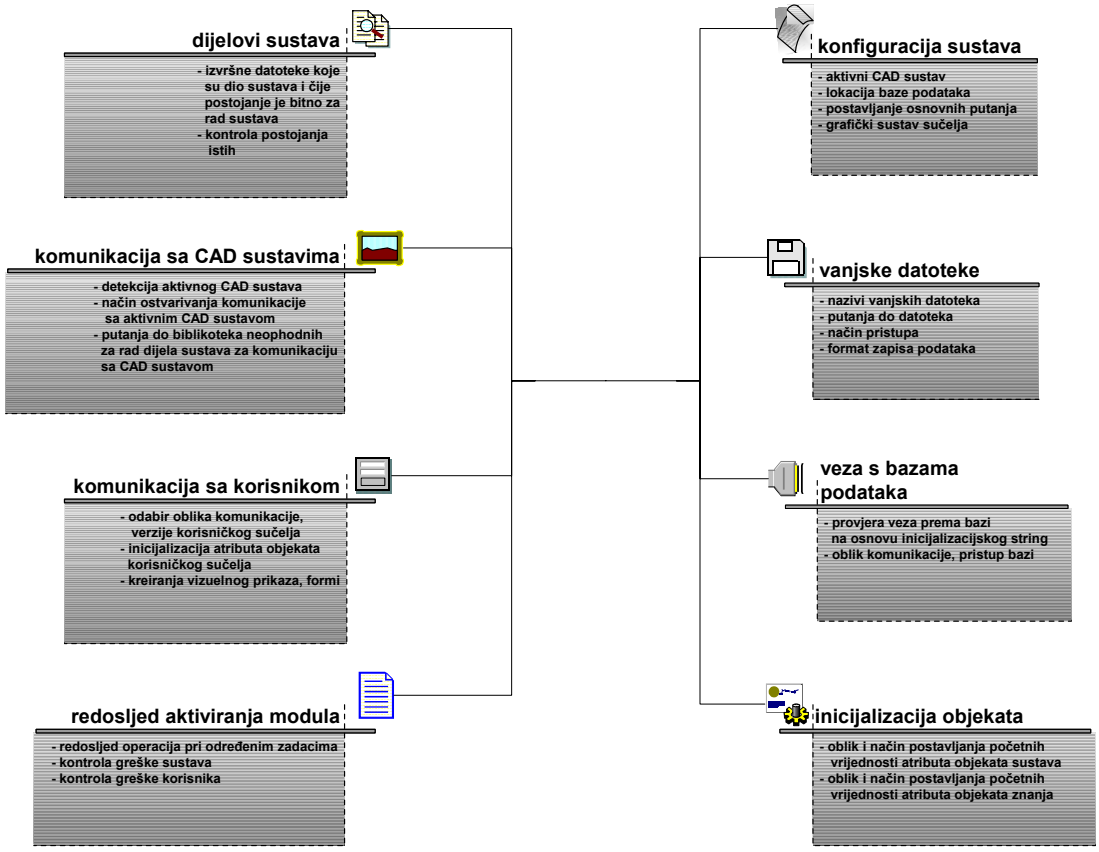


Slika 6-10 Struktura operativnog znanja

6.1.3 Proceduralno znanje

Proceduralno znanje je vezano uz implementaciju sustava tj. strukturu klasa te definiciju metoda i atributa klasa. Za razliku od konstrukcijskog i operativnog znanja, koje korisnik kreira tijekom rada preko korisničkog sučelja, proceduralno znanje je integrirano tj. kodirano u programski kod sustava i ne može se mijenjati od strane korisnika. Dijelovi proceduralnog znanja (Slika 6-11) su:

- informacije o vanjskim datotekama (datotekama koje služe za inicijalizaciju dijelova sustava i datotekama potrebnim u tijeku rada sustava); informacije se svode na podatke o imenu datoteke, putanji do datoteke i ulozi datoteke u sustavu,
- informacije vezana za komunikaciju s bazama podataka; naziv baze, putanja do baze, inicijalizacijski string, oblik komunikacije s bazom,
- informacije o dijelovima sustava i načinima inicijalizacije istih,
- informacije o konfiguraciji sustava, načinu promjene konfiguracije te načinu aktiviranja odabrane konfiguracije,
- informacije vezane za komuniciranje sa grafičkim sučeljem sustava tj. razmijeni podataka između korisnika i sustava,
- informacije vezane za komunikaciju između sustava i CAD aplikacije pridružene sustavu te modaliteti uporabe pojedinih modula za pripremu podataka za CAD aplikaciju,
- informacije o načinu kreiranja pojedinih objekata sustava, poglavito objekata konstrukcijskog i operativnog znanja te načinima popunjavanja vrijednost atributa objekata,
- informacije o redoslijedu aktiviranja pojedinih modula ovisno o situaciji tj. vrijednosti kontrolnih parametara.



Slika 6-11 Struktura proceduralnog znanja

7 Implementacijski model konstrukcijskog znanja

Temeljem prikaza predloženog koncepta proširenog CAD modela u prethodnoj glavi rada predložen je implementacijski model konstrukcijskog znanja. Struktura i dijelovi implementacijskog modela prikazat će se u ovoj glavi rada. Prošireni CAD model se sastoji od CAD modela i konstrukcijskog znanja spregnutog pravilima zaključivanja. CAD model se kreira uporabom neovisnih ali povezanih geometrijskih objekata. Geometrijski objekti su određeni atributima. Atributi su podaci koji opisuju fizičke komponente (npr. grede, oplatu, priključke, itd.). Atributi geometrijskih objekata koriste se i prilikom kreiranja znanja (ograničenja, relacije, postavke). Osnovni gradbeni elementi znanja su entiteti znanja. Entiteti znanja opisani su atributima. Vrijednosti i struktura atributa entiteta znanja mora se moći zapisati, pročitati, obrisati i promijeniti tijekom rada stoga je neophodan efikasan sustav rukovanja podacima[62]. Uporaba baze podataka za upravljanje podacima omogućuje sljedeće prednosti [130]:

- mogućnost pohrane i pristupa podacima neovisno o njihovoj uporabi, omogućuje istovremenu uporabu podataka,
- mogućnost prikaza strukture podataka, omogućuje dokumentiranje međuovisnosti,
- provjera redundancije podataka, čime je povećana konzistentnost podataka,
- upravljanje integritetom podataka, ostvarivanje istovremenog pristupa istim podacima od strane različitih korisnika,
- podrška upravljanju datotekama i kreiranju izvještaja "na zahtjev".

Relacije među podacima koji opisuju strojarsku konstrukciju su u većini slučajeva kompleksne tako da nedostatak razinskog pristupa postavlja dodatne probleme pri interpretaciji zapisa u bazi. Rješenje opisanih problema može se naći u uporabi STEP modela (vidi poglavlje 4.) kao osnove za kreiranje strukture zapisa podataka o konstrukcijskom znanju o proizvodu [131].

Budući da korisnik, konstruktor, na proizvod ili komponente proizvoda gleda kao na cjeline koje posjeduju identitet, ponašanje i stanje, sa strane koncipiranja sustava, nameće se objektni pristup. Uporaba objekata ne podrazumijeva direktno potpunu implementaciju

[62] DBMS – Database Management System

paradigme objektnog pristupa. Objektno orijentirani pristup modeliranju konstrukcijskog znanja omogućuje prirodnu dekompoziciju i kreiranje hijerarhijske strukture znanja. U ovom radu primijenjen je pristup modeliranja konstrukcijskog znanja uporabom objekata, a atributi objekata pohranjeni su u relacijsku bazu. Na taj način omogućeno je dijeljenje podataka između objekata te pristup podacima spremljenim u bazi i od strane drugih programskih aplikacija.

7.1 Struktura zapisa znanja

Na osnovu predložene strukture konstrukcijskog znanja dane u 6. poglavlju može se prikazati struktura pojedinih dijelova. Dijelovi konstrukcijskog znanja opisani su u poglavlju 6.1.1.1, a osnovni gradbeni entitet znanja u poglavlju 6.1.1.3. U ovom poglavlju opisano je značenje i varijacije pojedinih atributa entiteta znanja ovisno o dijelu konstrukcijskog znanja te struktura klasa i struktura zapisa znanja semantikom STEP standarda.

Na slici 7-1 prikazana je struktura zapisa znanja o znanju STEP standardom[63]. Prikazana struktura opisuje znanje o znanju kao proizvod određen preko entiteta *product*. Definicija znanja o znanju ostvarena je preko *product_definition* entiteta i vezana je prema *product* entitetu preko entiteta *product_definition_formation*. *Product_definition* entitet definira znanje o znanju u zadanom kontekstu. Kontekst definicije određen je entitetom *product_definition_context* i postavljen je na vrijednost konteksta koji određuje funkciju proizvoda. Funkcija proizvoda određena je vrijednošću atributa *name product_definition_context* entiteta. Korisnik prilikom kreiranja znanja o znanju može zadati funkciju izborom iz predodređenih vrijednosti ili može kreirati novu. Vrijednost atributa *life_cycle_stage product_definition_context* entiteta postavlja se na vrijednost **funkcija**. Opis zadane funkcije proizvoda upisuje se u *description* atribut *product_definition_formation* entiteta. Drugi entitet koji određuje kontekst proizvoda je *product_context*. STEP entitet *product_context* deriviran je iz *application_context_element* entiteta kao i *product_definition_context* entitet. Vrijednost atributa *name product_context* entiteta postavljena je na vrijednost **znanje_o_znanju**.

Budući da funkcija ovisi o korisnikovoj percepciji proizvoda u određenom okruženju postoji mogućnost definiranja različitih funkcija za isti proizvod. Mogućnost definiranja različitih funkcija za isti proizvod ostvarena je preko verzija znanja o znanju tj. semantikom STEP-a definirana formacija[64] preko *product_definition_formation* entiteta. Veza prema konstrukcijskim znanjima ostvarena je preko entiteta *next_assembly_usage_occurence*. Ovaj entitet je deriviran iz entiteta *assembly_component_usage* i ostvaruje vezu između pojedinog konstrukcijskog znanja i znanja o znanju preko atributa *relating_product_definition*, koji predstavlja pokazivač na znanje o znanju, i *related_product_definition*, koji predstavlja pokazivač na instancu konstrukcijskog znanja.

Svako definirano konstrukcijsko znanje ostvaruje vezu prema znanju o znanju i na taj način se preko znanja o znanju može pregledati instance konstrukcijskog znanja definirane za određeni proizvod. Znanje o znanju prvenstveno služi u organizaciji konstrukcijskog znanja, a i kao nositelj informacija o kreiranim instancama konstrukcijskog znanja (koje su kreirane, kako se nazivaju, tko ih je kreirao, kada, kada su zadnji put mijenjane, od koga itd.). Dijelovi

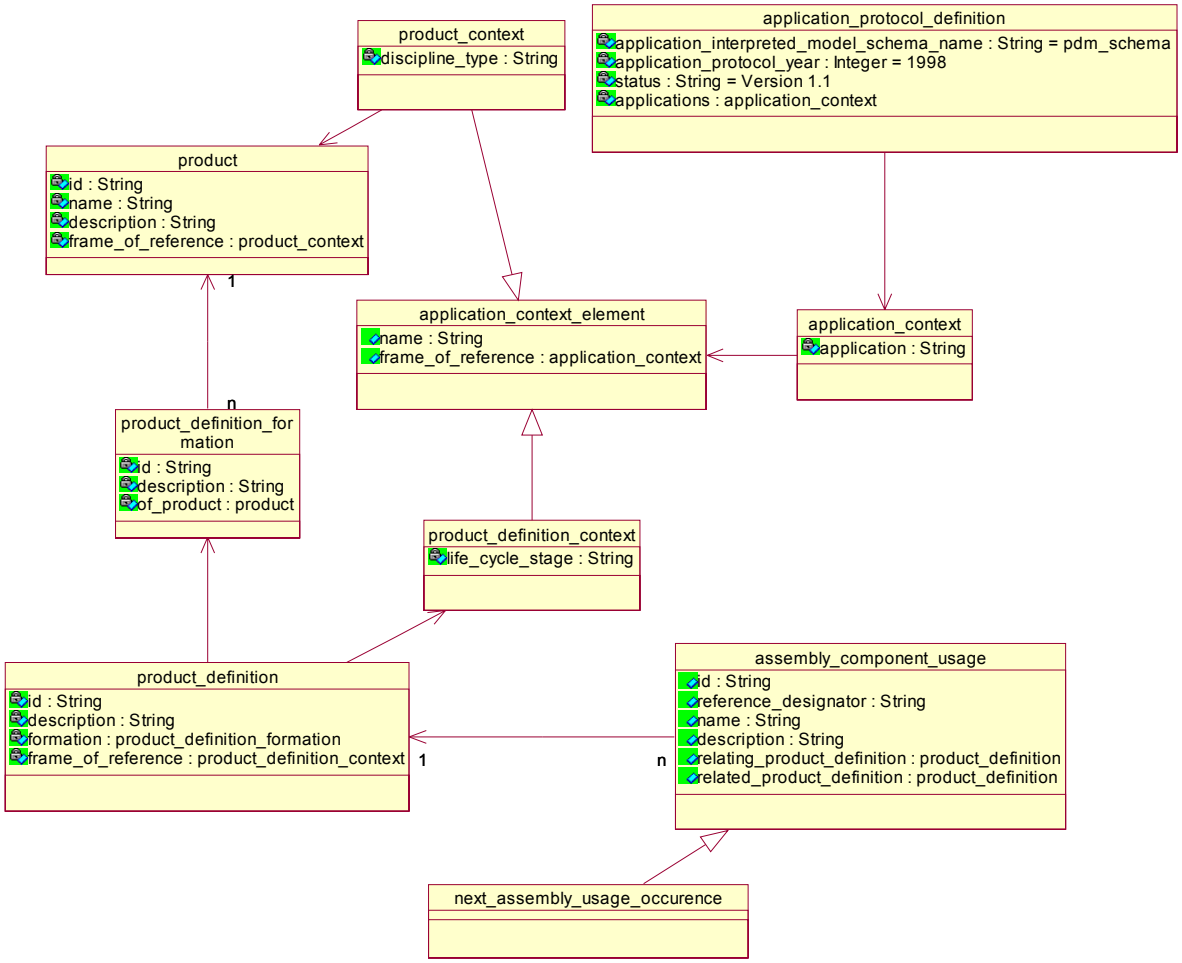
[63] U radu će se pod pojmom "zapis STEP standardom" podrazumijevati "zapis u skladu sa semantikom STEP standarda"

[64] Formacija je pojam koji se odnosi na STEP zapis, odnosno, entitet *product_definition_formation* pomoću kojega se definiraju različite inačice proizvoda. U kontekstu definiranja konstrukcijskog znanja, pojam formacija se odnosi na definicije različitih verzije istog tipa znanja.

informacija koji se koriste u okviru znanja o znanju preklapaju se sa informacijama dostupnim preko operativnog znanja. No, zbog postizanja što jednostavnijeg i fleksibilnijeg modela znanja o znanju, dijelovi znanja koji se koriste kroz mehanizme znanja o znanju implementirani su u operativno znanje.

Prilikom kreiranja znanja o znanju, za definirano znanje, povezuje se vrijeme i datum kreiranja te informacije o korisniku koji je kreirao znanje. Kreirano znanje o znanju tj. pojedine formacije povezuju se s informacijama o sigurnosti pristupa i statusu formacije. Na slici 7-1 nisu prikazani STEP entiteti koji su dio operativnog znanja a vežu se na znanje o znanju. Neprikazani entiteti odnose se na znanje o osobi i organizaciji te znanje o pravima, sigurnosti i statusu. Navedeni entiteti biti će objašnjeni naknadno.

Na razini zapisa znanja o znanju definirani su i atributi STEP entiteta koji vrijede za sve zapise dijelova znanja.

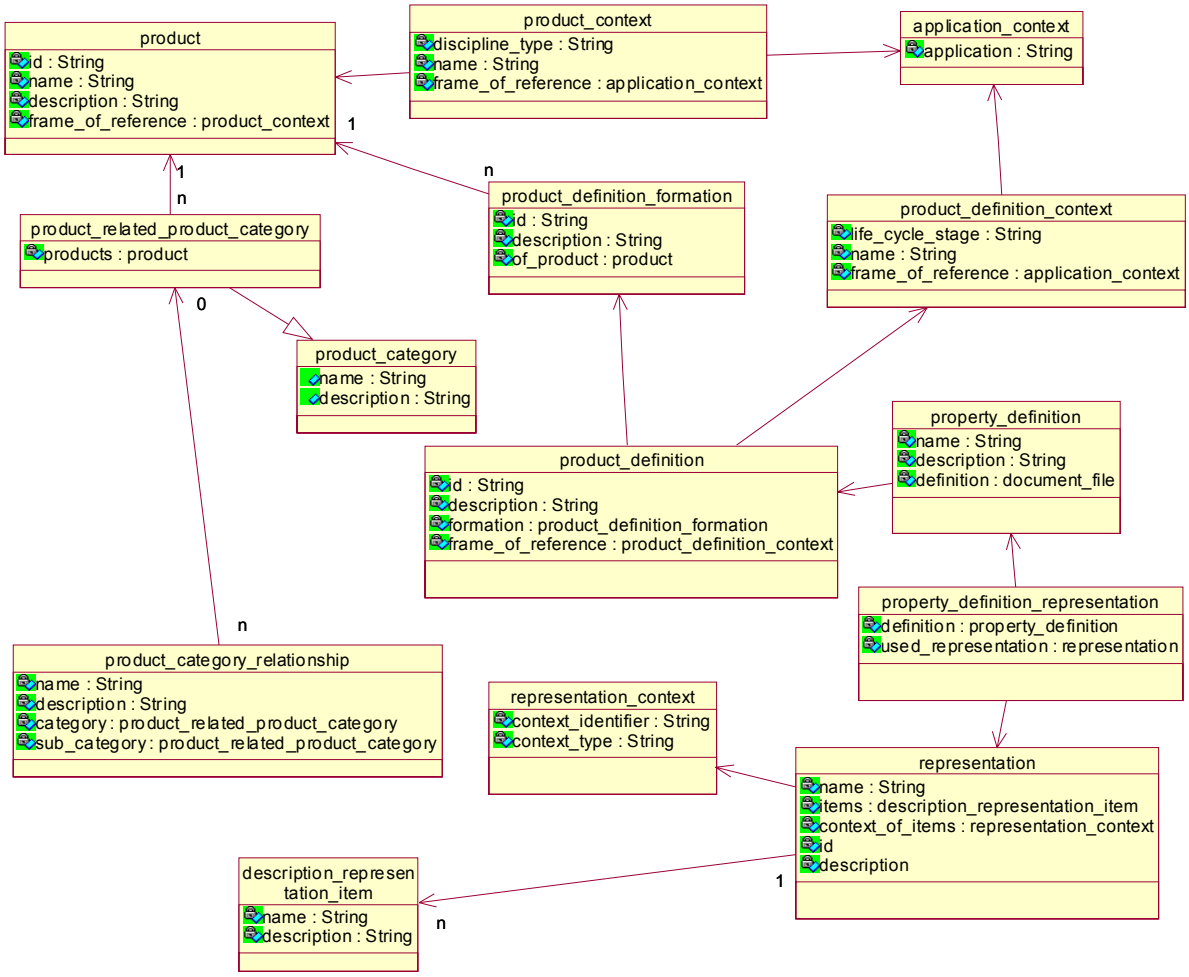


Slika 7-1 STEP Struktura zapisa znanja o znanju

Atributi STEP entiteta[65], odnosno atributi entiteta znanja o znanju i konstrukcijskog znanja, koji se odnose na sve dijelove zapisa su:

[65] Vidi poglavlje 4

- *application_context*
  - *application* – **mehaničke_konstrukcije** ("mechanical design")
- entitet *application_protocol_definition* koji definira atribute protokola:
  - *application_iterpreted_model_schema_name* – **pdm_schema** – naziv EXPRESS[66] sheme
  - *application_protocol_year* – **1998** - godina izdavanja sheme
  - *status* – **Version 1.1** – verzija sheme



Slika 7-2 STEP Struktura zapisa konstrukcijskog znanja

Na slici 7-2 prikazana je struktura zapisa konstrukcijskog znanja STEP standardom. Veza između znanja o znanju i konstrukcijskog znanja ostvarena je preko entiteta *next_assembly_usage_occurence* određenog u definiciji znanja o znanju. Konstrukcijsko znanje definirano je preko entiteta *product*, ali u drugačijem kontekstu u odnosu na znanje o znanju. Vrijednost atributa *name product_context* entiteta postavljena je na vrijednost **konstrukcijsko_znanje**. Preko *product_related_product_category* definirane su kategorije i pod kategorije konstrukcijskog znanja. Veza između kategorija i pod kategorija ostvaruje se

[66] Vidi [132], [133]

uporabom *product_category_relationship* entiteta. Kategorije u predloženom modelu podijeljene su prema dijelovima konstrukcijskog znanja na:

- **elementi**
- **relacije**
- **pravila**
- **postavke**
- **ograničenja**
- **dokumenti**
- **aplikacije**

Za kategoriju konstrukcijskog znanja – pravila moguće je kreirati nekoliko formacija. Svaka formacija ovisi o postavki *name* atributa *product_context* entiteta. Vrijednost *name* atributa postavlja se na naziv ponašanja koje se pridružuje pojedinoj verziji proizvoda. Dok se vrijednost *discipline_type* atributa *product_context* entiteta postavlja na vrijednost **ponašanje**. Ponašanje može biti jednako za sve formacije ili se može razlikovati od formacije do formacije. Korisnik ponašanje proizvoda može odrediti odabirom iz ponuđenih mogućnosti ili zadavanjem. Prilikom zadavanja ponašanja proizvoda opis se upisuje u *description* atribut *product_definition_formation* entiteta od strane korisnika, dok se prilikom odabira opis upisuje automatski od strane sustava. Različite formacije definicije konstrukcijskog znanja – pravila određuju se uporabom entiteta *product_definition_formation* (veza između definicije konstrukcijskog znanja – pravila i određene verzije proizvoda) i *product_definition* koji određuje formaciju proizvoda. Prilikom određivanja formacije proizvoda nulta formacija je rezervirana za globalne postavke, dok se ostale formacije kreiraju i imenuju od strane korisnika.

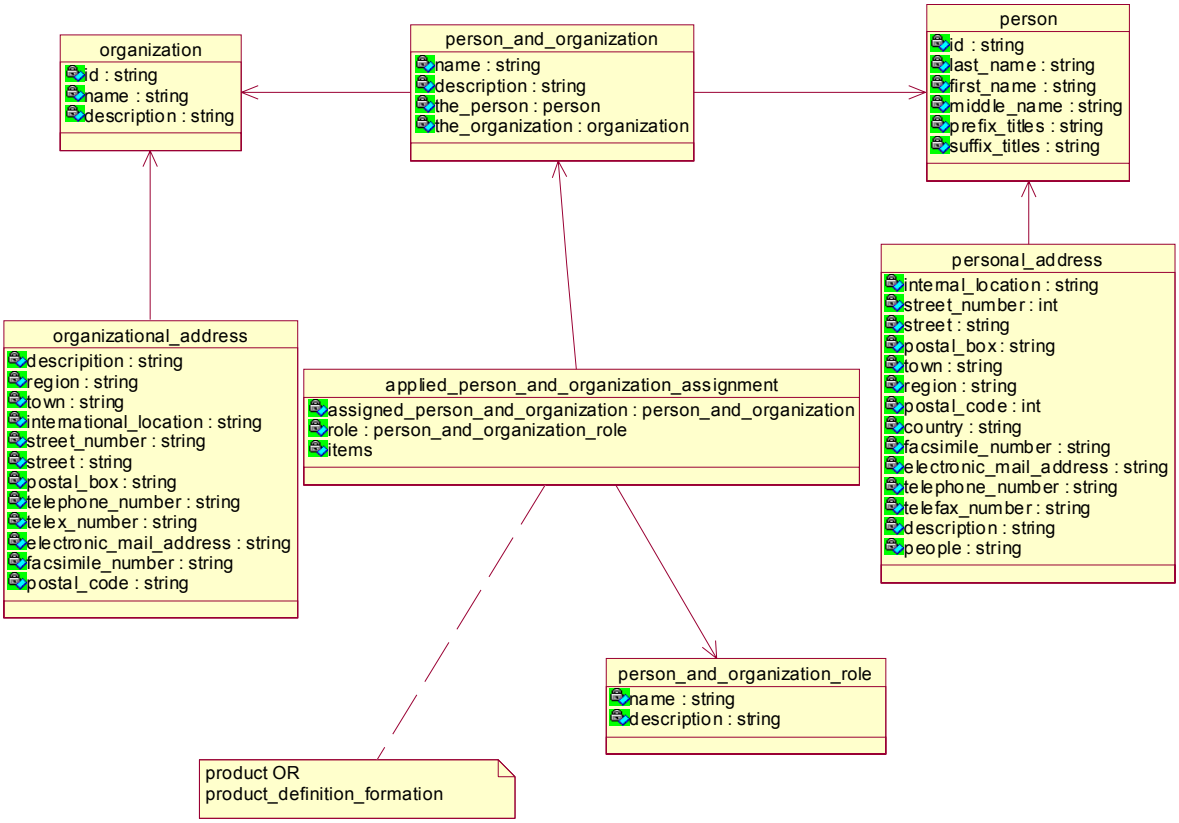
Gradbeni entiteti znanja postavljaju se uporabom *description_representation_item* STEP entiteta. Ovaj STEP entitet sadrži dva atributa: *name* i *description*. Atributu *name* pridružuje se vrijednosti entiteta znanja definiranih u poglavlju 6.1.1.3. Drugi atribut STEP entiteta služi kao dodatni opis ili opaska vezana za zadanu vrijednost *name* atributa. Budući da je standardom omogućeno kreiranje više instanci *description_representation_item* entiteta izraz entiteta znanja slaže se od svih kreiranih instanci redoslijedom navedenim u *representation* STEP entitetu. Zapis entiteta znanja kreira se u kontekstu definiranom *representation_context* entitetom. Atributu *context_type* mogu se pridružiti sljedeće vrijednosti:

- **parametri_relacija**
- **parametri_pravila**
- **parametri_elemenata**
- **parametri_postavki**
- **parametri_dokumenta**
- **parametri_ograničenja**
- **parametri_aplikacije**

Prikaz tj. zapis gradbenih entiteta znanja u zadanom kontekstu povezan je s definicijom znanja preko *property_definition_representation* i *property_definition* STEP entiteta. Entitet *property_definition* zadan je kao svojstvo definicije konstrukcijskog znanja tj. vezan je na *product_definition* STEP entitet.

Uporabom STEP entiteta *product_definition_formation* mogu se kreirati različite formacije pojedinih definicija znanja u sustavu. Kreiranje formacija znanja nije pod direktnom kontrolom korisnika, već sustav kreira nove instance definicije tj. formacije pojedinog znanja. Nove formacije se kreiraju u slučaju da već zadana formacija nije potvrđena od strane, u tu svrhu kreiranih, korisnika, ili u slučaju promjene već potvrđene formacije, koja nije odobrena ili je u postupku odobravanja.

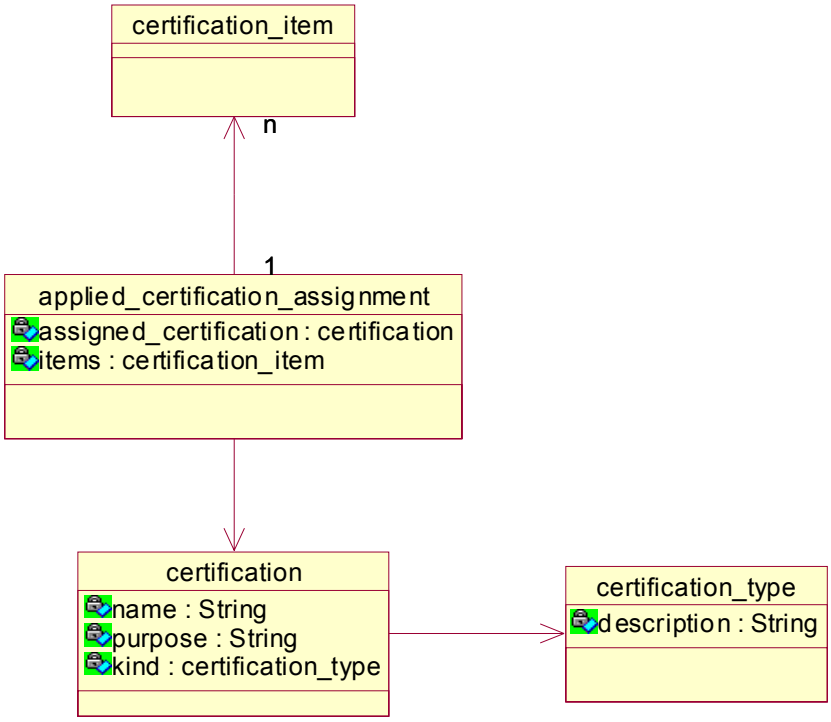
Prilikom kreiranja konstrukcijskog znanja, povezuju se, za kreirano znanje, vrijeme i datum kreiranja, te informacije o korisniku koji je kreirao znanje. Kreirano konstrukcijsko znanje tj. pojedine formacije povezuju se s informacijama o sigurnosti pristupa i statusu formacije. Na slici 7-2 nisu prikazani STEP entiteti koji su dio operativnog znanja, a vežu se za konstrukcijsko znanje. Neprikazani entiteti odnose se na znanje o osobi i organizaciji, te znanje o pravima, sigurnosti i statusu. Navedeni entiteti biti će objašnjeni u kasnijem dijelu rada.



Slika 7-3 STEP struktura zapisa informacija o organizaciji i osobi

Informacije o osobi i organizaciji su dio operativnog znanja. Na slici 7-3 prikazana je STEP struktura zapisa informacija o osobi i organizaciji. Struktura se sastoji od dijelova vezanih za informacije o osobi (ime, prezime, adresa, ...) i informacije o organizaciji ili odjelu u organizaciji tj. tvrtci u kojoj je osoba zaposlena. U daljnjem tekstu pod nazivom korisnik podrazumijevat će se postojanje informacija o osobi i organizaciji. Inicijalno postoji predodređen samo jedan korisnik **upravitelj_znanja**. "Upravitelj znanja" ima pravo pristupa svim dijelovima sustava i svim dijelovima proširenog CAD modela. Zadaća **upravitelja_znanja** je kreiranje i upravljanje korisnicima (unos svih relevantnih informacija vezanih za korisnike, mijenjanje i brisanje istih). Osim unosa informacija vezanih za korisnika (osoba i organizacija)

upravitelj_znanja dodjeljuje i mijenja ovlasti (certifikate) korisnicima te postavlja faktore sigurnosti na pojedine dijelove znanja i kontrolira status znanja. Entitet *person_and_organization* veže se na *product_definition_formation* entitet znanja o znanju ili svaku instancu konstrukcijskog znanja. Korisnik se također povezuje sa stupnjem odobrenjima tj. statusom pojedinih znanja. Dodatni opis korisnika i njegove uloge u projektu, tvrtci ili konstrukcijskom uredu moguć je preko atributa *name* (simboličkog imena) i *description* (opis uloge korisnika) *person_and_organization_role* entiteta.

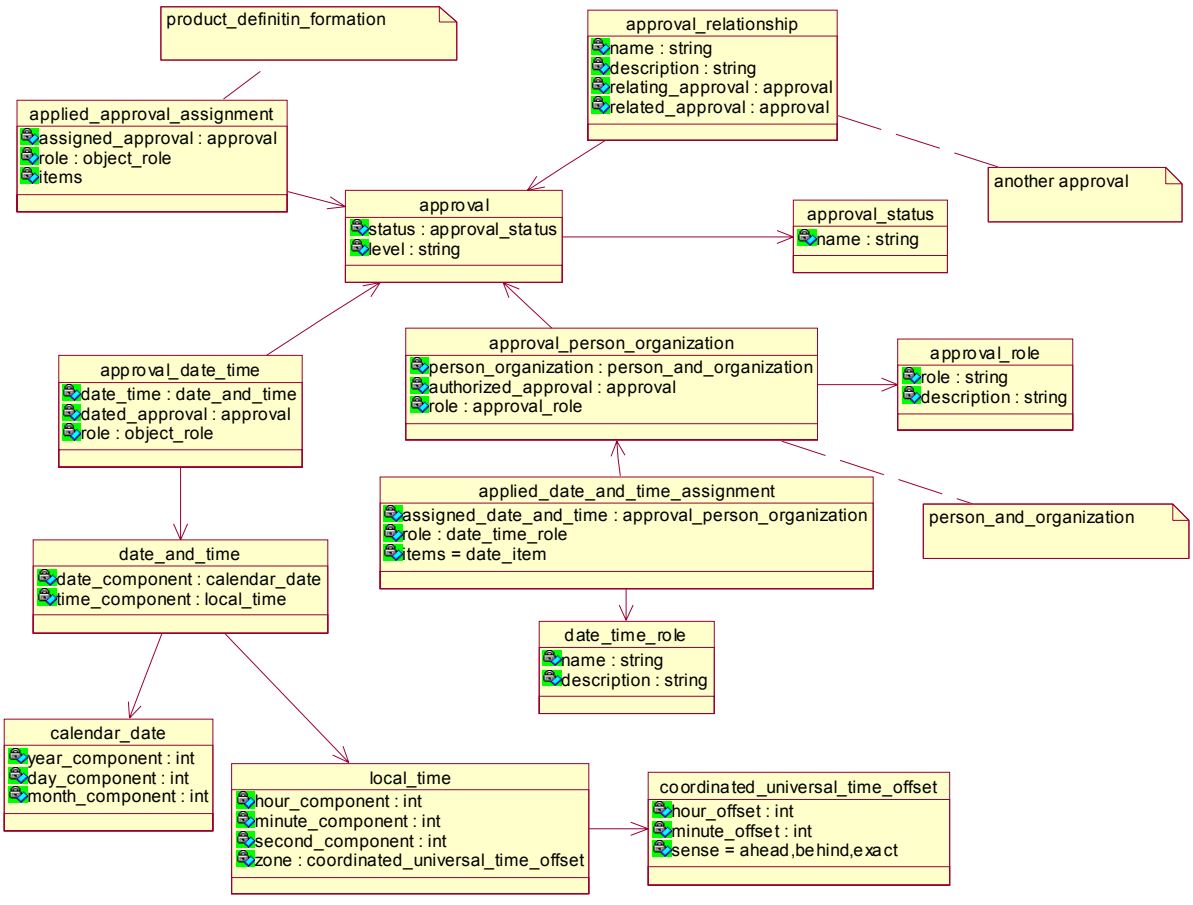


Slika 7-4 STEP struktura prava odnosno certifikata

Svakom kreiranom korisniku pridružene su ovlasti koje mu omogućuju obavljanje različitih akcija u sustavu (kreiranje i promjena znanja, brisanje znanja, promjena odobrenja, itd.). STEP struktura zapisa prava odnosno certifikata korisnika prikazana je na slici 7-4. Ovlasti korisniku dodjeljuje **upravitelj_znanja** prilikom kreiranja korisnika. Korisniku se mogu dodijeliti sljedeće ovlasti:

- **konstruktor** – osnovni oblik korisnika:
  - konstruktor može kreirati i mijenjati znanja koja imaju stupanj sigurnosti **slobodno** i **ograničeno**,
  - nema ovlasti kreiranja drugih korisnika niti promjene njihovih ovlasti,
  - nema ovlasti promjene stupnja sigurnosti znanja,
  - nema ovlasti mijenjati stupanj odobrenja tj. status znanja,
- **stariji_konstruktor** – viši oblik korisnika:
  - stariji konstruktor ima ovlasti kreirati i mijenjati znanja koja imaju stupanj sigurnosti **slobodno**, **ograničeno** i **povjerljivo**,
  - ima ovlasti mijenjanja stupnja sigurnosti do razine **povjerljivo**,
  - ima ovlasti mijenjati stupanj odobrenja tj. status znanja
  - nema ovlasti kreiranja drugih korisnika niti promjene njihovih ovlasti,

- **voditelj_konstrukcije** – najviši oblik korisnika:
  - ima ovlasti kreirati i mijenjati znanja koja imaju stupanj sigurnosti **slobodno**, **ograničeno**, **povjerljivo** i **tajna**
  - ima ovlasti mijenjanja stupnja sigurnosti do razine **tajna**,
  - ima ovlasti mijenjati stupanj odobrenja tj. status znanja
  - ima ovlasti kreiranja drugih korisnika i promjene njihovih ovlasti,
- **upravitelj_znanja** – određuje korisnika koji koristi sustav:
  - ukoliko je došlo do greške u radu sustava,
  - kada je potrebno inicijalno kreirati korisnika i/ili promijeniti informacije o njima,
  - ukoliko je neophodno napraviti "backup" sustava,
  - prilikom restauracije sustava,
  - prilikom resetiranja parametara sustava ili parametara proširenog CAD modela
- **gost** – korisnik koji ima ograničene ovlasti i posjeduje sljedeće prava:
  - pregledavati znanja koja imaju stupanj sigurnosti **slobodno**,
  - nema ovlasti mijenjanja parametara sustava niti parametara proširenog CAD modela.



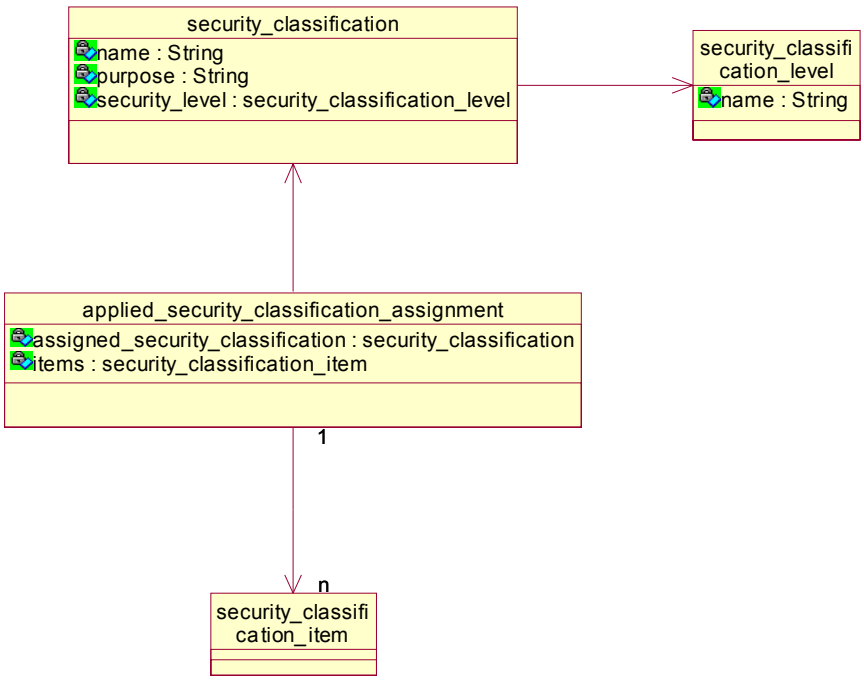
Slika 7-5 STEP struktura zapisa informacija o odobrenjima

Nakon kreiranja znanja, bilo znanja o znanju ili konstrukcijskog znanja, uz njega se veže i status znanja (stupanj odobrenja). Status znanja govori o uporabivosti znanja unutar proširenog CAD modela. Svakom znanju može se promijeniti status. Status mijenjaju korisnici koji posjeduju ovlasti (certifikate) za promjenu statusa znanja. STEP struktura statusa znanja

prikazana je na slici 7-5. Svako kreirano znanje može imati samo jedan status. Raspoloživi statusi znanja su sljedeći:

- **prijedlog** – prilikom kreiranja znanja, znanja o znanju ili konstrukcijskog znanja, status **prijedlog** automatski se dodijeli znanju od strane sustava, ova razina statusa određuje znanje koje se može koristiti samo od strane korisnika koji ga je kreirao, a pregledavati ga mogu svi korisnici,
- **u_postupku** – promjenu statusa znanja iz statusa **prijedlog** na status **u_postupku** mogu ostvariti samo korisnici koji imaju prava za tu operaciju; znanje koje ima status **u_postupku** može se koristiti od strane svih korisnika tijekom rada, ali ne smije postojati u završnoj verziji proizvoda,
- **odobreno** – odobrenje znanja mogu ostvariti **stariji_konstruktor** do razine **povjerljivo** i **voditelj_konstrukcije** na svim razinama, **odobreno** je krajnji status znanja, znanje koje je uputno koristiti u radu,
- **odbijeno** – status određuje znanje koje nije preporučljivo koristiti u radu i koje će se obrisati.

Status znanja se upisuje u *name* atribut *approval_status* entiteta. Osim informacije o statusu znanja, uz strukturu odobrenja vezane su i informacije o korisniku koji je promijenio ili zadao pojedino odobrenje tj. status, te vrijeme i datum promjene. U strukturi je ostavljena mogućnost naknadnog proširenja sustava odobrenja dodavanjem dodatnih statusa. Proširenje se može ostvariti pomoću *approval_relationship* entiteta i dodatne definicije *approval* entiteta. Atribut *name date_time_role* entiteta ima vrijednost **vrijeme_odobrenja**. Atribut *name approval_role* entiteta ima vrijednost **status_znanja**. Status, odnosno *approval* entitet, veže se za *product_definition_formation* entitet znanja o znanju i konstrukcijskog znanja preko *applied_approval_assignment* entiteta. Dodatni opis uloge odobrenja može se upisati preko atributa *role* i *description approval_role* entiteta.



Slika 7-6 STEP struktura zapisa sigurnosti

Stupanj sigurnosti kreiranog znanja određuje pristup i uporabu znanja od strane korisnika. STEP struktura zapisa sigurnosti prikazana je na slici 7-6. Mogući stupnjevi sigurnosti su sljedeći:

- **slobodno** – pristup znanju je potpuno slobodan; svi definirani korisnici sustava imaju potpunu slobodu za pregledavanje i promjenu bilo kojeg dijela znanja,
- **ograničeno** – pregledavanje i uporaba znanja je dozvoljena svim korisnicima osim **gost** korisniku; promjena i brisanje znanja je ograničena na kreatora znanja i korisnike s ulogama tj. certifikatom za navedene operacije,
- **povjerljivo** – pregledavanje, promjena i uporaba znanja dozvoljena je samo korisnicima s određenim ulogama (**stariji_konstruktor**, **voditelj_konstrukcije**),
- **tajno** – pristup znanjima u potpunosti dozvoljen je samo korisniku **voditelj_konstrukcije**, dok je korisniku **stariji_konstruktor** dozvoljeno samo pregledavanje.

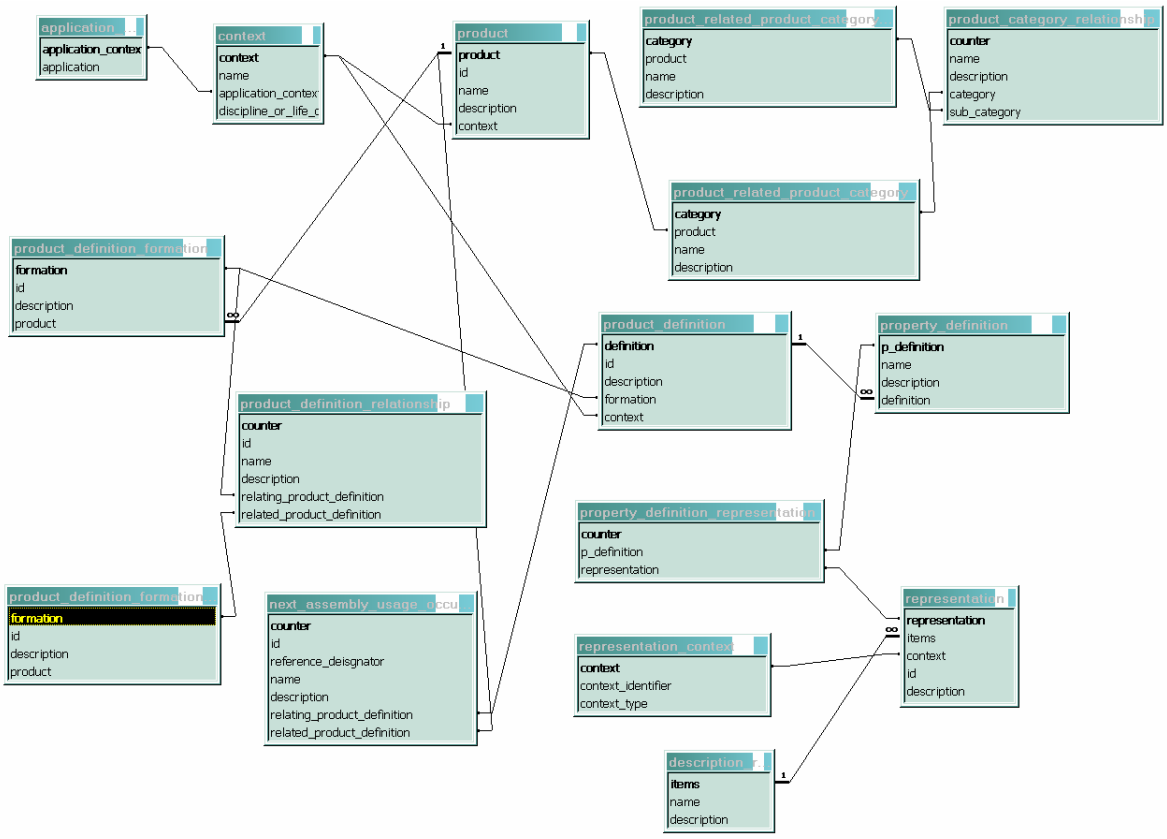
Vrijednost stupnja sigurnosti određena je vrijednošću atributa *name security_classification_level* entiteta. Dodatni opis stupnja sigurnosti moguć je uporabom atributa *purpose security_classification* entiteta. Postavljanje stupnja sigurnosti moguće je automatski od strane sustava u postupku kreiranja znanja ili na zahtjev korisnika. Sigurnost se veže za strukturu znanja o znanju i strukturu konstrukcijskog znanja preko *product_definition_formation* i *security_classification_item* entiteta. Sigurnost kao i status mogu se vezati za svaku formaciju znanja.

7.1.1 Struktura baze podataka

Na osnovu predložene STEP strukture zapisa podataka znanja o znanju, konstrukcijskom znanju, odobrenju, sigurnosti i ovlastima, razvijena je struktura tablica i relacija u bazi podataka. Prikaz relacija između tablica razložen je u pet dijelova zbog jasnoće prikaza. Prilikom kreiranja tablica i relacija u bazi podataka ostvareno je potpuno preslikavanje strukture STEP zapisa u bazu. Razlike između STEP zapisa i zapisa u bazi su na razini imenovanja pojedinih atributa entiteta.

U tablice baze podataka upisuju se vrijednosti atributa proširenog CAD modela. Svaki prošireni CAD model može imati jedan generički zapis i više instanci. Zbog fleksibilnijeg zapisa podataka atributi svake instance zapisuju se zasebno i nisu direktno vezani za generički zapis. Svaka promjena napravljena na zapisu instance ne utiče na druge instance niti na generički zapis, ali svaka promjena generičkog zapisa rezultira kreiranjem upozorenja korisniku i mogućim (ovisno o odluci korisnika) promjenama u zapisu instance.

Generiranje upozorenja nije ugrađeno u zapis podataka u bazi nego se ostvaruje programski. Objekt klase zadužene za upravljanje i manipulaciju zapisima u bazi, svaki put, prilikom zapisivanja ili čitanja podataka, kontrolira stanje generičkog zapisa te ukoliko je došlo do promjene, generira poruku upozorenja korisniku. Provjera generičkog zapisa se ostvaruje preko tablice promjena. U tablici promjena upisuju se podaci o promjenama generičkog zapisa. Osim osnovnih tablica (tablice STEP entiteta) i tablice promjena, u bazi podataka, nalaze se i dodatne tablice koje služe kao nositelj informacija potrebnih za ispravan rada sustava.

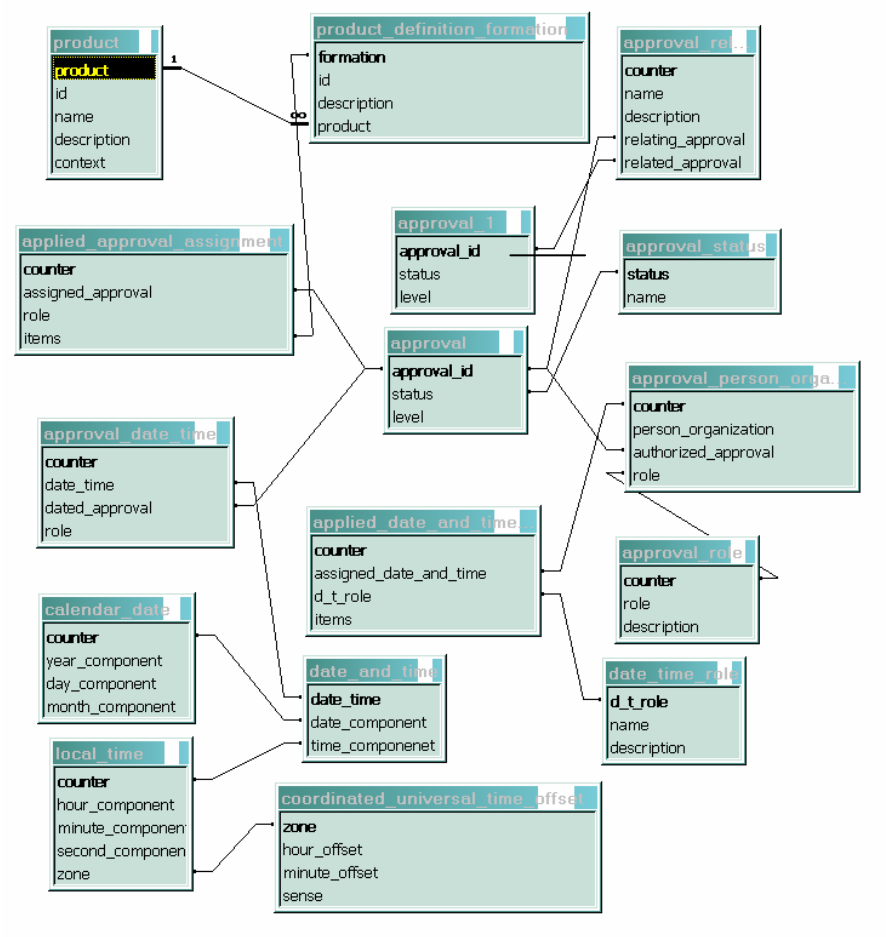


Slika 7-7 Struktura tablica zapisa znanja o znanju i konstrukcijskog znanja u bazi

Na slici 7-7 prikazana je struktura tablica i relacija za zapis znanja o znanju i zapis konstrukcijskog znanja. Obje strukture čine tablice definicije aplikacijskog konteksta te tablice definicije proizvoda (znanje o znanju ili konstrukcijsko znanja) i tablice definicije konteksta proizvoda.

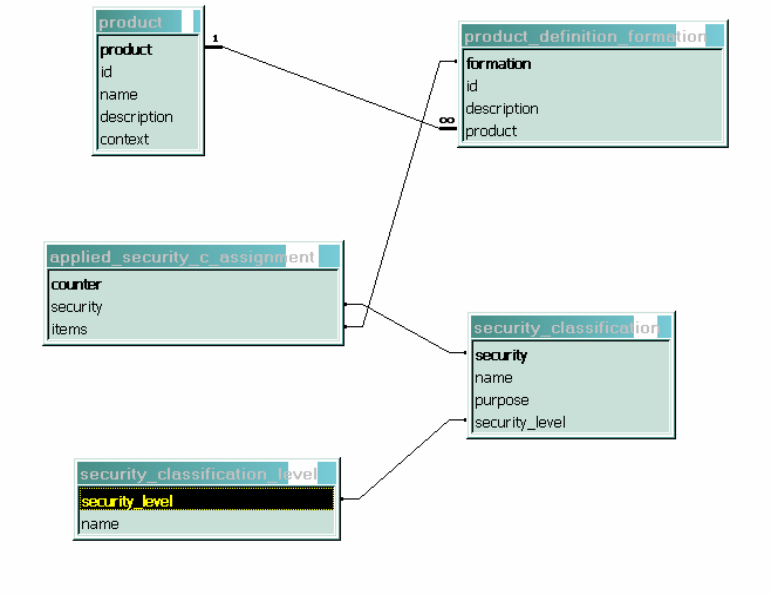
Budući da znanje o znanju i konstrukcijsko znanje posjeduju slične strukture zapisa podataka obje strukture prikazane su na slici 7-7. Struktura zapisa znanja o znanju sastoji se od dodatnih tablica koje služe za zapis podataka o konstrukcijskim znanjima vezanim za znanje o znanju. S druge strane konstrukcijsko znanje posjeduje dodatne tablice definicije kategorije i potkategorija te tablice za definiciju svojstava. Tablice za definiciju svojstava koriste se za zapis definicije konstrukcijskog znanja.

Struktura zapisa odobrenja prikazana je na slici 7-8. Osim prikaza strukture prikazana je i veza između odobrenja i znanja o znanju ili konstrukcijskog znanja. Veza je ostvarena preko tablice (STEP entiteta) *product_definition_formation*. Navedena tablica je nositelj informacija o odobrenjima vezanim na pojedine formacije znanja. Kao što je vidljivo iz slike, na zapis odobrenja vežu se i informacije o datumu i vremenu promjene ili zadavanja odobrenja, te informacije o korisniku tj. osobi i organizaciji kojoj osoba pripada, a koja je promijenila ili zadala odobrenje.



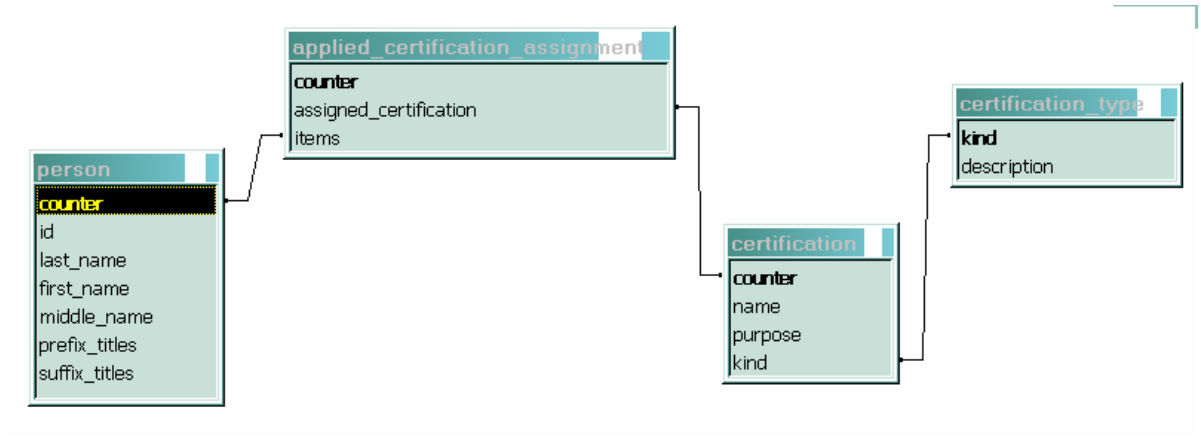
Slika 7-8 Struktura tablica i relacija za odobrenja

Na slici 7-9 prikazana je struktura tablica i relacija za zapis sigurnosti. Stupanj sigurnosti određuje mogućnost pristupa ili uporabe pojedinog znanja. Iz slike je također vidljivo na koji način se sigurnost veže za zapis znanja. Veza je ostvarena preko tablice tj. STEP entiteta *product_definition_formation*, što znači da svaka formacija znanja može imati vlastiti zadani stupanj sigurnosti.



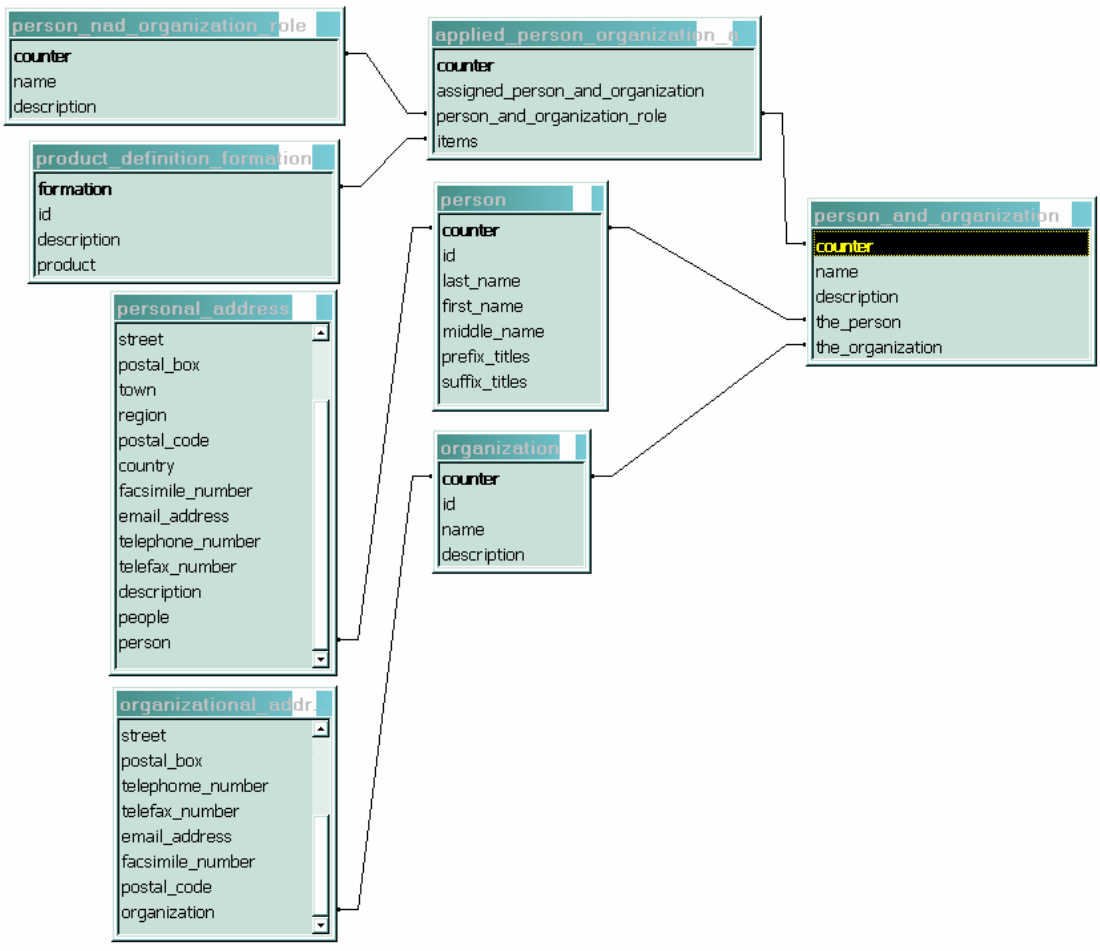
Slika 7-9 Struktura tablica i relacija za sigurnost

Na slici 7-10 prikazana je struktura tablica i relacija za zapis ovlasti korisnika. Relacijom je tablica *applied_certification_assignment* vezana za tablicu *person* i na taj način je ostvareno povezivanje ovlasti s korisnikom. Informacije o ovlastima jedne su od vitalnih informacija potrebnih za provjeru pristupa znanjima i manipulaciju istima.



Slika 7-10 Struktura tablica i relacija za ovlasti

Struktura tablica i relacija za zapis informacija o korisniku prikazana je na slici 7-11. Informacije o korisniku podijeljene su na informaciji o osobi i informacije o organizaciji u kojoj osoba radi. Tablica tj. STEP entitet *applied_person_organization_a* koristi se kao veza korisnika na ostale tablice tj. STEP entitete.



Slika 7-11 Struktura tablica i relacija za korisnika

Za svako kreirano znanje ili za promjenu istoga vezane su informacije o korisniku koji je proveo određenu operaciju. Informacije o korisniku koriste se i prilikom zadavanja odnosno promjene sigurnosti, odobrenja i ovlasti.

Svaka promjena znanja za generički model rezultira zapisom u tablici promjena. Podaci uneseni u tablicu promjena služe kao osnova za propagiranje promjena na instance. Tablica promjena sadrži sljedeća polja:

- *brojac* – automatski brojač redaka
- *id_znanja* – identifikacijska oznaka promijenjenog znanja

Dodatne tablice, koje služe kao nositelj informacija potrebnih za inicijalizaciju sustava, imaju sljedeću strukturu:

- tablica ovlasti_ini
  - *brojac* – automatski brojač redaka
  - *ovlasti* – moguće ovlasti u sustavu
  - *opis* – opis svake pojedine ovlasti
- tablica sigurnosti_ini
  - *brojac* – automatski brojač redaka
  - *stupanj_sigurnosti* – mogući stupnjevi sigurnosti
  - *opis* – opis svakog stupnja sigurnosti
- tablica odobrenja_ini
  - *brojac* – automatski brojač redaka
  - *stupanj_odobrenja* – mogući stupnjevi odobrenja
  - *opis* – opis svakog stupnja odobrenja
- tablica kategorija_ini
  - *brojac* – automatski brojač redaka
  - *kategorija_znanja* – kategorije konstrukcijskog znanja u sustavu
  - *opis* – kratak opis pojedine kategorije
- tablica konteksta_prikaza_ini
  - *brojac* – automatski brojač redaka
  - *kontekst_prikaza* – kontekst prikaza konstrukcijskog znanja
  - *opis* – kratak opis konteksta

7.1.2 Struktura klasa modela

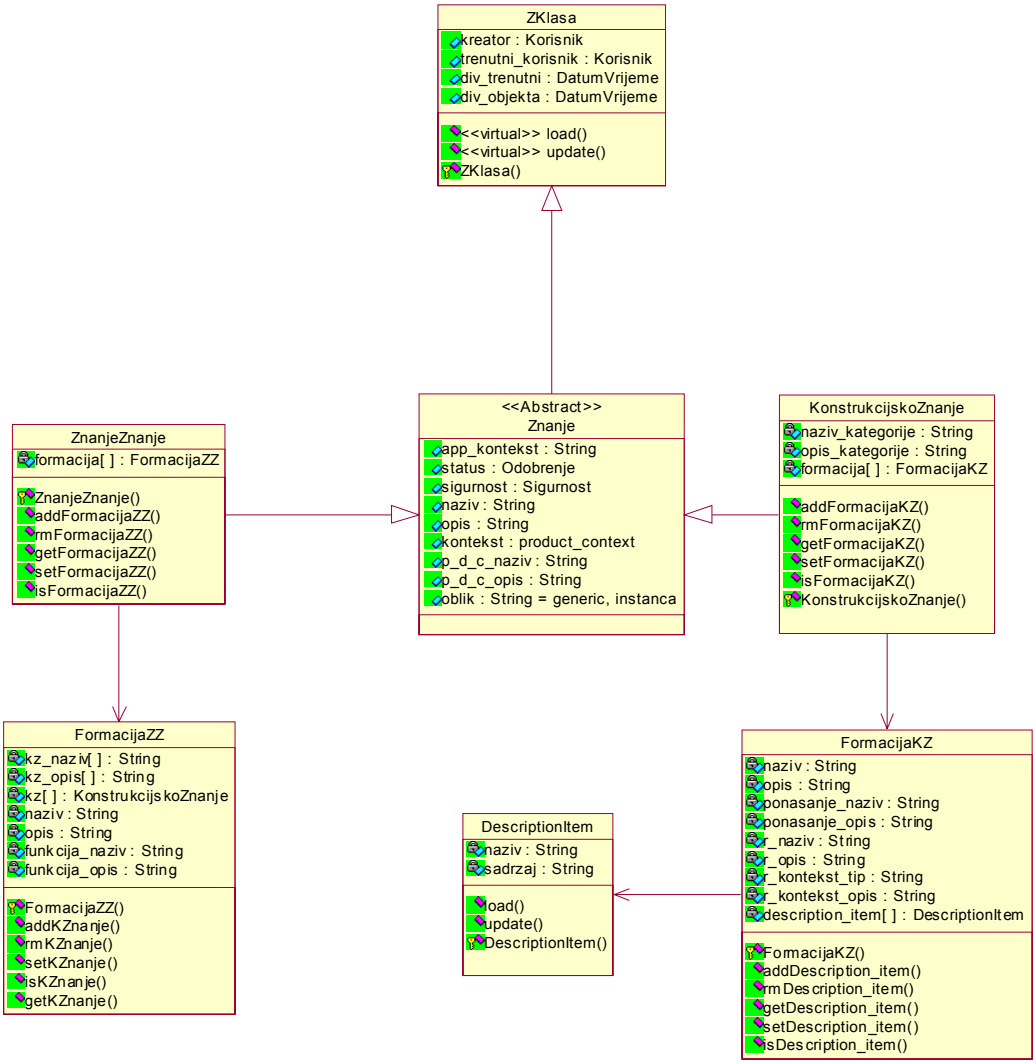
U prethodnim poglavljima opisana struktura zapisa znanja u ovom poglavlju prikazat će se uporabom objektnog pristupa. Objektno orijentirani pristup modeliranju konstrukcijskog znanja omogućuje prirodnu dekompoziciju i kreiranje hijerarhijske strukture znanja. Realizacija strukture klasa predloženog modela kreirana je u skladu sa pravilima JAVA [134], [135], [136] programskog jezika. Budući da je ciljana platforma za implementaciju proširenog CAD modela programska aplikacija ProEngineer® i J-Link®[137] kao veza između aplikacije i modela implicira se uporaba JAVA programskog jezika. Odabir JAVA programskog jezika kao osnove za prikaz klasa njihove strukture, atributa i metoda, ni u kom slučaju ne ograničava implementaciju modela u drugim objektno orijentiranim programskim jezicima. Jedna od činjenica koja podržava ovu tvrdnji je i povijest tj. zaleđe JAVA programskog jezika i bliskost sa C++[138] programskim jezikom.

Na slici 7-12 prikazana je struktura klase ***ZnanjeZnanje*** i ***KonstrukcijskoZnanje*** te struktura klasa ***ZKlasa***, ***FormacijaZZ***, ***FormacijaKZ*** i ***DescriptionItem***. Klasa ***ZKlasa*** sadrži atribute i metode koje se referenciraju ili koristite u drugim klasama, instancira se samo jednom i to prilikom aktiviranja sustava. U klasi ***ZKlasa*** definirani su sljedeći atributi i metode:

- *kreator* – objekt tipa ***Korisnik*** koji sadrži informacije o kreatoru znanja,
- *trenutni_korisnik* – objekt tipa ***Korisnik*** koji sadrži informacije o korisniku koji trenutno koristi sustav i radi sa proširenim CAD modelom,
- *div_trenutni* – objekt tipa ***DatumVrijeme*** koji sadrži informacije o trenutnom datumu i vremenu,
- *div_objekta* – objekt tipa ***DatumVrijeme*** koji sadrži informacije o datumu i vremenu kreiranja instance znanja,
- *load()* – virtualna metoda koja služi za učitavanja podataka o znanju iz tablica u bazi podataka,
- *update()* – virtualna metoda koja služi za zapisivanje ili promjenu podatka o znanju u tablicama baze podataka.
- *ZKlasa()* – metoda konstruktor klase ***ZKlasa***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

Osnovna klasa ***Znanje*** definirana je kao apstraktna klasa što znači da se na osnovu nje ne kreiraju instance odnosno objekti. Klasa ***Znanje*** služi kao nositelj zajedničkih atributa i metoda deriviranih klasa ***ZnanjeZnanje*** i ***KonstrukcijskoZnanje***. Derivirane klase mehanizmom nasljeđivanja imaju pristup atributima i metodama nadređene klase. U klasi ***Znanje*** sadržane su definicije sljedećih atributa i metoda:

- *app_kontekst* – znakovna varijabla koja sadrži vrijednosti aplikacijskog konteksta,
- *status* – objekt tipa ***Odobrenje*** koji sadrži informacije o odobrenju vezanom za znanje,
- *sigurnost* – objekt tipa ***Sigurnost*** koji sadrži informacije o stupnju sigurnosti vezanom za znanje,
- *naziv* – znakovna varijabla koja sadrži simboličko ime kreiranog znanja,
- *opis* – znakovna varijabla koja sadrži kratak opis kreiranog znanja,
- *kontekst* – znakovna varijabla koja sadrži vrijednost konteksta znanja,
- *p_d_c_naziv* – znakovna varijabla koja sadrži kontekst definicije instanciranog znanja,
- *p_d_c_opis* – znakovna varijabla koja sadrži kratak opis konteksta definicije instanciranog znanja,
- *oblik* – znakovna varijabla koja sadrži podatak o obliku modela, oblik može biti **generic** ili **instanca**.



Slika 7-12 Struktura klasa znanja o znanju i konstrukcijskog znanja

Navedeni atributi i metode dostupni su u klasama ***ZnanjeZnanje*** i ***KonstrukcijskoZnanje***. Prilikom instanciranja objekata navedenih klasa, instanciraju se i opisani atributi kojima se postave početne vrijednosti preko poziva konstruktora instancirane klase. Klasa ***ZnanjeZnanje*** i klasa ***KonstrukcijskoZnanje*** definiraju vlastite metode **load** i **update.** Definirane metode su prilagođene zapisu podataka znanja koje predstavlja određena klasa. Definiranje vlastitih metoda unatoč definiciji istih u nadređenoj klasi, moguće je iz razloga što su navedene metode definirane kao virtualne. Ukoliko je metoda deklarirana kao virtualna tada se implementacija metode može razlikovati od jedne do druge derivirane klase. Također je moguće i kreiranje više različitih metoda istog imena koje, ovisno o tipu i broju atributa metoda, mogu izvršavati različite zadaće. Ostvarenje navedenog je moguće preopterećivanjem metoda.

U klasi ***ZnanjeZnanje*** definirane su metode za upravljanje i manipulaciju informacijama znanja o znanju. Metode i atributi definirani u klasi ***ZnanjeZnanje*** su sljedeći (Slika 7-12):

- *ZnanjeZnanje()* – metoda konstruktor klase ***ZnanjeZnanje***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta,

- *addFormacijaZZ()* – metoda služi za instanciranje novog objekta klase ***FormacijaZZ*** i dodavanje istog u polje formacija,
- *rmFormacijaZZ()* – metoda služi za brisanje zadane instance klase ***FormacijaZZ*** iz polja formacija,
- *getFormacijaZZ()* – metoda služi za pretraživanje polja formacija i odabir te postavljanje radne formacije,
- *setFormacijaZZ()* – metoda služi za postavljanje vrijednosti atributa zadane formacije ili promjenu istih,
- *isFormacijaZZ()* – metoda služi za saznavanje trenutne radne formacije,
- *formacije[]* – polje pokazivača na objekte tipa ***FormacijaZZ.***

U klasi ***KostrukcijskoZnanje*** definirane su metode za upravljanje i manipulaciju informacijama vezanim za konstrukcijsko znanje. Metode i atributi definirani u klasi ***KonstrukcijskoZnanje*** su sljedeći (Slika 7-12):

- *naziv_kategorije* – znakovna varijabla koja određuje kategoriju kreiranog konstrukcijskog znanja, kategorije se učitavaju iz tablice kategorija,
- *opis_kategorije* – znakovna varijabla koja sadrži kratak opis kategorije kreiranog konstrukcijskog znanja,
- *formacije[]* – polje pokazivača na objekte tipa ***FormacijaKZ***,
- *addFormacijaKZ()* – metoda služi za kreiranje nove instance formacije tipa **FormacijaKZ** i dodavanje iste u polje formacije,
- *rmFormacijaKZ()* – metoda služi za brisanje zadane instance iz polja formacija,
- *getFormacijaKZ()* – metoda služi za pretraživanje polja formacija i odabir zadane te postavljanje radne formacije,
- *setFormacijaKZ()* – metoda služi za postavljanje vrijednosti atributa zadane formacije ili promjenu istih,
- *isFormacijaKZ()* – metoda služi za saznavanje trenutne radne formacije,
- *KonstrukcijskoZnanje()* – metoda konstruktor klase ***KonstrukcijskoZnanje***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

Prilikom kreiranja nove instance klase ***ZnanjeZnanje*** kreira se barem jedna instanca klase ***FormacijaZZ***. Instance klase ***FormacijeZZ*** su definicije formacija klase ***ZnanjeZnanje***. U klasi ***FormacijaZZ*** definirane su sljedeće metode i atributi (Slika 7-12):

- *kz_naziv[]* – znakovno polje koje sadrži simbolička imena instanciranih objekata konstrukcijskog znanja,
- *kz_opis[]* – znakovno polje koje sadrži kratak opis instanciranih objekata konstrukcijskog znanja,
- *kz[]* – polje pokazivača na objekte tipa ***KonstrukcijskoZnanje***,
- *naziv* – znakovna varijabla koja sadrži simboličko ime instanciranog znanja o znanju,
- *opis* – znakovna varijabla koja sadrži kratak opis instanciranog znanja o znanju,
- *funkcija_naziv* – znakovna varijabla koja sadrži simboličko ime funkcije definirane od strane korisnika i vezane za instancu znanja o znanju,

- *funkcija_opis* – znakovna varijabla koja sadrži kratak opis funkcije definirane od strane korisnika i vezane za instancu znanja o znanju,
- *FormacijaZZ()* – metoda konstruktor klase ***FormacijaZZ***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta,
- *addKZnanje()* – metoda služi za kreiranje nove instance konstrukcijskog znanja i dodavanje iste u polje pokazivača na objekte tipa ***KonstrukcijskoZnanje***,
- *rmKZnanje()* – metoda služi za brisanje instance konstrukcijskog znanja,
- *setKZnanje()* – metoda služi za postavljanje vrijednosti atributa instance konstrukcijskog znanja,
- *getKZnanje()* – metoda služi za pretraživanje polja instanci i postavljanje radne instance konstrukcijskog znanja,
- *isKZnanje()* – metoda služi za saznavanje trenutne radne instance konstrukcijskog znanja.

Instanciranjem objekta klase ***KonstrukcijskoZnanje*** instancira se barem jedan objekt klase ***FormacijaKZ***. U klasi ***FormacijaKZ*** definirani su sljedeći atributi i metode (Slika 7-12):

- *naziv* – znakovna varijabla koja sadrži simboličko ime instance formacije,
- *opis* – znakovna varijabla koja sadrži kratak opis instance formacije,
- *ponasanje_naziv* – znakovna varijabla koja sadrži simboličko ime ponašanja definiranog od strane korisnika i vezanog za formaciju instance konstrukcijskog znanja – pravila,
- *ponasanje_opis* – znakovna varijabla koja sadrži kratak opis ponašanja definiranog od strane korisnika i vezanog za formaciju instance konstrukcijskog znanja - pravila,
- *r_naziv* – znakovna varijabla koja sadrži simboličko ime prikaza,
- *r_opis* – znakovna varijabla koja sadrži kratak opis prikaza,
- *r_kontekst_tip* – znakovna varijabla koja sadrži naziv konteksta prikaza, kontekst prikaza se učitava iz dodatne tablice **konteksta_prikaza**,
- *r_kontekst_opis* – znakovna varijabla koja sadrži kratak opis konteksta prikaza, opis konteksta prikaza se učitava iz dodatne tablice **konteksta_prikaza**,
- *description_item[]* – polje pokazivača na objekte tipa ***DescriptionItem***,
- *FormacijaKZ()* – metoda konstruktor klase ***FormacijaKZ***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta,
- *addDescriptionItem()* – metoda služi za kreiranje nove instance prikaza konstrukcijskog znanja i dodavanje istog u polje pokazivača na objekte tipa ***DescriptionItem***,
- *rmDescriptionItem()* – metoda služi za brisanje instance prikaza konstrukcijskog znanja,
- *getDescriptionItem()* – metoda služi za pretraživanje polja instanci i postavljanje radne instance prikaza konstrukcijskog znanja,

- *setDescriptionItem()* – metoda služi za postavljanje vrijednosti atributa instance prikaza konstrukcijskog znanja,
- *isDescriptionItem()* – metoda služi za saznavanje trenutne radne instance prikaza konstrukcijskog znanja.

Svaka instanca formacije konstrukcijskog znanja instancira jedan ili više objekata prikaza tipa ***DescriptionItem***. Klasa ***DescriptionItem*** sadrži definiciju sljedećih atributa i metoda (Slika 7-12):

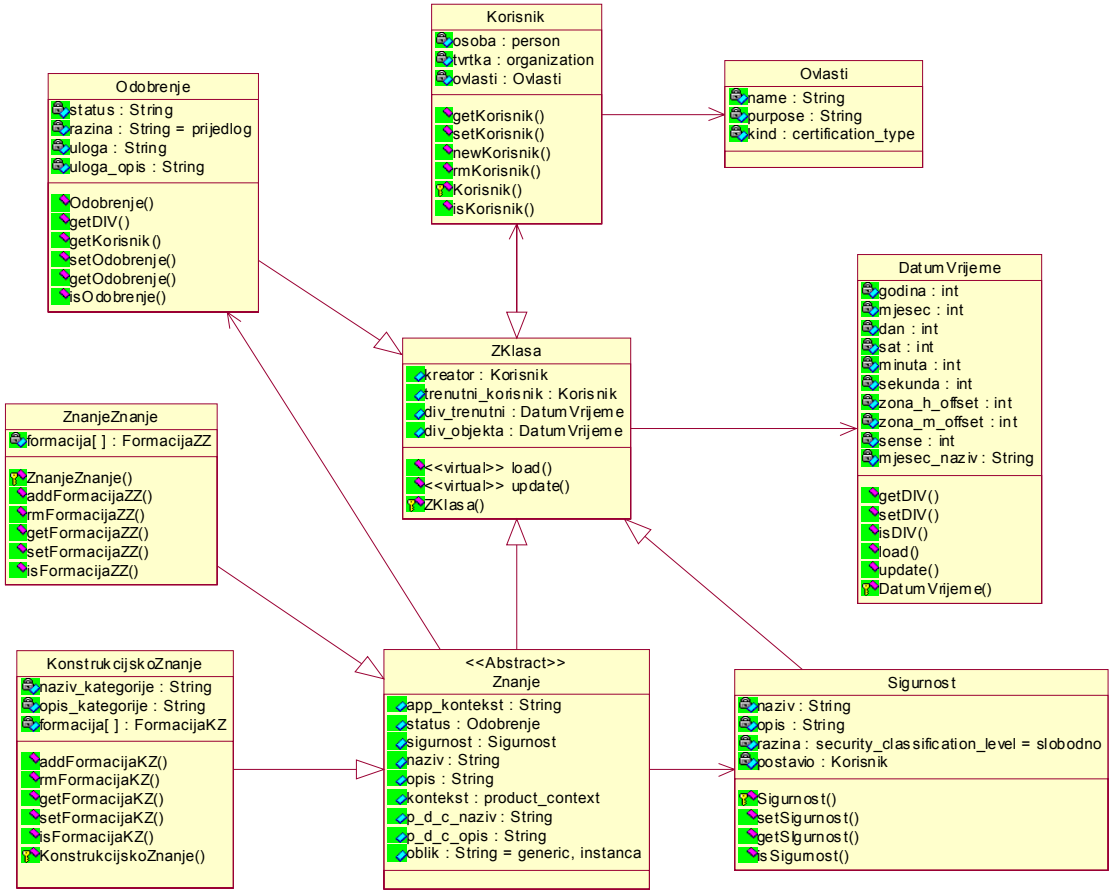
- *naziv* – znakovna varijabla koja sadrži simboličko ime prikaza,
- *sadrzaj* – znakovna varijabla koja sadrži gradbeni element znanja,
- *load()* – metoda koja služi za učitavanje podataka o gradbenim elementima prikaza iz tablica baze podataka,
- *update()* – metoda koja služi za promjenu i kreiranje podataka o gradbenim elementima prikaza u tablicama baze podataka,
- *DescriptionItem()* – metoda konstruktor klase ***DescriptionItem***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

Na slici 7-13 prikazana je struktura klasa ***Odobrenje***, ***Sigurnost***, ***Korisnik***, ***Ovlasti*** i ***DatumVrijeme*** te asocijacija navedenih klasa prema klasama znanja tj. klasi ***ZnanjeZnanje*** i klasi ***KonstrukcijskoZnanje***. Klasa ***Odobrenje*** veže se na definicije klasa ***ZnanjeZnanje*** i ***KonstrukcijskoZnanje*** mehanizmom nasljeđivanja iz apstraktne klase ***Znanje***. Što znači da i klasa ***ZnanjeZnanje*** i klasa ***KonstrukcijskoZnanje*** instancira barem jedan objekt tipa ***Odobrenje***, ***Sigurnost*** i ***Korisnik***. U klasi ***Odobrenje*** definirane su sljedeće metode i atributi:

- *status* – znakovna varijabla koja sadrži opis svrhe odobrenja,
- *razina* – znakovna varijabla koja sadrži razinu odobrenja, inicijalno svaka nova instanca konstrukcijskog znanja inicijalizirana je na razinu **prijedlog**,
- *uloga* – znakovna varijabla koja sadrži simboličko ime uloge odobrenja u kontekstu,
- *uloga_opis* – znakovna varijabla koja sadrži opis uloge odobrenja u kontekstu,
- *Odobrenje()* – metoda konstruktor klase ***Odobrenje***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

Objekti klase ***Sigurnost*** instanciraju se prilikom kreiranja svake instance ***ZnanjaZnanja*** i ***KonstrukcijskogZnanja.*** Klasa ***Sigurnost*** sadrži definiciju sljedećih metoda i atributa (Slika 7-13):

- *naziv* – znakovna varijabla koja sadrži simboličko ime definicije sigurnosti,
- *opis* – znakovna varijabla koja sadrži kratak opis definicije sigurnosti,
- *razina* – znakovna varijabla koja sadrži razinu sigurnosti vezanu za instancu znanja, inicijalno razina je postavljena na vrijednost **slobodno**,
- *Sigurnost()* – metoda konstruktor klase ***Sigurnost***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta,



Slika 7-13 Struktura klasa ***Odobrenje***, ***Sigurnost***, ***Korisnik***, ***Ovlasti*** i ***DatumVrijeme***

Prilikom instanciranja objekata znanja instancira se i objekt tipa ***Korisnik***. Budući da u sustavu postoji inicijalno samo jedan korisnik (**upravitelj_znanja**) u klasi ***Korisnik*** definirana je i metoda za kreiranje tj. dodavanje novih korisnika sustava. Za atribute koji su definirani u STEP standardu navest će se samo njihov naziv. Klasa ***Korisnik*** sadrži definicije sljedećih atributa i metoda:

- *osoba* – objekt tipa ***Person*** koji sadrži informacije o korisniku, u klasi ***Person*** definirani su sljedeći atributi:
  - *id*, *prezime*, *ime*, *srednje_ime*, *prefiks*, *sufiks*, *ulica*, *kucni_broj*, *grad*, *drzava*, *email*, *telefon*, *telefaks*, *opis*, *okrug*, *postanski_broj*, poštanski broj, *lokacija*, *pbox*, *fascimile*.
- *tvrtka* – objekt tipa ***Organization*** koji sadrži informacije o organizaciji, u klasi ***Organization*** definirani su sljedeći atributi:
  - *id*, *prezime*, *ime*, *srednje_ime*, *prefiks*, *sufiks*, *ulica*, *kucni_broj*, *grad*, *drzava*, *email*, *telefon*, *telefaks*, *opis*, *okrug*, *postanski_broj*, poštanski broj, *lokacija*, *pbox*, *fascimile*.
- *ovlasti* – objekt tipa ***Ovlasti*** koji sadrži informacije o ovlastima korisnika,
- *div_kreiranja* – objekt tipa ***DatumVrijeme*** koji sadrži datum i vrijeme kreiranja referenciranog objekta,
- *newKorisnik()* – metoda za kreiranje novog korisnika,

- *getKorisnik()* – metoda za pretraživanje korisnika,
- *setKorisnik()* – metoda za postavljanje i promjenu atributa korisnika,
- *rmKorisnik()* – metoda za brisanje korisnika,
- *isKorisnik()* – metoda za saznavanje informacija o trenutnom korisniku,
- *Korisnik()* – metoda konstruktor klase ***Korisnik***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta,
- *load()* – metoda koja služi za učitavanja podataka o korisniku iz tablica u bazi podataka,
- *update()* – metoda koja služi za zapisivanje ili promjenu podatka o korisniku u tablicama baze podataka.

Klasa ***Ovlasti*** se referencira samo iz klase ***Korisnik***. Prilikom instanciranja objekta tipa ***Korisnik*** automatski se instancira i objekt tipa ***Ovlasti***. U klasi ***Ovlasti*** definirani su sljedeći atributi i metode (Slika 7-13):

- *naziv* – znakovna varijabla koja sadržava simboličko ime ovlasti,
- *svrha* – znakovna varijabla koja opisuje svrhu ovlasti,
- *tip* – znakovna varijabla čiji sadržaj određuje ovlasti korisnika,
- *load()* – metoda koja služi za učitavanja podataka o korisniku iz tablica u bazi podataka,
- *update()* – metoda koja služi za zapisivanje ili promjenu podatka o korisniku u tablicama baze podataka,
- *Ovlasti()* – metoda konstruktor klase ***Ovlasti***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

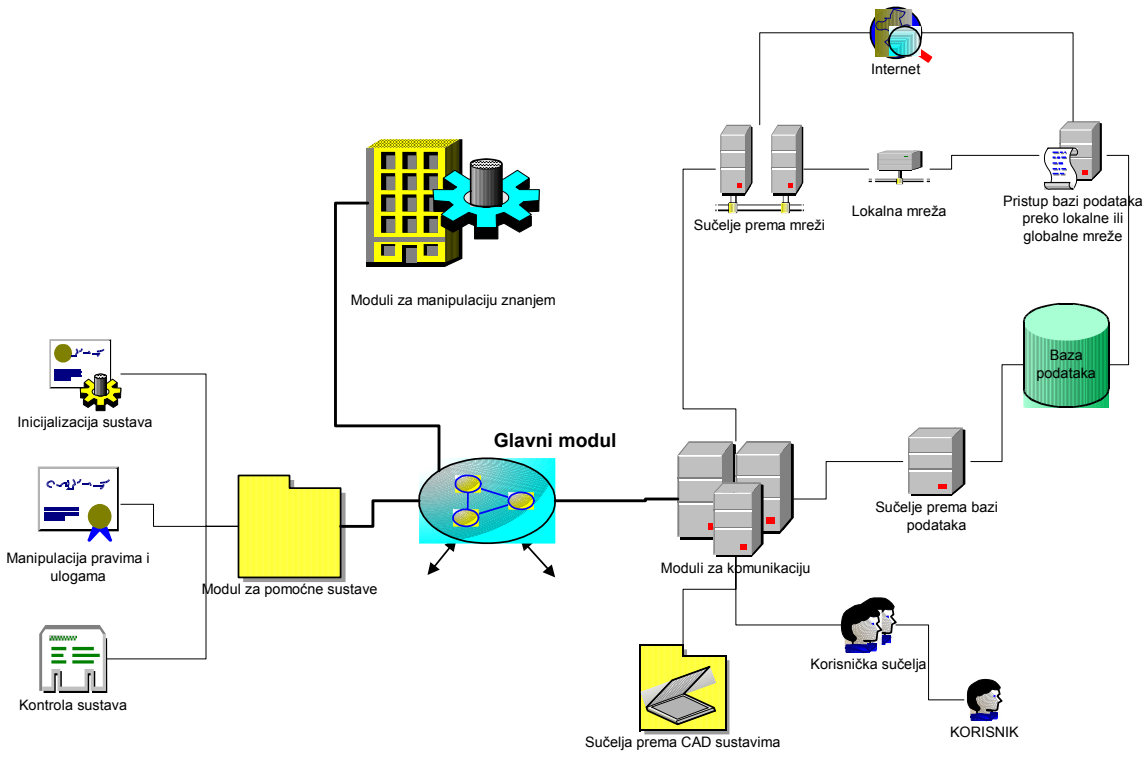
Prilikom instanciranja objekta tipa ***ZKlasa*** instancira se i objekt tipa ***DatumVrijeme***. U definiciji klase ***DatumVrijeme*** definirani su sljedeći atributi i metode:

- *godina*, *mjesec*, *dan*, *sat*, *minuta*, *sekunda*, *zona_h_offset*, *zona_m_offset*, *mjesec_naziv*,
- *getDIV()* – metoda za učitavanje vremena i datuma,
- *setDIV()* – metoda za postavljanje i promjenu vremena i datuma,
- *isDIV()* – metoda za saznavanje vremena i datuma,
- *DatumVrijeme()* – metoda konstruktor klase ***DatumVrijeme***, poziva se svaki put kada se instancira novi objekt, metoda služi za postavljanje početnih vrijednosti atributa objekta.

7.2 Opis sustava

U ovom poglavlju opisana je struktura i uloga pojedinih dijelova sustava za upravljanje i manipulaciju konstrukcijskim znanjem[67]. Osnovna zadaća sustava je omogućiti korisniku rad s proširenim CAD modelom. Osim ispunjenja osnovne zadaće sustav mora omogućiti upravljanje i manipulaciju konstrukcijskim znanjem, bazom podataka, korisničkim sučeljima te komunikacijom sa CAD aplikacijom. Struktura sustava prikazana je na slici 7-14. Sustav se sastoji od četiri grupacije modula:

- Glavni modul – nositelj je osnovne niti rada sustava, te sadrži pravila zaključivanja, pravila zaključivanja se aktiviraju ovisno o događajima u sustavu tako da se rad Glavnog modula može smatrati da je pogonjen događajima[68],
- Moduli za komunikaciju – kreiranje i provjera sučelja za komunikaciju sa korisnikom, bazom podataka i CAD aplikacijom,
- Moduli za manipulaciju znanjem – upravljanje i manipulacija konstrukcijskim znanjem tijekom rada sustava,
- Pomoćni sustavi – skup alata za rad sa datotekama, obradu greške, provjeru rada sustava te provjeru resursa sustava.



Slika 7-14 Struktura sustava za manipulaciju proširenim CAD modelom

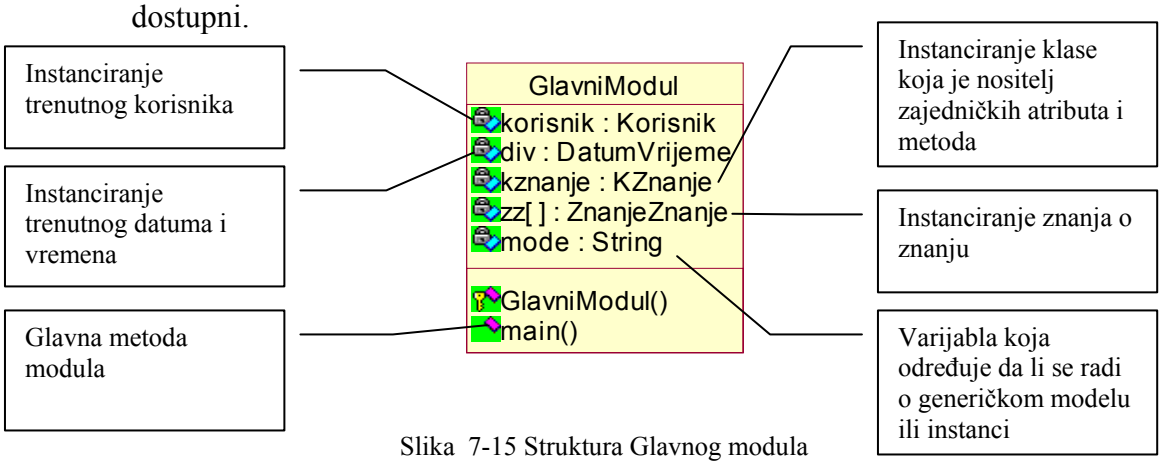
[67] Zbog kraćeg pisanja u daljnjem tekstu pod sustav podrazumijevati će se sustav za upravljanje i manipulaciju proširenim CAD modelom

[68] Pod pojmom pogonjen događajima smatra se "event driven" paradigma načina programiranja.

7.2.1 Glavni modul

Glavni modul je središnji dio sustava. Definiran je preko klase ***GlavniModul*** u kojoj su definirane metode i atributi neophodni za rad sustava. Budući da je sustav spregnut sa CAD aplikacijom, tj. sustav se može smatrati i jednim od sučelja CAD aplikacije, glavni modul se aktivira iz CAD aplikacije. U programski kod glavnog modula ugrađeno je i "rule based" [127], [139], [140] pravilo zaključivanja. Pravilo zaključivanja se sastoji od nekoliko grupacija ugniježđenih "if" programskih struktura. Svaka od grupacija aktivira se ovisno o događajima generiranim tijekom rada sustava. Programska struktura pravila zaključivanja ugrađena je u programski kod glavnog modula i ne može se mijenjati od strane korisnika niti od strane sustava. Struktura glavnog modula prikazana je na slici 7-15. Zadaće glavnog modula su:

- upravljanje sustavom i svim dijelovima sustava,
- provjera dostupnih resursa,
- kreiranje osnovnih objekata i postavljanje početnih vrijednosti,
- inicijalizacija sustavskog vremena,
- određivanje radne CAD aplikacije,
- određivanje, preko pravila zaključivanja, objekata koji će se instancirati te metoda koje će se pozivati,
- provjera postojanja mreže i resursa na mreži (navedeno je od iznimne važnosti ukoliko su dijelovi sustava distribuirani po računalima u mreži),
- odabir načina rada – ukoliko je za rad na sustavu prijavljen upravitelj znanja tada se aktiviraju dijelovi sustava koji služe za upravljanje i manipulaciju sustavom, u suprotnom se radi o "normalnoj" eksploataciji programa kada navedeni dijelovi nisu dostupni.



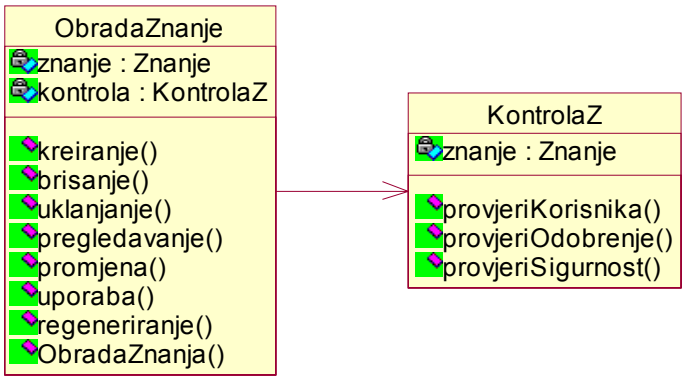
Slika 7-15 Struktura Glavnog modula

7.2.2 Moduli za obradu znanja

Moduli za manipulaciju znanjem dio su sustava namijenjenog kreiranju, uklanjanju, pregledavanju, brisanju, regeneriranju, promijeni i uporabi konstrukcijskog znanja proširenog CAD modela. Veza između modula ostvarena je preko glavnog modula. Glavni modul ovisno o potrebama sustava ili korisnikovim akcijama aktivira određenu metodu za manipulaciju znanjem. Na slici 7-16 prikazana je struktura klase ***ObradaZnanja***. Osim strukture klase ***ObradaZnanje*** prikazana je i struktura klase ***KontrolaZ*** koja služi za provjeru svojstava znanja tj. provjeru korisnika, odobrenja i sigurnosti instanciranog znanja. Budući da je količina zapisa o znanjima proširenog CAD modela znatna, prilikom uporabe pojedinih instanci proširenog CAD modela u

memoriju računala se ne učitava cijela struktura znanja već samo onaj dio koji je potreban za zadovoljenje funkcije pojedine metode ili akcije korisnika. Svaka metoda za obradu znanja može proslijediti zahtjev, preko glavnog modula, za učitavanjem dijela znanja iz baze podataka. Zadaće pojedinih metoda klase ***ObradaZnanja*** su sljedeće:

- kreiranje modela[69] – metoda služi za kreiranje novog generičkog zapisa proširenog CAD modela; prilikom aktiviranja metoda obavlja sljedeće zadaće:
  - kontrolira ovlasti korisnika za kreiranje modela,
  - kontrolirati postojanje istovjetnog model u bazi,
  - čita informacija sa geometrijskog modela, određuje da li se radi o dijelu ili sklopu, ukoliko se radi o dijelu učitava informacije o materijalu, jedinicama, nazivu dijela i radnom direktoriju, a ukoliko se radi o sklopu učitava informacije o jedinicama, nazivu sklopa i radnom direktoriju,
  - čita strukturu značajki u dijelu ili strukturu dijelova i značajki ako se radi o sklopu,
  - određuje aktivne komponente dijela ili sklopa te evidentira odnos dijete-otac i otac-dijete komponenti; potrebno je kontrolirati veze između komponenti modela te zapisati koje se komponente mogu isključiti, a koji ne tj. zapisati reference,
  - čita parametre komponenti,
  - čita dimenzije svake značajke,
  - čita interne relacija u komponentama,



Slika 7-16 Struktura klase za manipulaciju znanjem i kontrolne klase

- brisanje modela – prilikom aktiviranja metode brisanje potpuno se uništava generički zapis te zapisi o svakoj instanci proširenog CAD modela iz baze podataka te se briše datoteka koja sadrži geometrijski prikaz modela, aktiviranjem ove metode obavljaju se sljedeće zadaće:
  - brisanje entiteta znanja iz baze podataka te kreiranih instanci entiteta znanja,
  - brisanje zapisa i instanci konstrukcijskog znanja koje je nositelj instanci entiteta znanja,
  - brisanje znanja o znanju znanja, prilikom brisanja znanja o znanju slijedom veza između znanja briše se i konstrukcijsko znanje vezano za instancu znanja o znanju koje se briše i instance entiteta znanja koje su vezane za instance obrisanog konstrukcijskog znanja,
  - brisanje datoteka geometrijskog prikaza modela,

[69] Pod pojmom model smatra se prošireni CAD model tj. konstrukcijsko znanje i geometrijski CAD model

  - postavljanje zastavice za brisanje objektnog modela iz sklopova koji nisu aktivni, kada se otvori sklop, sa ubačenom instancom proširenog CAD modela, korisniku se pruža mogućnost uklanjanja tj. brisanja modela iz sklopa ili samo uklanjanje značajke proširenog CAD modela.
- uklanjanje modela – za razliku od brisanja, uklanjanje proširenog CAD modela samo ukloni tj. obriše instancu modela, dok se generički zapis i zapisi ostalih instanci u bazi podataka te datoteke geometrijskog prikaza ne diraju, aktiviranjem ove metode obavljaju se sljedeće zadaće:
  - kontrolira mogućnost uklanjanja objektnog modela iz aktivnog sklopa, naime, ne smije se narušiti integritet sklopa,
  - brisanje značajke proširenog CAD modela,
  - brisanje geometrijskog prikaza proširenog CAD modela,
  - brisanje instanci znanja obrisanog proširenog CAD modela,
- pregledavanje modela – metoda služi za pregledavanje dostupnih generičkih i instanciranih zapisa modela te strukture i atributa odabranog znanja, prilikom aktiviranja metoda obavlja sljedeće zadaće:
  - kontrolira pristup korisnika odabranom dijelu znanja,
  - određuje da li se učitavaju podaci o generičkom modelu ili instanci,
  - čita strukturu odabranog znanja o znanju[70],
  - čita strukturu odabranog konstrukcijskog znanja,
  - čita strukturu entiteta znanja odabranog konstrukcijskog znanja,
  - poziva prikaz grafičkog sučelja za pregled znanja objektnog modela,
- promjena modela – metoda za promjenu strukture i vrijednosti atributa proširenog CAD modela podijeljena je u dva dijela, prvi dio koristi se za promjenu strukture i vrijednosti atributa generičkog zapisa dok je drugi dio namijenjen promjeni strukture i vrijednosti atributa odabrane instance proširenog CAD modela,
  - promjena generičkog zapisa uključuje obavljanje sljedećih zadaća:
    - kontrolirati pristup generičkom zapisu,
    - učitavanje strukture znanja o znanju,
    - učitavanje strukture konstrukcijskog znanja,
    - učitavanje strukture entiteta znanja,
    - prikazivanja grafičkog sučelja za pregled znanja proširenog CAD modela,
    - prilikom korisnikovih promjena metoda aktivno kontrolira svaku unesenu promjenu, promjene se mogu unesti samo u granicama dozvoljenim postavkama i ograničenjima,
    - ukoliko dođe do promjene znanja postavlja se zastavica kao informacija instancama da je došlo do promjene znanja te da na osnovu toka moraju poduzeti akcije (ignorirati promjene, aktivirati promjene),
  - promjena zapisa o instancama uključuje obavljanje sljedećih zadaća:
    - kontrolirati pristup zapisu o instanci,
    - učitavanje strukture znanja o znanju
    - učitavanje strukture konstrukcijskog znanja,
    - učitavanje strukture entiteta znanja,
    - prikazivanja grafičkog sučelja za pregled informacije vezanim za aktivnu instancu proširenog CAD modela,

[70] Postupak učitavanja strukture znanja uključuje i učitavanje vrijednosti atributa znanja

- uporaba modela – metoda za uporabu modela aktivira se u slučaju kada korisnik želi ubaciti prošireni CAD model u sklop, prilikom aktiviranja ove metode obavljaju se sljedeće zadaće:
  - kontrolira ovlasti korisnika za uporabu modela,
  - učitava strukturu geometrijski prikaz modela ,
  - učitava strukturu znanja o znanju odabranog modela,
  - učitava strukturu konstrukcijskog znanja vezanog za odabranu instancu znanja o znanju,
  - učitava strukturu entiteta znanja odabranog konstrukcijskog znanja,
  - aktivira sučelje za odabir verzije i ubacivanje proširenog CAD modela u sklop,
  - ovisno o odabranoj verziji aktivira raščlambu zapisa entiteta znanja i pokreće metode ovisno o rezultatu raščlambe,
  - ubacuje geometrijski prikaz u sklop,
  - sputava slobode gibanja ubačenog geometrijskog prikaza u sklopu ovisno o informacijama iz znanja ograničenjima odabrane instance,
  - postavlja vrijednosti parametara i dimenzija geometrijskog modela,
  - izvršava interne relacije geometrijskog modela,
  - poziva metodu za regeneriranje dijela i regeneriranje sklopa,
  - zapisuje novo kreiranu instancu proširenog CAD modela,
- regeneriranje modela – metoda za regeneriranje modela regenerira model u dva prolaza, prvi prolaz se ostvaruje na razini zadovoljenja ograničenja proširenog CAD modela dok se drugi prolaz ostvaruje metodama regeneracije CAD aplikacije uz provjeru redoslijeda regeneracije i dubine regeneracije od strane sustava.

7.2.3 Moduli za komunikaciju

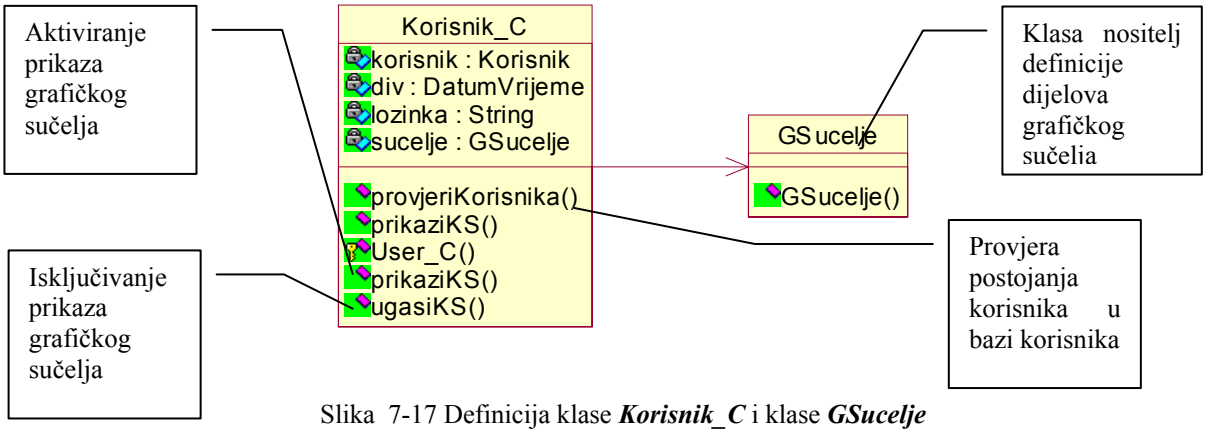
Moduli za komunikaciju ostvaruju komunikaciju između sustava i korisnika, sustava i baze podataka te sustava i CAD aplikacije. Komunikacija unutar sustava ostvaruje se preko glavnog modula. Implementacija modula za komunikaciju ovisi o raspoloživim programskim alatima i aplikacijama.

7.2.3.1 Korisničko sučelje

Korisničko sučelje ostvaruje komunikaciju između korisnika i sustava. Sučelje se sastoji od klase ***Korisnik_C*** i klase ***GSucelje*** (Slika 7-17). Prilikom instanciranja klase ***Korisnik_C*** kreira se instanca korisnika i sučelja te instance trenutnog vremena i datuma. Klasa ***GSucelje*** sadrži definicije dijelova grafičkog sučelja. Grafičko sučelje je kreirano u skladu sa JAVA AWT preporukama i sa dijelovima iz JAVA Swing biblioteka [141]. U sustavu postoji nekoliko definicija[71] klase ***GSucelje***. Svaka definicija ovisi o informacijama koje se žele prikazati i bibliotekama dostupnim za ciljanu implementacijsku platformu[72]. Dostupne informacije preko korisničkog sučelja ovise o korisniku koji je prijavljen za rad i njegovim ovlastima.

[71] Navedeno proizlazi iz mogućnosti implementacijskog sustava. Uporabom JAVA programskog jezika moguće je kreiranje nezavisnih kompilacija koda te njihova izmjena zamjenom .class datoteka.

[72] Pojedine CAD aplikacije posjeduju vlastite biblioteke za kreiranje grafičkog sučelja.

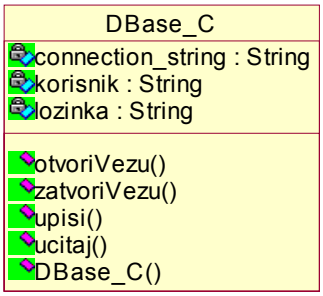


Slika 7-17 Definicija klase ***Korisnik_C*** i klase ***GSucelje***

7.2.3.2 Sučelje prema bazi podataka

Baza podataka može se nalaziti bilo gdje na lokalnom računalu ili na nekom umreženom računalu na koje radno računalo tj. korisnik ima pristup. Prilikom aktiviranja sustava jedan od pomoćnih modula učitava između ostalog i informaciju o smještaju baze podataka te korisničkom broju i lozinki potrebnoj za uspostavljanje veze prema bazi. Sustav s bazom podataka komunicira uporabom SQL upita. Zadaća modula za rad sa bazom podatka (Slika 7-18) je:

- provjera postojanja baze podataka sustava,
- prijava rada na bazu podataka uporabom korisničkog broja i lozinke kreirane u tu svrhu,
- kreiranje znakovne konstante[73] za vezu prema bazi i inicijalizacija znakovne varijable za vezu prema bazi,



Slika 7-18 Definicija klase za rad sa bazom podataka

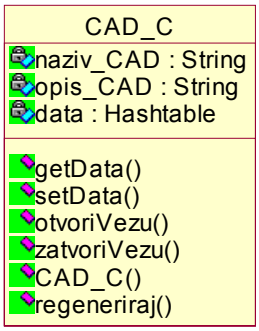
- aktiviranje metode za učitavanje i metode za zapisivanje podataka odabranog znanja, i jedna i druga metoda kao argument primaju podatak tipa ***Znanja***, što znači da, budući da su klasa ***ZnanjeZnanje*** i klasa ***KonstrukcijskoZnanje*** derivirane iz klase ***Znanje***, dovoljna je jedna metoda za ostvarenje učitavanja podataka i jedna metoda za zapisivanje podataka i za objekte tipa ***ZnanjeZnanje*** i objekte tipa ***KonstrukcijskoZnanje***; SQL upiti koji se prosljeđuju bazi podataka kreiraju se ovisno o tipu znanja koje se zapisuje u metodama *load()* i *update()* odabranog tipa znanja,
- zatvaranje veze prema bazi podataka.

[73] Connection string – skup znakova koji se koristi za otvaranje komunikacije prema bazi podataka

7.2.3.3 Sučelje prema CAD aplikaciji

Sustav je koncipiran na taj način da se može implementirati u različite CAD aplikacije. Ovisno o radnoj CAD aplikaciji aktivira se adekvatno sučelje. Navedeno je moguće ostvariti, uporabom JAVA programa, kreiranjem datoteka s pozivima CAD aplikaciji ili, uporabom drugog programskog jezika, kreiranjem preopterećenih metoda. Definicija klase koja je nositelj sučelja prema CAD aplikaciji prikazana je na slici 7-19. U definiciji klase definirani su sljedeći atributi i metode:

- *naziv_CAD* – znakovna varijabla koja sadrži naziv radne CAD aplikacije,
- *opis_CAD* – znakovna varijabla koja sadrži kratak opis radne CAD aplikacije,
- *data* – indeksirano polje objekata koje sadrži podatke (vrijednosti i nazive dimenzija i parametara sa CAD modela) vezane za CAD model,
- *getData()* – metoda za očitavanje vrijednosti i naziva dimenzija i parametara CAD modela,
- *setData()* – metoda za postavljanje vrijednosti dimenzija i parametara CAD modela,
- *otvoriVezu()* – otvaranje veze između sustava i CAD aplikacije,
- *zatvoriVezu()* – zatvaranje veze između sustava i CAD aplikacije,
- *regeneriraj()* – metoda koja služi za regeneriranje proširenog CAD modela i poziv metodi za regeneriranje CAD aplikacije.



Slika 7-19 Prikaz definicije klase ***CAD_C***

7.2.4 Moduli za pomoćne sustave

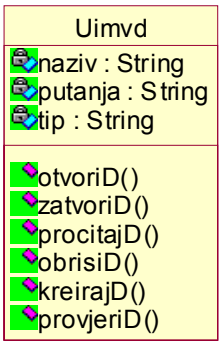
Dio sustava koji čine pomoćni moduli zadužen je za kreiranje i komunikaciju s vanjskim datotekama, provjeru postojanja svih radnih (.class i .ini) datoteka sustava, učitavanje inicijalizacijskih datoteka, obradu korisnika te obradu greške. Pomoćni sustavski moduli pozivaju se iz glavnog modula ovisno o događajima u sustavu. Pomoćne module čine moduli za:

- obradu greške – ovaj modul služi za obradu korisničkih grešaka i grešaka nastalih tijekom rada sustava; prilikom nastanka greške generira se izuzetak ("exception") te se poziva metoda za obradu greške instanciranog objekta tipa ***Greska*** (Slika 7-20); ovisno o kodu greške i mjestu nastanka greške iz baze grešaka učitava se *opis_greške* u kojem je opisana greška i eventualni uzrok te mogući postupci za rješenje nastale greške; nakon učitavanja podataka o grešci sve prikupljene informacije se prikazuju korisniku preko grafičkog korisničkog sučelja; korisnik ima nekoliko mogućnosti za nastavak rada u slučaju nastanka greške; mogućnosti ovise o vrsti nastale greške; pojedine greške (tipa "Unesena vrijednost je izvan granica!") zahtijevaju od korisnika

samo potvrđivanje informacije o nastaloj grešci, dok greške tipa "Sustavska greška: CAD model se ne može regenerirati!" zahtijevaju dodatne korisnikove radnje za otklanjanje greške,

Slika 7-20 Prikaz definicije klase ***Greska***

- upravljanje i manipulacija vanjskim datotekama (Slika 7-21) – zadaća modula za upravljanje vanjskim datotekama je otvaranje, zatvaranje odabrane datoteke te učitavanje i zapisivanje podataka iz datoteke ovisno o formatu zapisa podataka u datoteci; informacije o datotekama nalaze se zapisane u inicijalizacijskoj datoteci,

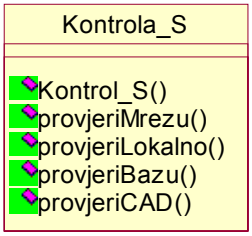


Slika 7-21 Prikaz definicije klase ***Uimvd***

- provjeru ovlasti i pristupa – svaki korisnik koji se prijavi za rad sa proširenim CAD modelom mora imati kreiran korisnički broj i potrebne ovlasti za uporabu modela; nadalje potrebno je tijekom rada sustava ispitivati stupnjeve sigurnosti odabranih znanja i ovlasti korisnika za rad s istima,
- upravljanje korisnicima (Slika 7-22) – ovaj modul se koristi u svrhu kreiranja, brisanja i promjene podataka o korisnicima sustava, u okviru modula omogućeno je i mijenjanje ovlasti korisnicima,

Slika 7-22 Prikaz definicije klase ***Uimkb*** za upravljanje i manipulaciju korisničkim brojevima

- provjeru sustava (Slika 7-23) – zadaća ovog modula je provjera postojanja i dostupnosti svih datoteka neophodnih za ispravan rad sustava; modul služi i za provjeru naziva i vrijednosti dimenzija i parametara aktiviranog proširenog CAD modela i postojanja svih neophodnih komponenti u sklopu ili značajki u dijelu.



Slika 7-23 Prikaz definicije klase ***Kontrola_S***

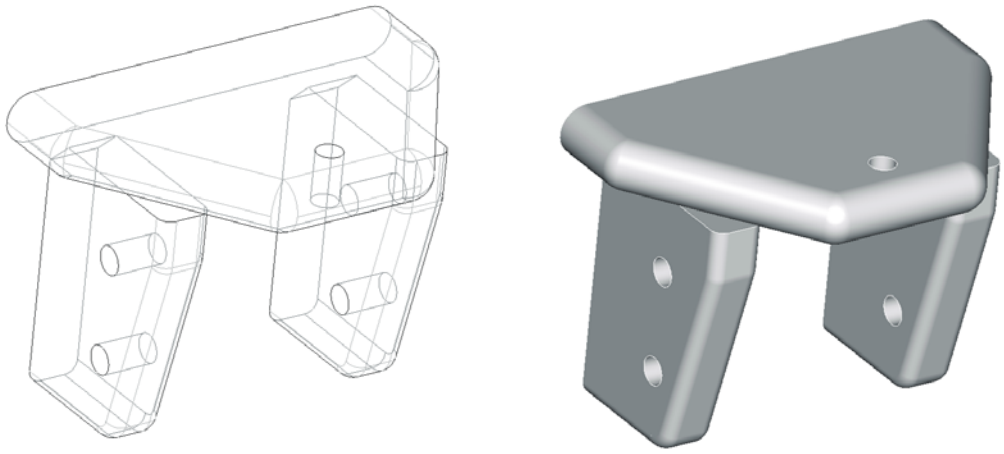
Pristup korisnika bazi podataka proširenog CAD modela moguć je i preko internet-a ili intraneta uporabom alata prilagođenih mrežnom radu. Korisnik, uporabom alata za pristup bazi podataka proširenog CAD modela, može kreirati, brisati i dodavati korisnike te mijenjati njihove ovlasti ovisno o njegovim ovlastima. Sa strane upravljanja i manipulacije znanjima proširenog CAD modela, korisnik može samo pregledavati generičke zapise i zapise instanci. Korisniku je omogućeno samo pregledavanje podataka u bazi podataka iz razloga što nije moguć pristup bazi CAD aplikacije ukoliko CAD aplikacija nije aktivna. Na taj način se osigurava ispravnost podatka i rješavaju problemi sinkronizacije podatka u bazi podataka i onih koji postoje u bazi CAD aplikacije. Aplikacija za pristup bazi podataka preko mreže posjeduje module za obavljanje sljedećih zadaća: provjeru korisnika, pregledavanja zapisa znanja, upravljanje i manipulaciju korisničkim brojevima. Navedeno se može ostvariti uporabom internet preglednika i JAVA "appleta".

Korisnik može kroz CAD aplikaciju mijenjati sve dimenzije i parametre geometrijskog modela s ograničenjem promjene dimenzija i parametara koji su vezani za znanje u proširenom CAD modelu. Ukoliko bi korisnik, putem naredbi CAD aplikacije, pokušao mijenjati vrijednosti i nazive dimenzija geometrijskog modela tada bi se javila poruka o grešci nakon aktiviranja glavnog modula tj. aktiviranja metode za provjeru sustava.

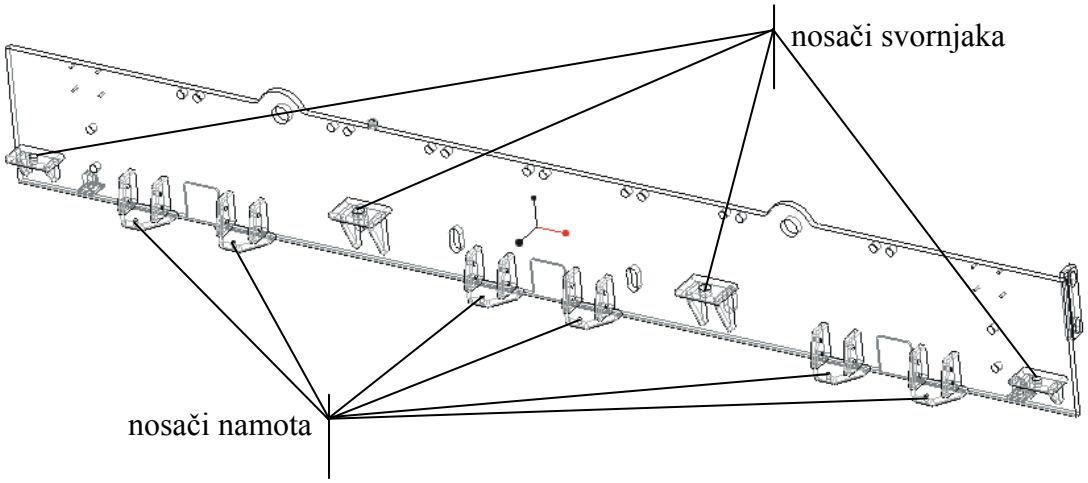
8 Prikaz primjene

U ovoj glavi primijenit će se razvijeni sustav na primjeru realne konstrukcije dijelova jezgre energetskog transformatora. Energetski transformator je uređaj za pretvorbu električne energije i tipičan je primjer varijantne konstrukcije. Sastoji se od četiri glavne cjeline: jezgre, kotla, poklopaca i konzervatora te dodatne opreme koja se instalira na kotao. Prošireni CAD model će se kreirati na geometrijskom modelu nosača namota jezgre transformatora (slika 8-1). Uloga nosača jezgre je višestruka. Ukoliko se nosači nalaze na donjoj steznoj ploči tada prenose aksijalne sile nastale uslijed težine namota i sile koje nastaju tijekom rada transformatora uslijed toplinske dilatacije. Ukoliko se nalaze na gornjoj steznoj ploči tada prenose aksijalne sile koje nastaju tijekom eksploatacije te osiguravaju položaj drvenih distantnih prstena tijekom ugradnje i rada.

Za svaki namot jezgre energetskog transformatora postoji ukupno osam nosača. Smještaj nosača namota na steznoj ploči prikazan je na slici 8-2. Jezgra transformatora posjeduje četiri stezne ploče, dvije na visokonaponskoj strani i dvije na niskonaponskoj strani. Stezne ploče služe za stezanje i osiguranje limova jezgre te kao osnova za pričvršćivanje nosača namota, nosača svornjaka za vertikalno stezanje i drugih nosača.

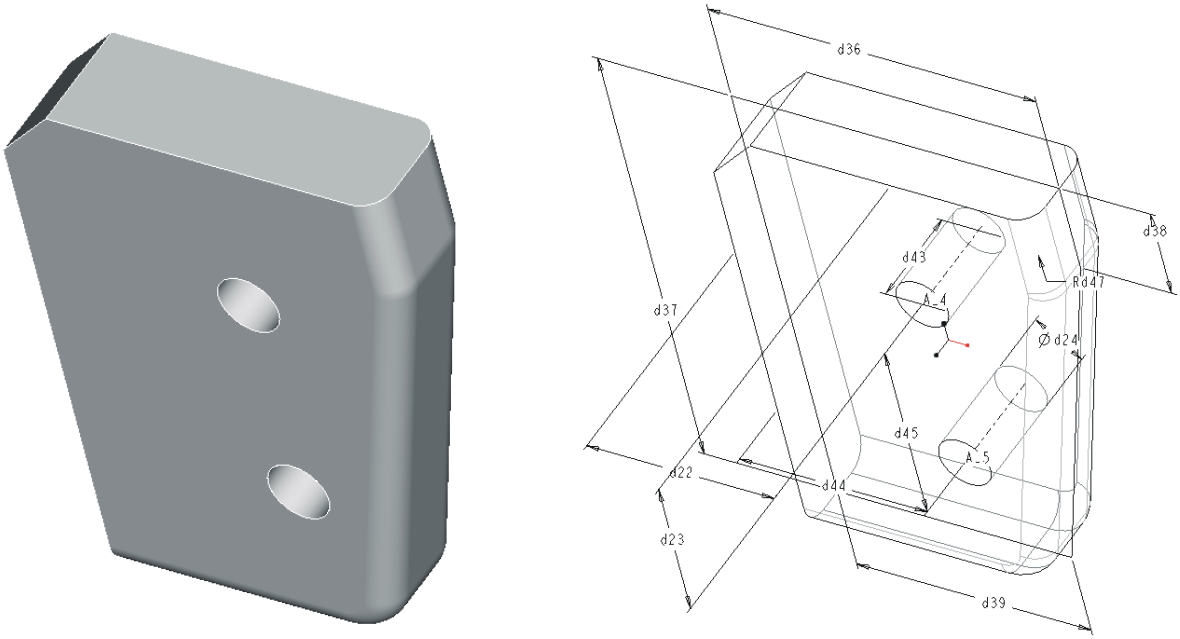


Slika 8-1 Model nosača namota energetskog transformatora

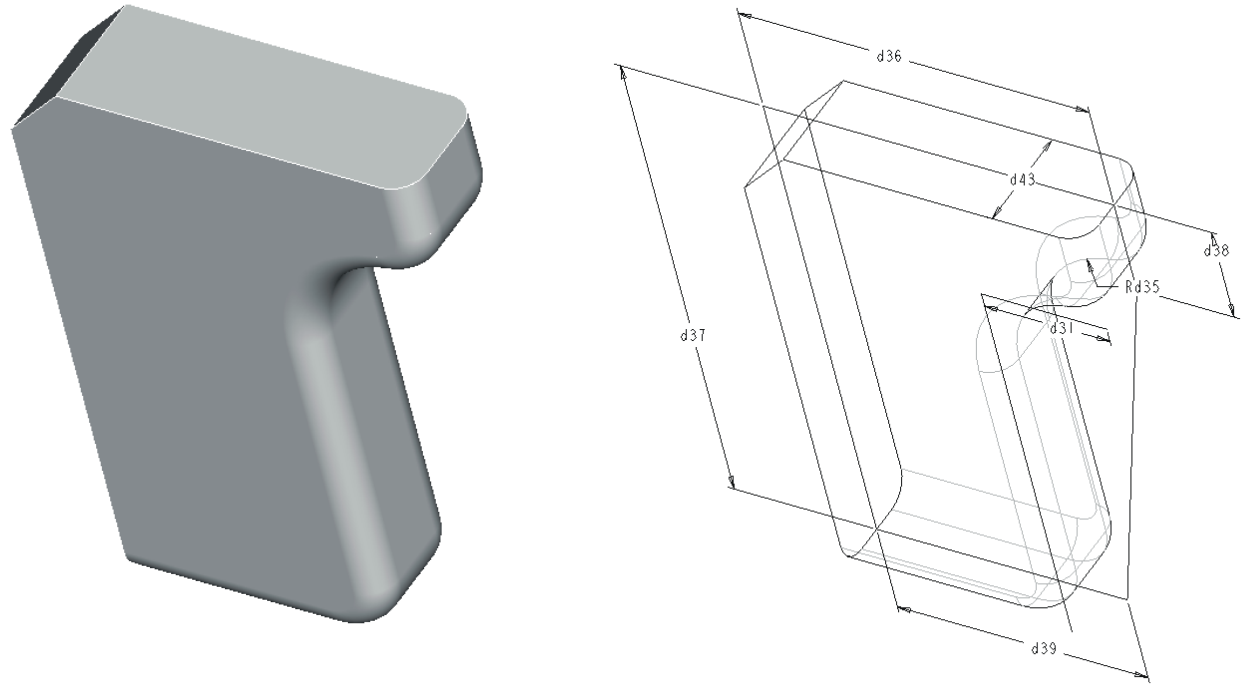


Slika 8-2 Model stezne ploče

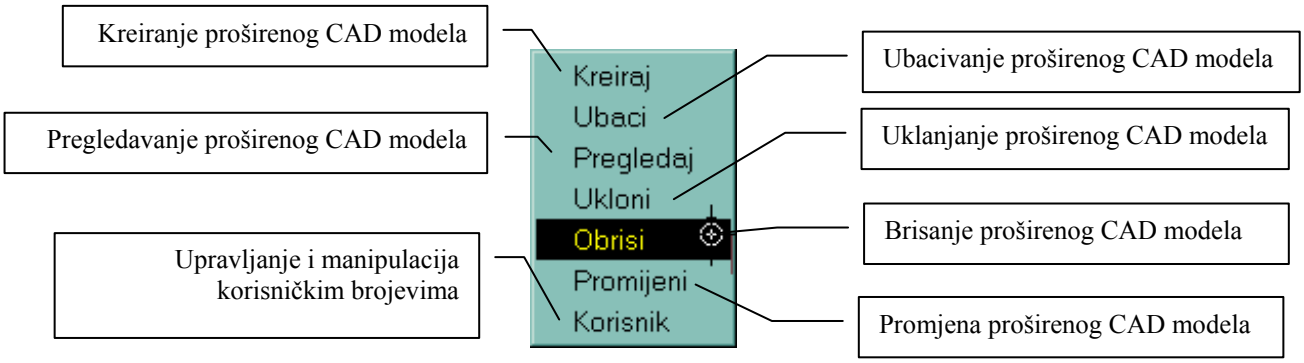
Model nosača namota sastoji se od tri dijela: osnovice (dio s oznakom B10_01_002) i dvije instance rebra (dio s oznakom B10_01_010-0X). Rebro nosaca kreirano je u dvije izvedbe: puno rebro (slika 8-3) i kutno rebro (slika 8-4). Prilikom definiranja proširenog CAD modela za rebro nosača namota kreirat će se dvije verzije: PunoRebro i KutnoRebro. U verziji PunoRebro uključit će se značajke koje određuju izvedbu rebra nosača prikazanoj na slici 8-4, te će se postaviti vrijednosti parametra d36 (vrijednost će se kopirati iz aktivirane aplikacije "Excel") i parametra d43 (vrijednost će se eksplicitno postaviti na 25). Ovisno o vrijednosti parametra d24 CAD modela postavit će se i vrijednost parametra d37 (vrijednost se odabire iz skupa ponuđenih vrijednosti).



Slika 8-3 Model punog rebra



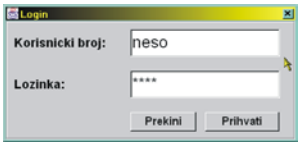
Slika 8-4 Model kutnog rebra



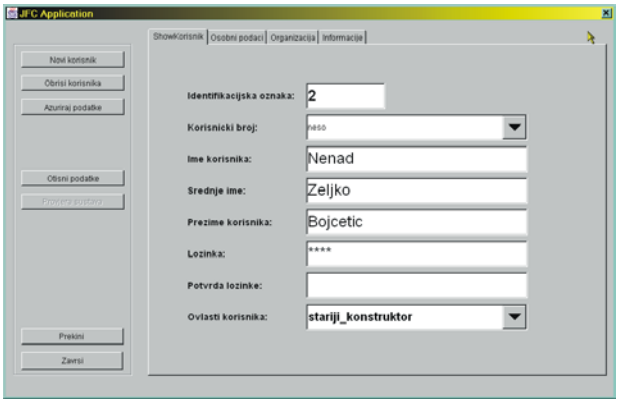
Slika 8-5 Izbornik za aktiviranje modula programskog sustava

Uporabom izbornika (slika 8-5) korisnik eksplicitno odabire pojedine naredbe i na taj način aktivirati module programskog sustava. Prilikom aktiviranja programski sustav zahtijeva od korisnika – konstruktora, unos korisničkog broja i zaporke (slika 8-6a). Ukoliko korisnik nije "poznat" sustavu potrebno je kreirati korisnika tj. dodijeliti mu korisnički broj i zaporku. Korisnički broj i zaporka korisniku dodjeljuje *Upravitelj* znanja odabirom opcije K**orisnik** sa glavnog izbornika.

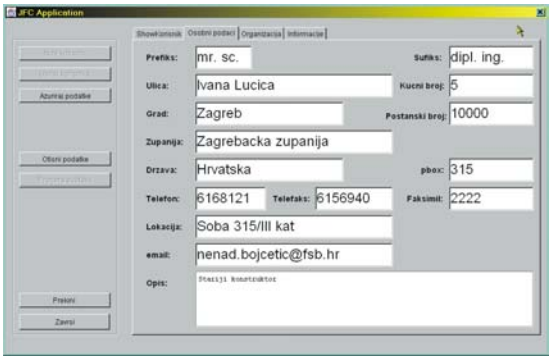
Nakon odabira navedene opcije aktivira se dio programskog sustava za kreiranje korisnika (slika 8-6b). U okviru definiranja korisnika zadaju se informacije o korisniku (slika 8-7a) te organizaciji u kojoj korisnik radi (slika 8-7b). Tijekom kasnije uporabe sustava, odabirom opcije za upravljanje korisnicima, moguće je dobiti uvid u informacije o modelima na kojima je pojedini korisnik radio te znanjima koje je kreirao tijekom rada. Na ekranima modula za upravljanje korisnicima moguće je promijeniti podatke o korisniku ili obrisati korisnika.



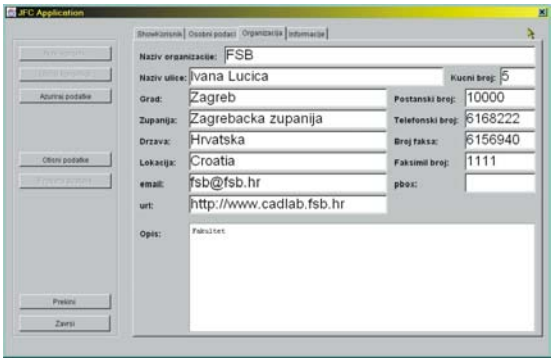
Slika 8-6 a) Prijava korisnika



Slika 8-6 b) Osnovni ekran za upravljanje korisnicima



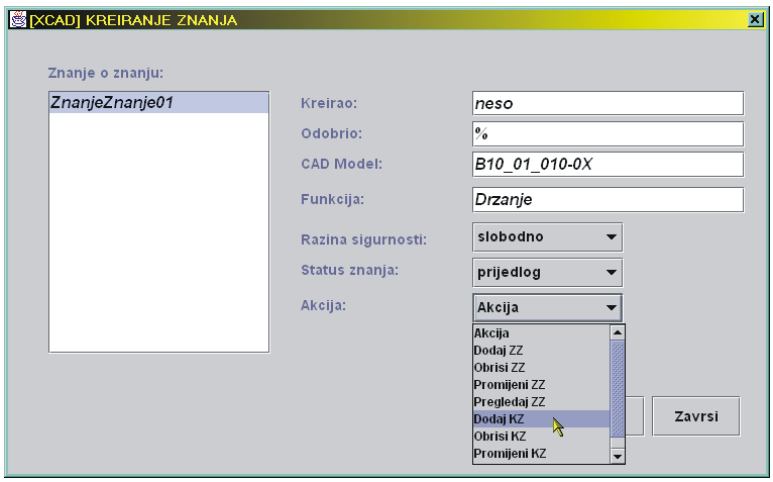
a) Unos osobnih podataka



b) Unos podataka o organizaciji

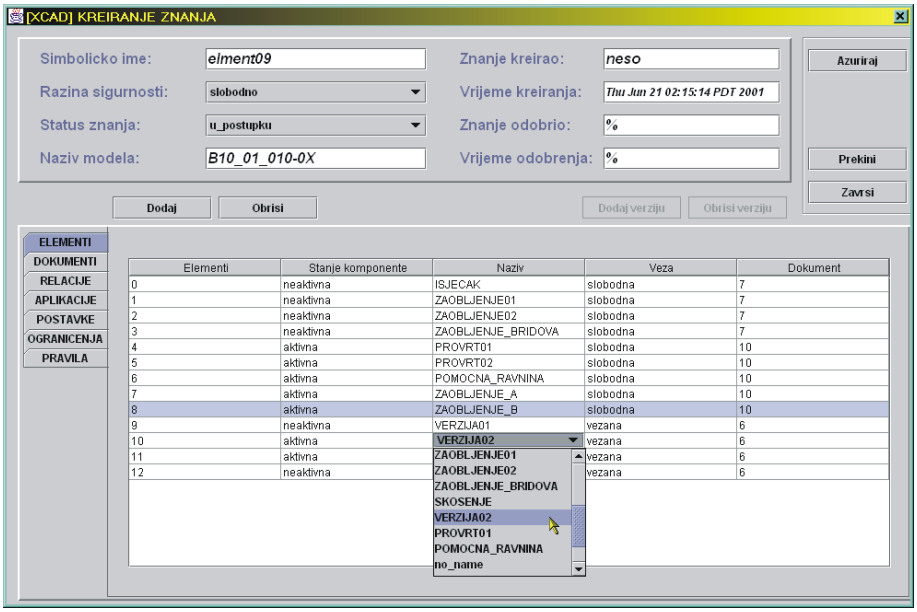
Slika 8-7 Dodatni ekrani za upravljanje korisnicima

Ukoliko korisnik posjeduje valjan korisnički broj i zaporku, može pristupiti kreiranju znanja za aktivni CAD model. Znanje se kreira odabirom opcije **Kreiraj** sa izbornika I**zbornik** na glavnom izborniku CAD aplikacije. Nakon aktiviranja navedene opcije otvara se prozor za definiranje znanja o znanju. Za aktivni CAD model moguće je definirati više znanja o znanju te kasnije za svako od definiranih kreirati konstrukcijska znanja. Prilikom zadavanja znanja o znanju potrebno je zadati simboličko ime i funkciju znanja. Funkcija se može zadati iz skupa predefiniranih funkcija (ovisno o internom dogovoru svake tvrtke ili ureda) ili zadavanjem funkcije po nekom standardu. U prikazanom primjeru funkcije, vezane za pojedina znanja, odabrane su prema [142].



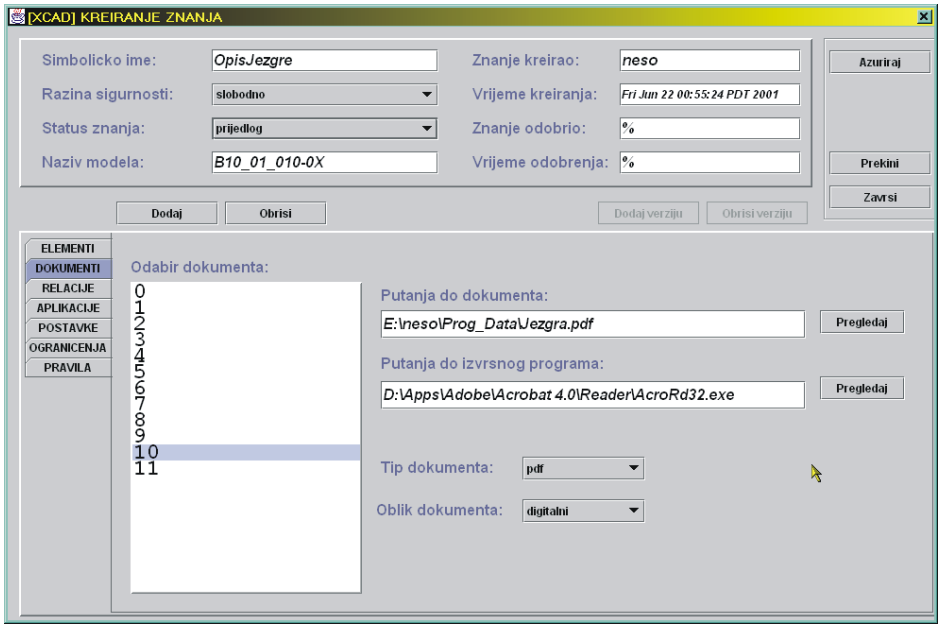
Slika 8-8 Prozor za definiranje znanja o znanju

Za definirano znanje o znanju potrebno je odrediti razinu sigurnosti i status znanja. Ime korisnika koji je kreirao znanje i naziv CAD modela za koji je znanje definirano automatski se učitava iz podataka sustava. Opcije za manipulaciju i upravljanje definiranim znanjem o znanju nalaze se na padajućem izborniku **Akcija**. Odabirom opcije **Dodaj KZ** aktivira se dio sustava za definiranje konstrukcijskog znanja (slika 8-9).



Slika 8-9 Prozor za definiranje konstrukcijskog znanja – ELEMENTI

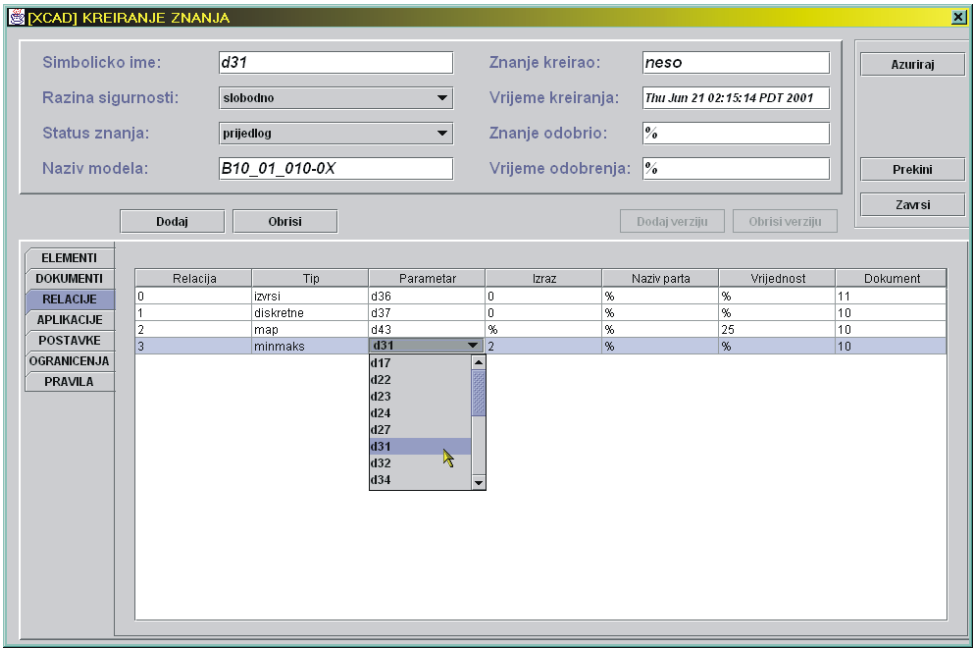
Na slici 8-9 prikazan je prozor za definiranje konstrukcijskog znanja – elementi. Prozor je podijeljen u dvije glavne cjeline. Na gornjem dijelu nalaze se informacije o simboličkom imenu znanja koje se definira, korisniku koji kreira znanje i korisniku koje je odobrio znanje, te nazivu CAD modela za koji se definira znanje, vrijeme i datum kreiranja i odobrenja znanja, te informacije o razini sigurnosti i statusu pojedinog znanja.



Slika 8-10 Prozor za definiranje konstrukcijskog znanja – DOKUMENTI

Drugi dio je kreiran kao okvir s karticama i nositelj je definicije pojedinih znanja. Ovisno o znanju koje se definira, pojedini grafički objekti sučelja su aktivni (npr. padajući izbornik, lista) ili neaktivi (polja u tablici ili polja za upis). Za dodavanje novog znanja – elementi korisnik mora odabrati dugme **Dodaj** nakon čega se u tablici dodaje novi prazan redak. U novo kreiranom retku korisnik odabire **Stanje komponente** i komponentu (**Naziv**), te zadaje tip veze i odabire dokument vezan za znanje koje se dodaje. Komponente koje se prikazuju u padajućem izborniku **Naziv** prethodno su očitane iz CAD modela. Odabrano znanje, bilo da se radi o znanju – elementi ili ostalim znanjima, može se obrisati odabirom dugmeta **Obrisi**. Nakon definiranja znanja – elementi korisnik može definirati znanje – dokumenti odabirom kartice **DOKUMENTI**.

Prozor za definiranje konstrukcijskog znanja – dokumenti prikazan je na slici 8-10. Odabirom dugmeta **Dodaj** dodaje se novo znanje – dokumenti. Za svako novo dodano znanje potrebno je zadati putanju do dokumenta i putanju do izvršnog programa koji će se uporabiti za pregledavanje sadržaja dokumenta. Oba podatka zadaju se odabirom odgovarajućeg dugmeta **Pregledaj**. Nakon odabira navedenog dugmeta aktivira se standardni Windows dijalog prozor za pregledavanje sadržaja tvrdog diska na kojem se odabere željena datoteka ili program. Odabir se prikaže u odgovarajućem polju za upis. Ukoliko korisnik ne zada putanju do izvršnog programa za pregledavanje sadržaja dokumenta uporabit će se podrazumijevajući program ovisno o odabranom tipu dokumenta. Oblik dokumenta moguće je odabrati kao digitalni ili fizički. Za odabrano znanje – dokumenti neophodno je i definirati razinu sigurnosti i status znanja.



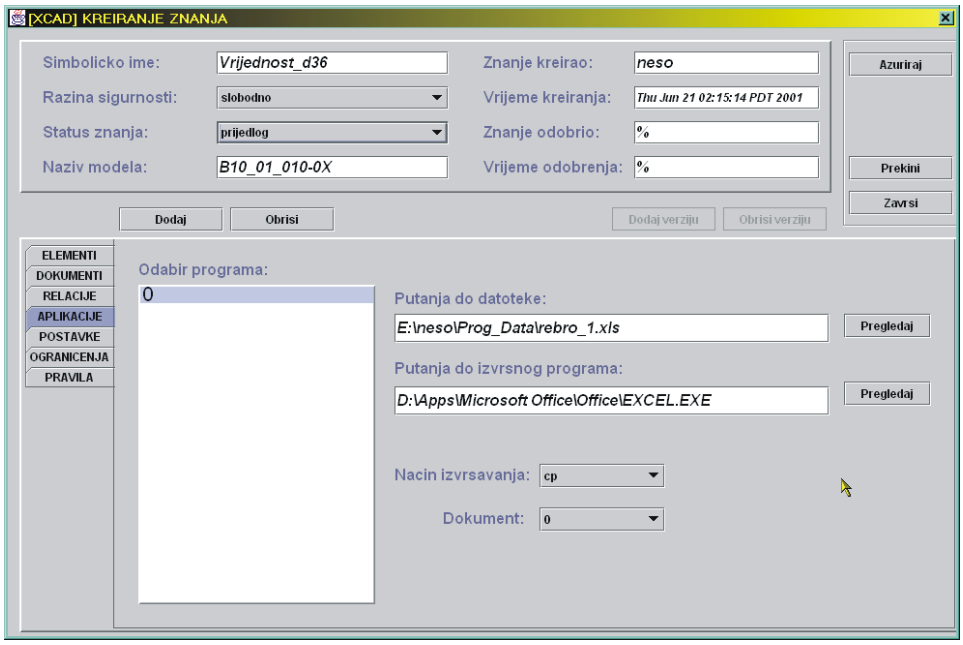
Slika 8-11 Prozor za definiranje konstrukcijskog znanja – RELACIJE

Odabirom kartice **RELACIJE** aktivira se prozor za definiranje konstrukcijskog znanja – relacije (slika 8-11). Novi redak u tablici kreira se odabirom dugmeta **Dodaj**. Nakon dodavanja novog retka korisnik zadaje način zadavanja vrijednosti parametra sa padajućeg izbornika **Tip** i simboličko ime parametra kojemu se definira vrijednost sa padajućeg izbornika **Parametar**.

Nadalje moguće je zadati eksplicitno vrijednost (ovisno o odabranom tipu zadavanja) parametra (polje **Vrijednost**), naziv dijela iz kojeg se očitava vrijednost parametra ili identifikacijska oznaka znanja – postavke pomoću kojeg se definira vrijednost parametra (polje

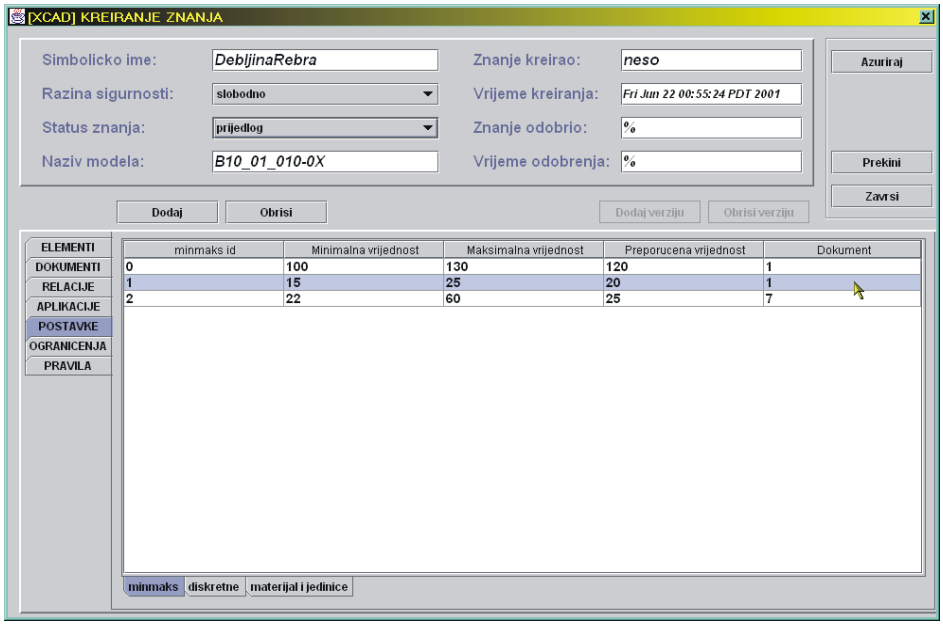
Izraz). Za definirano znanje – relacije moguće je vezati i znanje – dokument odabirom sa padajućeg izbornika **Dokument**.

Prilikom definiranja aplikacija veznih za aktivni prošireni CAD model potrebno je zadati simboličko ime aplikacije, putanju do radne datoteke i putanju do aplikacije te način izvršavanja programa i vezani dokument. Prozor za definiranje konstrukcijskog znanja – aplikacije prikazan je na slici 8-12.

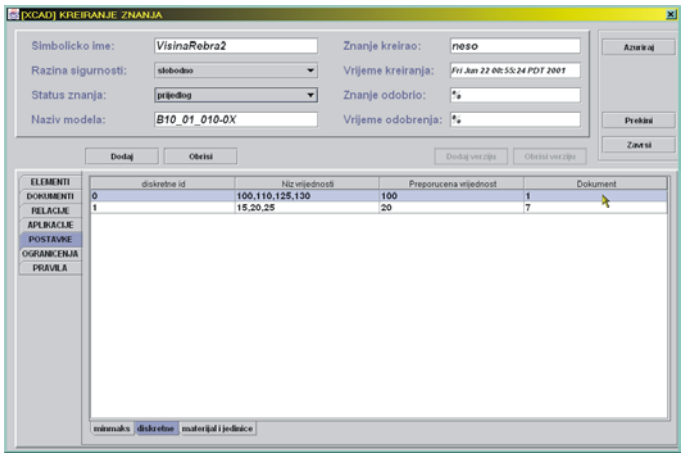


Slika 8-12 Prozor za definiranje konstrukcijskog znanja – APLIKACIJE

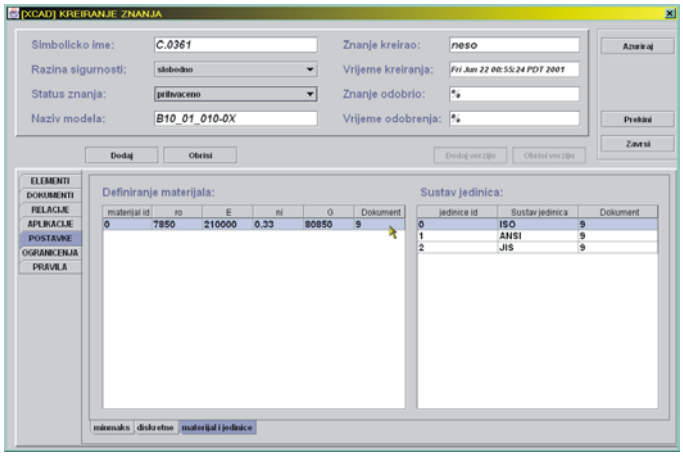
Postavke koje se koriste prilikom definiranja znanja – relacije (*minmaks*, *diskretne*), svojstva materijala i definicija sustava jedinica zadaju se odabirom kartica **minmaks**, **diskretne** te **materijal i jedinice** sa glavne kartice **POSTAVKE**. Primjer pojedinih postavki prikazan je na slikama: slika 8-13 **minmaks**, slika 8-14 **diskretne** te slika 8-15 **materijal** i **jedinice**.



Slika 8-13 Zadavanje **minmaks** postavki

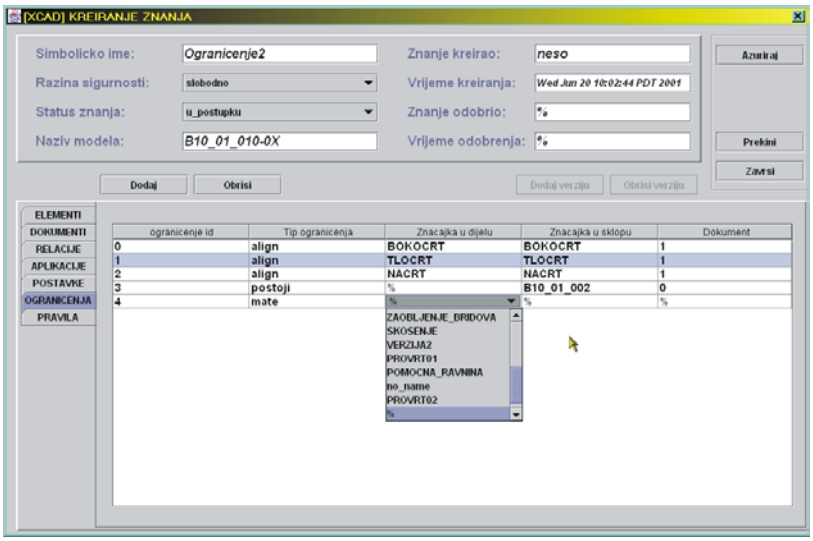


Slika 8-14 Zadavanje **diskretne** postavki



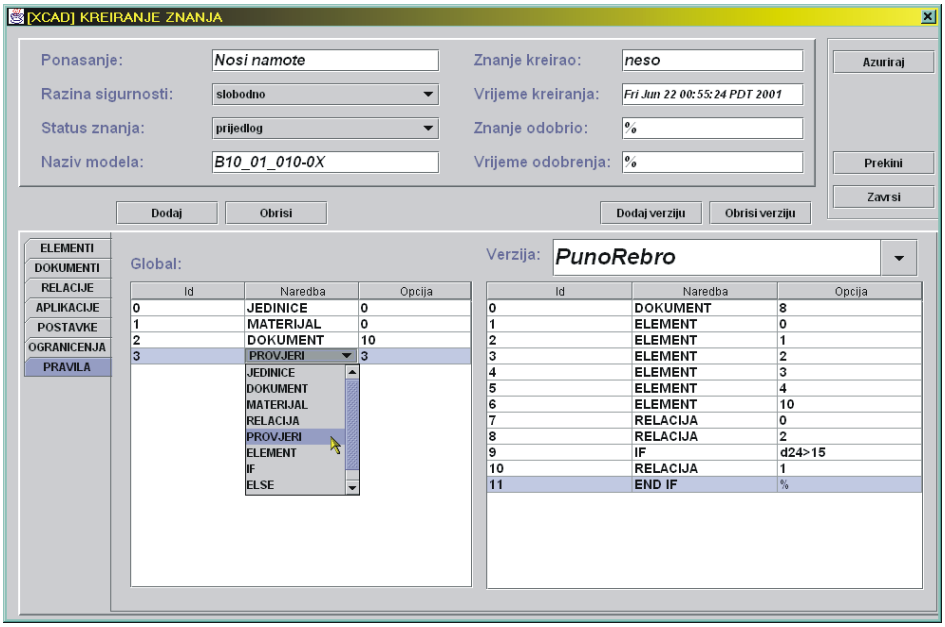
Slika 8-15 Zadavanje svojstava **materijala** i sustava **jedinica**

Na slici 8-16 prikazan je prozor za definiranje konstrukcijskog znanje – ograničenja. Nakon dodavanja novog retka korisnik odabire tip ograničenja (*mate*, *align*, *insert*, *orient*, *postoji*) sa padajućeg izbornika **Tip ograničenja**. Ovisno o odabranom tipu ograničenja potrebno je zadati značajku u aktivnom dijelu na koju se ograničenje odnosi (sa padajućeg izbornika **Značajka u dijelu**) i zadati naziv relevantne značajke u sklopu.



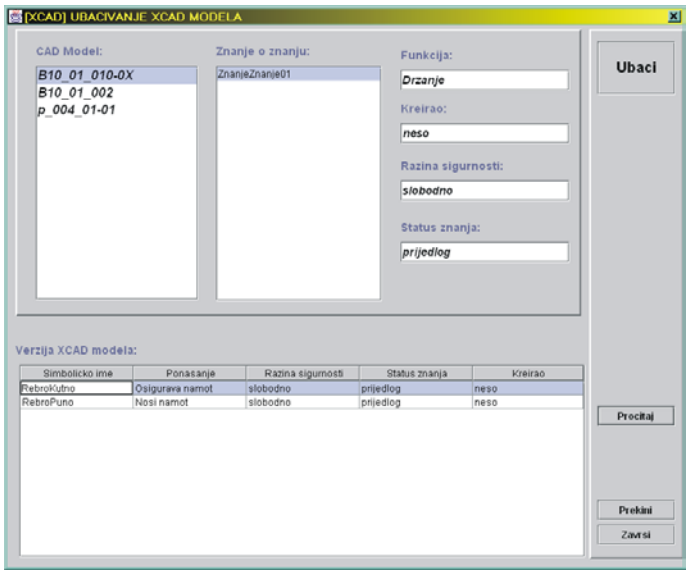
Slika 8-16 Prozor za definiranje konstrukcijskog znanje – OGRANIČENJA

Nakon zadavanja prikazanih konstrukcijskih znanja korisnik mora definirati pojedine verzije proširenog CAD modela (slika 8-17). Verzije proširenog CAD modela definiraju se odabirom kartice **PRAVILA**. Na ovoj kartici potrebno je zadati pravila koja vrijede za sve verzije (dio **Global**) i pravila koja vrijede za pojedinu verziju. Dodavanje i brisanje verzija ostvaruje se odabirom dugmeta **Dodaj verziju** ili **Obrisi verziju** dok se dodavanje i brisanje redaka u tablicama ostvaruje odabirom dugmeta **Dodaj** i **Obrisi**. Dijelovi verzije se definiraju odabirom ključnih riječi sa padajućeg izbornika **Naredba** i zadavanjem identifikacijske oznake znanja na koje se odabrana naredba odnosi. Prilikom zadavanja nove verzije potrebno je i zadati njezino ponašanje.



Slika 8-17 Prozor za definiranje verzija proširenog CAD modela

Definiranjem verzija proširenog CAD modela završen je dio definiranja znanja. Po završetku zadavanja konstrukcijskih znanja potrebno je odabrati dugme **Azuriraj** da bi se definirana znanja zapisala u bazu podataka.

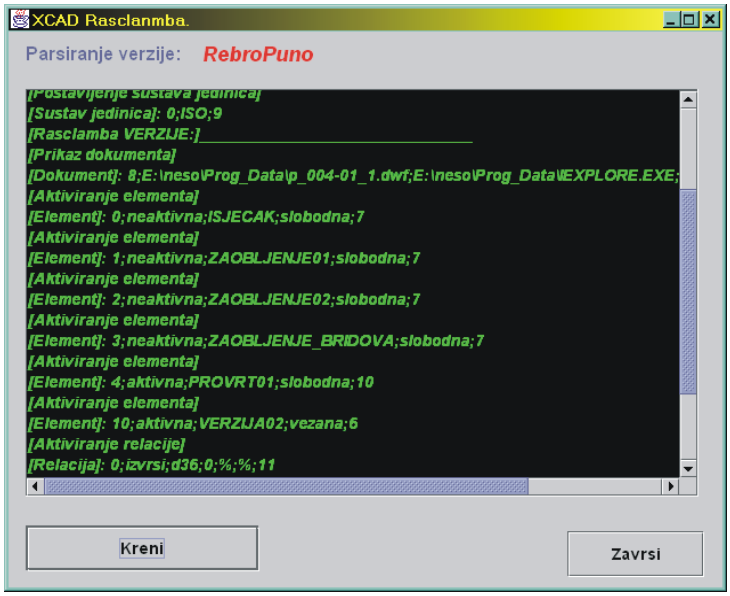


Slika 8-18 Prozor za ubacivanje proširenog CAD modela

Nakon završetka definiranja znanja korisnik može odabrati opciju **Ubaci** ukoliko želi ubaciti prošireni CAD model u aktivni sklop. Odabirom opcije **Ubaci** aktivira se prozor (slika 8-18) za odabir i ubacivanje proširenog CAD modela u aktivni sklop.

Odabirom dugmeta **Procitaj** iz baze podataka učitavaju se nazivi svih proširenih CAD modela i prikazuju u listi **CAD Model**. Odabirom nekog od ponuđenih naziva proširenog CAD modela u listi **Znanje o znanju** prikazat će se simbolička imena znanja o znanju definiranih za odabrani prošireni CAD modela. Korisnik odabirom pojedinog naziva znanja o znanju može pregledavati funkciju, kreatora te razinu sigurnosti i status odabranog znanja. Ujedno prilikom odabira znanja o znanju u donjem dijelu prozora, u tablici, prikazat će se nazivi verzija definiranih za odabrano znanje o znanju.

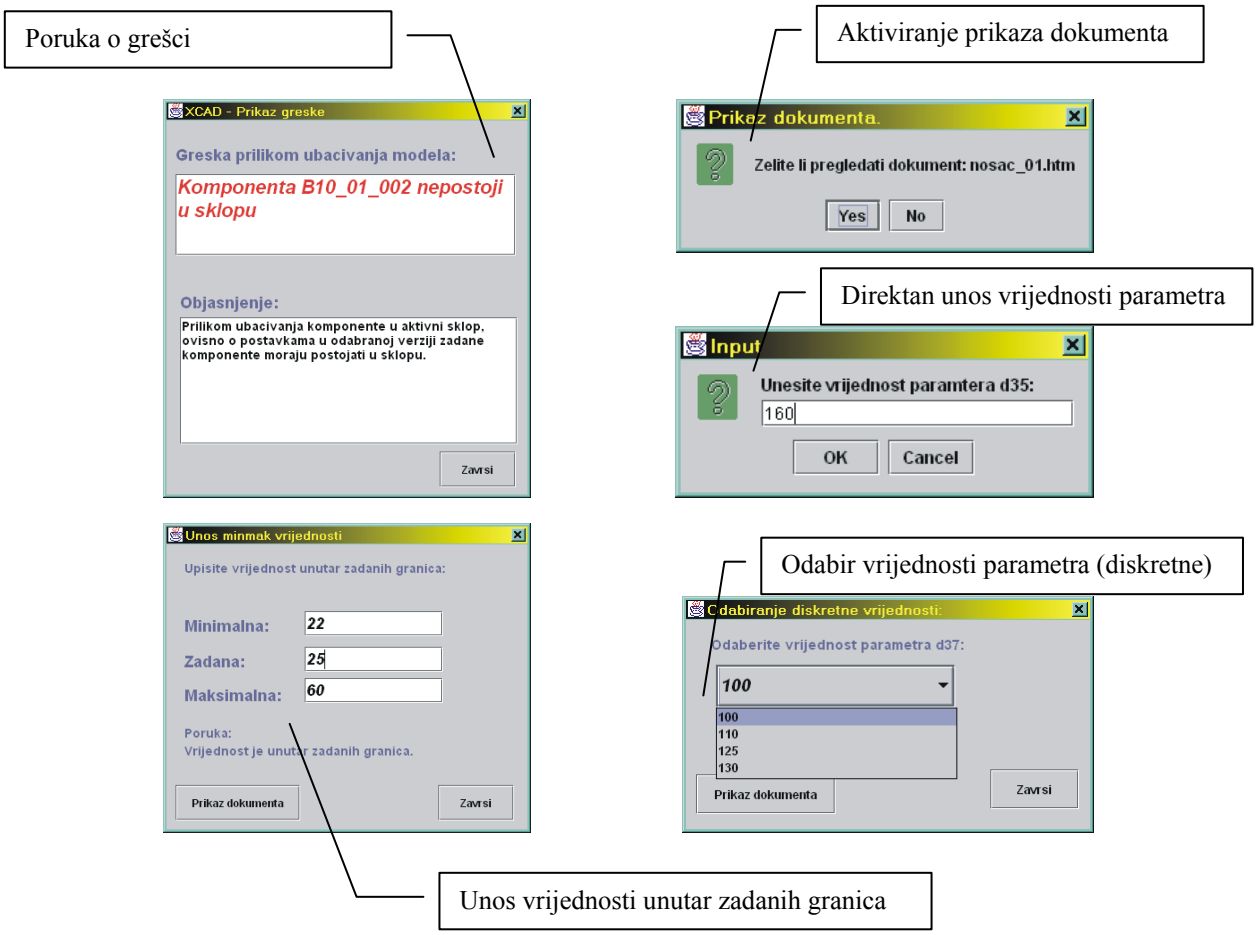
Pritiskom na dugme **Ubaci** odabrana verzija proširenog CAD modela biti će ubačena u aktivni sklop. Proces ubacivanja može se pratiti na informacijskom prozoru (slika 8-19).



Slika 8-19 Prozor s informacijama o procesu ubacivanja

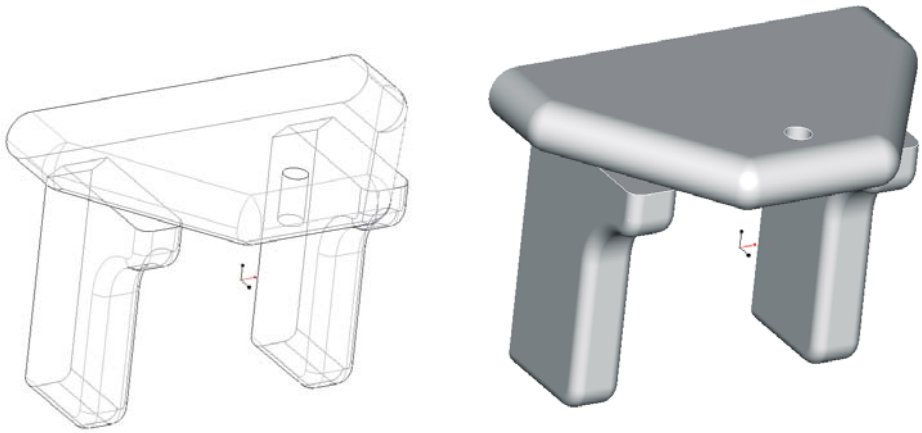
Prilikom raščlanjivanja odabrane verzije proširenog CAD modela korisnik može unositi vrijednosti pojedinih parametara ovisno o relacijama u verziji. Ukoliko prilikom raščlanjivanja dođe do pojave greške, ista se prikazuje na prozoru i korisniku se predlaže daljnji tijek akcija te ispisuje potpuni opis nastale greške. Na slici 8-20 prikazani su dijalozi za unos vrijednosti parametara.

Nakon što je odabrani prošireni CAD model ubačen u aktivni sklop vrijednosti parametara za ubačeni model zapisuju se u bazu instanci. Prilikom regeneracije sklopa, koji sadrži proširene CAD modele, za svaki model uspoređuju se trenutne vrijednosti parametara s vrijednostima parametara u bazi instanci te ukoliko se vrijednosti ne podudaraju generira se upozorenje i obavijesti se korisnik. Na slici 8-21(s kutnim rebrom) i 8-22 (s punim rebrom) prikazane su sklopovi kreirani uporabom proširenog CAD modela.

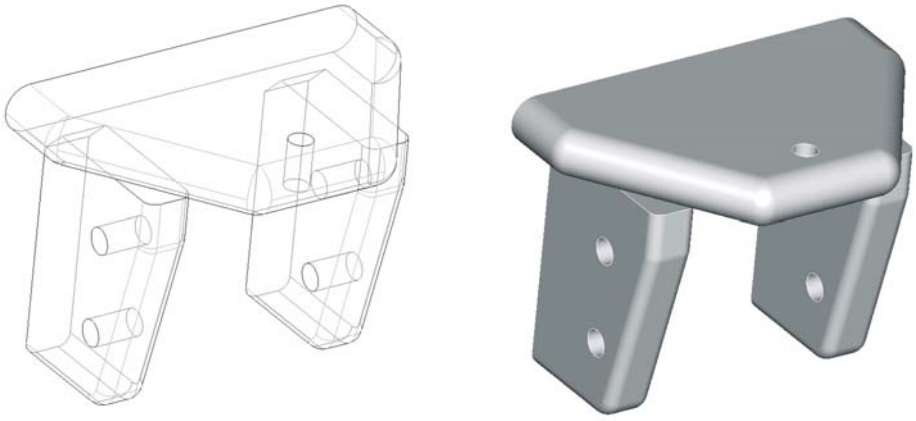


Slika 8-20 Prikaz prozora za unos vrijednosti parametara i prikaz poruka

Kreirani prošireni CAD modeli (dio s oznakom B10_01_002 i dio s oznakom B10_01_010-0X te sklop NOSAC_01) vezani su pravilima zaključivanja (koja se aktiviraju ovisno o akcijama na proširenom CAD modelu) sa definiranim konstrukcijskim znanjima te na taj način predstavljaju jedinstvenu cjelinu. Prošireni CAD modeli kreirani na opisani način ponašaju se kao aktivni objekti u procesu konstruiranja te na taj način olakšavaju i ubrzavaju rad konstruktora.



Slika 8-21 Prošireni CAD model (s kutnim rebrom)



Slika 8-22 Proširenі CAD model (s punim rebrom)

9 Zaključak

U uvodnom dijelu rada analizirane su karakteristike procesa konstruiranja, promatranjem konstruiranja kao procesa rješavanja konstrukcijskog zadatka. Dan je prikaz sadašnjeg stanja istraživanja u teoretskom i računalnom modeliranju proizvoda. U okviru ovih razmatranja analiziran je pristup računalnog modeliranja prikaza proizvoda uporabom značajki. Modeliranje uporabom značajki odabrano je kao platforma za kreiranje proširenog CAD modela proizvoda. Analizira različitih pristupa proučavanju znanja o proizvodu, opisanih u trećoj glavi rada, poslužila je kao osnova za kreiranje konceptualnog modela konstrukcijskog znanja. Zatim je razmatrana i opisana problematika modeliranja informacijskih sustava. U okviru ovog dijela rada detaljnije je razmatran standard za razmjenu informacija o proizvodu – STEP. Analizirane su metode modeliranja i projektiranja uporabom objektnog pristupa.

Prema postavljenom cilju ovoga rada trebalo je istražiti te realizirati koncept proširenog CAD modela proizvoda. Prošireni CAD model proizvoda treba uz geometrijske informacije, sadržane u CAD modelu, integrirati i konstrukcijsko znanje. U okviru rada kreiran je programski sustav za manipulaciju i upravljanje proširenim CAD modelom. Na osnovu provedenih istraživanja predložena je podjela znanja o proizvodu u programskom sustavu. Znanje je podijeljeno, prema ulozi u programskom sustavu, u tri kategorije: konstrukcijsko znanje, operativno znanje i proceduralno znanje.

Na osnovu analize znanja o proizvodu opisane u trećoj glavi rada predložena je struktura konstrukcijskog znanja o proizvodu. Predložena struktura kreirana je na osnovu vrste informacija koje sadrži, pojedino znanje, i utjecaja znanja na proizvod. Budući da je struktura konstrukcijskog znanja u predloženom obliku preopsežna, u radu se istraživanje ograničilo na znanje o strukturi uključujući i utjecajne faktore drugih znanja u okviru u kojem utječu na znanje o strukturi. Tako definirano znanje tvori konceptualni model konstrukcijskog znanja. Koncept modela konstrukcijskog znanja postavljen je tako da ne ovisi o domeni ili vrsti konstrukcijskog zadataka niti o programskom okruženju tj. uporabljenom CAD sustavu. Znanja o manipulaciji i upravljanju konstrukcijskim znanjem čine operativno znanje. Proceduralno znanje je usko vezano za programski sustav i implementaciju proširenog CAD modela. Konceptualni model operativnog i proceduralnog znanja predložen je u 6 glavi rada. U cilju kategorizacije i podjele modela, proširenog CAD modela, uvedeno je znanje o funkciji i znanje o ponašanju. Intencija uvođenja znanja o funkciji i ponašanju proizvoda je i u olakšavanju odabira modela proširenog CAD modela od strane konstruktora.

U daljnjem dijelu rada opisana je struktura računalnog modela konstrukcijskog znanja. Model konstrukcijskog znanja podijeljen je u logičke grupe: elementi, pravila, dokumenti, relacije, aplikacije, ograničenja i postavke. Logičke grupe čine gradbeni elementi znanja tj. entiteti znanja. Sintaksa zapisa entiteta znanja definirana je BNF notacijom. Na osnovu predloženog modela konstrukcijskog znanja definirana je struktura i relacije između podataka u bazi podataka. Struktura zapisa konstrukcijskog znanja sukladna je semantici STEP zapisa. Definiranje zapisa konstrukcijskog znanja u skladu sa semantikom STEP standarda omogućuje uporabu znanja i iz drugih programskih aplikacija.

Na osnovu predloženih modela znanja predložena je struktura programskog sustava za manipulaciju i upravljanje proširenim CAD modelom. U okviru strukturiranja programskog sustava definiran je implementacijski model konstrukcijskog znanja. Prilikom kreiranja implementacijskog modela koristio se objektni pristup tj. entiteti konceptualnog modela znanja preslikani su u klase objektnog modela. Osnovne elemente objektnog modela čine objekti, njihovi atribut i metode, te sa strane CAD aplikacije, koja služi za kreiranje geometrijskog dijela proširenog CAD modela, elementi geometrijskog modela (sklopovi, dijelovi i značajke). Metode zaključivanja implementirane su kroz proceduralno znanje. Sprega između CAD geometrijskog modela i zapisa znanja o proizvodu ostvarena je uporabom programskih biblioteka i razvijenih programskih procedura.

Sustav je realiziran uporabom programskog jezika JAVA. Kao platforma za implementaciju proširenog CAD modela uporabljen je Pro/ENGINEER CAD aplikacija. Veza između realiziranog programskog sustava i CAD aplikacije ostvarena je uporabom J-Link programskih biblioteka. Zapis strukture i relacija proširenog CAD modela ostvaren je uporabom MS Access baze podataka. Realizirani programski sustav kreiran je kao sučelje CAD aplikacije, tako da se korisnikove akcije prate tijekom njegova rada. Ovisno o korisnikovim akcijama aktiviraju se pojedini moduli programskog sustava, koji tada obavljaju zadaće u skladu s korisnikovim željama.

Realizirani sustav proširenog CAD modela testiran je na realnom primjeru konstrukcije dijelova jezgre energetskog transformatora. U CAD aplikaciji kreirani su geometrijski modeli dijelova jezgre. Za svaki kreirani geometrijski model aktiviran je realizirani sustav te su kreirana znanja i pohranjena u bazu podataka. Nakon kreiranja znanja u CAD aplikaciji kreiran je sklop, uporabom pohranjenih proširenih CAD modela. Prilikom kreiranja sklopa uporabljeni modeli manifestirali su *kvazi* inteligentno ponašanja čime je potvrđena hipoteza rada. *Kvazi* inteligentno ponašanje manifestiralo se u:

- prepoznavanju postojećeg stanja geometrijskog modela sklopa prilikom uporabe proširenih modela,
- aktiviranju programskih procedura i prikaza poruka korisniku ovisno o promjenama na uporabljenim proširenim modelima ili kreiranom sklopu,
- interakciji s korisnikom prilikom odabira i uporabe proširenog CAD modela,
- kontroli zadržavanja konzistentnosti sklopa i uporabljenih proširenih modela prilikom korisnikovih akcija.

Realizirani prošireni CAD model poslužiti će kao osnova za kreiranje naprednijih programskih alata za pomoć konstruktoru u procesu konstruiranja te kao podloga za koncipiranje modela koji mogu samostalno "učiti" i na taj način prikupljati informacije i znanja vezana za konstruiranje proizvoda.

Literatura:

[1] Bangemann M., Europe and the Global Information Society – Recommendations to the European Council, Brussels, 36. May 1994.

[2] P. L. McAlinden, B. Florida-James, K. Chao, P. Norman, Information and Knowledge Sharing for Distributed Design Agents, Proceedings of the 5th International Conference on Artificial Intelligence in Design, Lisbon, Portugal, 20-23 July 1998.

[3] M. Tanaka, N. Aoyama, A. Sugiura, Y. Koseki, Integration of Multiple Knowledge Representation for Classification Problems, Artificial Engineering in Engineering Vol. 9, No. 4, pp. 243. – 251., Elsevier, 1995.

[4] H. Merkamm, Information Management in the Design Process - Problems, Approaches and Solutions, Designers, the key to successful product development, Springer, Berlin, 1998.

[5] S. Kneebone, G. Blount, A Short Term Strategy for Product Knowledge Sharing and Re-Use, Proceedings of the 11th International Conference On Engineering Design, pp. 203. – 211., ICED 97, Tampere, august 1997.

[6] V. Salminen, E. Buckley, P. Malinen,J. Ritvas, S. Silakoski,A. Sauer, Global Engineering Networking – Turning Engineering Knowledge Into an Accessible Corporate Asset, Proceedings of the 11th International Conference On Engineering Design, pp. 165. – 173., ICED 97, Tampere, august 1997.

[7] J. Gausemeirer, L. Seifert, GENIAL: The Logical Framework for the Systematization of internal and External Engineering Knowledge, Proceedings of the 11th International Conference On Engineering Design, pp. 253. – 263., ICED 97, Tampere, august 1997.

[8] J. K. Gui, R. Launinen, Functional Deployment of Semantic Product Modeling in a Virtual enterprise, Proceedings of the 12th International Conference On Engineering Design, ICED 99, pp. 691. – 697., Münichen, august 1999.

[9] S. Wallmeier, P. Badke-Schaub., J Stempfle, H. Brikhofer, Empirical Diagnoses and Training in Design Departments, Proceedings of third international symposium on Tools and methods of competitive engineering, pp. 439. – 450, Delft, April 18. – 21. 2000.

[10] T. Ohmahci, A CAD System for Knowledge-based Mechanical Design and Optimization, Proceedings of the 11th International Conference On Engineering Design, pp. 239. – 247., ICED 97, Tampere, august 1997.

[11] S. Gorti, A. Gupta, J. G. Kim,R. D. Sriram,A. Wong, An object-oriented representation for product and design processes, Computer-Aided Design, Vol. 30, No. 7., pp. 489. – 501., Elsevier, Oxford, 1998.

[12] Y. W. Liang, P. O'Grady, Design with Objects: An Approach to Object-Oriented Design, TR98-01 http://www.uiowa.edu, 1998.

[13] G. Booch, Object Oriented Design with Applications, Benjamin – Cummings, 1994.

[14] R. Buhr, Machine charts for visual prototyping in system design, Technical Report 88-2, Carlton University, 1988.

[15] M. Jović, N. Bojčetić, D. Deković, Development of the Mechanical Assembly Description Model, Proceedings of the 5th International Design Conference Design '98,pp. 145. – 153., Dubrovnik, May 1998.

[16] D. Deković, N. Bojčetić, Structure of design domain knowledge base, Proceedings of the 12th International Conference On Engineering Design, pp. 1921. – 1925., ICED 99, Münichen, August 24-26, 1999.

[17] D. Deković, N. Bojčetić, Z. Herold, Implementation of Design Domain Knowledge in Design Process Environment, Proceedings of the 6th International Design Conference Design 2000,pp. 395. – 401., Dubrovnik, 23. – 26. May. 2000.

[18] D. Marjanović, N. Bojčetić, D. Deković, N. Pavković, D. Pavlić, M. Štorga, D. Žeželj, Design department data flow integration, Integrated Product Development Proceedings 3rd International Workshop, Vajna S., Clement S., Naumann T., Magdeburg , 2000.

[19] D. Marjanović, Implementacija ekspertnih alata u proces konstruiranja, Disertacija, FSB, Zagreb 1995.

[20] Proceedings of the IFIP TC 5/WG Workshop in Intelligent CAD, Boston, October 1987, Amsterdam, North-Holland, 1987.

[21] Proceedings of the IFIP TC 5/WG Second Workshop in Intelligent CAD, Cambridge, September 1988, Amsterdam, North-Holland, 1988.

[22] J. W. Scheregenberger, Attribution of Models, Proceedings of the 12th International Conference On Engineering Design, pp. 1191 – 1194., WDK, Münichen, August 1999.

[23] H Stachowiak H., Allgemeine Modeltheorie, Wien, Springer, 1973.

[24] M. M. Andreasen, B. A. H. Duffy, H. N. Moretensen, Relations in Machine Systems, Proceeding of WDK Workshop "Product Structuring", Delft University, 1995.

[25] V. Hubka, Theorie der Konstruktonsprozesse, Springer, 1976.

[26] G. D. Ullman, The Mechanical Design Process, McGraww-Hill, 1992

[27] D. Žeželj, N. Bojčetić, D. Marjanović, Interactive Model for Power Transformer's Core Design, Proceedings of the 13th International Conference On Engineering Design, ICED 01., WDK, Glasgov, August 2001.

[28] V. Akman, W. J. P. ten Hagen, T. Tomiyama, A fundamental and theoretical framework for an intelligent CAD system, CAD, Vol. 22, No. 6., pp. 352-367, Butterworth-Heinemann, Burgerss Hill 1990.

[29] B. Chandrasekaran, A Framework for Design Problem-Solving, Research in Engineering Design, Vol. 1, No. 2, 1989. , pp. 75. – 86., Springer-Verlag, New York, 1989.

[30] S. J. Colton, L. J. Dascanio, An Integrated, Intelligent Design Environment, Engineering with Computers, Vol. 7, No. 3, 1991., pp. 11. – 22., Springer, New York, 1991.

[31] K. N. Shaw, S. M. Bloor,A. de Pennington, Product Data Models, Research in Engineering Design, Vol. 1, No. 1., 43. – 50., Springer-Verlag, New York, 1989.

[32] C. A. McMahon, M. Xianyi, A Network Approach to Parametric Design Integration, Research in Engineering Design, Vol. 8, No. 1, pp. 14. – 32., Springer-Verlag, New York, 1996.

[33] S. Vajna, C. Burchardt, Dynamic Development Structures of Integrated Product Development, Journal of Engineering Design, Vol. 9, No. 1, pp. 3-16., Carfax publishing company, Oxford,1998.

[34] C. Hübel, B. Sutter, Supporting Engineering Applications by New Data Base Processing Concepts - An Experience Report, Engineering with Computers, Vol. 8, No. 1, pp. 31-49., Springer, New York, 1992.

[35] H. Meerkamm, Engineering Workbench - ein Schlüssel zur Lösung komplexer Konstruktionsprobleme, Proceedings of International Conference on Engineering Design ICED 95, pp. 1261. – 1268, WDK, Praha, august. 1995.

[36] J. Gauseimer, T. Frank, A. Hahn A., Integrated product development: An Integral Approach to Computer Aided Development of Advanced Mechanical Engineering Products, Proceedings of International Conference on Engineering Design ICED 95, pp. 1276. – 1289, WDK, Praha, august 1995.

[37] T. Jensen, An Empirical Study of Variant Design with a Designer's Workbench, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/277. – 282, WDK, Tampere, august 1997.

[38] Y. Bei, J. K. MacCallum, A Virtual Configuration Workbench for Product Development, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/337. – 342, WDK, Tampere, august 1997.

[39] S. J. Gero, Proceedings of fifth international conference on artificial intelligence in design - AID'98, pp. ix-x, Kluwer Academic Publishers, 1998.

[40] N. G. Blount, S. Clarke, Artificial Intelligence and Design Automation Systems, Journal of engineering Design, Vol. 5, No. 4, pp. 299. – 314., Carfax publishing company, Oxford,1994.

[41] J. Mostow, Towards Automated Development of Speialized Algorithms for Design Synthesis: Knowledge Compilation as an Approach to Computer-Aided Design, Research in Engineering Design, Vol. 1, No. 3, pp. 167. – 186., Springer, New York, 1990.

[42] S. G. Miller, S. J. Colton, The Complementary Roles of Expert Systems and Database Management Systems in a Design for Manufacture Environment, Engineering with Computers, Vol. 8, No. 3, pp. 139. – 149., Springer, New York,1992.

[43] T. Bardasz, I. Zeid, Applying analogical problem solving to mechanical Design, CAD, Vol. 23, No. 3, pp. 202. – 211., Butterworth-Heinemann, Burgerss Hill, 1991.

[44] J. Cagan,E. I. Grossman, J. Hooker, A Conceptual Framework for Combining Artificial Intelligence and Optimization in Engineering Design, Research in Engineering Design, Vol. 9, No. 1, pp. 20. – 34, Springer, New York, 1997.

[45] T. Mann, Expert systems for Design and Planning: Requirements and Expectations, Proceedings of International Conference on Engineering Design ICED 95, pp. 1565. – 1566, WDK, Praha, august 1995.

[46] G. D. Ullman, B. D' Ambrosio, A Proposed Taxonomy for Engineering Decision Support, International Conference on Engineering Design, pp. 611. – 617., ICED 95, WDK, Praha, august 1995.

[47] R. W. B. Forde, D. A. Russell, F. S. Stiemer, Object-Oriented Knowledge Frameworks, Engineering with Computers, Vol. 5, No. 4, pp. 79. – 89., Springer, New York, 1989.

[48] T. Smithers, Towards a knowledge level theory of design process, Proceedings of fifth international conference on artificial intelligence in design - AID '98, pp. 3. – 21, Kluwer Academic Publishers, 1998.

[49] R. Banares-Alcantara, Representing the engineering design process: two hypotheses, CAD, Vol. 23, No. 9, pp. 595. – 603., Butterworth-Heinemann, Burgerss Hill, 1991.

[50] A. D. Hoeltzel, H. W. Chieung, Factors that Affect Planning in a Knowledge-Based System for Mechanical Engineering Design Optimization with Application to the Design of Mechanical Power Transmissions, Engineering with Computers, Vol. 5, No. 2, pp. 47. – 62., Springer, New York, 1989.

[51] R. Žavbi, J. Duhovnik, Design environment for the design of mechanical drive units, CAD, Vol. 27, No. 10, pp. 769. – 781., Butterworth-Heinemann, Burgerss Hill, 1995.

[52] M. T. L. Blessing, R. N. Ball, Implementation and Initial Evaluation of a Process-Based Approach to Design, Proceedings of International Conference on Engineering Design ICED 97, pp. 2/271. – 2/276, WDK, Tampere, august 1997.

[53] R. N. Ball, P. Matthews, K. Wallace, Managing Conceptual Design Objects-an Alternative to Geometry, Proceedings of fifth international conference on artificial intelligence in design - AID '98, pp. 67. – 86, Kluwer Academic Publishers, 1988.

[54] T. D. Ndumu, A. B. Izzudin, D. L. Smith, Explaining Design Plans, Knowledge Based Systems, Vol. 9, No. 1, pp. 23. – 29., Elsevier, New York, 1996.

[55] H. Werner, C. Weber, State of the Art in Computer Based Design Tools (CBDT), European PhD-Seminar Engineering Design Research 2000, Baden-Baden, 2000.

[56] N. P. Suh, The Principles of Design, Oxford Univeristy Press, Oxford, 1990.

[57] W. E. Eder, Systematic Conceptualizing – With Computational Tools, Knowledge Intensive CAD, Volume 1, pp. 205. – 224, Chapman & Hall, Padstow, 1996.

[58] V. Hubka, W. E. Eder, Engineering Design - General Procedural Model of Engineering Design. Heurista, Zürich, 1992.

[59] V. Hubka, Principles of Engineering Design, Heurista, Zürich, 1988

[60] A. Kostelić, Znanost o konstruiranju, rukopis knjige, Zagreb, 1996.

[61] E. Oberšmit, Nauka o konstruiranju, metodičko konstruiranje i konstruiranje pomoću računala, SNL Liber, Zagreb, 1985.

[62] N. Murtagh, Artifact Configuration Across Different Design Levels, Knowledge Intensive CAD, Volume 1, pp. 137. – 151., Chapman & Hall, Padstow, 1996.

[63] R. Koller, Design Theory for the Mechanical Engineering, Spring-Verlang, 1985.

[64] G. Pahl, W. Beitz, Engineering Design, Springer-Verlang, 1994.

[65] H. Grabovski, R. S. Lossack, C. Weis, A Design Process Model based on Design Working Spaces, Knowledge Intensive CAD, Volume 1, pp. 245. – 262., Chapman & Hall, Padstow, 1996.

[66] A. I. Liewelyn, Review of CAD/CAM, Computer-Aided Design, Vol 21, No 5., pp. 297. – 302., Butterworth & Co, 1989.

[67] M. D. Kyran, The CRC Handbook of Mechanical Engineering, pp. 15.1. – 15.85, CRC Press, New York, 1997.

[68] J. J. Shah, M. Mantyla, Parametric and Feature-Based CAD/CAM, John Wiley & Sons, New York, 1995.

[69] M. F. Hordeski, CAD/CAM Techniques, Prentice-Hall, Reston, 1986.

[70] T. de Martino B. Falcidieno, S. Hasbinger, Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment, Computer-Aided Design, pp. 437. – 452., Vol. 30, No. 6, Elsevier, Oxford, 1998.

[71] R. Geelinek, O. W. Salomons, F. van Slooten, F. J. A. M. van Houten, H. J. J. Kals, Unified feature definition for feature based design and feature based manufacturing, University of Twente, http:// www.utwente.nl

[72] J. J. Shah, Conceptual development of form features and feature modelers, Research in Engineering Design, pp. 93. – 108, Vol. 3., Springer-Verlag, New York, 1991.

[73] J. J. Shah, Assessment of features technology, Computer-Based Design, pp. 58. – 66., Vol. 23., No. 5., 1991.

[74] C-X. Feng, C-C. Huang, A. kusiak, P-G. Li, Representation of functions and features in detail design, Computer-Aided Design, pp. 961. – 971., Vol. 28, No. 12., Elsevier, Oxford, 1996.

[75] M. K. Zamanian, S. J. fenves, C. R. Thewalt, S. Finger, A Feature-Based Approach to Structural Design, Engineering with Computers, pp. 1. – 9., Vol 7., Springer-Verlag, New York, 1991.

[76] S. Ohsuga, Toward intelligent CAD systems, Computer-Aided Design, pp. 315. – 337., Vol. 21., No. 5., Butterworth & Co, 1989.

[77] T. Tomiyama, Y. Umeda, M. Ishii, M. Yoshioka, Knowledge systematization for knowledge intensive engineering framework, Knowledge Intensive CAD Volume 1, pp. 33. – 52Chapman & Hall, New York, 1995.

[78] J-P. A. Barhes, Do we have the technology for supporting knowledge intensive CAD in large design projects?, Knowledge Intensive CAD Volume 1, pp. 23. - 33.Chapman & Hall, New York, 1995.

[79] J. M. Reddy, B. Chan, S. Finger, Patterns in design discourse: A case study, Knowledge Intensive CAD Volume 1, pp. 263. – 283.Chapman & Hall, New York, 1995.

[80] D. N. Batanov, A. K. Lekova, Data and knowledge integration through the feature-based approach, Artificial Intelligence in Engineering, pp. 77. – 83, Vol. 8., Elsevier, Oxford, 1993.

[81] Y. Zhang, A. H. B. Duffy, K. J. MacCallum, Requirements of computer-based modeling and management for product knowledge evolution support, Proceeding of 3rd WDK Workshop on Product Structuring, pp. 61. – 74, Delft, June 1997.

[82] Y. Zhang, K. J. MacCallum, A. H. B. Duffy, Product Knowledge Modeling and Management, Proceedings of the 2nd WDK Workshop on Product Structuring, pp. 141. – 150., Delft, June 1996.

[83] B. W. R. Forde, A. D. Russell, S. F. Stiemer, Object-Oriented Knowledge Frameworks, Engineering with Computers, pp. 79. – 89., Vol 5. Springer-Verlag, New York, 1989.

[84] W. Shen, J-P. Barthes, Na object-oriented approach for engineering design product modeling, Knowledge Intensive CAD Volume 1, pp. 170. – 187., Chapman & Hall, New York, 1995.

[85] Y. Shibata, An intellectual infrastrucure for integrating design knowledge, Knowledge Intensive CAD, pp. 3. – 19., Volume 2, Chapman & Hall, New York, 1996.

[86] J. Borg, K. J. MacCallum, Structuring knowledge of life-cycle consequences for supportong concurrent design exploration, Knowledge Intensive CAD Volume 2, pp. 208. – 224., Chapman & Hall, New York, 1996.

[87] B. Chandrasekaran, J. R. Josephson, Representing Function as Effect, http://www.utdallas.edu/~chandra/

[88] H. Takeda, Y. Shimomura, Y. Umeda, T. Tomiyama, Function Modeling: Confluence of Process Modeling and Object Modeling, http://is.aist-nara.ac.jp

[89] S. Prabhakar, A. K. Goel, Functional modeling for enabling adaptive design of devices for new environments, Artificial Intelligence in Engineering, pp. 417. – 444., Vol. 12, Elsevier, Oxford, 1998.

[90] H. Takeda, M. Yoshioka, T. Tomiyama, Y. Shimomura , Analysis of Design Process by Function, Behavior and Structure, http://is.aist-nara.ac.jp

[91] L. Chittaro, A. N. Kumar, Reasoning about function and its applications to engineering, Artificial Intelligence in Engineering, pp. 331. – 336., Vol. 12, Elsevier, Oxford, 1998.

[92] B. Chandrasekaran, J. R. Josephson, Function in Device Representation, http://www.utdallas.edu/~chandra/

[93] M .Rosenman, F. Wang, CADOM: A Component Agent-based Design-Oriented Model for Collaborative Design, Research in Engineering Design, pp. 193. – 205., Vol. 11., Springer-Verlag, New York, 1999.

[94] B. Chandrasekaran, J. R. Josephson, V. R. Benjamins, Ontology of Tasks and Methods, http://www.cis.ohio-state.edu/lair/

[95] A. N. Kumar, S. J. Upadhyaya, Component-onrtological representation of function for reasoning about devices, Artificial Intelligence in Engineering, pp. 399. – 415., Vol. 12., Elsevier, Oxford, 1998.,

[96] E. Fong, C. Dabrowski, M. Mitchell, C. K. Moriss, Database Management Systems in Engineering, National Institute of Technology, Gaithersburg, 1992.

[97] D. Schenk, P. Wilson, Information Modeling the EXPRESS Way, Oxford University Press, New York, 1994.

[98] J. C. Date, An Introduction to Database Systems, Addison-Wesley, Reading, 1991.

[99] J. Fowler, STEP for Data Management, Exchange & Sharing, Technology Appraisals, Twickenham, 1995.

[100] M. Varga, Baze podataka, DRIP, Zagreb 1994.

[101] J. M. Bernstein, H. Wong, A language facility for designing database intensive applications, ACM Trans. Database Systems, pp. 185. – 207., Vol 5., No 2., 1980.,

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

[103] W. Z. S. Su, Modeling integrated manufacturing with SAM*, pp. 34. –39.IEEE COmputing, Vol 19., No 1., 1986.,

[104] Deux, The Storz of O2, IEEE Trans. Knowledge and Data Engineering, Vol 2., No 1., 1990.

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

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

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

[108] IDEF1X Manual USAF Integrated Computer-Aided Manufacturing Program, D'Appleton Company, USA 1986.

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

[110] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, Reading, 1999.

[111] S. Brinkkemper, S. Hong, A. Bulthuis, G. van den Goor, The Object Modeling Technique, http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo-7.html

[112] S. J. Kemmerer, STEP The Grand Experience, NIST Special Publication 939, http://www.mel.nist.gov

[113] Introducing STEP, http://strategis.ic.gc.ca/STEPguide

[114] E. G. Schlechtendahl, CAD Data Transfer for Solid Models, Springer-Verlang, New York, 1989

[115] J. Owen, STEP An Introduction, Information Geometers, Midsomer Norton, 1993.

[116] J. S. Colton, R. C. Pun, Information Frameworks for Conceptual Engineering Design, Engineering with Coputers, pp. 22. – 32., Vol 10., Springer-Verlag, New York, 1994.

[117] A. Eliens, Object-Oriented Software Development, Addison-Wesley, New York, 1995.

[118] J. Robie, D. Bartels, A Comparison Between Relational nad Object Oriented Databases for Object Oriented Application Development, White paper from POET Software Corporation no. 800-950-8845, 1994.

[119] W-Y. Liang, P. O'Grady, Design with Objects; An Approach to Object-Oriented Design, http://www.uiowa.edu

[120] W. van Holland, W. F. Bronsvoort, An Object-Oriented Product Structure for Assembly Modelling, Proceedings of 2nd WDK Workshop on Product Structuring, pp. 151. – 157., Delft, June 1996.

[121] K. W. Tracy, P. Bouthoorn, Object-Oriented Artificial Intelligence Using C++, Coputer Scienece Press, New York, 1996.

[122] W. C. Regli, Digital Library Support for Engineering Design and Manufacturing, Proceedings of the 1999 ASME Design Engineering Technical Conferences, 12. – 15. September 1999., Las Vegas, Nevada

[123] V. Jagannathan, R. Dodhiawala, L. S. Baum, Blackboard Arhitectures and Applications, Academic Press, Boston, 1989.

[124] I. D. Craig, The Cassandra Architecture, Ellis Horwood, Chichester, 1989.

[125] A. G. Cohn, J. R. Thomas, Artificial Intelligence and its Applications, Wiley, Chichester, 1986.

[126] T. Tomiyama, T. Kiriyama, H. Yakeda D. Xue, H. Yoshikawa, Metamodel: A Key to Inelligent CAD systems, Research in Engineering Design, pp. 19. – 34., Vol. 1, No. 1, Springer-Verlag, New York, 1989.

[127] R. P. Cohen, A. E. Feigenbaum, The Handbook of Artificial intelligence, Vol. I, II, III, Addison-Wesley, 1986.

[128] ISO/IEC 1497, http://www.cl.cam.ac.uk/~mgk25/iso-ebnf.html

[129] Z. Dovedan, M. Smilevski, J. D. Zalokar, FORTRAN s tehnikama programiranja, Zveza za organizacij za tehnično kulturo Slovenije, Ljubljana, 1988.

[130] K. H. Law, T. Barslow, G. Widerhold, Management of Complex Structural Objects in a Relational Framework, Engineering wirh Computers, pp. 81.-92., Vol. 6., Springer-Verlag, NewYork, 1990.

[131] P. Gu, K. Chan, Product modelling using STEP, Computer-Aided Design, pp. 163. – 181., Vol. 27, No. 3., Butterworth-Heinemann, Oxford, 1995.

[132] D. Schenck, P. Wilson, Information Modeling the EXPRESS Way, Oxford University Press, New York, 1994.

[133] ISO 10303 – 11, Description methods: The EXPRESS language reference manual, ISO, Geneve, 1994.

[134] JAVA Unleashed, Sams.net, Indianapolis, 1996.

[135] R. Eckstein, M. Loz, D. Wood, JAVA Swing, O'reilly, Sebastopol, 1998.

[136] B. Excel, Thinking in JAVA, Prentice Hall, New Jersey, 1998.

[137] J-Link User's Guide, PTC - Documentation, Waltham, 2000.

[138] B. Stroustrup, The C++ Programming Language, Addison-Wesley, New York, 1993.

[139] P. H. Winston, Artificial Intelligence, Addison-Wesley, New York, 1993.

[140] G. F. Luger, W. A. Stubblefield, Artificial Intelligence – second edition, Addison-Wesley, New York, 1993.

[141] R. Eckstein, M. Loy, D. Wood, Java Swing, O'Reilly, Sebastopol, 1998.

[142] H. S. Mahendra, Systematic Mechanical Design: A Cost and Management Perspective, ASME Press, New York, 1997.

Životopis

Nenad Bojčetić rođen je 26. 10. 1965. godine u Sisku gdje je završio srednju školu. Studij strojarstva završio je na Fakultetu strojarstva i brodogradnje 1991. godine. Obranom teme "Korisničko sučelje sustava za konstruiranje mehaničkih sklopova" magistrirao je 1996. godine također na Fakultetu strojarstva i brodogradnje.

Od 1991. godine zaposlen je na Fakultetu strojarstva i brodogradnje kao znanstveni novak, a od 1993. godine kao asistent na Katedri za osnove konstruiranja Zavoda za konstruiranje. Tijekom rada na fakultetu sudjeluje u nastavi sljedećih kolegija: Uvod u računala, Primjena računala-K, Odabrane metode konstruiranja, Metode umjetne inteligencije, Konstruiranje pomoću računala. Kao honorarni asistent na Studiju Dizajna pri Arhitektonskom fakultetu izvodi vježbe iz kolegija: Osnove primjene računala i Oblikovanje pomoću računala. Na strojarskom odsjeku Tehničkog veleučilišta u Zagrebu izvodi vježbe iz kolegija Osnove računarstva i Konstruiranje pomoću računala . Voditelj je Laboratorija za konstruiranje (CADLab) Katedre za osnove konstruiranja.

Magistarski rad pod nalovom "Korisničko sučelje sustava za konstruiranje mehaničkih sklopova" obrani je 19. 12. 1996. godine

Znanstveni rad obuhvaća područja teorije konstruiranja, uporabe CAD/CAM/CAE programskih paketa i tehnologija, te tehnika programiranja. Osim u nastavi, aktivno je sudjelovao u organizaciji znanstvenih skupova DESIGN '98 i DESIGN 2000.

Aktivno sudjeluje u znanstvenom radu na projektima od 1991. godine, i to na projektu broj 2-08-173, "Model inteligentnog CAE sustava" (glavni istraživač prof. dr. sc. Aurel Kostelić). Od 1996. godine radi na projektu "Model inteligentnog CAD sustava" (broj 120-015, glavni istraživač prof. dr. sc. Dorian Marjanović)

Kao autor ili koautor objavio je 14 znanstvenih i 11 stručnih radova u Hrvatskoj i inozemstvu. Član je Hrvatskog društva za elemente strojeva i konstrukcije. Vlada engleskim jezikom.

Biography

Nenad Bojčetić was born on October 26th 1965. in Sisak where he completed his secondary school education. In 1991. he graduate in mechanical design form the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb. Since 1991. he has been an science novice and from 1993. an assistant at the Design Department of the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb.

In 1996. Nenad Bojčetić acquired the M. Sc. degree at the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb with the thesis "The Model of the User Interface for System for Mechanical Components Design" .

Besides teaching activities he took part in the organization of the international design conferences DESIGN '98 and DESIGN 2000 both held in Dubrovnik.

As the author or coauthor he has published 14 scientific papers and 11 technical reports in Croatia and abroad. He has good command of English.