SVEUCILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

OBJEKTNO ORIJENTIRANI PRISTUP MODELIRANJU PROCESA KONSTRUIRANJA

Disertacija

Mr. sc. **Neven Pavkovic**, dipl. inž.

Mentor:
Prof. dr. **Dorian Marjanovic**, dipl. inž.

Zagreb, 2000.

PODACI ZA BIBLIOGRAFSKU KARTICU

UDK:	658.512.2:681.3:621
Kljucne rijeci:	teorija konstruiranja, proces konstruiranja, konstruiranje pomocu racunala, objektno orijentirano modeliranje, objektno orijentirano programiranje, planiranje, prikazivanje procesa, objektne baze podataka
Znanstveno podrucje:	Tehnicke znanosti
Znanstveno polje:	Strojarstvo
Institucija u kojoj je rad izraden	Sveucilište u Zagrebu Fakultet strojarstva i brodogradnje
Mentor:	Prof. dr. sc. Dorian Marjanovic
Broj stranica:	147
Broj slika	63
Broj tablica:	11
Broj korištenih bibliografskih jedinica:	142
Datum obrane:	
Povjerenstvo:	Prof. dr. sc. Bojan Jerbic Prof. dr. sc. Dorian Marjanovic Prof. dr. sc. Jože Duhovnik Prof. dr. sc. Dragutin Ščap Prof. dr. sc. Izvor Grubišic
Institucije na kojima je rad pohranjen:	Fakultet strojarstva i brodogradnje Zagreb Nacionalna i sveucilišna knjižnica Zagreb

Zahvaljujem mentoru, prof. dr. Dorianu Marjanovicu na savjetima, korisnim raspravama i podršci u tijeku izrade ovog rada.
Primjedbe i savjeti clanova povjerenstva takoder su puno pomogli da konacna verzija rada bude kvalitetnija.

Djelatnici Laboratorija za osnove konstruiranja doprinijeli su ovom radu kroz brojne analize i rasprave tijekom zajednickih istraživanja na projektu kojeg je dio i ova disertacija.

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

Na kraju, posebno se zahvaljujem svojoj supruzi i roditeljima na velikom strpljenju i podršci.

SADRŽAJ

POPIS SLIKA:

POPIS TABLICA:

PREDGOVOR

Izrada ovog rada potaknuta je višegodišnjim istraživackim radom u podrucju unapredenja racunalne podrške procesu konstruiranja strojarskih proizvoda, te iskustvom autora u primjeni i razvoju elemenata CAD sustava.

Postojecim CAD sustavima nedostaje podrška odvijanju (modeliranju) konstrukcijskog procesa. Razvijeni teoretski modeli procesa konstruiranja vecinom nisu pogodni za direktnu racunalnu implementaciju. Promatrajuci postojece racunalne alate koji se koriste u procesu konstruiranja, takoder je potrebno istražiti principe i metode njihove integracije u cjeloviti (samim time i efikasniji) sustav za racunalnu podršku razvoju proizvoda. Unapredenje mogucnosti i efikasnosti primjene CAD sustava treba pridonijeti povecanju produktivnosti i boljoj organizaciji procesa konstruiranja, a time i povecanju profitabilnosti cjelokupnog proizvodnog sustava.

Istraživackim projektom broj 120-015 "Model inteligentnog CAE sustava" predvideno je istraživanje modeliranja racunalne podrške tijeku i kontroli odvijanja konstrukcijskog procesa, te je ova disertacija dio cjelokupnih istraživanja unutar navedenoga projekta. Ocekuje se da objektno orijentirani pristup modeliranju procesa konstruiranja doprinese fleksibilnosti i otvori mogucnosti implementacije više parcijalnih modela i metoda unutar istog okvira.

SAŽETAK

Tema ove disertacije razvoj je osnovne strukture objektno orijentiranog modela procesa konstruiranja. Predložena struktura trebala bi se upotrijebiti kao sustav "otvorene kutije s alatima". Pri tome je sustav koncipiran tako da njegova primjena ne ovisi o vrsti konstrukcije, niti je vezana za odredenu fazu procesa konstruiranja. U takvom pristupu, svaki pojam, relacija i druge "realne stvari" iz domene procesa konstruiranja pokušale su se modelirati kao objekti. Pretpostavljeno je da objektno orijentirana metodologija nudi pogodnije i fleksibilnije nacine modeliranja programskih sustava od drugih raspoloživih tehnika. Proces konstruiranja promatran je kao niz transformacija od inicijalnog stanja skupa podataka, ogranicenja i ciljeva do konacnog stanja koje predstavlja potpuni opis proizvoda koji se konstruira. Takve transformacije obavljaju pojedine osobe ili timovi, te izdvojeni ili integrirani programski alati.

Glavni rezultat istraživanja u ovom radu je prijedlog i definicija skupa osnovnih entiteta konceptualnog modela procesa konstruiranja. Osnovni strukturalni entiteti definirani su kao: konstrukcijski parametar, objekt prikaza proizvoda, objekt prikaza strukture proizvoda, sucelje programskog alata, akcija, konstruktor (kao aktivni djelatnik procesa), te konstrukcijski zadatak. Navedeni entiteti preslikani su u osnovne klase objektno orijentiranog racunalnog modela procesa konstruiranja. Kao središnja tema istraživanja, analizirana je kompleksna mreža relacija izmedu entiteta predloženog modela procesa konstruiranja. Pokazalo se da je takvu mrežu relacija moguce efikasno modelirati (i njome manipulirati) u okruženju objektne baze podataka. Model procesa konstruiranja graden je "bottom up" pristupom - osnovni strukturalni entiteti i mreža relacija koriste se kao gradbeni elementi složenih entiteta koji modeliraju tijek i kontrolu odvijanja procesa konstruiranja. Proces konstruiranja prikazan je planom - skupom cvorova i njihovih veza u usmjerenom grafu. Veze izmedu cvorova prikazuju tokove podataka i/ili unaprijed planirane redoslijede izvršavanja cvorova. Analizirane su moguce topologije prikaza plana, te procesi kreiranja i izvodenja plana, uz naznake mogucnosti implementacije tzv. "dinamickog planiranja" - mijenjanja plana u tijeku njegova izvodenja. Cvor plana konstruiranja modelira jednu etapu procesa konstruiranja koja ukljucuje provjeru preduvjeta, listu akcija, provjeru postuvjeta, te odlucivanje o daljnjem tijeku procesa. Preduvjeti i postuvjeti sadrže skupove ogranicenja i pravila u kojima se referenciraju parametri konstrukcije i atributi svih klasa objekata predloženog modela.

Struktura predloženog modela dokumentirana je u UML jeziku, korištenjem programskog paketa "Rational Rose 2000". Primjer implementacije modela realiziran je u "POET" objektnoj bazi.

Kljucne rijeci :

inženjersko konstruiranje, teorija konstruiranja, proces konstruiranja, konstruiranje pomocu racunala, modeliranje procesa konstruiranja, objektno orijentirano modeliranje, UML, objektne baze podataka, planiranje, prikazivanje procesa

UDK 658.512.2:681.3:621

OBJECT-ORIENTED APPROACH TO DESIGN PROCESS MODELLING

SUMMARY

The subject of this thesis is the development of the object-oriented design process model framework. The proposed framework should be used as an open toolbox, independently of the design task class and the design process phase. In such an approach, every occurrence, relation and other real-world 'things' from the domain of the design process are attempted to be modelled as objects. It is assumed that the object-oriented methodologies can provide more appropriate and flexible ways of software system modelling than the other techniques. The design process is here viewed as a sequence of transitions from an initial state of data, constraints and goals to a final state: a complete description of the mechanical artefact being designed. These transitions are allocated to individual participants or teams, and individual or integrated computational tools or models.

The main results of this research are identification and description of the basic structural, relational and behavioural entities of the design process model. The proposed model is built upon following basic structural entities: design parameter, product description object (any type of document), product structure object (part), interface to external software tool, action, designer (as an “actor” in the process) and the design task. These entities are modelled as the basic classes of the object-oriented design process model. The complex network of relations between design process model entities is analysed as a central issue of the presented research. It turned out that a very large and complex object network could be efficiently modelled and managed in an object database environment. Thus, a design process model is built using a "bottom up" approach in which the basic structural entities and the network of relations are used as a building blocks for more complex entities which model and control the design process flow. A design process is represented with design plan - a collection of nodes and their connections in a directed graph. The connections between nodes represent the information flow and/or the preplanned execution paths. Possible design plan topologies, the plan generation and exploitation issues are discussed, as well as the possibilities of implementing dynamic changes while the plan is being executed. A design plan node models one step of the design process, including: checking of preconditions, list of actions, checking the postconditions and deciding about the next step. Preconditions and postconditions include sets of constraints and rules. Constraints and rules include references to design parameters and attributes of all classes of objects that constitute the design process model.

The structure of the proposed model is documented in the UML language, using "Rational Rose 2000" software tool. An example of "real world" implementation is realised in POET object database.

Keywords :

engineering design, design theory, design process, computer-based design support, design process, design process modelling, object-oriented technology, UML, object databases, planning, process representations

UDC 658.512.2:681.3:621

1. Uvod

Zadatak inženjera je primjeniti svoje znanje u rješavanju tehnickih problema i optimirati to rješenje u danim ogranicenjima materijala, tehnologije i ekonomije. Radom na razvoju proizvoda - kreiranjem materijalnog svijeta, inženjeri doprinose unapredenju opce kvalitete života. Razvoj proizvoda, kao temelj inženjerskog djelovanja, može se podijeliti na konstruiranje, ili idejno stvaralaštvo, te samu izradu, odnosno proizvodno stvaralaštvo.
Znanje, ideje i sposobnosti konstruktora imaju fundamentalni utjecaj na prirodu izradenog proizvoda, njegov status na tržištu i njegovu ukupnu profitabilnost [1]. Konstruiranje je intelektualno nastojanje da se zadovolje odredeni zahtjevi na najbolji moguci nacin. To je inženjerska aktivnost koja djeluje na gotovo svaku sferu ljudskog života, temelji se na otkricima i zakonima znanosti i kreira uvjete za primjenu tih zakona na proizvodnju korisnih proizvoda [2].
Važnost konstruiranja (i s ekonomskog aspekta, i za opci razvoj civilizacije) dovodi do razvoja nove znanstvene discipline - Znanosti o konstruiranju. Unatoc velikom napretku istraživanja, naše razumijevanje konstruiranja ipak još nije doseglo potpunu zrelost [3]. Istraživaci vjeruju da je moguce dublje i fundamentalnije razumijevanje konstruiranja bez žrtvovanja kreativnosti koja generira nove proizvode, strojeve i sustave [3].
U dosadašnjem razvoju Znanosti o konstruiranju razvijeno je nekoliko znacajnijih opcenitih teorija konstruiranja, kao i veliki broj parcijalnih modela koji opisuju pojedine aspekte ili domene konstruiranja. Svejedno, može se reci da niti jedna od razvijenih teorija nije sama za sebe dovoljna za potpuni opis i formalizaciju procesa konstruiranja na nacin koji bi omogucio razvoj integrirane racunalne podrške [4].
Pri modeliranju racunalne podrške, proces konstruiranja može se tretirati kao niz transformacija informacija i generiranja informacija, od pocetnog stanja (zahtjeva) do krajnjeg stanja - potpunog opisa proizvoda. Proces konstruiranja ne bi trebalo promatrati kao staticku institucionaliziranu strukturu, nego kao dinamicku mrežu koja se stvara u realnom vremenu kako ciljevi, potrebe, prioriteti i resursi evoluiraju [5]. U okruženju integriranog sustava racunalne podrške, konstruktor bi trebao raditi sa "skupom alata" koji mu omogucuju da kreira parcijalne modele procesa konstruiranja prema svojim potrebama. Takav sustav trebao bi biti neovisan o vrstama konstrukcijskih zadataka, a ne bi ga trebalo vezati niti uz faze procesa konstruiranja.
Prednosti koje pruža objektna tehnologija (u odnosu na druge metode) trebala bi omoguciti razvoj željene strukture programskog sustava za kompletnu racunalnu podršku procesu konstruiranja. Objektno orijentirana analiza zapocinje ispitivanjem "stvari iz realnog svijeta", odnosno domene problema koji se rješava. Te "stvari" (objekti ili entiteti) karakterizirane su atributima i ponašanjem. U analizi se formiraju i opisuju "klase" iz domene problema i paralelno se modeliraju veze i "suradnja" medu objektima. Nakon toga, prelazi se sa modeliranja domene problema na modeliranje domene implementacije. Jedna od najvecih prednosti objektne tehnologije je u tome što su objekti isti i u domeni problema i u domeni implementacije, što uvelike olakšava razvoj i održavanje sustava. Objektno orijentirana metodologija modeliranja programskih sustava intenzivno se razvija posljednjih desetak godina i dovoljno je sazrela da bi krenula prema standardizaciji.

1.1 Opis zadatka (definicija problema)

Razvoj racunalne opreme i novih metoda programiranja otvorio je i nove mogucnosti za podizanje racunalne podrške konstruiranju na višu razinu. Postojecim CAD sustavima nedostaje model procesa konstruiranja i alati za povezivanje akcija, tj. planiranje i podršku prirodnom toku promišljanja konstruktora u tijeku rješavanja konstrukcijskog zadatka. Razvijene metode prikaza konstrukcijskog procesa koje se koriste u današnjim CAD sustavima ogranicenih su mogucnosti, te efikasno funkcioniraju samo kod dobro definiranih problema uske domene. Programski sustavi opce namjene redovito su koncipirani kao skup zasebnih programskih cjelina za podršku odredenim podrucjima aktivnosti konstruiranja (geometrijsko modeliranje, generiranje NC programa, analiza mehanickih svojstava, itd.), ali bez integriranog modela procesa konstruiranja.

Konstrukcijski ured cesto je usko grlo u procesu proizvodnje, a ujedno i mjesto gdje se koncentrira i obraduje najveci dio informacija o proizvodu. Konstruktor i/ili projektant trebao bi stoga imati na trenutnom raspolaganju prakticki sve informacije vezane uz proizvod, od ugovaranja do povratnih informacija iz eksploatacije. Upravo tu se ocituje nužnost integracije informacijskih tokova, pogotovo kod velikih proizvodnih sustava. Stoga se sve više ukazuje potreba za integracijom raznorodnih postojecih aplikacija, kao i za razvojem opcenitog postupka prijenosa podataka izmedu njih. Racunalni model procesa konstruiranja trebao bi podržati prirodni tok promišljanja, ali u isto vrijeme poslužiti i kao platforma za integraciju CAD programskih alata te baza podataka i znanja o proizvodu i procesu proizvodnje u konzistentno okruženje. U dosadašnjim istraživanjima [6], pokazalo se da realizacija ovakvih kompleksnih zahtjeva u relativno jednostavnijim programskim okruženjima (npr. relacijska baza podataka) zahtijeva velik utrošak programerskih resursa uz neizvjesne konacne rezultate.

Racunalni pristupi tretiranju procesa konstruiranja neovisno o domeni i stupnju ponovljivosti konstrukcije relativno su rijetki u literaturi. Iterativnost i intuitivnost cine modeliranje procesa konstruiranja vrlo zahtjevnim i kompleksnim pa se opceniti pristupi pretežno svode na teorijske modele, dok se racunalni modeli procesa konstruiranja uglavnom svode na primjenu odredene metode konstruiranja ili su ograniceni unutar odredene domene. Modeliranjem racunalne podrške planiranju i izvodenju konstrukcijskog procesa može se ostvariti samo približan model realnog procesa konstruiranja. Jedan od mogucih pristupa unapredenju racunalne podrške konstrukcijskog procesa je koncipiranje skupa programskih alata koji cine okruženje za modeliranje prikaza tijeka procesa konstruiranja, neovisno o domeni i vrsti konstrukcijskog zadatka.

Stupanj približenja racunalnog modela realnosti ovisi o:

- spoznajama teorije konstruiranja - razumijevanju procesa konstruiranja
- mogucnostima današnjih informatickih tehnologija (više programskih nego hardverskih) i metodologiji njihove primjene.

Objektno orijentiranom tehnikom programiranja moguce je koncipirati strukturalno vrlo kompleksne modele uz relativno razuman utrošak programerskih resursa.

Proces konstruiranja može se rašclaniti na strukturu operacija i aktivnosti koje se mogu promatrati kao procesi transformacije informacija unutar skupa informacija o proizvodu koji se konstruira. Tako poopceni proces može se opisati kao djelatnost koja se sastoji od niza akcija i njihovih medusobnih relacija. Svaka od akcija vrši pretvorbu informacija, tj. na temelju skupa ulaznih informacija generira skup izlaznih informacija. Akcije su etape unutar procesa, a granicna stanja etapa (odnosno medustanja) možemo zamisliti kao prostor stanja procesa. Svaki proces sastoji se od slijeda aktivnosti koji se može formulirati planom.

Rad se osniva na pretpostavci da jezgru modela tijeka procesa konstruiranja treba ciniti racunalni zapis plana modeliranog mrežom cvorova koji predstavljaju izvodenje akcija. Svaka akcija odnosno operator kao i njihove medusobne relacije u procesu konstruiranja mogu se prikazati kao objekti sa pripadajucim svojstvima. Prednosti nacela objektne tehnike programiranja prvenstveno trebaju omoguciti realizaciju mrežnog modela prostora stanja procesa, ali i koncipiranje kompleksne i fleksibilne strukture modela procesa konstruiranja, kao i implementaciju metoda umjetne inteligencije. Strukture inženjerskih podataka daleko su raznorodnije i složenije od podataka u npr. poslovnim aplikacijama. Veze medu strukturama su mnogobrojne, a iste strukture mogu imati razlicite uloge. Topologija procesa konstruiranja može se promatrati kao dvije kompleksne mreže relacija:

- složene višerazinske veze izmedu struktura inženjerskih podataka
- mreža relacija redoslijeda izvodenja etapa procesa, koja redovito ukljucuje i iterativne podprocese

Istraživanja ovog rada usmjeriti ce se na mogucnosti racunalnog modeliranja navedenih mreža relacija. Te dvije mreže relacija ne mogu se promatrati odvojeno, nego se i one medusobno prožimaju, tj. relacije jedne grupe mogu ukljucivati relacije druge grupe, kroz više razina referenciranja. Pod pojmom "mrežna topologija procesa konstruiranja", u daljnjem tekstu podrazumijevati ce se topologija spomenutih mreža relacija.

1.2 Cilj istraživanja i hipoteza

Naprednije metode upravljanja kompleksnim strukturama koje pruža objektna tehnologija trebaju omoguciti realizaciju modela procesa konstruiranja koji je realniji i transparentniji od modela realiziranih drugim metodama programiranja. Pri tome se pod realnijim modelom podrazumijeva da su semanticke razlike u odnosu na stvarni svijet što manje. Transparentniji model znaci da je konstruktoru prirodniji, razumljiviji i jednostavniji za ucenje i uporabu. Pored toga prednosti primjene objektne tehnologije ogledaju se u sustavima koji su fleksibilniji za primjenu, dogradnju i održavanje.
Osnovni cilj ovog rada je koncipiranje i kreiranje strukture racunalnog modela procesa konstruiranja korištenjem objektno orijentirane metodologije modeliranja. Za tako odredeni cilj postavlja se hipoteza u tri medusobno spregnute tocke:

1) *Mrežna topologija procesa konstruiranja modelira se objektno orijentiranim metodama.*

   Pretpostavlja se da je objektnim metodama moguce realizirati, zapisati i dogradivati kompleksnu višerazinsku mrežu relacija izmedu skupova i struktura podataka na efikasniji nacin nego drugim raspoloživim metodama. Ova hipoteza verificirati ce se realizacijom modela mreže relacija u objektnoj bazi podataka.

2) *Entiteti prostora fenomenološkog modela procesa konstruiranja preslikavaju se u prostor objektnog modela.*

   Nužan preduvjet u razradi prve tocke hipoteze je teoretsko uopcivanje i definiranje svih pojava i pojmova koji odreduju konstrukcijski proces kao entiteta fenomenološkog modela. Pri verifikaciji hipoteze nastojati ce se ostvariti preslikavanje "jedan za jedan", tj. da svakom entitetu fenomenološkog modela odgovara jedna klasa objektnog modela.

3) *Temeljem osnovnih klasa objektnog modela grade se složeni objekti koji modeliraju prikaz procesa i dinamiku izvodenja procesa konstruiranja.*

Pretpostavlja se da je moguce koncipirati model procesa konstruiranja "bottom-up" pristupom, tako da se složeniji elementi prikaza procesa grade kombiniranjem, referenciranjem i agregacijom osnovnih elemenata strukture modela. Takva koncepcija treba osigurati i vecu fleksibilnost modela pri uvodenju u eksploataciju.

Pri razradi i dokazivanju prve tocke hipoteze, nužno je prvo provjeriti i dokazati preduvjete postavljene u druge dvije tocke.
Primarni cilj razvoja "teoretskih" modela procesa konstruiranja je produbiti razumijevanje i spoznaje o procesu konstruiranja, te razviti egzaktne formalizacije pri opisivanju procesa.
Spoznaje "teoretskih" modela u ovom istraživanju nastojati ce se preslikati i primjeniti pri koncipiranju "racunalnog modela". Pri tome ce se u svakoj situaciji koncepcija modela prilagodavati mogucnostima racunalne implementacije. Koncipirani "racunalni model" predložiti ce se kao osnova kreiranja racunalne podrške tijeku odvijanja procesa konstruiranja. Tako formulirano istraživanje implicira zadovoljenje skupa zahtjeva, koji se mogu promatrati kao parcijalni ciljevi:

1. Racunalni model procesa konstruiranja treba biti opcenito primjenjiv, tj. ne smije ovisiti o fazi procesa konstruiranja niti o vrsti konstrukcijskog zadataka. Proces konstruiranja promatran kao proces obrade i generiranja informacija na isti nacin se može tretirati u svim fazama i za sve vrste zadataka. Stoga ce se struktura i elementi modela definirati prema kriterijima procesa obrade i generiranja informacija.
2. Entiteti i struktura modela trebaju poslužiti kao platforma za integraciju raznorodnih programskih alata koji se koriste u procesu konstruiranja.
3. Sustav treba biti koncipiran tako da služi kao "pomocnik" i "savjetnik" konstruktora, za razliku od težnji prema "automatskom konstruiranju"
4. Treba postici što veci moguci stupanj fleksibilnosti modela, odnosno primjenjivosti za razlicite uvjete i okoline izvodenja procesa konstruiranja. Jezgru modela i organizaciju strukture modela treba koncipirati tako da se cijeli sustav modela može prilagodavati i dogradivati prema specificnim potrebama i postojecim modelima i pravilima izvodenja procesa konstruiranja u odredenoj konkretnoj okolini (konstrukcijskom uredu).

1.3 Znanstveni doprinos

Razvijeni racunalni model procesa konstruiranja treba biti osnova integracije postojecih CAD alata i istovremeno služiti kao racunalna podrška odvijanju procesa konstruiranja na višoj razini apstrakcije zadatka nego sa sadašnjim programskim alatima.
Integrirani programski sustav koji bi uz standardne mogucnosti CAD (CAE) paketa sadržavao i model procesa konstruiranja trebao bi omoguciti konstruktoru da u kracem vremenu i s manje rutinskih aktivnosti može ponoviti procese analize i sinteze skupova informacija o proizvodu uz potrebne korekcije, a isto tako i da paralelno obraduje više varijanti prije odabira konacne. Pri tome model procesa konstruiranja treba sadržavati podršku planiranju i kontroli tijeka izvodenja procesa. Ocekuje se da se objektnim tehnologijama može realizirati implementacija tzv. "dinamickog planiranja". Realizacija zapisa mrežne topologije procesa konstruiranja trebala bi racunalni model više približiti realnosti. Znacajne mogucnosti unapredenja procesa konstruiranja mogu se ostvariti u uvjetima timskog rada na jednoj konstrukciji (što je danas vjerojatno prevladavajuca situacija). Racunalni model procesa konstruiranja treba omoguciti i unapredenje organizacije procesa i ubrzati komunikaciju izmedu clanova tima. Takoder je važno

koncipirati i entitete koji modeliraju procese "istovremenog inženjerstva" (concurrent engineering).
Osnovna struktura objektnog modela trebala bi u daljnjim istraživanjima poslužiti kao skup prijedloga i polaznih tocaka, te poticaj za šire usaglašavanje koncepcije i definicija entiteta racunalnog modela procesa konstruiranja, koji bi težio k standardizaciji. Na takvoj platformi mogli bi se u slijedecim fazama istraživanja efikasnije nadogradivati složeni programski alati koji ce sustav više približiti ideji “inteligentnog”, odnosno zamisli "modela radne okoline", tj. "radnog stola" ("designer's workbench-a").

1.4 Polazišta

Kako je Znanost o konstruiranju multidisciplinarna, kao polazišta za istraživanja definirana zadatkom ovog rada potrebno je prije svega izvršiti kompilaciju literature iz nekoliko podrucja. Problem definiran zadatkom rada obuhvaca podrucje Znanosti o konstruiranju, te odredena podrucja racunalskih znanosti, umjetne inteligencije i teorije grafova. Kao dio znanosti o konstruiranju nedostaje dobro utemeljena i detaljno razradena teorija CAD-a koja bi trebala biti "agregat" prije navedenih znanosti, cime bi postala polazište za daljnji razvoj CAD sustava.
Ova disertacija dio je ukupnih istraživanja na projektu razvoja modela CAD sustava cije karakteristike se trebaju približiti ideji “inteligentnog” CAD sustava. Rad se jednim dijelom nastavlja na istraživanja prikazana u [6] i [7]. U nedostatku boljeg ili prikladnijeg termina za karakterizaciju CAD ili CAE sustava koji bi sadržavao okruženje za modeliranje procesa konstruiranja, koristiti ce se pridjev “inteligentni”. U literaturi se taj pridjev cesto koristi [8], [9], ali inteligentni sustav prije je cilj (limes) kojem treba težiti nego prakticno ostvariva mogucnost. U novijoj literaturi koristi se i naziv "designer's workbench" [10], [11]. Detaljniji pregled literature dan je u drugom poglavlju.

Treba naglasiti da do sada (za širu prakticnu primjenu) nije razvijen CAD sustav opce namjene s implementiranim modelom procesa konstruiranja.
U znanosti o konstruiranju ne postoji detaljno razradena i usaglašena metoda prikaza i zapisa tijeka odvijanja konstrukcijskog procesa. Takvo stanje ne treba iznenaditi jer do sada je razvijeno (za razlicite domene i namjene) nekoliko desetaka metoda prikaza procesa, i tek se odnedavno pokušava krenuti prema standardizaciji u tom podrucju [12].
Da bi se moglo prici konacnoj sintezi modela procesa konstruiranja potrebno je provesti pocetne analize i istraživanja u više razlicitih podrucja. Pri tome se mogu razluciti sljedece polazišne tocke:

- analiza procesa konstruiranja, nacina razmišljanja i organizacije rada konstruktora
- postojeci opci modeli procesa konstruiranja i opce teorije konstruiranja
- analiza postojecih informacijskih modela i njihove primjene u konstruiranju
- analiza strukture inženjerskih podataka i dokumenata (kao nosilaca informacija)
- opcenite metode prikaza procesa, teorija grafova
- metode planiranja u umjetnoj inteligenciji
- opci principi i mehanizmi modeliranja
- principi objektno orijentirane metodologije modeliranja i razvoja programskih sustava

1.5 Metodologija istraživanja

U uvodnom dijelu rada analizirati ce se znacajke procesa konstruiranja i biti ce sagledano sadašnje stanje istraživanja u teoretskom kao i u racunalnom modeliranju procesa konstruiranja. Iz razmatranja znacajki procesa konstruiranja trebaju proizaci zahtjevi na koncepciju modela procesa konstruiranja, ali i ogranicenja - ne može se ocekivati da je moguce u potpunosti modelirati proces konstruiranja (ne postoji usaglašen model niti u Znanosti o konstruiranju).

Glavni dio istraživanja sacinjavati ce slijedeca razmatranja:

- promatranje modela procesa konstruiranja kao integriranog dijela cjelokupne informaticke podrške razvoju proizvoda, a pri tome se konstruiranje tretira kao proces obrade i generiranja informacija
- apstrahiranje[1] (teoretsko izlucivanje) svih pojava, stvari, dogadaja, informacija kao entiteta procesa konstruiranja
- preslikavanje entiteta u objekte, definiranje atributa, operacija i relacija objekata

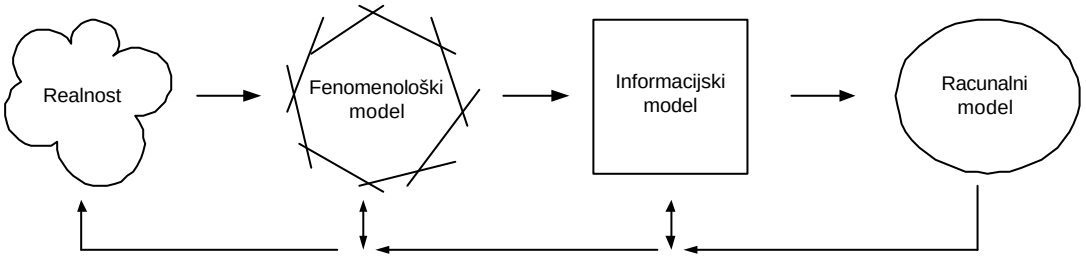
Analizirati ce se metode i principi objektnog modeliranja, na temelju kojih ce se predložiti koncepcija osnovne strukture sustava.

Pri definiranju entiteta modela, odnosno objekata krenuti ce se od najjednostavnijih, koji ce poslužiti kao gradbeni elementi za složene (kompozitne) objekte koji ce modelirati kompleksne pojmove i relacije. Dakle osnovna metoda gradnje sustava biti ce "bottom-up", dok ce se po potrebi kombinirati i "top-down" i "bottom-up" pristup. Principi gradnje sustava biti ce vodeni idejom "otvorene kutije s alatima" (open toolbox).

Korištenjem kombinacija i agregacija osnovnih entiteta, predložiti ce se nacini implementacije u model nekih od metoda prikaza i reorganizacije procesa konstruiranja.

Zakljucna razmatranja usporediti ce razradeni model sa drugim poznatim modelima, te analizirati mogucnosti primjene i predložiti pravce daljnjih istraživanja.

Sa stanovišta modeliranja, metodologija istraživanja može se prikazati kao niz preslikavanja prema slici 1 (autori Andreasen i Duffy [13]).



Slika 1: Preslikavanja od realnosti do racunalnog modela

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. Racunalni model temelji se na programskom jeziku, odn. odabranom programskom okruženju realizacije. Preslikavanja s lijeva u desno prikazuju proces istraživanja u kojem se fenomenološki model postepeno formalizira sredstvima informacijskih i racunalnih modela. Putanja s desna u lijevo oznacava proces verifikacije u kojem se model usporeduje s realnosti.

[1] Ovdje, i dalje u radu se rijec "apstrahirati" upotrebljava u znacenju izlucivanja, idealizacije, a ne u znacenju iskljucivanja, tj. "ne uzimanja u obzir"

2. Pregled stanja i smjerova istraživanja

Iako je Znanost o konstruiranju relativno mlada disciplina, detaljniji pregled i klasifikacija svih smjerova istraživanja, zahtjevna je i vrlo opširna zadaca koja bi odvela previše u širinu. Stoga ce biti istaknute samo one teme i smjerovi koji su relevantni za temu rada.
Veliki znacaj konstruiranja za profitabilnost proizvodnje dovodi do stalnog rasta interesa za ovu disciplinu, tako da danas vec postoje i multidisciplinarni timovi istraživaca.

Dosadašnji tijek razvoja Znanosti o konstruiranju karakterizira:

- velik broj razvijenih parcijalnih modela i teorija,
- neuskladenost metodologije,
- nekoliko razlicitih "škola konstruiranja" (Njemacka i Engleska, USA, Japan),
- posljednjih desetak godina vrlo brz razvoj racunalnih tehnologija bitno utice i na razvoj i otvaranje novih smjerova istraživanja.

Cini se kao da razvoj komercijalnih CAD i CAE sustava ne ide paralelno sa razvojem Znanosti o konstruiranju, odnosno do implementacije novih metoda i spoznaja u komercijalne CAD sustave ili ne dolazi ili ona prilicno kasni. Nekoliko je mogucih razloga za takvo stanje:

- tvrtke koje proizvode vrhunske CAD sustave nastoje da im podrucje primjene bude što šire, a novorazvijene metode cesto su samo parcijalni modeli za uže domene
- primjena novih spoznaja i metoda iz Znanosti o konstruiranju i inace se teško prihvaca u praksi [14], [15]
- nove metode trebaju proci vrijeme "dokazivanja" i prilagodbe konstruktora koji su navikli na ustaljeni nacin rada i pružaju otpor promjenama

Pregled stanja istraživanja stoga je u nastavku (sa gledišta procesa konstruiranja) podijeljen na dva dijela - "teoretski" - Znanost o konstruiranju i "prakticni" - razvoj CAD i CAE sustava. Dakako, razvoj CAD sustava temelji se na znacajnim znanstvenim doprinosima iz podrucja geometrijskog modeliranja, ali najveci nedostatak je upravo to da se radi najvecim dijelom samo o geometrijskom modeliranju.

2.1 Stanje i smjerovi istraživanja u Znanosti o konstruiranju

Može se reci da u podrucju znanosti o konstruiranju ne postoje zajednicki usaglašeni stavovi medu istraživacima o prikladnim metodologijama i prioritetnim smjerovima istraživanja. Takvo stanje uzrokuje odredenu kaoticnost, ali je i poticajno za daljnja istraživanja. Takoder se može uociti da nedostaje jedinstvena teorija CAD-a - ne postoji potpuna reprezentacija proizvoda, niti usaglašen model procesa konstruiranja proizvoda.

Jedan od mogucih razloga za takvo stanje je u cinjenici da je znanost o konstruiranju relativno mlada znanstvena disciplina. S druge strane mora se napomenuti da je proces

konstruiranja izuzetno kompleksan proces, te je stoga neke njegove znacajke teško razumjeti do te mjere da bi se mogle teoretski formalizirati.
Nekoliko autora nastojalo je sistematizirati dostignuca i dati pregled trenutnog stanja. Kao znacajnije ovdje ce se izdvojiti Hubku i Edera (1996) [16], te Finger i Dixona (1989) [17], [18]. Koliko je poznato autoru ovog rada, noviji pregledi stanja takvih opsega i takve razine detalja klasifikacije nisu napravljeni. Noviji pristupi sistematizaciji i vizije razvoja u buducnosti mogu se naci u [19] i [20]. VDI 2221 [21] daje detaljan pregled i klasifikaciju metoda konstruiranja, uz naznake njihove uporabe u pojedinim fazama procesa konstruiranja.

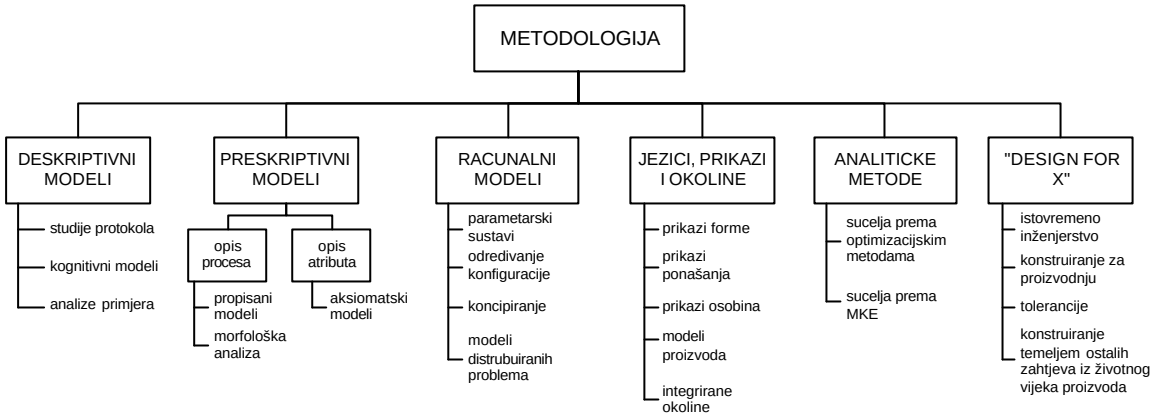
Hubka i Eder [16] daju detaljnu sistematizaciju Znanosti o konstruiranju, i to sa više razlicitih aspekata. Takoder daju i kronološki prikaz razvoja Znanosti o konstruiranju. Ovdje cemo izdvojiti dio njihovih razmatranja o trenutnom stanju znanja o konstruiranju.

> Velika kolicina znanja je akumulirana, ali vecinom kao "otoci znanja", jer nije bilo dovoljno sinteza.
> Raspoloživo znanje unutar znanja o konstruiranju nije jednoliko (uravnoteženo) pripremljeno, odnosno pojedina podrucja nisu jednakovrijedno tretirana.
> Pri razvoju metodologije konstruiranja neki problemi su se iskristalizirali ili u parcijalne zadatke ili u principe metodologije, npr.:
> - zadatak razvoja jasnog prikaza postupaka konstruiranja
> - zadatak najfinijeg moguceg strukturiranja procesa, sa jasnom separacijom individualnih aktivnosti, poglavito onih sa posebnim znacajkama (npr. traženje rješenja problema)
> - princip razrade što je moguce više alternativa i njihove optimizacije u ranim fazama procesa konstruiranja
> - zadatak prilagodbe opcih postupaka razlicitim situacijama:
>   - za timski rad
>   - za primjenu racunala
>
> Kvaliteta znanja nije jednolika i varira od iskustvenog znanja do preciznih izreka
> Put prema inženjerskoj praksi, odnosno široj primjeni još nije pronaden.

Dok su Hubka i Eder orijentirani na vrlo široku i detaljnu sistematizaciju i organizaciju Znanosti o konstruiranju, Finger i Dixon daju samo pregled stanja i klasifikaciju pravaca istraživanja u radovima [17] i [18].
Prema [17], podrucja istraživanja u znanosti o konstruiranju mogu se sistematizirati na slijedeci nacin (slika 2):

1) Deskriptivni modeli procesa konstruiranja
2) Preskriptivni modeli procesa konstruiranja
3) Racunalni modeli procesa konstruiranja
4) Jezici, prikazi i okruženja za podršku konstruiranju
5) Analiticke metode kao podrška donošenju odluka
6) Konstruiranje temeljem zahtjeva iz životnog vijeka proizvoda

Navedene kategorije nisu medusobno iskljucive, odnosno neka istraživanja mogu obuhvacati dva ili više podrucja.



Slika 2: Taksonomija podrucja istraživanja u znanosti o konstruiranju, prema [17]

Tema ovog rada obuhvaca istraživanja iz podrucja navedenih pod r. br. 1, 3 i 4 pa cemo izdvojiti neke od otvorenih tema istraživanja u tim podrucjima prema [18]. Premda je prikaz dan 1989. godine, vecina tema aktualna je i danas.

Deskriptivni modeli:

- Potrebno je dalje razvijati spoznajne (kognitivne) modele strategija konstruiranja da bi se bolje razumjelo kako konstruktori rade
- Još uvijek se premalo zna o dekompoziciji zadataka koji se rješavaju timskim radom, makar je to situacija pri konstruiranju vecine proizvoda.

Racunalni modeli procesa konstruiranja:

- Modeli i metode parametarskog konstruiranja su visoko zavisni o podrucju. Potrebno je istražiti mogucnosti unifikacije parametara konstrukcije, što ukljucuje i numericke i nenumericke podatke.
- Istraživanje mogucnosti promjene konfiguracija sklopova bez redefiniranja pocetnih projektnih parametriziranih informacija
- Definiranje funkcija strategijskih alata za rješavanje distribuirane problemske eksplozije pri razvoju ekspertnih sustava za podršku konstruiranju.
- Uloga fizikalnih principa u povezivanju forme i funkcije još nije u potpunosti razmotrena (shvacena).

Jezici, prikazi i okruženja za podršku konstruiranju:

- Podrucje istraživanja zajednicko za sve konstrukcijske probleme je prikaz mehanicke konstrukcije. Neke od tema istraživanja su :
  - prikaz nedovršene konstrukcije
  - prikaz procesa razvoja konstrukcije (ukljucivši kontrolu konfiguracije i kontrolu razlicitih verzija)
  - prikaz atributa konstrukcije koji nisu geometrijske prirode
  - veze i zavisnosti izmedu prikaza razlicitih atributa konstrukcije
  - implementacija svojstava i drugih apstraktnih pojmova u prikaz konstrukcije
- Potrebno je istražiti mogucnosti primjene formalnih gramatika i jezika za prikaz konstrukcije.
- Važno podrucje kojem nije posveceno dosta pažnje je kreiranje okruženja za konstruiranje (design environments) koje integrira raspoložive programske alate u konzistentan sistem za podršku konstruktoru.

Analiticke metode za podršku donošenju odluka:

- Kreirati CAD sustave koji podržavaju faze koncipiranja omogucujuci konstruktoru da kreira, modificira i analizira na višestrukim razinama apstrakcije i sa razlicitih gledišta.

2.2 Situacija u razvoju CAD i CAE sustava

Nagli razvoj racunalne opreme omogucio je i razvoj vrlo zahtjevnih i kompleksnih programskih paketa. CAE (Computer Aided Engineering) jedno je od podrucja primjene racunala s vrlo velikim zahtjevima na sve resurse i performanse racunala. Integracija CAD, CAM (i ostalih popularnih kratica koje pocinju sa "CA") u cjelovit i efikasan CIM sustav izuzetno je kompleksan proces i na teoretskom nivou, dok implementacija u praksi otvara još i niz novih problema. Modeliranje strukture i problemi prakticne realizacije takovog integriranog sustava i/ili njegovih podsustava tema je velikog dijela današnjih istraživanja u podrucju primjene racunala u cjelokupnom procesu proizvodnje. Pri tome postoji vrlo velik broj razlicitih pristupa u kojima se vecinom zbog opsežnosti problema interes sužava na odreden podsustav onoga što cini "ideju" CIM sustava. Osnovna svrha svih tih istraživanja je daljnje povecanje produktivnosti u svim aktivnostima koje prate proces proizvodnje. Proces konstruiranja ima najveci utjecaj na troškove i rokove u procesu proizvodnje, [22], [23], dakle i na kompletno poslovanje i profit pa je razumljivo da je vecina istraživanja posvecena upravo podrucju CAD-a. Vrlo cesto upravo je konstrukcijski ured "usko grlo" u procesu proizvodnje.

Na tržištu danas postoji velik broj CAx programskih paketa s velikim rasponom ponude mogucnosti i cijena. Vecina tih paketa je opce namjene, a postoje i specijalizirani za npr. domene elektronike, strojarstva ili arhitekture. Najveci broj komercijalnih programskih sustava može se svrstati u CAD sustave (2D ili 3D). Takoder se na tržištu nudi i velik broj CAM paketa, kao i paketa za numericke analize konstrukcija (najcešce metoda konacnih elemenata - FEM). Samo najskuplji programski sustavi nekoliko vodecih proizvodaca sadrže CAD, CAM, FEM, EDM/PDM i ostale komponente potrebne u inženjerskom radu. Medutim i tada su to još uvijek samo zasebne programske cjeline, a nema modela procesa konstruiranja u kojem bi se integrirale te zasebne funkcije.

Korištenje takvog skupa programskih paketa ne podržava u cjelini proces konstruiranja, a za ovladavanje svim segmentima sustava potrebno je utrošiti dosta rada i vremena. Gotovo se može reci da se jedna osoba nije u stanju jednako kvalitetno (produktivno) koristiti svim elementima sustava, nego i tu dolazi do specijalizacije. Dodatne mogucnosti unapredenja produktivnosti pružaju tzv. "makro jezici" kojima se može automatizirati i povezati niz radnji u korištenju sustava, ali to ne može kvalitativno doprinijeti rješavanju osnovnih nedostataka postojecih CAD sustava u koje možemo ubrojiti:

- manipuliranje pretežno samo geometrijskim (numerickim) informacijama
- nedostatak modela procesa konstruiranja, odnosno podrške prirodnom toku promišljanja
- nedostaju programski alati i sucelja za povezivanje pojedinih elemenata sustava
- vrlo je teško realizirati paralelno korištenje CAD sustava i drugih vrsta aplikacija, npr. aktiviranjem više procesa istovremeno - nedostaju sucelja, modeli nacina povezivanja i prijenosa podataka te kontrole redoslijeda izvodenja

Znacajan dio istraživanja mogucnosti unapredenja CAD (CAE sustava) temelji se na ideji da se primjenom metoda umjetne inteligencije omoguci modeliranje podrške prirodnom toku promišljanja i postupaka obrade informacija u tijeku procesa konstruiranja. To ukljucuje modeliranje prikaza tijeka procesa konstruiranja na nacin koji omogucuje obradu i numerickih i simbolickih informacija na razlicitim razinama apstrakcije. Cilj je istraživanja takav model što više približiti znacajkama procesa konstruiranja.

S informacijskog stanovišta možemo proces konstruiranja promatrati i kao iterativni proces izmedu obrade podataka i donošenja odluka.
Dobar dio razlicitih procesa obrade informacija nije moguce ili nije pogodno podržati CAD sustavom opce namjene - npr. baza podataka sa stanjem skladišta ili npr. programi za numericke proracune parametara dijelova ili sklopova. Za takve svrhe uglavnom treba razvijati aplikacije cija je domena usko vezana uz specificne zahtjeve okruženja u kojima se koriste.
Podrška donošenju odluka može se realizirati implementiranjem metoda umjetne inteligencije. Pocetke razvoja umjetne inteligencije karakterizira veliki entuzijazam i ocekivanja. Vremenom se došlo do spoznaja da razvijene metode imaju ogranicenja i da efikasno funkcioniraju kod primjene na dobro definirane probleme unutar uske domene. Po sadašnjim spoznajama ne može se ocekivati da je moguce razviti ekspertni sustav koji bi bio u stanju "sam" konstruirati, niti da je moguce formalizirati sveukupno znanje struke i postupke koje konstruktor koristi u radu. Primjena ekspertnih sustava, odnosno metoda umjetne inteligencije u CAD sustavima može riješiti neke njihove nedostatke - prvenstveno nedostatak obrade simbolickih i topoloških informacija. U obliku baze znanja mogu se pohraniti skupovi pravila i postupaka koji vrijede za odredenu proizvodnu okolinu, odnosno vrstu proizvoda. Medutim primjena metoda umjetne inteligencije ne rješava nedostatak modela procesa konstruiranja, odnosno nedostatak programske podrške za prikaz, planiranje i kontrolu tijeka odvijanja procesa konstruiranja.

2.2.1 Istraživanja unapredenja CAD i CAE sustava

Istraživanja u podrucju znanosti o inženjerskom konstruiranju dala su skup apstraktnih i opcenitih teorija sa svrhom organiziranja procesa i povecavanja efikasnosti. Jedan dio tih napora usmjeren je na proucavanje nacina prikaza procesa konstruiranja, a mogu se grubo svrstati u dvije kategorije:

- istraživanja opcih i apstraktnih prikaza procesa konstruiranja, uglavnom temeljena na tehnikama scenarija
- istraživanja modela procesa konstruiranja za odredene klase konstrukcijskih zadataka, temeljena na grafickim prikazima

Tipican proces razvoja proizvoda realizira se u malim koracima doradivanja postojecih proizvoda za razliku od razvoja potpuno novih [24]. Informacije (podaci) o proizvodu koji se konstruira spremaju se u baze podataka. Spremljeni skupovi podataka su pretežno geometrijski i strukturalni, te podaci o materijalu i tehnologiji izrade. Vecinom se ne spremaju informacije o tome kako i zašto se došlo do odredenog rješenja. Taj vrlo važan segment znanja u procesu konstruiranja ostaje skriven i nezapisan, dakle bez mogucnosti ponovnog korištenja i nije na raspolaganju svima u konstrukcijskom uredu. To su znanja o procesu konstruiranja i proizvodnje specificna za radnu okolinu u kojoj se proces odvija. Takva znanja cesto su i specificna za pojedinu osobu (konstruktora) - npr. individualne metode rada i steceno iskustvo.

Rezultati istraživanja, kao i iskustva iz industrije naglašavaju potrebu integracije informacijskih tokova u procesu proizvodnje, ciji je dio proces konstruiranja. Poduzeca koja su krenula tim smjerom, znatno su unaprijedila produktivnost i smanjila troškove. Ako promatramo informacijske tokove cijelog procesa proizvodnje, sigurno je da je proces konstruiranja najkompleksniji njegov dio. U procesu konstruiranja obraduje se i stvara najveci dio informacija o proizvodu, i to s najvecim udjelom heuristike. Vec je prije napomenuto i da proces konstruiranja ima najveci utjecaj na troškove proizvodnje.
Sve nabrojene cinjenice poticaj su velikog broja istraživanja procesa konstruiranja, pogotovo metoda i modela racunalom podržanog konstruiranja. Jedan dio istraživanja u podrucju CAD-a, bavi se integracijom CAD alata i implementacijom CAD alata u sve faze procesa konstruiranja. Razvijen je znacajan broj razlicitih pristupa i pravaca istraživanja. Za ilustraciju sadašnjih trendova istraživanja dan je kratak pregled znacajnijih radova, uz grubu klasifikaciju.

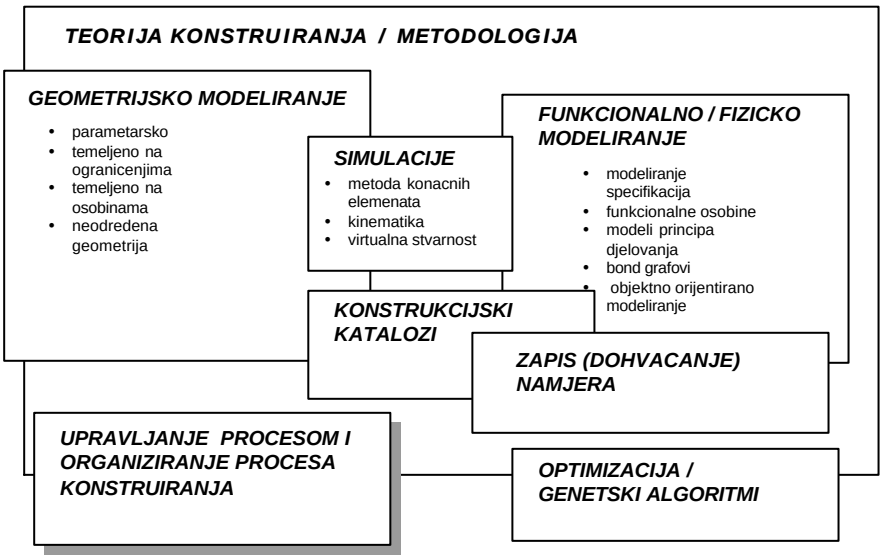
- Teoretske podloge za "inteligentne CAD sustave": Akman, Hagen i Tomiyama [25] predložili su "jezik za opis konstruiranja" sagraden na principima logickog i objektno orijentiranog programiranja. Jezik se temelji na opcoj teoriji konstruiranja po Yoshikawi, možda najboljoj do sada prezentiranoj teoriji konstruiranja.
  Chandrasekaran [26] je razvio strukturu zadataka u procesu rješavanja konstrukcijskih problema, proucavajuci razlicite metode i znanje i klasificirajuci podzadatke neovisno o pridruženom znanju.
  Colton i Dascanio [27] koncipirali su i implementirali integriranu, "inteligentnu" okolinu koja daje konstruktoru mogucnost izbora iz kataloga i konstruiranja uobicajenih dijelova.
  Vecina takvih istraživackih projekata bavi se "prijateljskim" korisnickim suceljima, integriranim sa modulima za kontrolu informacija. Ekspertni sustavi razvijeni za te svrhe, obicno su efikasni samo u ogranicenim domenama.
- Neki od pristupi u podrucju modeliranja modela podataka proizvoda ("product data models"): semantika modela [28], integracija više racunalnih procesa [29], minimizacija troškova razvoja proizvoda [30], procesiranje baza podataka [31].
- Cinjenica da konvencionalni CAD sustavi nisu koncipirani da bi asistirali konstruktoru u rješavanju kompleksnih problema potice istraživanja integriranih programskih okolina [32]. Gauseimer i Vajna bave se aspektima kvalitete i efikasnosti razvoja proizvoda [33], [30]. Modeli "radnih okolina" ("designers workbench") razvijaju se sa više razlicitih pristupa, i tu tek predstoji usaglašavanje ciljeva i metoda [34], [35].
- Razvoj modela konstruiranja temeljen na paradigmi umjetne inteligencije dao je snažan poticaj istraživanjima [36]. Blount i Clarke tretiraju konstruiranje kao aktivnost rješavanja problema koja se može automatizirati [37]. Mostow [38] istražuje tehniku "kompilacije znanja" kao transformaciju eksplicitno prikazanog znanja iz domene u efikasni algoritam za izvodenje odredenih zadataka u domeni. Uloge ekspertnih sustava i sustava za upravljanje bazama podataka u okolini "konstruiranja za izradu" razmotrene su u radu [39]. Metode rješavanja problema analogijom, primjenjene na inženjersko konstruiranje, cesto se upotrebljavaju u razvoju "inteligentnih agenata" [40]. Istraživanja koja kombiniraju umjetnu inteligenciju i metode optimizacije pokazala su znacajan utjecaj na formalizaciju i rješenja racunalnih metoda u inženjerskom konstruiranju [41]. Istraživanja primjene ekspertnih sustava sugeriraju da standardni pristupi nisu pogodni za konstruiranje i planiranje. Stoga u ovom podrucju treba razviti nove pristupe [42]. Ullman i D'Ambrosio predložili su taksonomiju racunalne podrške konstruiranju [43]. Opcenitiji pristupi bave se razvojem "okvira" znanja (knowledge frameworks). Forde i

koautori uvode strategiju koncipiranja objektno orijentiranog "okvira" [44]. Smithers [45] predlaže teoriju konstruiranja u obliku "razina znanja". Svoju teoriju upotrebljava kao prakticnu alternativu spoznajnoj teoriji konstruiranja.

- Racunalni modeli ovisni o domeni, dovedeni su gotovo do savršenstva u suženim prostorima interesa. Takvi slucajevi upravo naglašavaju razliku izmedu modeliranja procesa konstruiranja [46] i modeliranja odredenog zadatka. Razmatrajuci konstruiranje kao proces, ta istraživanja usredotocuju se na pronalaženje odgovarajuce metode za rješavanje problema. Racunalni modeli zadataka konstruiranja opcenito su specificni za dobro definiranu klasu problema kao u [47], [48] i u mnogim drugim radovima.
- Opcenitiji pristupi modeliranju procesa konstruiranja pokušaji su koji podržavaju izvodenje i oznacavanje alternativnih koncepata upotrebljavajuci negeometrijske entitete za podršku kompleksnim konstrukcijskim projektima. Spremljeni koncepti omogucuju implicitno zapisivanje konstrukcijskog znanja (design rationale) [49], [50]. Raste interes za modele procesa konstruiranja zasnovane na teoriji grafova. Eppinger modelira iteraciju u procesu konstruiranja pomocu grafova toka signala [51]. Schmidt [52] upotrebljava algoritme temeljene na gramatici grafova pri generiranju rješenja blizu optimalnih. Zadnja dva spomenuta modela mogu se kategorizirati i kao optimizacijski modeli. Planiranje i tehnike scenarija [53] predstavljaju daljnje iskorištenje paradigme umjetne inteligencije. Planovi konstruiranja i potreba za automatiziranim sustavima objašnjavanja, kao i njihova interakcija s konstruktorom, predmet su kontinuiranih istraživanja [54].

Pristupi koji se bave planiranjem procesa konstruiranja proizvoda relativno su rijetki u odnosu na istraživanja u koncipiranju procesa planiranja proizvodnje.

Weber i Werner predlažu pregled i sistematizaciju podrucja istraživanja primjene racunala u konstruiranju prema shemi na slici 3.



Slika 3: Podrucja istraživanja primjene racunala u znanosti o konstruiranju prema [55]

Istraživanja u ovom radu usmjerena su modeliranje upravljanja procesom i na unapredenje organizacije toka i razmjene informacija u procesu konstruiranja. Pri tome ce se koristiti spoznaje i metodologije teorije konstruiranja.

3. Analiza procesa konstruiranja

Prije svega trebamo biti svjesni da se modeliranjem racunalne podrške planiranju i izvodenju konstrukcijskog procesa može ostvariti samo približan model realnog procesa konstruiranja. Svrha ove analize je iznalaženje smjernica za koncipiranje modela, te odredivanje mogucnosti i nacina približenja znacajkama realnog procesa.
Kako je vec navedeno, približenje racunalnog modela realnosti, ovisi prije svega o spoznajama i razumijevanju procesa konstruiranja, a zatim i o mogucnostima i metodologiji primjene današnjih informatickih tehnologija.
Kao prvi korak analize, i najviše, razumijevanja procesa konstruiranja potrebno je razmotriti znacajke procesa konstruiranja.
Konstruktor u tijeku procesa konstruiranja svoje aktivnosti usmjerava ka cilju - rješenju postavljenog zadatka, pa analiza procesa konstruiranja s aspekta rješavanja zadatka daje smjernice za modeliranje prikaza tijeka procesa konstruiranja.

3.1 Znacajke procesa konstruiranja

Kao jedno od polazišta za razmatranje, navesti ce se znacajke procesa konstruiranja i konstruktorskog rada prema [22].

1) Konstrukcijski proces je u prvom redu proces prerade i generiranja informacija, gdje se na temelju ulaznih zahtjeva generira skup informacija koje opisuju proizvod.
2) Konstrukcijski proces je sinteza relativno dobro poznatih elemenata u jednu jedinstvenu, otprije nepoznatu cjelinu sa zahtjevanim odredenim svojstvima. Ta sinteza iziskuje kreativan, stvaralacki rad. Iz toga proizlazi važna znacajka procesa konstruiranja - covjek mora kontrolirati proces, odnosno imati pretežan udio u obavljanju potrebnih radnji.
3) S filozofskog gledišta proces konstruiranja je takoder i spoznajni proces: sustav na pocetku nepoznat - spoznaje se, odnosno postaje poznat. Na temelju toga može se reci da je spoznajna teorija takoder jedan izvor opcih zakonitosti za proces konstruiranja.
4) Konstruiranje se može promatrati i kao proces ucenja.
5) Svaki konstrukcijski zadatak može se riješiti na mnogo razlicitih nacina, odnosno može rezultirati sa razlicitim strojnim sustavima ili sklopovima. Ta karakteristicna mnogostrukost mogucih rješenja uvjetovana je kolicinom svojstava proizvoda koje se trebaju odrediti u postupku konstruiranja.
6) Svaki proces konstruiranja može se razložiti u manje cjeline (faze, dijelove procesa, etape, operacije) koje cine strukturu procesa.
7) Velika kompleksnost medusobno proturjecnih zahtjeva dovodi do potrebe za višestrukim ponavljanjem odredenih faza nakon pocetnog apstrahiranja i postavljanja

pretpostavki - dok se ne odrede potrebne vrijednosti. Iterativni postupak je jedna od tipicnih znacajki konstruiranja.

8) Do sada pretežno samostalna djelatnost (u okviru zadatka), sve više se pretvara u timski rad u kojem se koriste prednosti veceg informacijskog kapaciteta i medusobne razmjene ideja i postupaka.

9) Proces konstruiranja je vrlo zahtjevan kreativni rad, ali ne smije se promatrati kao umjetnost, nego kao znanstveni rad. Odredeni misaoni postupci (intuicija, nastajanje ideje) koji se ne mogu racionalno objasniti imaju obilježja umjetnicke kreativnosti. Ti postupci ne mogu se formalizirati u svrhu stvaranja cjelovitog teoretskog prikaza konstrukcijskog procesa.

Sa stanovišta teorije proizvoda, [56] mogu se navesti slijedece znacajke procesa konstruiranja:

- nagli porast potreba uvjetovan tržišnim zakonitostima
- period razvoja proizvoda je sve kraci
- vijek trajanja proizvoda je sve kraci
- javlja se pojam kriticne brzine konstruiranja
- raste kolicina proizvoda u serijskoj i masovnoj proizvodnji
- zahtjevi za kvalitetom takoder rastu
- troškovi proizvodnje i poslovanja se moraju svoditi na minimum
- najveci utjecaj na strukturu troškova proizvodnje ima konstrukcijsko rješenje proizvoda
- produktivnost tehnologije izrade proizvoda raste daleko brže od produktivnosti konstruiranja

Iz navedenih znacajki može se uociti mnogostrukost upliva na proces konstruiranja, kompleksnost procesa, a i širina potrebnih znanja. Detaljnija razmatranja upliva okoliša na proizvodni sustav s implikacijama na CIM, odnosno na proces konstruiranja i CAD mogu se naci u [57].
Kompleksnost procesa konstruiranja i mnogostrukost upliva okoline cine zadatak modeliranja procesa konstruiranja vrlo zahtjevnim. To se ocituje i u velikom broju razlicitih stavova o konstruiranju i u nizu razlicitih pristupa modeliranju procesa konstruiranja.

3.2 Konstruiranje kao rješavanje zadatka

Drugi dio zahtjeva na koncepciju racunalnog modela procesa konstruiranja trebalo bi definirati analiziranje i razvrstavanje postupaka koje konstruktor koristi u tijeku rješavanja konstrukcijskog zadatka. Prema [22] zadatak procesa konstruiranja je pretvorba postavljenih zahtjeva u opis željenog strojnog sustava. Drugim rijecima potrebno je definirati i opisati strojni sustav (proizvod) koji ce proizvoditi željene ucinke i pri tome ispunjavati zahtjevana svojstva. Osim postavljenih zahtjeva potrebno je voditi racuna i o ostalim aspektima tehnicke i ekonomske naravi - npr. tehnologicnost, troškovi, itd. - što vrijedi i za sam proizvod, i za njegovo funkcioniranje u životnom vijeku.

Kao okosnicu ovog poglavlja (ujedno i cijelog rada) trebalo bi razmotriti slijedeci niz principjelnih pitanja :

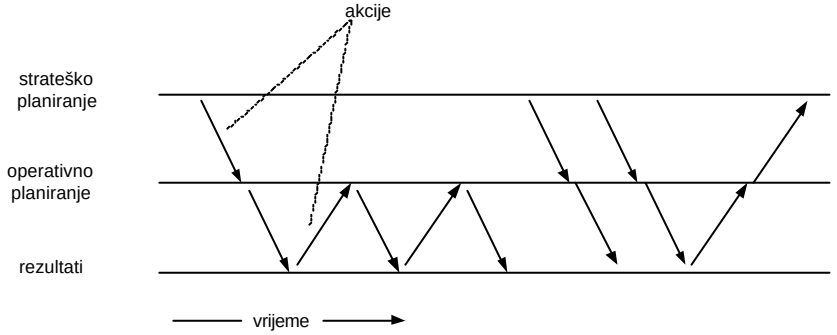
Kako konstruktor rješava zadatak, kako razmišlja, koje postupke koristi, kako organizira proces, kako se uci konstruirati ?

Razmatrajuci kako pristupiti obrazovanju konstruktora, Ullman [23] dolazi do zakljucka da ne postoji univerzalni “recept” rješavanja konstrukcijskog zadatka (i/ili problema). U uvodnim razmatranjima navodi slijedece:

1) Jedini nacin da se nauci konstruiranje - je konstruirati.
2) U tijeku rješavanja zadatka konstruktor upotrebljava tri vrste znanja:
   - znanje za generiranje ideja
   - znanje za procjenu, prosudivanje ideja
   - znanje za strukturiranje procesa konstruiranja

   Sposobnost generiranja ideja dolazi s iskustvom, ali je potreban i talent. Za prosudivanje ideja koristi se iskustveno znanje i znanje steceno obrazovanjem. Znanje potrebno za generiranje i prosudivanje (razmatranje) ideja ovisno je o domeni konstrukcijskog zadatka, dok je znanje o strukturiranju procesa konstruiranja vecinom neovisno o domeni zadatka.
1) Može se nauciti konstruirati kvalitetne proizvode, pod uvjetom da postoje odredene predispozicije, te dovoljno iskustvo za generiranje i prosudbu ideja.
2) Konstruiranje treba uciti paralelno u akademskoj i industrijskoj okolini.

Lindemann [58] razmatra model napredovanja kroz proces konstruiranja promatrajuci individualne nacine rada razlicitih konstruktora. Premda su promatrali rad konstruktora kroz niz godina, još uvijek nema kompletne slike i jasnog i detaljnog razumijevanja kako konstruktori napreduju kroz proces. Interesantan je Giaopulis-ov [59] prijedlog modela u kojem se promatraju tri razine - strateška, operativna i rezultati. Konstruktor pri odlucivanju i planiranju svog procesa zapravo neprekidno "šece" izmedu tih razina (slika 4).



Slika 4: Model procesa konstruiranja prema [59]

Akcije od strateškog prema operativnom planiranju promatraju se kao konkretizacija strateškog planiranja ("top-down" pristup). Akcije od razine rezultata prema operativnom planiranju promatraju se kao spontana analiza medurezultata ("bottom-up" pristup).
Analiza rezultata može uzrokovati i modifikacije strateškog planiranja.
Konstruktori prelaze sa razine strateškog planiranja na razinu operativnog planiranja u trenutku mijenjanja žarišta interesa. Na operativnoj razini razmatraju detaljnije i konkretnije planove. Akcije (kao rezultat takvih planova) vode do rezultata, koji mogu aktivirati slijedeci operativni plan sa daljnjim rezultatima. Povratak na stratešku razinu pojavljuje se u situacijama vanjskih interakcija ili teškoca na operativnoj razini.
Model Giaopulisa [59] opisuje dijelom i promatranja Lindemanna [58] koji razvija "fenomenološki model" procesa konstruiranja u kojem promatra prelaze i "skokove" izmedu razlicitih akcija, strategija i pogleda u tijeku konstruiranja. Navesti cemo neke od primjera takvih alternacija izmedu:

- apstraktnijih i konkretnijih razina
- cijelog sustava i detalja
- oblikovanja forme i proracuna
- sinteze i analize
- planiranih i oportunistickih koraka
- "bottom-up" i "top-down" pristupa
- poznatog i nepoznatog

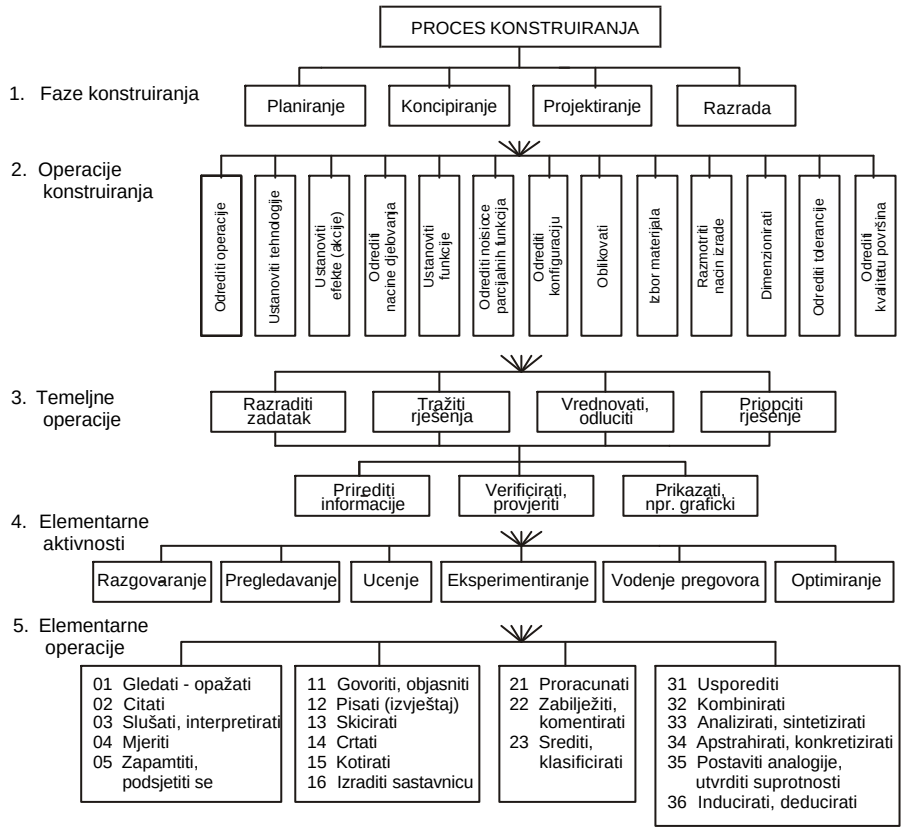
Prema [58] još uvijek su nejasni detalji procesa pronalaženja i izbora slijedeceg koraka.
Kljucna pitanja koje postavlja Lindemann:

- Koji skupovi važnih uvjeta ce dovesti do koje vrste akcije?
- Koja od alternativnih strategija i akcija je upotrebljiva i efikasna?

Fenomenološki model procesa konstruiranja još nije dovršen, a autor se pita da li ga je uopce i moguce kompletirati.

3.2.1 Struktura operacija u procesu konstruiranja

Svaki konstrukcijski proces može se rašclaniti na nekoliko kompleksnih faza, odnosno na operacije i korake. Struktura konstrukcijskog procesa može se prikazati kao niz strukturnih elemenata koji leže na razlicitim hijerarhijskim razinama. Pravila strukturiranja mogu poslužiti kao polazišta za upravljanje i vodenje, kao i planiranje konstrukcijskog procesa. Obzirom na temu i cilj ovog rada, posebno je interesantno razmotriti operacije koje konstruktor provodi u tijeku rješavanja zadatka. Prikaz strukture procesa konstruiranja u kojem su rašclanjene operacije i aktivnosti dan je prema [60].



Slika 5: Struktura aktivnosti u konstrukcijskom procesu prema [60]

Prikaz strukture konstrukcijskog procesa može se promatrati na dva nacina:

- kao hijerarhija kompleksnosti u aktivnostima konstrukcijskog procesa - gledano po vertikalnoj osi, svi elementi (operacije) iz nižih razina sadržani su u svakom od elemenata iz višim razina
- kao grupe (blokovi) aktivnosti (gledano po horizontalnoj osi) koji se ciklicki ponavljaju do postizanja ciljeva

Razmatranjem prikaza strukture operacija namece se ideja o razvoju skupa programskih alata za podršku obavljanju onih operacija (izmedu navedenih) koje se mogu algoritmizirati, a nisu (kao cjelina) podržane današnjim CAD sustavima. Naravno, takovi programski alati trebali bi omoguciti obavljanje operacija na poopcenom nivou apstrakcije, tj. neovisno o vrsti proizvoda i konstrukcijskog zadatka, uz mogucnost prilagodbe uvjetima okruženja u kojem se koriste. Razvoju takvih vrlo složenih programskih alata treba prethoditi razvoj integrirane programske okoline za modeliranje planiranja, pracenja i kontrole tijeka odvijanja akcija, odnosno skupa procedura koje obavljaju operacije. Pri tome se može pretpostaviti da se poopceni racunalni model odredene operacije može sastojati od niza programskih procedura - odnosno da se može graditi od elemenata sustava za podršku planiranju konstrukcijskog procesa.

Sa stanovišta racunalnog modeliranja, elementi od kojih se gradi model odredene operacije mogu biti bilo kakvi programi koji na bilo koji nacin vrše pretvorbu informacija unutar skupa informacija o proizvodu. Na temelju prethodnih postavki može se zakljuciti da pojam programske procedure (uz njenu funkciju i atribute) treba biti jedan od osnovnih elemenata odnosno entiteta sustava za podršku planiranju i izvodenju konstrukcijskog procesa. Pri tome osnovna obilježja (svojstva kao objekta-entiteta) svake programske procedure trebaju biti njena funkcija, odnosno namjena, te skup ulaznih i izlaznih atributa. U daljnjem tekstu uvesti cemo naziv "akcijska funkcija" za programsku proceduru kao entitet modela procesa konstruiranja.

Modeliranje poopcenih operacija može se realizirati uzastopnim pozivanjem niza akcijskih funkcija. Upotrebom postojecih programskih procedura (kao akcijskih funkcija) ili razvojem novih, uz osiguranje integralnog sucelja za prijenos podataka izmedu procedura, može se modelirati racunalska podrška izvodenju odredene konkretne operacije, koja vrijedi za odredeni zadatak ili klasu zadatka, odnosno konstrukciju, i to u uvjetima okruženja u kojem se odvija konkretan konstrukcijski proces.

Konstrukcijske operacije (na razini 2 prema slici 5) opcenito sadrže u sebi sve temeljne operacije sa razine 3 i nižih razina 4 i 5. One operacije i aktivnosti sa nižih razina koje se mogu algoritmizirati odnosno podržati racunalom na bilo koji nacin, mogu se racunalno modelirati izvodenjem jedne ili niza od nekoliko akcijskih funkcija. Kao akcijske funkcije mogle bi se dakle koristiti sve vrste komercijalnih ili aplikativnih programa koji se koriste u okruženju izvodenja konstrukcijskog procesa.

Naravno, uvijek je potrebno odrediti i redoslijed izvodenja akcijskih funkcija - što je zapravo definicija plana. Cilj procesa rješavanja zadatka (konstruiranja) je definiranje kompletne konstrukcije odnosno skupa svih informacija o proizvodu. Svaki složeniji zadatak može se razložiti na više podzadataka, od koji ce svaki imati svoj parcijalni cilj. To znaci da se za svaki podzadatak može postaviti redoslijed, odnosno plan izvodenja akcijskih funkcija cije ce izvodenje rezultirati pretvorbom pocetnog stanja odredenog podskupa informacija o proizvodu u željeno (ciljno) stanje. Drugim rijecima može se postaviti hijerarhija planova i podplanova koji cine ukupnu strukturu konstrukcijskog zadatka.

Za modeliranje racunalne podrške procesu konstruiranja nije nužno uvijek postavljati kompletan plan rješavanja zadatka na najvišem nivou apstrakcije, jer tu razinu kontrole

može preuzeti sam konstruktor na nacin da odreduje redoslijed izvodenja planova (odn. podplanova) koji mu služe kao podrška u procesu rješavanja podzadataka. Nacin strukturiranja hijerarhije planova i podplanova bitno ovisi o složenosti konstrukcijskog zadatka, znanju i iskustvu konstruktora kao i o raspoloživim programskim alatima (u konkretnom okruženju) koji se mogu koristiti kao akcijske funkcije. Ovdje se pretpostavlja da konstruktor sam treba modelirati podršku (planove) za rješavanje nekog zadatka ili klase zadataka iz svog djelokruga. Da bi to mogao efikasno napraviti mora imati na raspolaganju integriranu programsku okolinu koja treba sadržavati slijedece elemente:

1. Organizirano znanje o skupu akcijskih funkcija odnosno programa koje ce koristiti u tijeku rješavanja zadatka. Takova baza znanja prvenstveno treba sadržavati opise modela, namjene i ogranicenja svakog programa, te detaljne opise skupova ulaznih i izlaznih podataka, te nacina njihova transfera (u i iz programa).
2. Bazu znanja o proizvodu, pravilima izvodenja procesa konstruiranja i tehnologiji proizvodnje
3. Racunalni model procesa konstruiranja koji je koncepcijski dovoljno fleksibilan da se može prilagodavati i nadogradivati - pretpostavlja se da takove zahtjeve može ispuniti objektni model. Objektni model trebao bi sadržavati sve pojave i entitete iz "stvarnog svijeta" modelirane kao objekte.
4. Potpuno definiranu sintaksu svakog strukturalnog elementa modela i elemenata koji implementiraju kontrolu i vodenje procesa.

Osim modeliranja akcija i operacija potrebno je razmotriti koje još entitete treba ukljuciti u racunalni model procesa, obzirom na znacajke procesa konstruiranja.

3.2.2 Ogranicenja i odluke u procesu konstruiranja

Proces napredovanja od pocetne specifikacije (definicije zadatka ili problema) do rješenja, odnosno kompletnog skupa informacija o željenom proizvodu može se promatrati kao niz koraka u kojima se izmjenjuju procesi obrade informacija i donošenja odluka. Svaki od koraka može se naznaciti nekom odlukom koja na bilo koji nacin mijenja stanje unutar skupa informacija o proizvodu. Skup informacija o proizvodu cine svi generirani crteži, modeli, analize, proracuni, tehnicke upute, bilješke i prikupljeno znanje tijekom procesa konstruiranja.
Prema [23] mogu se razluciti dva razlicita gledišta o tome kako proces konstruiranja napreduje po koracima, odnosno od jednog stanja do slijedeceg.
Po jednom pristupu, opis proizvoda (konstrukcija) evoluira tijekom kontinuiranog (ciklickog) procesa usporedbe izmedu trenutnog stanja definiranosti skupa informacija o proizvodu i *cilja*, tj. skupa zahtjeva na konstrukciju definiranih konstrukcijskim zadatkom. Takav pristup implicira tocno poznavanje svih zahtjeva na pocetku rješavanja zadatka kako bi se jasno mogla definirati razlika u odnosu na trenutno stanje definiranosti konstrukcije. Razlika tih dvaju stanja tada kontrolira proces konstruiranja. Medutim, za vecinu konstrukcijskih zadataka i/ili problema, takav pristup je previše jednostavan, jer u pocetku procesa nisu svi zahtjevi precizno definirani, odnosno mogu se u tijeku procesa modificirati, pa ciljno stanje konstrukcije ne može na pocetku biti potpuno poznato.
Po drugom pristupu, na pocetku rješavanja zahtjevi na konstrukciju ogranicavaju skup mogucih rješenja na podskup svih mogucih rješenja. Kako proces konstruiranja napreduje, nova ogranicenja se dodaju da bi dalje reducirala moguca rješenja, koja se kontinuirano

eliminiraju do jednog konacnog rješenja. Drugim rijecima, konstruiranje je sukcesivno razvijanje i primjena ogranicenja dok ne preostane samo jedno rješenje.
Ogranicenja koja se dodaju u tijeku procesa konstruiranja proizlaze iz dva izvora. Jedan od izvora je opce strucno znanje konstruktora kao i znanje iz domene rješavanja zadatka. Kako svaki konstruktor raspolaže razlicitim znanjem, primjenjivati ce i razlicita ogranicenja, pa ce svaki riješiti zadatak na svoj, jedinstveni nacin.
Drugi tip ogranicenja koja se primjenjuju u tijeku konstrukcijskog procesa proizlazi iz odluka koje se donose u procesu konstruiranja. Odluke definiraju ogranicenja i na taj nacin mogu utjecati na slijedece odluke u daljnjem tijeku procesa. Može se reci da se vecina ogranicenja temelji na rezultatima konstrukcijskih odluka. Zbog toga je jedna od esencijalnih osobina konstruktora sposobnost donošenja odluka, ali na temelju dobro razradenih kriterija, i u trenutku kad raspolaže s dovoljnom kolicinom informacija.

Na osnovu prethodnih razmatranja može se uociti nekoliko osnovnih entiteta (objekata) u promatranju proceduralne prirode konstrukcijskog procesa: operacije, odnosno akcije, ogranicenja, odluke i ciljevi. Jedan od mogucih nacina modeliranja tijeka odvijanja procesa je u obliku niza akcija (operacija) koje se izvode planiranim redoslijedom, uz definiranu strukturu plana. Izvodenje svake operacije u pravilu mijenja stanje unutar skupa informacija o proizvodu koji se konstruira, pa nakon izvodenja akcije treba provjeriti definirana ogranicenja i donijeti odluke o daljnjem tijeku procesa. Pri tome se redoslijed izvodenja treba usmjeravati ka postizanju unaprijed definiranog cilja koji može biti krajnje rješenje zadatka ili jedno od parcijalnih rješenja.

3.2.3 Vrste zadataka, odnosno konstrukcija

Elementi strukture konstrukcijskog procesa - objekti (mehanicki sustav odn. proizvod), operacije i koraci te redoslijed njihova izvršavanja znatno variraju ovisno o zadatku. Znacajke, odnosno vrsta i složenost zadatka imaju bitan upliv na tijek i metode rješavanja. Prema stupnju odredenosti proizvoda zadatke, odnosno vrste konstrukcija možemo prema [56] razvrstati kao:

- ponovljene - izrada po vec gotovoj dokumentaciji uz eventualno prilagodavanje novoj tehnologiji (to je zapravo ponovljena izrada proizvoda)
- kvantitativno uskladene - kvalitativno ista konstrukcija, ali kvantitativno ponovljena na nacin da odredeni parametri dobivaju nove vrijednosti (cesto se u literaturi ovaj slucaj naziva parametarska konstrukcija)
- varijantne - osnovna funkcija i struktura proizvoda ostaju iste, ali se eventualno mijenjaju principi rada pojedinih parcijalnih funkcija
- prilagodne - osnovna funkcija ostaje ista, ali se uz izmjenu parcijalnih funkcija može mijenjati i struktura
- nove konstrukcije - izrada na osnovu novih principa rješenja kod istog, promijenjenog ili novog zadatka

Sustav za racunalnu podršku izvodenju procesa konstruiranja treba modelirati tako da ne ovisi o vrsti konstrukcijskog zadatka, ali da se može prilagoditi i koristiti u procesu rješavanja svake vrste zadatka ili unutar nekih složenijih vrsta zadataka barem za rješavanje parcijalnih zadataka. Naime, za nove ili vrlo složene konstrukcije teško je ocekivati da ima smisla postavljati model procesa (plan izvodenja) na najvišoj razini apstrakcije zadatka - npr. plan za konstruiranje lokomotive ili npr. plan za razvoj potpuno nove konstrukcije

nekog stroja specijalne namjene. Isto tako, za relativno jednostavne zadatke cesto nema smisla postavljati plan, ako se za isti utrošak vremena zadatak može i riješiti.

3.2.4 Metode konstruiranja

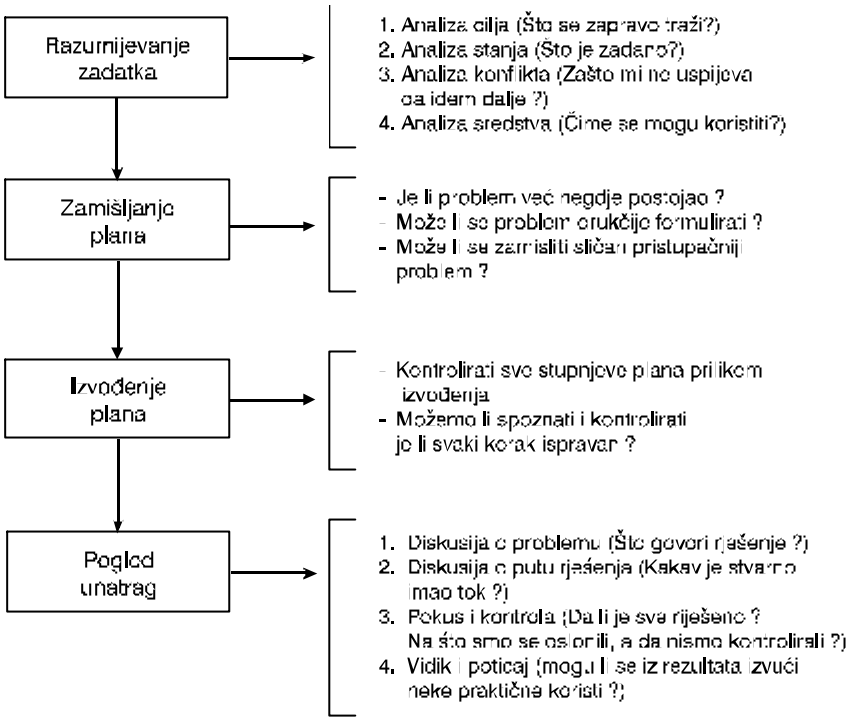
Prema [61] metodickim se konstruiranjem nastoji pomocu znanstvenih metoda razviti proces konstruiranja kao metodu koja omogucuje da se problematika konstruiranja rješava opcenito, a ne kao problematika konstruiranja sasvim odredenog stroja ili uredaja. Rijec je, zapravo o tome da se konstruiranje shvati kao proces u kojem se jednakim postupcima mogu rješavati razliciti zadaci. U literaturi iz podrucja znanosti o konstruiranju može se naci široki spektar prikaza i opisa razvijenih metoda konstruiranja, kao i metoda i principa traženja rješenja za pojedine parcijalne funkcije. Kao ilustraciju te cinjenice navedimo neke od autora: [1], [16], [23], [62], [63], [64], [65], [66], [67].
Implementacija postupaka i principa vecine razvijenih metoda metodickog konstruiranja u sustav za podršku konstrukcijskom procesu mogla bi biti tema razmatranja posebnog rada (za svaku od metoda).

Prikaz svih metoda konstruiranja bio bi preopširan, pa ce se ovdje samo izdvojiti Roth-ov [64] algoritamski postupak konstruiranja pomocu kataloga. Po ovom postupku u prvoj fazi provodi se analiza proizvoda ciji rezultat cini precizno formulirani zadatak. Tako formulirani zadatak sadrži potrebnu funkciju, tehnicke zahtjeve i potrebne troškove. Zahtjevi tvore kriterije pomocu kojih se u kasnijoj fazi može izvršiti izbor iz kataloga. Katalozi predstavljaju zbirku rješenja koja su se pokazala dobra za odredene konstrukcijske zadatke ili parcijalne funkcije. Katalozi mogu sadržavati informacije razlicitog sadržaja i rješenja razlicitog stupnja konkretizacije. U katalozima mogu biti : principi rješenja, fizikalni efekti, koncepcijska rješenja za kompletne zadatke, elementi strojeva, standardni dijelovi, materijali, gotovi dijelovi, itd. Mišljenje je autora, da se ovakva koncepcija kataloga može racunalno modelirati u obliku baze znanja, koja bi trebala biti koncipirana tako da se može koristiti u tijeku generiranja plana procesa konstruiranja. Navedeno mišljenje je i osnovni razlog za izdvajanje i navodenje ove metode konstruiranja.

3.2.5 Napredovanje u procesu rješavanja zadatka

Metode i principi traženja rješenja temeljene na heuristickim principima (odn. metodickim postupcima) predstavljaju alat za metodicki tok i upute za nacin promišljanja u tijeku konstruiranja. Stoga se namece pretpostavka da ove metode mogu poslužiti i kao smjernice modeliranja napredovanja u procesu rješavanja zadatka.

Ovdje ce se kao relevantna metoda (iz grupe metoda temeljenih na heuristickim principima) izdvojiti prikaz "metode tocno usmjerenih koraka".
Metoda tocno usmjerenih koraka sastoji se u postavljanju “pitanja samom sebi” na nacin koji usmjerava k rješenju. Predloženi nacin mišljenja odvija se u cetiri stupnja (slika 6): razumijevanje zadatka, zamišljanje plana, izvodenje plana i pogled unatrag.



Slika 6: Metoda tocno usmjerenih koraka

Prije zakljucnih razmatranja ovog poglavlja u kojem je analiziran proces konstruiranja, navesti cemo neke izvatke iz sustava postavki (izreka) navedenih u zadnjem poglavlju knjige [22] koje se (prema autoru) mogu interpretirati s jedne strane kao teze, a s druge strane kao konacni zakljucci.

- Postoji struktura relativno konkretnih operacija i njihovog redoslijeda, koja predstavlja idealan model napredovanja (u procesu konstruiranja), koji vrijedi za sve vrste konstrukcijskih zadataka.
- Model napredovanja za proces konstruiranja tehnickih sustava sadrži logican i probitacan redoslijed faza (operacija), koji je uskladen sa znacajkama sustava koji se konstruira, te pretpostavlja idealno stanje (položaj) operatora.
- Za odredeni projekt postoji jedan optimalan plan napredovanja koji odgovara konkretnoj definiciji zadatka i konkretnim operatorima procesa konstruiranja.

3.3 Zakljucci provedene analize procesa konstruiranja

Rezimirajuci razmatranja u ovom poglavlju može se postaviti pitanje kako treba koncipirati model racunalne podrške konstrukcijskom procesu, obzirom na navedene znacajke konstrukcijskog procesa i promatranje konstruiranja kao rješavanja zadatka.
Cilj istraživanja u projektu razvoja objektnog modela procesa konstruiranja nije razvoj "automata", vec fleksibilnog modela koji omogucuje manipulaciju informacijama u razlicitim oblicima i na razlicitim razinama apstrakcije, sukladno prirodnom toku promišljanja, odnosno razvoja konstrukcije. Takav sustav treba djelovati kao "pomocnik" konstruktora, na taj nacin da konstruktor može modelirati racunalnu podršku kakva mu za odredeni zadatak odgovara, koristeci raspoložive programske alate. Pri tome sustav treba biti primjenjiv neovisno o vrsti i domeni zadatka kao i o fazi procesa konstruiranja u kojoj

se koristi. Iz ovako postavljenih zahtjeva proizlazi da razvoju sustava treba pristupiti tako da se prvo razvija opca okosnica ili okolina (okruženje) koja podržava (i integrira) buduce dijelove sustava. Na temelju takve integrirane okoline mogu se dalje graditi više ili manje "inteligentni" podsustavi.

Napredovanje u procesu konstruiranja može se prikazati planom, tako da operacije u procesu konstruiranja predstavljaju operatore plana. U racunalnom modelu plana pojam operatora može se modelirati izvodenjem neke programske procedure koja na bilo koji nacin vrši odredenu pretvorbu informacija na nekom podskupu skupa informacija o proizvodu koji se konstruira. Iz razmatranja tijeka rješavanja konstrukcijskog zadatka slijedi da proces napreduje kroz donošenje odluka, te postavljanje i provjeru ogranicenja, što znaci da odluke i ogranicenja takoder treba kao objekte implementirati u strukturu plana. Pretpostavlja se da sam plan, odnosno zapis plana može biti temelj prikaza tijeka izvodenja procesa u objektnom modelu procesa konstruiranja. Da bi konstruktor mogao efikasno modelirati podršku rješavanju konkretnog zadatka iz svog djelokruga, mora imati na raspolaganju integriranu programsku okolinu u kojoj ce na temelju definirane sintakse generirati i nakon toga eksploatirati plan rješavanja zadatka. Pri tome nije nužno da postavlja plan rješavanja na najvišoj razini apstrakcije jer iz razmatranja metoda i principa rješavanja konstrukcijskih zadataka slijedi da se svaki konstrukcijski zadatak dade razložiti na skup jednostavnijih podzataka za koje su poznate metode rješavanja.

Pošto sustav ne bi trebao biti koncipiran kao "automat", potrebna je ljudska kontrola procesa konstruiranja, ali uz uvjet da sam racunalni model procesa u tome ima odgovarajucu ulogu (podršku), koliko god i gdje god je to moguce.

4. Pristupi i polazišta modeliranja procesa konstruiranja

Vec je ranije napomenuto da je do danas u Znanosti o konstruiranju predložen velik broj modela procesa konstruiranja razlicitih autora. Nekoliko modela svojim znacajem se istice, i obicno se u literaturi za njih koristi naziv "opce teorije konstruiranja". Pregled i osvrt na cetiri takva modela dan je u slijedecem dijelu ovog poglavlja.
Ipak, analiza tih modela nije sama po sebi dovoljna kao polazište za koncipiranje objektnog modela procesa konstruiranja. Potrebno je osim toga razmotriti još nekoliko podrucja i razlicitih pristupa unutar njih. Prije svega to se odnosi na razvijene informacijske sustave (modele) i njihovu primjenu u konstruiranju, te na pristupe planiranju u podrucju umjetne inteligencije. Kvaliteta modela (opcenito) ovisiti ce najviše o primjenjenim principima modeliranja, pa je kao dio ovog poglavlja dan i kratki osvrt na problematiku modeliranja. Od posebne važnosti za cilj ovog rada je i razmatranje metoda prikaza procesa i topologije prikaza akcija u procesu konstruiranja.
Sve navedene teme, odnosno podrucja, obradena su u ovom poglavlju kao uvodna razmatranja pristupa i polazišta koncipiranja entiteta objektnog modela procesa konstruiranja. Prikaz objektno orijentirane tehnologije modeliranja i programiranja izdvojen je kao slijedece poglavlje.

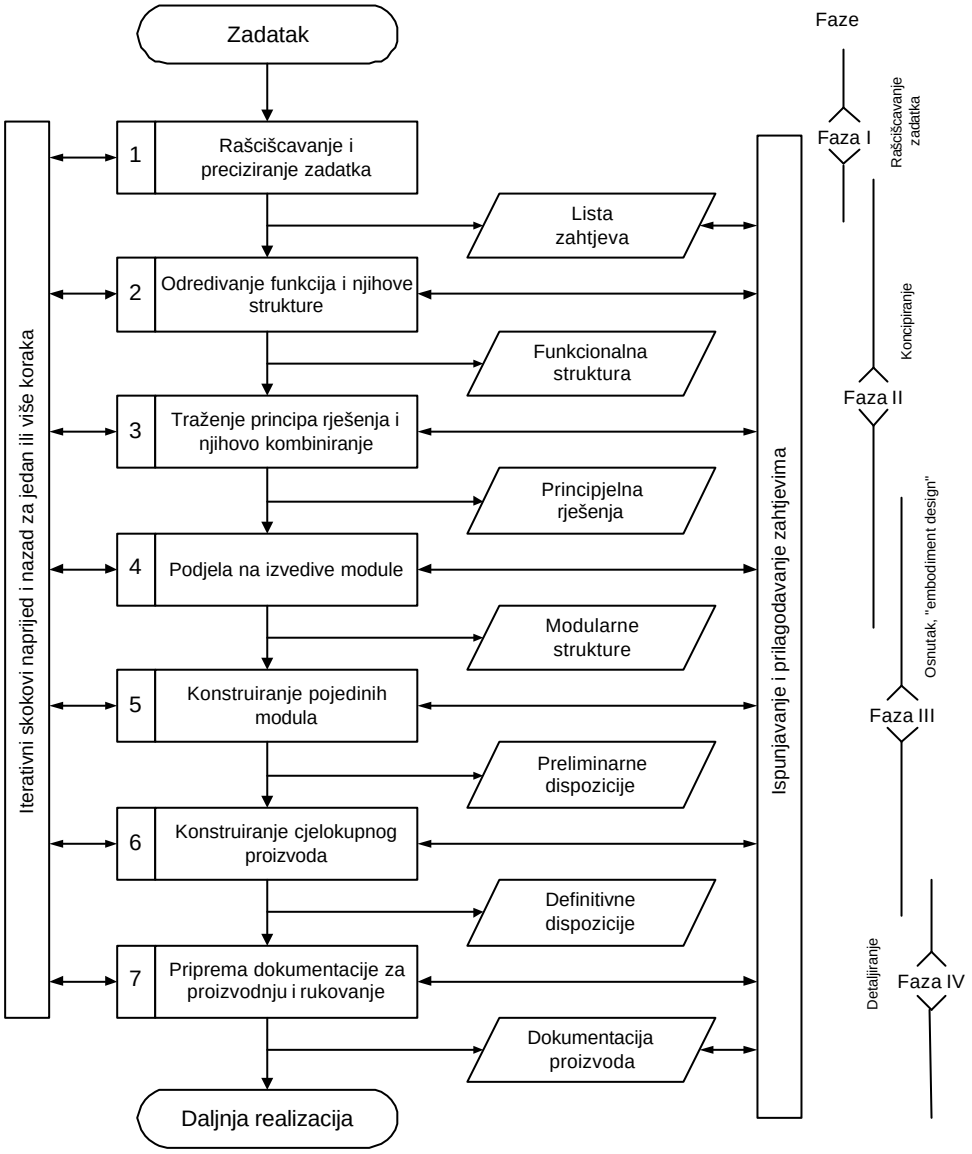
4.1 Pregled znacajnijih "teoretskih" modela procesa konstruiranja

Zakonitosti koje vladaju u procesu konstruiranja kao kreativnoj ljudskoj djelatnosti, nastoje se opisati teorijama cija su polazišta uvjetovana okolinom i svim njenim uplivima (filozofskim, povijesnim, sociološkim, gospodarskim, itd.). Do sada razvijene teorije mogu se razvrstati po okruženjima u kojima su nastale na njemacke (evropske), americke i japanske. U ovoj je glavi dan pregled jednog proceduralnog modela i tri (po mišljenju autora) najznacajnije teorije.

4.1.1 Proceduralni model procesa konstruiranja prema VDI 2221

U njemackom okruženju težnje za sistematizacijom i sustavnim metodickim pristupom konstruiranju najranije su zapocele i najviše su bile izražavane. Velik broj autora predlagao je svoje metode i principe. Prirodno je da je u takvom okruženju razvijen i proceduralni model procesa konstruiranja u obliku "preporuka" odn. "vodica" (richtlinie, guideline). Model je rezultat rada skupine autora i objavljen je u obliku preporuka [21], [68]. Namjena modela je opcenita, te su dani i primjeri upotrebe za razlicite industrijske grane. Ipak, razmatrane ideje cini se da su najviše vezane uz strojarsko konstruiranje [65].

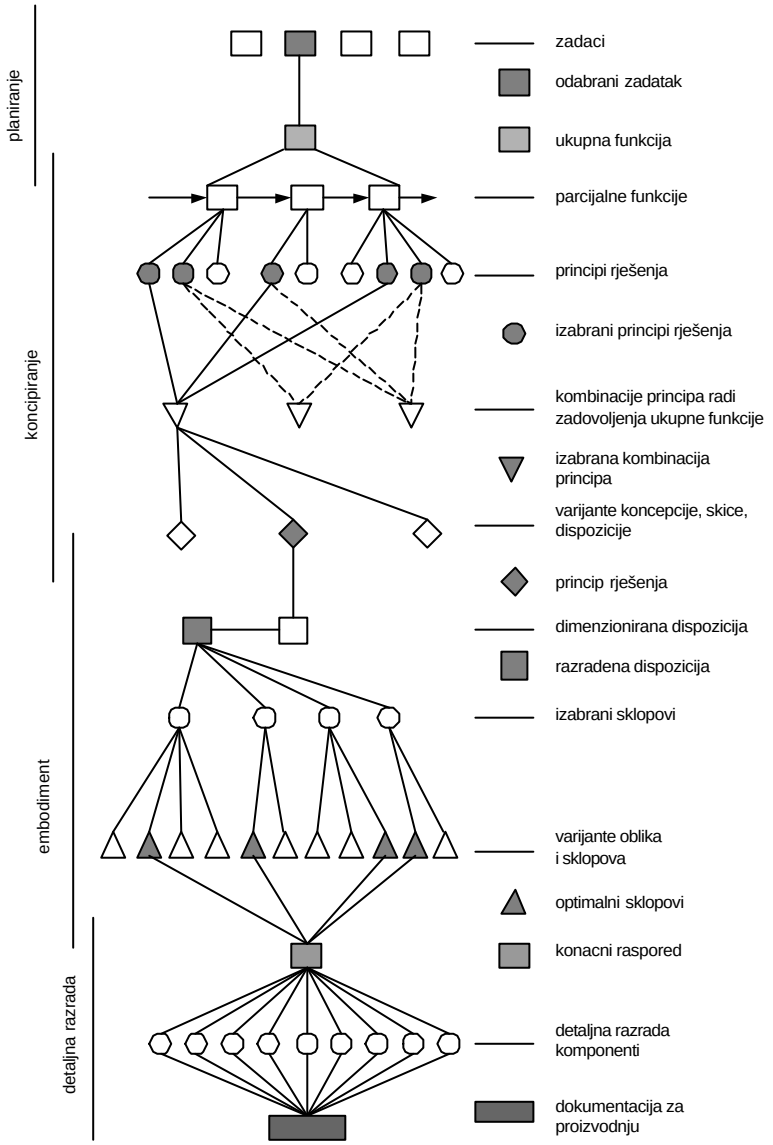
U modelu su naznaceni glavni koraci procesa od apstraktne razine raščišcavanja zadatka do detaljiranja konstrukcije. Prisutnost iterativnih petlji znaci da ovaj model nije vremenski orijentiran, kako se cesto shvaca [58].
Slika 7 prikazuje opci pristup procesu konstruiranja, odnosno korake i faze procesa. Nakon svakog koraka treba donijeti odluku da li se može prijeci na slijedeci ili treba ponoviti prethodni. Faze procesa mogu se promatrati i kao razine apstrakcije [58]. Oštre granice izmedu faza ne mogu se postaviti, a isto tako ne treba kruto promatrati niti redoslijed faza. Ovakav model ne prikazuje proces rješavanja problema, odnosno kako se dolazi do rješenja [65]. U procesu rješavanja problema rješenja se generiraju i testiraju unutar svake faze. To znaci da zapravo u svakoj fazi konstruktor prolazi kroz temeljni ciklus procesa konstruiranja, cesto i nekoliko puta.



Slika 7: Faze procesa konstruiranja prema [21]

U svakoj fazi treba misliti na više alternativa rješenja. Razrada svih varijanti rješenja kroz sve faze dovela bi do kombinatoricke eksplozije. S druge strane usredotocenje na samo jednu putanju unutar mreže mogucnosti znaci da se mogu previdjeti bolja ili najbolja

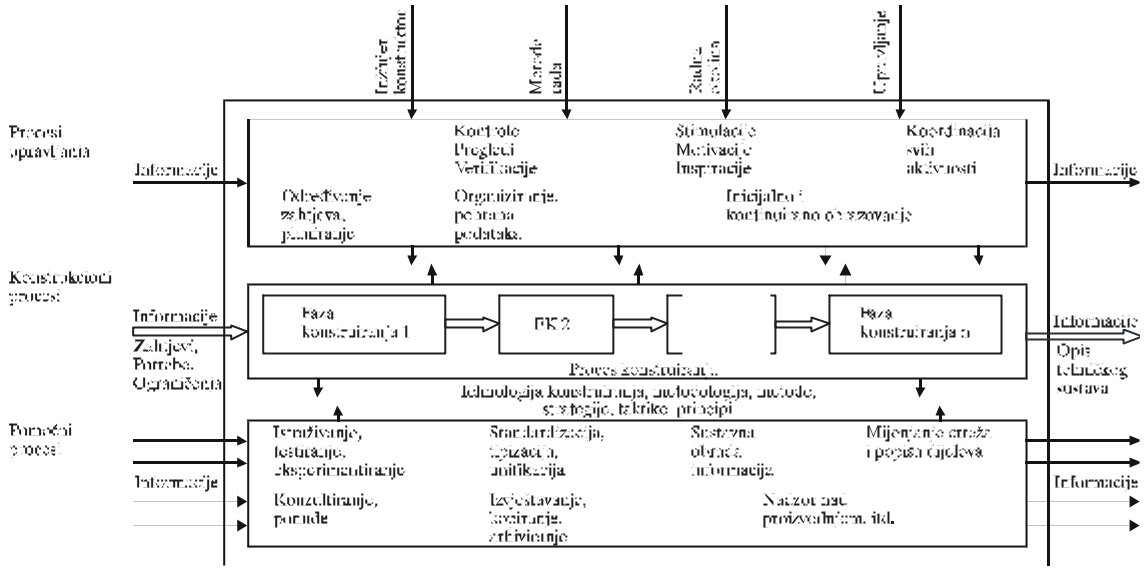
rješenja. Stoga konstruktor mora divergirati i konvergirati u svakoj fazi. Taj važni metodicki princip ilustriran je slikom 8.
Napomenimo ovdje da je ovakav nacin promatranja procesa (divergencija i konvergencija) izuzetno važan i za prikaz tijeka procesa u racunalnom modelu.



Slika 8: Dijagram toka procesa konstruiranja prema [68]

4.1.2 Opci model procesa konstruiranja (Hubka, Eder)

Opci model konstruiranja koji je postavio Hubka 1973. godine [22], daje pregled skupa aktivnosti, vrsta procesa i upliva na proces konstruiranja. Razvojem opceg modela konstruiranja autor je tijekom vremena postavio teoriju tehnickih sustava, te unutar nje noviji "opci proceduralni model konstruiranja" [60]. Autor prve verzije teorije je Hubka, ali novija engleska izdanja [16], [60] potpisuje i Eder.



Slika 9: Opci model procesa konstruiranja prema [22]

Opci model procesa konstruiranja može se interpretirati na slijedeci nacin:

- konstruiranje je proces transformiranja informacija od zahtjeva kupaca do potpunog opisa predloženog tehnickog sustava
- prikazuje se osnovna struktura procesa, ukljucujuci regulacijske, kontrolne i pomocne procese
- kao direktni operatori, konstruktori i njihova sredstva za rad izvode akcije (efekte) na skupu informacija (operanada konstrukcijskog procesa)
- prikazuje se utjecaj ostalih razlicitih faktora na proces konstruiranja (metode rada, radna okolina, upravljanje)

S ciljem da se predloženi model procesa konstruiranja osuvremeni, te omoguci primjena racunala, teorija je tijekom godina evoluirala do "opceg proceduralnog modela procesa konstruiranja" [60], [16]. U tom modelu proces konstruiranja dijeli se na cetiri osnovna dijela:

- Elaboriranje, odnosno pojašnjavanje zadane specifikacije
- Koncipiranje - postavljanje funkcionalne strukture i strukture osnovnih fizickih komponenti sustava
- Definiranje dispozicije sustava - preliminarno i konacno dimenzionalno rješenje
- Detaljna razrada rješenja, izrada dokumentacije

Svaka od navedenih osnovnih faza rašclanjuje se dalje na korake. Svaka faza sadrži operacije ispitivanja mogucnosti unapredenja, vrednovanja, izbora, odlucivanja, provjere i potvrdivanja. Faze se povezuju i odvijaju unutar procesa rekurzije, iteracije, dekompozicije i analize povratnih informacija.

Za ovako koncipiranu teoriju karakteristicno je da je vecina pojmova sistematizirana i opisana, te je teorija lako razumljiva, edukativna i pregledna. Neki autori navode da je osnovni nedostatak što se teorija osniva na fenomenološkom opisu, bez egzaktnog aparata, što otežava formalizaciju teorije kao polazišta za razvoj racunalnog modela procesa konstruiranja.

4.1.3 Opca teorija konstruiranja (Yoshikawa, Tomiyama)

Opca teorija konstruiranja (Yoshikawa, General Design Theory - GDT) pociva na tri abdukcione hipoteze iz kojih autor izvodi tri aksioma. Definicije pojmova i teoremi teorije grade se na formalnom aparatu aksiomatske teorije skupova [69], [70], [71]. Od prvog objavljivanja pred gotovo 20 godina proširena su razmatranja unutar GDT, te se sada u literaturi navodi i kao "Extended General Design Theory".
Proklamirani ciljevi ove teorije su:

- znanstveno utemeljenje procesa konstruiranja
- formaliziranje prakticno upotrebljivog znanja o metodologiji konstruiranja
- formalan nacin prezentiranja znanja o konstruiranju, s ciljem racunalne implementacije.

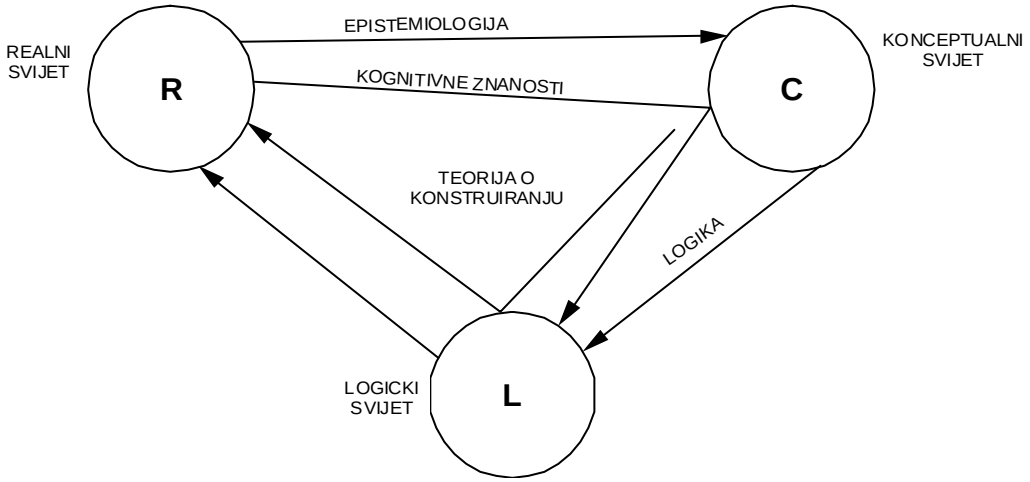
Razmatranja "Opce teorije konstruiranja" izvode se u kontekstu poimanja tri razlicita svijeta (i preslikavanjima izmedu njih - slika 10) :

- realnog R - gdje egzistiraju konkretni entiteti
- konceptualnog C - gdje zamišljamo entitete iz realnog svijeta, svaki pojedinac posjeduje svoj vlastiti konceptualni svijet.
- logickog L - kao svijet simbola, logike, matematike itd.

Tako su npr. fizika, epistemologija, kognitivne znanosti, logika, tehnologija i znanost o konstruiranju definirane kao neka od preslikavanja izmedu tih svjetova (slika 10).

Stoga, (po shvacanju autora teorije) konstruiranje je niz preslikavanja iz konceptualnog svijeta u realni svijet, preko logickog svijeta. U ovom se dijelu slažu gotovo svi pristupi fenomenu konstruiranja, s malo razlicitom terminologijom, naime cešci je termin "transformacija" od "preslikavanja".

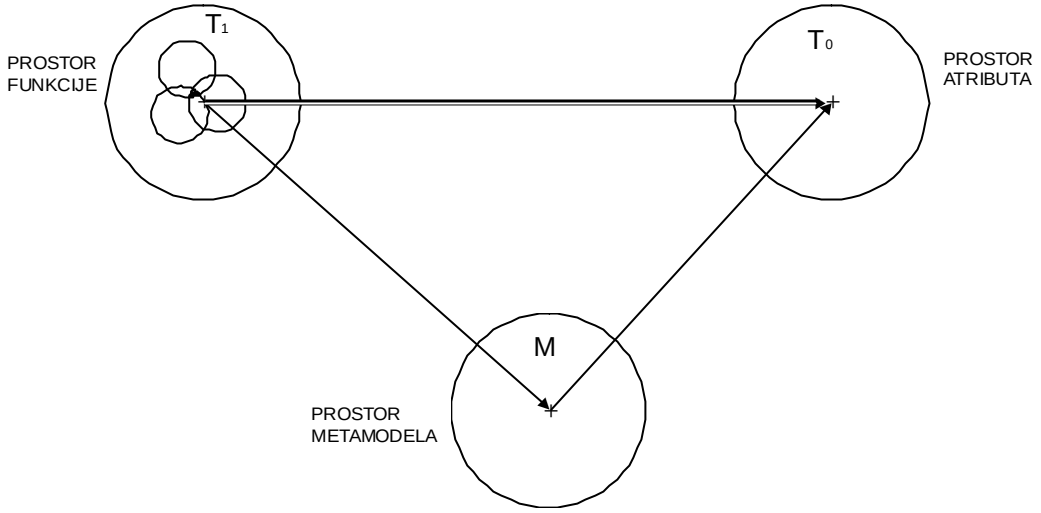
Konstruiranje se shvaca kao aktivnost kod koje se kreira entitet u realnom svijetu, od prve ideje koja se stvara u konceptualnom svijetu (na osnovu utjecaja iz realnog svijeta), kroz logicki svijet (u kojem egzistiraju samo simboli, npr. crteži).



Slika 10: Shema preslikavanja unutar prostora GDT prema [70]

Fizika je prikazana kao preslikavanje **R** $\rightarrow$ **C** $\rightarrow$ **L**, gdje je prvi dio preslikavanja promatranje fenomena koje je pokrivano epistemologijom, a druga polovina preslikavanja je primjena matematike ili logickih operacija koje služe da bi se fizikalni fenomeni opisali na objektivan (znanstveni) nacin.

Konstruiranje se definira kao preslikavanje neke tocke iz funkcionalnog prostora u tocku u prostoru atributa (uz idealno znanje).
Buduci je metrika prostora funkcije i prostora atributa obicno nekompatibilna, uveden je koncept metamodela (M), kao meduprostora u tom preslikavanju (slika 11).



Slika 11: Koncept metamodela [70]

Može se prigovoriti da GDT preidealisticki prikazuje konstruiranje, ali je ona okvir koji omogucuje racunalnu implementaciju. Naime teorija razmatra tri segmenta znanosti o konstruiranju: teoriju konstruiranja, teoriju objekata konstruiranja, i teoriju znanja. Temeljem toga realiziran je i sustav IDDL [72] koji eksploatira ideju opisa atributa entiteta funkcijama. Problem nesigurnih i netocnih specifikacija rješavaju autori kombinacijom nodalne logike i matematike predikata prvoga reda.
Zakljuciti cemo s mišlju da GDT ne može adekvatno predstavljati realno konstruiranje. Uvodenjem prihvatljivih ogranicenja i relaksirajuci prestroge zahtjeve formalnog modela zastupljenog pogotovo u "idealnom" dijelu GDT teorija može poslužiti (i vec je) kao smjernica za gradnju boljih CAD sustava.

4.1.4 Aksiomatska teorija konstruiranja (Suh)

Aksiomatska teorija konstruiranja [73] nudi sustavni pristup konstruiranju proizvoda i planiranju proizvodnje. Argumentacija i motivacija razvoja ove teorije slicne su kao i u prethodnim primjerima. Pri tome Suh naglašuje [74] da teorija "osigurava istovremeno formalni opis konstrukcijskog procesa i skup osnovnih principa odlucivanja". Proces konstruiranja podijeljen je na slijedece korake:

- odredivanje ciljeva konstruiranja koji zadovoljavaju zadani skup potreba narucitelja,
- generiranje ideja za kreiranje vjerojatnih rješenja,
- analiza predloženih rješenja,
- izbor rješenja koje najpovoljnije zadovoljava ciljeve,
- primjena (razrada[1]) odabranog rješenja

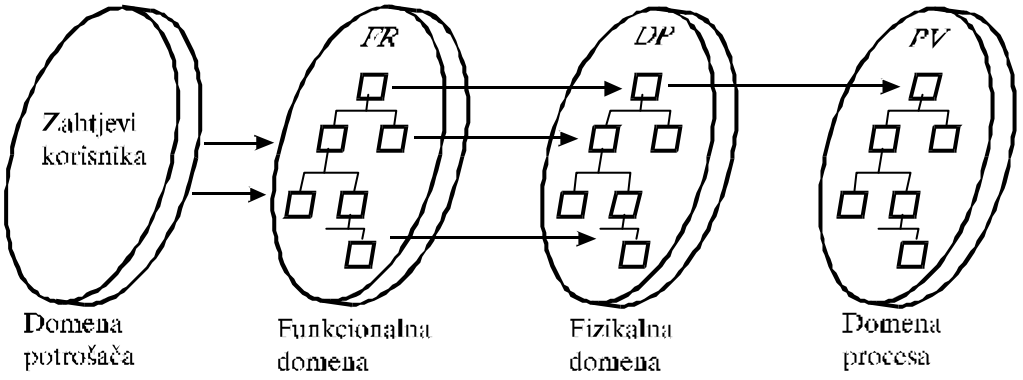
[1] Suh navodi "implementacija", naime projektiranjem (ili konceptualizacijom) je konstrukcija odredena, treba je realizirati razradom i izradbom.

Aksiomatski pristup opisanog konstrukcijskog procesa temelji se na slijedecim konceptima.

- Konstruiranje ukljucuje kontinuiranu obradu informacija izmedu i unutar cetiri razlicite domene (slika 12).
- Alternativna rješenja kreiraju se pridruživanjem zahtjeva specificiranih u jednoj domeni, skupu znacajki druge domene.
- Izlaz svake domene upotpunjuje se u hijerarhijskom nacinu od apstraktnog koncepta do potpuno odredenih informacija.
- Hijerarhijska dekompozicija u jednoj domeni ne može se provesti nezavisno od hijerarhijom susjednih domena, odnosno dekompozicija slijedi "cik-cak" preslikavanja unutar morfologija susjednih domena (Ovime Suh inzistira na morfološkoj slicnosti struktura dekompozicija susjednih, a time i svih domena. Formalno to nije nigdje dokazano).
- Dva aksioma tvore racionalnu bazu za vrednovanje predloženih alternativa rješenja i nakon toga izbor najbolje alternative.

Potrebe se korisnika pojavljuju u domeni korisnika (domena potrošaca), a formaliziraju u funkcionalnoj domeni kao skup koji ciji su clanovi funkcionalni zahtjevi (FR) kojima je odredeno nalaženje rješenja. Pri tome je svaki funkcionalni zahtjev (FR) po definiciji nezavisan od ostalih funkcionalnih zahtjeva. Kako se kompleksnost opisa povecava s brojem funkcionalnih zahtjeva za primjenu je teorije važno da se potrebe opisuju s minimalnim skupom nezavisnih zahtjeva.

Kreativna faza konstruiranja ili sinteza ukljucuje pridruživanje FR parametrima konstrukcije (DP) u fizikalnoj domeni. Broj mogucih rješenja za svaki dani skup FR ovisi o imaginaciji, i iskustvu konstruktora. Stoga aksiomi služe za odredivanje prihvatljivih rješenja.



Slika 12: Domene procesa konstruiranja po aksiomatskoj teoriji [74]

Aksiom 1 (aksiom nezavisnosti):

Osigurati nezavisnost funkcionalnih zahtjeva.

Aksiom 2 (informacijski aksiom):

Minimizirati informacijski sadržaj konstrukcije[1].

Prvi aksiom odreduje prirodu pridruživanja izmedu "onoga što se traži" (FR-a) i "kako se to postiže" parametara konstrukcije (DP-a). Suh smatra da "dobro konstrukcijsko rješenje" osigurava nezavisnost funkcionalnih zahtjeva.

[1] ovdje se "konstrukcija" odnosi na konstrukcijsko rješenje, koncepciju, logicki model, a ne na fizicku izvedbu. Ova primjedba vrijedi svuda u ovom poglavlju, osim gdje to nije izricito navedeno.

U drugom se, informacijskom, aksiomu odreduje sadržaj informacija konstrukcije kao mjera za odredivanje relativne vrijednosti i usporedbu alternativnih rješenja koja zadovoljavaju prvi aksiom.

4.1.4.1 Znacenje aksioma nezavisnosti

Pridruživanje izmedu *m*-komponentnog vektora **FR** (vektor funkcionalnih zahtjeva) i odgovarajuceg vektora parametara konstrukcije **DP** s *n* komponenata odredeno je izrazom:

$$\{\mathbf{FR}\} = [\mathbf{A}]\{\mathbf{DP}\} \qquad 4.1.4.1$$

gdje je **A** matrica konstrukcije kojom se odreduje priroda i odnos izmedu svakog funkcionalnog zahtjeva FR_i u **FR** i svakog parametra konstrukcije DP_j u **DP**.
Opcenito se onda clanovi matrice mogu prikazati kao:

$$A_{ij} = \frac{\partial FR_i}{\partial DP_j} \qquad 4.1.4.2$$

i povezuju komponente vektora **FR** s komponentama vektora **DP**.

Matrica A ima slijedeci oblik:

$$[\mathbf{A}] = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \qquad 4.1.4.3$$

Vrednovanje A_{ij} provodi se u fizikalnoj domeni tijekom konstruiranja, osim u slucaju kada je A_{ij}=*konst*.

Izraz 3.3.1 Suh naziva "jednadžbom konstrukcije" u kojem lijeva strana prikazuje želje formulirane u obliku zahtjeva koje treba ispuniti, a desna strana "kako mislimo zadovoljiti zahtjeve".

4.1.5 Osvrt na prikaz teorija o konstruiranju

U prethodnim poglavljima dan je sažeti prikaz istraživanja u podrucju znanosti o konstruiranju, te odabranih relevantnih teorija, od do sada razvijenih unutar znanosti.
Iz dosadašnjih razmatranja proizlazi da ne postoji jedinstvena teorija koja objašnjava fenomen konstruiranja. Pri izvodenju teorija, izražajni aparat koji se koristi za objašnjavanje zakonitosti podrucja nije od najvece važnosti. Teorija se može formulirati formalnim i neformalnim aparatom, bitno je da opisuje odnose i pojave podrucja promatranja dosljedno.
Iz prikaza danih u ovoj glavi rada može se zakljuciti.

- Sve prikazane teorije nastoje utvrditi zakonitosti koje vladaju u važnoj djelatnosti ljudi - stvaranju proizvoda.
- Sve teorije inzistiraju na sistematicnosti u konstruiranju.

- Polazne osnove teorija (na kojima se ove osnivaju) veoma su slicne, odnosno preklapaju se[1].
- Niti jedna (autoru ovoga rada poznata) teorija ne obuhvaca sve aspekte procesa konstruiranja.
- U svim se pristupima odvojeno promatraju konstrukcija, proces konstruiranja, te nacin provodenja procesa.
- Povezivanje funkcije i forme nastoji se formalno povezati, bez (do sada) rješenja na opcoj razini.
- Postoji koncenzus u prikazu procesa konstruiranja. Iako autori drugacije opisuju proces i razgranicenja faza u procesu, razlike nisu suštinske, vec terminološke.
- Slicnosti izmedu teorija vece su od razlika.
- Teorije se medusobno razlikuju u:
    - primjenjenoj metodologiji i terminologiji,
    - razini primjene egzaktnog (formalnog) aparata,
    - naglascima pojedinih segmenata razvoja proizvoda,
    - preporucenoj metodologiji (i tehnologiji).
- Navedene razlike (vecim dijelom) proizlaze iz razlicitosti osobnih iskustava autora.

Možemo zakljuciti da unutar znanosti o konstruiranju postoje razlicite škole koje opisuju fenomen konstruiranja razlicitim metodama.
Primjena racunala nesumnjivo predstavlja snažan impuls primjeni formalnog matematickog aparata u osnivanju teorija.
Poseban znacaj razvoja teorija ogleda se u sustavnom pristupu izucavanja konstruiranja. Medutim na sadašnjoj razini razvoja teorija, treba podržati samo ona htijenja koja omogucuju konstruktorima napredovanje od problema do rješenja, uz ekstenzivno i objektivno pretraživanje prostora rješenja, ne ogranicavajuci pri tome kreativnost konstruktora.
Može se zakljuciti da ni jedna teorija nije sveobuhvatna, niti pogodna kao osnova (polazište) za razradu objektnog modela procesa konstruiranja. Svaku teoriju možemo promatrati kao zaseban pristup i doprinos, cije spoznaje trebaju biti "ideje vodilje" pri koncipiranju modela. Odredene dijelove, odnosno razmišljanja vjerojatno je moguce i implementirati u objektni model.

Što se danas ocekuje od teorije konstruiranja?

Izmedu ostalih autora koji nude odgovore na takvo pitanje izdvojiti cemo Vajnu, [4] i [75]:

- da ukljuci sve znacajke cijelog životnog vijeka proizvoda (razmatranje svih aspekata termina "design for x")
- da ukljuci i podrži vremenski paralelne procese (istovremeno (paralelno) inženjerstvo)
- formalizacija strukture i sadržaja procesa konstruiranja u modelu procesa za deriviranje i realizaciju funkcija za cjelokupnu racunalnu podršku

Jedan od prvih koraka u razvoju teorije konstruiranja koja bi opisala cjelokupni proces konstruiranja prema [75] je korektno ujedinjavanje razlicitih parcijalnih modela uz korištenje postojecih mogucnosti matematickog modeliranja.

[1] Dovoljno je npr. usporediti polazišta, namjenu i obrazloženje razloga razvoja teorija u radovima Kesselringa (koji je na neki nacin duhovni otac srednje-europske škole) i Suha, ili Suha i Yoshikawe.

4.2 Problematika modeliranja

U kontekstu teme ovog rada, važno je postaviti pitanje kako treba modelirati? Kojih se principa treba držati pri koncipiranju modela? Kako kreiramo modele i kojim se tehnikama koristimo za prikaz modela?
Vrlo interesantna razmatranja o atribuciji modela, a možda i dio odgovora na ova pitanja daje Schregenberger u [76]. Schregenberger citira Stachowiak-a, koji je u [77] predložio integralnu koncepciju modela. Prema [77] izjave o *necemu* (o "originalima") uvijek se odnose na *odredenu svrhu*, prikazuju original sa *razlicitih aspekata*, potjecu *od nekoga*, usmjerene su *nekome*, i dio su *specificnog konteksta.*
Takva opca koncepcija omogucuje nam sistematizaciju i razmatranje bilo koje vrste ljudskog izraza. Modele upotrebljavamo za spremanje, procesiranje i razmjenu znanja. Optimalno upravljanje modelima odlucujuce je za uspješnost intelektualnog inženjerskog rada.
Froese [78], [79] razmatra problematiku modeliranja informacijskih sustava u domeni projektiranja u gradevinarstvu. U clanku [78] dan je pregled i komparativna analiza konceptualnih modela ("core models") procesiranja informacija konstruiranja u podrucju arhitekture i gradevinarstva. Kao najveci, i vjerojatno najznacajniji projekt u danom pregledu naveden je dio standarda ISO 10303 (STEP). Prema autoru nisu još razvijene komponente STEP-a za konceptualno modeliranje projektnih procesa (stanje 1994).

4.2.1 Informacijski modeli (sustavi)

Primjena baza podataka kao sustava za pracenje proizvodnje (financijsko, organizacijsko i logisticko) svakako ima veliki upliv i na modeliranje racunalne podrške konstruiranju. Kao informacijske sustave možemo promatrati i tzv. "EDM" sustave (engineering data management). EDM sustavi zapravo su jedna komponenta racunalne podrške procesu konstruiranja, koju treba suceljima povezati sa modelom procesa konstruiranja.
Prema [80]: Informacijski model je formalni opis ideja, cinjenica i procesa koji zajedno cine model dijela stvarnog svijeta, i osiguravaju eksplicitni skup interpretacijskih pravila. U idealnom slucaju informacijski model bi trebao osigurati potpunu, tocnu i jednoznacnu sliku pojave koja se modelira.
Razvoj informacijskih sustava s pripadajucim aplikacijama prvenstveno je vezan uz netehnicku problematiku, dok primjena ovakvih tehnologija u inženjerstvu zahtijeva, zbog drugacije prirode informacija s kojima se operira, u mnogocemu drugaciji pristup [81]. Stoga ce se razmotriti znacajke inženjerskih podataka, da bi se naznacili moguci problemi informacijskog modeliranja procesa konstruiranja.

4.2.1.1 Priroda inženjerskih podataka

Priroda inženjerskih podataka vrlo je razlicita od podataka u poslovnim aplikacijama, otkuda stiže vecina razvoja u podrucju tehnologije baza podataka. Uz potencijalno veliku kolicinu, podatke u inženjerskim aplikacijama karakterizira raznorodnost i vrlo složena povezanost izmedu struktura.
Prema [81] podaci u inženjerstvu mogu se opisati na slijedeci nacin:

- **Struktura inženjerskih podataka je neuniformna i nepredvidiva**. Instance jedne konceptualne strukture mogu varirati u velicini, a skup podataka koji prikazuje cijeli

proizvod može sadržavati veliki broj razlicitih struktura podataka, s relativno malim brojem instanci svake strukture.
- **Mnogi od koncepata zahtijevaju prikaz koji je mreža struktura i odnosa**. Npr. za prikaz žicanog modela potrebne su geometrijske informacije koje odreduju tocke/vrhove modela. Topološke informacije odreduju bridove koje povezuju tocke. Za vizualizaciju žicanog modela potrebne su i geometrijske i topološke informacije.
- **Veze medu strukturama su mnogobrojne, a iste strukture mogu imati razlicite uloge**. Npr. strojni dio je u jednom slucaju proizvod, a u drugom komponenta u sklopu složenijeg proizvoda.
- Znacajan udio podataka ovisi o postojanju drugih podataka.
- Cjelovitost skupa podataka relativna je u odnosu na stadij životnog vijeka proizvoda.
- **Razina tocnosti numerickih podataka varira ovisno o znacenju podataka i aplikaciji koja ih koristi**. Npr., aplikacija koja provjera mogu li se komponente složiti u sklop zahtjeva bitno vecu tocnost od aplikacije koja se koristi pri pakiranju.
- **Postoje složena pravila za instanciranje podataka.** Npr., ako je proizvod definiran kao sklop, mora imati više od jedne komponente, a svaka od komponenti mora imati barem jedan uvjet spajanja koji odgovara uvjetu spajanja druge komponente.
- **Nužno je osigurati cjelovitost podataka odgovarajucim algoritmima**. Npr., ako se obriše jedna od komponenti sklopa, struktura sklopa se mora prilagoditi.
- Jedna koncepcija prikaza može se na detaljiziranoj razini prikazati na više nacina. Npr., prikaz oblika može biti žicani model, rasterska slika ili solid model.

4.3 Proces konstruiranja i metode planiranja u umjetnoj inteligenciji

Istraživanja primjene spoznaja i metoda Umjetne inteligencije u konstruiranju pocela su otprilike prije dvadesetak godina. Nakon pocetnih rezultata istraživanja su se pocela razvijati u tri medusobno zavisna smjera [82]:

1. Razvoj tehnika prikaza inženjerskog znanja:
   - "blackboard sustavi"
   - "production rule" sustavi (ekspertni sustavi)
   - strukturirani objektni prikazi
2. Razvoj "boljih" modela procesa konstruiranja.
3. Razvoj arhitektura za integraciju razlicitih sustava za podršku konstruiranju.

Prema [82] modeliranje procesa konstruiranja još je uvijek nedovoljno rasvijetljeno podrucje. Nije još dovoljno dobro spoznato kako treba upotrijebiti racunalne sustave u podršci svim aktivnostima konstruktora (a ne samo geometrijskom modeliranju). Rad na modelima procesa konstruiranja razmatra spoznajne procese konstruktora dok radi, te vrste, izvore i manipulaciju znanjem. Ova istraživanja otvorila su treci pravac - kako integrirati razlicite vrste sustava rezoniranja i kako ih automatski kontrolirati i omoguciti efikasnu komunikaciju sa "ljudskim" korisnicima. Svrha je tih istraživanja spoznati (razumjeti) kako postici efikasno povezivanje ljudske i umjetne inteligencije u procesu konstruiranja.

Postavimo pitanje kako konstruktor organizira svoj rad, odnosno redoslijed izvodenja pojedinacnih aktivnosti? Ovom temom danas se bave i multidisciplinarni timovi, u koje su ukljuceni i psiholozi [83], [84], [85], [86]. Rezultati jedne od takvih studija prikazani su u

[87]. Rana istraživanja zakljucuju da se aktivnost konstruiranja hijerarhijski organizira, odnosno slijedi unaprijed postavljeni plan. Konstruktori svoje aktivnosti opisuju planovima, ali pokazalo se da je njihova stvarna aktivnost organizirana više oportunisticki.

Pretpostavlja se da primjena razvijenih metoda planiranja (u podrucju umjetne inteligencije) može pridonijeti približenju racunalnog modela realnom procesu konstruiranja. Stoga je u ovom poglavlju dan sažeti prikaz dosadašnjih istraživanja i dostignuca u dijelu podrucja Umjetne inteligencije koje se bavi razvojem metoda i algoritama za racunalsku podršku planiranju. Prikaz je dan prema kompilacijskom prirucniku [88]. Trebalo bi napomenuti da se te metode uglavnom primjenjuju u robotici, te u planiranju i vodenju tehnoloških procesa. Svrha je ovog prikaza i razmatranja iznaci smjernice za eventualnu implementaciju razvijenih metoda i spoznaja u modelu racunalne podrške planiranju i izvodenju konstrukcijskog procesa.
U svakodnevnoj terminologiji planiranje znaci odlucivanje o tijeku djelovanja, prije samog djelovanja (akcije). Za postavljeni (kompletirani) plan može se reci da je to uredena linearna lista operatora koji vode k rješenju problema, medutim ciljevi koji se dostižu primjenom operatora cesto imaju hijerarhijsku strukturu. Taj aspekt strukture planova potakao je jednu od najranijih definicija plana prema [89]:

Plan je svaki hijerarhijski proces u organizmu koji može kontrolirati redoslijed po kojem treba izvoditi operacije.

4.3.1 Pristupi planiranju

Pregled dostignuca i diskusija prema [88] orijentirani su na temu “planiranje i rješavanje problema”, s osvrtom na koristi koje donosi primjena planiranja. Kako su ta istraživanja opce prirode (neovisna o domeni problema), može se ocekivati da se odredene spoznaje i metode mogu primjeniti i u domeni planiranja rješavanja zadatka (i/ili problema) u procesu konstruiranja.

Prema [88] mogu se razluciti cetiri razlicita pristupa planiranju:

- **Nehijerarhijski**: generira slijed akcija kojima se postižu ciljevi. Ciljevi se mogu reducirati na jednostavnije ili se može upotrijebiti postupak analize u svrhu reduciranja razlike trenutnog i željenog (ciljnog) stanja prostora planiranja
- **Hijerarhijski:** prvo se "skicira" kompletan, ali djelomicno neodreden plan, koji se zatim razraduje do kompletnog slijeda detaljiziranih operatora rješavanja problema. Hijerarhijsko planiranje ukljucuje definiranje i planiranje u jednom ili više prostora apstrakcije. Plan se prvo generira u najvišem prostoru apstrakcije. Dobiveni “kostur” plana detaljizira se dalje na nižim nivoima apstrakcije
- **"Script based"**: koriste se vec postojece skice plana koje se pozivaju iz baze. Apstraktni koraci u planu popunjavaju se operatorima rješavanja za konkretni problem. Ovakav pristup zahtijeva velike kolicine specijalistickog znanja koje treba koristiti na raznim razinama opcenitosti (apstrakcije), dok se ne nadu odgovarajuci operatori za svaki korak odabrane skice plana.
- **Oportunisticki** pristup razvili su supružnici Hayes-Roth [90] da bi modelirali ljudske sposobnosti planiranja. U tu svrhu upotrijebili su adaptiranu "blackboard" kontrolnu strukturu. "Blackboard" je "mjesto za rašciščavanje" ("clearinghouse") sugestija o koracima (dijelovima) plana, koje daju programi - "specijalisti" za odredene vrste odlucivanja i postupaka u planiranju. Kontrola procesa u "blackboard" modelu je

asinhrona i oportunisticka: programi, odnosno komponente sustava postavljaju hipoteze bez odredenog redoslijeda, i upotrebljavaju hipoteze ostavljene od drugih programa ako im je to od koristi.

Na nacin slican oportunistickom pristupu, zapravo covjek planira u "svakodnevnom životu" - plan ne nastaje odjednom, "iz jednog komada", nego se formiraju "otoci" akcija koji se medusobno povezuju kad se za to ukaže prilika.
Oportunisticki pristup sadrži i "bottom up" komponentu, zbog mogucnosti implementiranja detaljiziranih akcija u plan koji se razvija. To je u suprotnosti sa "top-down" procesom razrade koji karakterizira hijerarhijsko planiranje, gdje se odluke na nižim (detaljnim) nivoima ne donose do zadnjeg moguceg trenutka u tijeku razvoja plana. Slijedeca bitna razlika ovog i ostala tri pristupa je u tome da može razviti "otoke" - dijelove plana nezavisno, dok hijerarhijski pristupi nastoje razviti cijeli plan na svakom od nivoa apstrakcije.

U svakom slucaju, ljudske sposobnosti planiranja daleko su sofisticiranije od bilo kojeg od navedenih pristupa planiranju u podrucju Umjetne inteligencije, ali im se oportunisticki pristup kao model najviše približava.

4.3.2 Pretraživanje i problem interakcije podproblema

Potrebno je još razmotriti dva problema koja su usko vezana uz temu poglavlja o metodama i tehnikama planiranja. To je problem ogranicavanja pretraživanja i problem interakcije podproblema.
Problemi s ogranicenjem pretraživanja nastaju kad treba pronaci ispravan redoslijed i korake koji dovode do rješenja, a postoji ogroman broj mogucih putanja od kojih vecina ne vodi k cilju. Tada nastaje tzv. kombinatoricka eksplozija jer broj kombinacija operatora raste eksponencijalno s brojem operatora.
Problem interakcije podproblema javlja se uvijek kad problem ima "spregnute (vezane) ciljeve", tj. više od jednog uvjeta koji mora biti zadovoljen. Poredak u kojem spregnuti ciljevi trebaju biti postignuti obicno nije definiran problemom, ali može biti kritican za pronalaženje rješenja. Može se reci da je to vrlo cesto ili gotovo uvijek vrijedi u rješavanju konstrukcijskih problema, ako ne za nalaženje jednog od mogucih rješenja, ali onda svakako za pronalaženje rješenja što bliže optimalnom.
Ponekad je zbog interakcije podproblema i nemoguce doci do rješenja, npr. ukoliko se spregnuti ciljevi u odredenom redoslijedu medusobno iskljucuju. Nažalost kod kompleksnijih problema ta vrsta informacija se na pocetku rješavanja ne mora odmah nazirati, nego se do njih dolazi zakljucivanjem tek u tijeku procesa rješavanja.
Problemi pretraživanja i interakcije podproblema su povezani jer dodatno pretraživanje u prostoru rješenja rezultira iz preuranjenog postavljanja proizvoljnog redoslijeda podciljeva.
Ako je redoslijed operatora neispravan (to se zakljucuje tek kad dode do interakcije podproblema), potrebno se vratiti nazad do tocke u planu koja je uzrokovala grešku i promijeniti plan - što je zapravo daljnje pretraživanje u prostoru rješavanja. Ukoliko je plan inicijalno loše postavljen i kao takav zahtijeva dosta vracanja i korekcija, to može uzrokovati dodatne troškove. Za interakcije podproblema u podrucju planiranja koristi se i naziv odnosno pojam ogranicenja. O ogranicenjima možemo izvesti zakljucke i iz pocetnih uvjeta problema ako su dani eksplicitno.

4.3.3 Okolina (prostor) izvodenja plana

U literaturi je objavljen veliki broj radova o kreiranju planova, dok se relativno malo pažnje posvecuje okolini u kojoj se planovi koriste. Najcešce su to za (odredeni) plan standardne okoline. Ovdje pojam standardne okoline treba prvenstveno shvatiti kao staticne, u smislu nepromjenjivosti situacije u kojoj se plan eksploatira.
Vec je receno da se plan sastoji od niza akcija. Ciljevi su odredeni za svaku akciju, i to svaku zasebno, jednu po jednu. Eksploatacija tako odredenog plana podrazumijeva da se iz stanja zadovoljenih inicijalnih uvjeta, za zadani cilj izvodi plan koji ce rezultirati stanjem zadovoljenog cilja. Pri tome okolina ne djeluje na izvodenje plana. Samo u planu unaprijed predvidene situacije odreduju nacin izvodenja odabranog plana. Plan se mijenja (bira ili odreduje alternativni) samo u situacijama kada postoje smetnje koje onemogucuju izvodenje trenutnog plana. Smetnje su planom nepredvidene situacije, tj. "vanjski poremecaji". Smetnje koje tako nastaju tretiraju se u upravljackim modulima za izvodenje plana kao neocekivane situacije, tj. slucajevi kada situacije nisu onakve kakve bi trebale biti. Pri tome upravljacki moduli realiziranih sustava za izvodenje plana ne razmatraju logicki karakter smetnji, nego samo konstatiraju da stanje okoline ne odgovara predvidenom.
Ovakva vrsta plana koji se eksploatira u gotovo statickim uvjetima ne odgovara osnovnim zahtjevima procesa konstruiranja. Realni proces konstruiranja se mora tretirati kao dinamicka pojava. Situacije se tijekom odvijanja procesa mogu mijenjati. Pri tome uzrok promjene ne mora biti izazvan iskljucivo odvijanjem neke od akcija plana. Takoder razlozi za promjenu plana (pa i prekid izvodenja) mogu biti uzrokovani nekom od akcija. Dinamicke promjene u procesu konstruiranja mogu se svrstati u interne i eksterne. Pod internim promjenama podrazumijevamo promjene koje su uzrokovane akcijama plana konstruiranja. Eksternim promjenama nazivamo promjene bilo kojih parametara procesa konstruiranja koje nisu uzrokovane planom. Redoslijed izvodenja planiranih akcija odnosno promjena plana uzrokovani su dakle promjenama situacije. Dinamicke promjene okoline (u skladu s prijašnjim izlaganjima to su promjene ocekivane situacije pri izvodenju plana) tijekom konstruiranja stalno su prisutne. Naime, konvencionalne akcije procesa konstruiranja su tako strukturirane da efekti pojedine akcije vrlo rijetko imaju utjecaj na samo jednu, slijednu akciju. Cešce efekti pojedine akcije uticu na veci broj parametara konstrukcije, koji su preduvjeti ostalih akcija pa tako u opcem slucaju mogu utjecati i na promjenu situacije.
Kao poseban problem prikaza konstrukcijskog procesa planom predstavlja se recepcija (, a potom i zadovoljenje) razlicitih spregnutih ciljeva (zahtjeva u konstrukcijskoj terminologiji) koji mogu proizaci tijekom rada. Drugim rijecima radi se o interakciji podproblema (razmatranoj u prethodnom poglavlju). Takovi se ciljevi ne moraju pojavljivati u "lijepom" sekvencijalnom nacinu, što uzrokuje potrebu za odredivanjem redoslijeda i prioriteta rješavanja ciljeva. Takvi slucajevi predstavljaju dinamicku promjenu situacije koja nije izazvana vanjskom smetnjom.
Medutim, (kao da i do sada navedeno nije dovoljno !) realne okoline mogu se mijenjati i na taj nacin da ne "ruše" izvodeni plan, nego da naprotiv, [53] nude nove mogucnosti izvodenja akcija, odnosno rješavanja.

Ako se osvrnemo na sve navedene probleme, može se postaviti pitanje da li postavljanje i korištenje plana u djelatnosti kao što je konstruiranje uopce ima smisla. Mišljenje je autora ovog rada da korištenje plana u svakom slucaju može prvenstveno poslužiti kao temelj integracije toka podataka izmedu raznorodnih aplikacija i kao temelj unapredenja organizacije procesa konstruiranja. Efekti primjene bili bi znacajniji u okolinama gdje

prevladava timski rad i gdje se uvodi paralelno inženjerstvo (concurrent engineering). Pri tome se "plan" tretira kao metoda prikaza tijeka procesa koja ima samo ogranicene mogucnosti rješavanja nepredvidenih situacija.
Takva upotreba plana trebala bi za odredene klase zadataka konstruktora osloboditi jednog dijela rutinskih aktivnosti, odnosno omoguciti mu da efikasnije i sistematicnije koristi raspoložive programske alate (ukljucujuci i konvencionalne CAD sustave). Na temelju programskog okruženja za eksploataciju definiranih "prototipnih" planova mogli bi se dalje graditi "inteligentniji" alati kojima bi se barem djelomicno (za odredene klase i domene zadataka) rješavali problemi razmatrani u ovom poglavlju.
Uz prije navedeno još ce se naglasiti da koncepcija sustava nece biti postavljena tako da se planovi kreiraju racunalnim modelima (odnosno "automatski"), nego je zamišljanje i kontrola procesa generiranja plana u potpunosti prepuštena covjeku cije su sposobnosti planiranja daleko sofisticiranije. To znaci da bi tako generirani planovi (ovisno o iskustvu i umješnosti konstruktora) trebali djelomicno izbjeci ili barem ublažiti navedene probleme dinamickog prostora izvodenja. Pri tome alat za generiranje plana konstruktoru treba poslužiti kao sredstvo za efikasnu obradu velikih kolicina podataka i kao sredstvo koje ce u tijeku procesa generiranja u najvecoj mogucoj mjeri otklanjati, odnosno sprecavati moguce greške.

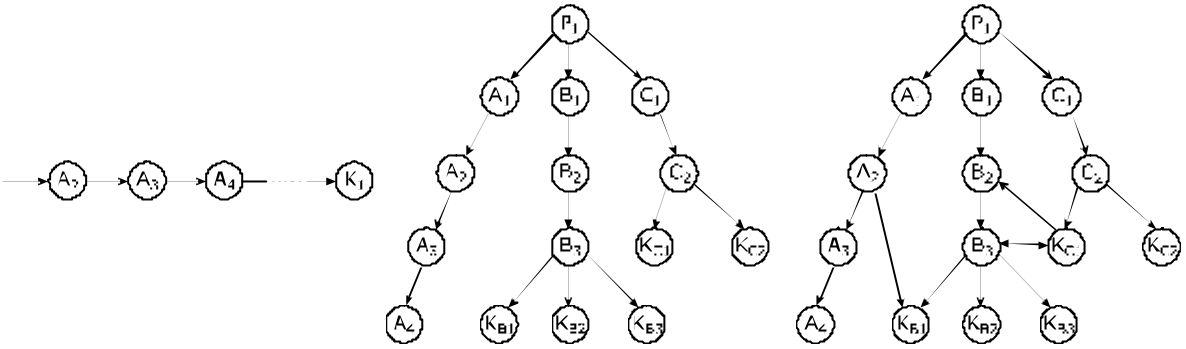
4.4 Topologija prikaza akcija u procesu konstruiranja

U prethodnim izlaganjima razmotrene su znacajke procesa konstruiranja i navedene neke smjernice modeliranja racunalne podrške procesu konstruiranja. Ovdje cemo ponovo naglasiti da u podrucju Znanosti o konstruiranju ne postoji usaglašena metoda prikaza tijeka procesa konstruiranja.
Proces konstruiranja promatran kao rješavanje zadatka i/ili problema može se prikazati prostorom stanja, [46], [91], gdje svako od mogucih stanja odgovara ili parcijalnom ili konacnom rješenju. Na taj nacin konstruiranje možemo promatrati kao niz akcija koje vode od inicijalnog stanja (definicije zadatka) do konacnog stanja (rješenja). U prethodnim razmatranjima zakljuceno je da sustav za racunalnu podršku planiranju i izvodenju procesa konstruiranja treba modelirati tako da ne ovisi o vrsti konstrukcijskog zadatka, a isto tako niti o fazi procesa konstruiranja u kojoj se primjenjuje. Ponovo cemo napomenuti da nije nužno koncipirati model sustava s ocekivanjima da bi se trebao koristiti na višim razinama apstrakcije zadatka, odnosno prostor stanja u kojem se izvode akcije može biti neki od mogucih podskupova cjelokupnog skupa informacija o proizvodu. Tako poopceni proces može se opisati kao djelatnost koja se sastoji od niza akcija i njihovih medusobnih relacija. Svaka od akcija vrši pretvorbu informacija, tj. na temelju skupa ulaznih informacija generira skup izlaznih informacija. Akcije su etape unutar procesa, a granicna stanja etapa (odnosno medustanja) možemo zamisliti kao prostor stanja procesa.

Topologiju tog prostora (odnosno topologiju akcija) možemo prikazati na najmanje tri nacina:

- linearnim prostorom (tj. etape su medusobno linearno povezane), što znaci da je tijek procesa iskljucivo linearan (slika 13a)
- hijerarhijskim stablom, što znaci da u procesu konstruiranja postoje grananja, ali da promjene u jednom cvoru ne uticu na stanje cvorova drugih grana (slika 13b)
- mrežom - koja je opcenitiji prikaz od stabla, a u stvari omogucuje bolji opis strukture i tijeka procesa konstruiranja jer pojedini cvor može biti referenciran iz

više prethodnika, a mogu se i definirati medurelacije izmedu cvorova razlicitih grana (slika 13c).



a (linearna) b (hijerar. stablo) c (mreža)

Slika 13: Tri topologije prikaza akcija u procesu konstruiranja

Sve tri razmatrane topologije akcija prikazane su usmjerenim grafovima.

Definicija grafa prema [92]:

Graf *G* sadrži dva konacna skupa: skup tocaka *V*, koje nazivamo cvorovima, i skup linija povezivanja *E*, koje nazivamo bridovima. Pri tome svaki brid povezuje dva cvora.

$$G = (V, E)$$

Definicija usmjerenog grafa prema [92]:

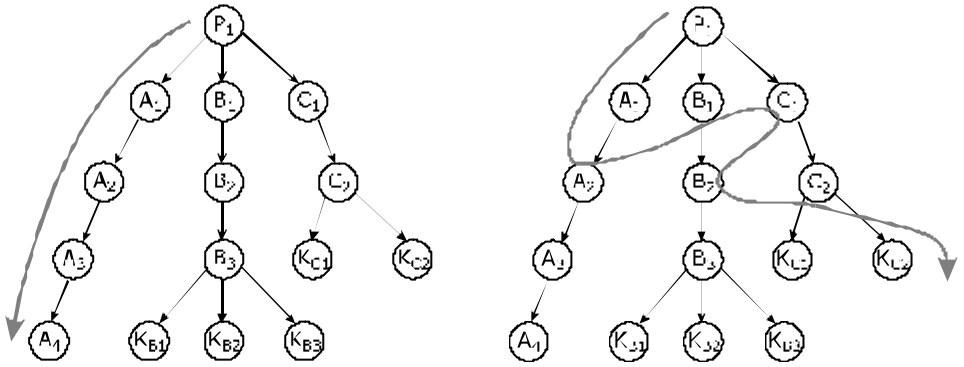
Usmjereni graf $G = (V, E)$ je graf u kojem svaki brid e = (i,j) ima smjer od "inicijalne tocke" (cvora) do "terminalne tocke" (cvora).
Pod uvjetom da su suprotnih smjerova, u usmjerenom grafu mogu postojati dva brida koja povezuju iste cvorove.

Ako koristimo usmjereni graf kao prikaz topologije akcija u procesu konstruiranja, treba razjasniti interpretaciju takvog prikaza. Za sada cemo pretpostaviti da prikaz procesa konstruiranja uvijek treba imati samo jedan "pocetni" cvor kojeg stavljamo na vrh grafa. (Opci usmjereni graf ne mora imati jedan "pocetni" cvor.) Linearna lista, hijerarhijsko stablo i mreža sa jednim pocetnim cvorom samo su specijalne vrste opceg usmjerenog grafa.
Hijerarhijsku mrežu promatramo kao prikaz koji se najviše može približiti realnim znacajkama procesa konstruiranja. Na vrhu te mreže su zahtjevi (specifikacije koje cine definiciju zadatka), a krajnji cvorovi predstavljaju akcije nakon cijeg izvršavanja stanje u prostoru procesa odgovara nekom od mogucih stanja rješenja. Meducvorovi predstavljaju medustanja u tijeku odvijanja procesa. Bridovi grafa prikazuju moguce putanje redoslijeda izvršavanja akcija i tokove podataka kroz proces. Bridovi se mogu interpretirati i kao prikaz medusobnog utjecaja podskupova informacija, naime promjena jednog medustanja (u jednom cvoru) može utjecati na podskupove informacija (atributa) drugih cvorova.

Sustav za podršku procesu konstruiranja treba koncipirati na slijedeci nacin:

- konstruktoru treba omoguciti lagano manevriranje izmedu cvorova mreže
- u tijeku izvodenja procesa treba omoguciti dodatnu obradu (od strane konstruktora) skupova informacija u svakom cvoru
- treba osigurati uvid u trenutno stanje unutar prostora stanja rješavanja zadatka

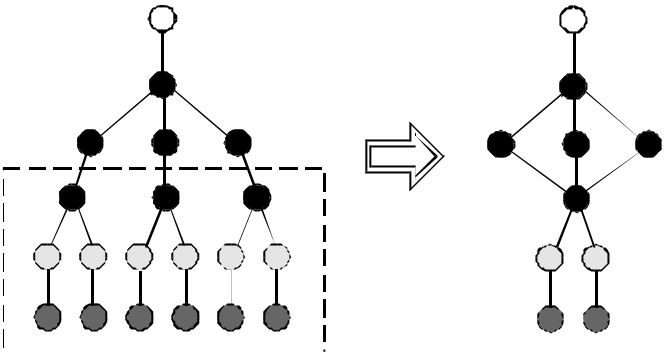
Prethodna istraživanja unutar projekta "Model inteligentnog CAE sustava" [7], [5], [93], oslanjala su se na prikaz procesa konstruiranja u obliku hijerarhijskog stabla. Medutim, iako je topologija prostora stanja konstrukcijskog procesa zapisana u obliku stabla, u tijeku izvodenja programskog sustava omoguceno je da se takav zapis eksploatira u obliku hijerarhijske mreže (slika 14), jer konstruktor može aktivirati cvorove i nepredvidenim redoslijedom (ne poštujuci relacije sljednik-prethodnik). U tom slucaju program za kontrolu izvodenja plana procesa konstruiranja ne obraduje informacije temeljem medurelacija izmedu cvorova, pa konstruktor mora sam obaviti takvu eventualnu dodatnu obradu.
Ovakav nacin modeliranja procesa konstruiranja odabran je jer je u pocetnoj fazi razvoja sustava zakljuceno da su algoritmi kontrole hijerarhijske mreže isuviše složeni da bi se mogli odmah implementirati. Stoga je nastavak istraživanja u ovom radu usmjeren na objektno orijentiranu metodologiju modeliranja, jer se pretpostavlja da ce njene prednosti omoguciti realizaciju, nadogradnju i održavanje mrežnog modela.



Slika 14: Izvodenje po predvidenim i nepredvidenim putanjama

Razmotrimo još mogucnosti prikaza iterativnosti procesa konstruiranja i uplive nacina odlucivanja na modeliranje prikaza procesa. Iterativnost i strategiju odlucivanja možemo naznaciti kao znacajke procesa rješavanja konstruktorskog zadatka koje u velikoj mjeri ovise o odabranoj metodi rješavanja zadatka, kao i o znanju, iskustvu i individualnim sposobnostima konstruktora.
Proces iteracije nema smisla prikazivati stablom, jer se ne može unaprijed odrediti broj ponavljanja. Iteracija se može simulirati prekidanim nacinom rada, ali tada treba pri kreiranju plana voditi racuna o skupovima ulaznih i izlaznih atributa cvorova, odnosno treba ih strukturirati prema ocekivanoj putanji kretanja kroz cvorove u tijeku iteracije. Ukoliko se u tijeku izvodenja pokaže potreba za kretanjem nepredvidenim putanjama, vrlo je vjerojatno da ce se pojavljivati situacije u kojima nisu odredene vrijednosti svih potrebnih ulaznih atributa cvorova, što ce dodatno usporavati proces izvodenja.
Odluke koje se donose u cvorovima s više od jednog sljedbenika imaju najveci upliv na strukturu prikaza procesa. Odluke mogu biti npr. tipa “uvjet zadovoljen” ili “uvjet nije zadovoljen” - koje bi generirale strukturu binarnog stabla. Odluke tipa izbora izmedu više mogucnosti mogu uzrokovati divergiranje stabla plana u širinu. Naime, nakon izbora izmedu npr. mogucih varijanti nekog dijela ili sklopa, daljnji proces može biti potpuno jednak, neovisno o odabranoj varijanti. To znaci da bi se isti skupovi cvorova ponavljali u svakoj grani, a ako bi bilo više cvorova s takovim odlukama, prikaz (plan procesa) bi divergirao u širinu i postao bi glomazan, s velikom redundancijom podataka (slika 15). To odmah implicira i više mogucnosti za pravljenje semantickih grešaka, nepreglednost i teže održavanje plana, duži proces prevodenja i vece zahtjeve za memorijom u tijeku izvodenja, te prostorom za spremanje zapisa plana na disku.

S druge pak strane, ako se vecina ili sva "grananja" i odlucivanja obavljaju unutar operacija odredenog cvora, prikaz procesa tada poprima strukturu linearne liste.



Slika 15: Redundantni zapisi u hijerarhijskom stablu

Prethodna razmatranja doprinose zakljucku da je nužno razviti model prikaza procesa u obliku mrežne topologije [94], [95]. Korištenjem objektne metodologije svaki cvor i svaki brid grafa bi trebalo modelirati kao objekte. Enkapsulacija svojstava i ponašanja (operacija), te klasifikacija cvorova i bridova trebala bi omoguciti realizaciju algoritama za kontrolu procesa prikazanog mrežnom topologijom.

4.4.1 Matricni prikaz grafova

Cvorovi grafa obicno se oznacavaju s v_1, v_2, …. ili jednostavno brojevima, a bridovi s e_1, e_2, …. ili s njihove dvije krajnje tocke (cvora) npr: e_1= (1,4), e_2=(1,2)
Brid (v_i, v_j) je svojstven (upadan, zavisan, eng. "incident") cvoru v_i (vrhu); isto vrijedi za v_j.
Broj bridova svojstvenih cvoru v naziva se stupanj cvora v.

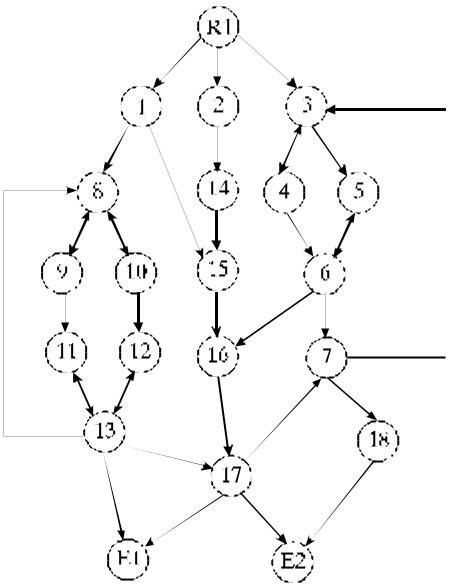
Dva cvora nazivamo susjednim (eng. "adjacent") cvorovima grafa ako su povezani bridom, odnosno ako cine dvije krajnje tocke brida.
Za racunalni prikaz grafova najpogodnije je koristiti matrice, pa slijedi prikaz nekoliko oblika matricnih prikaza grafova.

Definicija matrice susjedstva (adjacency matrix) grafa G:

$$A = [\, a_{ij} \,]$$

$$a_{ij} = \begin{cases} 1 & \text{ako } G \text{ ima brid } (i, j) \\ 0 & \text{ako nema brida} \end{cases}$$

Matrica susjedstva opceg grafa je simetricna. Za prikaz procesa konstruiranja svakako je prikladnije koristiti usmjereni graf. Element matrice susjedstva usmjerenog grafa, a_{ij} = 1, onda kad postoji usmjereni brid (od cvora i prema cvoru j). Matrica susjedstva usmjerenog grafa nije simetricna. Na slici 16 prikazan je primjer usmjerenog grafa i pocetni dio pripadajuce matrice susjedstva. Radi bolje preglednosti nule su izostavljene.



od cvora: \ prema cvoru:

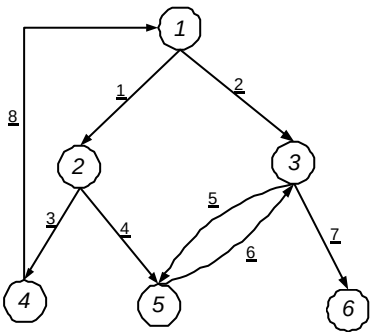
	R1	1	2	3	4	5	6	7	8	...
R1		1	1	1						
1									1	
2	1									
3					1	1				
4				1			1			
5							1			
6						1		1		
7				1						
8										
....										

Slika 16: Usmjereni graf i dio njegove matrice susjedstva

Definicija matrice incidencije grafa *G*:

$$A = [a_{jk}]$$

$$a_{jk} = \left\{ \begin{array}{l} +1 \text{ ako grana } k \text{ izlazi iz cvora } j \\ -1 \text{ ako grana } k \text{ ulazi u cvor } j \\ \quad 0 \text{ ako grana } k \text{ ne dira cvor } j \end{array} \right\}$$



GRANA: \ CVOR:

	1	2	3	4	5	6	7	8
1	1	1						-1
2	-1		1	1				
3		-1			1	-1	1	
4			-1					1
5				-1	-1	1		
6							-1	
....								

Slika 17: Primjer usmjerenog grafa i pripadajuce matrice incidencije

Matrica susjedstva opcenito je mnogo manja od matrice incidencije, pa se stoga pretežno upotrebljava za racunalni prikaz i spremanje grafa. Maksimalni moguci broj bridova u grafu iznosi*: n(n-1)/2*, gdje je *n* broj cvorova grafa.

Matricni prikazi nisu efikasni za grafove koji imaju mali broj bridova u odnosu na maksimalni moguci broj bridova. U takvim slucajevima obicno se koriste liste cvorova i liste bridova s kojima se lakše i brže manupilira nego s matricama. Primjer takvih listi prikazan je u tablici 1.

cvor	vezani bridovi
1	1, 2, 8
2	1, 3, 4
3	2, 5, 6, 7
4	3, 8
5	4, 5, 6
6	7

brid	krajnje tocke
1	1, 2
2	1, 3
3	2, 4
4	2, 5
5	3, 5
6	5, 3
7	3, 6
8	4, 1

Tablica 1: Liste cvorova i bridova primjera grafa sa slike 17

4.5 Metode prikaza procesa

U prethodnim razmatranjima topologije prikaza akcija u procesu konstruiranja, za sam prikaz procesa korišten je graf, odnosno cvorovi i njihove veze (bridovi). Opceniti graf ima mrežnu strukturu, dok su linearna lista i hijerarhijsko stablo samo specijalni oblici (slucajevi). Opci grafovi koriste se kao metode prikaza u raznim podrucjima - ekonomiji, za prikaz elektricnih mreža, molekularnih struktura, organizacijskih struktura, itd. Osim grafova, razvijeno je još mnogo metoda prikaza procesa (od kojih se neke temelje na grafovima).

Metode prikaza procesa (opcenite i specificne namjene) predmet su mnogih istraživanja. Americki nacionalni institut za standarde i tehnologiju (NIST) nastoji razviti opceniti jezik za prikaz procesa (Unified process specification language) koji bi trebao postati standardom. U tom projektu krenuli su s analizom prednosti i nedostataka postojecih metoda prikaza procesa. Zakljuceno je da niti jedna od postojecih metoda ne može ispuniti sve zahtjeve opcenitog prikaza. Stoga nastoje analizirati prednosti pojedinih metoda u odredenim domenama, te kombiniranjem karakteristika i elemenata više metoda doci do osnovnih koncepata opcenitog jezika za prikaz procesa.

Prema [12] analizirano je 26 metoda prikaza procesa, odnosno njihova primjenjivost za odredeni skup zahtjeva. Analiza je takoder pokazala da se neke osnovne metode i elementi prikaza koriste u svim analiziranim prikazima procesa. U dosta slucajeva analiziranih 26 metoda koriste se kombinacije pet osnovnih metoda prikaza:

- AND/OR grafovi
- dijagrami toka podataka (data flow diagrams)
- usmjereni grafovi (directed graphs)
- dijagrami tranzicije stanja (state transition diagrams)
- strukture stabla

Druga faza analize u projektu prema [12] ukljucuje odredivanje koje metode su pogodne za odredene vrste zahtjeva pri prikazu procesa. Razradena je klasifikacija na temelju znacajki koje opisuju prikaze. Navesti cemo neke od tih znacajki metoda prikaza procesa: - temeljene na ogranicenjima, temeljene na "EXPRESS-u", fokusirane na cilj, graficke, temeljene na grafovima, hijerarhijske, usredotocene na znanje, logicki orijentirane, objektno orijentirane, proceduralne, usredotocene na proces, usredotocene na stanja.

Na primjer za opis metode AP213 vrijede znacajke da je temeljena na "EXPRESS-u", usredotocena na proces, objektno orijentirana i tekstualna.
U slijedecoj fazi istraživanja prema [12] nastojati ce se kombiniranjem i spajanjem karaketristika i elemenata analiziranih postojecih metoda prikaza procesa doci do opcenitog modela koji bi se mogao standardizirati.
Iz razmatranja o metodama prikaza procesa može se zakljuciti da objektni model procesa konstruiranja treba koncipirati tako da u pocetnoj fazi sadrži jednu od metoda prikaza procesa, ali da treba ostati otvoren i za daljnje modifikacije te metode kao i za implementaciju drugih metoda.

5. Objektno orijentirani pristup modeliranju i programiranju

Temeljna je ideja (princip) objektno orijentiranog programiranja: modelirati aplikaciju kao skup objekata koji komuniciraju da bi postigli zajednicki cilj.

Važno je naglasiti da se objektno orijentirano programiranje ne bavi programiranjem u smislu razvoja algoritama i struktura podataka, nego ga treba promatrati kao skup sredstava za organiziranje programa, odnosno opcenitije kao tehnike za koncipiranje programa [96].

Osnovna sredstva strukturiranja programa su objekti. Objekti modeliraju entitete iz stvarnog svijeta, mogu obuhvacati apstrakcije kompleksnih fenomena ili mogu reprezentirati elemente programskog sustava (npr. stogove ili upravljanje grafickim prikazom). Operacijski gledano, objekti kontroliraju racunalni proces. Iz perspektive razvoja programa, najvažnija karakteristika objekata nije njihovo ponašanje kao takvo, nego cinjenica da se ponašanje objekta može opisati apstraktnom karakterizacijom njegova sucelja. Takva apstraktna karakterizacija dovoljna je za pocetno koncipiranje sustava. Stvarno ponašanje objekta može se implementirati i doraditi kasnije, prema potrebama.
Vjerojatno najvažniji doprinos objektne orijentacije programerskoj praksi je upotreba nasljedivanja pri odredivanju relacija izmedu objekata, odnosno klasa objekata. Nasljedivanje omogucuje inkrementalno dodavanje funkcionalnosti (specifikaciji). Na taj nacin osigurano je bolje konceptualno modeliranje - mogu se izvuci zajednicki dijelovi specifikacije i omogucena je ponovna upotrebljivost specifikacije. Ako se (disciplinirano) primjenjuje na odgovarajuci nacin, nasljedivanje omogucuje postupan razvoj specifikacije tipa klase objekata. Razliciti objekti razlicitih tipova mogu se promatrati kao elementi zajednickog "super" tipa.

5.1 Modeliranje objektima

Objektna orijentacija promatra racunalni program kao skup objekata, gdje svaki objekt modelira entitet ili dogadaj iz aplikacijskog problema (domene). Svi objekti rade (funkcioniraju) zajedno da bi postigli cilj zadatka postavljenog cjelokupnom sustavu. Središnji programski (softverski) koncept je "objekt". Objekt obuhvaca identitet, strukturu i ponašanje aplikacijskih entiteta koje reprezentira (modelira, prikazuje).

U stvarnom svijetu mnogi objekti su slicni - imaju zajednicka svojstva i ponašanje. Ipak, svaki objekt ima svoj identitet i svoje jednistvene vrijednosti (unutar) zajednickih svojstava. Unatoc jednistvenosti identiteta i vrijednosti svakog objekta, smislenije je opisivati objekte u grupama. Objektno orijentirani program opisuje objekte koji se pojavljuju u aplikaciji - to cini sa *klasama* cije instance su objekti. Dakle "objekt" je programski (softverski) koncept koji modelira aplikacijski entitet. "Klasa" je programski (softverski) koncept koji opisuje skup objekata.

Objekti se mogu usporediti sa varijablama u tradicionalnim programskim jezicima. Ipak, postoji znacajna razlika izmedu objekta i varijable. Varijabla obuhvaca samo "podatkovni" (informacijski) aspekt objekta (vrijednost), ali ne i ponašanje.
Kombinacija strukture podataka i deklaracije funkcija, zajedno sa sposobnošcu da iz sebe kreira instance razlicitih identiteta, naziva se klasom u objektno orijentiranim programskim jezicima. Klasa usko povezuje strukture podataka i njima pridružene procedure koje ih obraduju.

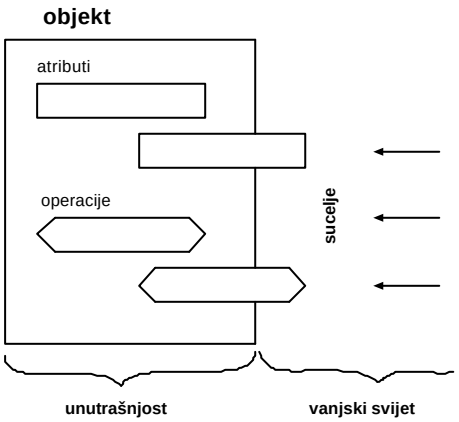
U objektno orijentiranom pristupu naglasak je na modeliranju stvarnosti u domeni problema umjesto stvaranja arhitekture modela sustava koja vodi k implementaciji.

Objektna tehnologija razvoja sustava koristi isti model kroz cijeli proces razvoja sustava:
- zapoceti sa objektno orijentiranom analizom
- konvertirati rezultate analize u objektne koncepte
- napisati objektno orijentirane programe

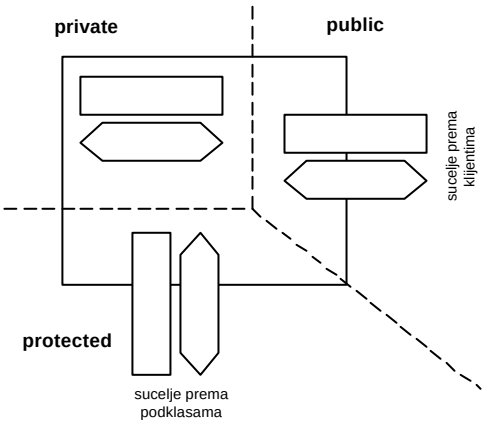
Dakle, možda najveca prednost objektne tehnologije je u konzistentnosti modela tijekom cijelog procesa razvoja programskog sustava. Što je sustav kompleksniji i veci, ta prednost više dolazi do izražaja.

5.2 Osnovna terminologija

Osnovni elementi objekata su atributi i operacije. Prema [97] ***atribut*** je informacijski detalj svojstven objektu. Ovisno o konkretnom programskom jeziku, atributi se nazivaju i varijablama ili svojstvenim poljima (member fields). ***Operacija*** je funkcionalni detalj svojstven objektu i spremljen kao dio objekta. Za operacije se koriste i nazivi svojstvena funkcija (member function) ili metoda. Pri modeliranju sustava korisno je prikazivati objekte i klase pomocu dijagrama. Dijagram objekta naglašava objekt kao nešto što ogranicava svoju "unutrašnjost" i komunicira sa "vanjskim svijetom" (slika 18).



Slika 18: Dijagram objekta prema [97]

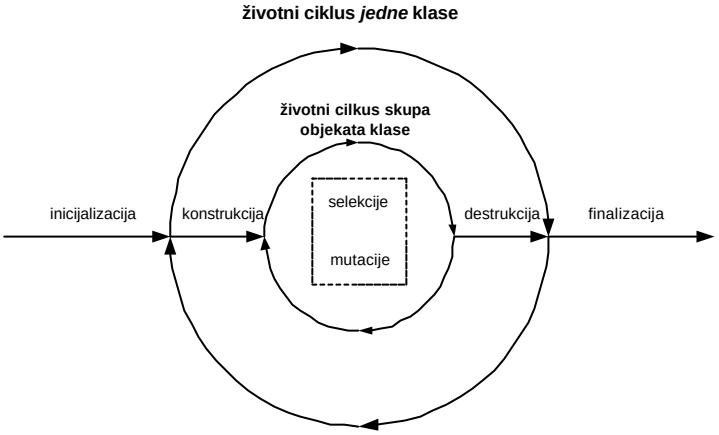


Slika 19: Zone pristupa objektu (prema [97])

Svi atributi i operacije koji su potpuno unutar objekta, skriveni su od vanjskog svijeta, odnosno oni su "ucahureni" (eng. encapsulated). Ostali objekti nemaju pristup do njih. Atributi i operacije koji djelomicno izlaze iz okvira objekta pristupacni su "vanjskom svijetu" i oni cine sucelje objekta. Razlikujemo dva nacina pristupa (slika 19) - pristup od

strane klijenata (objekata koji nisu hijerarhijski povezani s objektom kojem se pristupa) i pristup od strane podklasa iz hijerarhije klasa kojoj pripada objekt.
Prema [96] ***enkapsulacija*** promovira modularnost, tj. objekti se moraju promatrati kao blokovi za gradnju kompleksnog sustava. Kad se jednom dostigne odgovarajuca modularizacija, moguce je odgoditi konacne odluke o implementaciji. To svojstvo omogucuje brzu izradu prototipa modela. Enkapsulacija omogucuje separaciju procesa kreiranja klase na faze specifikacije i implementacije. Specifikaciju može izraditi specijalist za koncipiranje programskih sustava ("software designer"), a implementaciju programer.

Klasa opisuje skup entiteta koji imaju zajednicke temelje koncepta. Objekt je instanca klase, a klase služe kao predlošci za kreiranje objekata. Klase imaju životni ciklus koji dijele sa svojim objektima (slika 20).



Slika 20: Životni ciklus klase i njenih objekata prema [98]

Usporedba klasa sa konvencionalnim nacinima kategorizacije dovodi do izražaja pojam hijerarhije klasa (***nasljedivanje***). "Podklase" su zapravo proširenja i/ili specijalizacije postojecih klasa i nasljeduju njihova svojstva. Objektno orijentirani programski jezici upotrebljavaju klase za kategorizaciju entiteta koji se pojavljuju u aplikacijama. Kad je hijerarhija jednom postavljena, jednostavno se proširuje. Da bi se opisao novi koncept (entitet), nije nužno opisati sva njegova svojstva. Dovoljno je opisati samo razlike u odnosu na koncept unutar postojece hijerarhije.
Klasa opisuje strukture podataka i funkcionalnost svojih instanci. Nema uvijek potrebe da klasa ima instance. Klasa može poslužiti i kao osnovna klasa hijerarhije koja apstrahira zajednicka svojstva nekoliko deriviranih klasa. ***Apstraktna klasa*** je dakle klasa koja služi kao zajednicka osnovna klasa i nece imati instance.

Polimorfizam (viševrsnost) znaci imati sposobnost preuzeti mnogo varijanti oblika. U objektno orijentiranoj tehnologiji to se odnosi na mogucnost entiteta da se u tijeku izvodenja poveže sa instancama razlicitih klasa. U objektno orijentiranim jezicima mogu se razluciti dva oblika polimorfizma - inkluzijski i operacijski.
Polimorfizam, u kombinaciji sa dinamickim povezivanjem (dynamic binding) omogucava gradnju fleksibilnih sustava koje je lakše proširivati. ***Dinamicko povezivanje*** je mogucnost da se objekt poveže sa akcijom (operacijom) tek u tijeku izvodenja programa. Drugim rijecima dinamicko povezivanje odlaže povezivanje poziva funkcije i ciljnog bloka programskog koda do trenutka izvodenja programa.

Kompozitni objekt je agregacija drugih objekata, odnosno sadrži kompoziciju ili agregaciju drugih objekata kao podobjekata u svojoj implementaciji. Takvi podobjekti mogu biti instance klasa koje predstavljaju entitete, ili mogu i sami biti kompozitni objekti. Na taj nacin može se kreirati višerazinska hijerarhija sadržavanja (ukljucivanja) objekata. Kompozitni objekt upravlja postojanjem i meduzavisnostima svojih podobjekata, odnosno upravlja skupovima instanci podobjekata.

5.3 Razvoj metoda modeliranja objektno orijentiranih programskih sustava

Prvi problem s kojim se suocavaju programeri koji razvijaju sustave korištenjem objektnih tehnologija jest izbor odgovarajuce metodologije za proces razvoja. Od pocetaka razvoja objektnih jezika do devedesetih godina razvijeno je i korišteno mnogo razlicitih metodologija, svaka sa svojom varijantom notacije. Od pocetnih desetak došlo se do oko 50 metoda koje su korištene izmedu 1989. i 1994. Te metode neki autori [98] nazivaju metodama prve generacije. Takvo stanje bilo je na neki nacin kaoticno, jer niti jedna metoda nije prevladavala niti je mogla zadovoljiti sve zahtjeve iz prakse. Sve su te metode imale neke zajednicke koncepte, ali izražene na razlicite nacine. Takvo stanje cak je i odvracalo od upotrebe objektne tehnologije. Put prema spajanju i unifikaciji metoda poceo je sredinom devedesetih godina, kada se javljaju metode druge generacije.
U tom periodu razvija se nekoliko pravaca standardizacije, npr. CORBA, OPEN i UML. U isto vrijeme tri metode se izdvajaju kao najprominentnije: Booch-ova, OMT (Rumbaugh) i OOSE (Jacobson). Spomenuta tri autora spajaju svoje metode i zapocinju razvoj UML-a (Unified Modeling Language), ciju prvu verziju prikazuju 1995. OMG (Object Management Group) konzorcij za razvoj programskih sustava preuzima ulogu katalizatora za spajanje svih nastojanja u razvoju i standardizaciji UML-a. Okuplja se veliki broj tvrtki i iskusnih metodicara, te u srpnju 1997. nastaje prvi prijedlog standarda. Nakon slijedeceg koraka dorade, u studenom 1997. OMG grupa prihvaca UML kao standardni jezik modeliranja. Tako UML postaje temelj za *de facto* standard u domeni objektno orijentirane analize i koncipiranja [100]. To ne znaci da su sve ostale metode odmah istisnute, ali primjetan je trend sve vece upotrebe UML-a [98].

5.4 Unified Modeling Language (UML)

UML je, prema definiciji njegovih autora [101], jezik za vizualiziranje, specificiranje, konstruiranje i dokumentiranje rezultata procesa razvoja softvera kao i za modeliranje poslovnog sustava. UML takoder omogucuje pohranu, razmjenu i primjenu znanja u procesima rješavanja problema. UML ne propisuje nikakav odredeni pristup rješavanju problema, nego se može prilagoditi svakom pristupu.

Autori UML-a jasno razdvajaju jezik za modeliranje od razvojnog procesa. Iako ce se UML koristiti u sklopu definiranih procesa, pokazuje se da razlicite organizacije, razliciti tipovi projekata i razlicite problemske domene, traže i razlicite odgovarajuce razvojne procese. Primjerice, razvojni proces prikladan za tvrtku koja proizvodi programe za obradbu teksta za široko tržište ne može biti istovjetan razvojnom procesu za poznatog narucitelja, recimo, u zrakoplovnoj industriji. Medutim, jezik za objektno modeliranje može biti jedinstven.

Treba naglasiti i da UML nije samo notacija (nacin crtanja pojedinih dijagrama). To je skup koncepata u objektnom modeliranju. Kao i svaki jezik, UML ima definiranu sintaksu (ovdje je to graficka notacija i niz pravila vezanih uz dijagrame) i semantiku. Razvoj semantike jezika iziskivao je najviše napora, posebno u uskladivanju postojecih i uvodenju novih koncepata. Trebalo je definirati jezik koji je dovoljno bogat, a istodobno potpuno precizan. Semantika UML-a opisana je i metamodelom u samome UML-u.
UML je pogodan za modeliranje širokog spektra programskih sustava, npr. velikih poslovnih informacijskih sustava, distribuiranih Web aplikacija, pa i vrlo kompleksnih sustava realnog vremena.

5.5 Usporedba UML-a i EXPRESS-a

Da bi razjasnili neke moguce nedoumice, u ovoj glavi razmotriti cemo neke aspekte primjene UML-a i EXPRESS-a. EXPRESS, kao dio STEP standarda danas je jedan od prevladavajucih jezika modeliranja u podrucju informacijskih sustava za podršku proizvodnji. Moglo bi se stoga postaviti pitanje koji od ta dva jezika pogodnije koristiti pri izradi modela procesa konstruiranja? Koje su prednosti jednog ili drugog u odredenim situacijama i zašto?

EXPRESS [102] je dio kompleksnog sustava razvijenog za modeliranje i razmjenu podataka medu razlicitim sustavima. STEP (**ST**andard for the **E**xchange of **P**roduct Model Data) neformalno je ime koje se koristi za *ISO 10303 Industrial automation systems - Product data representation and exchange* standard kojim je opisana problematika oblikovanja i razmjene informacija [103]. Medutim, kako je STEP razvijan prvenstveno kao standard za **razmjenu** podataka o proizvodu, ne sadrži nikakve oblike modela procesa konstruiranja [104], [105].
Model procesa konstruiranja trebao bi moci implementirati razvijene informacijske strukture opisa proizvoda iz STEP standarda. Samo je po sebi jasno da informacijske strukture opisa proizvoda trebaju biti jedan od gradbenih elemenata modela procesa konstruiranja, jer su informacije o proizvodu zapravo produkt procesa konstruiranja. Postavlja se sada pitanje na koji nacin ostvariti takovu implementaciju (odnosno povezivanje), jer je sasvim vjerojatno da ce elementi STEP standarda uci u široku upotrebu u vecini proizvodnih sustava.
Vjerojatno najjednostavniji i najprirodniji nacin rješavanja tog problema je da se informacijske strukture iz STEP standarda tretiraju u modelu procesa konstruiranja kao objekti, odnosno komponente.

Navedimo i usporedimo neke znacajke EXPRESS-a i UML-a bitne za trenutne aspekte razmatranja.

EXPRESS je shema konceptualnog jezika, razvijen kao dio PDES/STEP projekta. Njime se specificiraju informacijski zahtjevi gotovo svih dijelova standarda. Služi odredivanju objekata koji pripadaju prostoru promatranja, informacijskih jedinica i pravila koja se odnose na te objekte.
EXPRESS je u osnovi "Entity-Relationship" orijentiran, no ukljucuje i neke elemente objektnih modela. Model entiteti-veze i relacijski model podataka uspješno se primjenjuju u poslovnim informacijskim sustavima u kojima su podaci razmjerno jednostavnijih tipova. U aplikacijama kao što je konstruiranje podržano racunalom javlja se potreba za

(semanticki) "bogatijim" tipovima podataka kojih u spomenutim modelima nema. Za razliku od objekata entiteti nemaju opis ponašanja u samom modelu entiteti-veze.
Dakle objekt opcenito odgovara pojavi entiteta modela entiteti-veze, osim što dodatno sadrži i opis svog ponašanja. Klasa odgovara tipu entiteta, osim što sadrži i opis ponašanja objekata klase.

I UML i EXPRESS su samo deklarativni, ali ne i izvršni jezici, pa se prema tome mogu koristiti samo za modeliranje. Izvršni kod se može dobiti generatorima koda, medutim UML sadrži gotovo sve potrebne specifikacije za izvršni kod vec u samom modelu. Pored toga struktura UML-a odgovara jezicima u koje se prevodi, dok struktura EXPRESS-a nema puno zajednickog s npr. C programskim jezikom. Dakle pri transformaciji EXPRESS-a dolazi do semanticke razlike i dobiveni programski kod je teško povezati sa izvornim modelom.

UML je semanticki daleko bogatiji jezik od EXPRESS-a. EXPRESS-G je graficka varijanta jezika i sadrži dijagrame, ali samo strukturalne, a ne i interakcijske i dinamicke. UML je semanticki bogatiji i od bilo kojeg postojeceg objektno orijentiranog jezika [101].

Iz iznesenog možemo zakljuciti da je UML svakako pogodniji i kompletniji jezik za modeliranje racunalnog sustava podrške procesu konstruiranja. Osnovna razlika ova dva jezika ocituje se u metodologijama. Kompleksnost procesa konstruiranja zahtijeva semanticki bogatiji jezik i prednosti objektne metodologije.

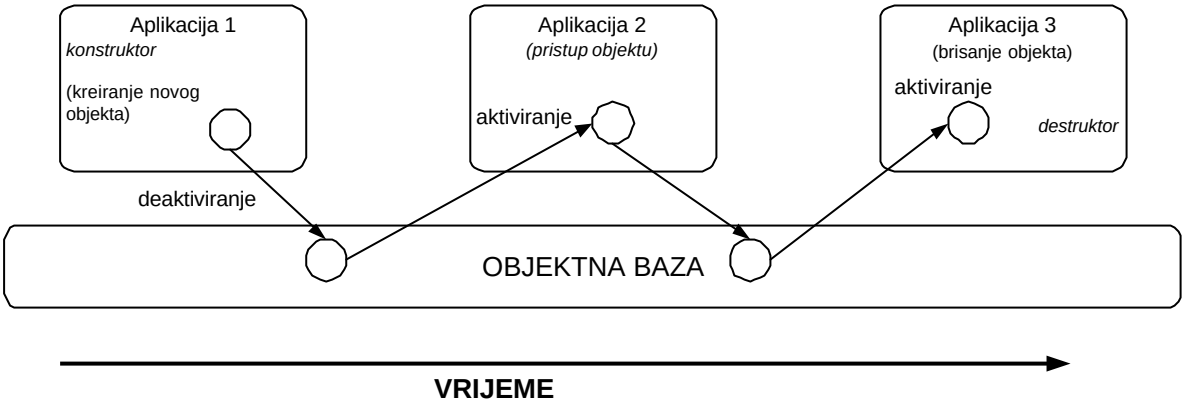
Medutim prednosti razvijenog standarda modeliranja informacija o proizvodu treba svakako iskoristiti u gradnji modela procesa konstruiranja. To znaci da se modeli informacijskih struktura razvijeni u EXPRESS-u trebaju implementirati u strukturu objektnog modela procesa konstruiranja. Zakljucimo dakle, da je moguce i potrebno kombinirati modeliranje pomocu oba jezika, s time da se strukture modelirane EXPRESS-om koriste kao podsustavi, odnosno parcijalni modeli (komponente ili podpaketi).

Odredeni dijelovi strukture STEP-a (aplikacijskih protokola i integriranih generickih resursa [104], [105]) mogu poslužiti i kao polazišta i uzorci za definiranje i koncipiranje gradbenih elemenata modela procesa konstruiranja.

5.6 Objektne baze podataka

Realno je pretpostaviti da ce neke od klasa objektnog modela procesa konstruiranja imati veliki broj instanci, odnosno objekata. Vjerojatno je takoder da ce u tijeku izvodenja racunalne podrške procesu konstruiranja prevladavati situacije u kojima vecina tih objekata nece biti aktivna, odnosno nece biti potrebe da budu stalno u memoriji. To pogotovo vrijedi za objekte koji nemaju nit kontrole ("thread") u aplikaciji. Vecina takvih pasivnih objekata mogla bi se ponovno iskoristiti i u slijedecim varijantama konstrukcije, odnosno novim zadacima. Permanentno spremanje objekata, odnosno korištenje objektne baze stoga se namece kao osnovna programska platforma implementacije objektnog modela procesa konstruiranja.
Objektna baza podataka kombinira semantiku objektno orijentiranog programskog jezika i mehanizme upravljanja i pretraživanja podataka konvencionalnih sustava baza podataka. Ako je objektna baza integrirana sa objektnim programskim jezikom, tada treba podržavati semantiku tog jezika - relacije postavljene u programu trebaju automatski biti prikazane u bazi kada se objekti spreme.

Objekti kojima se pristupa u transakcijama kopiraju se ili brišu iz memorijskog prostora aplikacije koja vrši transakciju. Te operacije nazivaju se aktiviranje i deaktiviranje objekata [99]. Slika 21 prikazuje proces aktiviranja i deaktiviranja objekta u tijeku njegova životnog ciklusa. Prva aplikacija kreira novi stalni (perzistentni) objekt. Kada aplikacija predaje nit kontrole, objekt se deaktivira i sprema u bazu. Druga aplikacija ucitava objekt, odnosno aktivira ga u memoriji. Na kraju transakcije objekt se opet deaktivira i zapisuje u bazu ako je bio modificiran. Treca aplikacija pristupa objektu, aktivira ga i briše. Pri tome se objekt ne deaktivira, niti zapisuje, nego se uklanja iz baze.



Slika 21: Proces aktiviranja i deaktiviranja objekata, prema [99]

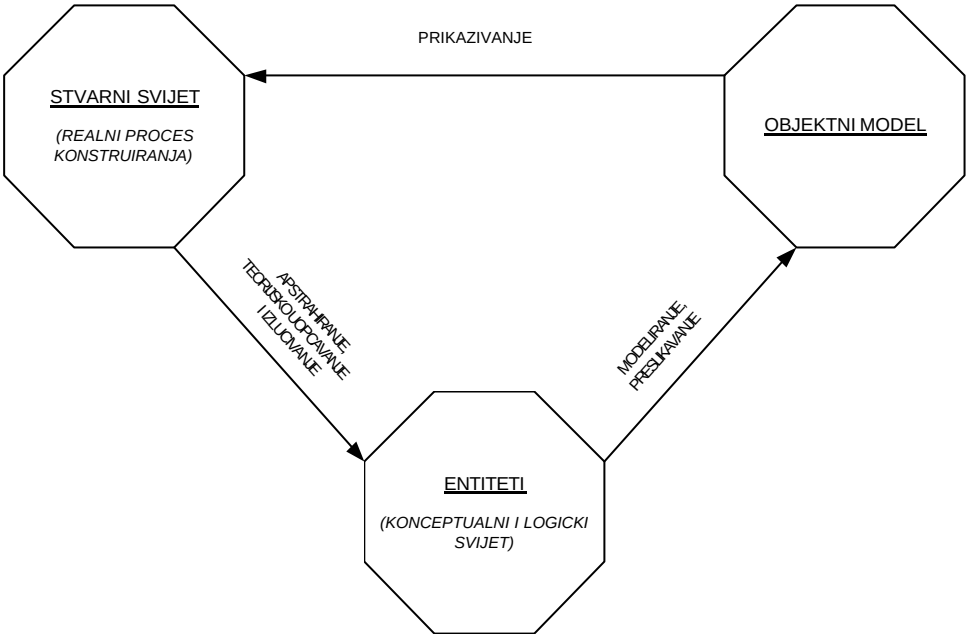
Ako se objekt spremi (zapiše) u bazu i nakon toga procita, treba se ponašati kao da nikad nije niti bio spremljen. To znaci da objekt procitan iz baze mora imati isti identitet, enkapsulaciju, strukturu nasljedivanja, polimorfizam i reference kao originalni objekt [106].
Neki proizvodaci objektnih baza (npr. "POET") tvrde da ispunjavaju takve uvjete. Za verifikaciju mogucnosti implementacije predloženog objektnog modela procesa konstruiranja odabrana je komercijalna objektna baza "POET". Spomenuti proizvodac jedan je od nekoliko vrhunskih, a ujedno je i evaluacijska verzija baze i dokumentacije lako dostupna. U osmom poglavlju biti ce detaljnije obrazložene neke od metoda i modela implementacije objektne baze koje vrijede konkretno za POET, a ne za objektne baze opcenito.
Tehnologija i metode implementacije objektnih baza još uvijek se razvijaju i sazrijevaju, tako da se komercijalne implementacije prilicno razlikuju u metodici i mogucnostima. ODMG organizacija (object data management group) razvija opceniti model koji teži prema standardizaciji, ali još uvijek postoje velike razlike izmedu razlicitih proizvodaca objektnih baza. Npr. prema [99], ODMG standard ne podržava spremanje C++ pokazivaca (pointera) kao dijelova objekta, dok POET objektna baza ima razvijen mehanizam njihovog spremanja i ponovnog aktiviranja nakon ucitavanja spremljenog objekta.

6. Koncipiranje objektno orijentiranog modela procesa konstruiranja

6.1 Preslikavanje pojmova realnog svijeta u konceptualni i objektni model

U obrazlaganju hipoteze rada, i u prethodnom poglavlju, vec je bilo govora o temeljnim principima u pristupu koncipiranju objektnog modela.
Meyer [107] formalno definira objektno orijentirano koncipiranje kao "konstrukciju programskih sustava u obliku strukturiranih skupova (kolekcija) implementacija apstraktnih tipova podataka. Neformalno, definira ga kao "metodu koja vodi do programskih arhitektura temeljenih na objektima kojima manipulira svaki sustav ili podsustav. Objekt se može promatrati kao smislena povezanost odredenog znanja i odredenih operacija.
Izgraditi sustav sa objektno orijentiranim pristupom, znaci analizirati problem i pronaci objekte ukljucene u sustav. Opce znacajke i ponašanje tih objekata modeliraju se i implementiraju kao klase u objektno orijentiranom programskom jeziku. Kada se jednom objekti domene problema modeliraju i kreiraju kao klase, te klase se spajaju zajedno u modeliranju sustava u racunalnom okviru. Takav "bottom-up" pristup, temeljen na podacima, koristi prijašnje napore kao polugu za kreiranje sustava izgradenih od "gotovih dijelova".
Dakle, prvenstveno ce se nastojati apstrahirati i definirati pojave i pojmovi koji odreduju konstrukcijski proces, da bi ih se zatim preslikalo u osnovne klase i objekte modela.

Koncipiranje objektnog modela procesa konstruiranja može se promatrati kao preslikavanje pojava i pojmova iz domene stvarnog svijeta u entitete u domeni konceptulanog i logickog svijeta (slika 22). Entiteti konceptualnog i logickog svijeta preslikavaju se u klase u domeni objektnog modela. Jedna od glavnih prednosti objektnog modeliranja je mogucnost poklapanja entiteta i objekata. Prema [108], cilj objektnog modeliranja je "jedan prema jedan" podudaranje izmedu entiteta u konceptualnom i logickom svijetu i objekata u racunalnom programu.



Slika 22: Apstrahiranje entiteta i njihovo preslikavanje u objektni model

Analogna razmatranja modeliranja kao preslikavanja iz domene realnog svijeta u domene fenomenološkog , informacijskog i racunalnog modela prikazana su slikom 1.

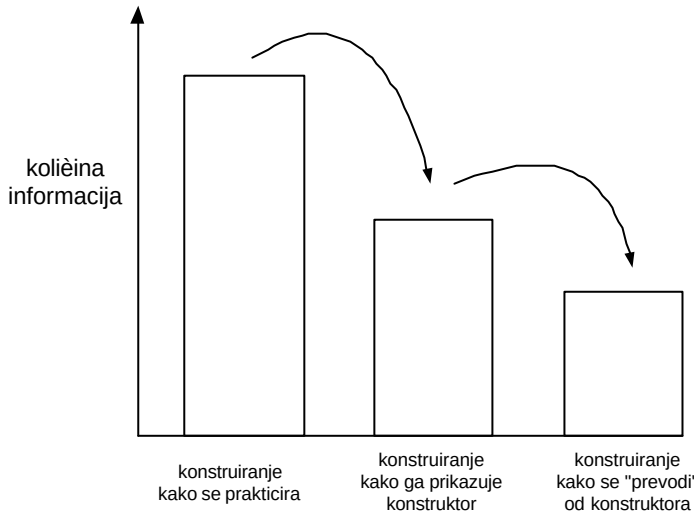
6.2 Problematika formiranja modela procesa konstruiranja u konceptualnoj domeni

Razmatrajuci opce teorije konstruiranja kao moguca polazišta koncipiranja modela, zakljuceno je da niti jedna opca teorija nije dovoljna sama za sebe, niti je pogodna kao osnovno polazište razvoja modela. Niti jedna od tih teorija ne promatra proces konstruiranja sa svih aspekata i nisu (u potpunosti) orijentirane informatickom modeliranju.
Opca teorija konstruiranja (Yoshikawa, Tomiyama) i aksiomatska teorija (Suh) vecinom se zadržavaju na višim razinama apstrakcije i modeliranja, ne razmatrajuci aspekte organizacije timskog rada, odn. suradnje i koordinacije clanova tima (u novije vrijeme time se bave Duffy i Andreasen [109]), nacine dekompozicije i organizacije izvršavanja zadataka, pracenje i vodenje procesa. Opci model procesa konstruiranja (Hubka) širi je i sveobuhvatniji, ali se osniva na fenomenološkom opisu, bez egzaktnog aparata.
Takvi pristupi prvenstveno nastoje produbiti razumijevanje procesa konstruiranja, ali s druge strane ne daju bitne doprinose unapredenju organizacije procesa i skracivanju rokova u danas prevladavajucim okruženjima timskog rada.
Sve navedeno ne znaci da opce teorije treba u ovom pristupu zanemariti, nego naprotiv treba nastojati kombinirati i koristiti njihove spoznaje kao principe koncipiranja strukture i pojedinih entiteta modela.
Cilj pristupa predloženog u ovom radu je interpretativni racunalni model koji bi trebao obuhvatiti što je moguce više pojava i pojmova realnih procesa konstruiranja promatranog sa svim suceljima prema cjelokupnom okruženju proizvodnog procesa. Ponovimo stoga citat Vajne [4] i [75] - što se danas ocekuje od teorije konstruiranja:

- da ukljuci sve znacajke cijelog životnog vijeka proizvoda (razmatranje svih aspekata termina "design for x")
- da ukljuci i podrži vremenski paralelne procese (istovremeno (paralelno) inženjerstvo)
- formalizacija strukture i sadržaja procesa konstruiranja u modelu procesa za deriviranje i realizaciju funkcija za cjelokupnu racunalnu podršku

Razmotrimo još neke teme znacajne za koncipiranje i implementaciju modela procesa konstruiranja.

Posljednjih dvadesetak godina nekoliko istraživackih timova u USA veliku je pažnju posvetilo metodama vizualizacije mreže interakcija konstrukcijskih zadataka i parametara. [110], [111], [112], [113], [114], [115]. Razvijene metode možemo promatrati i kao parcijalne modele procesa konstruiranja (parcijalne jer promatraju samo dio informacija i samo odredene aspekte).
Nedostatak je navedenih istraživanja što promatraju staticne prikaze procesa, na kojima nastoje reorganizacijom unaprijediti proces.
Medutim stanje u okolini procesa konstruiranja nije staticno - stvaraju se novi resursi, dolazi do organizacijskih promjena i promjena u raspodjeli zadataka. Struktura procesa konstruiranja može se mijenjati cak i u vrijeme trajanja konstruiranja jednog proizvoda, usporedo sa napredovanjem kroz faze procesa [5]. U vecini konstrukcijskih ureda postoje propisane procedure "kako se radi", medutim uvijek treba ocekivati da ce "ono što se stvarno dogada" biti redovito za nijansu drugacije. Prema [5] u tijeku formiranja konkretnog modela može doci do gubitka informacija, tj. u procesu prikazivanja realnog stanja ("kako se inace radi") (slika23). Ovi problemi ce postojati u slucajevima kada je "snimanje stanja" pri formiranju modela odvojeni proces od stvarne svakodnevne konstruktorske prakse.



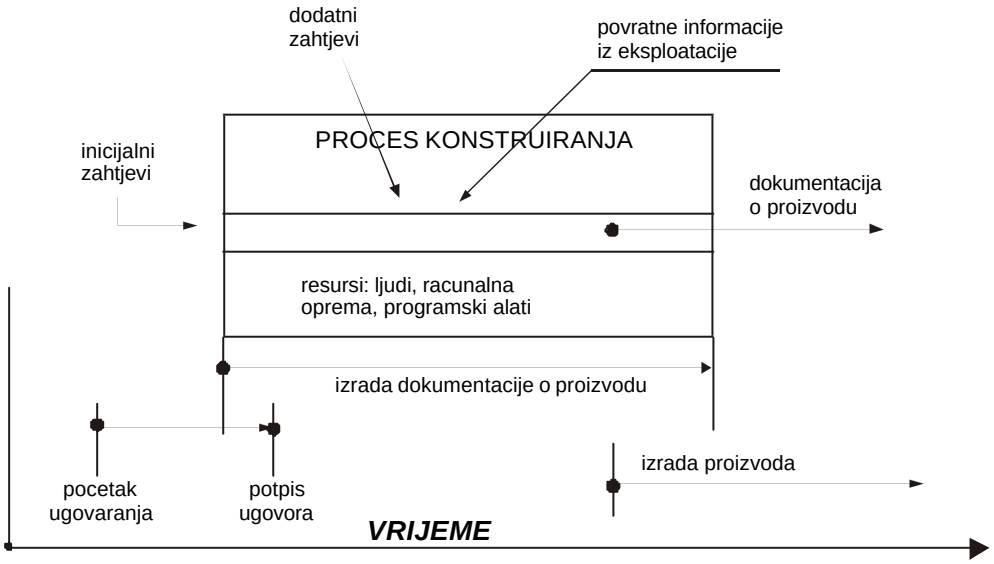
Slika 23: Gubitak informacija pri formiranju prikaza konkretnog procesa konstruiranja [5]

Jednom formirani racunalni model procesa konstruiranja biti ce kasnije podložan stalnom doradivanju i održavanju, bilo zbog unapredivanja ili zbog pracenja promjena u okolini procesa. Održavanje implementiranog modela zahtijevati ce i dodatni angažman konstruktora u odnosu na dotadašnji nacin rada.
Proces konstruiranja pri modeliranju ne bi trebalo promatrati kao staticnu institucionaliziranu strukturu, nego kao dinamicnu mrežu koja nastaje u realnom vremenu usporedo s evoluiranjem konstrukcijskih ciljeva, potreba i prioriteta [5].

Stoga se predlaže koncipirati strukturu modela kao "otvorenu kutiju s objektima". To znaci da treba omoguciti jednostavno dodavanje novih entiteta (klasa) i njihovo semanticko povezivanje sa postojecima. Pri formiranju modela procesa konstruiranja u konkretnom sustavu koriste se raspoloživi entiteti, uz eventualnu prilagodbu i dodavanje novih. Pri tome nije nužno da se koriste svi entiteti iz modela. Ovakva koncepcija predlaže se stoga jer nije realno za ocekivati da model može biti dovoljno opcenit da bi obuhvatio sve varijacije situacija u razlicitim realnim okruženjima. Stoga ga treba koncipirati na nacin koji bi omogucio implementaciju barem dijela do sada razvijenih metoda za unapredivanje i/ili modeliranje procesa.

Realni sustav cije konceptualno modeliranje se razmatra je proces konstruiranja zamišljen u uredu koji koristi racunalnu opremu u mrežnom okruženju i u kojem vec postoji iskustvo u korištenju programskih alata za podršku konstruiranju. Pretpostavlja se da se u procesu koriste CAD paketi, programi za proracune, baze podataka i druge vrste programskih alata. Ulazne informacije u proces konstruiranja su zahtjevi tržišta, odnosno narucitelja, a skup izlaznih informacija je dokumentacija o konstruiranom proizvodu.

Pri modeliranju tako definiranog realnog sustava treba uzeti u obzir i vremenske odnose procesa izrade dokumentacije o proizvodu u odnosu na ugovaranje s naruciocem i pocetak izrade proizvoda. U modernim procesima istovremenog inženjerstva vremenski intervali navedenih procesa cesto se preklapaju (slika 24).



Slika 24: Izrada dokumentacije o proizvodu u odnosu na ugovaranje i izradu proizvoda

To znaci da proces izrade dokumentacije zapocinje i prije konacnog potpisa ugovora, a izrada nekih komponenti proizvoda zapocinje i prije nego je dovršena kompletna dokumentacija. Takve situacije namecu dodatne zahtjeve i kompliciraju proces konstruiranja, ali su nužnost koju diktira tržište stalnim skracivanjem rokova izrade proizvoda. Cesto se pocetni zahtjevi promijene (ili se postavljaju dodatni) i nakon završnog definiranja ugovora. Takve situacije javljaju se ipak samo kod nekih vrsta proizvoda (složena postrojenja i njihove komponente). Ovdje to napominjemo da bi se naglasilo da se proces konstruiranja ne može uvijek promatrati kao proces obrade statickog skupa inicijalnih zahtjeva, što je prevladavajuci model u Znanosti o konstruiranju. Utjecaj na proces konstruiranja imati ce i povratne informacije iz eksploatacije isporucenih varijanti proizvoda.

Konceptualni model procesa konstruiranja nastojati ce se postaviti tako da ne bi trebao biti ni preskriptivni ni deskriptivni (u smislu klasifikacije prema [17]), nego ce se prvenstveno nastojati da bude prilagoden racunalnoj primjeni. Konstruiranje ce se promatrati kao proces transformiranja i generiranja informacija, pretpostavljajuci da je velik dio tih informacija zapisan u racunalnom obliku. Drugim rijecima konstruiranje se promatra kao proces kojem treba modelirati informaticku podršku, preslikavajuci uobicajeni nacin rada u domenu objektnog programskog sustava.
Na temelju izloženog pristupa promatranju i modeliranju realnog sustava predložiti ce se entiteti koncepualnog i logickog modela. Pri koncipiranju strukture nastojati ce se da model bude dovoljno opcenit da bude široko primjenjiv, ali da istovremeno sadrži dovoljno mogucnosti specijalizacije da bi se mogao efikasno prilagoditi specificnostima konkretne okoline.

Najveci dio softverskog tržišta (pa i objektnih sustava) usmjeren je na razvoj podrške poslovnim procesima. Usporedimo li konceptualno modeliranje poslovnih procesa sa modeliranjem procesa konstruiranja, može se zakljuciti da su poslovni procesi ipak znatno jednostavnije strukture nego proces konstruiranja, za koje je lakše apstrahirati konceptualni model. Pored toga dobar dio transakcija i operacija u poslovnim sustavima strogo je definiran i odreden zakonima i propisima. Za razliku od toga velik dio transakcija u procesu konstruiranja ne može se niti predvidjeti, niti propisati.

Konceptualno modeliranje procesa konstruiranja dodatno otežava i okolnost da takav model nije u potpunosti sazrio niti usaglašen niti na razini Znanosti o konstruiranju - odnosno ne postoji jednistven i usaglašen fenomenološki model. Koliko je poznato autoru ovog rada, takva sistematizacija, orijentirana na racunalnu primjenu, još nije provedena u Znanosti o konstruiranju. U Znanosti o konstruiranju nedostaje "teorija CAD-a", što naglašavaju i Akman, Hagen i Tomiyama [25].

6.3 Proces razvoja objektnog modela

U ovom poglavlju razmotriti cemo proces razvoja modela u objektnoj domeni sa informatickog (programskog) aspekta. Treba napomenuti da konceptualno i informacijsko (objektno) modeliranje ne treba promatrati odvojeno, jer postoji velik broj medusobnih upliva i interakcija. Zapravo se ta dva procesa odvijaju paralelno, u iterativnim ciklusima.
Realno je ocekivati da ce pristup u kojem ce se pojave i pojmovi koji odreduju konstrukcijski proces nastojati apstrahirati i preslikati u objekte, kao rezultat dati opsežan i kompleksan model, odnosno programski sustav. Idealno bi bilo kad bi se preslikavanje iz konceptualne u objektnu domenu realiziralo "jedan za jedan", tj. da svakom entitetu odgovara jedna klasa. Da bi se to ostvarilo, pri formiranju entiteta u konceptualnom modelu treba uzeti u obzir i zahtjeve racunalne implementacije, tj. voditi racuna o:

- organizaciji programskog sustava
- izboru strukturalnih elemenata i njihovih sucelja kojima se spajaju u cjelovit sustav
- ponašanju elemenata, specificiranom u njihovim interakcijama i kolaboracijama
- kompoziciji elemenata strukture i ponašanja u progresivno vece podsustave

Ovdje treba naglasiti da je razvoj svih metodologija (pa i objektne) uvijek bio potican najvecim dijelom problemima iz domene upravljanja poslovnim sustavima, što je i razumljivo, jer je taj segment softverskog tržišta daleko najveci. Makar se smatra da su

razvijeni objektni programski jezici opce namjene, ipak bi se moglo reci da izvorišta razvoja imaju svoje uplive na primjenjivost u razlicitim domenama.

Kvaliteta i primjenjivost (upotrebljivost) informacijskog modela djelomicno proizlazi i iz principa modeliranja, odnosno stilova korištenih pri kreiranju konceptualnog modela. Jedan od primjera skupa principa modeliranja prema [78], postavio je West [116].
Navedimo neke principe:

> Entitete je bolje prikazati i nazvati prema njihovoj "prirodi" ("underlaying nature") nego prema ulozi koji imaju u odredenom kontekstu.
> Sve asocijacije izmedu fizickih i logickih objekata trebalo bi prikazati kao entitete radije nego samo kao relacije izmedu objekata.

Još neki interesantni principi modeliranja razmatrani su u "Primitive-Composite" pristupu prema [117]. Entiteti se mogu temeljiti na nekoliko razlicitih konceptualnih dekompozicija ("conceptual breakdowns"). Na primjer "konstruktor" se može definirati u smislu svoje uloge kao osobe, zaposlenika, resursa u aktivnosti, fizickog objekta sa jedinstvenom lokacijom, clana tima, entiteta koji egzistira u nekom stanju, itd. Svaka od tih razlicitih klasifikacija može se prikazati upotrebom "primitivnog" objekta. Cjelokupni entitet koji kombinira sve karakteristike primitiva naziva se "kompozitni objekt". Iz te perspektive, nastaje veliki problem za konceptualne modele da unaprijed definiraju sve moguce kombinacije i permutacije svih primitivnih dekompozicija.

Proces razvoja složenih i sofisticiranih programskih sustava karakteriziran je iterativnošcu. Sukcesivna poboljšanja i inkrementalni rast efikasnosti rješenja problema odvijaju se kroz nekoliko ciklusa [101]. Kako je vec navedeno, realno je ocekivati da ce predloženi pristup rezultirati sa vrlo složenim i opsežnim modelom programskog sustava.

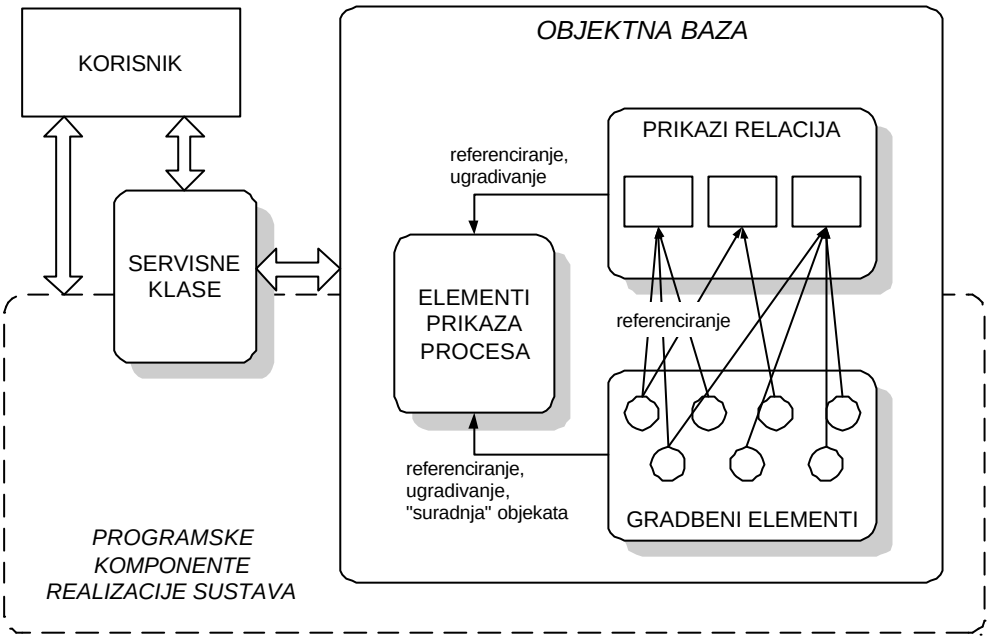
Stoga ce se opseg ovog rada ograniciti samo na definiranje i razradu entiteta konceptualnog modela, te nacina njihovog preslikavanja u klase objektnog modela. U objektnom modelu definirati ce se struktura klasa, njihovi atributi, operacije i relacije, do razine na kojoj može zapoceti razvoj programskog koda i detalja implementacije. Objektno orijentirani pristup podržava apstrakciju na razini objekta. Razvoj sustava može se stoga odvijati na razini objekata ignorirajuci ostatak sustava tako dugo dok god je to potrebno, pa nije nužno obracati pažnju na razinu implementacije.
Cilj tako postavljenog istraživanja je stvoriti temelj ("kostur modela") koji ce biti dovoljno opisan i dovoljno precizan da bi omogucio daljnji razvoj - eventualno kao polazna tocka ili poticaj za šire usaglašavanje koncepcije i definicija entiteta racunalnog modela procesa konstruiranja.
Definicije klasa i svi ostali elementi "kostura" objektnog modela zapisati ce se u obliku UML modela, korištenjem programskog paketa "Rational Rose". Implementacija tako zapisanog modela realizirati ce se u programskom okruženju objektne baze.

6.4 Koncepcija osnovne strukture (arhitekture) sustava

Temeljem svih dosadašnjih razmatranja i istraživanja predlaže se osnovna shema arhitekture sustava prikazana na slici 25. Prema [118] u dobro koncipiranom modelu, klase trebaju biti uredno biti grupirane u podsustave koji su slabo spregnuti. Veze izmedu klasa iz razlicitih podsustava trebaju biti minimizirane. Arhitektura sustava treba sadržavati što je

moguce više stablinih apstrakcija, koje treba izolirati od onih za koje je vjerojatnije da ce biti podložne promjenama.
Vodeci se tim principima, predložena arhitektura sustava podijeljena je u cetiri osnovne grupe (podsustava) klasa:

- osnovni gradbeni (strukturalni) elementi
- elementi prikaza relacija
- elementi prikaza i kontrole izvodenja procesa konstruiranja
- "servisne" klase koje realiziraju "ponašanje" modela, i skup programskih komponenti realizacije sustava



Slika 25: Shema osnovnih elemenata strukture objektnog modela procesa konstruiranja

Predložena arhitektura nastoji implementirati "bottom up" pristup gradnji modela, stoga se prvo polazi od najjednostavnijih entiteta (elemenata) koji ce se koristiti kao gradbeni elementi složenijih entiteta. Strukturalni (gradbeni) elementi modeliraju osnovne pojmove realnog svijeta odvijanja procesa konstruiranja. Ovdje se predlaže slijedeca gruba kategorizacija strukturalnih elemenata na:

- objekte na kojima se izvodi proces konstruiranja (koji "trpe" sve akcije i operacije): zapisi informacija o proizvodu i logicki modeli "fizickih" komponenti proizvoda
- objekte koji "djeluju" u procesu: konstruktori, programski alati, akcije - kao dalje nedjeljive etape procesa

Relacije izmedu entiteta u procesu konstruiranja daleko su složenije nego one u npr. poslovnim sustavima. Stoga se pretpostavlja da ce neke od složenijih zavisnosti biti pogodnije modelirati kao zasebne entitete, jer mehanizmi objektne tehnologije za modeliranje relacija u vecini slucajeva nece biti dovoljni ili pogodni.
Elementi prikaza i modeliranja relacija cine stoga posebnu grupu (podsustav) klasa, ciji objekti takoder sudjeluju i u gradnji složenih entiteta (objekata). Objekti prikaza relacija na razlicite nacine prikazuju relacije izmedu skupova gradbenih elemenata iste klase ili izmedu skupova gradbenih elemenata razlicitih klasa. Pri tome objekti prikaza relacija sadrže skupove referenci na gradbene elemente cije relacije prikazuju.

Elementi prikaza i kontrole odvijanja procesa konstruiranja apstrakcije su koje ce proizaci iz dekompozicije procesa konstruiranja promatranog kao niza akcija pretvorbe informacija. Dakle to ce biti prikazi etapa, stanja i situacija u procesu, te mreže njihovih veza. Elementi prikaza i kontrole procesa koncipirati ce se kao složeni objekti u koje ce se ukljucivati objekti osnovnih gradbenih elemenata na slijedece nacine: referenciranjem, ugradivanjem (kompozitni objekti) i modeliranjem "suradnje" izmedu objekata.
Na temelju složenih objekata prikaza procesa mogu se dalje graditi modeli odredenih situacija, postupaka i metoda konstruiranja, prilagodeni potrebama konkretnog okruženja primjene. U navedenom pristupu ogleda se fleksibilnost koncepcije "otvorene kutije s alatima".

Prve tri razmatrane grupe klasa modeliraju osnovnu "staticku" strukturu sustava. Sve klase iz tih grupa imati ce skupove instanci koji ce svi zajedno tvoriti model procesa konstruiranja. Realno je pretpostaviti da ce klase iz prve tri grupe imati veliki broj instanci, stoga je predvideno da se sve njihove instance spremaju u objektnu bazu. Objektna baza predstavlja onda zapis staticke strukture modela procesa konstruiranja. Takvoj "statickoj" strukturi cetvrta grupa ("servisnih" klasa) dodaje "dinamicki" aspekt. Servisne klase sadrže skupove operacija koje modeliraju ponašanje sustava u eksploataciji. Primjeri takvih operacija su npr. generiranje plana procesa konstruiranja i izvodenje plana. U cetvrtu grupu svrstane su i programske komponente realizacije sustava - skup sucelja za komunikaciju s korisnicima sustava, sucelja prema postojecim bazama znanja, programsko okruženje objektne baze, modeli organizacije zapisa i arhiviranja planova, itd.
Predvideno je da u cetvrtoj grupi klasa budu klase koje sadrže samo operacije, bez modela struktura podataka. Takve klase nece imati više instanci, pa se ne spremaju u objektnu bazu.

Razrada predložene arhitekture sustava kombinirati ce dalje analizu i koncipiranje. Analizom ce se nastojati modelirati proces konstruiranja *otkrivanjem* klasa i objekata koje cine rjecnik (vokabular) domene problema. Pri koncipiranju *izmišljati ce se* apstrakcije i mehanizmi koji realiziraju (omogucavaju) ponašanje koje taj model zahtijeva.

7. Prijedlog entiteta objektnog modela procesa konstruiranja

Kako je u prethodnom poglavlju obrazloženo, u razradi objektnog modela procesa konstruiranja entiteti su svrstani u cetiri grupe (tablica 2):

- osnovni gradbeni (strukturalni) elementi
- elementi prikaza relacija
- elementi prikaza i kontrole izvodenja procesa konstruiranja
- programske komponente realizacije sustava i "servisne" klase koje realiziraju "ponašanje" modela

GRADBENI ELEMENTI	ZAPISI RELACIJA IZMEĐU OBJEKATA ILI ATRIBUTA	PRIKAZ I IZVOĐENJE PROCESA	KOMPONENTE REALIZACIJE
parametar baza parametara objekt prikaza proizvoda objekt strukture proizvoda akcija sucelje programskog alata zadatak konstruktor	zavisnosti parametara zavisnosti konstrukcijskih zadataka relacije pripadnosti izmedu razlicitih klasa objekata izrazi - relacije izmedu atributa objekata ogranicenja - relacije parametara i funkcionalnih zahtjeva pravila odlucivanja - relacije izmedu uvjeta i akcija	plan konstruiranja cvor plana konstruiranja matrica prikaza veza izmedu cvorova plana zapis tijeka izvodenja procesa konstruiranja zapis stanja procesa u izvodenju postupak kreiranja plana postupak izvodenja plana nacrt plana konstruiranja	"servisne" klase - kreiranje i izvodenje plana rjecnik objektne baze objektna baza sa strukturom plana konstruiranja arhiva kreiranih planova arhiva izvedenih planova sucelja prema bazama znanja i bazama podataka baza znanja o programskim alatima sustava

Tablica 2: Entiteti i komponente objektnog modela procesa konstruiranja

U ovom poglavlju dane su definicije i prikazi predloženih entiteta unutar svake od navedenih grupa, što možemo promatrati kao detaljnu razradu konceptualnog modela sustava. U danim prikazima razmatrano je i preslikavanje predloženih entiteta u klase objektnog modela. Sve klase objektnog modela imaju iste nazive kao i entiteti konceptualnog modela. Detaljnija razrada implementacije i realizacije entiteta kao strukture klasa i relacija u konkretnom objektnom modelu (programskom sustavu) razradena je u osmom poglavlju.

7.1 Strukturalni (gradbeni) elementi

7.1.1 Parametar konstrukcije

Promatrajuci proces konstruiranja kao proces transformacije i generiranja informacija može se smatrati da je podatak, odnosno u racunalnom modelu varijabla, temeljna i nedjeljiva jedinica informacije. Varijabla dakle predstavlja najjednostavniji entitet u modelu procesa konstruiranja. Varijable ce se u predloženom modelu nazivati "parametri konstrukcije", što je najcešci naziv u literaturi (npr. [110], [119], [120]).
Parametri konstrukcije promatrati ce se kao osnovne jedinice zapisa informacija o proizvodu koji se konstruira.

Vrijednosti parametara odreduju se u tijeku procesa konstruiranja, ali vrijednost parametra može biti poznata vec i na pocetku procesa konstruiranja ako se radi o podacima iz liste zahtjeva ili o nekom podatku varijantne konstrukcije.
U objektnom modelu parametri se mogu modelirati kao atributi objekata ili kao zasebni objekti.
Razmatrajuci znacajke procesa konstruiranja, posebno u timskom okruženju, može se uociti potreba za još nekim elementima u sintaksi parametra osim samog zapisa vrijednosti podatka - kao npr. "status vrijednosti", fizikalna jedinica i hiperveza na zapis relevantnog znanja o parametru. U takvom pristupu parametar se ne može modelirati kao jedan atribut objekta, nego eventualno kao skup (struktura) povezanih atributa. Odmah se namece zakljucak da je, slijedeci principe objektnog modeliranja, pogodnije onda parametre modelirati kao objekte koji ce enkapsulirati takve skupove atributa.

Entitet "parametar konstrukcije", preslikan na objekt u modelu procesa konstruiranja enkapsulira:
- *zapis vrijednosti podatka o proizvodu,*
- *status vrijednosti (koji se mijenja u izvodenju procesa konstruiranja),*
- *reference na dodatne opise - znanja o procesu konstruiranja i proizvodu,*
- *zapise "namjera" konstruktora - prijedloge, argumente i pretpostavke.*

Može se reci i da objekt "parametar konstrukcije" modelira "životni ciklus" i zapis znanja o podatku konstrukcije. Parametri kao objekti sadrže slijedeci skup atributa:

$P = \{n, a, v, s, f, h\}$, pri cemu je:

n ::= naziv
a ::= adresa u bazi parametara
v ::= vrijednost
s ::= status vrijednosti
f ::= fizikalna jedinica
h ::= hiperveza na opis parametra i zapis relevantnog znanja

Pri tome v poprima tocno jednu vrijednost iz skupa mogucih stanja V:
$v \in V$, gdje je $V = \{v_1, v_2, v_3\}$

v_1 ::= zapisana je jedna vrijednost
v_2 ::= zapisana je donja i gornja granica kontinuiranog intervala unutar kojeg je vrijednost
v_3 ::= zapisan je niz (polje) diskretnih vrijednosti

Isto tako *s* poprima tocno jednu vrijednost iz skupa mogucih stanja *S*:

$s \in S$, gdje je $S = \{s_1, s_2, s_3, s_4, s_5\}$

s_1 ::= vrijednost je odredena, ali još se može mijenjati
s_2 ::= vrijednost odredena, i ne može se više mijenjati
s_3 ::= pretpostavljena jedna vrijednost
s_4 ::= vrijednost je pretpostavljena unutar kontinuiranog intervala
s_5 ::= vrijednost je pretpostavljena kao jedna od vrijednosti iz niza diskretnih vrijednosti

Dozvoljene kombinacije vrijednosti *s* i *v* podskup su kartezijevog produkta skupova njihovih mogucih stanja:

$K = \{(v_1,s_1), (v_1,s_2), (v_1,s_3), (v_2,s_1), (v_2,s_4), (v_3,s_5)\}$, $K \subseteq V \times S$

Implementacija kontrole razlicitih kombinacija *s* i *v* može se riješiti sa razlicitim klasama parametra ili sa kontrolnom procedurom implementiranom u generiranje nove instance parametra.

Pojam "statusa vrijednosti" može biti osobito znacajan u iterativnim procesima i informacijski spregnutim konstrukcijskim zadacima. Oznacavanje da je vrijednost parametra privremena, pretpostavljena, unutar odredenih granica i slicno može omoguciti modeliranje algoritama rješavanja iterativnih, odnosno spregnutih zadataka i unaprijediti komunikaciju unutar tima koji ih rješava.

Uz status vrijednosti trebalo bi vezati i *prijedloge* i *argumente*, što bi bilo posebno interesantno u okruženjima timskog rada, gdje više konstruktora "dijeli" vrijednosti kljucnih parametara, odnosno suraduje na odredivanju njihovih vrijednosti. Pri tome svaki konstruktor, u okviru svog parcijalnog zadatka ima i svoje zahtjeve (prijedloge) na vrijednost parametara koje potkrepljuje svojim argumentima, te treba naci kompromisna rješenja. Zapisivanje prijedloga i argumenata uz status vrijednosti parametra olakšalo bi komunikaciju izmedu clanova tima, a ujedno bi se na taj nacin zapisala i "povijest" razvoja konstrukcije, odnosno razlozi donošenja pojedinih odluka. Prijedlog modela takvog zapisa podataka razvijen je u radu [121].
Implementacija zapisa prijedloga i argumenata zahtijeva razvoj odgovarajucih sucelja i operacija koje trebaju biti dio modela ponašanja sustava.

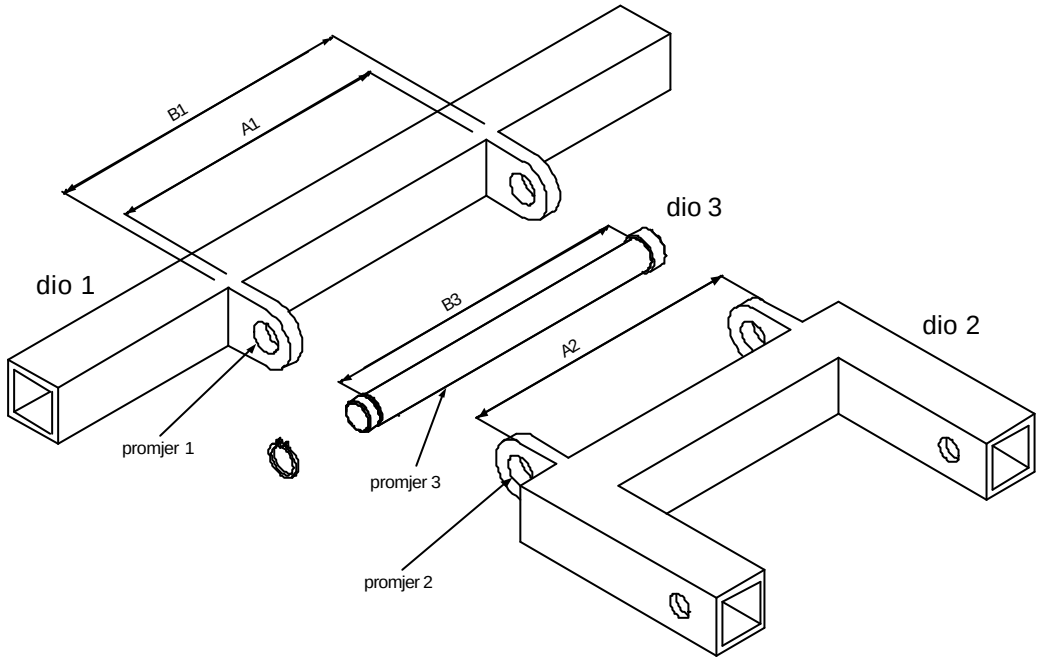
Parametre možemo nazivati i konstrukcijske varijable ili atributi konstrukcije. U daljnjem razvoju, odnosno pri programskoj realizaciji modela može se razmotriti slijedece pitanje: Da li svi podaci konstrukcije trebaju biti modelirani kao parametri koji su zasebni objekti? Potvrdni odgovor znacio bi manipuliranje sa vrlo velikim brojem objekata u sustavu, što bi ipak trebalo nastojati izbjeci. Stoga bi trebalo one konstrukcijske podatke za koje se sigurno može ustanoviti da ih je dovoljno modelirati kao atribute objekata, takvima i zadržati.
Koji parametri konstrukcije bi onda nužno trebali biti modelirani kao zasebni objekti, a koji mogu biti "samo" atributi drugih objekata ?

Inženjerske strukture podataka redovito su znatno kompleksnije od onih u poslovnim sustavima. Razmotrimo situaciju u kojoj su pojedini dijelovi i sklopovi konstrukcije modelirani kao posebni objekti, a svi podaci odnosno parametri konstrukcije modelirani su iskljucivo kao atributi takvih objekata.

Lako je pokazati da ce u navedenom pristupu, doci do situacija u kojima ce, s konstrukcijskog aspekta, isti parametar biti modeliran kao atribut razlicitih objekata (slika 26). Takvi objekti, opcenito gledano ne moraju pripadati istoj hijerarhiji klasa.
Primjeri takvih parametara najcešce ce biti geometrijski podaci (kote dosjeda) koje moraju imati istu osnovnu vrijednost (zanemarujuci tolerancije) na razlicitim objektima, odnosno dijelovima i sklopovima proizvoda. Ukoliko se vrijednost parametara modeliranih kao atributa ne može "prenijeti" mehanizmom nasljedivanja, nužno je da se njihovim vrijednostima upravlja na jednom mjestu do kojeg ce svi objekti imati pristup. Dakle, u takvim situacijama parametri nužno moraju biti modelirani kao zasebni objekti.

7.1.1.1 Modeliranje konstrukcijskih parametara kao zasebnih objekata

Ovdje cemo detaljnije razmotriti jedan primjer situacije u kojoj je nužno parametre konstrukcije modelirati kao zasebne objekte.
Pri razmatranju problematike modeliranja parametara konstrukcije vec je spomenuto da u jednom od mogucih pristupa modeliranju oni mogu pripadati (kao atributi) odredenim objektima. Ovdje se radi o objektima koji sadrže bilo koji oblik prikaza informacija o konstrukciji ili su to objekti koji predstavljaju sucelje prema skupu informacija o konstrukciji. U daljnjem tekstu zvati cemo ih objektima "prikaza proizvoda", a detaljnije ce biti razmotreni u slijedecem poglavlju, kao složeniji gradbeni elementi modela procesa konstruiranja.

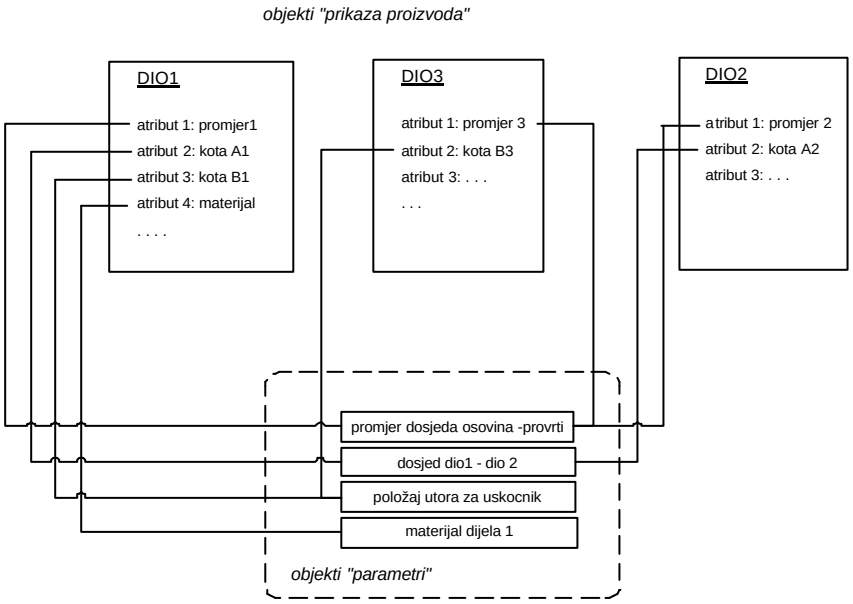


Slika 26: Primjer parametara koji moraju dijeliti istu vrijednost

Razmotrimo primjer jednostavnog sklopa sa slike 26. Neka su zapisi informacija o dijelovima sklopa modelirani kao hijerarhijski nezavisni objekti "prikaza proizvoda". Razmatrani objekti nisu dakle 3D CAD modeli dijelova i sklopa, nego takve objekte promatramo kao objekte koji sadrže sucelja prema CAD modelima i njihovim skupovima geometrijskih parametra. Drugim rijecima objekti "prikaza proizvoda" trebaju zapisati osnovne informacije iz CAD modela u obliku svojih atributa, odnosno oni su sucelja

transfera informacija izmedu CAD modela i modela procesa konstruiranja. Premda je na slici 26 vrlo jednostavna konstrukcija, pretpostavimo da dijelove ne radi isti konstruktor i da nisu modelirani unutar istog parametarskog 3D CAD modela (prikazani dijelovi mogu npr. biti elementi složenih sklopova). Neka je svaki od dijelova modeliran "svojim" objektom prikaza proizvoda, koji "zapisuje" geometrijske parametre kao svoje atribute. Razmotrimo kote dosjeda u sklopu. Kote "A1" i "A2", te "B1" i B3" moraju imati istu vrijednost nazivne mjere, isto tako i promjeri provrta i promjer osovine. Kako osigurati da zapisi ovih parametara konstrukcije kao atributa razlicitih objekata imaju istu vrijednost? Ako u tijeku razvoja konstrukcije dolazi do cestih promjena vrijednosti konstrukcijskih parametara, kako osigurati pravovremenu komunikaciju izmedu konstruktora, ako je odredivanje svakog od povezanih parametara dio razlicitih zadataka razlicitih konstruktora? To znaci da svakim od objekata prikaza proizvoda manipulira drugi konstruktor, i to u razlicitim vremenskim razdobljima.
Ovdje se predlaže rješenje u kojem objekti prikaza proizvoda ne sadrže u atributima zapise vrijednosti konstrukcijskih parametara nego pokazivace ("pointere") na objekte, odn. instance klase "parametar" koji enkapsuliraju zapis vrijednosti konstrukcijskih parametara. Manipuliranje svojstvima i zapisom vrijednosti konstrukcijskog parametra na samo jednom mjestu (unutar objekta "parametar"), osigurava ispravnost konstrukcije, ali i otvara mogucnosti sofisticiranije kontrole pristupa zapisu vrijednosti parametra. Isto tako otvara se i mogucnost modeliranja komunikacije izmedu konstruktora koji manipuliraju sa zajednickim parametrima u obliku razmjene prijedloga i argumenata.

Na slici 27 prikazana je shema referenciranja konstrukcijskih parametara cija vrijednost je zajednicka za više dijelova modeliranih kao "objekata prikaza proizvoda". Atributi objekata sadrže pokazivace na objekte klase "parametar". Atributi mogu imati razlicite nazive u razlicitim objektima (npr. "kota A1", "kota A2"), ali moraju pokazivati na zajednicki parametar, ciji zapis vrijednosti dijele. U ovom primjeru objekt klase parametar, "dosjed dio1-dio2", sadržavati ce dakle zajednicku vrijednost nazivne mjere. Tolerancije mogu biti zapisane kao dodatni atributi objekata klase "parametar" ili u svakom od objekata prikaza proizvoda posebno.



Slika 27: Povezivanje "zajednickih" atributa objekata referenciranjem istog parametra

Objekti prikaza konstrukcije mogu sadržavati atribute koji su svojstveni samo jednom objektu, ali takoder referenciraju objekt klase "parametar" - npr. atribut "materijal" dijela 1 (slika 27).
Isto tako atributi objekata modela procesa konstruiranja za koje to nije nužno, ne trebaju biti modelirani kao zasebni objekti, odn. objekti klase "parametar". To ce najcešce biti atributi koji "ne sudjeluju" ni u kakvim interakcijama sa drugim objektima, odnosno nema potrebe za njihovim eksponiranjem izvan dosega samog objekta.

Slika 26 vrlo je jednostavan primjer "dijeljenja" iste vrijednosti, odnosno relacije jednakosti izmedu konstrukcijskih parametara koji su geometrijski podaci. Opcenito gledano, nije nužno da samo geometrijski podaci razlicitih objekta konstrukcije budu povezani i modelirani na ovaj nacin, nego se predložena koncepcija može primjeniti i na ostale vrste podataka.
Treba naglasiti da u ovom primjeru promatramo vrlo jednostavnu konstrukciju odnosno sklop. Pri spajanju više složenih dijelova i podsklopova u konacni proizvod razmatrani problem može biti i te kako znacajan. Realne situacije (u složenim konstrukcijama) sadržavati ce daleko veci broj povezanih parametara i složenije relacije izmedu njih, gdje npr. promjena jednog parametra utjece na nekoliko drugih. U takvim slucajevima potrebno je zapisati i sve relacije izmedu parametara, što ce biti razmotreno u poglavlju 7.2.4.

Iz razmatranog primjera može se zakljuciti:

- U tijeku formiranja objekata prikaza konstrukcije treba voditi racuna o svim atributima koji su na bilo koji nacin povezani s atributima drugih objekata (direktno ili indirektno), te takve atribute nužno treba modelirati kao objekte, odn. instance klase "parametar"
- Može se ocekivati da (pogotovo za složene konstrukcije) nije moguce odmah na pocetku uociti i definirati sve meduzavisnosti unutar skupa objekata prikaza konstrukcije. To znaci da se parametri kao objekti i njihove relacije trebaju moci definirati i ažurirati u cijelom tijeku procesa konstruiranja.
- Struktura hijerarhije, odnosno dekompozicije objekata prikaza konstrukcije, od bitnog je utjecaja na modeliranje i izbor parametara. Npr. u razmatranom primjeru dijelovi su mogli biti modelirani kao jedan objekt (sklop), u kojem su mjere dosjeda "interni atributi".

Osim navedenih situacija, kao zasebni objekti trebaju biti modelirani i oni parametri za koje nije pogodno ili moguce modelirati ih kao atribute drugih objekata iz raznih drugih razloga - npr. zbog implementacije "statusa vrijednosti", hiperveze, zapisa znanja, omogucavanja pristupa raznim programskim alatima, itd. Svakako bi kao zasebne objekte trebalo modelirati skup kljucnih parametara konstrukcije (gabariti, performanse i slicno) da bi se omogucio brži i jednostavniji pristup i pregled statusa i vrijednosti takvih podataka.
Realno je pretpostaviti da ce postojati i znacajan skup parametara (tj. podataka) konstrukcije kojima nužno nije potrebna oznaka statusa vrijednosti i hiperveza, pa se mogu modelirati kao "obicni" atributi objekata kojima pripadaju.

7.1.2 Baza parametara

Kompletan skup informacija o proizvodu redovito sadrži vrlo velik broj podataka. Informacije se grubo mogu klasificirati kao geometrijske, strukturalne (topološke) i tehnološke. Svaka složenija konstrukcija biti ce tako opisana sa nekoliko tisuca ili desetaka tisuca podataka. Ta cinjenica odmah namece potrebu da se parametri (kao modeli konstrukcijskih podataka) moraju strukturalno organizirati radi lakšeg manipuliranja sa tako

brojnim skupom podataka. Za tu svrhu u predloženom modelu predvidena je "baza parametara". Termin "baza" u ovom se kontekstu koristi u nedostatku prikladnijeg, makar predloženi entitet ne mora imati sva svojstva baze podataka i nije nužno da se realizira u programskoj okolini baze podataka. (Možda jednako ili više neprikladni termini mogli bi biti npr. "skup", "kontejner" ili "pregled" parametara).

Baza parametara modelira logicku organizaciju skupa parametara konstrukcije modeliranih kao objekata, uz kontrolu jedinstvenosti naziva parametra i pristupa atributima parametra, te sadrži i operacije pretraživanja i pregledavanja.

Baza organizira (sadrži) sve parametre konstrukcije koje je iz bilo kojeg razloga pogodno ili nužno modelirati kao zasebne objekte. Glavne operacije koje treba sadržavati baza parametara:

- manipuliranje strukturom parametara
- kontrola pristupa parametrima i transfera vrijednosti parametara prema ostalim entitetima sustava

Osim osiguranja jedinstvenih vrijednosti "zajednickih" parametra, treba izdvojiti još jednu važnu funkciju baze parametara - da služi kao univerzalno sucelje, odnosno izdvojeni "servis" transfera vrijednosti parametara.

Cesto ce se u konstrukcijskim uredima naici na skup raznorodnih aplikacija koje su nastajale kroz dugi niz godina, cije ulazno-izlazne procedure nisu povezane. Ovdje se "programskim alatima" (procedurama) smatraju programi (najcešce proracuni) pisani u proceduralnim jezicima koji u velikom broju slucajeva nisu integrirani medusobno, niti sa CAD modelima. Vrlo je vjerojatno da ce u vecini slucajeva biti isplativije takve alate ne mijenjati i pokušati ih implementirati u objektni model, nego razvijati nove operacije unutar objektnog modela koje bi zamijenile takve alate. Jedan od nacina implementacije je izrada posebnih dodatnih procedura za svaki program koje ce brinuti o transferu podataka prema bazi parametara. Dakle za svaki pojedini programski alat treba razviti sucelje koje ga pokrece i obavlja transfer podataka. Sucelja moraju poznavati strukturu baze parametara kao i strukturu ulazno/izlaznih operacija programskog alata kojem pripadaju.
U takvom pristupu baza parametara osigurava jedinstvenost zapisa vrijednosti i jednostavniju dostupnost konstrukcijskih parametara (jer se "izdvajaju" iz svojih objekata). Medutim dodatni problem ovakvog pristupa je potreba za "dvostrukim" ažuriranjem sucelja - sa strane eventualne promjene konfiguracije baze parametara i sa strane dorada i promjena programskih alata.

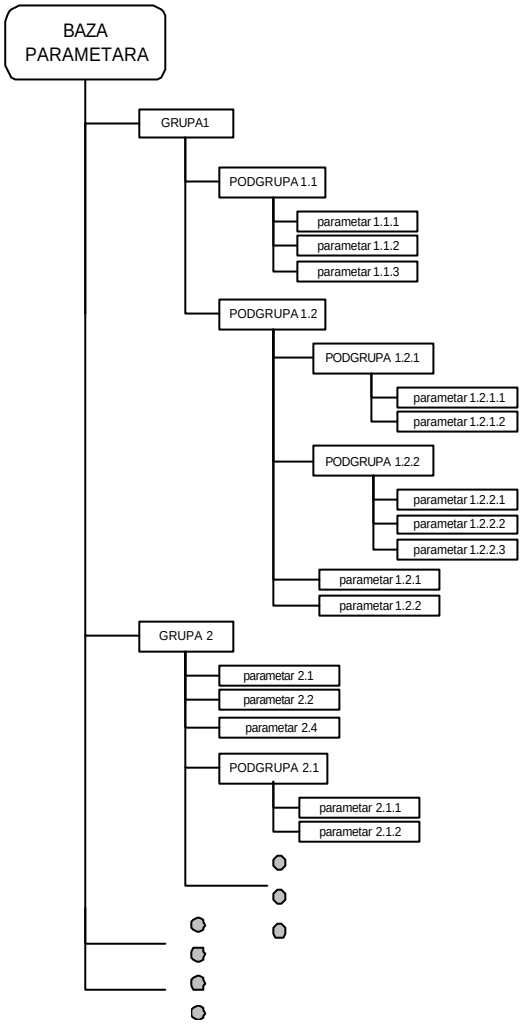
Preko "sucelja programskih alata" i baze parametara osigurava se dakle transfer vrijednosti:

- izmedu atributa objekata "prikaza proizvoda" (koji referenciraju objekte klase "parametar") i "vanjskih" programskih alata
- izmedu više "vanjskih" programskih alata (procedura), koje koriste odredene konstrukcijske parametre u slijedu

7.1.2.1 Organizacija baze parametara

Realno je pretpostaviti da ce baza parametara sadržavati vrlo velik broj parametara u modelima složenijih konstrukcija. Konstruktori bazu popunjavaju postupno, a potrebno je osigurati i jednostavne postupke ažuriranja, pregledavanja i pretraživanja. Stoga je svakako potrebno bazu parametara strukturirati na nacin koji olakšava navedene operacije.

Kategorizacija parametara svakako može olakšati snalaženje (pronalaženje i manipuliranje) u velikom skupu parametara, pa se u ovom radu predlaže bazu parametara strukturirati u obliku hijerarhijskog stabla (slika 28), tj. na isti nacin kako je organizirano "stablo" direktorija i datoteka. U slucaju npr. nekoliko tisuca parametara koji su svi smješteni na istoj razini hijerarhije snalaženje konstruktora u takvoj linearnoj strukturi bi bilo vrlo teško.



Slika 28: Organizacijska struktura baze parametara

Na slici 28 osnovne kategorije su prikazane kao grupe. Svaka grupa može se dalje granati na podgrupe, a parametri se mogu pohranjivati u svim razinama, pocevši od osnovnih grupa. Izuzetno je važno da svaki parametar ima svoju jedinstvenu adresu u predloženoj strukturi. Adresa parametra podatak je s kojim ce manipulirati vecina klasa (entiteta) objektnog modela procesa konstruiranja. Dakle, osim što mora biti jedinstvena, adresa parametra ne smije se mijenjati nakon pocetnog definiranja, ili svi objekti koji referenciraju parametar moraju koristiti pokazivace ("pointere") na adresu parametra u strukturi baze parametara. Da li parametar treba imati i jedinstveni naziv? Mišljenje je autora da nije prakticno inzistirati na tome da naziv bude jedinstven u bazi, nego samo unutar pojedine kategorije, odnosno grupe. Naime ako se postavi zahtjev na jedinstvenost unutar cijele baze, to može stvoriti probleme u odredivanju ("izmišljanju") naziva parametara koji vec imaju uvriježene slicne ili iste termine u praksi odredenog okruženja. Ako se nazivu parametra pridoda i naziv kategorije kojoj pripada, tada takva oznaka mora biti jedinstvena. Takva oznaka parametra više govori o parametru nego sam naziv.

7.1.2.2 Operacije u bazi parametara

U procesu razvoja nove konstrukcije baza parametara nastaje i "puni" se usporedo s razradom funkcionalne strukture i razradom komponenti, odnosno s kreiranjem objekata prikaza konstrukcije. Pri konstruiranju varijantne i ponovljene konstrukcije može se koristiti postojeca struktura (prethodno završene varijante) koja se eventualno prilagodava i dograduje. Bez obzira na vrstu konstrukcije ciji proces se modelira, baza parametara trebala bi sadržavati slijedece osnovne operacije:

- dodavanje i brisanje grupe i pojedinacnog parametra,
- promjena hijerarhije grupa,
- agregacija grupa, rastavljanje grupa (dekompozicija),
- premještanje parametra i/ili grupe unutar hijerarhije,
- promjena naziva parametra i/ili grupe,
- operacije promjene vrijednosti parametra i ostalih uz njega vezanih zapisa,
- pregledavanje i pretraživanje po razlictim kriterijima - upitima

Ovdje smo naveli samo skup osnovnih operacija koje modeliraju strukturu i organizaciju baze parametara, no pored njih potrebno je realizirati i operacije kontrole pristupa bazi i transfera podataka prema suceljima programskih alata, o cemu ce biti rijeci nešto kasnije.

7.1.2.3 Pristup realizaciji modela baze parametara

Model baze parametara moguce je realizirati na više nacina:

- u okruženju objektne baze podataka
- u okruženju relacijske baze podataka
- kao skup ASCII datoteka i potrebnih aplikacija za realizaciju svih operacija

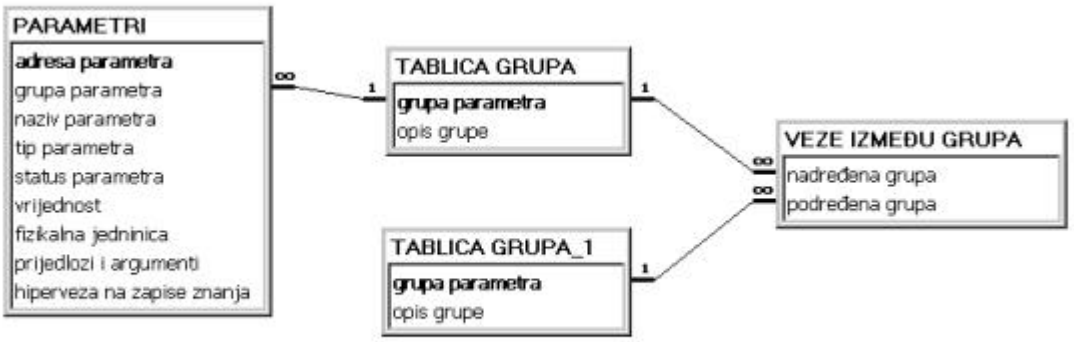
Svaki od navedenih pristupa ima svoje prednosti i mane. Treba naglasiti da sva tri pristupa moraju osigurati siguran rad u okruženju racunalne mreže.

Objektna baza svakako je "najprirodniji" pristup, obzirom na realizaciju cijelog objektnog modela procesa konstruiranja. Svaki parametar modelira se kao objekt, pa je logicno da se sprema u objektnu bazu. U istoj bazi bili bi spremljeni i objekti klasa koje sadrže sve potrebne operacije modeliranja i implementacije baze parametara.

Relacijska baza pogodna je zbog lakše implementacije mehanizama manipuliranja strukturom i pogotovo pretraživanja, te ispisa izvješca. Medutim, relacijsku bazu teže je implementirati u objektno orijentirano okruženje, a kompliciranije je realizirati i postupke transfera podataka prema objektima sustava.

Osnovna prednost pristupa u kojem se baza parametara realizira kao skup ASCII datoteka i pripadajucih aplikacija je u daleko jednostavnijem transferu podataka prema programskim alatima. Ulazno - izlazne procedure programskih paketa specificne namjene vecinom su realizirane tako da citaju i pišu po ASCII datotekama. Medutim kod ovakvog pristupa ostaje puno više posla pri realizaciji potrebnih aplikacija za modeliranje strukture baze, osiguranje jedinstvenosti adrese parametra i kontrolu pristupa. Ako se nazivi i adrese parametara (misli se na adrese u logickoj strukturi baze) zapisuju u ASCII datoteke teže je realizirati i mehanizme zaštite od nehoticnih grešaka pri upisivanju vrijednosti.

Kao ilustracija prethodnih razmatranja, na slici 29 prikazan je prijedlog osnovne strukture tablica i relacija kao implementacije strukture baze parametara u okruženju relacijske baze.



Slika 29: Prijedlog realizacije baze parametara u relacijskoj bazi

Predloženu strukturu u relacijskoj bazi podataka moguce je realizirati na nacin da su svi podaci parametara zapisani u jednoj tablici, što olakšava manipulaciju s adresom parametra i operacije pretraživanja.

Pri realizaciji predloženog modela pokazalo se ipak da objektna baza ima najviše prednosti, pogotovu u modeliranju vrlo složenih struktura relacija izmedu parametara i drugih klasa objektnog modela. Pored toga objektna baza efikasno rješava i probleme višestrukog referenciranja i kontrole pristupa parametrima. Detaljnije su ove teme razmotrene u osmom poglavlju.

7.1.2.4 Kontrola pristupa bazi parametara i zapisivanje vrijednosti

Baza parametra struktura je podataka i pripadajucih operacija koje moraju osigurati siguran rad i kontrolu pristupa u mrežnom okruženju. Ovdje cemo nabrojati neke od situacija za koje treba realizirati mehanizme kontrole pristupa.

- U jednom trenutku samo jedan korisnik (aktivni objekt) može mijenjati strukturu (hijerarhiju grupa) unutar baze, ostali objekti sustava za to vrijeme ne mogu imati pristup niti za pisanje niti za citanje.
- Istovremeno više objekata može imati pristup za pisanje, ako pri tome ne mijenjaju strukturu, i pri tome upisuju vrijednosti razlicitih parametara.
- Pristup za citanje može imati više objekata istovremeno.

Poseban je problem kontrola pristupa i mijenjanje vrijednosti parametara koji se obraduju u više konstrukcijskih zadataka koji se izvode paralelno i koji su "informacijski spregnuti", što znaci da se vrijednost jednog te istog parametra može mijenjati u svakom od tih zadataka. Ako bilo koji od spregnutih zadataka treba promijeniti vrijednost parametra, potrebno je da o tome obavijesti i ostale. Pored toga, za takve je slucajeve potrebno realizirati mehanizme komunikacije izmedu zadataka u smislu traženja suglasnosti za takve promjene - da li ostali zadaci mogu prihvatiti novu vrijednost parametra? Za takve je situacije predviden zapis prijedloga i argumenata kao dio strukture parametra. Ova problematika detaljnije je razradena i modelirana u [121]. Noviji pristup ovoj problematici razraden je u [119].

Navedena situacija takoder se razmatra i u poglavlju o matrici prikaza informacijske zavisnosti konstrukcijskih zadataka. Informacijski spregnuti konstrukcijski zadaci izvode se u iterativnim ciklusima, a takve situacije karakteristicne su za procese istovremenog inženjerstva (eng. concurrent engineering).

Potrebno je razmotriti još jedan dodatni problem vezan za sucelja prema bazi parametara, sadržaj baze i kontrolu pristupa parametrima. Tradicionalni proceduralni programski alati (npr. numericki proracuni) u tijeku izvodenja procesa konstruiranja obavljati ce u vecini situacija transfer vrijednosti skupova parametara samo povremeno, u diskontinuiranim iterativnim ciklusima.
Transfer vrijednosti parametara prema CAD modelima daleko je složeniji proces. Konstruktor u pravilu radi na jednom CAD modelu u kontinuiranom procesu kroz duže vrijeme. Pri tome može primjenom naredbi CAD paketa promijeniti vrijednosti nekih geometrijskih parametara mnogo puta. Postavlja se pitanje da li je nužno svaku takvu promjenu odmah zapisati u bazu parametara? Vjerojatno takav pristup ne bi bio isplativ, a bilo bi ga i vrlo teško realizirati.
Dodatan problem za takve situacije je pracenje dinamike promjena vrijednosti parametara. Promjena vrijednosti jednog geometrijskog podatka u parametriziranom CAD modelu uzrokuje promjene vrijednosti svih ostalih podataka koji su s njime povezani geometrijskim relacijama i ogranicenjima.

Kako u navedenim situacijama realizirati zapis, tj. transfer vrijednosti parametara u bazu?

Aplikacija (sucelje) koja povezuje CAD model i bazu parametara svakako treba obaviti transfer vrijednosti prilikom svakog otvaranja i zatvaranja (spremanja) CAD modela. Ovdje pretpostavljamo da se koriste parametarski CAD sustavi koji koriste interne varijable za geometrijske parametre modela. Neki vrhunski CAD paketi danas vec imaju module koji omogucavaju kontrolu pristupa i "vlasništva" nad komponentama složenih sklopova u okolinama timskog rada.
Za pocetnu fazu razvoja baze parametara dovoljno je realizirati upisivanje u status vrijednosti parametra da je "aktivan u otvorenom CAD modelu". To znaci da je pristup do vrijednosti takvog parametra onemogucen sve dok se doticni CAD model ne zatvori, odnosno obavi transfer vrijednosti svih parametara iz baze koje je koristio. Ovdje ostaje otvoreno pitanje kako riješiti situaciju istovremenog rada na više CAD modela koji sadrže nekoliko zajednickih (istih) parametara.
Da bi predloženi mehanizam funkcionirao, svako sucelje CAD modela mora imati tocnu evidenciju o pripadajucim parametrima.
Medutim, tu se javlja novi problem: novogenerirane parametre CAD modela (koji nastaju u tijeku rada s modelom) treba definirati u bazi i u sucelju CAD modela i baze. To znaci da konstruktor paralelno sa radom na CAD modelu, mora koristiti i sucelje modela prema bazi parametara, ili "pamtiti" sve nove parametre, pa ih naknadno upisivati u bazu preko sucelja baze, i pored toga ažurirati listu parametara u sucelju modela. U oba slucaja za ispravan transfer vrijednosti odgovoran je konstruktor koji radi na CAD modelu.
Nacin rješavanja razmatranih problema bitno ovisi i o faktorima konkretnog okruženja procesa konstruiranja:

- mogucnostima konkretnog CAD sustava koji se koristi u odredenom okruženju
- nacinu kreiranja CAD modela, odnosno korištenja CAD sustava (rijetko koji konstrukcijski ured koristi CAD sustav na "pravi" nacin, odnosno ne izvlaci maksimum mogucnosti sustava)
- dekompoziciji konstrukcije u CAD modele sklopova i podsklopova, raspodjeli zadataka unutar tima

7.1.2.5 Uloga baze parametara (rezime)

Rezimirajuci, još jednom cemo naglasiti osnovne zadace baze parametara:

- da osigura nadzor nad vrijednostima zajednickih atributa raznorodnih objekata, odnosno parametara konstrukcije koji se koriste u više razlicitih prikaza konstrukcije,
- da omoguci realizaciju mehanizma transfera podataka izmedu raznorodnih i neintegriranih aplikacija bez velikih zahvata u programski kod takvih aplikacija,
- da omoguci pregled "stanja razvoja" konstrukcije "na jednom mjestu", odnosno da nije potrebno pregledavati više raznorodnih dokumenata i modela da bi se vidjele trenutne vrijednosti i status kljucnih parametara konstrukcije.

Unapredenje organizacije manipulacije podacima konstrukcije i kontrole pristupa podacima konstrukcije jedan od prvih preduvjeta unapredenja organizacije i kvalitete izvodenja cjelokupnog procesa konstruiranja. To pogotovo vrijedi u okolinama timskog rada i istovremenog inženjerstva. Komponenta vremena u takvim situacijama vrlo je važna - cim je informacija definitivno definirana, treba ju odmah proslijediti dalje. Cesto je informacije važno staviti na raspolaganje vec i kad su u statusu "prijedloga".

U prethodnim razmatranjima navedeno je i nekoliko problema koje pri realizaciji modela baze parametara nije jednostavno riješiti. Može se onda postaviti pitanje da li baza parametara treba biti entitet modela procesa konstruiranja ili je bolje manipulaciju s atributima objekata riješiti na druge nacine? Mišljenje je autora da i pored navedenih problema treba izdvojiti manipulaciju s kljucnim podacima konstrukcije na jednom mjestu, jer takav pristup može znacajno unaprijediti organizaciju zapisa i tokova razmjene informacija u tijeku procesa konstruiranja. Daljnja detaljnija istraživanja trebala bi dati rješenja razmatranih problema.
Realizacija baze parametara kao entiteta objektno orijentiranog modela procesa konstruiranja zahtijevati ce citav skup klasa i operacija, a treba se i odluciti za jedan od ranije predloženih pristupa.

Na kraju treba napomenuti da se predložena struktura baze parametara ne bi trebala smatrati nadomjestkom sustava za upravljanje informacijama o proizvodu, tzv. EDM/PDM sustava (engineering / product data management). Vec je prije naglašavano da baza parametara ne treba sadržavati sve podatke o proizvodu, nego ona sadrži samo kljucne podatke za povezivanje razlicitih objekata prikaza konstrukcije, podatke koje dijele informacijski spregnuti zadaci i podatke koji kolaju izmedu raznorodnih programskih alata.
Bazu parametara najbolje bi mogli opisati kao univerzalno sucelje za sve podatke koji trebaju biti izvan dosega objekata prikaza konstrukcije kojima pripadaju. Takvi podaci mogu se promatrati i kao "izdvojena proširenja" atributa objekata prikaza konstrukcije, ili "atributi modelirani kao objekti".
Dakle baza parametara nije repozitorij svih informacija proizvodnog sustava, nego samo zajednicko sucelje objekata modela procesa konstruiranja.

7.1.3 Prikaz proizvoda (skup informacija o proizvodu)

Proizvod procesa konstruiranja je skup svih informacija potrebnih da se proizvod izradi i pravilno eksploatira. Svaka složenija konstrukcija sadrži u tom skupu informacija veliki broj raznorodnih dokumenata, odnosno zapisa informacija, npr.: projektni dokumenti (preliminarne sheme i dispozicije, elaborati), montažni i radionicki crteži, proracuni, sastavnice, tehnološke upute i postupci, podaci za narudžbu standardnih dijelova, upute za rukovanje i održavanje, itd. Skup svih informacija o proizvodu sacinjen je dakle od raznorodnih oblika zapisa i organizacije informacija, koji nastaju u razlicitim fazama procesa konstruiranja. Pri tome se mnoge informacije pojavljuju u više dokumenata (zapisa). Svaki konstrukcijski ured ima organizirani sustav identifikacije i pohrane dokumentacije. U takvim sustavima obicno postoje i interni propisi nacina kreiranja i tokova kolanja dokumenata. Ovisno o znacajkama proizvoda i tehnologije proizvodnje takvi sustavi znatno variraju jer su prilagodeni konkretnim potrebama. Najveci raspon razlika našao bi se izmedu nezavisnih projektnih ureda i konstrukcijskih ureda unutar proizvodnih sustava.

Opcenito gledano, skup informacija o proizvodu sastoji se od niza podskupova raznorodnih oblika zapisa, unutar kojih se neke informacije preklapaju, tj. pojavljuju se u više dokumenata, zapisane u razlicitim oblicima. Sustavi organizacije takvih skupova informacija variraju i specificni su za odredeno okruženje odvijanja procesa konstruiranja.

Zapisi informacija o proizvodu temeljni su entiteti u "realnom svijetu" procesa konstruiranja kojeg razmatramo i modeliramo, jer su oni proizvod, odnosno cilj procesa konstruiranja. Stoga ih treba apstrahirati i modelirati kao objekte, ali prije toga treba razmotriti što takvi objekti predstavljaju i koja je njihova uloga u modelu procesa konstruiranja. Kako nazvati takve objekte? Npr. pogodan naziv u engleskom jeziku bio bi "product information object". Ovaj naziv nema prikladnog doslovnog prijevoda, pa cemo koristiti naziv "prikaz proizvoda" ("objekt prikaza proizvoda") ili "skup informacija o proizvodu". Pri tome se podrazumijeva da je "prikaz proizvoda" odredeni podskup cjelokupnog skupa informacija o proizvodu.

Entitet "prikaz proizvoda" modelira odredenu vrstu zapisa informacija o proizvodu. Modelira postojanje, odnosno enkapsulira sve pojmove i dogadaje iz "životnog ciklusa" odredenog nosioca zapisa informacija o proizvodu. Objekt "prikaz proizvoda" sadrži operacije sucelja za transfer i generiranje podataka iz skupa informacija koji modelira.

Takav objekt model je konkretnog dokumenta ili skupa podataka, odnosno nosioca informacija o proizvodu. Pod "modelom" dokumenta ovdje se podrazumijeva:

- identifikacija dokumenta unutar modela procesa konstruiranja
- skup operacija koje cine sucelje "stvarnog dokumenta" (tj. zapisa skupa informacija) i objektnog modela procesa konstruiranja
- skup referenci na parametre u bazi parametara i skup atributa objekta, odnosno svi podaci koji cine doticni podskup informacija o proizvodu
- dogadaji (stanja) koji cine "životni ciklus" dokumenta - definiranje (otvaranje), izrada, završetak, kontrola, odobravanje, pohrana u arhivi, prosljedivanje na tehnološku razradu ili u proces proizvodnje

Primarna uloga objekta prikaza proizvoda je da "enkapsulira" sve navedene informacijske aspekte i aspekte ponašanja odredene klase dokumenta odnosno zapisa informacija o proizvodu. Pri modeliranju objekata prikaza proizvoda potrebno je klasificirati sve vrste

zapisa informacija o proizvodu, izdvojiti zajednicke atribute i operacije, te definirati hijerarhiju klasa na vrhu koje bi bila genericka klasa.

U vecini slucajeva informacije o proizvodu biti ce zapisane na racunalu, ali ovaj entitet treba moci modelirati i druge oblike zapisa informacija.

Ako objekt predstavlja racunalni zapis informacija, tada može i sadržavati taj zapis (djelomicno ili u potpunosti), ali može funkcionirati i kao sucelje prema aplikaciji koja sadrži i obraduje racunalni zapis (npr. CAD paket).

Ako objekt predstavlja skup informacija koji nije zapisan na racunalu, tada ce funkcija takvog objekta najcešce biti samo "administrativne" prirode - da modelira "postojanje" i cirkulaciju takvog dokumenta u procesu konstruiranja.

Objekt prikaza proizvoda može sadržavati, (ali ne i nužno) informacije i o procesu konstruiranja proizvoda. Ovi objekti su složene strukture (kompozitni objekti) i mogu sadržavati i druge objekte iz skupa entiteta modela procesa konstruiranja.

Kriteriji klasifikacije i nacini realizacije konkretnih podklasa prikaza proizvoda bitno ce ovisiti o vrsti i razini korištenja postojece programske podrške procesu konstruiranja u odredenom realnom okruženju. Na primjer ako se u konkretnom okruženju koristi CAD programski paket koji sadrži podršku za kreiranje strukturne sastavnice proizvoda, onda nece biti potrebno kreirati sastavnicu kao posebnu klasu prikaza proizvoda.

Na objektima prikaza proizvoda izvodi se najveci dio operacija u procesu konstruiranja. Na kraju procesa skup ovih objekata sacinjava dokumentaciju o konstruiranom proizvodu, odnosno skup svih potrebnih informacija za proizvodnju i eksploataciju proizvoda.

Klasifikacija objekata prikaza proizvoda tema je slijedeceg poglavlja. Ovdje cemo navesti zajednicke atribute i operacije koje treba imati svaka klasa, a koje nasljeduje od generické klase na vrhu hijerarhije klasifikacije.

7.1.3.1 Atributi klase "prikaz proizvoda"

Skup osnovnih atributa objekata prikaza proizvoda: $D = \{i, o, k, p, u, s, R\}$
pri cemu je:

- *i*::= identifikator u sustavu klasifikacije i pohrane dokumenata
- *o*::= opis
- *k*::= odgovorna osoba - referenca (pokazivac) na konstruktora
- *p*::= putanja do datoteke (ili direktorija, ako prikaz sadrži skup datoteka) - koristi se za racunalni zapis informacija o proizvodu
- *u*::= dostupnost, odnosno ogranicenja upotrebe, (skup referenci na konstruktore) - zapisuje se tko ga smije mijenjati, a tko samo citati
- *s*::= status objekta odnosno dokumenta (ili stanje u životnom ciklusu)
- *R*::= skup referenci ("pointera") na parametre u bazi parametara (slika 30)

Pri tome *s* poprima tocno jednu vrijednost iz skupa mogucih stanja *S:*
$s \in S$, gdje je $S = \{s_1, s_2, s_3, s_4\}$

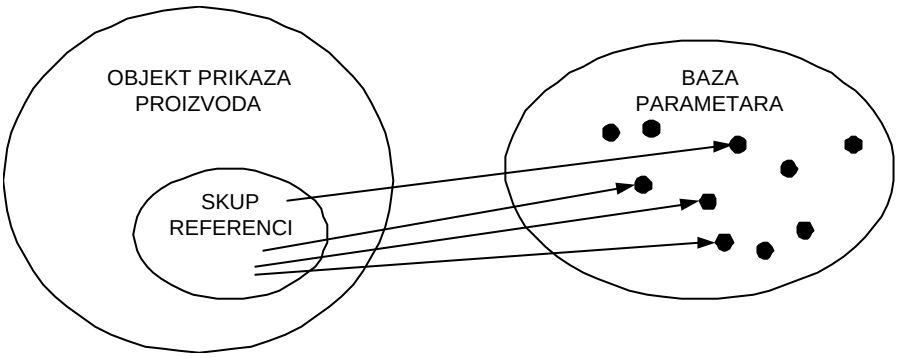
s_1::= aktivna je obrada dokumenta
s_2::= dokument završen, ceka odobrenje
s_3::= dokument završen i odobren
s_4::= dokument zatvoren (izmjene moguce samo uz posebno odobrenje ili dogovor)

7.1.3.2 Operacije klase "prikaz proizvoda"

- Skup operacija koje trebaju biti dio procesa generiranja nove instance objekta:
  - dodjeljivanje identifikatora, inicijalizacija statusa, referenciranje parametara, generiranje i inicijalizacija parametara, referenciranje odgovorne osobe.
- Upravljanje referencama na parametre.
- Skup operacija sucelja prema nosiocu informacija kojeg modelira (za transfer vrijednosti prema parametrima iz baze, slika 32).

7.1.3.3 Relacije (asocijacije) klase "prikaz proizvoda"

Objekt prikaza proizvoda sadrži skup referenci na parametre u bazi parametara (slika 30, analogno slici 27).
Na taj nacin se operacijama unutar objekta prikaza proizvoda omogucuje pristup do parametara koji su modelirani kao zasebni objekti. Ovakav skup referenci može se promatrati kao relacija izmedu instance prikaza proizvoda i skupa instanci parametara, logicki organiziranih u bazi parametara. Detaljna razrada relacija izmedu razlicitih klasa objekata prezentirana je u poglavlju 7.2, a realizacija ovako postavljenih relacija u objektnoj bazi prezentirana je u osmom poglavlju. Stoga se shema na slici 30 ovdje daje samo kao uvod u daljnja razmatranja, bez detaljnije razrade.



Slika 30: Skup referenci na parametre u bazi parametara kao podskup objekta prikaza proizvoda

7.1.3.4 Primjeri objekata prikaza proizvoda i klasifikacija

Za ilustraciju prethodnih razmatranja, odnosno koncepta objekta prikaza proizvoda, navesti cemo i razmotriti primjere takvih objekata. Ponoviti cemo da takvi objekti nisu sami racunalni zapisi (kao npr. CAD model), nego u vecini slucajeva njihova sucelja.
Npr. 2D crtež postoji kao datoteka i kao iscrtani dokument na papiru. Radi se zapravo o istom skupu informacija, zapisanom na razlicite nacine, odnosno na razlicitim medijima. U oba slucaja zapisa treba identificirati i "administrirati" postojanje takvog zapisa, a u slucaju racunalnog zapisa, potrebno je realizirati operacije sucelja prema bazi parametara i drugim objektima modela procesa konstruiranja. Ako se radi o parametarskom 2D crtežu tada ce se u odredenoj fazi procesa konstruiranja pokrenuti operacije sucelja (sadržane u objektu prikaza proizvoda) koje ce izvršiti transfer vrijednosti parametara iz baze parametara prema internim varijablama crteža i na taj nacin nastati ce nova varijanta crteža. Takav novi crtež treba dobiti svoj broj (kao novi dokument), treba biti iscrtan (isplotan), pohranjen u arhivi i

proslijeden dalje u tehnološki odjel. Uloga objekta prikaza proizvoda koji modelira takav dokument je dakle da "administrativno" prati njegov "životni ciklus" i da mu osigura potrebne operacije sucelja prema bazi parametara i prema drugim objektima modela procesa konstruiranja.

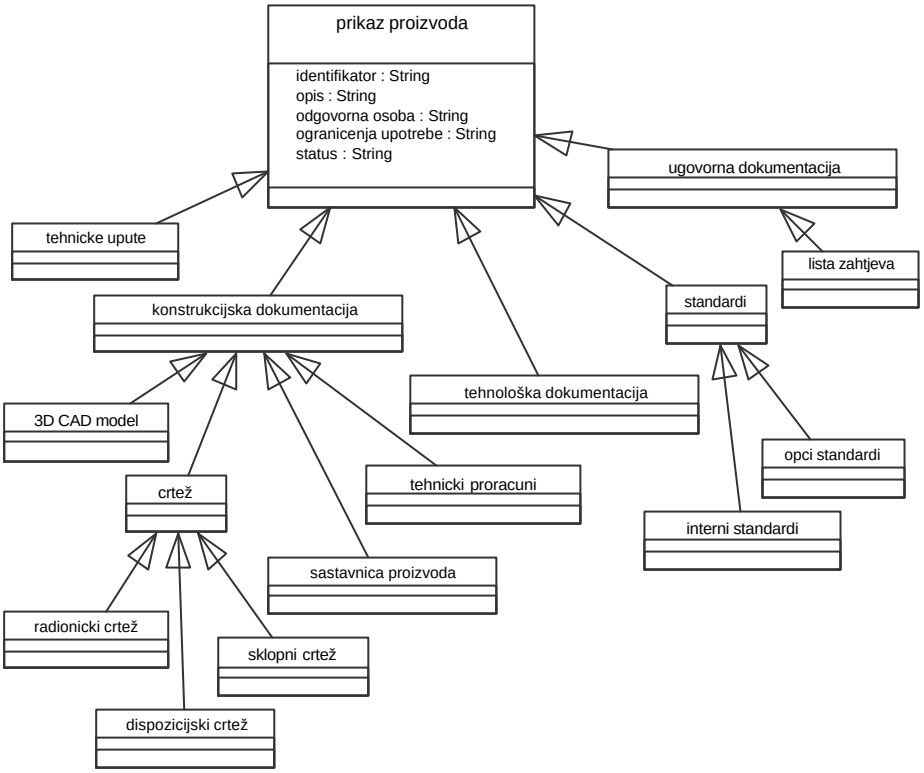
Razjasnimo još što se podrazumijeva pod "administrativnim pracenjem životnog ciklusa". Da bi se efikasno unaprijedila organizacija procesa konstruiranja i da bi se konstruktora oslobodilo najveceg dijela rutinskih aktivnosti, potrebno je razviti aplikacije koje ce automatizirati operacije manipulacije s dokumentima. Jedan primjer takve aplikacije je automatsko dodjeljivanje jedinstvene oznake svakom novokreiranom dokumentu, te evidencija svih dokumenata u zajednickoj bazi podataka cijelog proizvodnog sustava. Postoji citav niz programskih sustava koji pokrivaju takve namjene (EDM/PDM sustavi), medutim to su opcenite "ljuske" ili "toolbox-ovi", realizirane vecinom u relacijskim bazama podataka, te je potrebno uložiti dosta dodatnog rada pri njihovom prilagodavanju i implementaciji u konkretno okruženje. Važniji nedostatak takvih sustava u razmatranom kontekstu je cinjenica da nisu primarno orijentirani na proces konstruiranja nego pretežno na poslovne (financijske i informacijske) aspekte proizvodnog ciklusa. Stoga je u nekim slucajevima isplativije razvijati takve sustave pojedinacno, prema pravilima organizacije dokumentacije konkretnog okruženja. Primjer takvog sustava realiziran je kao dio projekta [122], [123]. Navedeni projekt je sustav baza podataka konstrukcijskog ureda koje sadrže sve podatke o dokumentaciji proizvoda (oznake, stanje dokumenata, rokovi) i podatke o raspodjeli zadataka izmedu konstruktora. Sustav je realiziran u mrežnom okruženju, korištenjem "intranet" tehnologije.

Navedimo sada primjere dokumenata, odnosno zapisa informacija koji se mogu modelirati kao objekti prikaza konstrukcije:

- zahtjevi narucitelja, (podaci ugovora)
- projektni podaci, preliminarni dispozicijski crteži i sheme
- tehnicki proracuni (ulazno-izlazne liste), kalkulacije troškova, elaborati
- 3D CAD model - dio i/ili sklop, 2D crtež, sastavnica (osnovni dokumenti za proizvodnju)
- podaci za narudžbu standardnih i katalоških dijelova i materijala
- gradevinska dokumentacija, studije utjecaja na okoliš (za procesna postrojenja)
- tehnološke upute za proizvodnju
- tehnicke upute za eksploataciju i održavanje
- interni standardi proizvodnog sustava, katalozi internih standardiziranih dijelova
- katalozi dobavljaca, katalozi standardnih dijelova

Zadnja dva navedena primjera nisu kao skupovi informacija direktni proizvodi procesa konstruiranja, ali se njihove komponente koriste u cjelokupnom prikazu proizvoda.

Razrada klasifikacije objekata prikaza proizvoda dovela bi do upotrebe konkretnijih naziva objekata (klasa) kao npr. dokument, crtež, proracun cvrstoce, itd. Takve klasifikacije pogodnije je napraviti prema specificnim potrebama konkretnog okruženja. U svakom slucaju, "objekt prikaza proizvoda" možemo promatrati kao genericku (apstraktnu) klasu na vrhu klasifikacijske hijerarhije. Jedan primjer prijedloga klasifikacije zapisa informacija o proizvodu prikazan je na slici 31. Jednostavni primjeri objekata prikaza proizvoda i njihovo povezivanje sa bazom parametara prikazani su na slici 32.



Slika 31: Dijagram prijedloga klasifikacije prikaza proizvoda

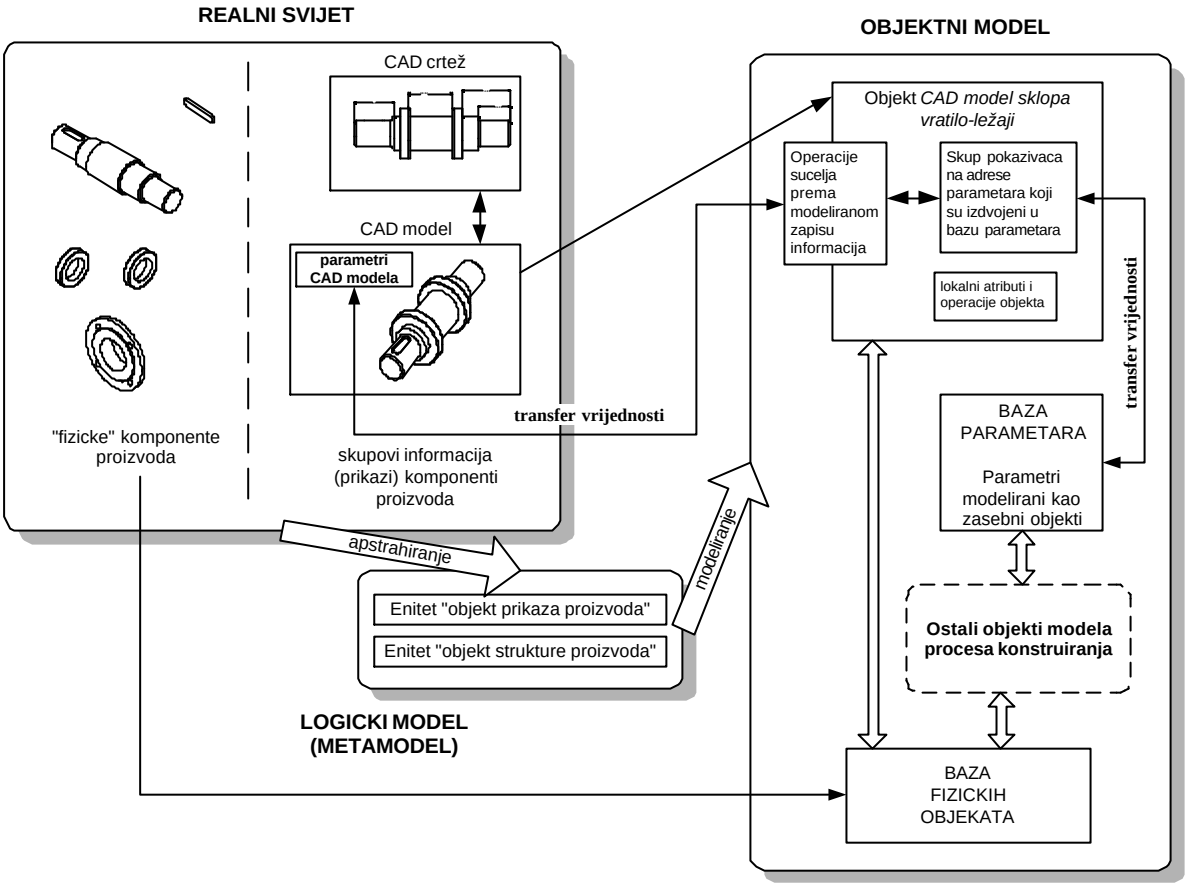
7.1.3.5 Kreiranje objekata prikaza proizvoda

Objekti prikaza proizvoda kreiraju se u svim fazama procesa konstruiranja. Ako se radi o potpuno novoj konstrukciji, objekti se mogu kreirati usporedo s razradom funkcionalne strukture proizvoda.
Objekt prikaza proizvoda nastaje zapravo s kreiranjem nove instance odredene klase zapisa informacija o proizvodu. Vjerojatno nije uvijek nužno nakon kreiranja novog dokumenta (odn. zapisa informacija) kreirati i objekt koji ga "predstavlja" unutar objektnog modela procesa konstruiranja. Kreiranje objekta prikaza proizvoda nužno ce biti u situacijama gdje je potrebno koristiti operacije sucelja prema bazi parametara i prema drugim objektima modela procesa konstruiranja.
Kod ponovljenih i varijantnih konstrukcija zapisi informacija (odnosno dokumenti) mogu se kreirati na temelju postojecih uzoraka i predložaka koji se dograduju i mijenjaju (npr. "skeleton, layout, template") ili se kao osnova mogu koristiti pohranjeni zapisi vec izvedenih konstrukcija. U takvim situacijama klase objekata prikaza proizvoda mogle bi se realizirati kao parametarske, što bi olakšalo kreiranje, odnosno instanciranje objekata.

7.1.3.6 Modeliranje prikaza proizvoda

Razmatranje objekta prikaza proizvoda zakljuciti ce se primjerom principa modeliranja pojmova iz realnog svijeta preslikavanjem entiteta iz "realnog svijeta" u logicki i objektni model (shema na slici 22). U ovom primjeru izdvojeno je modeliranje skupa informacija o

proizvodu i modeliranje "fizicke komponente" proizvoda, uz naglašavanje semanticke razlike izmedu ta dva pojma. Ujedno su prikazane i interakcije i reference objekta prikaza proizvoda sa ostalim objektima modela procesa konstruiranja.



Slika 32: Preslikavanje informacija o proizvodu u logicki i objektni model

Skupovi informacija o proizvodu preslikavaju se u objekte prikaza proizvoda. Jedna od primarnih funkcija objekta prikaza proizvoda je osigurati transfer vrijednosti podataka iz zapisa informacija o proizvodu prema bazi parametara (i obrnuto). U konkretnom primjeru odredeni skup parametara CAD modela treba modelirati kao parametre (odnosno objekte) u bazi parametara. Objekt prikaza proizvoda koji modelira konkretni CAD model sadržavati ce skup (niz) pokazivaca (pointera) na adrese objekata (parametara) u bazi parametara. Sam transfer vrijednosti navedenih parametara bitno ce ovisiti o vrsti zapisa informacija o proizvodu, odnosno o mogucim nacinima ucitavanja i eksportiranja podataka. Stoga za svaku vrstu zapisa informacija o proizvodu treba postojati posebna podklasa objekta prikaza proizvoda koja ce sadržavati odgovarajuce operacije transfera podataka prema bazi parametara. Ostali objekti prikaza proizvoda, kao i druge klase objekata mogu pristupati parametrima u bazi koji pripadaju trenutno aktivnom objektu prikaza proizvoda, pri cemu atributi stanja svakog parametra odreduju kontrolu pristupa.

Slika 32 istice razliku u modeliranju skupova informacija o proizvodu i modeliranju "fizickih" komponenti proizvoda (što se u svakodnevnoj praksi cesto poistovjecuje). Skup informacija o proizvodu može sadržavati informacije o jednom ili više fizickih objekata. Isto tako jedan fizicki objekt (ili skup fizickih objekata) mogu biti opisani sa više raznorodnih skupova informacija.

Fizicke komponente proizvoda kao objekte modela procesa konstruiranja potrebno je modelirati u situacijama koje su navedene u slijedecem poglavlju.

7.1.4 Struktura proizvoda (objekt iz fizicke domene)

Dokumenti, odnosno sve vrste zapisa informacija cine skup podataka o proizvodu. Pored informacija o proizvodu, u proizvodnom sustavu kao entiteti egzistiraju i dijelovi i sklopovi proizvoda kao stvari, "fizicki" objekti. Npr. razlikujemo "vratilo" i "crtež vratila". Neki objektni pristupi (npr.[124], [125]) konstruiranju razlikuju "fizicke" i "nefizicke" entitete i tretiraju ih ravnopravno. Postoje i pristupi u kojima se ne razmatra modeliranje zapisa informacija kao objekata nego se orijentiraju samo na modeliranje i odnose "fizickih" objekata konstrukcije [126], [127].
Po mišljenju autora nužno je promatrati i modelirati oba aspekta i pri tome omoguciti jednostavne mehanizme preslikavanja iz fizicke u "informacijsku" domenu i obratno. Ako se u ovim razmatranjima oslonimo na Suh-ovu aksiomatsku teoriju konstruiranja, tada možemo objekte promatrati u cetiri domene: domena potrošaca, funkcionalna domena, fizikalna domena i domena procesa (slika 12). Suh tretira proces konstruiranja kao cik-cak preslikavanja izmedu navedenih domena. Objekte prikaza proizvoda možemo pridružiti Suh-ovoj domeni procesa, a objekti koji modeliraju komponente proizvoda pripadali bi u fizikalnu domenu.

Razmotrimo u kojim situacijama u procesu konstruiranja ce trebati koristiti objekte prikaza iz fizicke domene, odnosno kada se javlja potreba za takvim objektima?
Pretežno ce to biti situacije u kojima treba vezati odnosno skupiti ("enkapsulirati"), zapise informacija i operacije uz pojam konkretnog fizickog objekta. Primjeri takovih situacija mogu biti npr:

- zapis referenci na parametre koji sudjeluju u interakcijama (meduzavisnostima) s drugim objektima fizicke domene
- zapisi informacija testiranja i ispitivanja podsklopova ili cijelog gotovog proizvoda, zapisi povratnih informacija iz proizvodnje i eksploatacije
- skup informacija za dijelove koje rade subliferanti
- zapisi tehnoloških procesa izrade i sklapanja
- formiranje objekta cija je uloga modeliranje i pracenje "životnog ciklusa" dijela u proizvodnom sustavu - takvi objekti definiraju se u odjelu konstrukcije, ali se koriste dalje u cijelom proizvodnom ciklusu
- podaci o internim standardnim i kataloškim dijelovima - sve situacije korištenja standardnih dijelova

Druga mogucnost korištenja objekata prikaza fizicke domene je modeliranje skupova kompleksnih relacija koje se ne odnose na hijerarhiju sklopova i podsklopova. Ovdje se prvenstveno misli na funkcionalne relacije - odnose dijelova i sklopova u ispunjavanju parcijalnih funkcija tehnickog sustava. Primjer takvog pristupa predložen je u radu [128].

Modeliranje svakog pojedinog dijela složene konstrukcije samo u svrhu jedinstvene identifikacije, generiralo bi vrlo veliki broj objekata. Stoga je mišljenje autora da objekte fizicke domene treba koristiti samo u ranije predloženim situacijama. Identifikaciju svakog pojedinog strojnog dijela jednostavnije je rješavati bazama podataka, odnosno komponentama EDM/PDM sustava.

Kriteriji klasifikacije objekata strukture proizvoda specificni su za svako konkretno okruženje izvodenja procesa konstruiranja. Stoga se ovdje predlaže samo jednostavna klasifikacija na dio, podsklop i sklop sa generickom klasom na vrhu hijerarhije. Modeli klasifikacije i relacija izmedu elemenata strukture proizvoda mogu se preuzeti i iz STEP standarda [103], [104], [105].

Kao osnovni atributi objekta strukture proizvoda predlažu se: identifikator, naziv, opis i status (u životnom vijeku).

Objekti strukture proizvoda mogu se povezati relacijama sa objektima prikaza proizvoda, i to relacijama "jedan prema više" u oba smjera, tj. jedan objekt prikaza proizvoda prikazuje više objekata strukture proizvoda, i s druge strane, jedan objekt strukture proizvoda je prikazan u više objekata prikaza proizvoda. Detaljnije ce ova relacija biti razradena u poglavlju 7.2 i u osmom poglavlju.

7.1.5 Akcija

Do sada prikazani entiteti objektnog modela procesa konstruiranja cine osnovu staticke (informacijske) strukture modela, odnosno možemo ih promatrati kao osnovne informacijske gradbene elemente. Jedan od ciljeva ovog istraživanja je modelirati prikaz dinamike (tijeka) odvijanja procesa. Stoga se uvodi entitet "akcija" kao jedan od osnovnih gradbenih elemenata prikaza procesa.

Entitet "akcija" modelira pozive operacija objekata modela procesa konstruiranja. Akcije se kreiraju i planiraju prije izvodenja procesa, a realiziraju se u tijeku izvodenja procesa konstruiranja.

Akcija se ovdje promatra u kontekstu generiranja, transformiranja i prikaza informacija (samo u informatickom kontekstu), a ne i u kontekstu metodologije i postupaka konstruiranja, kao što je npr. Hubkina klasifikacija aktivnosti i operacija u procesu konstruiranja (slika 5). Medutim tako koncipirani entitet "akcije" može poslužiti kao gradbeni element, tj. skup akcija može modelirati elementarne operacije s razine 5 na slici 5.
Složeniji kompozitni entiteti prikaza procesa konstruiranja, te njihove agregacije i asocijacije trebaju poslužiti kao osnova za modeliranje aktivnosti i operacija sa viših razina kompleksnosti (slika 5).
Akciju možemo promatrati kao element upravljanja izvodenjem procesa konstruiranja koji se ugraduje u proces u tijeku njegova izvodenja. Drugim rijecima akcije su elementi upravljanja procesom koji se unaprijed planiraju prije izvodenja procesa da bi se realizirale (ovisno o uvjetima i situacijama) za vrijeme izvodenja procesa konstruiranja.

7.1.5.1 Izvršavanje akcije u procesu konstruiranja

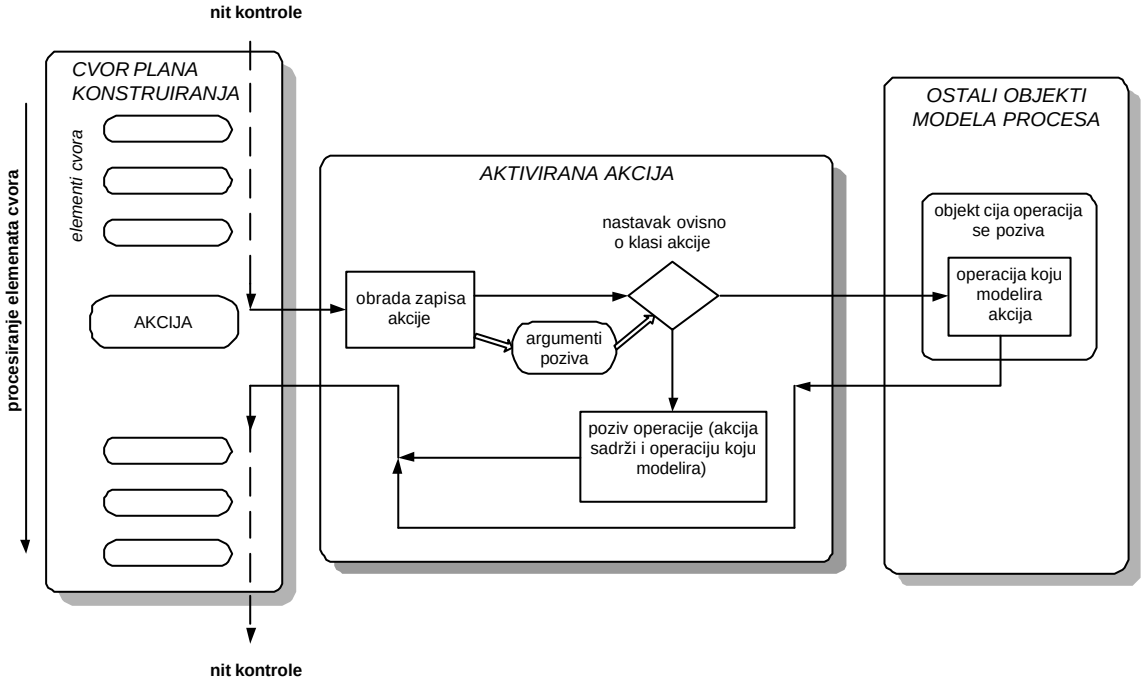
Postavlja se pitanje na koji nacin se može riješiti izvršavanje (realizacija) planirane akcije u tijeku izvodenja procesa konstruiranja. Vec je receno da ce akcija biti jedan od gradbenih elemenata složenijih entiteta prikaza procesa konstruiranja - cvora plana konstruiranja - koji je detaljno razraden u poglavlju 7.3.
Instanca akcije sadrži zapis akcije koji se opcenito sastoji od naziva objekta i procedure koju treba pozvati, te liste argumenata poziva. U tijeku izvodenja procesa konstruiranja treba takav zapis "dekodirati" u poziv odgovarajuce operacije tocno odredenog objekta, što

se može realizirati postupkom tzv. "parsiranja". Pojedine klase akcija mogu sadržavati još i neke dodatne specificne elemente sintakse zapisa akcije. Stoga, da bi se pojednostavnila procedura dekodiranja svaka klasa akcije treba imati svoju operaciju dekodiranja koja "zna" kako se "obraduje" samo njena varijanta sintakse zapisa.

Izvodenjem procesa konstruiranja treba upravljati procedura koja ima glavnu "nit kontrole" ("thread of control") nad cijelim sustavom, pa upravlja i procesom izvodenja cvora plana konstruiranja. Procesiranje cvora plana ukljucuje obradu elemenata cvora predvidenim redoslijedom. Cvor pristupa svojim elementima preko referenci na njihove adrese.
Dakle, kad procesiranje cvora naide na element "akciju", nit kontrole predaje se tom objektu tako da se poziva njegova operacija dekodiranja, tj. obrade zapisa akcije. Rezultat takve obrade zapisa akcije su parametri potrebni za poziv operacije ucitani u memoriju. Ovisno o klasi aktivirane akcije proces se može nastaviti na dva nacina:

- aktivirana instanca akcije poziva svoju operaciju (sa "inicijaliziranim" argumentima) - takva operacija specificna je za tu klasu akcije, tj. ona je njen "zadatak" - npr. ucitavanje objekta iz baze
- aktivirana instanca akcije poziva operaciju nekog drugog objekta modela procesa konstruiranja i njoj predaje nit kontrole, koju ova po izvršavanju vraca akciji.

U oba slucaja aktivirana instanca akcije nakon izvršavanja vraca nit kontrole procesu obrade cvora (procesiranju cvora).



Slika 33: Izvršavanje akcije u tijeku izvodenja procesa konstruiranja

Konstruktoru se na ovaj nacin omogucava da planira pozive operacija entiteta modela procesa konstruiranja, bez nužnog znanja programiranja, ali uz uvjet poznavanja argumenata i naziva operacija objekata predloženog modela. Kako su objekti modeli entiteta iz realnog svijeta procesa konstruiranja, ucenje korištenje predloženog sustava ne bi trebalo predstavljati problem, jer konstruktor ne mora poznavati detaljnu strukturu operacija koje koristi kao akcije u procesu, odnosno zapis akcije služi mu kao sucelje prema pozivima operacija.

Ovako modeliran entitet "akciju" možemo promatrati i kao sucelje izmedu operacija objekata i kontrole izvodenja procesa konstruiranja. Drugim rijecima, preko takvog sucelja proces izvodi planirane operacije na objektima.

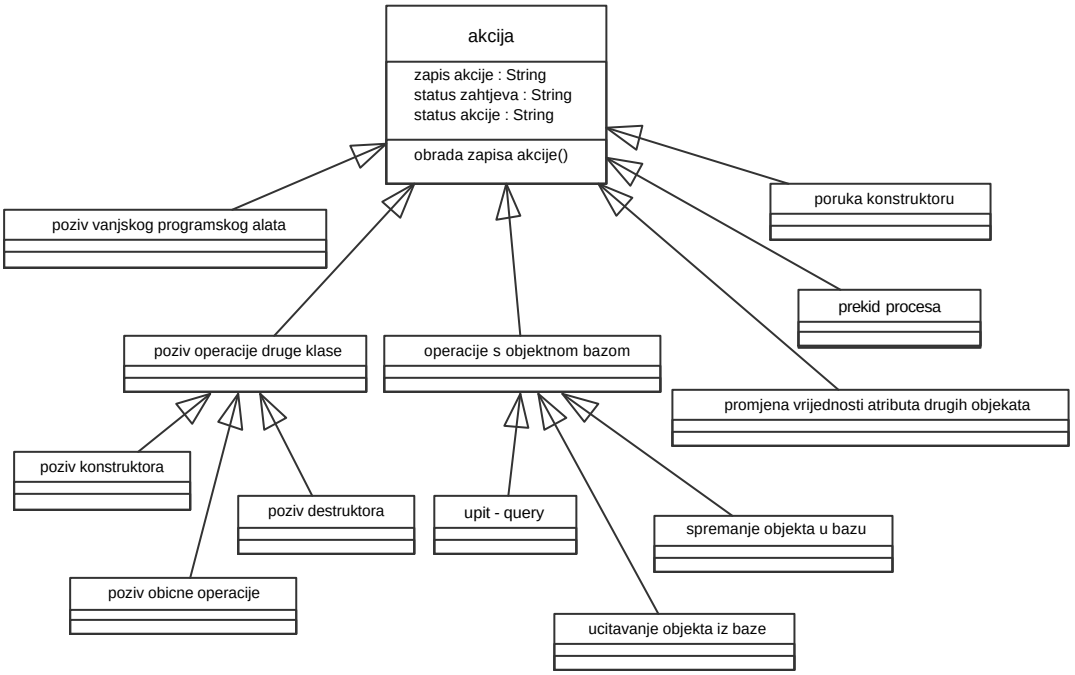
7.1.5.2 Klasifikacija akcija

Kako je vec navedeno akcija se ovdje modelira u kontekstu generiranja, transformiranja i prikaza informacija, pa se na tome temelji i predložena klasifikacija akcija. Dakle primarni kriterij klasifikacije je nacin djelovanja akcije na objekte modela procesa konstruiranja.

Predlažu se slijedece vrste djelovanja akcija:

- mijenjanje stanja jednog ili više objekata pridruživanjem novih vrijednosti atributima
- ucitavanje objekta iz baze, spremanje objekta u bazu
- poziv operacije objekta
- poziv "vanjskih" programskih alata (npr. CAD modela)
- kreiranje novih instanci klasa ili brisanje iz memorije nepotrebnih (poziv konstruktora i destruktora odredene klase)
- slanje poruka konstruktoru i obrada odgovora
- prikaz stanja procesa (stanja jednog ili više objekata)
- pregled skupova informacija u situacijama kad na temelju toga treba donositi odluke ili npr. provjeriti, kontrolirati, odobriti, sintetizirati, itd…

Hijerarhija klasa prikazana je na slici 34. Kao posebna grupa izdvojene su klase koje pozivaju operacije drugih klasa objektnog modela, te klase operacija sucelja prema objektnoj bazi. Promjena vrijednosti atributa drugih objekata vrijedi samo za one atribute kojima je dozvoljen pristup izvan njihove klase (deklarirani kao "public").



Slika 34: Prijedlog klasifikacije akcija

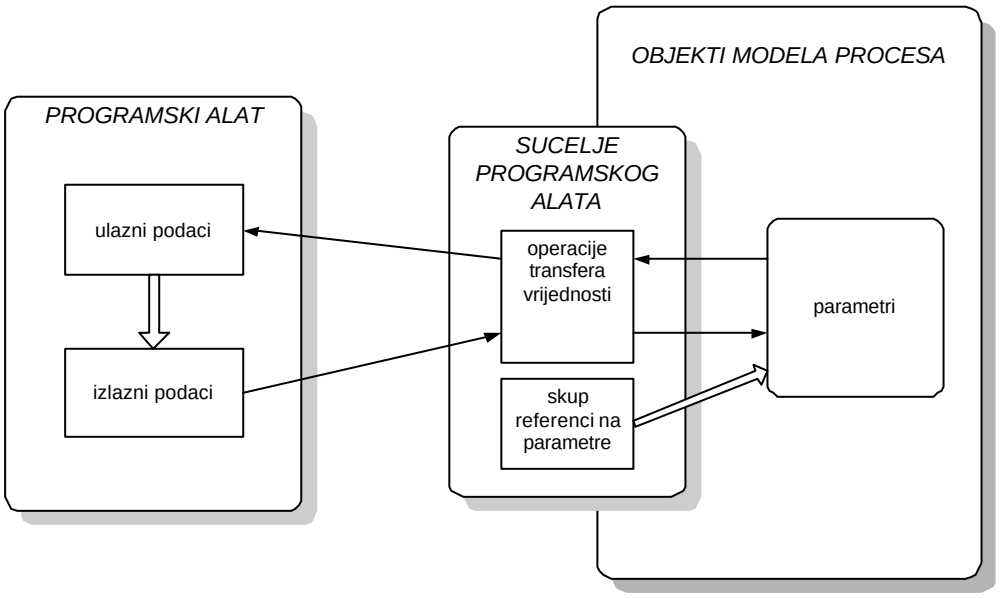
Osnovni atributi akcije su: zapis akcije (operacije koju treba pozvati) prema specificnoj sintaksi svake klase akcije, te status zahtjeva za izvodenjem akcije i status izvršavanja akcije. Osnovna operacija akcije koja se nasljeduje od generic klase je obrada (dekodiranje) zapisa, medutim svaka klasa akcije ima svoju implementaciju operacije obrade zapisa.

7.1.6 Sucelje programskog alata

U poglavlju o bazi parametara vec je razmatran transfer podataka iz objekata modela procesa konstruiranja prema "vanjskim programskim alatima". "Vanjskim" alatima smatraju se svi oblici racunalne podrške koja se koristi u procesu konstruiranja. Najcešce ce to biti razni numericki proracuni pisani u proceduralnim jezicima, no može biti i npr. tablicni kalkulator, ekspertni sustav, baza podataka, itd. Da bi se takvi oblici programske podrške integrirali sa objektnim modelom potrebno je koncipirati sucelja programskih alata prema objektima modela.

Entitet "sucelje programskog alata" enkapsulira skup operacija za transfer podataka izmedu objekata modela procesa konstruiranja i programskih alata koji nisu dio objektnog modela procesa konstruiranja.

Svaki pojedini "vanjski" programski alat treba imati svoju instancu sucelja koje ga pokrece i obavlja transfer podataka. Transfer podataka tece u oba smjera (slika 35). Sucelja moraju poznavati strukturu baze parametara kao i strukturu ulazno/izlaznih podataka i nacine dobave podataka programskog alata kojem pripadaju. Pri konkretnoj implementaciji treba posebno voditi racuna o ažuriranju sucelja pri eventualnim doradama programskih alata.



Slika 35: Transfer vrijednosti parametara preko sucelja programskog alata

Atributi sucelja programskog alata: naziv programskog alata, putanja do izvršne datoteke, putanje do ulaznih i izlaznih datoteka, status programskog alata (obzirom na dostupnost i pokretanje).

Osnovne operacije sucelja programskih alata su operacije transfera podataka u oba smjera.

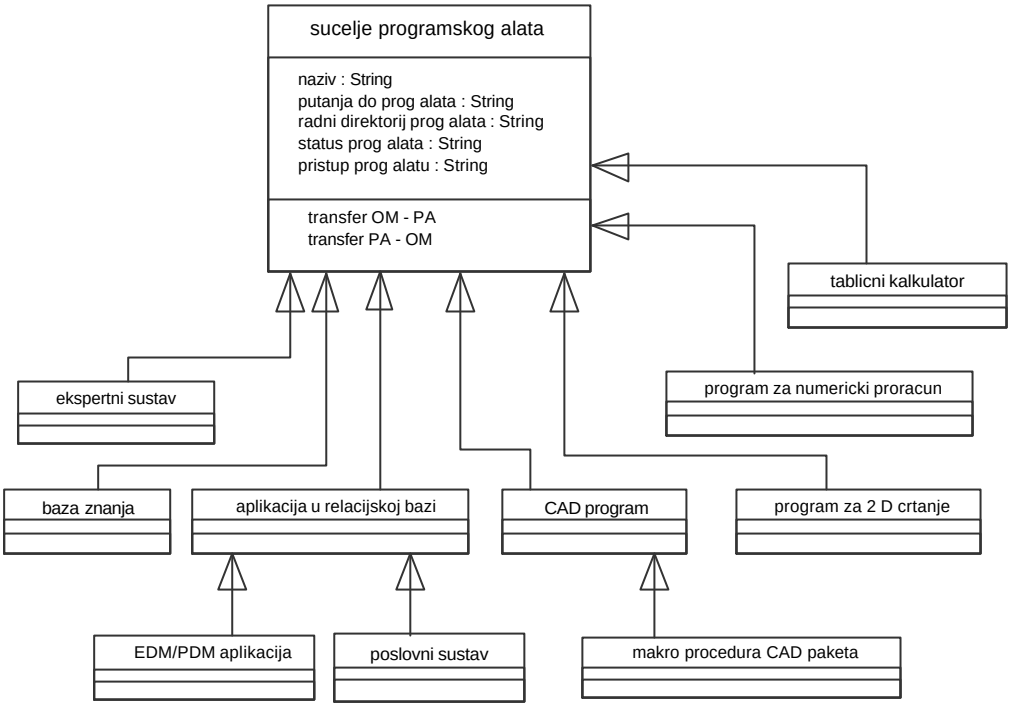
Vanjski programski alati pozivaju se korištenjem posebne klase akcije koja aktivira sucelje alata i predaje mu nit kontrole procesa. Instance takve klase akcija sadrže reference na pripadajuce instance sucelja.

7.1.6.1 Klasifikacija sucelja programskih alata

Klasifikaciju sucelja programskih alata treba provesti primarno prema kriteriju razlicitih nacina komuniciranja (pokretanja i transfera) sa vanjskim alatima koji nisu direktni dijelovi sustava modela procesa konstruiranja. Medutim treba uzeti u obzir i kriterije klasifikacije samih programskih alata. Prema oba navedena kriterija vrlo je teško precizno definirati klasifikaciju jer ona bitno ovisi o specificnostima konkretnog okruženja izvodenja procesa konstruiranja - koji programski alati se koriste, kojim metodama su razvijeni, te na kojem stupnju razvoja je korištenje racunalne podrške.

Stoga ce se ovdje predložiti nekoliko kriterija (aspekata) klasifikacije programskih alata koji se ne smatraju konacno definiranim niti potpunim:

- prema vrsti operacije koju programski alat obavlja, odnosno podržava :
  - preliminarni ili kontrolni proracun dimenzija i/ili cvrstoce strojnog dijela ili sklopa
  - kalkulacija troškova (materijala, izrade i sl.)
  - izbor izmedu razlicitih varijanti rješenja na temelju skupa pravila
  - generiranje crteža strojnog dijela ili sklopa
- prema fazi procesa konstruiranja u kojoj se primjenjuje :
  - u fazi koncipiranja vjerojatno ce se cesto koristiti ekspertni sustavi
  - za izbor nosioca parcijalnih funkcija mogu poslužiti baze znanja i katalozi principa rješenja
  - u fazi projektiranja pretežno ce se koristiti programi za tehnicke proracune i kalkulacije troškova
  - u tijeku konstrukcijske razrade pretežno se koristi komercijalni CAD paket i procedure razvijene u njegovom “makro” jeziku
- prema vrsti programskog jezika ili programske platforme:
  - program pisan u proceduralnom jeziku (npr. C, Fortran, Pascal); makro jezik komercijalnog CAD paketa; ljuska (shell) za ekspertni sustav; baza podataka; tablicni kalkulator, itd.
- prema nacinu zadavanja ulaznih i ispisa izlaznih podataka :
  - koriste se datoteke, sucelja za transfer koja su dio aplikacije ili kombinacije oba pristupa
- prema nacinu izvodenja i složenosti programskog alata :
  - samo jedna izvršna (“exe”) datoteka, ili više komponenti (dodatne dinamicke biblioteke)
  - posebni zahtjevi za okolinom izvodenja (to vrijedi za npr. makro procedure CAD sustava - mora biti aktivan i CAD sustav)



Slika 36: Klasifikcija sucelja programskih alata

Slika 36 dijagram je klasifikacije sucelja programskih alata po kriteriju vrste, odnosno namjene programskog alata. Predložena klasifikacija samo je primjer za ilustraciju prethodnih razmatranja. Preciznija klasifikacija može se odrediti tek pri implementaciji sustava u konkretno okruženje, prema njegovim specificnostima.

7.1.7 Zadatak

Nastojanja za unapredivanjem organizacije rada konstrukcijskog ureda promatraju i konstrukcijski zadatak u informatickom kontekstu. U takvom pristupu promatra se tok informacija izmedu zadataka, rokovi, resursi i odgovornosti.

Entitet "zadatak" modelira informacijske tokove i organizacijske aspekte realnog konstruktorskog zadatka u okruženju timskog rada.

Prema VDI [21] zadatak integrira slijedece informacije: ulazne podatke (zahtjeve), izlazne podatke - cilj (funkcije i karakteristike), rokove, troškove, i organizacijske podatke.

Objekt "zadatak" treba dakle enkapsulirati sve navedene skupove informacija kao skupove svojih atributa ili kao skupove referenci na druge objekte modela procesa konstruiranja.

Zadatak se može promatrati i kao dio dekompozicije procesa konstruiranja u okruženju timskog rada. Potreba za definiranjem zadatka kao objekta javljati ce se kod vrlo složenih konstrukcija, gdje na pojedinim sklopovima rade grupe konstruktora, te postoji i hijerarhija odgovornosti i rukovodenja unutar i izmedu grupa. U takvim slucajevima zadatak modelira i enkapsulira i sve potrebne organizacijske informacije. Pri tome jedan zadatak može rješavati jedan ili više konstruktora.

7.1.7.1 Sadržaj zadatka

Informacije koje zadatak treba integrirati (enkapsulirati) mogu se podijeliti na dvije osnovne grupe:

1. zahtjevi i ciljevi zadatka - mogu biti zapisani u nekoliko oblika:
   - zahtijevane funkcije i karakteristike dijela konstrukcije koji je predmet zadatka

   Za varijantne i ponovljene konstrukcije:
   - reference na parametre ciju vrijednost i/ili statuse treba odrediti (ti parametri vec postoje kao instance)
   - lista parametara koje treba generirati (kreirati) i odrediti im vrijednost
   - reference na vec postojece objekte prikaza proizvoda koje treba razraditi i lista objekata prikaza koje treba kreirati - sa željenim vrijednostima statusa takvih objekata
2. organizacijski podaci:
   - atributi zadatka: naziv, pocetak, planirani završetak, odgovorna osoba, status: nije pocet, u izradi, završen
3. zapisi informacijske zavisnosti o drugim zadacima u procesu - reference na druge zadatke koji su informacijski spregnuti sa odredenim zadatkom

Svi navedeni oblici zapisa zahtjeva i ciljeva zadatka ne mogu se za sve vrste konstrukcija unaprijed isplanirati, nego se oni formiraju u tijeku izvršavanja zadatka (parametri i objekti prikaza proizvoda).

Razrada relacija zadatka s drugim objektima modela procesa konstruiranja dana je u poglavlju 7.2, a implementacija je razradena u osmom poglavlju.

7.1.8 Konstruktor

Entitet "konstruktor" enkapsulira informacije i operacije vezane uz osobe iz konstrukcijskog tima, organizaciju i podjelu rada i odgovornosti, te hijerarhiju kontrole pristupa dokumentima koji su proizvod procesa konstruiranja. Skup instanci "konstruktora" modelira dogadaje i pojmove vezane za osobe i njihove interakcije kao "izvodace" procesa konstruiranja.

Ovaj objekt sadrži slijedece osnovne skupove referenci:
- na dodijeljene zadatke
- na objekte prikaza proizvoda i objekte strukture proizvoda za koje je odgovoran

Ovi skupovi referenci omogucavaju koncipiranje i kreiranje operacija za podršku planiranju i pracenju rokova i opterecenja u okruženju timskog rada. Takve operacije mogu dati pregled stanja (dovršenosti) zadataka i objekata prikaza proizvoda na kojima radi jedan konstruktor ili grupa konstruktora.

Osnovni atributi objekta "konstruktor": ime i prezime, funkcija, odnosno položaj u hijerarhiji rukovodenja.

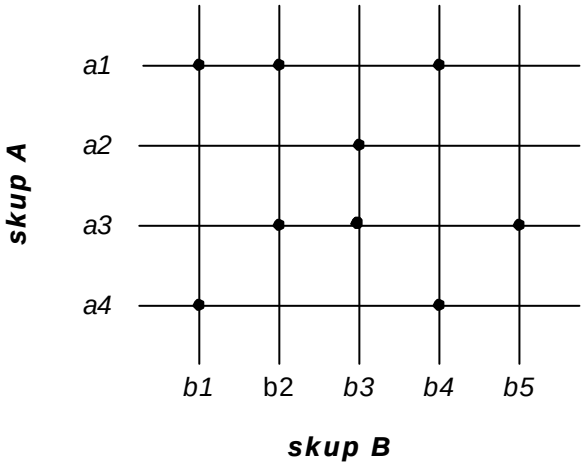
Eventualna klasifikacija može se provesti pri implementaciji sustava prema specificnostima konkretnog ureda i njegovoj organizaciji rada.

Prava pristupa pojedinog konstruktora odredenoj klasi objekata modela procesa konstruiranja mogu se zapisivati u obliku atributa svake klase. Asocijacije u obrnutom smjeru - zapisivanje u svakoj instanci konstruktora cemu sve ima pristup bilo bi kompliciranije realizirati, a i rezultirale bi velikim brojem referenci. Model organizacije prava pristupa dokumentima specifican je za svako konkretno okruženje, stoga se ova problematika detaljnije ne razmatra, odnosno treba ju rješavati pri implementaciji sustava.

Objekt "konstruktor" može sadržavati i atribute i operacije za modeliranje komunikacije (slanje poruka) medu clanovima tima. Takve poruke generiraju se direktno iz operacija, u tijeku izvodenja procesa konstruiranja (ovdje se ne misli na poruke koje šalje "konkretni" konstruktor izvan konteksta operacija objektnog modela procesa konstruiranja). Primjer takvih poruka je razmjena prijedloga i argumenata u situacijama izvodenja informacijski spregnutih zadataka. U cvoru plana mogu se unaprijed isplanirati akcije koje ce poslati poruke drugom konstruktoru, ovisno o nastaloj situaciji.

7.2 Relacije u objektnom modelu procesa konstruiranja

Prema [98] relacije su konceptualne ili konkretne tvorbe koje povezuju dva ili više drugih entiteta. Znacenje veze može se definirati na razlicite nacine unutar razlicitih semantickih modela [129].
Osnovni strukturalni elementi modela procesa konstruiranja medusobno su zavisni i povezani na mnogo razlicitih nacina, tj. topologija prostora relacija izmedu elemenata ima oblik mreže u kojoj je svaki cvor (koji prikazuje element) povezan sa vecinom drugih cvorova.
Stoga model procesa treba sadržavati posebne klase objekata za prikaz i manipulaciju sa relacijama izmedu entiteta modela. Takve relacije mogu biti izmedu gradbenih elemenata modela medusobno, kao i izmedu gradbenih i kompozitnih elemenata. Opcenito, model treba imati mogucnost zapisa relacija izmedu bilo koje dvije vrste elemenata.



Slika 37: Graficki prikaz binarne relacije

U teoriji skupova relacija se definira kao podskup Kartezijevog produkta dva skupa AxB:

$$R \subset AxB = \{ (a,b) \in R \mid a \in A, \; b \in B \}$$

Primjer grafickog prikaza relacije prikazan je na slici 37.

Kako klasificirati relacije, odnosno koje su moguce vrste relacija? Analizirajuci postojece informacijske modele, nailazi se na razlicite klasifikacije relacija.

Relacija je opceniti pojam kojim se oznacava povezivanje entiteta, istog ili razlicitog tipa. Broj entiteta povezanih jednom relacijom odreduje dimenzionalnost, te se stoga mogu razlikovati binarne i višestruke relacije. Relacije, takoder, mogu biti usmjerene, od jednog objekta ka drugom, a neki informacijski modeli podržavaju i višesmjerne relacije (multidirectional).

U predloženom modelu osloniti cemo se na klasifikaciju relacija prema autorima UML-a [101] koji modeliraju (koriste) cetiri vrste relacija:

- generalizacija (specijalizacija)
- asocijacija
- relacija zavisnosti
- realizacija

Relacije generalizacije u objektom modelu implementiraju se postavljanjem hijerarhije klasa. U daljnjim razmatranjima vecinom cemo se baviti asocijacijama izmedu elemenata modela procesa konstruiranja.

Važno je uzeti u obzir da tijekom izvodenja procesa konstruiranja dolazi do generiranja novih veza izmedu objekata u modelu procesa, što znaci da elementi za prikaz relacija ne bi trebali biti staticne strukture, jer se njihov sadržaj stalno mijenja.

7.2.1 Matricni prikaz relacija

Od posebnog su interesa za predloženi model procesa konstruiranja relacije zavisnosti i asocijacija izmedu entiteta modela. Neki pristupi modeliranju procesa konstruiranja, npr. [130], [110] koriste matricne prikaze relacija izmedu objekata u procesu konstruiranja. U matricnom obliku relacija se prikazuje kao "oznaka" u celiji matrice koja indicira postojanje zavisnosti izmedu elementa stupca i retka kojima celija pripada. Takav prikaz zapravo je analogan grafu relacije prikazanom na slici 37. Najcešca upotreba matricnog prikaza relacija u istraživanjima metoda unapredenja procesa konstruiranja je matricni prikaz informacijske zavisnosti konstrukcijkih zadataka, tzv. "design structure matrix", koji je predložio Steward [111], [112].
Zapis relacija u matricnom obliku pogodan je zbog toga što se u relativno kompaktnom obliku može dobiti pregled medusobnih zavisnosti unutar vecih skupova objekata što može biti znacajno za planiranje redoslijeda izvodenja zadataka u procesu konstruiranja. Osim informacijske zavisnosti u matricnom obliku mogu se zapisati i drugi oblici veza izmedu objekata (npr. relacije pripadnosti, referenciranja, "odgovornosti za" i slicno).

Jedan od nedostataka matricnog prikaza je u cinjenici da zapis "oznake" u celiji matrice samo indicira postojanje relacije (veze), dok ništa ne govori o svojstvima (semantici) instance relacije. Taj problem može se djelomicno riješiti na više nacina:

- klasifikacijom oznaka, pri cemu svaka oznaka ima odredeno znacenje
- zapisom relacije u celiju matrice (koji može biti djelomicno skriven pri prikazu matrice (npr. kao zapis komentara u celiju u programskom paketu MS Excel)
- tako da se u celiju matrice zapisuje pokazivac na puni zapis i opis relacije

Matricni prikaz relacija izmedu skupova objekata može biti prilicno nepregledan ako broj stupaca i redaka (odnosno elemenata) bude puno veci od deset do dvadeset. Za takve situacije potrebno je razraditi procedure koje prikazuju samo odredeno relevantno podrucje matrice, kao što je npr. riješeno u [5].

Druga mogucnost je razbijanje osnovne matrice na submatrice (podmatrice) da bi se manipuliralo samo sa onim podrucjem matrice koje je za danu situaciju relevantno. Na ovaj nacin mogu se pojednostavniti operacije pregledavanja i ažuriranja sadržaja velikih matrica. Primjer submatrice prikazan je na slici 38. Submatrica je reducirani prikaz matrice dobiven brisanjem stupaca 1, 4, 6 i 7, te brisanjem redaka 1, 3, 4, i 7. Ovakav pristup podrazumijeva razradu svih potrebnih operacija za dinamicka definiranja i manipulaciju sa submatricama.
U ovakvom pristupu samo jedan redak ili stupac matrice takoder se mogu promatrati kao submatrice.

	1	2	3	4	5	6	7	8
1				x			x	
2			x			x		
3				x	x		x	
4		x					x	
5		x		x				x
6			x			x		
7				x				
8					x	x		

→

	2	3	5	8
2		x		
5	x			x
6		x		
8			x	

Slika 38: Osnovna matrica i jedna varijanta submatrice

Razmotrimo još graf relacije izmedu dva skupa (slika 37). Neka su elementi skupova A i B instance klasa (skupovi objekata) koje implementiraju entitete modela procesa konstruiranja. Matricni prikaz sa "oznacenom celijom" analogan je takvom grafu relacije. Realno je pretpostaviti da ce vecina grafova relacija izmedu dva skupa objekata imati malen broj "oznacenih" celija u odnosu na ukupni broja celija, odnosno da ce relacija biti "malen" podskup kartezijevog produkta skupova A i B. U takvim slucajevima matricni prikaz (graf relacije) može se u prikazu svesti na listu sa dva stupca (slika 39). U lijevom stupcu liste svi su elementi skupa A, a u desnom stupcu liste zapisuju se (u jednoj celiji) oznake (redni brojevi) svih elemenata skupa B koji su u relaciji sa elementom skupa A. Na taj nacin svi stupci matrice svode se na jedan. Ovakav nacin prikaza matrice ima smisla u slucajevima kad je broj "oznacenih" celija matrice u recima znatno manji od broja stupaca matrice. Treci stupac takvog prikaza može sadržavati zapis semantike relacije, ako je element skupa A povezan na semanticki isti nacin sa svim elementima skupa B. Primjer takvog zapisa je npr. algebarska jednadžba $a_i = f(b_j)$.

	b_1	b_2	b_3	b_4	b_5	b_6	….	b_n
a_1				x				
a_2			x			x		
a_3				x	x			
a_4		x						
a_5		x		x				x
….								
a_n					x	x		

→

	elementi skupa B
a_1	b_4
a_2	b_3,b_6
a_3	b_4,b_5
a_4	b_2
a_5	b_2,b_4,b_n
….	
a_n	b_5,b_6

semantika relacije
$a_1 = (b_4)^2$
$a_2 = b_3 + b_6$
…..

Slika 39: Preslikavanje matrice u listu

U osmom poglavlju biti ce pokazano kako se matricni prikaz relacije sveden na listu može realizirati pri implementaciji sustava. Potrebno je postaviti dvije asocijacije izmedu klasa A

i B koje predstavljaju skupove svojih instanci, odnosno elemente skupova A i B. Pri tome se asocijacije realiziraju skupovima pokazivaca ("pointera").

7.2.2 Mreža relacija izmedu entiteta modela procesa konstruiranja

Razmotrimo sada koje vrste entiteta i na koji nacin treba povezivati u modelu procesa konstruiranja. Za tu svrhu formirana je matrica u kojoj su elementi redaka i stupaca osnovni gradbeni (strukturalni) entiteti modela procesa konstruiranja (tablica 3).

Opcenito gledano, u logickom modelu procesa konstruiranja mogle bi se postavljati relacije izmedu svih definiranih entiteta, odnosno mogli bi promatrati mrežu veza u kojoj je svaki entitet povezan sa svim ostalim entitetima.

Ova razmatranja kljucna su za hipotezu rada, jer ce pri razradi implementacije sustava biti pokazano da je objektni model pogodan za modeliranje i realizaciju takve mreže relacija, odnosno mrežne topologije procesa konstruiranja.
Drugim rijecima ovako postavljeni logicki model mreže relacija moguce je preslikati u objektni model, realiziran u objektnoj bazi, što ce biti pokazano u osmom poglavlju.

Jasno je da ce neke od relacija (iz mreže svih mogucih relacija) biti za modeliranje procesa konstruiranja bitno znacajnije od drugih, i isto tako je jasno da neke relacije nece imati smisla postavljati u kontekstu modelu procesa konstruiranja.

Matrica prikazana tablicom 3 prikazuje mrežu relacija izmedu entiteta koji se u objektnom modelu preslikavaju u klase. Klase u konkretnom modelu sadrže skupove instanci, tj. skupove objekata. Tako svaku celiju ove matrice možemo promatrati kao jednu relaciju izmedu skupova instanci dviju klasa, a citavu matricu kao skup takvih relacija.

Oznaka "x" matrice stavljena je u one celije matrice, za koje se smatra da ima smisla modelirati relacije izmedu takvih objekata, a oznake "?" teme su za razmatranje. Predložena shema razmatra samo binarne veze (izmedu dva elementa). Sve oznake su ispod dijagonale matrice - razmatra se samo povezivanje elemenata (bez uzimanja u obzir smjera veze). Razliciti smjerovi asocijacija biti ce razmotreni kasnije (npr. "P-A" i "A-P").
U daljnjem tekstu razmotrene su predložene varijante relacija uz osvrt na mogucnosti primjene. Vrijedno bi bilo razmotriti i moguce složenije veze izmedu više elemenata (višestruke relacije), medutim za te slucajeve vjerojatno bi se prikaz takvih veza u matricnom obliku pokazao nepogodnim.

		P	BP	PP	OSP	SPA	A	Z	K	OPP
parametri	P	x								
baza parametara	BP	?	?							
prikaz proizvoda	PP	x	?	x						
objekt strukture proizvoda	OSP	x	?	x	x					
sucelje programskog alata	SPA	x	?	x	x	?				
akcija	A	?	?	?	?	x	?			
zadatak	Z	x	?	x	x	?	?	x		
konstruktori	K	?	?	x	x	?	?	x	?	
objekti prikaza procesa	OPP	x	?	x	x	x	x	x	?	x

Tablica 3: Povezivanje strukturalnih elemenata modela procesa konstruiranja

Relacije izmedu strukturalnih elemenata modela procesa konstruiranja najcešce ce biti informacijske zavisnosti ili relacije pripadnosti.

U daljnjem tekstu razraditi ce se prikazi onih relacija izmedu skupova objekata modela procesa konstruiranja koji po mišljenju autora trebaju biti elementi modela. Pri tome ce se kao metoda zapisa relacije koristiti matrica ili matrica svedena na listu. U ovim razmatranjima ograniciti cemo se samo na binarne relacije.

Izloženi matricni prikaz (poglavlje 7.2.1 i tablica 3) dan je stoga primarno kao teorijski model za razmatranje skupa (mreže) relacija izmedu entiteta modela. Entiteti se preslikavaju u klase, tj. relacije se postavljaju izmedu skupova instanci dviju klasa ili unutar skupa instanci iste klase.

7.2.3 Matrica informacijske zavisnosti konstrukcijskih zadataka

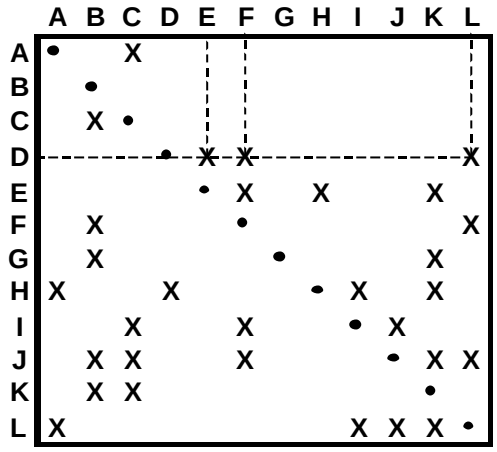
Najznacajnija relacija unutar predloženog skupa entiteta modela procesa konstruiranja svakako je prikaz informacijske zavisnosti izmedu konstrukcijskih zadataka. Ovo je relacija na skupu instanci konstrukcijskih zadataka koju cemo razmatrati u matricnom zapisu. Istraživanja mogucnosti unapredenja organizacije procesa konstruiranja, temeljena na analizi ovakvog matricnog prikaza pocinju osamdesetih godina [111], [112]. Kako je vec navedeno, uobicajen naziv ovakvog prikaza u literaturi je "Design Structure Matrix". Znacajna istraživanja organizacije konstrukcijskih zadataka u procesima istovremenog inženjerstva provode Eppinger [110], [113], i Rogers [114], [115]. Wallace [5] koristi ovu matricu kao jedan od elemenata za prikaz toka informacija u sustavu integracije programskih alata za podršku konstruiranju.

Slika 40 prikazuje matricu u njenoj originalnoj binarnoj formi prema [111], [112].

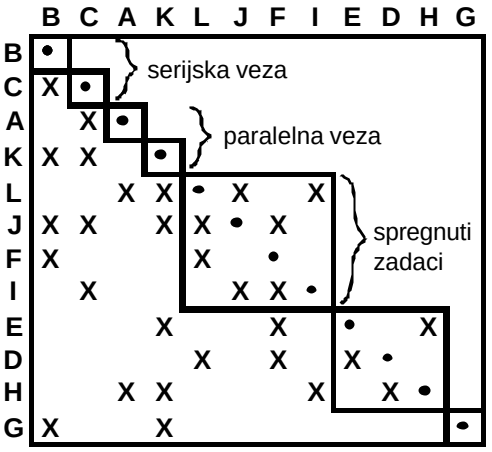
Konstrukcijski zadaci prikazani su identicnim oznakama u stupcu i retku matrice. Oznacene celije pokazuju koji zadaci moraju osigurati informacije da bi se odredeni zadatak mogao izvršiti.

ovisi o:		1	2	3	4	5	6	7	8	9	10
zadatak 1	1										
zadatak 2	2	x									
zadatak 3	3						x				
zadatak 4	4		x			x					
zadatak 5	5			x							x
zadatak 6	6			x				x			
zadatak 7	7								x		
zadatak 8	8		x			x					
zadatak 9	9										
zadatak 10	10				x				x		

Slika 40: Matrica prikaza informacijskih zavisnosti konstrukcijskih zadataka



Slika 41:"Neorganizirana matrica"



Slika 42:Ista matrica nakon reorganizacije

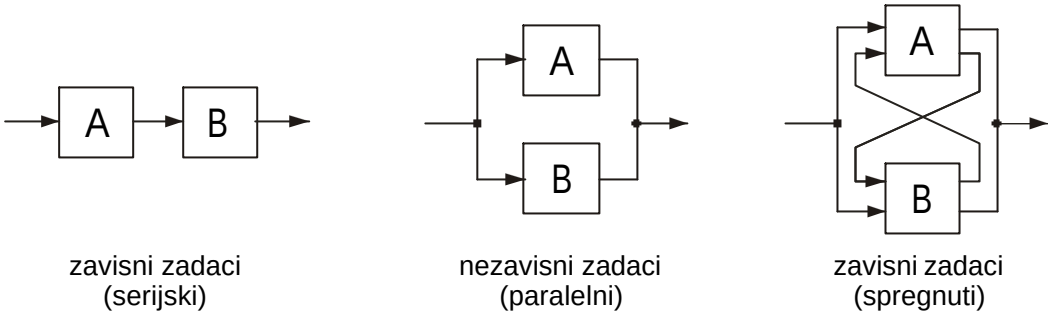
Razmotrimo primjer matrice sa slike 41.

Oznake u retku D postoje u stupcima E, F i L, što znaci da izvodenje zadatka D zahtijeva transfer informacija iz zadataka E, F i L. Iz toga slijedi da bi se tri spomenuta zadatka trebala izvršiti prije zadatka D. Dijagonalni elementi ove matrice nemaju nikakvog znacenja.

Prvi korak analize strukture ove matrice (procesa konstruiranja) je iznalaženje redoslijeda izvodenja konstrukcijskih zadataka u kojem bi sve oznake u matrici bile ispod glavne dijagonale. U tom slucaju svi zadaci bi se izvodili tek nakon što prime sve potrebne informacije od svojih prethodnika. Nažalost takve situacije se rijetko mogu postici u praksi, pogotovo ako se u proces proizvodnje (i konstruiranja) nastoji uvesti istovremeno inženjerstvo ("concurrent engineering"). Slika 42 prikazuje matricu sa slike 41 nakon što je zamjenom redaka i stupaca promijenjen redoslijed izvršavanja zadataka s ciljem da što manje relacija bude iznad glavne dijagonale. Postupci reorganizacije redoslijeda izvršavanja zadataka redovito kao rezultat daju matricu u kojoj se mogu uociti blokovi veza oko glavne dijagonale. Blokovi na dijagonali pokazuju informacijske sprege zadataka uzrokovane povratnim vezama. Preostale veze ispod glavne dijagonale prikazuju "isporuku" informacija zadacima koji slijede.

Reorganizirana matrica (slika 42) pokazuje da zadatak C ovisi o zadatku B, dakle redoslijed izvršavanja je B-C. Zadaci A i K ovise oba o zadatku C, ali se mogu izvoditi paralelno (jer A i K ne ovise medusobno). "Blokovi" koji sadrže zadatke L-J-F-I i E-D-H prikazuju dva skupa spregnutih zadataka. Problem izvodenja skupova spregnutih zadataka predstavlja veliki izazov pri uvodenju metoda istovremenog inženjerstva, makar se takva situacija može pojaviti i u drugacije organiziranim procesima konstruiranja.

Slika 43, prema [110] prikazuje tri navedene situacije iz matrice na primjeru izvodenja dva konstrukcijska zadatka.



Slika 43: Tri moguca redoslijeda izvršavanja dva konstrukcijska zadatka

Za spregnute zadatke teško je odrediti redoslijed njihova izvršavanja, cesto ce to biti nekoliko iterativnih ciklusa. Skup spregnutih zadataka trebao bi se izvoditi simultano, a transfer informacija koje uzrokuju spregnutost mogao bi poprimiti oblik pregovaranja izmedu "izvodaca" zadataka. U takvim situacijama od znacajne koristi trebao bi biti atribut "status vrijednosti" u objektu "parametar konstrukcije". Naime, pri simultanom izvodenju skupa spregnutih zadataka, odredene informacije cije vrijednosti još zadatku nisu poznate morati ce biti pretpostavljene. Pretpostavke ce kasnije morati biti korigirane, što znaci da ce se zadaci izvoditi u nekoliko iterativnih ciklusa. Parametri koji uzrokuju spregnutost skupa zadataka svakako trebaju biti spremljeni u bazu parametara jer se na taj nacin može realizirati koordinacija odnosno "pregovaranje" o trenutnoj vrijednosti i statusu zajednickih informacija, odnosno podataka (parametara).

U dosadašnjim istraživanjima primjene prikazane matrice razvijeno je nekoliko metoda reorganizacije redoslijeda izvršavanja zadataka:

- metoda temeljena na binarnoj matricnoj algebri [131]
- metode temeljene na ekspertnim sustavima i genetickim algoritmima [114], [115]
- postupci prema [132]

7.2.3.1 Proširenja i mogucnosti primjene matrice zavisnosti konstrukcijskih zadataka

U binarnom obliku zapisa matrica zavisnosti konstrukcijskih zadataka prikazuje samo da li neki zadatak informacijski ovisi o drugome ili ne. Binarni podatak ništa ne govori o "jacini", odnosno znacaju te povezanosti. Eppinger [51] stoga uvodi u matricu "težinski faktor" zavisnosti zadataka, a u glavnu dijagonalu zapisuje pretpostavljeno vrijeme trajanja zadatka. U takvom pristupu algoritmi reorganizacije redoslijeda nastoje bliže glavnoj dijagonali smjestiti oznake s vecom vrijednosti težinskog faktora. U dosta slucajeva spregnutost zadataka teško je izraziti numericki. Nadalje, svaki element matrice mogao bi biti vektor, odnosno niz vrijednosti koje na razlicit nacin izražavaju zavisnost zadataka.
Umjesto "težinskog faktora" može se uvesti i klasifikacija povezanosti zadataka. Eppinger predlaže razlikovanje "ulaza", "kontrole" i "povratne veze".
Svaki od takvih razlicitih pristupa zahtijeva i drugacije analiticke postupke reorganizacije redoslijeda izvodenja zadataka. Ovisno o potrebama i složenosti procesa konstruiranja u odredenoj situaciji (okolini) mogu se modelirati razlicite podklase matrice zavisnosti konstrukcijskih zadataka, sa razlicitim prikazima i postupcima reorganizacije redoslijeda zadataka.

Zapis zavisnosti konstrukcijskih zadataka u matricnom obliku svoju primjenu u procesu konstruiranja treba naci prvenstveno kao podloga planiranju i pracenju redoslijeda

izvodenja zadataka u procesu konstruiranja. U okruženjima gdje je proces konstruiranja vrlo složen, sa vecim timovima ljudi i vrlo složenim proizvodima, ova metoda može dati znacajni doprinos, pogotovo ako se nastoji uvesti paralelno (istovremeno) inženjerstvo. U takvim slucajevima primjene, mogu se ustanoviti blokovi informacijski spregnutih zadataka. Takav zapis može služiti kao podloga za unapredenje rješavanja iteracijskih procesa, kao i za eventualno redefiniranje strukture konstrukcijskih zadataka.

7.2.4 Relacije zavisnosti parametara konstrukcije

Relacije zavisnosti izmedu parametara konstrukcije najbrojnije su i najsloženije u procesu konstruiranja. Vecina tih zavisnosti algebarske su jednadžbe, ali mogu se javljati i drugi oblici zavisnosti. Proces rješavanja sustava jednadžbi s parametrima konstrukcije jedna je od kljucnih aktivnosti (podprocesa) u procesu konstruiranja. Strukture sustava jednadžbi u procesu konstruiranja razmatraju se u [120]. Autori daju i prijedloge reformulacija strukture kojima se može smanjiti razina kompleksnosti sustava i unaprijediti proces odlucivanja. Složene konstrukcije na kojima radi tim konstruktora sadrže veliki broj (pretežno geometrijskih) parametara cije meduzavisnosti je potrebno sistematizirati i organizirati u zajednickom obliku zapisa. Prijedlog takvog zapisa u izloženom modelu procesa konstruiranja je matrica relacija zavisnosti izmedu parametara konstrukcije.

Matricni zapis meduzavisnosti parametara konstrukcije analogan je zapisu zavisnosti zadataka konstrukcije. Oznaka "x" u celiji matrice znaci da odredeni parametar ovisi o drugom parametru (tablica 4). Zapis u jednom retku pokazuje o kojim sve parametrima ovisi odredeni parametar: $p_i = f(p_j)$.

		p1	p2	p3	p4	p5	….	pn
parametar 1	p1		x		x			
parametar 2	p2	x				x		
parametar 3	p3							
parametar 4	p4		x					
parametar 5	p5				x			
. . . . .	. . . .							
parametar n	pn			x				

Tablica 4: Matrica zavisnosti parametara konstrukcije

Realno je ocekivati da ce svaka malo složenija konstrukcija sadržavati veliki broj parametara. Postavlja se pitanje što uciniti s matricnim prikazom u takvim slucajevima, jer postaje velik i nepregledan, s malim brojem oznacenih celija.
Pri manipulaciji sa prikazom zavisnosti parametara mogu se koristiti podmatrice, (prema shemi iz poglavlja 7.2.1, odnosno prikazivati samo ona podrucja matrice koja su trenutno interesantna.
Drugo je pitanje kako zapisati matricu reda velicine npr. približno 1000 parametara, dakle ukupno 10^6 celija. Svakako da nema smisla zapisivati sve vrijednosti celija, jer ce vrlo vjerojatno samo mali postotak imati "oznaku" postojanja zavisnosti. Dovoljno je uz svaki parametar zapisati samo adrese, odnosno redne brojeve stupaca drugih parametara o kojima on ovisi. Iz takvog se zapisa jednostavnim algoritmom može generirati i matricni prikaz (odnosno podmatrica odredenog podrucja) u situacijama kada je to potrebno. Postupak preslikavanja matrice u listu vec je razmatran i prikazan na slici 39. Tablica 5 prikaz je

matrice zavisnosti parametara konstrukcije svedene na listu sa tri stupca. Drugi stupac matrice sadrži adrese parametara o kojima ovisi odredeni parametar, a treci stupac sadrži zapis semantike zavisnosti: $p_j = f(p_i, i \in \{1, \ldots n\}, i \neq j)$, gdje je n broj svih definiranih parametara u bazi parametara.

redni br. parametra:	ovisi o:	zapis jednadžbi zavisnosti
p_1	p_2, p_4, p_{78}	$p_1 = f(p_2, p_4);\ p_1 = f(p_2, p_{78})$
p_2	p_1, p_5, p_{36}	$p_2 = f(p_1, p_5, p_{36})$
p_3	p_{24}, p_{78}	. . . . .
p_4	p_2, p_{20}, p_{35}	
p_5	p_4, p_{153}	
. . . . .	. . . . .	
p_j	$p_i, i \in \{1, \ldots n\}, i \neq j$	$p_j = f(p_i, i \in \{1, \ldots n\}, i \neq j)$
. . . . .	. . . . .	. . . . .
p_n		

Tablica 5: Shema zapisa zavisnosti parametara konstrukcije

Funkcija f zavisnosti parametara najcešce ce biti jedna ili više algebarskih jednadžbi, no može biti i druga vrsta funkcije, ovisno o vrsti parametra (npr. funkcija može biti u obliku "if-then" pravila).

Zapis zavisnosti parametara konstrukcije može nam dati još jednu vrlo korisnu informaciju. Cesto je posebno važno znati kako promjena jednog parametra utjece na ostale parametre u konstrukciji, odnosno koji sve parametri ovise o odredenom parametru? Takva informacija može se dobiti kao rezultat upita ("query-a") u kojem se prikazuju svi reci liste u kojima se traženi parametar pojavljuje u drugom stupcu liste.

Matrica zavisnosti parametara konstrukcije može poslužiti kao podloga za izradu matrice zavisnosti konstrukcijskih zadataka. Ukoliko se u matricnom obliku zapišu relacije pripadnosti svakog od parametra konstrukcijskim zadacima, interesantno bi bilo istražiti da li je moguce realizirati algoritam koji bi na temelju zapisa u obje matrice automatski generirao matricu zavisnosti konstrukcijskih zadataka.

7.2.5 Relacije "pripadnosti" izmedu razlicitih klasa objekata

U prethodna dva poglavlja razmatrane su relacije zavisnosti i mogucnosti zapisa njihove semantike. Relacije "pripadnosti" jednostavnije su strukture jer nije potrebno posebno zapisivati semantiku relacije, nego samo naznaciti "što cemu pripada". Relacije pripadnosti u pravilu postoje izmedu skupova objekata dviju razlicitih klasa. Zapis ovih relacija takoder se sa matricnog oblika može svesti na listu sa dva stupca. Tablica 6 prikazuje primjer zapisa relacije pripadnosti objekata klase B objektima klase A. Desni stupac tablice sadrži reference na neprazni podskup objekata klase B koji "pripadaju" odredenom objektu klase A. Pri tome je n broj objekata klase A, a k je broj objekata klase B.

Objekti klase A	objekti klase B koji pripadaju objektu klase A
a_1	b_2, b_3, b_{12}
a_2	b_3, b_4
a_3	b_2, b_{10}
……	……..
a_i	$b_i, i \in \{1, \ldots k\}$
……	……..
a_n	……

Tablica 6: Prikaz relacije "pripadnosti" izmdu objekata dviju klasa

Dalje slijede razmatranja nekoliko primjera relacija "pripadnosti". Sve takve relacije mogu se zapisati prema shemi prikazanoj u tablici 6. Koristeci mogucnosti objektne baze, ovakve skupove relacija relativno je jednostavno realizirati, kao i manipulirati s njima, što ce detaljnije biti prikazano u osmom poglavlju.

7.2.5.1 Relacija objekt prikaza proizvoda - parametar

Ova relacija odreduje (pokazuje) koji parametar "pripada" kojem objektu prikaza proizvoda, prema shemi iz tablice 6. Svaki objekt prikaza proizvoda sadrži dakle svoj skup referenci na parametre cije vrijednosti su dio zapisa informacija o proizvodu kojeg taj objekt modelira. Gotovo redovito se vrijednosti nekih parametara zapisuju u više prikaza proizvoda, odnosno ti parametri ce biti referencirani u više objekata prikaza proizvoda. Upravo ta cinjenica je i razlog modeliranja parametara kao objekata, što je detaljnije vec razmotreno u poglavlju 7.1.2.
Pri generiranje svake nove instance objekta prikaza proizvoda potrebno je upisati reference na parametre, odnosno zapisati novi redak u listu koja prikazuje relaciju.

Iz predložene sheme zapisa moguce je i dobiti pregled koji objekti "dijele", tj. koriste odredeni parametar. Takvi pregledi mogu biti znacajni za situacije u procesu konstruiranja u kojima se objekti koji dijele isti parametar obraduju istovremeno.

Osim referenci na parametre koji mu pripadaju, objekt prikaza proizvoda sadrži i operacije sucelja za transfer vrijednosti parametara prema zapisu informacija o proizvodu kojeg modelira.

7.2.5.2 Ostale relacije "pripadnosti"

Ovdje ce se navesti primjeri još nekoliko relacija "pripadnosti" izmedu razlicitih klasa objekata koje se prema mišljenju autora mogu koristiti u modelu procesa konstruiranja:

- Relacija konstruktor - zadatak evidencija je dodjele zadataka konstruktorima.
- Relacija konstruktor - objekt prikaza proizvoda evidentira za koje je prikaze proizvoda pojedini konstruktor odgovoran.
- Relacija akcija - sucelje programskog alata povezuje instance klase akcije za poziv programskog alata i sucelje poziva programskog alata koje se koristi.

7.2.6 Izrazi kao elementi zapisa ogranicenja i pravila

U poglavlju 3.2.2 razmatrano je napredovanje kroz proces konstruiranja provjerom ogranicenja i odlucivanjem. Entiteti koji modeliraju etape procesa konstruiranja trebaju dakle ukljuciti procese provjere i dodavanja ogranicenja, kao i procese odlucivanja temeljem analize skupova pravila. Provjera ogranicenja i pravila odlucivanja trebaju dakle biti dio strukture zapisa plana procesa konstruiranja. Konstruktor kreira plan konstruiranja u kojem treba predvidjeti i zapisati provjere ogranicenja i skupove pravila na temelju kojih se donose odluke. Ogranicenja i pravila detaljno se razraduju u slijedeca dva poglavlja.

Kao osnovni element zapisa ogranicenja i pravila može se izdvojiti "izraz". Pojam izraz (expression), analogan je u razmatranom kontekstu "izrazu" kao dijelu naredbe, kako se definira u gotovo svim programskim jezicima - kao niz sintaktickih elemenata koji se sastoji od operanada i operatora.
Ako se izraz interpretira kao ogranicenje ili kao uvjet u pravilu, tada se odreduje njegova vrijednost kao istina ili laž. Izraz je dakle leksicka struktura kojom se formiraju zapisi provjere ogranicenja i pravila odlucivanja. Izraz se može definirati i kao "predikatna formula" nad datim jezikom.
Izraz se sastoji od operanada i operatora, gdje su operandi atributi objekata modela procesa konstruiranja ili numericke ili znakovne konstante. Pri tome izraz mora sadržavati barem dva operanda (od kojih bar jedan treba biti atribut objekta) i barem jedan operator.

```
izraz ::= <operand> | <operator> | <operand> { <operator> | <operand> }
operand ::= <varijabla> | <konstanta>
operator ::= <aritmeticki operator> | <relacijski operator> | <logicki
             operator> | <zagrada>
varijabla ::=  #naziv_objekta.naziv_atributa
konstanta ::= <numericka konstanta> | <znakovna konstanta>
logicki operator ::= <&&> | <||> | <and> | <or>
aritmeticki operator ::=   <+> | <-> | <*> | </> |
relacijski operator ::= <=> | <≠> |<<=> | <>=> | <<> | <>>
zagrada ::=  <(> | <)>
```

Slijedi nekoliko primjera izraza. U primjerima nazivi objekata su skraceni na oblik "p *i*", a atributi na "v" (vrijednost) i "s" (status).

```
#p1.v = #p3.v - #p5.v / 2.81
#p20.v = 'C4732' or  #p20.v = 'C1531'
#p4.s = 'prijedlog' or #p4.s = 'neodredjen'
#p1.v + #p3.v > #p8.v / 10.5
#p1.v > 10 and #p2.s = 'odredjen'
#p5.v > #p3.v  or  (#p2.v +10) < #p4.v
```

Osnovni operandi izraza su varijable koje su atributi objekata. Varijabla se sastoji od naziva objekta i naziva atributa tog objekta. Na ovaj nacin u izrazima se mogu koristiti svi atributi svih objekata modela procesa konstruiranja. Ovdje pretpostavljamo da se domene varijabli i domene operatora u izrazu uvijek poklapaju, tj. da su izrazi semanticki ispravni.
Izraz se zapisuje kao niz znakova u tijeku kreiranja plana konstruiranja. Vrijednost izraza odreduje se "parsiranjem" u tijeku izvodenja plana konstruiranja. Postupkom parsiranja varijablama se dodjeljuju trenutne vrijednosti atributa objekata, primjenuju se operatori i odreduje se da li je izraz istinit ili ne. Osim samog zapisa izraza kao niza znakova, instanca

klase "izraz" sadrži i skup pokazivaca (pointera) na objekte ciji atributi se pojavljuju kao elementi (varijable). Implementacija tako postavljene strukture biti ce detaljnije razradena u osmom poglavlju.

Prethodna razmatranja promatraju "izraz" samo kao leksicku strukturu, bez obzira na njegovo znacenje u kontekstu izvodenja procesa konstruiranja. Izrazi se mogu koristiti kao elementi provjere ogranicenja ili kao uvjeti u skupovima pravila odlucivanja. Izrazi trebaju dakle biti elementi klase "ogranicenje" i klase "pravilo". Za oba slucaja operacija parsiranja, odnosno sintaksa izraza je ista, samo je razlicit kontekst uporabe izraza. Klase "ogranicenje" i "pravilo", dijele dakle istu formu zapisa izraza i operaciju parsiranja izraza. Izraz se onda može modelirati kao klasa sadržana u spomenutim klasama ili ogranicenja i pravila trebaju sadržavati pokazivace na pripadajuce instance izraza.

7.2.7 Ogranicenja

Proizvod koji se konstruira mora zadovoljiti skup funkcionalnih zahtjeva. Funkcionalni zahtjevi ispunjavaju se indirektno mijenjanjem jednog ili više parametara konstrukcije, fizickog oblika i operativnih uvjeta [120]. Uobicajeno je da se funkcionalni zahtjevi izražavaju kao funkcije parametara konstrukcije, drugih funkcionalnih zahtjeva i razlicitih "internih varijabli" ("prijelaznih ili pomocnih" varijabli) [120]. Interne varijable i parametri konstrukcije cine vektor nepoznanica konstrukcije $\mathbf{P} = [p_1, p_2, p_3, \ldots, p_n]$. Funkcionalni zahtjevi cine vektor $\mathbf{F} = [F_1, F_2, F_3, \ldots, F_n]$. Relacije izmedu zahtjeva i nepoznanica mogu se izraziti kao:

$$f_i (\mathrm{F}, \mathrm{P}) = 0 \quad i = 1, k$$
$$g_j (\mathrm{F}, \mathrm{P}) \leq G_j \quad j = 1, s$$

Navedene jednadžbe i nejednadžbe u terminologiji znanosti o konstruiranju nazivaju se "ogranicenja". Pri tome neke od navedenih relacija ne moraju sadržavati funkcionalne zahtjeve, nego samo parametre i "interne varijable". Treba napomenuti da relacije ogranicenja nisu relacije zavisnosti izmedu parametara, (razmatrane u poglavlju 7.2.4) makar se u nekim slucajevima mogu i poklapati.

Proces konstruiranja napreduje kroz postavljanje i zadovoljavanje ogranicenja. Stoga model procesa konstruiranja mora sadržavati postupke kreiranja i provjere ogranicenja. U predloženom modelu procesa konstruiranja nepoznanice konstrukcije i funkcionalni zahtjevi modelirani su kao atributi razlicitih klasa objekata. Dakle možemo reci da ogranicenja definiraju prostor prihvatljivih vrijednosti atributa objekata [7].

Promatrajuci tijek (napredovanje) procesa konstruiranja, ogranicenja se provjeravaju kao preduvjeti i postuvjeti koji moraju biti ispunjeni u odredenim situacijama, odnosno etapama procesa. Rezultat provjere ogranicenja samo je informacija da li je ono ispunjeno ili ne. Proces provjere ogranicenja nema nikakvog utjecaja na atribute objekata koji su sadržani u relaciji ogranicenja.

Entitet "ogranicenje" modelira zapis i proces provjere ogranicenja, te sadrži slijedece atribute:

- pokazivac (pointer) na zapis izraza koji sadrži jednadžbu ili nejednadžbu ogranicenja
- opis - dodatni komentari i pojašnjenja
- vrsta upozorenja ako ogranicenje nije ispunjeno - nema upozorenja, ispis poruke, ispis poruke i prekid procesa

- status ogranicenja kod zadnje provjere: ispunjeno, nije ispunjeno
- status provjeravanja: aktivno - provjerava se, pasivno - još se ne provjerava
- zapis stanja svih provjera u tijeku procesa

Ogranicenje sadrži pokazivac na izraz cija se istinitost provjerava. Izraz je leksicka struktura prema sintaksi predocenoj u poglavlju 7.2.6. Operacija parsiranja izraza odreduje da li je izraz istinit (ogranicenje zadovoljeno) ili nije.

Vecina provjera ogranicenja biti ce zapravo granicni uvjeti iterativnih procesa. Zapis stanja atributa u trenutku provjere ogranicenja može biti korisna informacija za donošenje odluka u slijedecim koracima iteracije. Isto tako konstruktor može zapisati i obrazložiti (u obliku teksta) što je poduzeo u svakom koraku iteracije u kojem ogranicenje nije bilo ispunjeno. Na taj nacin zapisuju se informacije o procesu razvoja konstrukcije koje mogu biti izuzetno korisne za razvoj slijedecih varijanti konstrukcije. Takve vrijedne informacije najcešce ostaju nezabilježene i neiskorištene. Jedan od pravaca istraživanja unapredenja racunalne podrške konstruiranju upravo je orijentiran na spomenute probleme ("design intent and rationale capturing").

Operacije za generiranje i zapis ogranicenja potrebno je koncipirati tako da se provjeravaju tipovi atributa, odnosno treba osigurati da operandi i operatori budu iz iste domene.

7.2.8 Pravila odlucivanja

Do sada razmatrani entiteti objektnog modela procesa konstruiranja modeliraju osnovne strukture podataka i skupove relacija izmedu takovih objekata. Time je modelirana staticka struktura domene procesa konstruiranja, ali ne i struktura dinamike izvodenja procesa. Modeliranje prikaza i kontrole izvodenja procesa tema je slijedeceg poglavlja (7.3). Kao osnovni elementi usmjeravanja tijeka izvodenja procesa konstruiranja u predloženom modelu koncipirana su "pravila odlucivanja". Pravila odlucivanja mogu se razmatrati i kao relacije izmedu akcija koje se izvode u procesu i stanja i vrijednosti atributa objekata. Pravila odlucivanja dakle su i relacije i elementi kontrole izvodenja procesa. Prikaz pravila odlucivanja ukljucen je u poglavlje o relacijama i stoga što je "izraz" jedan od osnovnih elemenata pravila, pa se izlaganja nadovezuju na prethodna dva podnaslova o izrazima i ogranicenjima.
Pravila odlucivanja elementi su prikaza tijeka procesa konstruiranja koja se generiraju i zapisuju prije izvodenja procesa, odnosno kontrolne strukture koje su objekti projektiranja procesa [7]. Dodatne kontrolne strukture na višoj razini koje upravljaju procesom konstruiranja predmet su razmatranja u poglavlju 7.3.

Pravila odlucivanja koncipirana su u obliku "IF *uvjet* THEN *akcija*" (ili premisa - akcija), pri cemu zadovoljenje uvjeta pokrece akciju. Struktura pravila može sadržavati jedan uvjet i više akcija. Kao uvjet pravila koristi se leksicki izraz, odnosno entitet "izraz", kako je definiran u poglavlju 7.2.6. Sintaksa pravila:

```
pravilo ::= if '(' <izraz> ')' then
                <niz_akcija>
            [ else
                <niz_akcija> ]
            end if
niz_akcija ::= <akcija> { ';' <akcija> }
```

```
akcija ::=  naziv_objekta.naziv_operacije // ovo je samo opcenita
                naznaka, jer pojedine klase akcija imaju razlicite sinatkse zapisa akcije
izraz ::= <operand> | <operator> | <operand> { <operator> |
            <operand> }    // prema sinatksi danoj u 7.2.6
```

Pri evaluaciji pravila odreduje se vrijednost izraza (uvjet) kao istina ili laž i na temelju toga odlucuje se koji niz akcija ce se izvršiti . Akcije koje pravilo pokrece mogu biti bilo koje od klasa akcija koje su razmatrane u 7.1.5. Na taj nacin, ovisno o vrijednostima, statusima i medusobnim odnosima atributa objekata, "odlucuje" se koja ce se operacija kojeg objekta pokrenuti. Pri tome akcija ne mora biti samo operacija objekta, nego može biti i promjena vrijednosti atributa nekog od objekata.

Predloženim konceptom moguce je modelirati vrlo složena pravila odlucivanja, sa vrlo bogatom semantikom na raspolaganju. Pod semantickim elementima pravila odlucivanja podrazumijevaju se:

- atributi svih klasa koje cine model procesa konstruiranja,
- mogucnost pozivanja neke od operacija bilo koje klase iz modela procesa konstruiranja,
- klasifikacija semantike akcija (slika 34) - promjena vrijednosti atributa, operacije s objektnom bazom, itd.

Na taj nacin pravila odlucivanja mogu se promatrati i kao implementacija znanja u model procesa konstruiranja. Npr. ako je akcija promjena vrijednosti parametra - tada dakle pravilo služi kao znanje za odredivanje vrijednosti parametra.

Svaka instanca pravila odlucivanja sadrži zapis pravila u obliku niza znakova, koji se u tijeku izvodenja procesa konstruiranja parsira. Entitet "pravilo odlucivanja" modelira dakle zapis i proces obrade (evaluacije) pravila odlucivanja, te sadrži slijedece atribute i operacije:

- pokazivac (pointer) na zapis cijelog pravila
- opis - dodatni komentari i pojašnjenja
- status pravila kod zadnjeg izvodenja - koji su uvjeti bili ispunjeni, odnosno koje su akcije izvedene
- zapis stanja svih izvodenja u tijeku procesa
- operaciju parsiranja pravila, koja predaje kontrolu odabranoj akciji

Vecina pravila odlucivanja možda ce se u tijeku procesa konstruiranja izvesti i nekoliko puta, kao dijelovi iterativnih procesa. Stoga je korisno zapisati koji su uvjeti bili ispunjeni u svakom od izvodenja, iz istih razloga kao i kod provjera ogranicenja (razmatranja na kraju poglavlja 7.2.7). U daljnjem razvoju može se tako predvidjeti i zapis primjedbi i obrazloženja konstruktora uz svako izvodenje pravila.

7.2.9 Generiranje i ažuriranje zapisa ogranicenja i pravila odlucivanja

Kao i operacije generiranja zapisa ogranicenja i operacije generiranja zapisa pravila odlucivanja trebaju sadržavati mehanizme kontrole ispravnosti sintakse i semantike pravila.

Zapisi ogranicenja i pravila odlucivanja u pravilu bi se trebali generirati prije izvodenja procesa konstruiranja, no nema nikakve zapreke da se oni mijenjaju i dodaju i u tijeku izvodenja procesa. Na taj nacin mogu se donekle ublažiti posljedice nepredvidenih situacija

u planu procesa, odnosno možemo govoriti o jednoj vrsti dinamickog prilagodavanja plana procesa u tijeku izvodenja! Time bi se mogucnosti kontrolnih struktura izvodenja procesa konstruiranja za jedan mali korak približile ljudskim mogucnostima dinamickog planiranja.

Predložena koncepcija može se realizirati na nacin da kontrolne operacije izvodenja procesa omogucavaju interakcije s kontruktorom prije i nakon procesa obrade ogranicenja i pravila odlucivanja. U takvim interakcijama omogucavala bi se promjena zapisa postojecih ogranicenja i pravila odlucivanja, kao i dodavanje novih. Štoviše, takve operacije interakcija s konstruktorom mogu biti i akcije u samim pravilima odlucivanja!

Pokretanje interakcije s konstruktorom može se i unaprijed planirati korištenjem npr. atributa ogranicenja i pravila, cija vrijednost bi odredivala da li prije ili poslije obrade treba pokrenuti operaciju dorade ili dodavanja ogranicenja ili pravila.

Relacije ogranicenja i pravila odlucivanja mogu se u "pseudo" formi zapisivati u bazu znanja procesa konstruiranja. Pri tome "pseudo" forma znaci da izrazi ne sadrže reference na konkretne objekte, nego su napisani u simbolickom obliku. Jedan prototip takvog sustava u relacijskoj bazi podataka realiziran je u [6].

7.3 Elementi prikaza i kontrole izvodenja procesa konstruiranja

Središnja tema ovog istraživanja je koncipiranje prikaza, odnosno modela procesa konstruiranja temeljeno na principima razvoja objektno orijentiranog modeliranja programskih sustava. U dosadašnjim izlaganjima predloženi su osnovni strukturalni elementi (entiteti) modela cija je funkcija modeliranje inženjerskih struktura podataka i kompleksne mreže relacija izmedu njih. Do sada razmotreni entiteti modeliraju pretežno samo staticke aspekte procesa konstruiranja. Modeliranje entiteta koji opisuju dinamicke aspekte znatno je zahtjevnija zadaca. Izlaganja u drugom i trecem poglavlju rada daju pregled procesa konstruiranja kao izuzetno kompleksne ljudske djelatnosti. Stoga se i prikazani teoretski modeli procesa konstruiranja dosta razlikuju u pristupima i aspektima koje razmatraju. Niti jedna od razvijenih opcenitih metoda prikaza tijeka procesa (razmatranih u poglavlju 4.5) ne može u potpunosti modelirati proces konstruiranja. Kratki pregledi i kritike uporabe nekoliko metoda prikaza procesa konstruiranja mogu se naci u radovima [133] i [134]. Fokus istraživanja ovog rada je koncipiranje dovoljno fleksibilnog okvira (okruženja) unutar kojeg se model procesa konstruiranja može adaptirati i nadogradivati prema specificnim potrebama pojedinih korisnika.

Kako je vec navedeno, u Znanosti o konstruiranju još ne postoji jedinstveni (usaglašeni) model procesa konstruiranja, pa je stoga i terminologija koja se koristi u istraživanjima iz te teme prilicno razlicita. Cesta je situacija da su nazivi semanticki istih pojmova razliciti u radovima razlicitih autora, no takva situacija ne treba iznenaditi, jer je ocito da i metodologija istraživanja vjerojatno još nije dovoljno sazrela i nije usaglašena.

U ovom radu za prikaz dinamickog aspekta procesa konstruiranja koristiti ce se naziv "plan konstruiranja". Pri tome taj termin obuhvaca i strukture podataka i operacije koje zajedno cine model odvijanja procesa u realnom vremenu, kao i model toka informacija izmedu statickih objekata sustava. Prema [88] plan je svaki hijerarhijski proces u organizmu koji može kontrolirati redoslijed po kojem treba izvoditi operacije.

Plan konstruiranja u predloženom modelu procesa konstruiranja definira se kao model dekompozicije i tijeka izvodenja procesa konstruiranja u kojem se organiziraju i predvidaju tokovi informacija i redoslijed izvršavanja akcija, te postavljaju kontrolni uvjeti.

Plan konstruiranja upravlja akcijama koje se izvode na objektima koji modeliraju strukture podataka i mrežu relacija izmedu njih. Elemente plana konstruiranja možemo podijeliti na prikaz tijeka odvijanja procesa i na upravljanje izvodenjem procesa.

Staticki prikaz procesa ne zadovoljava zahtjeve rješavanja situacija koje se javljaju u izvodenju procesa konstruiranja (razmatranja u poglavljima 4.3.2 i 4.3.3). Stoga elementi plana koji upravljaju izvodenjem procesa trebaju u tijeku procesa omoguciti promjene elemenata prikaza procesa i kontrolnih uvjeta, o cemu je vec bilo govora u poglavlju 7.2.9.

U poglavlju 4.3 razmotreni su pristupi planiranju rješavanja problema. Proces konstruiranja može se promatrati kao proces rješavanja problema, stoga se može pretpostaviti da se tehnike planiranja rješavanja problema mogu primjeniti (preslikati) na proces konstruiranja. Pri tome su od posebnog interesa metode oportunistickog planiranja i metoda temeljena na nacrtima uzoraka planova ("script based planning"). Kako implementirati te dvije metode u objektni model procesa konstruiranja? Jedna od mogucnosti je razdvojiti izradu plana konstruiranja na više razina (i/ili faza):

- izrada "nacrta" (skice) plana
- rješavanje konfliktnih situacija (spregnuti zadaci i ciljevi, algoritmi iteracije)
- generiranje konacnog zapisa plana po kojem ce se izvoditi proces konstruiranja

Nacrt plana ne mora sadržavati strogo formalni zapis plana, niti mora biti definitivno odreden redoslijed izvodenja operacija. Na temelju takve pocetne skice plana treba uociti, odnosno pokušati predvidjeti sve moguce konfliktne situacije i nacine njihova rješavanja.

Konacni zapis plana skup je entiteta i procedura koje cine racunalni model dinamickog aspekta procesa konstruiranja, odnosno služi kao alat za racunalnu podršku organizaciji i kontroli izvodenja procesa konstruiranja.

7.3.1 Nacrt plana konstruiranja

Kako je vec napomenuto, nacrt (skica) plana zamišljen je kao zapis informacija koji ne bi trebao slijediti unaprijed zadani skup formalnih pravila zapisivanja. Promatran kao entitet objektnog modela procesa konstruiranja, nacrt plana treba sadržavati sva potrebna sucelja prema osnovnim strukturalnim elementima modela i elementima za prikaz relacija. Dakle, konstruktor treba imati mogucnost pristupa do svih relevantnih informacija za izradu nacrta plana konstruiranja odredene varijantne, ponovljene ili nove konstrukcije. Na temelju nacrta plana rašcišcavaju se konfliktne situacije i prilazi se izradi "formalnog, konacnog" plana konstruiranja.
Ideja ovakvog koncepta je ne prisiljavati konstruktora da vec u pocetnoj fazi kreiranja plana konstruiranja mora prilagodavati svoja promišljanja, organizaciju rada i toka informacija, propisanim pravilima i sintaksi elemenata "formalnog" plana. Nacrt plana tako postaje mjesto za diskusiju i rašcišcavanje, a sam zapis nacrta plana može sadržavati prikaze i podatke iz strukturalnih i relacijskih entiteta modela. Takav prikaz takoder nužno ne mora sadržavati konacni redoslijed izvodenja akcija i operacija. Postavlja se pitanje kako programski realizirati takvu koncepciju? Potrebno je realizirati skup programskih alata za pretraživanje, dohvacanje i kopiranje informacija iz ostalih entiteta modela, a pri tome

najvecu pažnju treba posvetiti organizaciji i pristupu bazi znanja o konstruiranju proizvoda. Spomenuta baza znanja treba sadržavati prvenstveno znanje o procesu konstruiranja, dakle iskustva i procese promišljanja i odlucivanja iz prethodnih konstrukcija. Struktura baze znanja kao dijela modela procesa konstruiranja predložena je u [135].
U strukturu nacrta plana treba implementirati i sucelja prema alatima koji sadrže implementacije metoda za odredivanje redoslijeda izvršavanja akcija i zadataka, kao npr. operacije "preslagivanja" matrice informacijske zavisnosti konstrukcijskih zadataka.
Jezgru nacrta plana može ciniti lista aktivnosti gdje nije nužno odreden redoslijed aktivnosti, nego se samo navode potrebne aktivnosti, odnosno koraci ka postizanju cilja. Takva lista može se zapisivati u tekstualnom obliku, ili u obliku matrice, gdje se aktivnost povezuje sa temom ili objektom na kojeg djeluje. Elementi zapisa mogu biti i reference na strukturalne i relacijske objekte pohranjene u objektnim bazama podataka. Takvi objekti koriste se kao gotovi uzorci pri izradi nacrta plana kao i pri izradi konacnog formalnog plana.
Korak dalje u razvoju predloženog koncepta mogla bi biti implementacija metoda generiranja formalnog zapisa plana na temelju nacrta plana, ali do tog stupnja razvoja predstoji još mnogo istraživackog rada.
U ovoj fazi istraživanja važnije je razmotriti kako (i da li je moguce) koncipirati proces izrade nacrta plana na kojem radi više konstruktora istovremeno. Za takav slucaj mogla bi se koristiti "blackboard" kontrolna struktura, ali svakako ostaje otvoreno pitanje kako koordinirati rad više "planera" i na kraju spojiti plan u jednu cjelinu?

Kao zakljucak razmatranja o nacrtu plana navedimo njegove osnovne funkcije:

- da bude pocetna faza u procesu generiranja plana procesa konstruiranja, u kojoj se na "labaviji" nacin može prikazati proces, ali sa referencama na "prave" (postojece) objekte
- da posluži kao mjesto za diskusiju i rašciščavanje, te postupno odredivanje redoslijeda izvodenja akcija
- ranije otkrivanje mogucih konfliktnih situacija, zbog neopterecenosti formalizmom zapisa plana

7.3.1.1 Matrica "aktivnost - tema"

Kao jedan od elemenata nacrta plana (ili njegova jezgra), može se usvojiti matrica "aktivnost-tema", prema [130], [49] . Prema prijedlogu autorice, "oznaka" u celiji matrice bilježi da je obavljena odredena aktivnost na temi (tablica 7). U sustavu koji Blessing predlaže, matrica se koristi za bilježenje procesa konstruiranja sa svrhom ponovne uporabe takvog zapisa u slijedecoj varijanti konstrukcije. Predloženi nacin zapisa može modelirati nacrt plana konstruiranja, ako se umjesto u tijeku procesa, matrica popunjava prije izvodenja procesa. Pri tome oznake u celijama matrice mogu biti i redni brojevi koji oznacavaju preliminarni redoslijed aktivnosti.
Prema [130] popis tema i aktivnosti može se prilagoditi potrebama specificnog okruženja, odnosno procesa konstruiranja. Teme se rješavaju u tri koraka: generiranje, procjena (odredivanje) i izbor. Generiranje rezultira prijedlozima koji se odnose na zadanu temu. Taj stupac može sadržavati i druge informacije relevantne za prijedlog. Procjena i izbor mogu oboje rezultirati sa argumentima i odlukama. Argumenti podržavaju ili pobijaju prijedlog, te služe kao temelj za donošenje odluka. Argumenti mogu takoder dovesti i do novih prijedloga. Odluke opisuju status prijedloga.

	AKTIVNOSTI		
TEME	generiraj	procijeni	izaberi
problem			
zahtjev			
funkcije			
koncept			
geometrija			
materijal			
održavanje			
troškovi			

Tablica 7: Matrica "aktivnost-tema" prema [130]

Blessing povezuje ovu matricu sa objektima konceptualne strukture proizvoda. Kao element objektnog modela procesa konstruiranja ova matrica može se vezati uz objekte strukture proizvoda, objekte prikaza proizvoda i zadatke. Pri tome se ove matrice mogu i klasificirati prema popisu tema i aktivnosti koje bi sadržavale.
Teme i aktivnosti nisu predvideni kao entiteti objektnog modela procesa konstruiranja, stoga se ova matrica nece modelirati na isti nacin kao i npr. matrice zavisnosti izmedu objekata. Matricu "aktivnost-tema" pogodnije je modelirati skupovima atributa i operacijom, jer nije nužno da svaki redak, stupac ili celija matrice matrice budu zasebni objekti. Tako definirana klasa može onda biti dio procesa podrške kreiranju plana konstruiranja.

7.3.2 Elementi plana konstruiranja (dekompozicija i prikaz procesa konstruiranja planom)

Osnovu prikaza procesa konstruiranja u predloženom modelu ciniti ce dekompozicija procesa na "etape" koje cine zaokružene cjeline generiranja i transformacija informacija. Etapa procesa je proces obrade informacija koji je dio jednog konstruktorskog zadatka.
Kao metoda prikaza tijeka odvijanja procesa koristiti ce se usmjereni graf, prema razmatranjima u 4.4. Prema [12], niti jedna od postojecih metoda prikaza procesa ne može ispuniti sve zahtjeve opcenitog prikaza, što je detaljnije obrazloženo u poglavlju 4.5. Stoga ce se uoceni nedostaci odabrane metode prikaza procesa nastojati nadoknaditi implementacijom dodatnih semantickih elemenata i eventualnim kombiniranjem elemenata postojecih metoda prikaza procesa.
U predloženom pristupu cvor usmjerenog grafa modelirati ce "etapu" procesa konstruiranja. Bridovi usmjerenog grafa modeliraju moguce putanje redoslijeda izvodenja etapa i moguce putanje tokova informacija. Pri tome usmjereni graf ima strukturu mreže sa jednim pocetnim cvorom - prema razmatranjima izloženim u poglavlju 4.4.
Cvor plana i veze izmedu cvorova cine staticke elemente zapisa prikaza procesa. Skup operacija koje upravljaju izvodenjem procesa konstruiranja na temelju zapisa statickih elemenata i interakcija sa konstruktorom modeliraju dinamicki aspekt izvodenja procesa konstruiranja.
U nastavku izlaganja definirati ce se i opisati cvorovi plana, veze cvorova i njihov zapis.

7.3.3 Cvor plana konstruiranja

U predloženom modelu, etapa (cvor plana) procesa konstruiranja definira se kao kombinacija nepraznog skupa akcija (naredbi) A_j koje transformiraju objekt O, koji se nalazi u stanju S^O_i , u slijedece (novo stanje) S^O_{i+1} :

$$E = \{ A_n : S_j (A_j (S^O_i) = S^O_{i+1} \}$$

Definicija se može proširiti i na skup objekata, odnosno skup akcija transformira stanje svakog od objekata u slijedece novo stanje. Pri tome akcije mogu i generirati nove objekte i nakon toga im mijenjati stanje. U predloženoj definiciji "objekti" su: parametri, objekti prikaza proizvoda, objekti strukture proizvoda, konstrukcijski zadaci, te objekti prikaza relacija izmedu skupova instanci istih ili razlicitih klasa.

Predložena definicija etape procesa, odnosno cvora plana, analogna je definiciji "radnog prostora konstruiranja", prema [136]. Autori modeliraju proces konstruiranja pomocu "radnih prostora konstruiranja (design-working-space)" - skupova objekata koji cine okolinu za rješavanje konstrukcijskog problema. Radni prostor grupira objekte iz svih faza procesa konstruiranja i strukturira znanje o procesu konstruiranja da bi se odredilo (fiksiralo) stanje konstrukcije. Nova stanja izvode se iz definiranog stanja primjenom "uzoraka procesa".

Osnovna razlika modela predloženog u ovom radu u odnosu na model predložen u [136] i [137], je u cinjenici da se "radni prostor konstruiranja" temelji primarno na geometrijskom modeliranju konstrukcije. Model procesa konstruiranja predložen u ovom radu orijentiran je primarno na unapredenje organizacije i integracije racunalne podrške. Pri tome se entiteti realnog svijeta procesa konstruiranja nastoje preslikati u entitete objektnog modela koji ce ciniti sustav dovoljno fleksibilan da se pri implementaciji može prilagoditi razlicitim okruženjima izvodenja procesa konstruiranja. Istraživanja u ovom radu usmjerena su primarno na informacijsko modeliranje procesa konstruiranja.

Akcije u etapi procesa konstruiranja, prema definiciji etape, mijenjaju stanje objekata. Osnovni uvjet za izvodenje odredene akcije treba biti dakle postojanje samog objekta. Pored toga objekt mora biti u odredenom stanju koje akcija zahtijeva da bi se mogla izvršiti, odnosno da bi njeno izvršavanje imalo smisla. Cvor plana kao model etape procesa konstruiranja treba dakle sadržavati operacije provjere preduvjeta izvršavanja akcija. Planirati proces znaci odredivati što se u odredenoj etapi procesa treba postici, odnosno odrediti ciljeve svake etape. Ciljeve etape možemo promatrati i kao skup uvjeta koji moraju biti zadovoljeni, dakle cvor plana treba sadržavati operacije provjere zadovoljenja ciljeva.
U predloženom modelu preduvjeti i ciljevi modelirati ce se kao zahtijevano pocetno (ulazno) stanje objekata i zahtijevano konacno (izlazno) stanje objekata koji se obraduju u etapi procesa, tj. u cvoru plana konstruiranja.

Tako modelirana etapa procesa (cvor plana) sadrži skup akcija koje obavljaju transformacije od ulaznog stanja k izlaznom. Pod transformacijama se podrazumijevaju:

- promjene vrijednosti atributa objekata
- postavljanje ili mijenjanje relacija izmedu objekata
- pozivanje operacija objekata koji mijenjaju vlastito stanje ili stanje drugih objekata
- pozivanje "vanjskih" programskih alata koji mijenjaju stanje objekata prikaza proizvoda i objekata strukture proizvoda
- generiranje novih objekata, postavljanje i mijenjanje njihovih inicijalnih stanja

- pregledi i analize stanja (skupova atributa i objekata), te odlucivanje o daljnjem tijeku procesa konstruiranja

Cvor plana entitet je koji kontrolira procese obrade informacija na skupovima razlicitih klasa objekata. Postavlja se pitanje kako (pri kreiranju plana) u cvoru zapisati što treba raditi, na cemu, i kojim redoslijedom. Pri tome treba nastojati da takav zapis nema krutu sintaksu.
Stoga je ovdje odabrana koncepcija da se redoslijed obrade elemenata cvora implementira u skup operacija koje vode proces obrade cvora, a da zapis "na cemu" se obrada radi i zapis liste akcija koje rade obrade budu u obliku skupova referenci na objekte (slika 44). Objekte koje cvor referencira treba dakle kreirati nezavisno od samog cvora, tj. oni ne cine zapis cvora, pa se na taj nacin dobija na fleksibilnosti zapisa. Druga prednost takve koncepcije je u tome da se isti objekti mogu referencirati i iz više cvorova.
Takva koncepcija otvara i mogucnost promjene referenci cvorova u tijeku izvodenja plana, odnosno prilagodavanja plana nepredvidenim situacijama koje nastaju u tijeku izvodenja, što je zapravo implementacija dinamickog planiranja. Na predloženi nacin cak i sam plan konstruiranja (u tijeku izvodenja, bez interakcije sa konstruktorom) može mijenjati i prilagodavati "sam sebe", tj. akcije jednog cvora mogu na temelju uvjeta u pravilima odlucivanja promijeniti reference tog cvora ili reference drugih cvorova! Jedan od pristupa razradi inkrementalnog dinamickog planiranja prezentiran je u radu [138].

Cvor sadrži dakle skupove referenci na objekte modela procesa konstruiranja. Pri tome neki skupovi referenci mogu sadržavati objekte jedne i samo jedne klase, dok neki mogu sadržavati objekte razlicitih klasa. Nužno je tocno definirati što koji skup referenci može sadržavati radi algoritma procesa obrade cvora. Takve skupove referenci promatramo kao elemente cvora. Redoslijed obrade elemenata cvora (skupova referenci) implementiran je u operaciju procesa obrade cvora. Cvorovi plana mogu se klasificirati prema tome koje elemente sadrže i prema eventualno razlicitim algoritmima procesa obrade.
Jedan element obrade cvora skup je referenci na objekte modela procesa konstruiranja, pa treba odrediti i redoslijed obrade pojedinih referenci unutar samog skupa referenci.
Redoslijed obrade referenci unutar skupa može se zapisati u obliku tablice, gdje svaka referenca ima svoj "redni broj" obrade (tablica 8). Na predloženi nacin nije nužno zapisivati odmah reference onim redom kojim ih treba obradivati, nego se taj redoslijed može odrediti i zapisati naknadno. Pri tome se pretpostavlja da je skup referenci implementiran kao polje pokazivaca (pointera). Svakom clanu polja pokazivaca odgovara jedan clan polja redoslijeda obrade. Ovakav nacin zapisa takoder omogucuje i jednostavne promjene redoslijeda obrade referenci u tijeku izvodenja plana konstruiranja. Tablica može sadržavati i naziv objekta koji se referencira, radi preglednosti pri kreiranju plana konstruiranja. Primjer predloženog nacina zapisa redoslijeda obrade unutar skupa referenci cvora koji sadrži pet elemenata prikazan je tablicom 8.

polje pokazivaca na objekte	redni broj obrade
p [1]	3
p [2]	2
p [3]	1
p [4]	5
p [5]	4

Tablica 8: Primjer zapisa redoslijeda obrade skupa referenci cvora

Na analogan nacin može se riješiti zapis redoslijeda obrade ako skup referenci sadrži pokazivace na instance dviju razlicitih klasa. Shema takvog zapisa prikazana je tablicom 9. Svaka klasa ima "svoje" polje pokazivaca na objekte, pa je i zapis rednog broja obrade podijeljen na dva odgovarajuca polja. Ako je broj instanci klase A jednak n, a broj instanci klase B jednak k, tada redni brojevi imaju vrijednosti od 1 do $n+k$, i mora biti ispunjen uvjet da je $R_a \cap R_b = \varnothing$. Analogno se zapis redoslijeda može proširiti i na više od dvije klase objekata. Skup referenci potrebno je razdvojiti na zasebna polja pokazivaca za svaku klasu stoga što objektna baza može implementirati polje (skup) pokazivaca samo na skup instanci iste klase.

polje pokazivaca na objekte klase A	polje pokazivaca na objekte klase B	redni broj obrade objekta klase A	redni broj obrade objekta klase B
p_a [1]	p_b [1]	$i_a \in R_a$, R_a={1, ..n+k}	$i_b \in R_b$ R_b={1, ..n+k}
p_a [2]	p_b [2]		
. . .	. . .	. . .	. . .
. . .	p_b [k-1]		
p_a [n-1]	p_b [k]		i_{bk}
p_a [n]		i_{an}	

Tablica 9: Zapis redoslijeda obrade unutar dva skupa referenci cvora

U slijedecem poglavlju razmotriti ce se proces izvodenja cvora, a nakon toga rezimirati ce se što sve cvor kao složeni objekt treba sadržavati.

7.3.3.1 Proces izvodenja (obrade) cvora

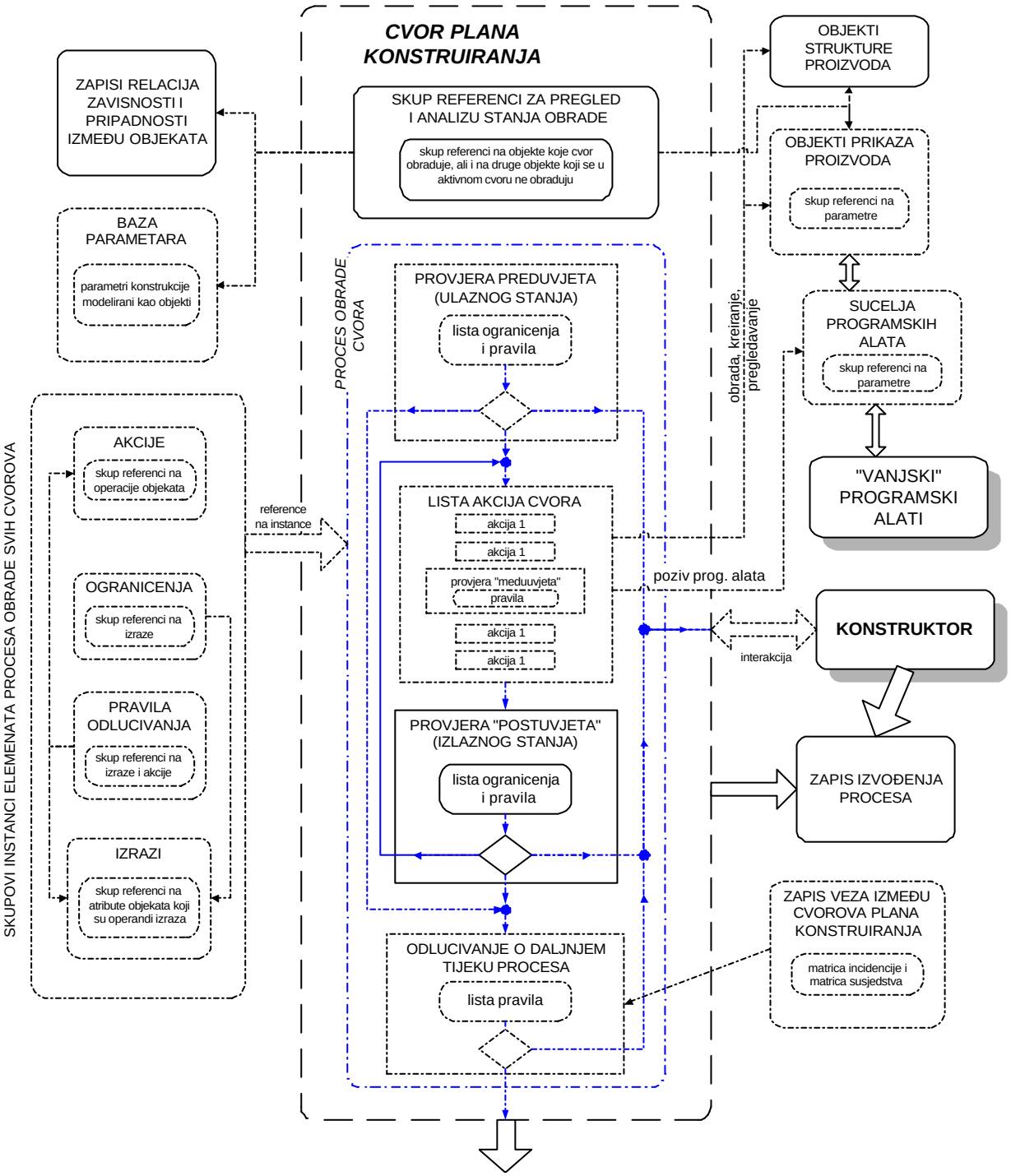
Pri razmatranju procesa izvodenja cvora pretpostavlja se da se istovremeno izvršava samo jedan cvor plana konstruiranja. U trenutku "aktiviranja" cvora pokrece se operacija koja upravlja izvodenjem cvora, tj. procesom obrade elemenata cvora. Shema procesa obrade elemenata cvora i reference cvora na skupove objekata modela procesa konstruiranja prikazani su na slici 44.

Osnovni koraci (etape) procesa obrade cvora mogu se naznaciti kao:

1. Provjera preduvjeta (zahtjevanog ulaznog stanja) - obraduje se skup referenci na ogranicenja i pravila odlucivanja. Zahtijevano (planirano) ulazno stanje skup je uvjeta i ogranicenja koji moraju biti ispunjeni da bi imalo smisla izvoditi predvidene akcije cvora.
2. Izvodenje liste akcija - procesira se skup referenci na akcije, no lista akcija može sadržavati i reference na ogranicenja ili pravila - kao provjere "meduuvjeta".
3. Provjera "postuvjeta", tj. da li je postignuto planirano izlazno stanje - obraduje se skup referenci na pravila i ogranicenja. Ako nije postignuto planirano izlazno stanje, moguc je povratak na neku od akcija (iz liste akcija) ili se proces obrade prekida. Povratak na odredenu akciju realizira se kroz interakciju s konstruktorom koji bira redni broj akcije koja ce se izvršiti. Proces obrade se u tom slucaju vraca na odabrani korak i nastavlja se dalje prema istim koracima kao i u prethodnom koraku iteracije.

4. Odlucivanje o daljnjem tijeku procesa - izbor i aktiviranje slijedeceg cvora, interakcija s konstruktorom ili prekid procesa. Slijedeci cvor bira se na temelju zapisa veza izmedu cvorova u matricama susjedstva i incidencije
5. Bilježenje tijeka obrade u "stanje procesa" tokom obrade, u trenutku eventualnog prekida procesa i po završetku obrade cvora.

Dijagram toka redoslijeda obrade elemenata cvora prikazan je na slici 44. Ujedno su shematski prikazane i direktne i indirektne reference cvora. Cvor sadrži reference na akcije, koje referenciraju operacije svih klasa objekata, te reference na ogranicenja i pravila koja referenciraju atribute svih klasa objekata modela procesa konstruiranja.
Preko direktnih i indirektnih referenci cvor zapravo grupira sve skupove informacija (objekte) koji se obraduju u jednoj etapi procesa konstruiranja. Na taj nacin modeliran je "radni prostor" koji sadrži i model izvodenja procesa konstruiranja. Obrada, provjera informacija i odlucivanje izvode se po unaprijed planiranom redoslijedu koji se u tijeku izvodenja može prilagodavati novonastalim (nepredvidenim) situacijama.
Osim elemenata (skupova referenci) koji se u cvoru procesiraju, cvor može sadržavati i skup "ulaznih" i skup "izlaznih" referenci koje nisu ukljucene u proces obrade. Ti skupovi referenci predvideni su kao "pregled" stanja objekata koji konstruktor može aktivirati na pocetku i na kraju izvodenja cvora ili u trenutku prekida procesa obrade. Na taj nacin konstruktor može u željenom trenutku dobiti uvid u stanje razvoja konstrukcije. U navedenim skupovima mogu biti referencirani i objekti koji se ne obraduju u cvoru koji ih referencira, ali su relevantni za analize i usporedbe, odnosno za usmjeravanje tijeka procesa i odlucivanje.



Slika 44: Reference cvora na druge objekte i proces izvodenja cvora

Predloženi nacin izvodenja (obrade) cvora (cvorova), odnosno plana sadrži i neke elemente "oportunistickog" i dinamickog planiranja. Zapravo svaki cvor funkcionira kao svojevrstan "otok" planiranja koji sam rješava svoj dio posla u etapi procesa konstruiranja.

Provjera preduvjeta i postuvjeta koji provodi cvor može se promatrati kao usporedba unaprijed planirane i stvarne dinamicke situacije u tijeku izvodenja plana. Pregled stanja referenciranih objekata koristi konstruktor u situacijama kad se proces obrade cvora prekida i traži intervencija konstruktora.

Cvor kao "otok planiranja" modeliran je svojom operacijom obrade elemenata cvora, odnosno ugradenim znanjem "što treba ciniti" u odnosu na situacije koje mogu nastupiti u tijeku izvodenja procesa.

Algoritam procesa obrade elemenata cvora odvija se po slijedecim fazama (dijagram toka prikazan je na slici 44):

1. Provjera preduvjeta:
   - ako su sva ogranicenja zadovoljena, i uvjeti pravila tako odreduju, prelazi se na izvodenje liste akcija
   - ako neko od ogranicenja nije zadovoljeno ili se u nekom od pravila tako odredi, moguce su dvije putanje nastavka:
     - traženje intervencije konstruktora, nakon koje on bira tocku nastavka procesa
     - skok na odlucivanje o daljnjem tijeku procesa (tocka 4)
2. Lista akcija:
   - redom se izvode akcije, izmedu kojih se mogu pojavljivati i provjere meduuvjeta, koje mogu rezultirati sa istim putanjama nastavka kao i nakon provjere preduvjeta
3. Provjera postuvjeta:
   - ako su sva ogranicenja zadovoljena, i ako uvjeti pravila tako odreduju, prelazi se na odlucivanje o daljnjem tijeku procesa, odnosno izbor slijedeceg cvora plana konstruiranja
   - ako neko od ogranicenja nije zadovoljeno ili se u nekom od pravila tako odredi, moguce su dvije putanje nastavka:
     - traženje intervencije konstruktora, nakon koje on bira tocku nastavka procesa
     - povratak na odredenu akciju u listi akcija, dalje se izvode akcije koje slijede
4. Odlucivanje o daljnjem tijeku procesa:
   - na temelju zapisa veza izmedu cvorova plana u matricama incidencije i susjedstva, te na temelju skupa pravila, bira se slijedeci cvor plana i pokrece proces njegove obrade
   - ako nastupe nepredvidene situacije, ili uvjeti pravila tako odreduju, traži se intervencija od konstruktora, koji odreduje slijedeci cvor ili prekida proces

Plan konstruiranja, odnosno mreža veza izmedu cvorova prikaz je tokova podataka, mogucih (unaprijed planiranih) putanja i redoslijeda izvršavanja, kao i situacija izvodenja. Na temelju znanja zapisanog u skupu pravila, cvor može odlucivati koje ce dalje cvorove aktivirati i kojim redoslijedom - odnosno da li se može realizirati planirani redoslijed ili se treba prilagoditi trenutnoj situaciji, odnosno tražiti odluku od konstruktora što treba dalje raditi. Prilagodba novoj situaciji može se realizirati promjenom zapisa elemenata cvora, ali i dogradivanjem plana u tijeku izvodenja - dodavanjem novih veza i cvorova.

7.3.3.2 Reference, operacije i atributi cvora

Nakon prikaza procesa izvodenja cvora, ovdje ce se rezimirati što sve objekt "cvor plana konstruiranja" treba sadržavati. Cvor plana najkompleksniji je entitet modela procesa konstruiranja koji "grupira" sve potrebne objekte za modeliranje "radnog prostora" izvodenja etape procesa konstruiranja. Model radnog prostora cine skupovi referenci na razlicite klase objekata i skup operacija koje upravljaju procesom izvodenja cvora.

Cvor plana sadrži slijedece skupove referenci:

- Za pregled i analizu stanja obrade - reference na parametre, objekte prikaza proizvoda, objekte strukture proizvoda, i objekte zapisa relacija zavisnosti i pripadnosti. Ovi skupovi referenci nisu ukljuceni u proces obrade cvora.
- Na elemente procesa obrade:
  - ogranicenja i pravila navedena u provjeri preduvjeta
  - akcije koje su navedene u listi akcija
  - ogranicenja i pravila provjere "postuvjeta"
  - skup pravila odlucivanja o daljnjem tijeku procesa konstruiranja

Svaki od navedenih skupova referenci mora imati i svoje liste redoslijeda obrade (prema tablicama 8 i 9).
Navedeni skupovi referenci mogu se promatrati kao "direktne" reference cvora. Ogranicenja i pravila sadrže reference na izraze i akcije, izrazi sadrže reference na atribute objekata, a akcije sadrže reference na operacije objekata - pa sve ove reference možemo promatrati kao "indirektne" reference cvora. Opcenito gledano, na taj nacin cvor plana može u procesu obrade "dohvatiti" neki od atributa ili pozvati neku od operacija bilo kojeg objekta modela procesa konstruiranja. Implementacija ovakve koncepcije razraditi ce se u osmom poglavlju.

Cvor, kao najkompleksniji entitet predloženog modela procesa konstruiranja sadrži i najveci dio modela dinamike izvodenja procesa koji se realizira skupom operacija. Osnovna operacija koja kontrolira proces obrade cvora poziva se u trenutku aktiviranja cvora. Osnovne tocke algoritma procesa obrade cvora navedene su u prethodnom poglavlju.

Pored implementacije opisanog algoritma obrade, operacije cvora trebaju modelirati i podržavati i slijedece funkcije:

- Dekodiranje (parsiranje) zapisa elemenata obrade cvora i zapisa redoslijeda obrade unutar pojedinog skupa referenci.
- Zapis tijeka izvodenja procesa, koji bi se trebao realizirati tako da je, izmedu skupa opcija, moguce birati što ce se, i u kojim fazama obrade procesa zapisivati. Svakako bi trebalo zapisivati kroz koje grane prolazi proces obrade, odnosno koji uvjeti su bili ispunjeni.
- Implementaciju sucelja za interakcije s konstruktorom:
  - Sucelje odlucivanja o daljnjem tijeku procesa - konstruktor treba imati pregled redoslijeda obrade referenciranih elemenata cvora, i mogucnost analize stanja skupova objekata.
  - Sucelje za formiranje upita za pregled i analizu stanja obrade.
  - Sucelje za pregled zapisa izvodenja procesa i za dodavanje primjedbi (u zapis izvodenja procesa konstruktor može upisivati i svoja objašnjenja - koja je situacija nastupila, kako ju je riješio, što je eventualno promijenio u planu.
- Implementaciju "poštanskog sanducica" za slanje i primanje poruka od drugih cvorova. Spomenute poruke predlažu se kao "podsjetnici" na situacije koje nastupaju u izvodenju plana, koji cine dio zapisa cvora. Poruke može zapisivati operacija izvodenja cvora i konstruktor u tijeku prekida procesa obrade. U iterativnim procesima neki cvorovi ce se izvoditi više puta, pa zapisi primjedbi u svakom izvodenju cvora mogu biti od koristi za usmjeravanja procesa iteracije. Analogno, cvor može "poslati" (zapisati) poruku drugom cvoru, koja može biti korisna informacija pri izvodenju, odnosno analizi stanja obrade

drugog cvora. "Poštanski sanducic" treba odvojiti od "zapisa izvodenja procesa", jer poruke ovdje promatramo kao informacije koje su od koristi za usmjeravanje tijeka izvodenja plana procesa konstruiranja. Takav nacin zapisa "poruka" bio bi od znacajne koristi u slucaju da se više cvorova izvodi istovremeno, kao razliciti racunalni procesi. Kako takva pretpostavka zahtijeva daleko složenije koncepcije implementacije procesa obrade cvora (i izvodenja cijelog plana procesa konstruiranja), u opsegu ovog rada ona se nece razmatrati.

Kako cvor sadrži skup operacija i skupove referenci (a ne zapise elemenata), znaci da na odredenoj razini ima relativno staticnu strukturu. Implementacija razlicitih varijanti strukture cvora i procesa obrade cvora može se stoga riješiti klasifikacijom. Pri generiranju plana treba dakle kreirati instance odredene klase cvorova i inicijalizirati njihove skupove referenci (pointera), s time da se redoslijed obrade unutar skupova referenci može odrediti i mijenjati naknadno.

Predlažu se slijedeci atributi cvora:

- naziv cvora u planu procesa konstruiranja
- oznaka cvora u zapisu plana konstruiranja - matrici incidencije i matrici susjedstva
- opis cvora - opis etape procesa konstruiranja, uloge i zadatka cvora u kontekstu plana procesa konstruiranja
- redni broj ponavljanja izvodenja cvora
- zapis promjena stanja cvorova koja su nastupila u tijeku izvodenja plana u odnosu na originalno (pocetno planirano i zapisano stanje)

7.3.3.3 Veze cvorova u planu konstruiranja

Cvor plana modelira jednu etapu procesa konstruiranja, a skup cvorova i veza izmedu njih modeliraju cjelokupni proces konstruiranja. U poglavlju 4.4 razmatrana je topologija prikaza akcija u procesu konstruiranja i matricni prikazi usmjerenog grafa.
Usmjereni graf odabran je kao metoda prikaza tijeka odvijanja procesa konstruiranja. Takav graf ima jedan "pocetni cvor", a ostali cvorovi mogu biti povezani u obliku mreže (tj. svaki cvor može biti vezan sa bilo kojim drugim cvorom). Pocetni cvor razlikuje se od ostalih cvorova po tome što nema niti jedan "ulazni brid" (vezu). Krajnji cvorovi takvog grafa su oni koji nemaju niti jedan izlazni brid (vezu). Osnovna pretpostavka razmatranja koja slijede jest da se istovremeno može izvoditi samo jedan cvor plana konstruiranja.
U poglavlju 4.5 naglašeno je da (prema [12]) niti jedna do sada razvijena metoda prikaza procesa (i opcenite i specificne namjene) ne može ispuniti sve zahtjeve za prikazom procesa konstruiranja. Stoga ce se u predloženom modelu razmotriti mogucnosti i pretpostavke dodavanja nekih semantickih elemenata u prikaz procesa pomocu usmjerenog grafa.
Jedna od takvih mogucnosti je razlikovanje više vrsta (klasa) veza izmedu cvorova grafa. U matricnom prikazu grafa svaka vrsta veze može imati svoju oznaku koja se zapisuje u celiju matrice.

U ovoj fazi istraživanja predlažu se dvije vrste veza:

- veze dozvoljenih putanja (redoslijeda) izvršavanja cvorova u planu
- veze "prijenosa" informacija i "ažuriranja" zapisa elemenata cvora

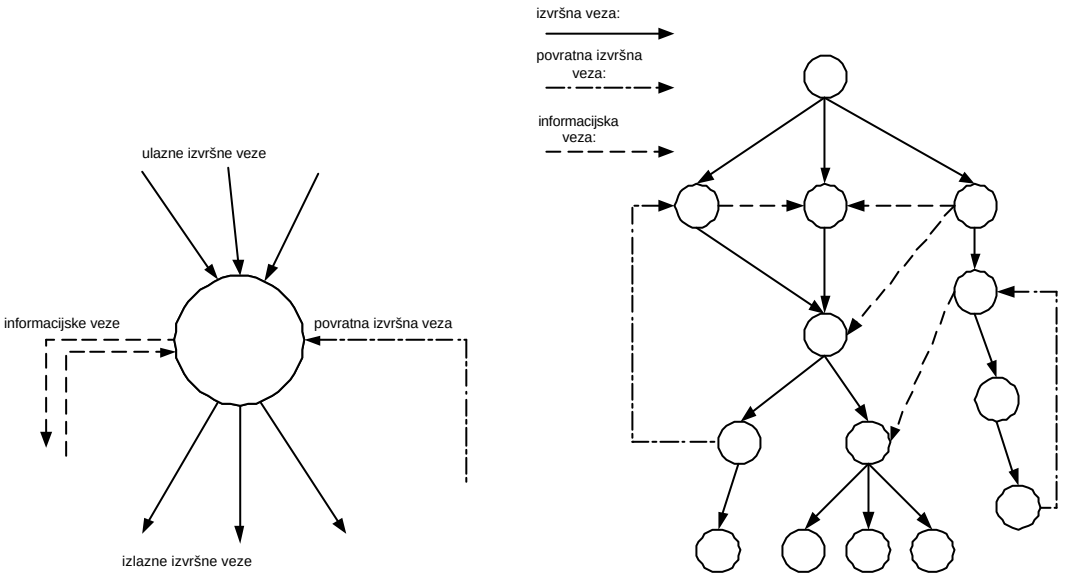
Pri tome svaka dva cvora u planu mogu biti povezana sa obje vrste veza ili samo jednom od njih. Obje vrste veza mogu biti i ulazne i izlazne iz cvora.

Veze dozvoljenih putanja izvršavanja oznacavaju koji se slijedeci cvorovi mogu aktivirati nakon završetka obrade odredenog cvora. Takve veze predstavljaju "planirani" redoslijed izvršavanja plana koji se zapisuje u matricni prikaz grafa. Ukoliko na temelju pravila odlucivanja o nastavku procesa (na kraju obrade cvora) nije moguce odrediti slijedeci cvor, proces obrade cvora tražiti ce odluku od konstruktora. Konstruktor tada ne mora poštivati predvidene veze izmedu cvorova, nego može odabrati bilo koji cvor za nastavak procesa. U daljnjem tekstu ova vrsta veza oznacavati ce se kao "izvršne" veze.
Ponovno izvršavanje cvora u iterativnom ciklusu može u nekim situacijama zahtijevati drugacije algoritme procesa obrade cvora od algoritma obrade pri prvom aktiviranju. Stoga je važno "povratnu" vezu takoder izdvojiti kao posebnu vrstu (klasu) veze. Pocetni cvor plana može imati "povratnu ulaznu vezu", ali ne i druge izvršne veze kao ulazne.

Veze prijenosa informacija oznacavaju dozvoljene utjecaje cvora koji se obraduje na druge cvorove plana. Pri tome "utjecaj" ovdje znaci da aktivni cvor može mijenjati neki dio zapisa neaktivnog cvora, što je zapravo implementacija dinamickog planiranja, tj. prilagodavanja plana novonastalim situacijama u tijeku izvodenja. Aktivni cvor dakle može promijeniti popis referenci, redoslijed izvodenja referenci, te zapise izraza, pravila i akcija. Zapis ove vrste veza nema bitnog utjecaja na tijek izvodenja plana, ali u cjelini zapisa plana daje sliku o tome koji dijelovi plana i "sa kojih mjesta" se mogu mijenjati u tijeku izvodenja. Za ovu vrstu veza u daljnjem tekstu koristiti ce se naziv "informacijske" veze.

Zapis, odnosno slanje poruka izmedu cvorova nije vezano uz informacijske veze, tj. aktivni cvor može poslati (zapisati) poruku bilo kojem drugom cvoru plana.

Primjer razmatranih vrsta veza izmedu cvorova prikazan je na slici 45.



Slika 45: Vrste veza cvorova plana konstruiranja

Izložena razmatranja veza izmedu cvorova u usmjerenom grafu oslanjaju se na pretpostavku da se istovremeno izvršava samo jedan cvor. Pretpostavka izvodenja više cvorova plana istovremeno zahtijeva razvoj daleko složenijih procesa obrade cvorova, sinhronizacije i komunikacije izmedu cvorova. Takoder bi tada trebalo riješiti tocke "dijeljenja" jednog procesa na više njih i tocke "spajanja" više procesa u jedan, eventualno sve pod kontrolom jednog zajednickog procesa na višoj razini. Detaljna razmatranja ovih otvorenih tema izvan su opsega ovog rada, ali ovdje ce se ipak razmotriti kakvi su moguci

utjecaji pretpostavke izvodenja više cvorova istovremeno na metodu zapisa plana pomocu usmjerenog grafa. Postavlja se stoga pitanje da li metoda usmjerenog grafa može (uz eventualne semanticke dodatke) biti osnova prikaza odvijanja procesa konstruiranja i pod pretpostavkom izvršavanja više cvorova istovremeno.

S takvom pretpostavkom informacijske veze cvorova imaju daleko veci znacaj u procesu izvodenja plana.
Ako se više cvorova izvršava paralelno, može biti važno da neki cvor proslijedi informaciju cim ju odredi (definira), da bi odmah postala raspoloživa drugim aktivnim cvorovima.
U takvoj situaciji "informacijske veze" mogu se promatrati i kao "feedforward" veze, tj. cim neki od cvorova odredi vrijednost nekog kljucnog parametra (ili atributa objekta) može o tome obavijestiti druge cvorove s kojima je povezan informacijskim vezama i kojima je taj podatak nužan. Na taj nacin može se modelirati proces paralelnog izvodenja informacijski spregnutih zadataka (ta tema razmatrana je u poglavlju 7.2.3).

Prva mogucnost semantickog proširenja usmjerenog grafa koja se namece pod pretpostavkom izvodenja više cvorova istovremeno je postavljanje "AND" i "OR" uvjeta na mjestima više izlaznih ili više ulaznih grana u cvorove. Tako postavljeni uvjeti odredivali bi zapravo tocke dijeljenja ("AND" izlaz) jednog procesa na više paralelnih podprocesa i tocke spajanja ("AND" ulaz) više paralelnih podprocesa u jedan proces.

Na slici 46 prikazan je primjer usmjerenog grafa kao prikaza plana uz dodatak "AND" i "OR" uvjeta na mjestima gdje postoji više ulaznih ili izlaznih veza cvorova. Promatrajuci ulazne i izlazne veze cvorova mogu se razluciti cetiri razlicite situacije:

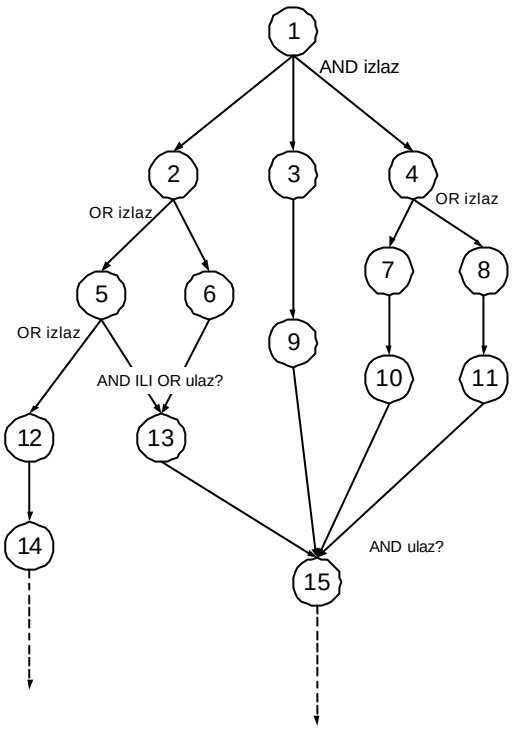
- nakon izvršavanja odredenog cvora slijedi mogucnost izvršavanja samo jednog cvora kao slijedeceg u procesu
- jedan cvor ima jednu ulaznu vezu, tj. aktivirati ce se pozivom tocno odredenog cvora koji mu prethodi
- postoji više izlaznih veza, tj. nakon odredenog cvora slijedi mogucnost izbora nastavka izvršavanja procesa izmedu više drugih cvorova
- jedan cvor ima više ulaznih veza, tj. taj cvor aktivirati ce se pozivom nekog od nekoliko mogucih prethodnika

Prve dvije navedene situacije za razmatranu temu nisu interesantne. Za druge dvije situacije kao pocetnu pretpostavku uzmimo slijedece vrste (kombinacije) postavljanja uvjeta izvodenja slijednika i prethodnika odredenog cvora:

- Svi slijednici cvora (više izlaznih veza) izvršavaju se paralelno ("AND " izlaz) - nit kontrole procesa dijeli se na više istovremenih podprocesa.
- Izvršava se samo jedan izmedu više mogucih slijednika cvora ("OR" izlaz).
- Izvršava se nekoliko od, ali ne svi slijednici cvora (kombinacija "AND" i "OR" operatora).
- Preduvjet izvršavanja cvora je da su svi prethodnici bili izvršeni ("AND" ulaz).
- Preduvjet izvršavanja cvora je izvršavanje bilo kojeg od prethodnika ("OR" ulaz).
- Preduvjet je izvršavanje nekoliko od prethodnika (ali ne nužno svi) (kombinacija "AND" i "OR" operatora).

Lako je pokazati da sve kombinacije ovakvih postavki nije uvijek moguce u planu realizirati za jedan cvor nezavisno od ostalih cvorova i strukture cijelog grafa. Naime, dolaziti ce do preklapanja (pobijanja) postavljenih izlaznih (za slijednike) i ulaznih (za prethodnike) uvjeta izvršavanja. Na slici 46 to se uocava na primjerima cvorova 13 i 15.

Ako se kao preduvjet izvršavanja cvora 13 zahtijeva da i cvor 5 i cvor 6 budu izvršeni, onda izlaz iz cvora 2 ne bi trebao biti "OR" nego "AND".
Isto tako izlaz iz cvora 5 u tom slucaju trebao bi biti "AND". Ako cvor 15 "spaja" sve procese iz cvorova 13, 9, 10 i 11 tada na cvoru 4 ne bi trebao biti "OR" izlaz nego "AND".
Ili se npr. može postaviti da cvor 15 zahtijeva izvodenje cvorova 13 i 9 ili 10 ili 11. Medutim uvjeti izvodenja cvora 13 odredeni su i uvjetima izvodenja cvorova 5 i 6.



Slika 46: Povezivanje razlicitih vrsta cvorova plana

Postavlja se dakle pitanje ima li onda smisla implementirati ovakav nacin odredivanja uvjeta paralelnog izvršavanja više cvorova plana. Svakako da bi opisani problem postao daleko izraženiji i teže rješiv na složenijem primjeru plana.

Da bi se osigurala konzistentnost razmatranih uvjeta, u proces kreiranja plana trebalo bi implementirati algoritam kontrole mogucnosti postavljanja uvjeta. U svakoj situaciji postavljanja novog "AND" uvjeta, kontrolna procedura trebala bi analizirati citav do tada kreirani plan i odrediti da li je novi uvjet moguce postaviti ili ne. Takva kontrolna procedura trebala bi simulirati izvodenje plana i pri tome "oznacavati" izvršene cvorove. Svakom cvoru plana pripadala bi jedna logicka varijabla kojoj bi se mijenjalo stanje nakon "prolaza kroz cvor", odn. izvršavanja cvora. Nakon simulacije izvršavanja plana, na temelju analize stanja "oznaka" neposrednih prethodnika cvora na kojem se postavlja uvjet, moglo bi se odrediti da li je uvjet moguce postaviti bez kolizije s drugim prethodno postavljenim uvjetima. Medutim, za tako postavljenu koncepciju kontrolnog algoritma treba još istražiti kako bi se riješile povratne veze, tj. petlje.

Detaljnom analizom i razradom predloženog algoritma kontrole ovaj rad se nece dalje baviti. Stoga ce se još samo razmotriti jedno od jednostavnijh rješenja razmatranog problema - da se uvjeti paralelnog izvodenja više cvorova postavljaju ili samo za dijelove grafa sa više slijednika cvora (na izlazu iz cvora) ili samo za više prethodnika (na ulazu u cvor). Pri tome se "OR" ilzaz i "OR" ulaz smatraju "normalnim" nacinom izvodenja plana koji se u grafu ne moraju posebno oznacavati.

Može se pretpostaviti da je znacajnije u planu unaprijed odrediti što treba uciniti na izlazu iz cvora. Preduvjeti ulaza, odnosno izvršavanja cvora više su staticke prirode. Ovdje se pretpostavlja da je lakše rješavati moguce neplanirane situacije na ulazu u cvor, (u tijeku izvodenja plana), a unaprijed u planu predvidjeti nacine izvodenja izlaznih grana.

Medutim ostaje i dalje otvoreno pitanje nacina spajanja više podijeljenih procesa. Npr. ako cvor 1 ima "AND" izlaz, nastavlja se sa tri paralelna procesa. Ako se nakon cvora 5 prede na cvor 13, i ako cvorovi 2, 4, i 5 imaju "OR" izlaze onda bi se sva tri procesa mogla spojiti opet u cvoru 15. U takvoj situaciji trebalo bi odrediti da li onda cvor 15 treba cekati završetak svih od cvorova 13, 9, 10 ili 11, ili bi mogao krenuti cim jedan od njih završi? Druga mogucnost je da se nakon cvora 5 prede na cvor 12. Tada bi se procesi koji su krenuli iz cvorova 3 i 4 spojili u cvoru 15, ali bi ostao aktivan i proces sa cvora 12, pa ako se dalje dodaju novi uvjeti, veze i grananja u grafu koji dodatno kompliciraju situaciju …

Na temelju izložene analize može se zakljuciti da prikaz procesa usmjerenim grafom može prikazati proces izvodenja plana i uz pretpostavku izvodenja više cvorova istovremeno, ali uz odredena ogranicenja i moguce komplikacije. Stoga bi trebalo istražiti koji su nužni uvjeti i ogranicenja implementacije tako postavljene koncepcije prikaza i kontrole izvodenja procesa konstruiranja. Metoda prikaza matrice incidencije usmjerenog grafa svodenjem na dvije liste, (tablica 1) omogucuje relativno jednostavnu implementaciju zapisa uvjeta izvodenja slijednika odredenog cvora.

7.3.3.4 Zapis stanja cvorova ("stanje procesa")

Pretpostavka istovremenog izvršavanja više cvorova plana zahtijeva dopune zapisa plana i atributa cvorova, na temelju kojih bi aktivni cvorovi mogli “komunicirati” u tijeku izvodenja plana. Stoga ce se ovdje razmotriti “stanje cvora” kao dodatni atribut cvora i “zapis stanja cvorova” kao podloga za razvoj algoritama uskladivanja izvodenja više paralelnih procesa obrade cvorova.

Proces konstruiranja u velikoj vecini slucajeva timski je rad, dakle odvija se na više radnih mjesta paralelno, a svi rade na istom proizvodu. Plan procesa konstruiranja može se onda postaviti za svakog clana grupe posebno, s time da bi trebao postojati i zajednicki plan koji modelira koordiniranje pojedinacnih aktivnosti i upravlja tokovima razmjene podataka izmedu clanova tima. Druga mogucnost (koja se cini jednostavnijom za realizaciju) jest da se model procesa konstruiranja fokusira na skup zadataka koje treba riješiti, bez obzira da li na njima radi jedna ili više osoba. U takvom pristupu, jedan plan procesa konstruiranja imao bi u trenutku izvodenja jedan ili više aktivnih podprocesa koji bi se odvijali na razlicitim cvorovima racunalne mreže. Sustav distribuirane obrade dijelova konstrukcije razvijaju u [139], na temelju internet tehnologije.

Za uskladivanje izvodenja više podprocesa plana, nužno je da aktivni cvorovi imaju “uvid” u stanje svih ostalih cvorova u mreži, posebno u trenucima odlucivanja o nastavku procesa.
To znaci da u toku izvodenja plana treba postojati “zapis stanja procesa” koji je na raspolaganju svim aktivnim cvorovima. Svaki cvor plana u jednom trenutku nalazi se u jednom od mogucih stanja, i to stanje zapisano je u “stanju procesa”. Svaka promjena stanja nekog cvora odmah se bilježi u “stanju procesa”. Pri tome treba osigurati da, u trenutku dok neki od cvorova analizira stanje procesa, drugi cvorovi ne smiju mijenjati svoja stanja.
Osnovna stanja cvora su “aktivan” i “nije aktivan”.

Predlažu se slijedece varijante aktivnog stanja cvorova:

- obavlja operacije provjere preduvjeta - za to vrijeme ne smije se mijenjati stanje drugih cvorova
- izvodi svoj niz akcija ili provjera postuvjeta
- u tijeku je odlucivanje o nastavku procesa – za to vrijeme ne smije se mijenjati stanje drugih cvorova
- u interakciji s konstruktorom, i ceka njegov odgovor
- cvor je aktivan, ali proces obrade stoji i ceka: odgovor poruke koju je poslao drugom cvoru, ili na informacije od drugih cvorova, ili da neki drugi cvor završi svoj proces obrade
- cvor "analizira" stanje procesa – za to vrijeme drugi cvorovi ne smiju mijenjati svoja stanja

Pored stanja cvora, u zapisu stanja procesa trebalo bi za svaki cvor voditi i evidenciju aktiviranja, te za svako aktiviranje zapisati i koji ga je prethodnik "pozvao", odnosno pokrenuo.
Iz izloženog može se zakljuciti da pretpostavka izvodenja više cvorova plana istovremeno otvara citavih niz novih pitanja i problema koje treba riješiti. Pri tome treba istražiti ogranicenja mogucnosti implementacije, razviti koncepte uskladivanja više razlicitih procesa obrade cvorova, tj. "sustav semafora" za komunikaciju preko "stanja procesa". Stoga ce se daljnjim razmatranjima zadržati pretpostavka izvodenja samo jednog cvora istovremeno. Istraživanjima sustava slicne koncepcije bave se Clarkson i Hamilton [119]. Uz svaki konstrukcijski zadatak oni vežu status zadatka i razinu "pouzdanosti" vrijednosti parametara zadatka, kojima se odreduje da li izvodenje zadatka može zapoceti ili ne.

7.3.4 Sucelje za prikaz plana konstruiranja

Predložena koncepcija plana konstruiranja može se promatrati kao skup osnovnih objekata - (cvorova i matrice zapisa veza cvorova) i skup svih ostalih objekata koje cvorovi referenciraju direktno ili indirektno. Svi ti objekti spremljeni su u objektnoj bazi. Dakle plan nema "formalnog" zapisa u obliku npr. nizova (slogova) leksickih elemenata. Odreden oblik takvog zapisa mogao bi se formirati, no on zasigurno ne bi mogao obuhvatiti sve razine referenciranja izmedu objekata. Bilo da se radi o "formalnom" zapisu plana, bilo o skupu objekata u objektnoj bazi, iz takvih zapisa teško je (gotovo nemoguce) steci pregled cijelog plana, odnosno tijeka izvodenja. Stoga je potrebno koncipirati skup operacija sucelja koje ce formirati pregledne prikaze plana konstruiranja na temelju zapisa svih referenci izmedu objekata u objektnoj bazi. Takav pregled plana nužan je za kontrolu plana prije izvodenja i u tijeku izvodenja, ali prije svega je konstruktoru potreban u procesu kreiranja novog plana.
Sucelje prikaza plana konstruiranja pretežno ce sadržavati razne varijante upita (query-a) koje trebaju realizirati prikaze referenciranih objekata kroz više razina referenciranja. Npr. cvor ima skup referenci na pravila i listu redoslijeda njihovog izvodenja, svako pravilo ima reference na izraze i akcije, izrazi na atribute objekata, a akcije na operacije objekata. Sucelje prikaza plana treba podržavati prikaze plana za razlicite situacije koje nastupaju u procesima kreiranja, kontrole i izvodenja plana. Operacije sucelja prikaza plana nisu dio samog plana konstruiranja, jer su to dijelovi programske realizacije sustava koji se ne spremaju u objektnu bazu.

7.3.5 Kreiranje plana konstruiranja

Objektna baza sastoji se od dviju komponenti - "rjecnika baze" (database dictionary) i same baze u koju se spremaju objekti. Rjecnik baze zapis je svih definicija klasa, njihovih relacija i hijerarhijske strukture. Na jednom rjecniku može se temeljiti više baza koje ce sve imati istu strukturu klasa, ali razlicite skupove instanci. Stoga svaki plan konstruiranja može biti zapisan u posebnu objektnu bazu. Tako svaku bazu kreiranu na temelju rjecnika koji sadrži sve definicije klasa modela procesa konstruiranja možemo promatrati kao "prazni" plan. U procesu kreiranja plana treba generirati sve potrebne instance klasa (objekte) i inicijalizirati reference izmedu njih (pointere), te takve objekte spremiti u bazu. Na kraju procesa kreiranja plana baza ce sadržavati zapis "gotovog" plana koji se može koristiti.
Procesom kreiranja plana treba upravljati skup operacija koje trebaju osigurati slijedece osnovne funkcije:

- Prikaz plana - sucelje koje u svakom koraku kreiranja plana treba omoguciti:
  - prikaz svih kreiranih elemenata plana, do najniže razine referenciranja (prema prethodnom poglavlju),
  - razlicite varijante parcijalnih prikaza strukture - upite i pretraživanja, unakrsne preglede,
  - prikaze raspoloživih varijanti izbora u razlicitim situacijama (posebice pri referenciranju).
- Mehanizme kontrole osiguranja sintakticke ispravnosti i potpunosti plana i referenciranih elementa.
- Mehanizme ažuriranja strukture plana uz osiguranje referencijalnog integriteta, (npr. brisanje ili promjena strukture instanci koje referenciraju ili su referencirane).
- Povezivanje sa bazama znanja i podataka o proizvodu i procesu konstruiranja.

Jedan dio postupaka kontrole ispravnosti elemenata plana i njihovih referenci trebao bi se implementirati u operacije kreiranja nove instance svake od klasa. Svaka klasa mora sadržavati posebnu vrstu operacije za kreiranje svojih instanci, tj. objekata. Takva operacija u objektnoj terminologiji naziva se "konstruktor". Dakle svaka klasa mora u svojoj operaciji kreiranja instanci sadržavati implementaciju postupka kontrole ispravnosti u odnosu na proces kreiranja plana. Na taj nacin neke od mogucih grešaka mogle bi se eliminirati vec na zacetku.

Operacije za podršku procesu kreiranja plana trebalo bi koncipirati tako da se u svakom koraku (situaciji) kreiranja plana provjerava sintakticka ispravnost svakog elementa koji se u tom trenutku obraduje. U ovoj fazi istraživanja ne može se sa sigurnošcu tvrditi da je moguce osigurati potpunu kontrolu, tj. da ce uvijek biti kreiran sintakticki potpuno ispravan plan. Bez obzira na potvrdu ili pobijanje ovakve teze, svaki od entiteta modela procesa konstruiranja treba i u svojim operacijama sadržavati kontrolne mehanizme. Ovdje se misli na operacije koje se aktiviraju u tijeku izvodenja procesa konstruiranja - npr. dekodiranje (parsiranje) izraza, proces obrade cvora, itd. Kontrolni mehanizmi u operacijama entiteta modela procesa pri obradama grešaka u najviše slucajeva tražiti ce intervenciju od konstruktora.

Druga vrsta problema je semanticka ispravnost i "kvaliteta" plana. Operacije podrške kreiranju plana ni na koji nacin ne mogu osigurati izradu "kvalitetnog", tj. "dobrog" plana, cije ce izvodenje sigurno (i možda optimalno) dovesti do željenog cilja. Koliko je plan "dobar" ili nije, može se provjeriti tek u procesu izvodenja plana.

Skup operacija kreiranja plana može se takoder uvrstiti u entitete modela procesa konstruiranja. Takav skup operacija može se implementirati kao jedna tzv. "servisna" klasa, koja nece imati instance. Takva klasa ne sprema se u objektnu bazu, jer nije dio odredenog plana, nego je zajednicka za sve planove konstruiranja koji se kreiraju u odredenom okruženju.

Kao podloga za nastavak istraživanja, predložiti ce se neke operacije i koraci procesa kreiranja plana:

- kreiranje parametara, odredivanje i upis u bazu pocetnih vrijednosti parametra (koje su npr. poznate iz liste zahtjeva)
- popunjavanje matrice meduzavisnosti parametara (za novu konstrukciju popunjavanje, a za varijantnu samo doradivanje vec postojece matrice)
- definiranje funkcionalnih zavisnosti izmedu parametara - zapisivanje matrice meduzavisnosti parametara
- definiranje objekata prikaza konstrukcije
- definiranje zadataka, odredivanje ciljeva zadataka, popunjavanje matrice zavisnosti konstrukcijskih zadataka
- "preslagivanje" matrice zavisnosti konstrukcijskih zadataka, odredivanje sekvencijalnih, paralelnih i spregnutih zadataka
- definiranje cvorova plana (u odnosu na zadatke):
  - odredivanje pocetnih uvjeta izvršavanja, lista akcija, postuvjeta
  - kreiranje izraza, akcija i pravila odlucivanja
  - odredivanje skupova referenci i njihovog redoslijeda izvršavanja u procesu obrade cvora
- definiranje i zapisivanje veza izmedu cvorova
- definiranje pravila odlucivanja o redoslijedu izvršavanja cvorova

Proces kreiranja plana trebalo bi koncipirati tako da konstruktor može "preskakati", odnosno naizmjence koristiti operacije, u onom redoslijedu kako mu u odredenoj situaciji odgovara, a da se u svakom trenutku može vratiti na prethodne korake i izvršiti preinake.
Detaljna razrada i implementacija postupka kreiranja plana, uz sve prije navedene zahtjeve i funkcije, izvan je opsega ovog rada i može biti tema posebnog istraživanja.

7.3.5.1 Uzorci planova

Za proces kreiranja plana važno je i koncipirati organizaciju spremanja kreiranih planova, kao i koncipirati proces kreiranja na temelju uzoraka djelomicno dovršenih planova.
Ako se svi objekti svih planova spreme u istu bazu, jednostavnije bi bilo riješiti upotrebu istih elemenata u više razlicitih planova. U takvoj koncepciji, u bazi bi bilo spremljeno više instanci matrica prikaza veza izmedu cvorova koje bi pripadale pojedinim planovima. Medutim, ovdje se javlja problem inicijalizacije vrijednosti atributa, jer objekt koji može biti rerefenciran u više planova, ne mora imati iste vrijednosti atributa u tim planovima.
Stoga je kao pogodnija odabrana koncepcija u kojoj se svaki kreirani plan sprema u posebnu objektnu bazu.
Planovi konstruiranja za razlicite izvedbe varijantnih konstrukcija sadržavati ce veliki broj objekata koji su zajednicki za sve izvedbe ili za pojedine varijante izvedbe. Za takve slucajeve mogu se kreirati i spremiti kao posebne baze "uzorci" planova koji sadrže sve zajednicke objekte. Na taj nacin proces kreiranja treba "poludovršeni" plan samo razraditi do kraja - tj. inicijalizirati atribute, neke od referenci i eventualno dodati ili obrisati neke

objekte i reference. Prvi korak pri kreiranju novog plana bilo bi stvaranje nove kopije uzorka plana, da bi "original" uzorka ostao nepromijenjen. Prema predloženoj koncepciji mogu se razraditi uzorci planova za skupove varijanti konstrukcije, ali i za razlicite situacije i metode konstruiranja.

7.3.6 Izvodenje plana konstruiranja

Razmatrati ce se proces izvodenja plana pod pretpostavkom izvodenja samo jednog cvora istovremeno. Novokreirani plan spremljen je kao posebna objektna baza koja se temelji na definicijama klasa u "rjecniku baze". Operacija izvodenja plana treba otvoriti bazu plana, ucitati pocetni cvor iz baze u memoriju i pokrenuti proces obrade pocetnog cvora. Svaki cvor na završetku svog procesa obrade poziva proces obrade slijedeceg cvora. Proces obrade može se prekidati interakcijama sa konstruktorom, nakon cega on odreduje tocku nastavka procesa. U takvim slucajevima konstruktor može odrediti redoslijed izvršavanja cvorova koji nije predviden u planu, tj. zapisan u matrici veza izmedu cvorova. Sa stanovišta procesa izvodenja plana treba razlikovati tri podklase cvorova:

- pocetni cvor (svaki plan ima samo jedan pocetni cvor)
- meducvor - po završetku svoje obrade bira slijedeci cvor
- krajnji cvor - njegov završetak obrade ujedno je i kraj procesa izvodenja plana

Svaki plan P mora sadržavati jedan i samo jedan pocetni cvor P i najmanje jedan krajnji cvor K, uz proizvoljni broj meducvorova M:

$$P = \{ P, \; \mathring{a} M_i , \; \mathring{a} K_j \} \qquad i \hat{I} \{0, \ldots n\} ; \qquad j \hat{I} \{1, \ldots m\}$$

Iterativni procesi modeliraju se u predloženoj koncepciji plana povratnim izvršnim vezama (slika 45). To znaci da se može ponavljati izvodenje odredenih nizova cvorova, analogno pojmu "petlje" u programskim jezicima. U tijeku izvodenja iterativnih procesa mogu se promijeniti pocetni uvjeti ili mogu posati raspoložive informacije koje to prije nisu bile. Primarno za takve slucajeve predvidene su mogucnosti doradivanja plana u tijeku njegovog izvodenja, prema razmatranjima u poglavlju 7.3.3. Procesi iteracije mogu biti unaprijed planirani, ali mogu biti i neplanirani - uzrokovani i greškama i propustima konstruktora ili dodatnim ili promijenjenim zahtjevima narucitelja proizvoda (slika 24, poglavlje 6).

Operacija procesa izvodenja plana trebala bi biti koncipirana tako da omoguci prekid izvodenja plana, i nakon toga nastavak na istoj tocci i sa istim stanjem plana kao da prekida nije niti bilo. Koncepcija plana koji je u cijelosti spremljen u objektnoj bazi omogucuje jednostavnu realizaciju takvih zahtjeva.
Citav tijek procesa obrade svakog cvora i interakcije sa konstruktorom bilježe se u "zapis tijeka izvodenja procesa" koji se može koncipirati kao ASCII datoteka.

Plan konstruiranja, nakon što se izvede, zajedno sa svim svojim referenciranim objektima, zapravo cini organizirani skup zapisa informacija o proizvodu.
Stoga bi trebalo razlikovati dva stanja odredenog plana: "neizvedeno" - inicijalno stanje, i "izvedeno" - završno stanje. Za svaki pojedini plan trebalo bi sacuvati "neizvedeno" inicijalno stanje kao posebnu bazu, a pored toga mogu se arhivirati i sva izvedena stanja plana. Stoga prije izvodenja novokreiranog plana treba prvo arhivirati kopiju inicijalnog stanja. Uz svako izvedeno stanje plana takoder treba arhivirati i datoteku sa zapisom tijeka izvodenja procesa.

Jedno okruženje izvodenja procesa konstruiranja (konstrukcijski ured) vjerojatno ce s vremenom kreirati i koristiti dosta razlicitih varijanti planova, pa ce se ovdje predložiti koncepcija organizacije manipuliranja sa zapisima planova.

Da bi se organizirao zapis izvedenih kopija planova, u model procesa konstruiranja uvodi se objekt "plan", koji treba sadržavati listu svih svojih izvodenja, tj. popis naziva i putanja do datoteka s objektnim bazama koje sadrže zapise izvedenih varijanti plana.

Objekt "plan" sadrži slijedece atribute:

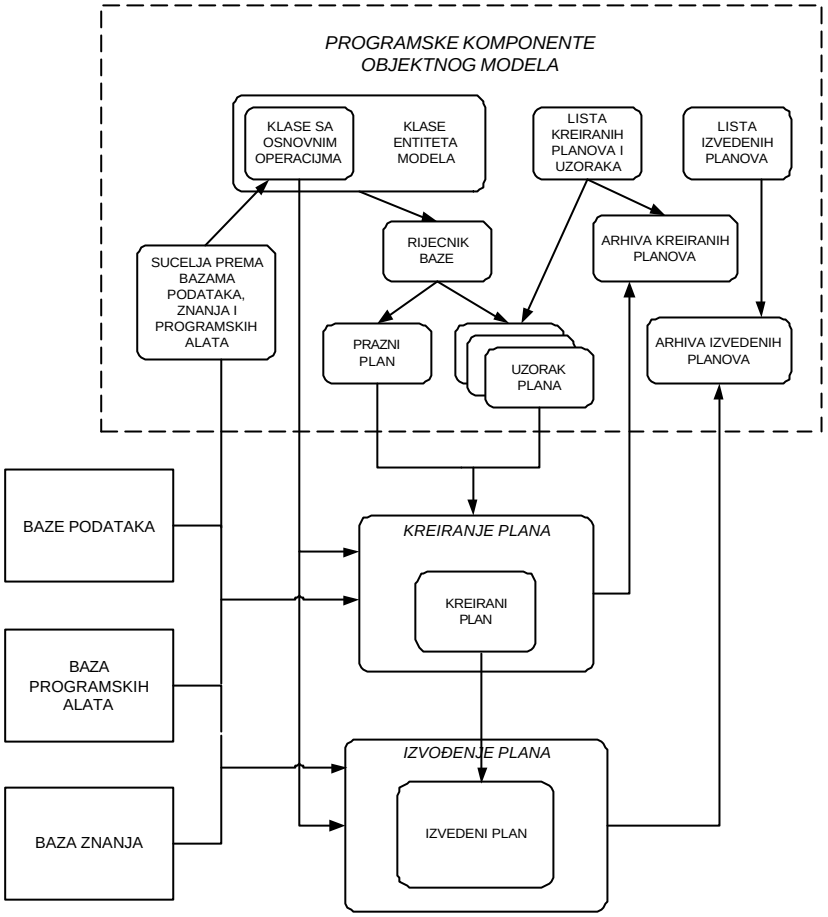
- naziv plana, opis plana
- referencu na konstruktora koji je kreirao plan i koji je za njega odgovoran
- putanju do datoteke koja sadrži rjecnik baze
- putanju do datoteke koja sadrži inicijalno stanje plana
- popis izvodenja kopija inicijalnog stanja plana koji sadrži slijedece elemente:
  - referencu na konstruktora koji je "izveo" kopiju plana
  - naziv datoteke sa zapisom izvedene kopije plana i naziv datoteke sa odgovarajucim zapisom tijeka izvodenja plana

Skup operacija izvodenja plana, kao i operacije kreiranja plana mogu se implementirati kao jedna "servisna klasa" koja nema instance i ne sprema se u objektnu bazu. Operacije izvodenja plana trebaju ukljuciti i sucelja prikaza plana. U tijeku izvodenja plana takva sucelja prikazuju stanja bitnih elemenata odvijanja procesa. Sucelja prikaza plana trebaju u trenucima prekida procesa omoguciti konstruktoru pregled i mijenjanje bilo kojeg elementa plana.

7.4 Komponente realizacije sustava racunalne podrške konstruiranju

Predloženi model racunalne podrške procesu konstruiranja temelji se na objektno orijentiranom pristupu modeliranju i razvoju programskih sustava. Jezgru takvog sustava cini skup klasa koje modeliraju entitete definirane u sedmom poglavlju rada. Operacije kreiranja i izvodenja plana realiziraju se kao "servisne" klase, dok sve ostale klase cine "rjecnik" objektne baze na temelju kojeg se kreiraju baze sa "strukturalnim kosturom" plana procesa konstruiranja.

Koncepcija predloženog objektnog modela razvijana je s ciljem da se implementira kao nadogradnja, a ne kao zamjena postojece strukture racunalne podrške. Stoga ce se u ovom poglavlju ukratko razmotriti odnos programskih komponenti objektnog modela procesa konstruiranja i ostalih programskih alata racunalne podrške konstruiranju i procesu proizvodnje. Pri tome se pretpostavlja situacija implementacije predloženog modela u okruženje koje vec koristi sve oblike racunalne podrške konstruiranju.



Slika 47: Komponente racunalne podrške u procesima kreiranja i izvodenja plana konstruiranja

U ovom poglavlju objektni model procesa konstruiranja promatra se kao dio cjelokupnog sustava racunalne podrške procesu konstruiranja (slika 47). Pri ovom razmatranju izdvojiti ce se baze podataka i baze znanja, a ostale komponente sustava racunalne podrške promatrati ce se kao skup programskih alata.

Proces obrade cvora plana konstruiranja koristi i poziva "vanjske" programske alate preko instanci klase "sucelja programskog alata". U procesu kreiranja i izvodenja plana konstruktor mora imati na raspolaganju organizirane zapise znanja o nacinu korištenja i implementiranim metodama programskih alata, pogotovo za alate specificne domene. Stoga se predlaže koncepcija "baze programskih alata".

Isto tako konstruktor pri kreiranju i izvodenju plana treba imati i sucelja prema bazama znanja i bazama podataka. Korištenje takvih sucelja treba implementirati u operacije kreiranja i izvodenja plana (slika 47).

7.4.1 Programske komponente objektnog modela procesa konstruiranja

Na slici 47 prikazane su osnovne komponente realizacije objektnog modela procesa konstruiranja.

Skup svih klasa modela procesa razmatranih u poglavljima 7.1, 7.2 i 7.3 temelj je za kreiranje rjecnika objektne baze plana konstruiranja. Sve spomenute klase trebaju biti dio aplikacije cijim preprocesiranjem nastaje rjecnik baze. Operacije kreiranja i izvodenja plana

zasebne su aplikacije, ali i one trebaju sadržavati sve klase objektnog modela, pa su dio osnovnog skupa klasa, ali ne i rjecnika objektne baze. Operacije kreiranja i izvodenja plana trebaju kreirati instance svih drugih klasa, spremati ih u bazu i ucitavati iz baze.
Na temelju rjecnika baze kreira se objektna baza koja sadrži samo definicije klasa, bez instanci, odnosno objekata. Takva baza (prazni plan) podloga je za kreiranje "uzorka plana" - djelomicno dovršenog plana koji sadrži sve što je zajednicko svim izvedbama varijantnih konstrukcija. Plan konstruiranja kreira se na kopiji "praznog plana" ili na kopiji "uzorka plana". Prije izvodenja kreirani plan arhivira se za slijedecu upotrebu. Nakon procesa izvodenja plan konstruiranja sadrži sve vrijednosti parametara konstrukcije i skupove referenci na zapise informacija o konstrukciji. Izvedeni plan zapravo je zapis konstrukcije i tijeka izvodenja procesa konstruiranja i kao takav se ne može ponovo izvesti, nego se arhivira. Operacije kreiranja i izvodenja plana koriste sucelja prema bazama podataka, bazi znanja i bazi programskih alata.
Sve navedene komponente trebale bi biti kontrolirane jednom "glavnom" aplikacijom koja upravlja sa više objektnih baza. Arhiva kreiranih planova i arhiva izvedenih planova mogu se realizirati kao posebne aplikacije ciji je zadatak vodenje evidencije o datotekama sa zapisima planova i implementacija sucelja za pretraživanja takvih skupova baza. Spomenute arhive mogu se realizirati kao objektne baze u kojima su spremljene instance objekta "plan". Atributi objekta "plan" navedeni su u poglavlju 7.3.6.

7.4.2 Baze znanja i baze podataka

U procesu konstruiranja koristi se gotovo cjelokupno znanje kojim raspolaže okruženje u kojem se proces konstruiranja odvija (proizvodno poduzece ili projektni ured). To je samo jedan od razloga da koncipiranje baze znanja predstavlja vrlo zahtjevan zadatak (i kvalitativno i kvantitativno). Istraživanjima nacina zapisa i eksploatacije znanja, i primjeni metoda umjetne inteligencije u CAE sustavima posveceno je više pažnje nego racunalnoj podršci vodenju i odvijanju samog konstrukcijskog procesa.

U ovom radu predlaže se koncipiranje baze znanja kao sustava “otvorene kutije s alatima”. Drugim rijecima to bi trebao biti skup raznorodnih metoda, implementiranih raznorodnim programskim alatima koji bi bili povezani zajednickom kontrolnom strukturom i mehanizmom izmjene informacija. Pri tome bi se za odredene vrste znanja i zahtjeva na prikaz znanja koristile za njih pogodne metode ili kombinacije metoda. Mehanizam povezivanja raznorodnih alata, odnosno dijelova baze znanja mogao bi se ostvariti u obliku “meta - baze” [140]. Takva “meta-baza” sadržavala bi znanje o svim pojedinim dijelovima baze znanja, odnosno o vrstama i domenama znanja, te implementiranim metodama prikaza i pristupa znanju, tj. nacinima korištenja znanja. Domene znanja mogu se razvrstati na proizvode i njihove dijelove, parametre sredstava i okruženja za proizvodnju, tehnološke postupke, metode projektiranja i proracuna, propise i preporuke, standarde, kao i cjelokupnu dokumentaciju, odnosno nosioce zapisa informacija koji kolaju u proizvodnom sustavu. Kao dio “meta - baze” moglo bi se realizirati i sucelje povezivanja sa komponentama objektnog modela procesa konstruiranja.
Prijedlog strukturiranja baze znanja kao dijela sustava racunalne podrške procesu konstruiranja razvijen je u [135].
Povezivanje procesa kreiranja plana s bazom znanja trebalo bi realizirati na takav nacin da konstruktor u svakom trenutku procesa kreiranja plana može pretraživati bazu znanja (bez prekida procesa kreiranja) i eventualno i prenositi podatke iz baze znanja. U situacijama

kad proces obrade plana zatraži intervenciju od konstruktora, takoder se treba omoguciti pristup bazi znanja.

Operacije procesa kreiranja i izvodenja plana konstruiranja trebaju takoder omoguciti i pristupanje bazama podataka. U procesu konstruiranja koriste se razlicite baze podataka:
katalozi standardnih dijelova, EDM / PDM sustavi, a konstruktoru mogu trebati i neki podaci iz poslovnih sustava. U objektnom modelu procesa konstruiranja trebalo bi stoga implementirati zajednicko sucelje za pozive i pristup pojedinim bazama podataka.

7.4.3 Baza programskih alata

"Programskim alatima" smatraju se svi oblici racunalne podrške koja se koristi u procesu konstruiranja, a nisu implementirani kao komponente objektnog modela procesa konstruiranja Najcešce ce to biti razni numericki proracuni pisani u proceduralnim jezicima, no može biti i npr. tablicni kalkulator, ekspertni sustav, baza podataka, itd. Pri tome se mogu koristiti i vlastite razvijene aplikacije (za odredenu usku domenu) kao i komercijalne aplikacije opce namjene. Predloženi model procesa konstruiranja integrira takve raznorodne oblike racunalne podrške korištenjem sucelja programskih alata. Sucelja moraju poznavati strukturu ulazno/izlaznih podataka i nacine dobave podataka programskog alata kojem pripadaju. U procesu kreiranja i izvodenja plana konstruktor mora imati na raspolaganju pregled mogucnosti i implementiranih metoda pojedinih programskih alata. Pogotovo to vrijedi za domenom zavisne alate, gdje ce npr. jedan proracun cesto imati i nekoliko razlicitih verzija. Stoga se ovdje predlaže "baza vanjskih programskih alata" kao pomocna komponenta kreiranja i izvodenja plana procesa konstruiranja. Takva baza sadržavala bi kategorizaciju i detaljne opise svakog programskog alata. Skup zapisa znanja o programskim alatima može se realizirati kao objektna ili kao relacijska baza podataka. Primjer prototipa takve baze realiziran je u [5].

8. Realizacija predloženog objektnog modela

Prethodna (sedma) glava rada opisuje konceptualni i informacijski model procesa konstruiranja, prema slici 1 i slici 22. U ovoj glavi razraditi ce se prijedlog preslikavanja (implementacije) entiteta fenomenološkog modela u racunalni model, tj. u programski sustav implementiran u objektno orijentiranom jeziku. UML jezik odabran je za specificiranje, dokumentiranje i vizualizaciju elemenata i strukture predloženog objektnog programskog sustava. Odredeni aspekti implementacije predloženih entiteta vec su razmatrani u prethodnom poglavlju - npr. koncepcija zapisa plana konstruiranja kao skupa objekata spremljenih u objektnu bazu. Mogucnosti modeliranja relacija u objektnoj bazi diktiraju smjernice modeliranja nekih od entiteta predloženih u sedmom poglavlju rada. Stoga ce se na pocetku dati primjer jednostavne objektne baze koja ilustrira razlicite nacine modeliranja relacija. Primjer je dan kao uvod u predloženu koncepciju implementacije svih do sada razmatranih relacija izmedu skupova objekata istih ili razlicitih klasa. Relacije modelirane kao skupovi referenci jednog objekta, ili skupovi objekata iste klase koji sadrže skupove referenci, cine osnovu koncepcije realizacije predloženog objektnog modela.
Realizacijom takve koncepcije potvrduje se teza "da je objektnim metodama moguce efikasno modelirati mrežnu topologiju procesa konstruiranja". U ovom poglavlju biti ce pokazano da je u objektnoj bazi moguce modelirati i zapisati kompleksnu mrežu relacija izmedu entiteta modela, prema razmatranjima u poglavlju 7.2.2, tablica 3. Modelirana mreža relacija može biti vrlo složena, sa više razina referenciranja. Objektna baza posjeduje efikasne mehanizme manipulacije sa velikim brojem objekata i njihovih referenci pri ucitavanju, spremanju i brisanju objekata iz baze.

8.1 Modeliranje relacija

Objektna baza ne sprema samo objekte, nego i sve relacije izmedu objekata koje su definirane u aplikaciji. Kad se spremljeni objekt ucita, sve njegove reference se reaktiviraju, tj. ucitavaju se i svi referencirani objekti i podaci, a pokazivaci se inicijaliziraju na prave memorijske adrese. Definiciji klase ciji se objekti žele spremiti u bazu, treba prethoditi kljucna rijec "persistent". Ovdje ce se dati kratki primjeri mogucih nacina definiranja i spremanja relacija izmedu objekata u objektnoj bazi "POET". Postupkom "preprocesiranja", programski kod svih klasa definiranih kao stalne ("persistent") prevodi se u tzv. "rjecnik baze" (database dictionary). Rjecnik baze posebna je datoteka koja sadrži definicije svih klasa i njihovih relacija. Jedan rjecnik može biti osnova za više baza koje imaju istu strukturu, ali razlicite podatke.
Relacije se mogu se kategorizirati prema nacinu implementacije:

- pokazivac ("pointer") na jedan objekt
- skup pokazivaca na niz objekata ("set")
- reference prema zahtjevu ("on demand references")
- ugradeni (sadržani) objekti, ("embedded objects")

8.1.1 Pokazivaci na objekte

Referenciranje objekta korištenjem pokazivaca (pointera) u C++ jeziku, zapravo je modeliranje relacije. U objektnoj bazi takva relacija se može spremiti, ali i referencirani objekt mora takoder biti oznacen kao "persistent". Na ovaj nacin mogu se modelirati relacije "jedan prema jedan". Pokazivac može referencirati objekt iste klase ili neke druge klase.

```
persistent class Osoba
{
      . . . .
      Osoba  * VaznaOsoba;
      // referencirani objekt je takoder instanca klase "Osoba"
};
```

8.1.2 Ugradeni objekti

Ugradeni objekti ("embedded objects") spremljeni su direktno unutar stalnog objekta. Ugradeni objekt nema identiteta, pa ne može postojati (biti spremljen) kao takav u objektnoj bazi. Ugradeni objekti mogu biti stalni i tranzijentni, u oba slucaja spremaju se unutar objekta koji ih sadrži, bez objektnog identiteta.

```
class Adresa      //ovo nije stalna ("persistent") klasa
{
      . . . .
};

persistent class Osoba
{
      PtString          Ime;
      PtString          Prezime;
      Adresa            Domicil;    // ugradeni objekt
      . . . .
};
```

8.1.3 Skup

Pri definiranju modela baze podataka relacije "jedan prema više" mogu se modelirati pomocu skupova. Objekti u skupu mogu biti pokazivaci na stalne objekte, "on demand" reference, tranzijentni ("non-persistent") objekti ili literali osnovnih tipova podataka jezika C++. POET sadrži tri nacina implementacije skupova koji na razlicite nacine upravljaju memorijom.
Skupovi ne smiju sadržavati stalne (persistent) klase direktno jer bi to moglo rezultirati višestrukim kopijama istog objekta u memoriji, što narušava identitet objekta. Isto tako skup ne smije sadržavati pokazivace na osnovne tipove podataka i pokazivace na tranzijentne objekte.

```
persistent class Osoba
{
      PtString    Ime;
      PtString    Prezime;
      lset<Dijete*>     Djeca;  //skup pokazivaca na instance klase
                                //"Dijete"
};
```

8.1.4 Reference prema zahtjevu

POET objektna baza rješava reference na stalne objekte ucitavanjem referenciranog objekta u memoriju i smještanjem njegove memorijske adrese u pokazivac (pointer). Na taj nacin uvelike se pojednostavljuje programiranje translatiranjem referenci iz baze direktno u C++ reference. Medutim takav pristup može zauzeti znacajne kolicine memorije ako objekti sadrže mnogo referenci, a može i usporiti proces kada treba ucitati sve referencirane objekte. Kljucna rijec "ondemand" deklarira reference na stalne objekte koje ne treba automatski ucitavati, nego se ucitavaju po potrebi, tek na poseban zahtjev aplikacije. Na taj nacin moguce je kontrolirati koji ce se referencirani objekti ucitati, i kada. Moguce je deklarirati skupove "ondemand" referenci koji su posebno korisni u situacijama kada skup sadrži veliki broj referenci. Stoga korištenje skupova "on demand" referenci omogucuje modeliranje i manipuliranje velikim i kompleksnim mrežama relacija, kakve postoje izmedu objekata modela procesa konstruiranja.
Ovakvo svojstvo POET objektne baze važno je za dokazivanje hipoteze rada - da je u objektnoj bazi moguce modelirati mrežnu topologiju procesa konstruiranja.

```
persistent class Prikaz proizvoda
{
      PtString          Naziv;
      PtString          Identifikator;
      . . . .

      ondemand<Sucelje programskog alata>       Proracun vratila;
                        // referenca "prema zahtjevu" na objekt druge klase
      cset<ondemand<Parametar>>     Parametri;
                    // skup "on demand" referenci na pripadajuce parametre
};
```

Referenca "na zahtjev" je referenca na stalni objekt, ona upotrebljava identitet objekta da uspostavi referencu (relaciju). Stoga ova vrsta reference može pokazivati samo na stalne objekte.

8.1.5 Zavisni objekti

Zavisni objekti (dependent objects) omogucuju kreiranje relacija izmedu objekata uz referencijalni integritet. Zavisni objekt ne može postojati bez objekta koji ga posjeduje. Kada se objekt "vlasnik" briše, briše se takoder i zavisni objekt.

```
persistent class Tvrtka : public Osoba
{
      . . . .
      PtString          Naziv tvrtke;
      depend            cset<Osoba*>      zaposlenici;
                  // ako se obriše "Tvrtka" obrisati ce se i svi
                  // referencirani zaposlenici
};
```

8.1.6 Primjer zapisa razlicitih vrsta relacija u POET bazi

U ovom poglavlju razraden je jednostavni primjer modeliranja relacija izmedu objekata, kao ilustracija prethodno iznesenog, i kao uvod u prikaz implementacije predloženog

objektnog modela procesa konstruiranja u POET objektnoj bazi. U primjeru je definirano pet klasa - "Zadatak", "Konstruktor", “Parametar ", “Sadržaj” i Adresa".
Slike 48, 49, i 50 su razlicite varijante prikaza strukture jedne instance klase "Zadatak" i “Konstruktor”. Manji (unutarnji) prozor prikaz je strukture odabranog objekta iz baze, a veci prozor pregled je svih objekata u bazi. U lijevoj polovici prozora objekta prikaz je strukture relacija odabranog objekta (pokazivaci i ugradeni objekti).
Na slikama je u desnoj polovici manjeg prozora "otvoren" prikaz objekata koji su u relaciji s objektom.

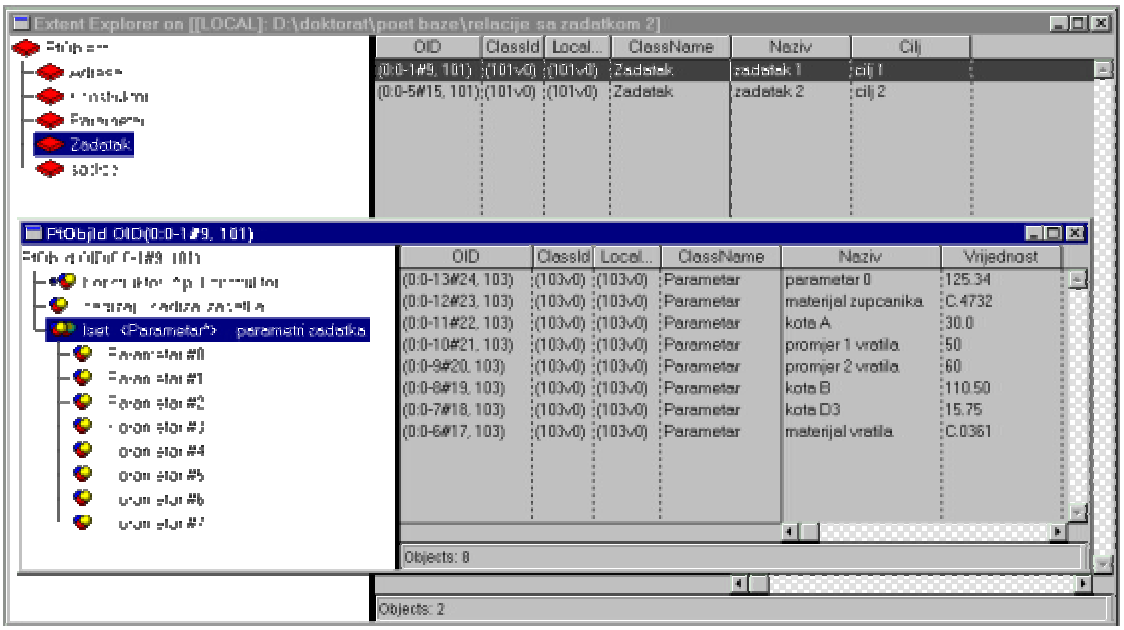
Klasa "Zadatak" sadrži tri vrste relacija:

- Objekt "sadržaj zadatka" je ugradeni objekt klase "Sadržaj" (slika 49).
- Pokazivac (pointer) na objekt klase “Konstruktor”. Na slici 49 "otvoren" je prikaz objekta "konstruktor" ciju adresu sadrži pokazivac. Na ovaj nacin modelirana je relacija "jedan prema jedan".
- Skup (set) pokazivaca na objekte klase "Parametar", koji modelira relaciju "jedan prema više", tj. koji su kljucni parametri zadatka. Na slici 48 "otvoren" je prikaz popisa parametara, odnosno popis instanci na koje pokazuje skup pokazivaca (pointera).

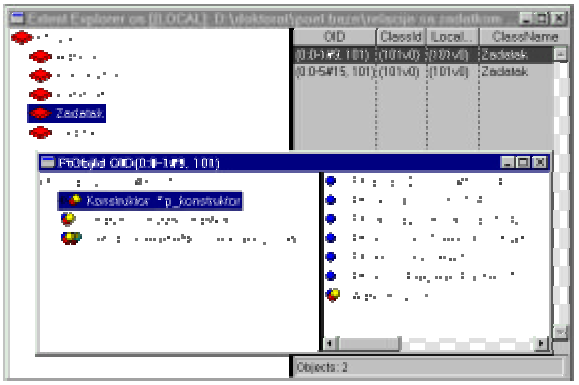
Klasa “Konstruktor” sadrži ugradeni objekt klase “Adresa” (slika 50).

Programski kod primjera:

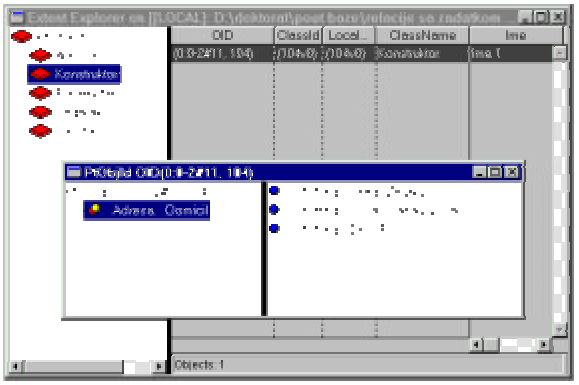
```
persistent class Zadatak
{
      Konstruktor*      p_konstruktor;
      PtString    Naziv;
      PtString    Cilj;
      sadrzaj     sadrzaj_zadatka;
      lset<Parametar*>  parametri_zadatka;
};

persistent class Parametar
{
      PtString    Naziv;
      PtString    Vrijednost;
};

persistent class Konstruktor
{
      PtString    Ime;
      PtString    Prezime;
      Adresa      Domicil;
};

persistent class Adresa
{
      PtString    Grad;
      PtString    Ulica;
      PtString    Broj;
};

persistent class sadrzaj
{
      PtString    opis_sadrzaja;
};
```



Slika 48: Skup referenci na objekte ("lset")



Slika 49: Pokazivac na objekt druge klase



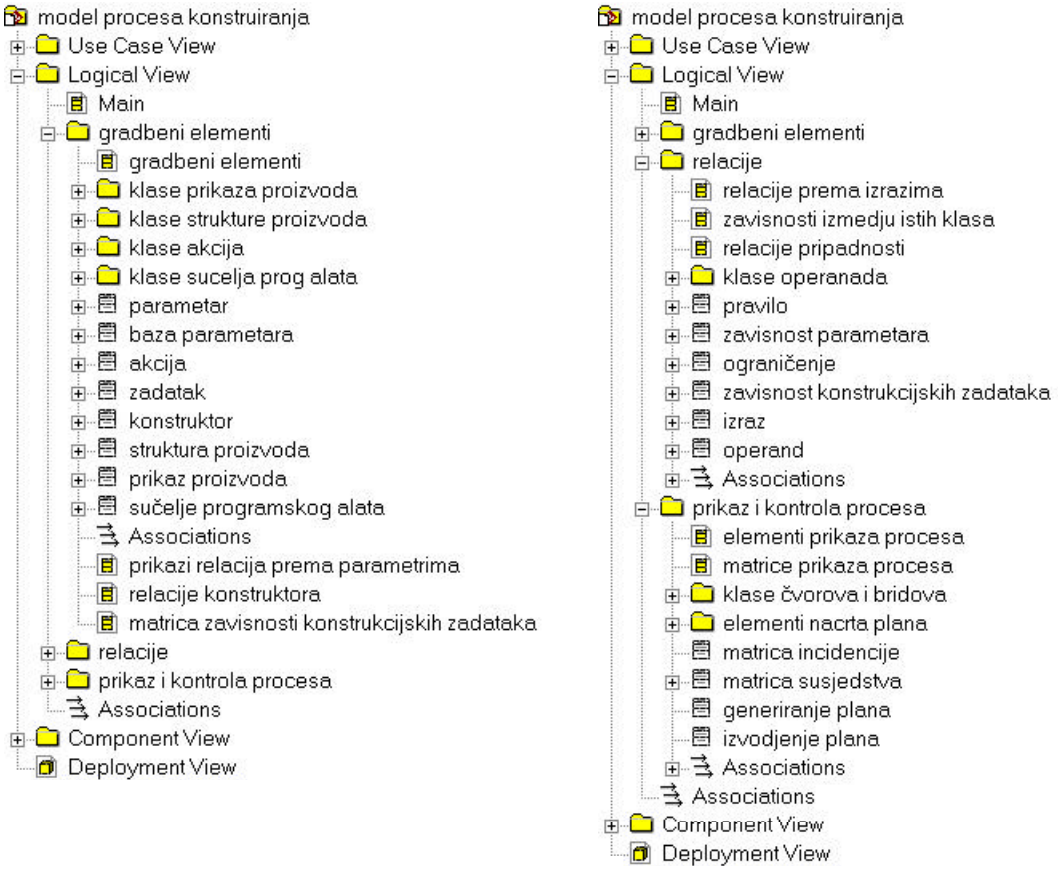
Slika 50: Ugradeni objekt

8.2 Specifikacija predloženog objektnog modela korištenjem UML-a

Premda su dijagrami najuocljiviji dio UML-a, postoji niz informacija koje nastaju u procesu modeliranja, a koje se bolje prikazuju kao tekstualni opis ili tablice. Zadatak je alata za modeliranje da omoguci lak prijelaz izmedu razlicitih vidova opisa modela, uz ocuvanje konzistentnosti cijelog modela. Autori UML-a razvili su i takav alat za modeliranje - "Rational Rose", koji je zapravo svojevrsna baza podataka o modelu - svim definiranim dijagramima, klasama, njihovim relacijama, atributima i operacijama. Spomenuti programski alat, što je još važnije, osigurava konzistentnost i sintakticku ispravnost modela.

Slika 51 u dva dijela prikazuje osnovnu strukturu zapisa definicija klasa i njihovih relacija, kako izgledaju u sucelju programskog paketa "Rational Rose". Na lijevoj polovici slike ekspandirana je lista elemenata u paketu "gradbeni elementi", a u desnoj polovici ekspandirani su paketi "relacije" i paket "prikaz i kontrola procesa". Pojam paket (package)

element je UML-a koji omogucuje kategorizaciju i organizaciju elemenata modela u obliku hijerarhijskog stabla. Paket može sadržavati klase, relacije izmedu klasa i dijagrame. Relacije zavisnosti i pripadnosti izmedu instanci klasa u UML-u modeliraju se kao asocijacije (associations).



Slika 51: Struktura zapisa objektnog modela procesa konstruiranja

Dijagrami klasifikacije prikaza proizvoda, akcija i sucelja programskih alata dani su u sedmoj glavi, pa se ovdje nece posebno navoditi. U nastavku ovog poglavlja nastojati ce se predociti nacini implementacije elemenata (entiteta) modela procesa konstruiranja istim redoslijedom kako su razradeni u sedmoj glavi.

8.2.1 Gradbeni elementi modela procesa konstruiranja

Entiteti razradeni u poglavlju 7.1, preslikani su skup klasa organiziranih u paketu "gradbeni elementi" (lijevi dio slike 51). Relacije izmedu ovih klasa razmatrati ce se izdvojeno, na nekoliko dijagrama koji su dio paketa "relacije".

Klase "akcija", "prikaz proizvoda", "struktura proizvoda" i "sucelje programskog alata" apstraktne su klase. Takve klase služe kao "nosioci" hijerarhije klasifikacije, tj. one nece imati instance. Instance ce imati njihove podklase kao specijalizacije apstraktne klase. Podklase nasljeduju sve atribute i operacije nadredene klase, ali i sve definirane asocijacije (relacije) nadredenih klasa, što je posebno važno za predloženu koncepciju realizacije mreže relacija izmedu klasa u modelu. Klasa "baza parametara" takoder nece imati više instanci, ona se implementira kao "servisna klasa" sa skupom operacija. Operacije "baze

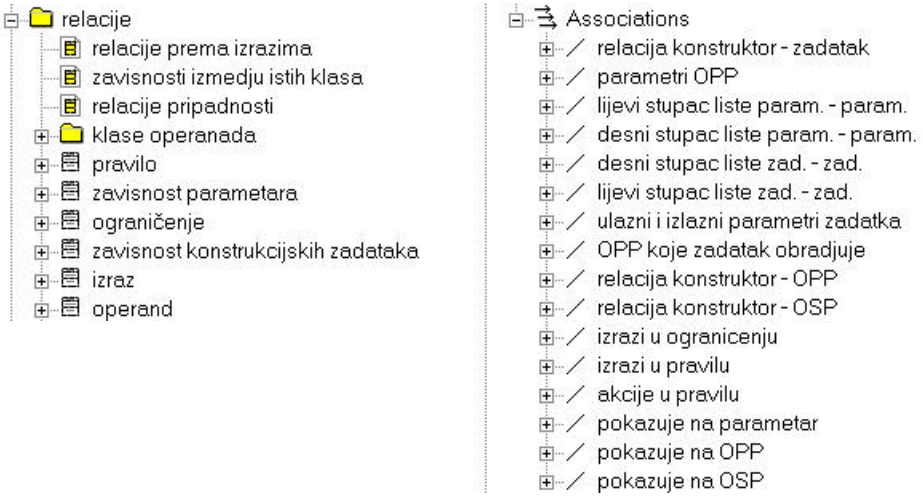
parametara" implementiraju hijerarhijsku klasifikaciju skupa svih instanci parametara korištenjem atributa parametra "naziv grupe". Svaki parametar ima oznaku grupe kojoj pripada (slika 28 u 7.1.2.1), a strukturom grupa upravlja baza parametara. Na ovaj nacin izbjegava se klasifikacija parametara koja bi dodatno zakomplicirala asocijacije (relacije) u kojima sudjeluje parametar, kao i kreiranje novih grupa parametara u tijeku generiranja plana (jer bi trebalo dodati novu podklasu u model baze za svaku novu grupu parametara). Dodavanje nove podklase znaci i definiranje nove varijante rjecnika baze, što znaci i kreiranje plana ispocetka.

8.2.2 Relacije u modelu procesa konstruiranja

Entiteti predloženog modela procesa konstruiranja razradeni u poglavlju 7.2 preslikani su u klase i asocijacije paketa "relacije" (slika 51 i 52).
UML jezik razlikuje cetiri vrste relacija:

- zavisnost - promjena specifikacije jednog elementa utice na drugi element koji ga upotrebljava,
- generalizacija - relacija izmedu klase i podklase koja se iz nje derivira,
- realizacija - semanticki je kombinacija zavisnosti i generalizacije,
- asocijacija - strukturalna relacija koja specificira veze izmedu objekata istih ili razlicitih klasa.

Sve relacije razmatrane u poglavlju 7.2 semanticki odgovaraju asocijacijama.
Pojam asocijacije u UML-u, odgovara pojmu relacije u relacijskim bazama. Jedna klasa tako može imati relacije "jedan prema jedan" i "jedan prema više" sa drugim klasama. Relacija "jedan prema više" znaci da je svaka instanca npr. "klase A" povezana sa više instanci npr. "klase B", tj. svaka instanca "klase A" sadrži skup (polje) pokazivaca na skup instanci "klase B". Asocijacija "jedan prema jedan" modelira se tako da objekt "klase A" sadrži pokazivac (pointer) na adresu objekta "klase B". Prikaz modeliranja i zapisa asocijacija "jedan prema više" i "jedan prema jedan" u objektnoj bazi dan je u poglavlju 8.1. Slika 52 sadrži listu svih asocijacija definiranih u paketu "relacije". Graficki prikazi ovih asocijacija kombinirani su na nekoliko dijagrama cija razmatranja slijede.



Slika 52: Klase i asocijacije u paketu "relacije"

Relacije "pripadnosti" razmatrane u poglavlju 7.2.5 preslikane su u asocijacije "jedan prema više" izmedu dviju razlicitih klasa.

8.2.2.1 Relacije zavisnosti

Relacije zavisnosti razmatrane u 7.2.3 i 7.2.4 preslikane su u klase, cija jedna instanca predstavlja jedan redak matrice prikaza zavisnosti. Na slici 52 to su klase "zavisnost parametara" i "zavisnost konstrukcijskih zadataka". Klasa "zavisnost parametara" povezana je sa dvije asocijacije sa klasom "parametar", i analogno je klasa "zavisnost konstrukcijskih zadataka" povezana sa dvije asocijacije sa klasom "zadatak". Navedene asocijacije nazvane su "lijevi stupac liste" i "desni stupac liste" (slika 52). Predložena koncepcija temelji se na postupku svodenja matrice na listu (slika 39), prikazanom u poglavlju 7.2.1.
Modelirane relacije zavisnosti postavljene su izmedu skupova instanci iste klase (tablica 10). Matricni prikaz takve relacije može se promatrati kao niz redaka u kojem svaki redak predstavlja jednu relaciju "jedan prema više", tj. jedna instanca klase ovisi o skupu drugih instanci klase: $a_i = f(a_j),\ j \in \{1, \ldots, n\},\ j \neq i$.

	a_1	a_2	a_3	a_4	a_5	a_6	….	a_n
a_1				x				
a_2			x			x		
a_3	x			x	x			
a_4		x						
….								
a_n					x	x		

→

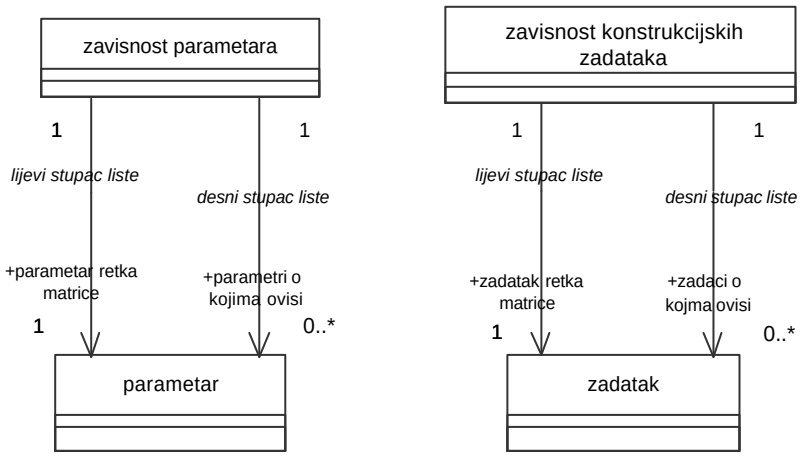
	instance povezane sa instancom retka	semantika relacije
a_1	a_4	$a_1 = (a_4)^2$
a_2	a_3, a_6	$a_2 = a_3 + a_6$
a_3	a_1, a_4, a_5	…..
a_4	a_2	
….		
a_n	a_5, a_6	

Tablica 10: Zapis skupa realcija zavisnosti izmedu instanci iste klase, sveden na listu

Matricni zapis takve relacije može se na predloženi nacin modelirati kao klasa u kojoj svaka instanca klase predstavlja jedan redak matrice, a skup svih instanci klase cini prikaz cijele matrice. Kako se onda modelira jedna instanca takve klase, tj. jedan redak matrice?
Neka je klasa koja modelira matricu "klasa M", a "klasa A" neka je klasa zavisnost cijih instanci je prikazana matricom. Svaka instanca klase M sadrži dvije asocijacije prema klasi A:

- jedan prema jedan - pokazivac na instancu klase A, "ciji je redak " matrice
- jedan prema više - skup pokazivaca na instance klase A o kojima ovisi instanca klase A ciji je to redak

Pored toga, svaka instanca klase "M" može sadržavati i tekstualni zapis semantike relacije. Na slici 53 je UML prikaz razmatranih asocijacija izmedu klasa.



Slika 53: Matrice zavisnosti parametara i zadataka

Objektni model matricnog zapisa relacija zavisnosti unutar skupa svih parametara sastoji se dakle od:

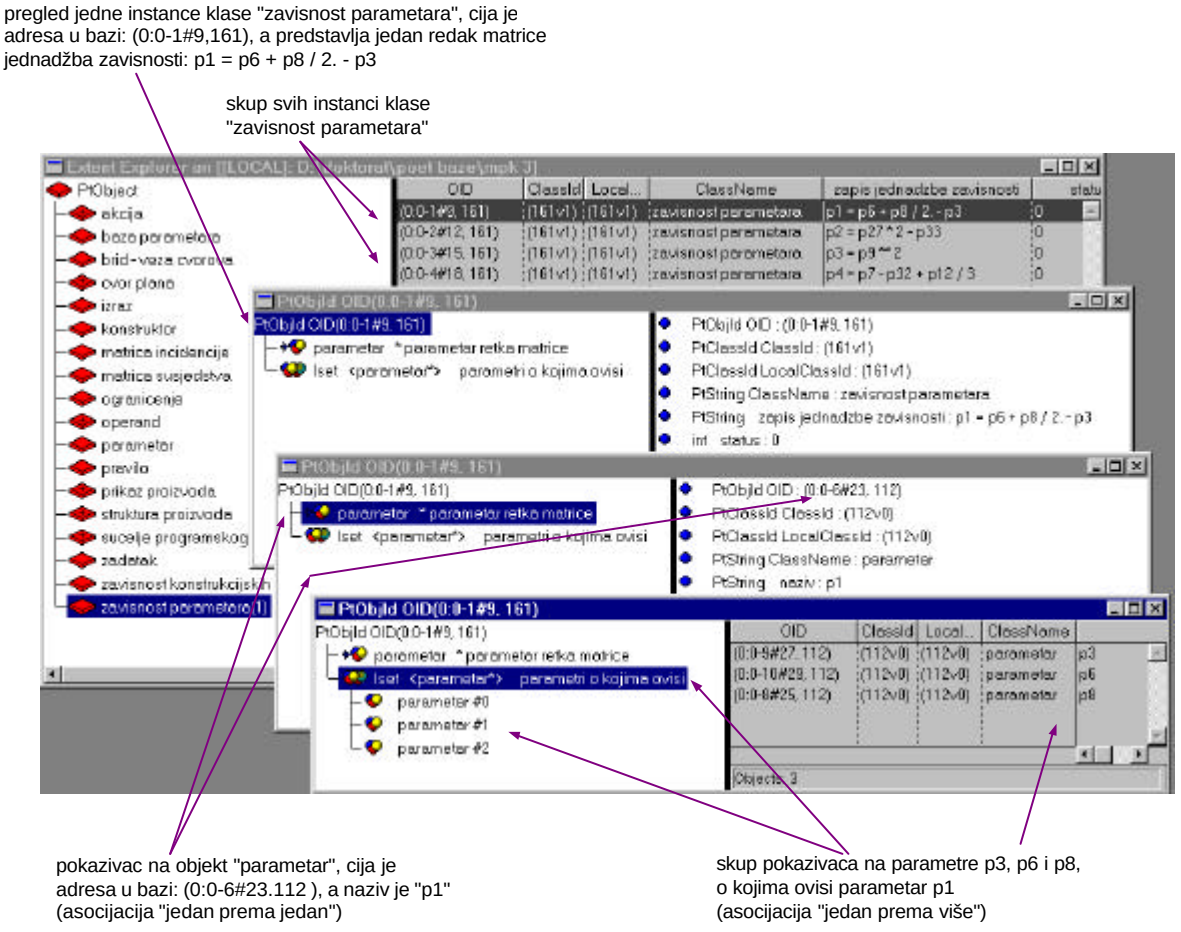
- skupa instanci klase "zavisnost parametara",
- dvije asocijacije izmedu klasa "zavisnost parametara" i "parametar".

Analogno je modeliran i matricni zapis zavisnosti unutar skupa konstrukcijskih zadataka.

Realizacija predložene koncepcije u objektnoj bazi prikazana je na primjeru matrice zavisnosti parametara. Svaka instanca klase "zavisnost parametara" sadrži:

- pokazivac na objekt "parametar" koji je u lijevom stupcu matrice svedene na listu, neka se taj parametar zove p_l
- skup pokazivaca ("lset") na objekte klase "parametar" o kojima ovisi parametar p_l
- tekstualni zapis jednadžbi zavisnosti parametara

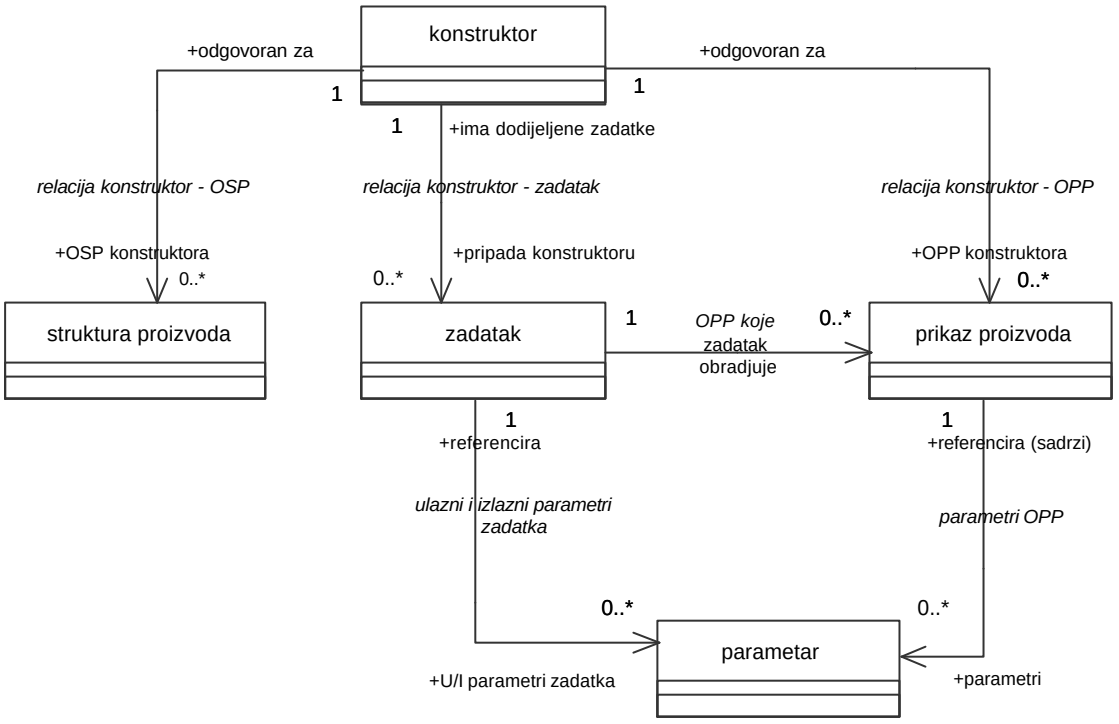
Na slici 54, u tri manja prozora dane su razlicite varijante prikaza jedne instance klase "zavisnost parametara". Prvi manji prozor s gornje strane prikaz je atributa, gdje se vidi zapis jednadžbe zavisnosti parametara. Dalje slijedi prozor u kojem je "otvoren" prikaz parametra na kojeg pokazuje pokazivac - to je parametar "ciji je redak matrice". U trecem prozoru "otvoren" je prikaz skupa pokazivaca na sve parametre o kojima ovisi promatrani parametar (ciji je redak matrice").



Slika 54: Pregled referenci jedne instance klase "zavisnost parametara"

8.2.2.2 Relacije pripadnosti

Relacije "pripadnosti" razmatrane u poglavlju 7.2.5, modelirane su jednostavnije nego relacije zavisnosti, kao asocijacije "jedan prema više" izmedu dviju razlicitih klasa. Ako promatramo relaciju "instance klase B pripadaju instancama klase A", tada svaka instanca klase "A" sadrži skup pokazivaca na instance klase "B" koje joj pripadaju. Pri tome se matricni prikaz relacije svodi na listu, prema tablici 6. Na slici 55 je UML prikaz razmatranih asocijacija izmedu klasa.



Slika 55: Relacije pripadnosti izmedu razlicitih klasa

8.2.3 Izrazi, ogranicenja i pravila odlucivanja

Prema sintaksi izloženoj u 7.2.6, izraz se sastoji od operanada i operatora, gdje su operandi atributi objekata modela procesa konstruiranja ili numericke ili znakovne konstante:

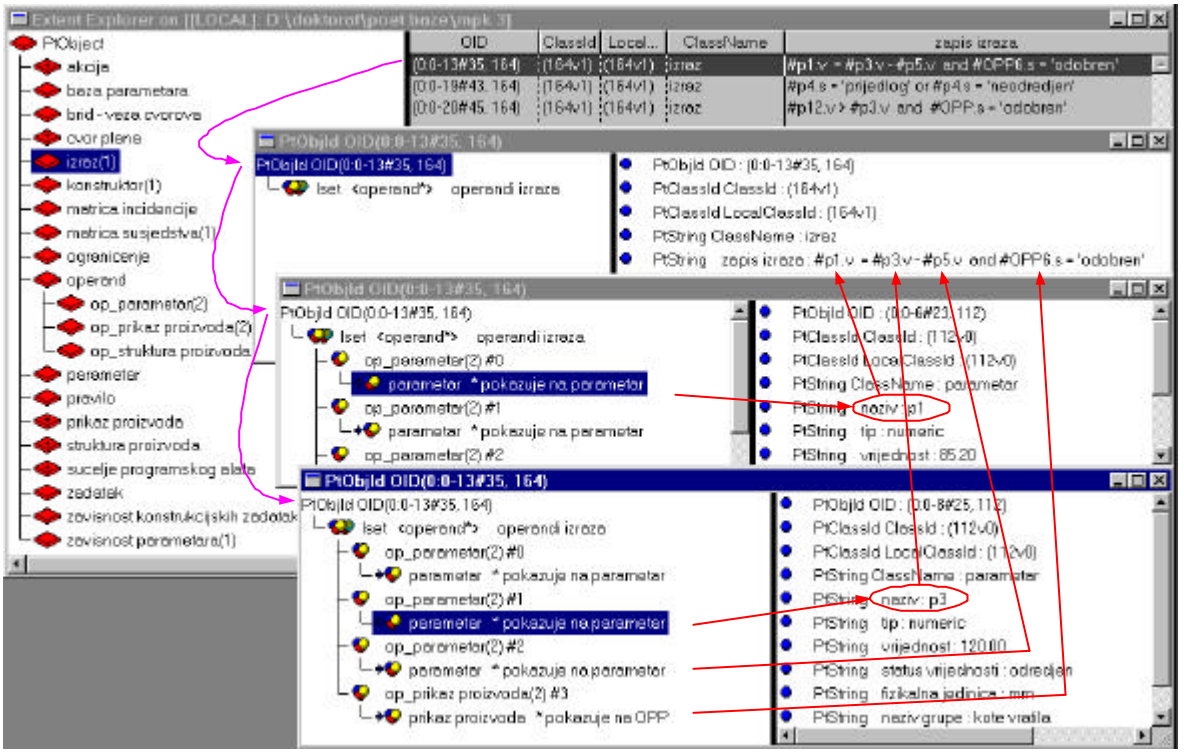
```
izraz :: <operand> | <operator> | <operand> { <operator> | <operand> }
operand ::= <varijabla> | <konstanta>
operator ::= <aritmeticki operator> | <relacijski operator> |
             <logicki operator> | <zagrada>
varijabla ::=  #naziv_objekta.naziv_atributa
```

Osim samog zapisa izraza kao niza znakova, instanca klase "izraz" mora sadržavati i skup pokazivaca (pointera) na objekte ciji atributi se pojavljuju kao varijable izraza. Medutim, opcenito gledano, ti objekti mogu biti razlicitih klasa, koje nisu derivirane iz jedne zajednicke klase. Skup pokazivaca može sadržavati pokazivace samo na objekte jedne klase i njenih podklasa. Dakle, ako bi instanca klase "izraz" imala skup pokazivaca na objekte koji se u izrazu pojavljuju kao varijable, svi ti objekti morali bi biti instance iste klase ili njenih podklasa.

Stoga se ovdje predlaže rješenje u kojem izraz može referencirati i razlicite klase objekata.

Uvodi se klasa "operand", kao apstraktna klasa sa skupom podklasa, gdje svaka podklasa odgovara jednoj od klasa objekata modela procesa konstruiranja. Tako npr. klasi "parametar" odgovara klasa "op_parametar", koja je podklasa klase "operand". Svaka podklasa klase "operand" sadrži pokazivac na objekt odgovarajuce klase. Instanca klase "izraz" sadrži skup pokazivaca na klasu "operand", a time i na njene podklase. Na taj nacin klasa "operand" služi kao "posrednik" koji preko svojih podklasa realizira skup pokazivaca na razlicite klase.

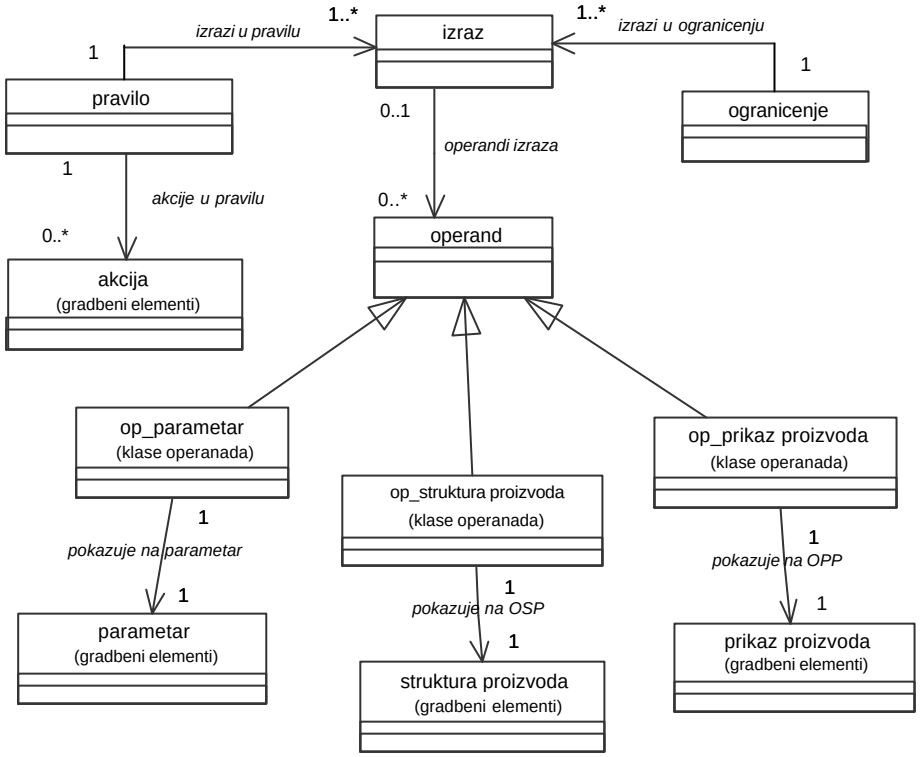
Primjer zapisa referenci izraza u objektnoj bazi dan je na slici 56. Tri manja prozora na razlicite nacine prikazuju elemente zapisa jedne instance klase "izraz". U gornjem manjem prozoru vidi se zapis izraza, a u dva prozora ispod njega "otvoreni" su prikazi svih referenci izraza. Prva razina referenciranja je skup pokazivaca (lset) na instance podklasa klase "operand". U drugoj razini referenciranja, svaka instanca podklasa klase "operand" sadrži pokazivac na objekt ciji je atribut operand u izrazu. Primjer zapisanog izraza sadrži cetiri operanda (p1.v, p3.v, p5.v i OPP6.s), pa analogno instanca klase "izraz" sadrži cetiri pokazivaca na instance klase "operand". Od toga, prva tri pokazuju na instance klase "op_parametar", a cetvrti pokazuje na instancu klase "op_prikaz proizvoda". Obje navedene klase su podklase klase "operand". Instance klase "op_parametar" sadrže pokazivace na objekte klase "parametar" ciji su nazivi "p1", "p3" i "p5". U dva donja prozora "otvoreni" su prikazi pokazivaca na objekte "p1" i "p3". Instanca klase "op_prikaz proizvoda" sadrži pokazivac na objekt klase "prikaz proizvoda", ciji je naziv "OPP6".



Slika 56: Reference izraza na operande

Prema predloženoj koncepciji zapisa referenci izraza, operacija "parsiranja" izraza može dohvatiti sve vrijednosti atributa objekata koji su operandi u izrazu, a ujedno operandi izraza mogu biti atributi objekata razlicitih klasa. Izrazi se referenciraju u ogranicenjima i pravilima odlucivanja. Jedna instanca ogranicenja sadrži jedan izraz, dok jedna instanca pravila odlucivanja može sadržavati više izraza. Pravila odlucivanja osim skupa referenci na izraze sadrže i skup referenci na akcije.

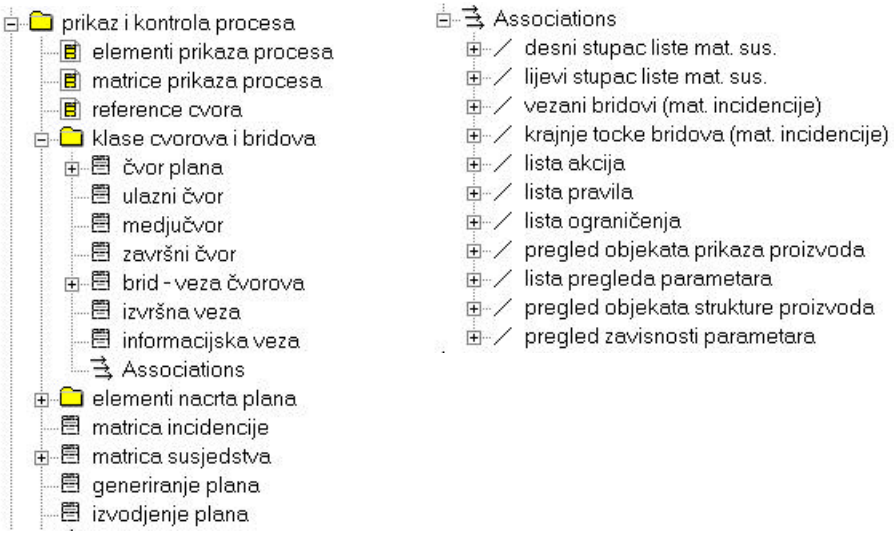
UML dijagram svih razmatranih asocijacija klase "izraz" dan je na slici 57. Pune strelice oznacavaju relacije generalizacije (nasljedivanja), a "obicne" strelice oznacavaju asocijacije (relacije "jedan prema više" ili "jedan prema jedan"). Na slici 57, radi preglednosti prikazane su samo tri podklase klase "operand".



Slika 57: Asocijacije klase "izraz"

8.2.4 Elementi prikaza i kontrole izvodenja procesa konstruiranja

Entiteti predloženog modela procesa konstruiranja razradeni u poglavlju 7.3 preslikani su u klase i asocijacije paketa "prikaz i kontrola procesa" (slika 58). Klase "cvor plana" i "veza cvorova" apstraktne su klase, cije su podklase prikazane na slici 59.

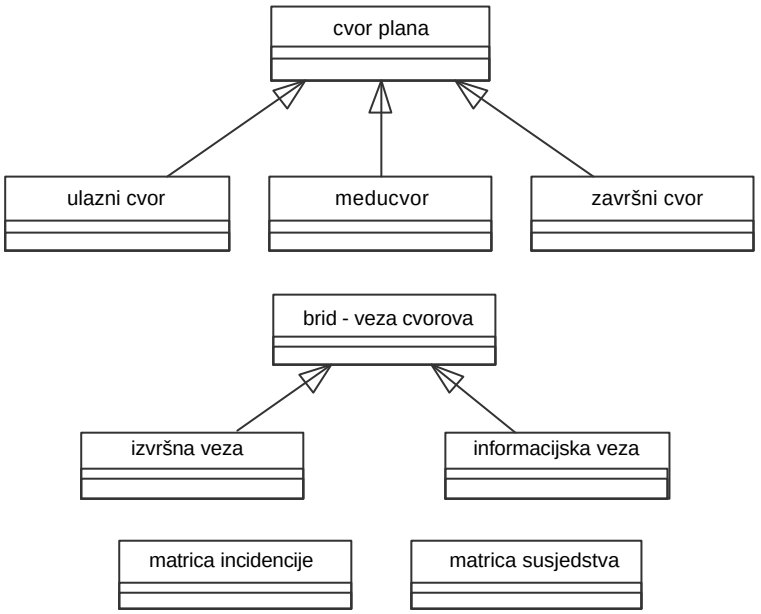


Slika 58: Klase i asocijacije u paketu "prikaz i kontrola procesa"

Matrica susjedstva i matrica incidencije sadrže zapis veza izmedu cvorova plana konstruiranja. Klase "generiranje plana" i "izvodenje plana" sadrže skupove operacija, nemaju više instanci i ne spremaju se u objektnu bazu. Prve cetiri asocijacije u paketu "prikaz i kontrola procesa" realiziraju matrice incidencije i susjedstva, a preostale su reference cvora plana na elemente koje obraduje u tijeku izvodenja.

8.2.4.1 Klase cvorova i veza izmedu njih

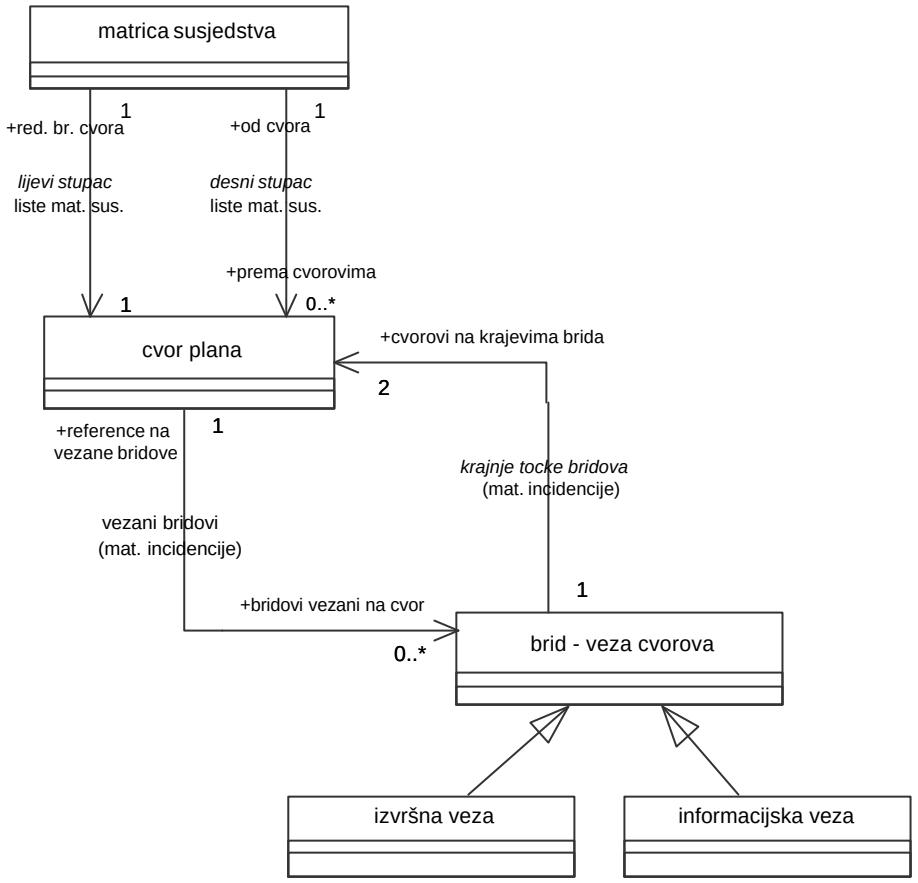
Klasa "cvor plana" ima tri podklase - "ulazni cvor", "meducvor" i "završni cvor", prema razmatranjima u 7.3.6. Sve tri podklase nasljeduju sve skupove referenci od klase "cvor plana", prema slici 59. Razlike izmedu ove tri vrste cvora su u operaciji procesa obrade cvora, npr. "završni cvor" nema odlucivanja o daljnjem tijeku, on prekida proces i zatvara bazu i datoteku sa zapisom tijeka izvodenja.



Slika 59: Klase cvorova i njihovih veza

Klasa "brid - veza cvorova" ima podklase "izvršna veza" i "informacijska veza" prema iznesenom u poglavlju 7.3.3.3. Podklase veze cvorova referenciraju se u zapisu matrice incidencije (slika 60).

Veze izmedu cvorova plana zapisuju se u matricnom obliku, u matricu susjedstva i matricu incidencije, prema prikazanom u 4.4.1. Matrica susjedstva implementirana je na isti nacin kao i matrice prikaza relacija zavisnosti parametara i zavisnosti zadataka. Svaka instanca matrice susjedstva implementira jedan redak matrice. Pri tome svaka instanca ima jedan pokazivac na jedan cvor plana (ciji je to redak matrice) i skup pokazivaca na cvorove koji su s njim povezani. Dakle, klase "matrica susjedstva" i "cvor plana" povezane su asocijacijom "jedan prema jedan" i "jedan prema više" (slika 60).

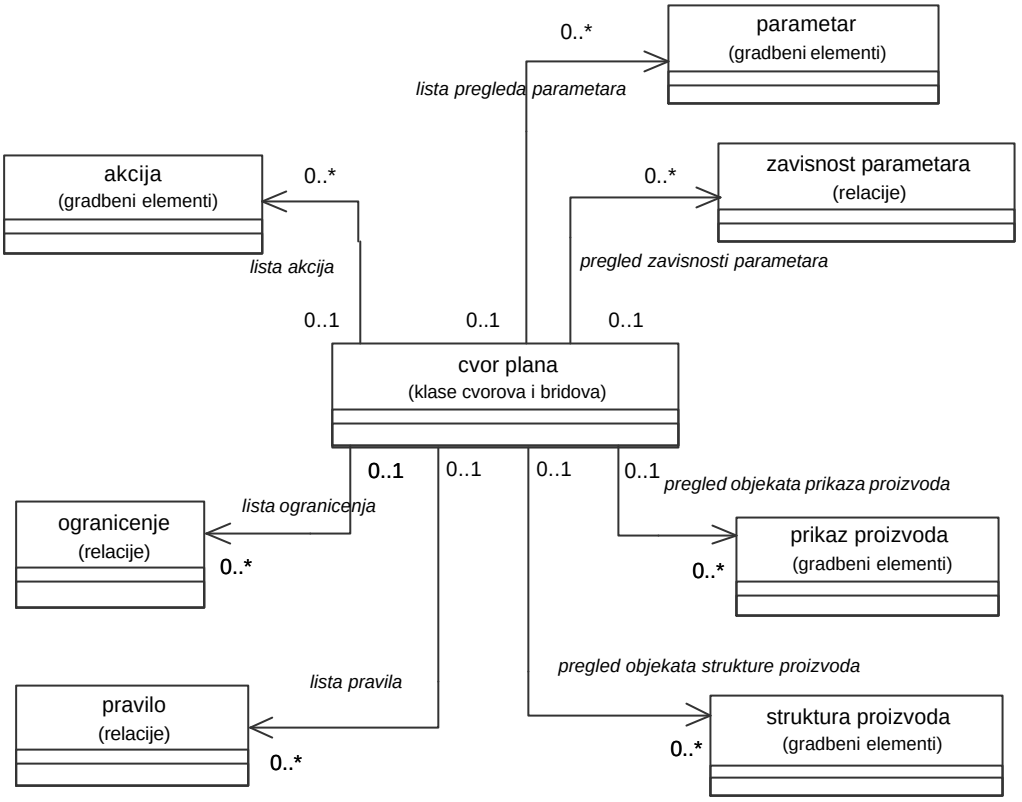


Slika 60: Matrice zapisa veza izmedu cvorova plana

Matrica incidencije modelirana je takoder sa dvije asocijacije. Pri tome se matrica svodi na dvije liste, prema tablici 1. U prvoj listi zapisano je za svaki cvor koji su bridovi vezani na njega. Takva lista modelirana je asocijacijom "jedan prema više" izmedu cvora plana i veza cvorova. Pri tome ce svaka instanca cvora plana sadržavati skup pokazivaca na instance podklasa veze cvorova. U drugoj listi na koju se svodi matrica incidencije zapisano je za svaki brid (vezu cvorova) koje su krajnje tocke (cvorovi) tog brida. Takva lista modelirana je relacijom "jedan prema dva", tj. svaka instanca veze cvorova sadrži polje od dva pokazivaca na instance cvora plana.

8.2.4.2 Reference cvora plana konstruiranja

Cvor plana konstruiranja sadrži skupove referenci na entitete modela procesa konstruiranja koje obraduje. Proces obrade i reference cvora detaljno su razmotreni u poglavljima 7.3.3.1 i 7.3.3.2. Sve reference cvora modelirane su kao asocijacije "jedan prema više", odnosno kao skupovi pokazivaca (pointera). Na slici 61, s lijeve strane su reference na akcije, ogranicenja i pravila, koji su "objekti" procesa obrade cvora, a s desne strane su reference na objekte "pregleda stanja konstrukcije". Reference cvora na njemu susjedne cvorove u planu (matrica susjedstva) i reference na bridove grafa (matrica incidencije), radi preglednosti izdvojene su na slici 60.



Slika 61: Asocijacije (relacije) cvora plana konstruiranja

Specifikacijama i realizacijom predloženog skupa entiteta odredena je osnovna struktura objektnog modela procesa konstruiranja koji zadovoljava zahtjeve i ciljeve postavljene u uvodnom dijelu rada, a potvrdena je i postavljena hipoteza.

Predloženi model orijenitran je primarno na unapredenje organizacije i integracije racunalne podrške. Zamišljeno je da ovako koncipirani model bude jezgra cjelokupnog sustava racunalne podrške konstruiranju, koja bi ispunjavala slijedece zadace:

- unapredenje organizacije procesa, planiranjem redoslijeda akcija i tokova informacija
- unificiranje i standardiziranje nacina zapisivanja informacija o proizvodu u okruženju timskog rada
- zapisivanje tijeka izvodenja procesa, a što je posebno važno, kljucnih odluka i njihovih objašnjenja
- integriranje postojecih raznorodnih alata racunalne podrške

8.3 Primjer plana konstruiranja

U ovom poglavlju razraden je primjer plana procesa konstruiranja na postupku konstrukcije rotora elektromotora srednjih snaga. Radi se o varijantnoj konstrukciji u kojoj se koristi velik broj standardnih dijelova i dijelova kojima se samo prilagodavaju dimenzije. Ovaj primjer odabran je zbog poznavanja problematike iz konkretnog proizvodnog okruženja - autor ovog rada tijekom nekoliko godina razvijao je programsku podršku za konstruiranje navedenog sklopa. Karakteristike elektromotora srednjih snaga iz standardnog programa

cesto ne odgovaraju u potpunosti zahtjevima narucioca, pa se u takvim situacijama razvija nova varijanta izvedbe. Pri tome se polazi od elektromagnetskog projektnog proracuna, kojim se generiraju osnovni parametri nove varijante izvedbe sklopova koji odreduju karakteristike elektromotora. Izlazni podaci elektromagnetskog proracuna osnova su za konstrukciju nove "nestandardne" izvedbe. Takva nova izvedba, mora naravno koristiti standardno kucište, valjne ležaje, ventilator i ostale dijelove koji nisu direktno vezani na postizanje zahtjevanih karakteristika elektromotora. Ostali dijelovi - rotorski i statorski paket limova, bakreni dijelovi, izolacija, vratilo, tlacne ploce i balansni prsten moraju se konstruirati prema podacima elektromagnetskog projektnog proracuna. Ovaj primjer ograniciti ce se samo na konstrukciju "nove varijante" sklopa rotora.

Rotor elektromotora može se dakle promatrati kao parametarska i adaptivna konstrukcija. Parametri koji se najcešce mijenjaju su dužina i promjer paketa limova, a o njima ovise dimenzije vratila, bakrenih štapova u paketu, izolacije i još nekih dijelova.

Kod elektromotora srednjih snaga kritican parametar konstrukcije najcešce je progib vratila ispod rotorskog paketa limova. Progib je to veci, što je veca dužina paketa za odredeni razmak ležaja, odnosno dužinu kucišta. Pri ugradnji rotor nije moguce idealno centrirati u odnosu na stator, tako da uvijek postoji odredeni ugradbeni ekscentricitet. Ekscentricnost uzrokuje nesimetricnost magnetskog polja u toku rada motora, tako da uvijek postoji odredena rezultantna magnetska sila koja privlaci rotor k statoru. U tijeku perioda zaleta motora vrijednosti privlacne magnetske sile su najvece (istog ili višeg reda velicine od težine rotora). Tada može doci do vrlo skupe havarije - struganja rotora po statoru. Isto tako, u normalnom radnom režimu ekscentricitet ne bi trebao biti veci od 20 postotaka zracnog raspora izmedu rotora i statora.

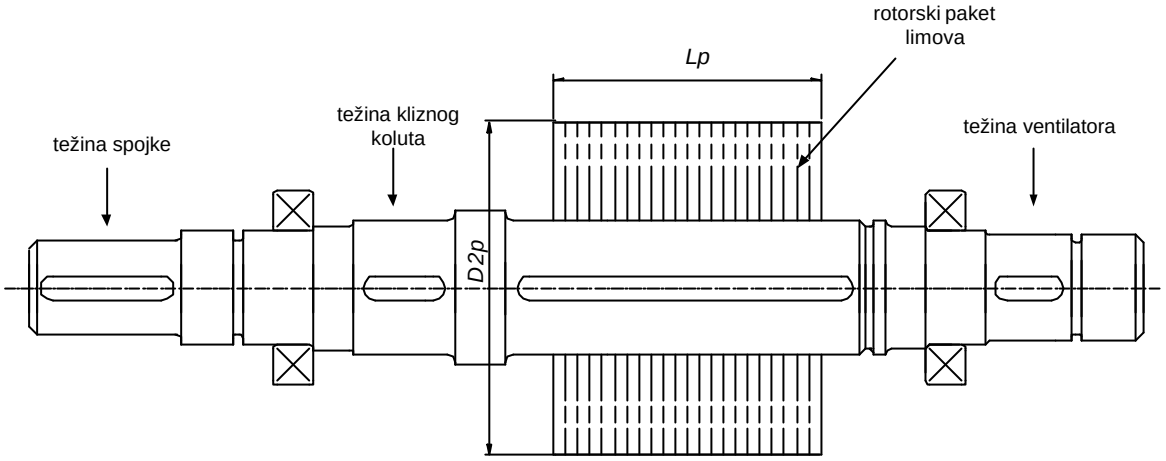
Zbog svih navedenih razloga svakako se mora izvršiti kontrolni proracun za svaku novu varijantu izvedbe sklopa rotora. Ovisno o vrsti elektromotora iz proizvodnog asortimana, izvedbe vratila konstrukcijski su dosta raznorodne. Stoga je za proracun progiba, nagiba, momenta i poprecne sile u presjecima vratila odabrana metoda tzv. prijenosnih matrica koja se može poopciti na bilo kakav oblik i položaj vratila (horizontalno ili vertikalno). Programski alat za navedenu metodu proracuna realiziran je u Fortranu 77.

Medutim primjena navedene metode zahtijeva dosta rutinskog posla na pripremi i upisu ulaznih podataka za proracun. Vratilo treba podijeliti na manje segmente (polja) za koje treba na temelju radionickih crteža izracunati i odrediti u prosjeku oko stotinu podataka, što traje po nekoliko sati. U razmatranom primjeru pretpostaviti cemo da je razvijen poseban programski alat koji automatski priprema sve potrebne ulazne podatke za kontrolni proracun sklopa vratilo-ležaji-rotorski paket. Na temelju analize rezultata kontrolnog proracuna, treba odluciti da li svi parametri konstrukcije nove izvedbe zadovoljavaju predvidena ogranicenja. Osim vec navedene provjere progiba i ekscentriciteta rotora u radu, potrebno je provjeriti naprezanje u kriticnom presjeku rukavca vratila, te vijek trajanja i nagib u ležaju s pogonske strane u slucajevima kada se na rukavac montira remenica. U tom slucaju potrebno je u pripremljeni skup podataka za proracun dodati i podatke o sili zatezanja remena.

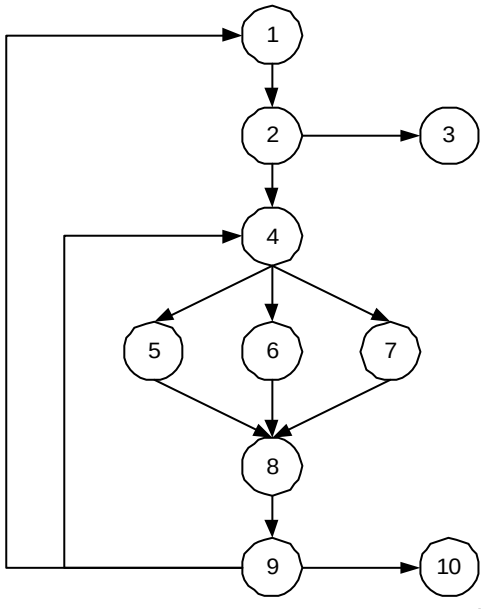
Ako neka od ogranicenja nisu zadovoljena, treba odluciti da li se problem može riješiti promjenama konstrukcije, ili je potrebno promijeniti i osnovne parametre paketa limova, tj. ponoviti elektromagnetski projektni proracun. U oba slucaja potrebno je ponoviti kontrolni proracun sklopa vratilo-ležaji-rotorski paket. Ako svi parametri zadovoljavaju ogranicenja, mogu se razraditi radionicki crteži svih dijelova sklopa rotora.

Opisani se konstrukcijski zadatak cesto (svakodnevno) ponavlja, a sastoji se gotovo samo od rutinskih aktivnosti - dohvacanja i obrade podataka, prilagodnog konstruiranja, te kontrolnog proracuna. Postepenim razvojem programskih alata, baza podataka o dijelovima

sklopa, i parametarskih CAD modela, te njihovim integriranjem može se znacajno skratiti vrijeme potrebno za rješavanje zadatka. Povezivanje programskih alata, te kontrola redoslijeda njihovog izvodenja i toka podataka izmedu njih može se ostvariti korištenjem zamišljenog primjera plana koji ce biti iznesen.
Na slici 62 prikazana je skica parametarske konstrukcije sklopa vratilo-ležaji-rotorski paket limova, asinhronog kliznokolutnog elektromotora.



Slika 62: Skica konstrukcije rotora asinhronog elektromotora



Slika 63: Graficki prikaz razmatranog primjera plana

Tijek opisanog procesa konstruiranja modeliran je planom prikazanim na slici 62. U ovom prikazu navesti ce se namjena i opisati proces obrade svakog cvora. Plan konstruiranja prema predloženoj koncepciji skup je objekata i mreže njihovih referenci zapisanih u objektnoj bazi. Stoga ce se dalje navoditi koji objekti se u odredenom cvoru obraduju.

Cvor 1:

Zadatak zapocinje elektromagnetskim projektnim proracunom, koji odreduje sve potrebne podatke za konstrukciju i kontrolni proracun rotora. Prva akcija cvora poziva program za elektromagnetski projektni proracun. Sucelje ovog programskog alata obavlja transfer izracunatih podataka prema parametrima koji su modelirani kao objekti. U ovom cvoru odreduje se cijeli niz parametara, tzv. "podaci namota i paketa limova". Izdvojiti cemo samo parametre potrebne za definiranje opisane konstrukciju i kontrolni proracun progiba (u tablicnom prikazu plana (na kraju poglavlja) oznaceni su kao "grupa1"):

Lp: dužina rotorskog paketa limova
D2p: vanjski promjer rotorskog paketa,
D1p: unutarnji promjer rotorskog paketa
delta: zracni raspor
B1: indukcija u zracnom rasporu, *P1*: koeficijent polariteta,
Cf: Carterov faktor (za proracun magnetske sile izmedu rotora i statora)
Pn: snaga, *T*: moment
n: broj okretaja,

Objekti prikaza proizvoda koji se obraduju u cvoru 1 su lista zahtjeva narucioca i racunski podaci elektromagnetskog proracuna.

Cvor 2:

U ovom cvoru pokrece se sucelje prema bazi podataka svih izvedenih varijanti. Baza se pretražuje po vrijednostima kljucnih parametara elektromagnetskih karakteristika motora. Ako se pronade odgovarajuce dovoljno slicno ili identicno rješenje, poziva se cvor 3.
Ako niti jedno od postojecih rješenja iz arhive ne odgovara (što je najcešci slucaj), treba pripremiti podatke za konstruiranje nove varijante rotora, te se u tom slucaju poziva cvor 4.

Cvor 3:

Ako je pri pretraživanju baze pronadeno rješenje koje odgovara projektnim podacima proizašlim iz zahtjeva narucitelja, potrebno je samo lansirati radne naloge pripremi proizvodnje. U tom slucaju sucelje baze treba “prepisati” identifikacijske brojeve dokumentacije vec izvedenog rješenja. Objekti prikaza proizvoda koji se obraduju u cvoru 3 su baza izvedenih rješenja i radni nalozi pripremi proizvodnje.

Cvor 4:

Zadatak ovog cvora je pripremiti sve potrebne podatke za konstruiranje sklopa vratilo-ležaji-rotorski paket. Elektromotor može imati prijenos momenta na jednu ili obje strane vratila, a razlicite su i konstrukcije horizontalne i vertikalne izvedbe, stoga je svakoj od ovih vrsta konstrukcije dodijeljen poseban cvor (5, 6 i 7). Prva akcija u cvoru 4 ustanovljava o kojoj vrsti izvedbe elektromotora se radi. Slijedeca akcija poziva sucelje odgovarajuceg CAD modela izvedbe elektromotora. Ovdje je pretpostavljeno da za svaku vrstu izvedbe postoji parametarski CAD model u kojem je vec definiran oblik, tj. “layout” konstrukcije. Zadatak sucelja CAD modela je transfer vrijednosti parametara koje su odredene u cvoru 1 prema vrijednostima parametara CAD modela. Nacin izvedbe sucelja ovisi najviše o mogucnostima konkretnog CAD programa koji se koristi.
U procesu odlucivanja, ovisno o narucenoj vrsti izvedbe elektromotora, poziva se jedan od cvorova 5, 6 ili 7.

Cvorovi 5,6,7:

Cvorovi 5, 6 ili 7 "otvaraju" parametarski CAD model sklopa rotora.. Kad konstruktor završi konstrukciju nove varijante sklopa rotora, "zatvara" CAD model, a sucelje CAD modela treba "prepisati" sve novoodredene vrijednosti u odgovarajuce parametre modelirane kao objekte. Ovdje izdvajamo samo parametre koji su ulazni podaci u kontrolni proracun sklopa rotora (u tablicnom prikazu plana oznaceni su kao "grupa2"):

Is: broj segmenata vratila
ds_i, ls_i : promjeri i dužine segmenata vratila
ol_i: oznake odabranih kuglicnih ležajeva
G_i: težine tlacnih ploca, balansnih prstena, ventilatora
e: pretpostavljeni ugradbeni ekscentricitet

U procesu konstruiranja rotora potrebno je provjeriti i skup razlicitih ogranicenja koja se postavljaju pri provjeri postuvjeta preliminarne izvedbe konstrukcije (u cvorovima 5, 6 i 7). Ako CAD program koji se koristi ima takve mogucnosti, ogranicenja se mogu postaviti unutar modela, u suprotnom nakon zatvaranja CAD modela u procesu obrade cvorova 5, 6 ili 7 pokrece se provjera postuvjeta. Postuvjeti sadrže izraze u kojima su referencirani parametri odredeni u cvoru 1 i u cvoru 5, 6 ili 7.
Ako neka od ogranicenja nisu ispunjena, potrebno je mijenjati dimenzije dijelova sklopa rotora, dakle u procesu obrade cvora vratiti se na otvaranje CAD modela. Ogranicenja u prikazanom primjeru plana uglavnom se odnose na izbor parametara izolacijskih dijelova i razmake dijelova unutar sklopa kojih se konstruktor mora pridržavati obzirom na medudjelovanja magnetskih polja.

Cvor 8:

Cvor 8 treba "prikupiti" sve potrebne podatke za kontrolni proracun rotora. To su parametari generirani elektromagnetskim proracunom u cvoru 1 i geometrijski parametri, te težine dijelova, odredeni u jednom od cvorova 5, 6 ili 7. Cvor 8 poziva programski alat koji treba kreirati ulaznu ASCII datoteku kontrolnog proracuna rotora. Na temelju dimenzija segmenata vratila (odredenih u prethodnom cvoru) treba podijeliti svaki segment na više polja i izracunati i upisati sve potrebne podatke u ulaznu datoteku kontrolnog proracuna. Ovaj postupak detaljnije je opisan u [141] i [142]. Programski alat za kreiarnje ulazne datoteke koristi i bazu podataka o standardnim kuglicnim ležajevima.

Cvor 9:

Cvor 9 poziva programski alat za kontrolni proracun, a sucelje programskog alata obavlja transfer vrijednosti iz izlazne datoteke proracuna prema parametrima cije granice vrijednosti treba provjeriti nakon proracuna (u tablicnom prikazu plana oznaceni su kao "grupa3"):

lh: vijek trajanja ležajeva, *nl*: nagib elasticne linije vratila na mjestu ležaja
rmax: maksimalni progib vratila ispod rotorskog paketa limova
nk: prvi kriticni broj okretaja
sigmax: maksimalno naprezanje vratila

Pravila odlucivanja u cvoru 9:

1. Ako je proracun pokazao "struganje" rotora po statoru - povratak na elektromagnetski projektni proracun - potrebno je skratiti paket, ili uzeti drugu visinu osovine (slijedecu vecu), dakle proces se vraca na cvor 1.

2. Ako je progib veci od 20 % zracnog raspora - konzultirati konstruktora koji odlucuje da li se prelazi na promjenu konstrukcije (cvor 4) ili promjenu projekta (cvor 1).
3. Ako je kriticna brzine vrtnje blizu nominalne (+/- 10%) - konzultirati konstruktora koji odlucuje o daljnjem tijeku procesa
4. Ako je vijek trajanja ležaja manji od zahtijevanog, ili je nagib elasticne linije u ležaju veci od dvije kutne minute, povratak na cvor 4, izbor drugog ležaja i ponavljanje kontrolnog proracuna
5. Ako svi prethodni uvjeti nisu ispunjeni, znaci da konstrukcija zadovoljava - može se nastaviti na cvoru 10

Cvor 10:

Ovaj cvor pokrece se ako su svi uvjeti kontrolnog proracuna zadovoljeni, a zadatak cvora je razrada radionicke dokumentacije. Ako se koristi CAD program koji na temelju 3D modela može generirati crteže, ovaj cvor otvara opet isti CAD model kao i u cvorovim 5, 6 ili 7. Nakon zatvaranja CAD modela slijedeca akcija cvora treba generirati identifikacijske brojeve crteža. Na kraju treba upisati vrijednosti kljucnih parametara u bazu gotovih rješenja.

Slijedi pregled svih objekata plana po klasama. Objektima su dodijeljene oznake koje se koriste u tablicnom prikazu plana danom na kraju ovog poglavlja.

Objekti prikaza proizvoda:

- lista zahtjeva narucioca (OPP1)
- racunski podaci elektromagnetskog proracuna (OPP2)
- baza izvedenih varijanti (podaci elektromagnetskog proracuna relevantni za pretraživanje) (OPP3)
- parametarski CAD modeli sklopa rotora razlicitih varijanti izvedbi elektromotora (OPP4, OPP5, OPP6)
- baza podataka o standardnim kuglicnim ležajevima koji se ugraduju u sve izvedbe elektromotora (OPP7)
- podaci kontrolnog proracuna sklopa rotora (OPP8)
- radionicki crteži dijelova sklopa rotora (OPP9)

Programski alati:

- elektromagnetski projektni proracun rotorskog paketa limova i bakra (PA1)
- baza podataka o izvedenim varijantama (PA2)
- 3D parametarski CAD sustav (PA3)
- priprema i izrada ulazne datoteke kontrolnog proracuna rotora (PA4)
- kontrolni proracun sklopa rotora (PA5)

Sucelja programskih alata:

- sucelje elektromagnetskog projektnog proracuna: transfer ulaznih podataka iz liste zahtjeva i transfer izlaznih podataka elektromagnetskog proracuna u skup parametara navedenih u slijedecoj tocci (SPA1)
- sucelje parametarskog CAD modela: transfer podataka iz skupa parametara s geometrijskim podacima rotorskog paketa limova (ulazni podaci za prilagodnu

konstrukciju) i transfer podataka prema skupu parametara s geometrijskim podacima vratila (preliminarna konstrukcija na kojoj treba provesti kontrolni proracun) (SPA2)
- sucelje kontrolnog proracuna rotora: transfer podataka prema skupu parametara koji su kljucni rezultati proracuna i koji se referenciraju u provjerama ogranicenja i pravilima odlucivanja (SPA3)

Kao rezime prethodnih izlaganja primjera plana dan je tablicni prikaz iz kojeg se vidi shema direktnih i indirektnih referenciranja prije navedenih grupa objekata u pojedinom cvoru plana. Pritom su korištene oznake objekata navedene u ranijem tekstu u zagradama.

cvor	objekt prikaza proizvoda koji se obraduje	akcije	sucelje programskog alata	pozvani programski alat	parametri koji se odreduju	pravila odlucivanja
1	OPP1, OPP2	poziv PA1, preko SPA1	SPA 1	PA1	grupa 1	
2	OPP3	postavljanje upita u bazi		PA2		ako je varijanta vec izvedena, pozovi cvor 3
3	nalog pripremi proizvodnje	obrada naloga				
4	OPP4, OPP5, OPP6	poziv SPA2	SPA2			izbor narucene varijante izvedbe elektormotora
5	OPP4, OPP5, OPP6	otvaranje i rad na CAD modelu		PA3	grupa 2	
6	OPP4, OPP5, OPP6	otvaranje i rad na CAD modelu		PA3	grupa 2	
7	OPP4, OPP5, OPP6	otvaranje i rad na CAD modelu		PA3	grupa2	
8	OPP2, OPP7,: OPP4, 5 ili 6	poziv PA4		PA4		
9	OPP8	poziv PA5	SPA3	PA5	grupa 3	5 pravila navedenih u opisu cvora 9
10	OPP9	otvaranje i rad na CAD modelu		PA3		

Tablica 11: Shema referenci cvorova razmatranog primjera plana

9. Zakljucak

U skladu s postavljenim ciljem u radu je koncipirana osnovna struktura objektnog modela procesa konstruiranja. Realni sustav cije modeliranje je razmatrano jest proces konstruiranja zamišljen u uredu koji koristi racunalnu opremu u mrežnom okruženju i u kojem je vec "uhodano" korištenje programskih alata za podršku konstruiranju.
U uvodnom dijelu analizirane su znacajke procesa konstruiranja, uz promatranje konstruiranja kao procesa rješavanja zadatka. Dan je pregled polazišta koncipiranja modela, pri cemu su razmatrani teorijski modeli konstruiranja, primjena metoda planiranja (iz podrucja umjetne inteligencije), opcenite metode prikaza procesa i topologija prikaza redoslijeda izvršavanja akcija u procesu konstruiranja. Analizirane su metode modeliranja i projektiranja objektnih sustava, uz osvrt na problematiku modeliranja informacijskih sustava u inženjerstvu.

Postavljen je koncept metodologije istraživanja u kojem se pojave i koncepti realnog svijeta procesa konstruiranja apstrahiraju i preslikavaju u entitete konceptualne domene. U drugoj stavci hipoteze pretpostavljeno je da se tako definirani entiteti mogu potom preslikati u klase objektnog modela procesa konstruiranja. Ovakva metodologija istraživanja odabrana je da bi se kreirao objektni model procesa konstruiranja koji je "prirodniji" i razumljiviji, odnosno ni deskriptivan ni preskriptivan, nego usmjeren racunalnoj primjeni. U objektno orijentiranom pristupu naglasak je na modeliranju stvarnosti u domeni problema umjesto stvaranja arhitekture modela sustava koja vodi k implementaciji. Stoga se nastojalo preslikati entitete u objekte principom "jedan za jedan", tj. da jednom entitetu konceptualne domene odgovara tocno jedan entitet objektne domene. Predloženi entiteti konceptualnog modela detaljno su razradeni u sedmom poglavlju rada, u kojem su razradena i njihova preslikavanja u klase objektnog modela. Time je potvrdena treca stavka hipoteze. Struktura i definicije predloženih entiteta, te realizacija njihovog preslikavanja u klase objektnog modela originalan su znanstveni doprinos.

Željelo se razviti sustav koji bi se mogao primjenjivati neovisno o fazi procesa konstruiranja i vrsti konstrukcijskog zadataka. Ovako postavljeni zahtjev nastojalo se ispuniti tretiranjem procesa konstruiranja primarno kao procesa obrade i generiranja informacija - ne gledajuci obradu informacija s konstrukcijskog, nego s informatickog aspekta. Pri tome se pretpostavlja da (s informatickog aspekta) opcenite znacajke procesa obrade i generiranja informacija ne ovise o fazi procesa i vrsti konstrukcije.

U takvom (informatickom) pristupu promatrane su složene mreže relacija izmedu struktura inženjerskih podataka, te je postavljena prva stavka hipoteze - da je takvu mrežnu topologiju moguce efikasno modelirati objektno orijentiranim metodama. U poglavlju 7.2 (tablica 3) razmatrane su mogucnosti povezivanja entiteta definiranih kao osnovnih "gradbenih" elemenata modela. Zakljuceno je da je u objektnom modelu moguce postaviti i zapisati kompletnu mrežu relacija (svaki entitet sa svakim), ali da jedan dio tih relacija nema znacenja u kontekstu procesa konstruiranja. Stoga su dalje razradene samo one relacije zavisnosti i relacije "pripadnosti" izmedu gradbenih entiteta modela koje imaju svoj semanticki smisao u procesu konstruiranja.

Predložen je zapis relacija izmedu skupova instanci klasa u matricnom obliku, gdje se matrice svode na liste sa nekoliko stupaca. Prikazi relacija, definirani u poglavlju 7.2, realizirani su u okruženju objektne baze podataka.
Pokazalo se da je predloženim pristupom moguce zapisati vrlo složene mreže relacija, sa višestrukim razinama referenciranja (povezivanja). Mehanizmi upravljanja relacijama izmedu objekata u objektnoj bazi daju daleko sofisticiranije i efikasnije mogucnosti modeliranja složenih mreža relacija, nego drugi pristupi (npr. relacijske baze podataka). Drugaciji pristupi zahtijevali bi realizaciju isuviše složenih algoritama i struktura podataka i procedura koji bi bili izuzetno zahtjevni za implementaciju i održavanje. Uz to, osnovna je prednost objektnog pristupa što se relacije ne postavljaju izmedu "stupaca tablica", nego izmedu objekata, koji su modeli pojmova i inženjerskih struktura podataka preslikanih iz konceptualnog modela procesa konstruiranja. Preslikavanjem složenih inženjerskih struktura podataka u skup tablica gubi se jedan dio semantike modela.
Realiziranjem zapisa mreže relacija (izmedu objekata u koje su preslikani entiteti konceptualnog modela) u okruženju objektne baze, dokazana je prva stavka postavljene hipoteze. Razmatranja strukture i nacin realizacije mreže relacija u predloženom objektnom modelu procesa konstruiranja, originalan su znanstveni doprinos ovog rada.

Mehanizmi nasljedivanja u objektnoj tehnologiji daju još jednu znacajnu prednost predloženoj koncepciji. Podklase nasljeduju sve definicije relacija svojih "nadredenih klasa", pa se sustav može vrlo jednostavno dogradivati uvodenjem novih podklasa kao daljnjih specijalizacija vec definiranih klasa. Time se omogucava inkrementalno unapredivanje i daljnji razvoj modela, pri cemu ne treba raditi nikakve promjene osnovne strukture zapisa relacija u vec definiranim klasama.

Fleksibilnost predloženog modela pri uvodenju u eksploataciju nastojala se ostvariti "bottom up" pristupom pri gradnji elemenata modela. Entiteti konceptualnog modela i klase objektnog modela u koje su entiteti preslikani koncipirani su tako da cijeli sustav djeluje u odnosu na korisnika kao "otvorena kutija s alatima". Tako koncipiran model trebao bi biti otvoren za dogradnje i prilagodavanja specificnim uvjetima odredene realne okoline. Stoga je postavljena treca stavka hipoteze - da se temeljem osnovnih klasa modela grade složeni objekti koji modeliraju prikaz i dinamiku izvodenja procesa.

Na vrhu hijerarhije, kao najsloženiji objekt je "plan konstruiranja" definiran kao model dekompozicije i tijeka izvodenja procesa konstruiranja. Osnovni element plana je "cvor plana" definiran kao model etape procesa konstruiranja u kojem se odvija proces izvodenja akcija na objektima zapisa informacija o proizvodu. Osim liste akcija, cvor plana enkapsulira provjere preduvjeta i postuvjeta izvodenja akcija, te zapis znanja o planiranom tijeku izvodenja procesa konstruiranja u obliku pravila odlucivanja. Osnovni entiteti modela (definirani u poglavlju 7.1) i entiteti prikaza relacija (definirani u poglavlju 7.2) korišteni su kao gradbeni elementi složenog entiteta "cvor plana".

Plan je realiziran i zapisan u objektnoj bazi kao skup "cvorova plana", veza cvorova, i svih objekata koje cvorovi referenciraju. U radu su usporedene pretpostavke istovremenog izvodenja samo jednog cvora i istovremenog izvodenja više razlicitih cvorova. Pri tome su analizirane implikacije na mogucnosti prikazivanja veza cvorova plana metodom usmjerenog grafa, uz dodatak klasifikacije veza i njihovog medusobnog uvjetovanja. Temeljem navedene analize predložene su smjernice daljnjih istraživanja u kojima bi trebalo koncipirati još složenije entitete koji bi efikasnije mogli modelirati dinamiku izvodenja procesa.

U izloženim razmatranjima pokazalo se da je korištenjem "bottom up" pristupa moguce graditi model procesa konstruiranja tako da se složeniji entiteti grade korištenjem osnovnih, cime je dokazana treca stavka postavljene hipoteze. Predloženi pristup realizira sa fleksibilnijim sustavom, u kojem su definirani entiteti stabilniji u odnosu na promjene okoline (domene) koju modeliraju, nego entiteti koji bi proizašli iz "top-down" pristupa. Sustav u kojem su klase "slabo" medusobno zavisne lakše je i programski održavati.
Definicije i struktura plana konstruiranja i cvora plana konstruiranja nastavak su i proširenja istraživanja u [6] i [7]. Razmatranja mogucnosti modeliranja dinamike procesa izvodenja plana u okruženju predloženog objektnog modela originalan su znanstveni doprinos ovog rada. Na osnovu mogucnosti koje pruža objektna tehnologija predložena su rješenja koja mogu implementirati tzv. "dinamicko planiranje", odnosno mijenjanje i prilagodavanje plana u tijeku izvodenja, sukladno novonastalim, u planu nepredvidenim situacijama. U ovoj fazi istraživanja predvideno je da takve promjene plana radi primarno konstruktor, medutim predložena struktura i algoritam izvodenja plana omogucuju da se u plan ugrade i operacije i algoritmi kojima bi plan mogao mijenjati "sam sebe", odnosno elemente (atribute objekata i reference) svog zapisa u objektnoj bazi.

Sazrijevanje UML-a rješava barem jedan od dosadašnjih problema primjene objektno orijentirane tehnologije - postojanje velikog skupa razlicitih metodologija razvoja. UML objedinjava najbolje iz nekoliko vodecih metoda i donosi unifikaciju. Na taj nacin može postati podloga za komunikaciju i suradnju više istraživackih timova koji mogu razvijati i razmjenjivati parcijalne modele koristeci istu notaciju i metodologiju. U takvom pristupu mogao bi se pokrenuti razvoj usaglašenog racunalnog modela procesa konstruiranja. Proces razvoja mogao bi se odvijati kroz cikluse: prijedlozi, rasprava, poboljšanja i kretanje k standardizaciji. Slican proces vec se duže odvija u razvoju STEP standarda, ali koliko je autoru poznato, nema nastojanja da se u STEP ukljuci model procesa konstruiranja.

Pri eventualnom implementiranju racunalnog modela procesa konstruiranja u konstrukcijski ured svakako bi trebalo razmotriti koliki je odnos uloženog rada pri formiranju i održavanju modela u odnosu na unapredenje procesa koje bi se moglo postici. Realno je pretpostaviti da ce taj odnos znatno varirati ovisno o vrsti proizvoda koja se konstruira, i o složenosti i znacajkama procesa konstruiranja u razlicitim sustavima. Autor smatra da bi predloženi model bio efikasan za složene varijantne konstrukcije gdje prevladava timski rad, tj. opcenito u situacijama gdje su naglašeni problemi organizacije i kontrole izvodenja procesa konstruiranja.

LITERATURA

[1] Pahl G., Beitz W., Engineering Design, The Design Council, London, 1988.

[2] Martyrer E., Der Ingenieur und das Konstruieren, Konstruktion, Vol. 12, pp. 1-4, 1960.

[3] Dixon J. R., Finger S., Uvodna rijec (editorial), Research in Engineering Design, Vol. 1 No. 1, 1989.

[4] Vajna S., Wegner B., Optimization of the Product Development Process with Evolution Algorithms and Simultaneous Enginerring, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/67-3/70, WDK, Heurista, 1997.

[5] Wallace D. R., Abrahamson S. M., Borland N. P., Design Process Elicitation Through the Evaluation of Integrated Model Structures, Proceedings of ASME Design Engineering Technical Conference, DETC 99, Las Vegas, 1999.

[6] Pavkovic N., Kreiranje baze scenarija ekspertnog CAD sustava, magistarski rad, Zagreb, FSB, 1996.

[7] Marjanovic D., Implementacija ekspertnih alata u proces konstruiranja, Disertacija, FSB Zagreb, 1995.

[8] Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.

[9] Proceedings of the IFIP TC 5/WG 5.2 Second Workshop on Intelligent CAD, Cambridge, rujan 1988, Amsterdam, North-Holland, 1990.

[10] Mortensen N. H., Design modelling in a Designer's Workbench, disertacija, Lyngby, Technical University of Denmark, 1999.

[11] Blankenburg D., Dhainaut M., Designer's Advanced IT Workbench, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1873-1876, WDK, 1999

[12] Schlenoff C., Knutilla A., Ray S., An Approach to Analyzing Existing Process Representations, Internet stranica: http://www.mel.nist.gov/psl/pubs/spain/paper.htm, 1997.

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

[14] Eder W. E., Design Modeling-A design Science Approach (and Why Does Industry Not Use It?), Journal of Engineering Design, Vol. 9, No. 4, pp. 354-371, 1998.

[15] Maffin D., Engineering Design Models: context, theory and practice, Journal of Engineering Design, Vol. 9, No. 4, pp. 315-327, 1998.

[16] Hubka V., Eder W. E., Design Science, London, Springer, 1996.

[17] Finger S., Dixon J.R., A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive and Computer-Based Models of Design Process, Research in Engineering Design, Vol1, No1, 1989, 51-67

[18] Finger S., Dixon J.R., A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis, and Design for the Life Cycle, Research in Engineering Design, Vol1, No2, 1989, 121-137

[19] Wallace K., Developing a Vision of Engineering Design in the Future, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1665-1670, WDK, 1999.

[20] Horvath I., Vergeest J. S. M., Engineering Design Research: Anno 2000, Proceedings of the 6th International Design Conference DESIGN 2000, pp. 23-29, Zagreb, FSB, WDK, 2000.

[21] VDI 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte, Berlin, Beuth Verlag, 1993.

[22] Hubka V., Theorie der Konstruktionsprozesse, Springer, 1976

[23] Ullman D. G., The Mechanical Design Process, McGraw - Hill, 1992

[24] Hollins B., Design processes to develop the product generation after the product generation after next, Proceedings of the 5th International Design Conference DESIGN '98, pp. 17-22, Zagreb, FSB, WDK, 1998.

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

[26] Chandrasekaran B., A Framework for Design Problem-Solving, Research in Engineering Design, Vol. 1, No. 2, pp. 75-86., 1989.

[27] Colton J. S., Dascanio J. L., An Integrated, Intelligent Design Environment, Engineering with Computers, Vol. 7, No. 3, pp. 11-22., 1991.

[28] Shaw N. K., Bloor M. S., de Pennington A., Product Data Models, Research in Engineering Design, Vol. 1, No. 1, pp. 43-50., 1989.

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

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

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

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

[33] Gauseimer J., Frank T., 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, Heurista, 1995.

[34] Jensen T., 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, Heurista, 1997.

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

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

[37] Blount G. N., Clarke S., Artificial Intelligence and Design Automation Systems, Journal of engineering Design, Vol. 5, No. 4, pp. 299-314., 1994.

[38] Mostow J., 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., 1990.

[39] Miller G. S., Colton J. S., 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., 1992.

[40] Bardasz T., Zeid I., Applying analogical problem solving to mechanical Design, CAD, Vol. 23, No. 3, pp. 202-211., 1991.

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

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

[43] Ullman D. G. , D' Ambrosio B., A Proposed Taxonomy for Engineering Decision Support, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995

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

[45] Smithers T., 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.

[46] Banares-Alcantara R., Representing the engineering design process: two hypotheses, CAD, Vol. 23, No. 9, pp. 595-603., 1991.

[47] Hoeltzel D. A., Chieung W. H., 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., 1989.

[48] Žavbi R., Duhovnik J., Design environment for the design of mechanical drive units, CAD, Vol. 27, No. 10, pp. 769-781., 1995.

[49] Blessing L.T.M., Ball N.R., 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, Heurista, 1997.

[50] Ball N., Matthews P., Wallace K., 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.

[51] Eppinger S. D., Nukala M. V., Whitney D. E., Generalised Models of Design Iteration Using Signal Flow Graphs, Research in Engineering Design, Vol. 9, No. 3, pp. 112-123., 1997.

[52] Schmidt L. C., Cagan J., GGREADA: A Graph Grammar-Based Machine Design Algorithm, Research in Engineering Design, Vol. 9, No. 3, pp. 195-213., 1997.

[53] Pollack M. E., The uses of plans, Artificial Intelligence, Vol. 57, No. 1, pp. 43-68., 1992.

[54] Ndumu D.T., Izzudin B.A., Lloy Smith D., Explaining Design Plans, Knowledge Based Systems, Vol. 9, No. 1, pp. 23-29., 1996.

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

[56] Kostelic A, Znanost o konstruiranju, rukopis knjige, Zagreb, 1996.

[57] Kostelic A., CAD kao podsustav CIM, Predavanje HAZU, Zagreb, 1994.

[58] Lindemann U., A Model of Design Processes of Individual Designers, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 757-762, WDK, 1999

[59] Giaopulis A., Modell für effektive Konstruktionsprozesse, Aachen, Shaker, 1998.

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

[61] Oberšmit E., Nauka o konstruiranju, metodicko konstruiranje i konstruiranje pomocu racunala, SNL Liber, Zagreb, 1985.

[62] Koller R., Konstruktionsmethode für den Maschine, Geräte, und Apparatebau, Springer, 1976.

[63] Rodenacker, W, Methodisches Konstruieren, Springer, 1979.

[64] Roth K., Konstruieren mit Konstruktionskatalogen, Springer, 1980.

[65] Eekels J., Roozenburg N.F.M, Product design: Fundamentals and Methods, Wiley, 1995.

[66] Hollins B., Pugh S., Successful Product Design, London, Butterworths, 1990.

[67] Cross N., Engineering Design Methods - Strategies for Pruduct Design, Chichester, Wiley, 1994.

[68] VDI Richtlinie 2222, Sheet 1, Konzipieren technischer Produkte, Düsseldorf, VDI Verlag, 1973.

[69] Yoshikawa H, General Design Theory as a Formal Theory of Design, in Intelligent CAD I ed. Yoshikawa, Gossard, Procc. of IFIP TC5/WG5.2, pp. 51-63, North Holland, Boston, 1987

[70] Tomiyama T., Yoshikawa H., Extended General Design Theory, Design Theory for CAD, ed. Yoshikawa, Warman, Procc. Of IFIP TC5/WG5.2, Tokyo, North Holland, 1987

[71] Tomiyama T., Kiriyama T., Takeda H., Xue D., Yoshikawa H., Metamodel: A Key to Intelligent CAD Systems, Research in Engineering Design, Vol. 1, No. 1, pp. 19-34, 1989

[72] Veerkamp P., Akman V., Bernus P., ten Hagen P. J. W., IDDL: A Language for Intelligent Interactive Integrated CAD Systems, Intelligent CAD Systems II, pp. 59-74, Springer, 1989

[73] Suh N. P, Principles of Design, UP, Oxford, 1990

[74] Albano L. D., Suh N. P., Axiomatic design and concurrent engineering, CAD, Vol. 26, No. 7, pp. 499-504, 1994

[75] Bercsey T., Vajna S., Ein autogenetischer Ansatz für die Konstruktionstheorie als Beitrag zur Vollständigen Beschreibung des konstruktionsprozesses, Teil 1 und 2, CAD-CAM Report, Vol. 14, No. 2, No. 3, 1994.

[76] Schregenberger J. W., Attribution of Models, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 1191-1194, WDK, 1999

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

[78] Froese T., Models of Construction Process Information, Journal of Computing in Civil Engineering, ASCE, 1995

[79] Froese T., Rankin J., Representation of Construction Methods in Total Project Systems, 5 th Congress On Computing On Civil Engineering, Boston, ASCE, 1988

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

[81] Morris K. C., Mitchell M., Dabrowski C., Fong E., Database Managemnet Systems in Engineering, National Institute of Technology, Gaithersburg, 1992

[82] Smithers T., AI-based design versus geometry-based design or Why design cannot be supported by geometry alone, CAD, Vol. 21, No. 3, pp. 141-150, 1989

[83] Dörner D., Thought and Design - Research Strategies, Single-case Approach and Methods of Validation, Designers, the key to successful product development, Berlin, Springer, 1998.

[84] Blessing L.T.M., Chakrabarti A., Wallace K. M., An Overview of Descriptive Studies in Relation to a General Design Research Methodology, Designers, the key to successful product development, Berlin, Springer, 1998

[85] Ehrlenspiel K.,Günther J., How Do Designers from Practice Design?, Designers, the key to successful product development, Berlin, Springer, 1998

[86] Von der Weth R., Having a Nose for Good Solutions - The Development of Individual Strategies for the Design Process, Designers, the key to successful product development, Berlin, Springer, 1998

[87] Visser W., Designers' activities examined at three levels: organization, strategies and problem-solving processes, Knowledge Based Systems, Vol. 5, No. 1, pp. 92-104, 1992.

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

[89] Miller G. A., Galanter E., Pribram K.H., Plans and the structure of behavior, New York, Holt, 1960.

[90] Hayes-Roth, B. Human planning processes, Rep. No. R-2670-ONR, Rand Corp., Santa Monica, Calif., 1980

[91] Banares-Alcantara R., Design Support Systems for Process Engineering: I. Requirements and Proposed Solutions for a Design Process Representation, University of Edinburgh, Department of Chemical Engineering, Technical Report 1994-07, 1994

[92] Kreyszig E., Advanced Engineering Mathematics, New York, J. Wiley, 1993.

[93] Bojcetic N., Korisnicko sucelje sustava za konstruiranje mehanickih sklopova, magistarski rad, FSB Zagreb, 1996.

[94] Pavkovic N., Štorga M., Marjanovic D., Generating Design Plans in Relational Database Environment - Problems and Improving Possibilities, Proceedings of the 5th International Design Conference DESIGN '98, pp. 93-98, Zagreb, FSB, WDK, 1998.

[95] Pavkovic N., Marjanovic D., Structuring a Designers Workbench with Object Oriented Design Plans, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1401-1406, WDK, 1999.

[96] Eliëns A., Principles of Object-Oriented Software Development, Reading, Addison -Wesley, 1995.

[97] Ege R. K., Object-Oriented Programming with C++, Academic Press, 1994.

[98] Alhir S. S., UML in a Nutshell, O'Reilly, Sebastopol, 1988.

[99] Jordan D., C++ Object Databases, Programming with ODMG Standard, Reading, Addison Wesley, 1998

[100] Quatrani T., Visual Modeling with Rational Rose and UML, Reading, Addison Wesley, 1999.

[101] Booch G., Rumbaugh J., Jacobson I., Unified Modeling Language User Guide, Addison Wesley, 1999.

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

[103] ISO 10303, Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles, Geneve, ISO, 1994.

[104] ISO 10303 - 41, Integrated generic resources: Fundamentals of product description and support, Geneve, ISO, 1994.

[105] ISO 10303 - 43, Integrated generic resources: Representation structures, Geneve, ISO, 1994.

[106] POET 5.0 Programmer's Guide, San Mateo, POET Software Corporation, 1997.

[107] Meyer B., Object-Oriented Software Construction

[108] Eckel B., C++ Inside & Out, Berkeley, Osborne McGraw-Hill, 1993.

[109] Duffy A. H. B., Andreasen M. M., O' Donell F. J., Design Co-ordination, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 113-118, WDK, 1999

[110] Eppinger S. D., Whitney D. E., Smith R. P., Gebala D. A., A Model-Based Method for Organizing Tasks in Product Development, Research in Engineering Design, Vol. 6, No. 1, pp. 1-13, 1994.

[111] Steward D. V., The Design Structure System: A Method for Managing the Design of Complex Systems, IEEE Tansactions on Engineering Management, Vol. EM-28, No. 3, pp. 71-74, 1981.

[112] Steward D. V., Systems Analysis and Management, New York, Petrocelli Books, 1981.

[113] Eppinger S. D., Model-base Approaches to Managing Concurrent Engineering, Journal of Engineering Design, Vol. 2, No. 4, pp. 283-290, 1991.

[114] Rogers J. L., McCulley C. M., Bloebaum C. L., Integrating a Genetic Algotithm into a Knowledge-Based System for Ordering Complex Design Processes, NASA Technical Memorandum 110247, 1996.

[115] Rogers J. L., Reducing Design Cycle Time and Cost Through Process Resequencing, Proceedings of International Conference on Engineering Design ICED 97, WDK, Heurista, 1997.

[116] West M., Developing High Quality Data Models, Shell Int. Petroleum, Report IC 91-077 T4, The Hague, 1993

[117] Howard H. C., Abdalla J. A., Phan D. H., A Primitive-Composite Approach for Structural Data Modelling, Journal of Computing in Civil Engineering, Vol. 6, No. 1, pp. 19-40, 1992

[118] King G. W., Object-Oriented really is better than Structured, Internet stranica: www.cs.uwa.edu.au/Why OOP.htm, 1995.

[119] Clarkson P. J., Hamilton J. R., 'Signposting', A Parameter-driven Task-based Model of the Design Process, Research in Engineering Design, Vol. 12, No. 1, pp. 18-38, 2000.

[120] Watton J. D., Rinderle J. R., Improving Mechanical Design Decisions with Alternative Formulations of Constraints, Journal of Engineering Design, Vol. 2, No. 1, pp. 55-68, 1991.

[121] Nagy R. L., Ullman D. G., Dietterich T. G., A Data Representation for Collaborative Mechanical Design, Research in Engineering Design, Vol. 3, No. 4, pp. 233-242, 1992.

[122] Pavlic D., Marjanovic D., Štorga M., The Implementation of Web-Based Technologies in Engineering Data Management, Proceedings of the 6th International Design Conference DESIGN 2000, pp. 23-29, Zagreb, FSB, WDK, 2000.

[123] Marjanovic D., Bojcetic N., Dekovic D., Pavkovic N., Pavlic D., Štorga M., Žeželj D., Design Department Data Flow Integration, Proceedings of 3rd International Workshop "Integrated Product Development", Magdeburg, rujan 2000, Otto-von-Guericke University Magdeburg.

[124] Gorti S. R., Gupta A., Kim G. J., Sriram R. D., Wong A., An object-oriented representation for product and design processes, CAD, Vol. 30, No. 7, pp. 489-501., 1988.

[125] Liang W.Y., O'Grady P., Design With Objects: an Approach to Object-Oriented Design, CAD, Vol. 30, No. 12, pp. 943-956, 1998

[126] Werner H., Muth M., Weber C., Functional Modeling Using an Object-Oriented Design System, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/234-3/238, WDK, Heurista, 1997.

[127] Korpela T., The Role of Object-Oriented Analysis in Modelling of Technical Processes, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 853-856, WDK, 1999.

[128] Jovic M., Razvoj modela za opisivanje i sprezanje sklopova mehanickih konstrukcija, magistaski rad, FSB Zagreb, 1997.

[129] Varga M., Baze podataka, Zagreb, ZEUS, 1994.

[130] Blessing L., A Process-Based Approach to Computer Supported Engineering Design, Proceedings of International Conference on Engineering Design ICED 93, Vol. 3., pp. 1393-1400, WDK, Heurista, 1993.

[131] Ledet W. P., Himmelblau D. M., Decomposition Procedures for the Solving of Large Scale Systems, Advances in Chemical Engineering, Vol. 8, pp. 185-254, 1970

[132] Gebala D. A., Eppinger S. D., Methods for Analyzing Design Procedures, ASME Conference on Design Theory and Methodology, pp. 227-233, Miami, 1991.

[133] Park H., Cutkosky M. R., Framework for Modeling Dependencies in Collaborative Engineering Processes, Research in Engineering Design, Vol. 11, No. 2, pp. 84-102, 1999.

[134] Smith R. P., Morrow J. A., Product development process modeling, Design Studies, Vol. 20, No. 3, pp. 237-261, 1999.

[135] Herold Z., Strukturiranje baze znanja u procesu konstruiranja, disertacija, Zagreb, FSB, 1997.

[136] Grabowski H., Lossack R.-S., Weis C., A Design Process Model based on Design Working Spaces, Proceedings of the IFIP TC5 WG5.2 International Conference on Knowledge Intensive CAD, Vol. 1, pp. 245-262, Helsinki, Chapman & Hall, 1995.

[137] Grabowski H., Lossack R.-S., Knowledge based design of complex products by the concept of design working spaces, Proceedings of the IFIP TC5 WG5.2 International Conference on Knowledge Intensive CAD, Vol. 2, pp. 79-98, Pittsburgh, Chapman & Hall, 1996.

[138] Sutton R. A., Planning by Incremental Dynamic Programming, Proceedings of Ninth Conference on Machine Learning, pp. 353-357, Morgan-Kaufmann, 1991.

[139] Abrahamson S., Wallace D., Senin N., Sferro P., Integrated Design in a Service Marketplace, MIT CADlab, 1999.

[140] Stanek J., Aufbau von Wissenbasen in Ingenieurdatenbanksystemen zur Integralen Produktentwicklung, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995

[141] Pavkovic, N, Programi za proracun vratila i ležaja, te crtanje vratila - tehnicke upute za korištenje, elaborat br. 3.020928, Koncar - Srednji elektricni strojevi, 1991.

[142] Pavkovic, N, Programi za proracun vratila i ležaja, te crtanje vratila - upute za održavanje programa, elaborat br. 3.020930, Koncar - Srednji elektricni strojevi, 1991.

BIBLIOGRAFIJA

[1] Abeln O., Meerkamm H., Storath E., Weber U., The CAD Reference Model - On Its First Approach to Praxis, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995

[2] Abramovici M., Gerhard D., Use of PDM in Improving Design Processes - State of the Art, Potentials and User Perspectives, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/317-3/322, WDK, Heurista, 1997.

[3] Al-Salka M. A., Cartmell M. P., Hardy S. J., A Framework for a Generalized Computer-based Support Environment for Conceptual Engineering Design, Journal of Engineering Design, Vol. 9, No. 1, pp. 57-88, 1998

[4] Andreasen M. M., Design Methodology, Journal of Engineering Design, Vol. 2, No. 4, pp. 321-335, 1991

[5] Andreasen M. M., Pulkkinen A., Modular Engineering at ICED '99, 17. WDK Workshop, Rigi, WDK, 2000.

[6] Andreasen M. M., Wognum N., Considerations on a design typology, Proceedings of the 3rd International Workshop "Integrated Product Development", IPD 2000, Magdeburg, Otto-von-Guericke Univeristy Magdeburg, 2000.

[7] Arbab F., Design Object Modeling, Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.

[8] Austin S., Baldwin A., Baizhan L., Waskett P., Analytical Design Planning Technique: a Model of the detailed building design process, Design Studies, Vol. 20, No. 3, pp. 279-296, 1999.

[9] Bashir H. A., Thomson V., Metrics for design projects: a review, Design Studies, Vol. 20, No. 3, pp. 263-277, 1999.

[10] Bashir H. A:, Thomson V., Estimating Design Complexity, Journal of Engineering Design, Vol. 10, No. 3, pp. 247-257, 1999.

[11] Bauert F., Ball N. R., The integrated design framework: blackboards as a communication medium between designers and software modules, Proceedings of International Conference on Engineering Design ICED 93, Vol. 3., pp. 1351-1357, WDK, Heurista, 1993.

[12] Beitz W., Feldhusen J., Management Systems and Program Concepts for an Integrated CAD Process, Research in Engineering Design, Vol. 3, No. 2, pp. 61-74, 1991.

[13] Beitz W., Helbig D., Entwicklung Produkt - und unternehmensorientierter Konstruktionsleitsysteme, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995

[14] Birmingham R. W., Information as the Raw Material of Design: an Educational Perspective, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/439-3/444, WDK, 1999.

[15] Blum A. L., Furst M. L., Fast Planning Through Planning Graph Analysis, Artificial Intelligence, No. 90, pp. 281-300, 1997

[16] Bojcetic N., Jovic M., The model of user interface in design process, Proceedings of International Conference on Engineering Design ICED 95, pp. 1563-1564, WDK, Heurista, 1995.

[17] Boudouh T., Noyes D., Aldamondo M., Modeling Reactive Design Processes in Concurrent Engineering, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 333-336, WDK, 1999

[18] Bruce M., Cooper R., Vazquez D., Effective design management for small businesses, Design Studies, Vol. 20, No. 3, pp. 297-315, 1999.

[19] Cannon D. M., Leifer L. J., The Place of Design Research in the World of Contemporary Philosophical Thought, and Vice-Versa: an Illustrative example, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1635-1640, WDK, 1999

[20] Cao Q., Protzen J.P., Managing design information: Issue-Based Information Systems and Fuzzy Reasoning System, Design Studies, Vol. 20, No. 4, pp. 343-362, 1999.

[21] Chen K. Z., Identifying the Relationship among Design Methods: Key to Successful Applications and Developments of Design Methods, Journal of Engineering Design, Vol. 10, No. 2, pp. 125-141, 1999.

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

[23] Christiaans H.H.C.M., van Andel J., Information Processing and Storage during the Design Process: The Use of a Flexible Information System, Designers, the key to successful product development, Berlin, Springer, 1998

[24] Clarkson P. J., Hamilton J. R., "Signposting the Design Process", Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 107-112, WDK, 1999
[25] Cronemyr P., Ronnback A., Eppinger S. D., Process Improvement Simulations Using the Work Transformation Model, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 349-352, WDK, 1999
[26] Cross N., Natural intelligence in design, Design Studies, Vol. 20, No. 1, pp. 25-39, 1999.
[27] Date C. J., An Introduction to Database Systems, Raeding, Addison Wesley, 1991.
[28] Dekovic D., Bojcetic N., Structure of Design Domain Knowledge Base, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1921-1924, WDK, 1999
[29] Deng Y.-M., Britton G. A., Tor S. B., A Design Perspective of Mechanical Function and its Object-Oriented Representation Scheme, Engineering with Computers, Vol. 14, No. 4, pp. 309-320, 1998.
[30] Dixon L. A., Colton J. S., An Anchoring Adjustment Process Model for Redesign, Journal of Engineering Design, Vol. 9, No. 4, pp. 297-313, 1998.
[31] Domajnko T., Juric M. B., Rozman I., Selecting correct object storage system, Proceedings of the 22nd International Convention MIPRO '99, Vol. 1, pp. 99-102, Zagreb, MIPRO, 1999
[32] Dong Y., Goh A., An Intelligent Database for Engineering Applications, Artificial Intelligence in Engineering, Vol. 12, No. 1-2, pp. 1-14, 1997
[33] Drabble B., EXCALIBUR: a program for planning and reasoning with processes, Artificial Intelligence, Vol. 62, No. 1, pp. 1-40, 1993
[34] Drisis L., What Do Our Designers Really Want? A Survey, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 567-570, WDK, 1999
[35] Duffy A. H. B., Legler S., Rationalising Past Designs for Reuse, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 377-380, WDK, 1999
[36] Eastman C. M., Bond A. H., Chase S. C., A Formal Approach for Product Model Information, Research in Engineering Design, Vol. 2, No. 2, pp. 65-80, 1991.
[37] Eastman C. M., Fereshetian N., Information models for use in product design: a comparison, CAD, Vol. 26, No. 7, pp. 551-572, 1994.
[38] Eastman C., Parker D. S., Jeng T-S., Managing the Integrity of Design Data Generated by Multiple Applications: The Principle of Patching, Research in Engineering Design, Vol. 9, No. 2, pp. 125-145, 1997.
[39] Eastman, C. M., Bond, A. H., Chase, S. C., A data model for design databases, Artificial Intelligence in Design '91, pp. 339-366, Butterworth-Heineman Ltd, Oxford, 1991.
[40] Ehrlenspiel K., Practicians - How they are Designing? …And Why?, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 721-726, WDK, 1999
[41] Endebrock K., Welp E. G., Object-Oriented Product and Process Modeling for Transparent Cost Estimation, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 493-496, WDK, 1999
[42] Eversheim W., Graessler R., Schulten I., Parallel Processes Managed by an Integrated CAD-CAPP System, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 381-384, WDK, 1999
[43] Fertalj K., Kalpic D., An Object Based Software Development Technique, Proceedings of the 21st International Conference on Information Technology Interfaces, ITI '99, pp. 469-474, Zagreb, SRCE, University of Zagreb, 1999
[44] Fowler J., STEP for Data Management, Exchange & Sharing, Twickenham, Technology Appraisals, 1995.
[45] Fricke G., Succesful Individual Approaches in Engineering Design, Research in Engineering Design, Vol. 8, No. 3, pp. 151-165, 1996
[46] Fuh J. Y. H., Chang C-H., Melkanoff M. A., The development of an integrated and intelligent CAD/CAPP/CAFP environment using logic-based reasoning, CAD, Vol. 28, No. 3, pp.217-232, 1996.
[47] Fulcher A. J., Hills P., A Taxonomy of Design Research Topics by Multivariate Agglomerative Clustering, Journal of Engineering Design, Vol. 9, No. 4, pp. 342-353, 1998.
[48] Gabbert U., Wehner P., The Product Data Model as a Pool for CAD-FEA Data, Engineering with Computers, Vol. 14, No. 2, pp. 115-122, 1998
[49] Gao Y., Zeid I., Bardasz T., DEPSY: A Design plan System Supporting Case-based Mechanical Design, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995.
[50] Gauseimer J., Lewandowski S., Kespohl H. D., Pusch R., Seifert L., Gateway Integration of Global Engineering Networking (GEN) and Product Data Management (PDM), Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 703-708, WDK, 1999

[51] Gero J. S., Mc Neill T., An approach to the analysis of design protocols, Design Studies, Vol. 19, No. 1, pp. 21-61, 1998.
[52] Gershenson J. K., Stauffer L. A., A Taxonomy for Design Requirements form Corporate Customers, Research in Engineering Design, Vol. 11, No. 2, pp. 103-115, 1999.
[53] Giaopulis A., Schlüter A., Ehrlenspiel K., Günther J., Effizientes Konstruieren durch Generierendes und Korrigierendes Vorgehen, Proceedings of International Conference on Engineering Design ICED 95, pp. 477-483, WDK, Heurista, 1995.
[54] Girard P., Eynard B., Rimmer D., Hein L., Re-engineering of Design Processes in a Design Co-ordination Envirinment, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 337-340, WDK, 1999
[55] Goker M. H., Retrieving Adaptable Solutions from a Design Repository, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1389-1394, WDK, 1999
[56] Grabowski H., Rude S., Pocsai Zs., Ontology Technology the Key for Intelligent Migration and Retrieval of Information in Engineering Networks, Proceedings of International Conference on Engineering Design ICED 97, Vol. 2, pp. 2/231-2/236, WDK, Heurista, 1997.
[57] Halpin, T. A., Nijssen, G. M., Conceptual Schema and Relational Database Design, Prentice-Hall, USA, 1989.
[58] Hansen C. T., Identification of Design Work Patterns by Retrospective Analysis of Work Sheets, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 775-780, WDK, 1999
[59] Höhn B.-R., Steingröver K., Dyla A., Computer Aided Product Development, Proceedings of the 5th International Design Conference DESIGN 2000, Zagreb, FSB, WDK, 2000.
[60] Huang G. Q., Mak K. L., Developing a Generic Design for X Shell, Journal of engineering Design, Vol. 8, No. 3, pp. 251-260, 1997
[61] Hubka V., Eder W. E., Theoretical Approach in Design Methodology, Designers, the key to successful product development, Berlin, Springer, 1998
[62] IDEF1X Manual, USAF Integrated Computer-Aided Manufacturing Program. D'Appleton Company, USA, 1986.
[63] Jonas W., Design as problem-solving? Or: here is the solution-what was the problem?, Design Studies, Vol. 14, No. 2, pp. 157-170, 1993.
[64] Kalpic B., Polajnar A., Model of the Holistic Information Integration of an Enterprise, Strojarstvo, Vol. 39, No. 6, pp. 275-280, 1997
[65] Kimura F., Architecture of Intelligent CAD Systems, Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.
[66] Krishnamurthy M. V., Smith F. J., Integration of scientific data and formulae in an object-oriented knowledge-based system, Knowledge Based Systems, Vol. 7, No. 2, pp. 135-141, 1994
[67] Langdon P., Chakrabarti A., Browsing a Large Solution Space in Breadth and Depth, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1865-1868, WDK, 1999
[68] Lau H., The New Role of Intranet/Internet Technology for Manufacturing, Engineering with Computers, Vol. 14, No. 2, pp. 150-155, 1998
[69] Lee H. H., Arora J. S., Object-Oriented Programming for Engineering Applications, Engineering with Computers, Vol. 7, No. 3, pp. 225-235, 1991
[70] Lemay L., Perkins C. L., Teach Yourself JAVA in 21 days, Indianapolis, Sams.net Publishing, 1996.
[71] Liang J., Chang T. Y. P., Chan C. M., An Object-Oriented Database Management System for Computer-Aided Design of Tall Buildings, Engineering with Computers, Vol. 14, No. 4, pp. 275-286, 1998
[72] Liddament T., The computationalist paradigm in design research, Design Studies, Vol. 20, No. 1, pp. 41-56, 1999.
[73] Lippold C., Welp E. G., Multi-Domain Configuration System for Analysis and Synthesis of Mechatronic Conceptual Designs, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 829-834, WDK, 1999
[74] Lockledge J. C., Salustri F. A., Defining the Engine Design Process, Journal of Engineering Design, Vol. 10, No. 2, pp. 109-123, 1999.
[75] Maher M. L., Rutherford J. H., A Model for Synchronous Collaborative Design Using CAD and Database Management, Research in Engineering Design, Vol. 9, No. 2, pp. 85-98, 1997
[76] Marjanovic D., Design Process Representation in the Developmnet of Design Support Environment, Proceedings of International Conference on Engineering Design ICED 97, Vol. 2, pp. 2/705-2/708, WDK, Heurista, 1997.

[77] Marjanovic D., Pavkovic N., The Structure of an ICAE System, Proceedings of International Conference on Engineering Design ICED 95, pp. 1363-1368, WDK, Heurista, 1995.
[78] Marples M.L., The Decisions of Engineering Design, IRE Transactions on Engineering Managment, EM, 8, 55-70, 1961
[79] Martin J., Principles of Object-Oriented Analysis and Design, New Yersey, Prentice Hall, 1993
[80] Martin J., Principles of Object-Oriented Analysis and Design, New jersey, Prentice-Hall, 1993.
[81] McCullough, Issues in Intelligent CAD, Proceedings of the IFIP TC 5/WG 5.2 Second Workshop on Intelligent CAD, Cambridge, rujan 1988, Amsterdam, North-Holland, 1990.
[82] McMahon C. A., Lowe A., Culley S. J., An Information-Connection Model for Design, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1651-1656, WDK, 1999.
[83] Meerkamm H., Information Management in the Design Process - Problems, Approaches and Solutions, Designers, the key to successful product development, Berlin, Springer, 1998.
[84] Mills J. J., A Taxonomy of the Product Realization Process Environment, Research in Engineering Design, Vol. 4, No. 4, pp. 203-213, 1993.
[85] Moore J., Stader J., Chung P., Jarvis P., Macintosh A., Ontologies to Support the Management of New Product Development in the Chemical Process Industries, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 159-164, WDK, 1999
[86] Myopolous J., Cohen P., Borgida A., Sugar L., Semantic networks and the generation of context, IJCAI 4, 134-142, 1975.
[87] Object Management Group, Unified Modeling Language Specification, version 1.3, Framingham, OMG, 1999.
[88] Ohsuga S., A Consideration to Intelligent CAD Systems, Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.
[89] Onosato M., Yoshikawa H., A Framework on Formalization of Design Object for Intelligent CAD, Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.
[90] Oosterman B. J., Gaalman G. J. C., Kuijpers F. P. J., Finding Structures in Product Development, University of Groningen, Research Report 99A06, 1998.
[91] Pavkovic N., Defining and Generating Design Plans - an Approach to the First Phase in Exploitation of an ICAD System, Proceedings of International Conference on Engineering Design ICED 97, Vol. 2, pp. 2/297-2/300, WDK, Heurista, 1997.
[92] Peak R. S., Fulton R. E., Nishigaki I., Okamoto N., Integrating Engineering Design Analysis Using a Multi-representation Approach, Engineering with Computers, Vol. 14, No. 2, pp. 93-114, 1998
[93] Peckham J., Maryanski F., Semantic Data Models, ACM Comput. Surv., Vol. 20, No. 3, pp. 153-189, 1988.
[94] Pejtersen A. M., Sonnenwald D. H., Buur J., Govindaraj T., Vicente K., The Design Explorer Project: Using a Cognitive Framework to Support Knowledge Exploration, Journal of engineering Design, Vol. 8, No. 3, pp. 289-301, 1997
[95] Peters T. J., Demurijan S. A., McCartney R., Needham D. M., Object Modeling to Localize Knowledge for Feature Interrelationships, Proceedings of the IFIP TC5 WG5.2 International Conference on Knowledge Intensive CAD, Vol. 2, pp. 199-207, Pittsburgh, Chapman & Hall, 1996.
[96] POET Software Corporation, POET Internet homepage, A Comparison Between Relational and Object Oriented Databases for Object Oriented Application Development, www.poet.com., 1997.
[97] Randall R. L., Bristow D. J., Drake R., Front-End Process Definition for Projects Engaged in Significant Technology Transition, Internet stranica: http://source.asset.com/stars/loral/pubs/stc96/fep96/fep.htm, 1996
[98] Reich Y., A Critical Review of General Design Theory, Research in Engineering Design, Vol.7, No. 1, pp. 1-18, 1995
[99] Reinders, H, Design Information Deployment in “Design Assistants”, International Conference on Engineering Design, ICED 95, WDK, Heurista, 1995
[100] Reymen I., Domain Independent Description of Design Situations, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 1215-1218, WDK, 1999
[101] Rigopoulus D. R., Oppenheim I. J., Design Objects and their Representation: Buildings and Structures, Proceedings of the IFIP TC 5/WG 5.2 Second Workshop on Intelligent CAD, Cambridge, rujan 1988, Amsterdam, North-Holland, 1990.
[102] Rodgers P. A., Huxor A. P., Caldwell H. M., Design Support Using Distributed Web-Based AI Tools, Research in Engineering Design, Vol. 11, No. 1, pp. 31-44, 1999

[103] Rodgers P. A., The role of artificial intelligence as 'text' within design, Design Studies, Vol. 19, No. 2, pp. 143-160, 1998.
[104] Roozenburg N. F. M., Eekels J., Product Design: Fundamentals and Methods, Chichester, Wiley, 1995.
[105] Roozenburg N.F.M., Dorst K., Describing Design as a Reflective Practice: Observations on Schön's Theory of Practice, Designers, the key to successful product development, Berlin, Springer, 1998
[106] Rosen D. W., Peters T. J, The Role of Topology in Engineering Design Research, Research in Engineering Design, Vol. 8, No. 2, pp. 81-98, 1996
[107] Rosenman M. A., Gero J. S., Modeling Multiple Views of Design Objects in a Collaborative CAD Environment, CAD, Vol. 28, No. 3, pp. 193-205, 1996.
[108] Rugelj J., Intelligent Agent for Construction and Maintenance of Knowledge Trees, Proceedings of the 22nd International Convention MIPRO '99, Vol. 1, pp. 156-158, Zagreb, MIPRO, 1999
[109] Salustri F. A., Venter R. D., An Axiomatic Theory of Engineering Design Information, Engineering with Computers, Vol. 8, No. 4, pp.197-212, 1992
[110] Salustri F. A., Venter R. D., Towards a new computational model for engineering software systems, Proceedings of International Conference on Engineering Design ICED 93, Vol. 3., pp. 1359-1368, WDK, Heurista, 1993.
[111] Savage J. C. D., Miles C., Moore C. J., Miles J. C., The interaction of time and cost constraints on the design process, Design Studies, Vol. 19, No. 2, pp. 217-233, 1998.
[112] Saynisch M., Advanced Product Configuration Change Management, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 317-320, WDK, 1999
[113] Schon A., Meerkamm H., Components for a Mechatronic Design Workbench, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 817-822, WDK, 1999
[114] Schregenberger J. W., The Further Development of Design Methodologies, Designers, the key to successful product development, Berlin, Springer, 1998
[115] Schulz J., Keutgen I., Birkhofer H., User-Oriented Presentation of Available Categorised Knowledge Providing On-Demand a Flexible, Relevant Knowledge-base Access, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 171-176, WDK, 1999
[116] Shahin T. M. M., Sivaloganathan S., Development of a Computer-Based Design Reuse System, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1383-1388, WDK, 1999
[117] Shen W., Barthes J.-P., An object-oriented approach for engineering design product modeling, Proceedings of the IFIP TC5 WG5.2 International Conference on Knowledge Intensive CAD, Vol. 1, pp. 171-187, Helsinki, Chapman & Hall, 1995.
[118] Shin Y., Han S., Data Enhancement for Sharing of Ship Design Models, CAD, Vol. 30, No. 12, pp. 931-941, 1998
[119] Smith R. P., Tjandra P., Experimental Observation of Iteration in Engineering Design, Research in Engineering Design, Vol. 10, No. 2, pp. 107-117, 1998
[120] Suh N. P., A Theory of Complexity, Periodicity and the Design Axioms, Research in Engineering Design, Vol. 11, No. 2, pp. 116-131, 1999.
[121] Suh N. P., Axiomatic Design Theory for Systems, Research in Engineering Design, Vol. 10, No. 4, pp. 189-209, 1998.
[122] Štorga M., Pavkovic N., Marjanovic D., Execution of Design Plans in Relational Database Environment, Proceedings of the 5th International Design Conference DESIGN '98, pp. 221-226, Zagreb, FSB, WDK, 1998.
[123] Takeda H., Tomiyama T., Yoshikawa H., Logical Formalization of Design Processes for Intelligent CAD Systems, Proceedings of the IFIP TC 5/WG 5.2 Second Workshop on Intelligent CAD, Cambridge, rujan 1988, Amsterdam, North-Holland, 1990.
[124] Taura T., Kubota A., A Study on Engineering History Base, Research in Engineering Design, Vol. 11, No. 1, pp. 45-54, 1999
[125] Troussier N., Pourroy F., Tollenaere M., Trebucq B., A Model to Reperesnt Mechanical Calculation Process in an Integrated Design Context, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 1235-1238, WDK, 1999
[126] Tsichritizis, D., Hierarchical database managament, ACM Computer Surveys, Vol. 8, No. 1, pp. 105-124, 1976.
[127] Ullman D. G., A Taxonomy for Mechanical Design, Research in Engineering Design, Vol. 3, No. 3, pp. 179-189., 1992.

[128] Ullman D. G., D'Ambrosio B., A Proposed Taxonomy for Engineering Decision Support, Proceedings of International Conference on Engineering Design ICED 95, pp. 611-616, WDK, Heurista, 1995.
[129] Upton N., Blessing L., Methods for Evaluating Engineering Information Technology, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 253-256, WDK, 1999
[130] Vajna S., An Introduction to Virtual Product Development, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 257-260, WDK, 1999
[131] Vajna S., Burchardt C., Richter A., Organizing Integrated Product Development With Network Structures, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 393-396, WDK, 1999
[132] Vajna S., Schabacker M., Evaluating Benefits of Investments into Engineering Processes, Procedures, Methods and Tools, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 249-252, WDK, 1999
[133] Vajna S., Schabacker M., Schmidt R., Freisleben D., Methodical and Systematical Parametrisation in Product Modeling, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 531-534, WDK, 1999
[134] Van Handenhoven E., Trassaert P., Design Knowledge and Design Skills, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 153-158, WDK, 1999
[135] Varga M., Conversion of Relational into Multidimensional Database Schema, Proceedings of the 21st International Conference on Information Technology Interfaces, ITI '99, pp. 331-338, Zagreb, SRCE, University of Zagreb, 1999
[136] Waldron M. B., Modeling of the Design Process, Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Boston, listopad 1987, Amsterdam, North-Holland, 1987.
[137] Wartzack S., Meerkamm H., Shortening the Process Chain by Early Consideration of the Interaction Between Product and Process, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp.1007-1010, WDK, 1999
[138] Welp E. G., Braun P., Integration Processors as Intelligent System Coupling in Process of Engineering, Proceedings of International Conference on Engineering Design ICED 97, Vol. 3, pp. 3/329-3/336, WDK, 1997.
[139] Welp E. G., Braun P., Knowledge Processing as a Solution and Decision-Making Aid in Early Phases of Embodiment Design, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 3, pp. 1965-1968, WDK, 1999
[140] Werner H., Ahmed S., Design Capturing with a Model System Using Event Triggered Procedures, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 1011-1014, WDK, 1999
[141] Wood W. H., A Methodology for Transforming Information into Design Knowledge, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 1, pp. 131-136, WDK, 1999
[142] Woyak S. A., Myklebust A., Functionality and Data Integration of Software Modlules Through Dynamic Integration, Journal of Engineering Design, Vol. 9, No. 2, pp. 137-158, 1988.
[143] Yabuki N., Law K. H., An Object-Logic Model for the Representation and Processing of Design Standards, Engineering with Computers, Vol. 9, No. 3, pp. 133-159, 1993
[144] Yan X. T., MacCallum K., An Integrated Novel Approach to Enhancing Interdisciplinary Product Design, Proceedings of International Conference on Engineering Design ICED 97, Vol. 1, pp. 1/266-1/270, WDK, 1999.
[145] Yassine A. A., Falkenburg D. R., A Framework for Design Process Specifications Management, Journal of Engineering Design, Vol. 10, No. 3, pp. 223-234, 1999.
[146] Yazdani B., Holmes C., Four Models of Design Definition: Sequential, Design Centered, Concurrent and Dynamic, Journal of Engineering Design, Vol. 10, No. 1, pp. 25-37, 1999.
[147] Zeid I., Bardasz T., Bhaskara S., Gao Y., Design plan systems for cas-based mechanical design: a survey and open issues, Proceedings of International Conference on Engineering Design ICED 93, Vol. 3., pp. 1369-1376, WDK, Heurista, 1993.
[148] Zeng Y., Gu P., A Set-Theoretic Model of Design Process, Proceedings of the 12th International Conference on Engineering Design ICED 99, Vol. 2, pp. 1179-1182, WDK, 1999
[149] Žavbi R., Razvoj izdelka s povezavo funkcije in delovnih principov, disertacija, Fakultet za strojništvo, Ljubljana, 1998

DODATAK1: RJECNIK POJMOVA

Parametar konstrukcije:

Entitet "parametar konstrukcije", preslikan na objekt u modelu procesa konstruiranja enkapsulira:

- zapis vrijednosti podatka o proizvodu,
- status vrijednosti (koji se mijenja u izvodenju procesa konstruiranja),
- reference na dodatne opise - znanja o procesu konstruiranja i proizvodu,
- zapise "namjera" konstruktora - prijedloge, argumente i pretpostavke.

Baza parametara:

Modelira logicku organizaciju skupa parametara konstrukcije modeliranih kao objekata, uz kontrolu jedinstvenosti naziva parametra i pristupa atributima parametra, te sadrži i operacije pretraživanja i pregledavanja.

Prikaz proizvoda:

Entitet "prikaz proizvoda" modelira odredenu vrstu zapisa informacija o proizvodu. Modelira postojanje, odnosno enkapsulira sve pojmove i dogadaje iz "životnog ciklusa" odredenog nosioca zapisa informacija o proizvodu. Objekt "prikaz proizvoda" sadrži operacije sucelja za transfer i generiranje podataka iz skupa informacija koji modelira.

Struktura proizvoda (prikaz strukture proizvoda):

Entitet "prikaz strukture proizvoda" enkapsulira zapise informacija i operacije vezane uz pojam konkretnog fizickog objekta. Pored informacija o proizvodu, u proizvodnom sustavu kao entiteti egzistiraju i dijelovi i sklopovi proizvoda kao stvari, "fizicki" objekti. Npr. razlikujemo "vratilo" i "crtež vratila".

Akcija:

Entitet "akcija" modelira pozive operacija objekata modela procesa konstruiranja. Akcije se kreiraju i planiraju prije izvodenja procesa, a realiziraju se u tijeku izvodenja procesa konstruiranja.

Sucelje programskog alata:

Sucelje programskog alata enkapsulira skup operacija za transfer podataka izmedu objekata modela procesa konstruiranja i programskih alata koji nisu dio objektnog modela procesa konstruiranja.

Zadatak:

Entitet "zadatak" modelira informacijske tokove i organizacijske aspekte realnog konstruktorskog zadatka u okruženju timskog rada.

Konstruktor:

Entitet "konstruktor" enkapsulira informacije i operacije vezane uz osobe iz konstrukcijskog tima, organizaciju i podjelu rada i odgovornosti, te hijerarhiju kontrole pristupa dokumentima koji su proizvod procesa konstruiranja. Skup instanci "konstruktora" modelira dogadaje i pojmove vezane za osobe i njihove interakcije kao "izvodace" procesa konstruiranja.

Mrežna topologija procesa konstruiranja:

Pojam "mrežna topologija procesa konstruiranja" u ovom radu podrazumijeva topologiju dvije kompleksne mreže relacija:

- složene višerazinske veze izmedu struktura inženjerskih podataka
- mreže relacija redoslijeda izvodenja etapa procesa, koja redovito ukljucuje i iterativne podprocese

Te dvije mreže relacija ne mogu se promatrati odvojeno, nego se i one medusobno prožimaju, tj. relacije jedne grupe mogu ukljucivati relacije druge grupe, kroz više razina referenciranja.

Izraz:

Izraz je sintakticka struktura kojom se formiraju zapisi provjere ogranicenja i pravila odlucivanja. Pojam izraz (expression) analogan je "izrazu" kao dijelu naredbe, kako se definira u gotovo svim programskim jezicima. Izraz se sastoji od operanada i operatora, gdje su operandi atributi objekata modela procesa konstruiranja ili numericke ili znakovne konstante.

Ogranicenje:

Interne varijable i parametri konstrukcije cine vektor nepoznanica konstrukcije
P = [p_1, p_2, p_3, …, p_n]. Funkcionalni zahtjevi cine vektor F = [F_1, F_2, F_3, …, F_n]. Relacije izmedu zahtjeva i nepoznanica mogu se izraziti kao:

f_i (F, P) = 0 $\quad i = 1, k$

g_j (F, P) $\leq G_j \quad j = 1, s$

Navedene jednadžbe i nejednadžbe u terminologiji znanosti o konstruiranju nazivaju se "ogranicenja".

Pravila odlucivanja:

Osnovni su elementi usmjeravanja tijeka izvodenja procesa konstruiranja u predloženom modelu. Pravila odlucivanja mogu se razmatrati i kao relacije izmedu akcija koje se izvode u procesu i stanja i vrijednosti atributa objekata. Pravila su koncipirana u obliku "IF *uvjet* THEN *akcija*" (ili premisa - akcija), pri cemu zadovoljenje uvjeta pokrece akciju. Struktura pravila može sadržavati jedan uvjet i više akcija. Kao uvjet pravila koristi se leksicki izraz, odnosno entitet "izraz". Sintaksa pravila:

```
pravilo ::= if '(' <izraz> ')' then
                  <niz_akcija>
            [ else
                  <niz_akcija> ]
            end if
```

Plan konstruiranja:

U predloženom modelu procesa konstruiranja definira se kao model dekompozicije i tijeka izvodenja procesa konstruiranja u kojem se organiziraju i predvidaju tokovi informacija i redoslijed izvršavanja akcija, te postavljaju kontrolni uvjeti.

Cvor plana konstruiranja:

U predloženom modelu, etapa *(cvor plana) procesa konstruiranja* definira se kao kombinacija nepraznog skupa akcija (naredbi) A_j koje transformiraju objekt O, koji se nalazi u stanju S^O_i , u slijedece (novo stanje) S^O_{i+1} :

$$E = \{ A_n : S_j (A_j (S^O_i) = S^O_{i+1} \}$$

Definicija se može proširiti i na skup objekata, odnosno skup akcija transformira stanje svakog od objekata u slijedece novo stanje. Pri tome akcije mogu i generirati nove objekte i nakon toga im mijenjati stanje.

ŽIVOTOPIS

Neven Pavkovic roden je 05. 07. 1962. u Zagrebu gdje je završio osnovnu i srednju školu (Matematicko-informaticki obrazovni centar). Fakultet strojarstva i brodogradnje upisao je 1981. godine. Diplomirao je 1987. godine, na usmjerenju "Strojarske konstrukcije". Za uspjeh u studiju nagraden je Medaljom FSB-a.

Od rujna 1987. do sijecnja 1988. radio je u poduzecu "Koncar - Transformatori" kao konstruktor u odjelu za konstrukciju alata i naprava. Od sijecnja 1988. do ožujka 1991. bio je zaposlen u poduzecu "Koncar - Srednji elektricni strojevi", u odjelu za razvoj konstrukcije. Radio je na poslovima razvoja programske podrške, primjene racunala u odjelu konstrukcije i proracuna cvrstoce dijelova elektromotora. U tom razdoblju objavio je dva strucna elaborata.

01. 03. 1991. zasnovao je radni odnos na FSB-u, kao asistent za znanstvenu disciplinu "Elementi strojeva i konstruiranje", znanstveno podrucje strojarstvo, u Zavodu za mehanicke konstrukcije, Katedra za elemente strojeva i konstruiranje. Kao asistent držao je nastavu iz predmeta "Primjena elektronickih racunala", "Uvod u racunala", "Informatika", "Konstruiranje pomocu racunala", "Primjena racunala" i "Metodicko konstruiranje".

Magistarski rad pod naslovom "Kreiranje baze scenarija ekspertnog CAD sustava" obranio je 17. 10. 1996.

Osim u nastavi, aktivno je sudjelovao i u organizaciji medunarodnih znanstvenih skupova DESIGN '98 i DESIGN 2000 održanih u svibnju 1998. i svibnju 2000. u Dubrovniku. Govori i piše engleski, a služi se i njemackim jezikom.

Kao koautor objavio je jedan znanstveni rad u casopisu, a kao autor ili koautor 20 znanstvenih radova u zbornicima, od toga 8 na medunarodnim znanstvenim skupovima održanim u inozemstvu.

BIOGRAPHY

Neven Pavkovic was born on July 5th 1962 in Zagreb, where he completed his primary and secondary education (mathematics highschool). In 1987 he graduated in mechanical design from the Faculty of Mechanical Engineering and Naval Architecture of the University of Zagreb. For the excellent success in study, he was awarded the Faculty Medal.

From September 1987 until January 1988 he worked in the "Koncar - Transformers" factory, as a designer in tools design department. From January 1988 until March 1991 he was employed in the "Koncar - Medium Size Electrical Machines" factory, as a research engineer in the design department. His line of work was the development of computer based design support tools and the mechanical stress calculations of electromotor parts. In that period he published two project report studies.

Since March 1st 1991 he has been an assistant at the Design Department of the Faculty of Mechanical Engineering and Naval Architecture of Zagreb University.

In 1996 Neven Pavkovic acquired the M. Sc. degree at the Faculty of Mechanical Engineering and Naval Architecture of Zagreb University with the thesis "Creating Database of Scenarios for Expert CAD System". He has a good command of English and can read German fairly well.

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

As a co-author he has published one paper in a scientific journal. As a first author or a co-author he has published 20 scientific papers in conference proceedings, 8 of which were international conferences being held abroad.