Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Ivan Vadla

Zagreb, 2009.

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Voditelji rada:
Prof. dr. sc. Neven Pavković
Doc. dr. sc. Nenad Bojčetić

Ivan Vadla

Zagreb, 2009.

Izjavljujem da sam ovaj rad izradio samostalno, služeći se stečenim znanjem i navedenom literaturom.

Zahvaljujem se voditeljima rada prof. dr. sc. Nevenu Pavkoviću i doc. dr. sc. Nenadu Bojčetiću na strpljenju, ustupljenoj literaturi i korisnim savjetima tijekom izrade rada.

Posebnu zahvaljujem svojoj obitelji na podršci i strpljenju tijekom školovanja.

Ivan Vadla

SAŽETAK

Cilj ovog rada je izraditi programski sustav koji će omogućiti prikazivanje i spremanje tijeka promišljanja i odlučivanja u procesu konstruiranja. Rad se sastoji od dva glavna dijela, razmatranja teorijskih osnova u prvom dijelu i izrade programskog sustava.

U uvodnom dijelu ukratko su razmotrene osnove razvoja proizvoda i konstruiranja, kao i razlozi za prikazivanje tijeka promišljanja u procesu konstruiranja.

Drugi dio, koji pokriva treće i četvrto poglavlje bavi se sustavima za upravljanje podacima o proizvodu, koji imaju za cilj olakšati i ubrzati procese razvoja i konstruiranja.

Nakon toga, u petom poglavlju postavljene su teoretske osnove za izradu adekvatnog programskog sustava, za koji nakon toga postavljeni zahtjevi.

Sedmo poglavlje objašnjava funkcije i svojstva izrađenog programskog sustava, te je nakon toga prikazan primjer njenog korištenja.

U završnom dijelu rada razmotrene su mogućnosti promjene i poboljšanja razvijenog programskog sustava, kao i mogućnosti njegovog povezivanja s ostalim elementima sustava upravljanja znanjem.

SADRŽAJ

POPIS SLIKA V

POPIS TABLICA VI

1 UVOD ..... - 1 -

2 RAZVOJ I KONSTRUIRANJE PROIZVODA ..... - 2 -

2.1 RAZVOJ PROIZVODA ..... - 2 -

2.2 KONSTRUIRANJE ..... - 5 -

3 PLM I PDM SUSTAVI ..... - 8 -

3.1 SUSTAVI ZA UPRAVLJANJE ŽIVOTNIM CIKLUSOM PROIZVODA (PLM) ..... - 8 -

3.1.1 PLM 2.0 ..... - 11 -

3.2 SUSTAVI ZA UPRAVLJANJE PODACIMA O PROIZVODU ..... - 11 -

3.2.1 Funkcije PDM sustava ..... - 13 -

3.2.2 Arhitektura PDM sustava ..... - 14 -

3.2.3 Integracija PDM i CAD sustava ..... - 16 -

4 PLM/PDM SUSTAV SMARTEAM ..... - 17 -

4.1 SMARTEAM V5R17 ..... - 17 -

4.1.1 Učitavanje SmarTeam sustava ..... - 19 -

4.1.2 SmarTeam Editor ..... - 19 -

4.1.3 Kreiranje korisničke strukture ..... - 21 -

4.1.4 Dodavanje nove vrste dokumenata ..... - 24 -

5 SUSTAVI PRIKAZIVANJA LOGIČKE PODRŠKE KONSTRUIRANJU ..... - 26 -

5.1 LOGIČKA PODRŠKA ..... - 26 -

5.2 SUSTAVI ZA LOGIČKU PODRŠKU ..... - 28 -

5.3 SHEME I MODELI ZA PRIKAZIVANJE LOGIČKE PODRŠKE KONSTRUIRANJU ..... - 31 -

5.4 IBIS SUSTAV ..... - 33 -

6 ZAHTJEVI ZA PROGRAMSKI SUSTAV ..... - 36 -

6.1 ELEMENTI GRAFIČKOG PRIKAZA ..... - 36 -

6.2 FUNKCIJE GRAFIČKOG PRIKAZA ..... - 38 -

6.3 POVEZIVANJE ELEMENATA ZAPISA SA VANJSKIM DATOTEKAMA ..... - 39 -

6.4 INDEKSIRANJE ELEMENATA GRAFIČKOG PRIKAZA U SVRHU PRETRAŽIVANJA ..... - 40 -

6.5 FUNKCIJE PROGRAMSKOG SUSTAVA ..... - 40 -

7 IZRAĐENI PROGRAMSKI SUSTAV ..... - 42 -

7.1 ALAT ZA IZRADU ..... - 42 -

7.2 POJAŠNJENJE DIJELOVA I FUNKCIJA PROGRAMSKOG SUSTAVA ..... - 43 -
7.2.1 Dijelovi sustava ..... - 43 -
7.2.2 Funkcije programskog sustava i metode izvođenja ..... - 49 -
8 PRIMJER KORIŠTENJA REALIZIRANOG SUSTAVA ..... - 56 -
9 MOGUĆNOSTI PROMJENE I UNAPRJEĐENJA PROGRAMSKOG SUSTAVA ... - 59 -
9.1 TRENUTNO STANJE PROGRAMSKOG SUSTAVA ..... - 59 -
9.2 MOGUĆNOSTI DALJNJEG RAZVOJA ..... - 60 -
9.3 RAZVOJ SAMOSTALNOG SUSTAVA ..... - 60 -
9.4 INTEGRIRANI IBIS SUSTAV ..... - 61 -
10 POVEZIVANJE REALIZIRANOG SUSTAVA S OSTALIM DIJELOVIMA SUSTAVA UPRAVLJANJA ZNANJEM ..... - 63 -
11 ZAKLJUČAK ..... - 64 -
12 LITERATURA ..... - 66 -
13 PRILOG

POPIS SLIKA

Slika 1. Faze u razvoju proizvoda, [1]
Slika 2. Utjecaj razvoja (konstruiranja) na povećanje troškova, [1]
Slika 3. Faze u procesu konstruiranja, [4]
Slika 4. Shema korištenja PLM-a, [8]
Slika 5. Životni ciklus proizvoda – zahtjev za PLM, [6]
Slika 6. Područja PDM-a
Slika 7. Arhitektura PDM sustava, [10]
Slika 8. Skupina učitanih komponenata
Slika 9. Korisničko sučelje SmarTeam Editor-a
Slika 10. Korisničko sučelje alata za manipulaciju korisnicima sustava
Slika 11. Prikaz korisnika i sučelja za promjenu uloga u korisničkih skupina
Slika 12. Prikaz ovlaštenja s obzirom na ulogu korisnika i klasu
Slika 13. Korisničko sučelje za definiranje postavki novih vrsta dokumenata
Slika 14. Definiranje postavki nove vrste dokumenata
Slika 15. Definiranje aplikacije za manipulaciju novim dokumentima
Slika 16. Generalna arhitektura sustava za logičku podršku konstruiranju, [18]
Slika 17. Shema osnovnih elemenata IBIS-a, [21]
Slika 18. Primjer grafa dobivenog korištenjem osnovnog IBIS-a, [21]
Slika 19. Shema prikaza gIBIS grafa, [22]
Slika 20. Shema funkcija na grafičkom prikazu
Slika 21. Funkcije i mogućnosti programskog sustava
Slika 22. Početno sučelje C# alata
Slika 23. Klase definirane u programskom sustavu
Slika 24. Glavna forma
Slika 25. Glavna forma s formom *Help* (legenda)
Slika 26. IBIS forma - dokument
Slika 27. *Problem* sa mogućim obilježjima
Slika 28. *Ideja* sa mogućim obiljžjima
Slika 29. *Argument* sa mogućim obilježjima
Slika 30. Primjer elementa *slika*
Slika 31. Primjer elementa *vanjska poveznica*
Slika 32. Pregled svih korištenih klasa i metoda
Slika 33. Primjer dodanih elemenata
Slika 34. Gotov IBIS dokument za odabrani primjer
Slika 35. IBIS dijagram pokrenuti iz SmarTeam PDM sustava

Slika 36. Shema odnosa IBIS i PDM sustava
Slika 37. Samostalni programski sustav IBIS
Slika 38. IBIS sustav integriran u PDM sustav

POPIS TABLICA

Tablica 1. Zahtjevi komponenata SmarTeam sustava, [13]
Tablica 2. Popis administratorskih alata SmarTeam-a, [14]
Tablica 3. Usporedba najvažnijih shema prikazivanja logičke podrške, [19]
Tablica 4. Grafičke oznake elemenata prikaza

1 UVOD

U današnje vrijeme, kada je svaki novi proizvod na iskušenju, i zapravo na rubu propasti, vrlo je važno smanjiti troškove proizvodnje. Oni se jednostavno mogu smanjiti degradacijom kvalitete proizvoda. No, ukoliko to ne želimo, moguće je okrenuti se prema primjeni novih znanja i spoznaja, koje omogućuju bolje iskorištavanje resursa tvrtke, smanjuju vrijeme plasiranja proizvoda na tržište i čak poboljšavaju kvalitetu proizvoda. Nova znanja mogu uključivati nove materijale, nove tehnologije, ali mogu uključivati i nove načine i metode rada.

Kao što je već mnogim inženjerima poznato, prilikom razvoja novih proizvoda, jedan od generatora potencijalno velikih troškova je upravo proces konstruiranja, ili, proces razvoja proizvoda u najužem smislu, koji su dalje u radu ukratko pojašnjeni. Upravo zbog tog smanjenja troškova, skraćivanja vremena razvoja proizvoda, pa i kreiranja boljih i kvalitetnijih proizvoda, bitno je na neki način bilježiti i prikazivati proces promišljanja i odlučivanja u procesu konstruiranja. Bilježenjem tijeka promišljanja olakšalo bi se članovima konstrukcijskih timova praćenje novonastalih problema, ili načina njihovih rješavanja. U procesu konstruiranja mnoge se informacije mogu izgubiti neposredno nakon dobivanja odgovarajućeg rješenja. Kako bi se to izbjeglo, te kako bi barem dio promišljanja članova konstrukcijskog tima ostao zabilježen, potrebno je na neki način bilježiti, nastale probleme, ideje za njihovo rješavanje, te ostale argumente koji proizađu prilikom rada u konstrukcijskom timu i osmišljavanja rješenja za nastale probleme. Za tu svrhu razvijene su metode i sustavi podrške konstruiranju. Neke od metoda su kasnije obrađene u radu, te se na jednoj od njih bazira i cilj ovog rada, dokument u kojem ostaje zabilježen tijek promišljanja i odlučivanja konstruktora.

Budući da upotreba samo jedne vrste dokumenata, izdvojene od ostatka baze znanja u današnje vrijeme na bi bila od velikog značaja, potrebno ga je koristiti i integrirati u neki od sustava za upravljanje životnim ciklusom proizvoda, ili sustava za upravljanje podacima o proizvodu. Na taj način, tijek promišljanja i odlučivanja efikasno bi se pohranio unutar tvrtke, ili institucije za koju je relevantan i bilo bi osigurano jednostavno dijeljenje i preuzimanje postojećeg znanja pohranjenog u dokumentu.

2 RAZVOJ I KONSTRUIRANJE PROIZVODA

2.1 RAZVOJ PROIZVODA

Razvoj proizvoda vrlo je širok pojam, on ne obuhvaća samo dio vezan uz osmišljavanje pojedinog proizvoda, već sadrži više faza u kojima se prepoznaju potrebe potencijalnih kupaca, stanje sličnih proizvoda na tržištu, te prepoznaju problemi i zahtjevi izrade.

U širem smislu, razvoj proizvoda bi se mogao opisati kao skup aktivnosti koje imaju svoj početak u inicijalnoj ideji o nekom proizvodu, nakon čega slijedi istraživanje tržišta, osmišljavanje koncepata, odabir prikladne varijante proizvoda, izrada prototipova ukoliko je potrebna, te tek nakon toga slijedi proizvodnja i prodaja proizvoda.

Prema [1] razvoj proizvoda u užem smislu je zapravo niz koraka kojima se skup ulaza transformira u skup izlaza. Iako se ne doima tako, u tim koracima mnogo su češći koraci misaone, ili organizacijske prirode, nego fizičke prirode. Ovdje su uključene sve aktivnosti potrebne za izradu proizvoda na temelju ranije definiranih zahtjeva, kao što su razrada početnih ideja, određivanje principa rada, detaljne razrade rješenja, pa sve do priprema za proizvodnju, [2].

Životni ciklus proizvoda, kao što se to dade naslutiti iz ranijeg pojašnjenja pojma razvoja, sastoji se od više faza, koje se također mogu prikazati u širem i užem smislu. U širem smislu razvoja proizvoda, prema [3] faze su:

1. *Generiranje ideje:* Za ovu fazu bi se moglo reći *„dobra ideja zlata vrijedi“*. Ideje za novi proizvod često se dobivaju istraživanjem tržišta, trendova kod potrošača, istraživanjem konkurencije, ciljanih skupina korisnika i sl. Nakon ovakvih istraživanja, najčešće slijedi tzv. „*brainstorming*“, budući da se u današnjem društvu i globalnoj ekonomiji rijetko prije toga dođe do isplative ideje za novi proizvod.

2. *Razmatranje ideje:* Nakon što se pronađe potencijalno isplativa ideja za proizvod, potrebno je razmotriti njegovu stvarnu isplativost, prije nego što se resursi tvrtke usmjere prema njenom ostvarivanju. U ovoj fazi potrebno si je postavljati pitanja kao što su: *Hoće li korisnik profitirati na neki način korištenjem proizvoda? ; Kakve su naznake rasta tržišta ciljane skupine korisnika? ; Da li konkurenti imaju bolje, ili lošije slične proizvode? ; Da li je tehnički izvediva i isplativa proizvodnja?*

3. *Razvijanje koncepata:* U ovoj fazi izrađuju se koncepti i osmišljavaju varijante proizvoda. Također, detaljiraju se podaci potrebni za izradu proizvoda. Resursi tvrtke su već dodijeljeni za razvoj novog proizvoda, te nakon ove faze, eventualni neuspjeh proizvoda stvarao bi vrlo velike gubitke.

4. *Poslovna analiza:* Analiziraju se moguće cijene proizvoda, s obzirom na kvalitetu proizvoda i konkurenciju. Također se analizira moguća dobit i količina prodaje proizvoda.

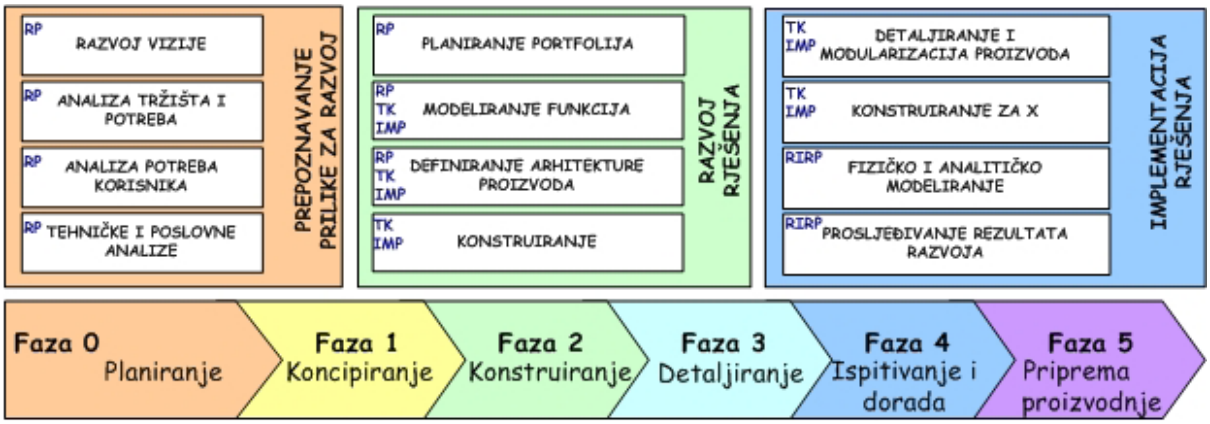
5. *Testiranje:* Nakon izrađenog prototipa vrši se testiranje zahtijevanih funkcija proizvoda. Također, moguće je primijeniti testiranje sa odabranom grupom korisnika, kako bi se zabilježila i njihova zapažanja i mišljenja o proizvodu. Ukoliko je potrebno, u ovoj fazi moguće je napraviti preinake na proizvodu tamo gdje je to potrebno.

6. *Proizvodnja:* Ukoliko je sve dobro prošlo u fazi testiranja, kreće se u punu proizvodnju, budući da bi u ovoj fazi uspjeh proizvoda trebao biti gotovo zagarantiran (barem pokriće troškova).

7. *Prodaja i plasiranje na tržište:* Završna faza razvoja proizvoda je zapravo njegovo plasiranje na tržište. U nju pripada i oglašavanje i osiguravanje distribucijske mreže do ciljanih tržišnih skupina.

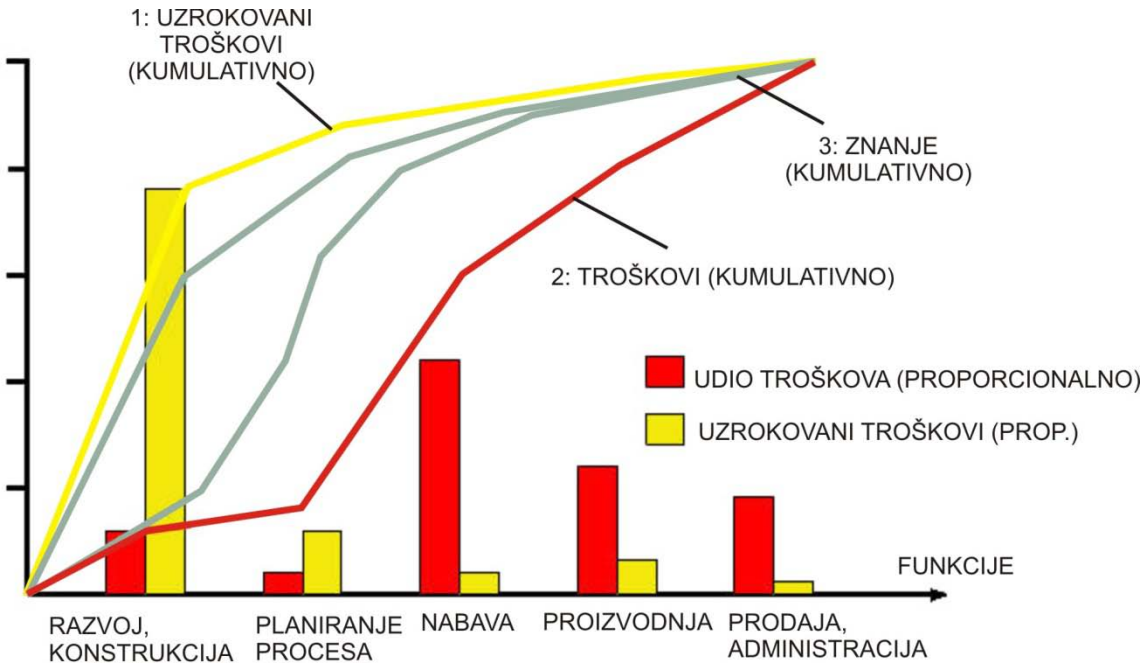
Iako u ovih sedam faza nije navedeno, nakon plasiranja proizvoda na tržište, tvrtka se mora pobrinuti i za održavanje proizvoda, te zbrinjavanje, čime se zapravo završava životni ciklus proizvoda.

U užem smislu, faze razvoja proizvoda prema [1], su: planiranje, koncipiranje, konstruiranje, detaljiranje, ispitivanje i dorada, te priprema proizvodnje, što je ilustrirano na slici 1..



Slika 1. Faze u razvoju proizvoda, [1]

Od svih navedenih faza razvoja proizvoda, razvoj se može također shvatiti u najužem smisli kao „Faza 2“ na slici ranije, tj. konstruiranje. Odluke u fazi konstruiranja utječu na sve ostale faze životnog ciklusa proizvoda, tj. razvoja, te imaju velik utjecaj na troškove u proizvodnji posebice novog proizvoda, što je ilustrirano na slici 2., prema [1].



Slika 2. Utjecaj razvoja (konstruiranja) na povećanje troškova, [1]

Kao što je iz slike 2 vidljivo, faza razvoja (konstruiranja) može znatno povisiti ukupne troškove proizvoda, a dokumenti za praćenje tijeka promišljanja i odlučivanja, koji su cilj ovog rada, u tom smislu su od velikog značaja. Njihov cilj je olakšati i ubrzati fazu konstruiranja, što za uzrok ima smanjeno vrijeme razvoja, te u konačnici veći profit. U nastavku je pobliže pojašnjeno što je to konstruiranje, kao faza razvoja proizvoda.

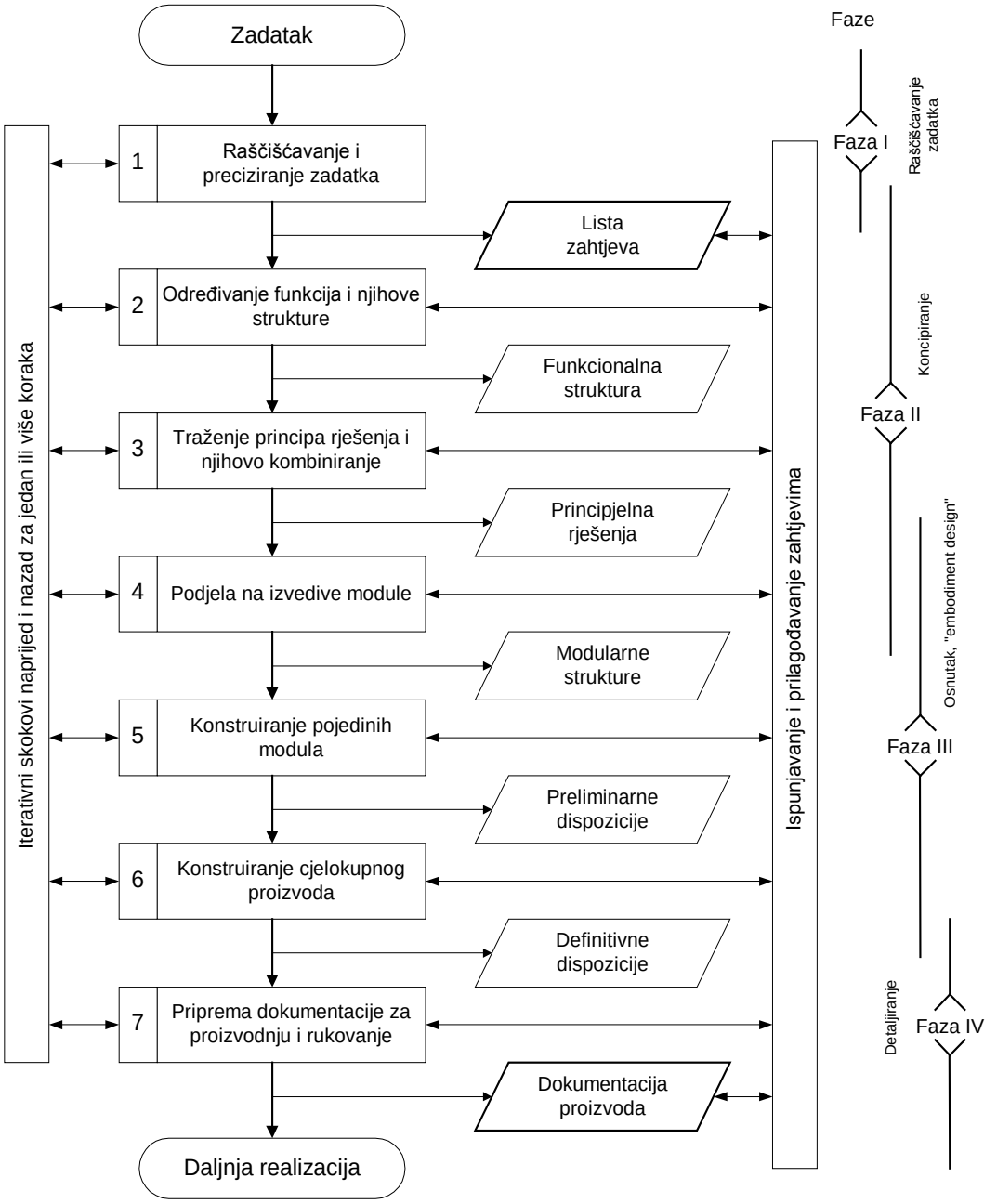
2.2 KONSTRUIRANJE

Ukratko,prema [1], konstruiranje se može sažeti kao skup procesa i aktivnosti potrebnih da se donese odluka o izgledu i obliku proizvoda, s obzirom na njegovu funkciju. No, konstruiranje je mnogo kompliciraniji proces nego što je to prethodno prikazano.

Konstruiranje se tako može opisati i kao pretvorba unaprijed postavljenih zahtjeva u konkretno, tehnološki izvedivo rješenje. Potrebno je potpuno definirati i opisati proizvod, koji će imati zadana svojstva, zadovoljavati postavljene želje i ispunjavati željene funkcije.

U prvom redu, konstruiranje je prema [4], proces prerade i generiranja informacija koje na temelju ulaznih zahtjeva definiraju i opisuju proizvod. Konstrukcijski proces često je sinteza od prije poznatih elemenata u novu cjelinu sa novim određenim svojstvima, što zahtjeva od konstruktora kreativan i snalažljiv rad. No, konstruiranje nije uvijek sinteza postojećih, poznatih elemenata, već je i osmišljavanje potpuno novih elemenata, za potrebe razvoja novih proizvoda, što zahtjeva veliko iskustvo i znanje konstruktora, kako bi se što brže mogao prilagoditi novoj situaciji. Zbog ovakvih situacija, konstruiranje je gotovo uvijek konstantni proces učenja.

Kako se gotovo svaki konstrukcijski zadatak može riješiti na više načina, usklađivanje zahtjeva za željenim svojstvima i oblikom proizvoda može se pokazati kao dugotrajan i zahtjevan proces. Tako se svaki proces konstruiranja može prema [4] raščlaniti na više faza, koji se iterativno ponavljaju, ukoliko se javi potreba, kao što je prikazano na slici 3.



Slika 3. Faze u procesu konstruiranja, [4]

Kako bi se konstrukcijski problemi i zadaci mogli uspješno riješiti, poželjno je voditi se postavkama prema [5], a one su:

1. Naučiti konstruirati se može samo konstruiranjem.
2. Za rješavanje problema, konstruktor upotrebljava tri vrste znanja:
    - za generiranje ideja
    - za procjenu i prosuđivanje
    - za strukturiranje procesa konstruiranja

3. Uz dovoljno iskustva za generiranje i prosuđivanje ideja, moguće je konstruirati kvalitetne proizvode.
4. Konstruirati je potrebno učiti u akademskom i industrijskom okruženju.

U svakoj fazi prikazanoj na slici 3 moguće je bilježiti tijek promišljanja i odlučivanja, čime se eventualne pogreške i loša rješenja mnogo brže prepoznaju, te se tako i problemi brže rješavaju. Ovakav zabilježen tijek promišljanja ostaje na ponovno korištenje u slučajevima izrade sličnog proizvoda, ili za učenje novih članova konstrukcijskog tima.

Budući da su vrste konstruiranja prema [6]:

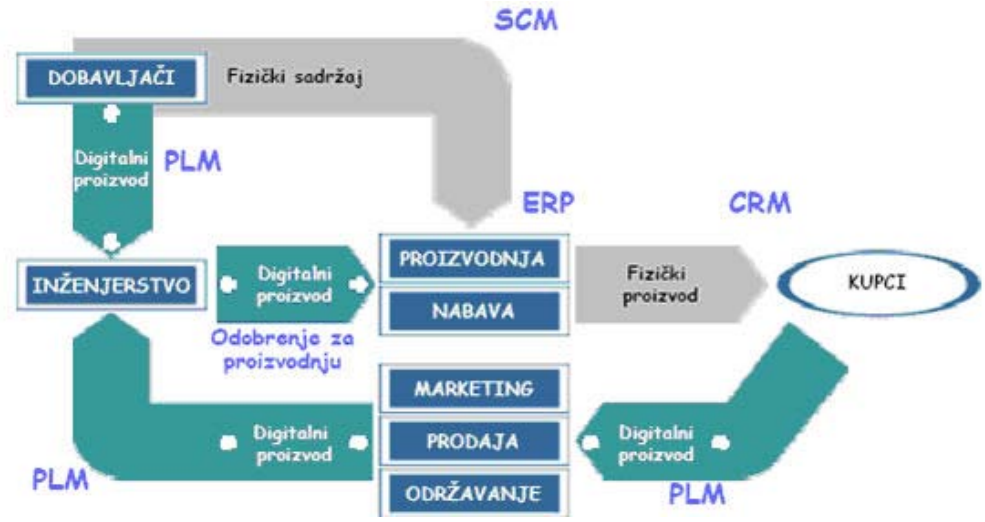
- *Izvorno konstruiranje* – novi koncept koji nikada nije postojao
- *Ponovljeno konstruiranje* – biranje između postojećih normiranih dijelova
- *Rekonstruiranje* – mijenjanje oblika
- *Varijantno konstruiranje* – mijenjanje postojećeg dijela, ali zadržavanje izvornog koncepta
- *Prilagođeno konstruiranje* – prilagodba poznatog rješenja za novu zadaću,

korištenje zapisanog tijeka promišljanja prijašnjih procesa konstruiranja moglo bi se pokazati od velike koristi.

3 PLM I PDM SUSTAVI

3.1 SUSTAVI ZA UPRAVLJANJE ŽIVOTNIM CIKLUSOM PROIZVODA (PLM)

Upravljanje životnim ciklusom proizvoda (nadalje PLM[1]) je proces upravljanja cijelim životnim vijekom proizvoda, od njegovog koncepta, preko dizajna i proizvodnje, sve do korištenja i uklanjanja. PLM predstavlja skup aktivnosti koje omogućuju tvrtkama efikasnu i efektivnu inovativnost, te upravljanje čitavim nizom usluga koje su vezane za pojedine faze životnog ciklusa proizvoda. PLM je uobičajeno jedan od temelja informacijske infrastrukture tvrtke, uz CRM[2] (Customer Relationship Managment), SCM[3] (Supply Chain Managment), i ERP[4] (Enterprise Resource Planning). CRM omogućuje komunikaciju i razmjenu informacija sa kupcima, SCM omogućuje isto sa dobavljačima, a ERP pomaže u planiranju i raspoređivanju resursa unutar tvrtke ili kompanije. Dodatno, tvrtke i kompanije koje među svoje djelatnosti imaju uključenu i proizvodnju, moraju također razviti, opisati, definirati, upravljati i razmjenjivati informacije o svojim proizvodima, a to je upravo ono što PLM kao informacijska tehnologija modelira. Na slici 4 prikazana je shema korištenja PLM-a u terminima virtualno-stvarno. [7]



Slika 4.Shema korištenja PLM-a, [8]

[1] PLM (eng. Product Lifecycle management) = upravljanje životnim vijekom proizvoda

[2] CRM (eng. Customer Relationship Management = upravljanje distribucijom

[3] SCM (eng. Supply Chain Management) = upravljanje lancem nabave

[4] ERP (Enterprise Resource Planning) = upravljanje resursima tvrtke (poduzeća)

Glavni ciljevi korištenja PLM-a su:

- Kreiranje strateških izvora podataka o proizvodima i procesima
  - Aktivno uključivanje i integracija svih čimbenika razvoja
  - Prikupljanje i dijeljenje informacija o proizvodu duž cjelokupnog vrijednosnog lanca
- Omogućavanje suradnje s kupcima, dobavljačima i partnerima
  - Podrška inovativnosti dobavljača i partnera
  - Podrška uključivanju korisnika u razvoj
- Unaprjeđenje fleksibilnosti proizvoda i procesa
  - Interakcija s novim kupcima i stvaranje novih prilika za pružanje dodatnih usluga
  - Podrška distribuiranom pristupu razvoja i proizvodnji
  - Omogućavanje brzog dodavanja ili zamjene dobavljača ili partnera

Suština PLM-a je zapravo stvaranje centralnog sustava upravljanja svim podacima vezanim uz proizvode i tehnologije, koji se koriste za pristup informacijama i znanju. PLM je disciplina nastala iz alata kao što su CAD[5]/CAM[6] i PDM[7], ali se može promatrati kao integracija ovih alata s metodama, ljudima i procesima kroz sve etape životnog vijeka proizvoda. Također, kod PLM-a nije bitna samo tehnologija softvera, nego i poslovna strategija. [7]

Kako bi PLM sustav bio u potpunosti funkcionalan trebao bi biti u mogućnosti pohranjivati i upravljati podacima za:

- Osmišljavanje proizvoda
  - Specifikacije i zahtjevi za proizvod
  - Konceptualni dizajn
- Konstruiranje proizvoda
  - Detaljna tehnička razrada
  - Provjera izvedivosti i analize (simulacije)
  - Alati za izradu
- Ostvarivanje proizvoda
  - Planiranje proizvodnje
  - Izvođenje proizvodnje
  - Sklapanje i montaža

[5] CAD (eng. Computer Aided Desing) = konstruiranje pomoću računala

[6] CAM (eng. Computer Aided Manufacturing) = proizvodnja pomoću računala

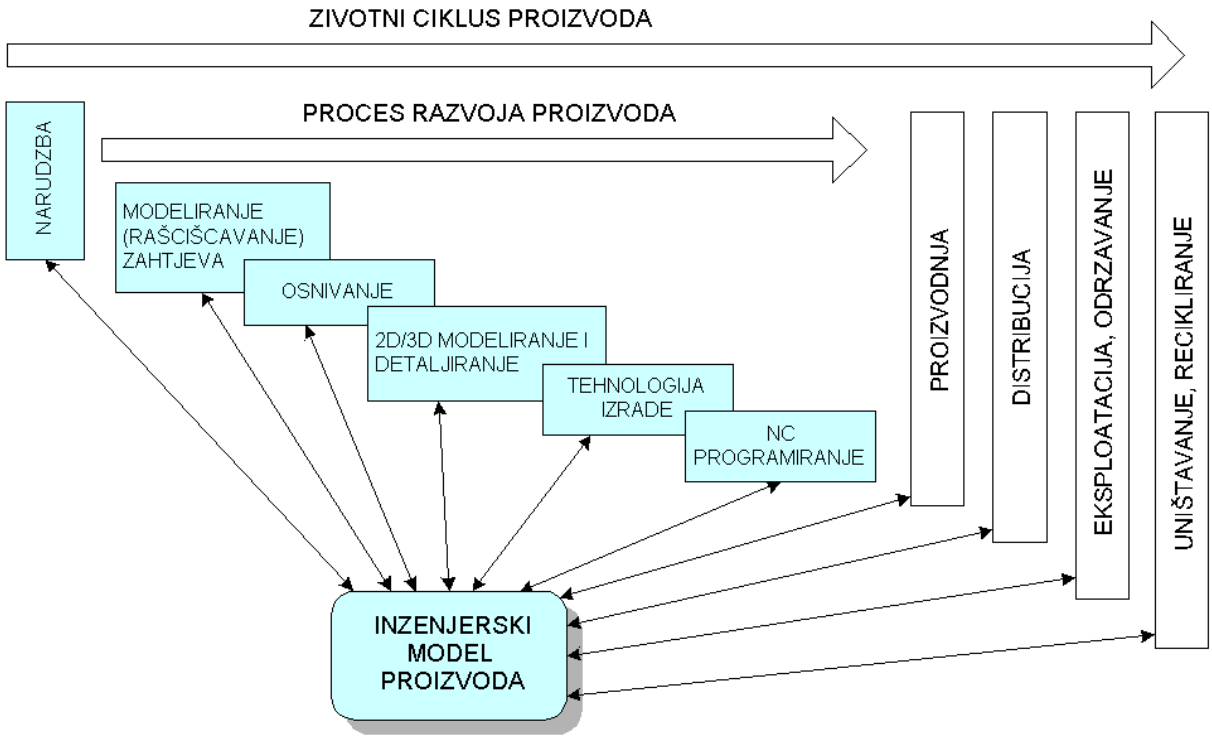
[7] PDM (eng. Product Data Managemenr) = upravljanje podacima o proizvodu

- Provjera kvalitete
- Usluge vezane uz proizvod
  - Prodaja i distribucija
  - Upotreba i korištenje
  - Održavanje i servisiranje
  - Odlaganje i recikliranje

Za slučaj razvoja proizvoda, primjenom PLM-a se prema [7], želi:

- Smanjiti vrijeme isporuke i plasiranja proizvoda na tržište
- Povećati kvalitetu proizvoda
- Smanjiti troškove izrade prototipova i testiranja proizvoda
- Stvoriti uštede služeći se podacima iz prethodnih faza razvoja proizvoda, ali i originalnih podataka
- Optimirati proces izrade i razvoja proizvoda
- Smanjiti otpad u proizvodnji
- Uštedjeti na materijalu, vremenu i toku informacija

Ukratko, kako bi PLM sustav ispunjavao u potpunosti svoju namjenu, trebao bi pokrivati sva područja prikazana na slici 5.[6]



Slika 5.Životni ciklus proizvoda – zahtjev za PLM, [6]

PLM sustav, kao softversko rješenje obuhvaća četiri glavna područja:

1. Upravljanje proizvodom i portfoliom proizvoda
2. Konstruiranje proizvoda (CAD-CAx)
3. Upravljanje procesima proizvodnje
4. Upravljanje podacima o proizvodu (PDM)

PLM sustavi u današnje vrijeme još su uvijek više filozofija poslovanja i upravljanja podacima o proizvodu i razvoju proizvoda, nego što je egzaktni informacijski sustav. Današnji vodeći proizvođači PLM sustava kao što su Dassault Systemes, SofTech i PTC, još uvijek nisu u mogućnosti u potpunosti pokriti sva područja koja ideja PLM-a pretpostavlja. Većina softverskih rješenja su ipak na razini PDM sustava, no, svakim danom i oni se unaprjeđuju i približavaju ideji PLM-a.

3.1.1 PLM 2.0

Godine 2008. Dassault Systemes je predstavio novi koncept PLM-a, PLM 2.0. Ideja ovog koncepta zasniva se na razvoju Internet aplikacija, koje slijede koncept *„Web 2.0“*. Za slučaj PLM-a 2.0, prema [7], PLM sustavi se:

- Baziraju na upotrebi interneta (*eng.* w*eb-based*).
- Fokusiraju na komunikaciju (korisnik- korisnik, sustav- korisnik) preko interneta, skupnu inteligenciju (znanje), te Internet zajednice.
- Proširuju izvan dosega tvrtke koja ih koristi.
- Mogu pokrenuti procedure sustava preko Interneta.

Slijedeći ovaj novi koncept, PLM sustavi kao filozofija poslovanja i upravljanja životnim vijekom proizvoda, približavaju se početnoj ideji PLM-a, te je za vjerovati kako će i ostali proizvođači PLM sustava slijediti ovaj koncept.

3.2 SUSTAVI ZA UPRAVLJANJE PODACIMA O PROIZVODU

Upravljanje podacima o proizvodu (nadalje PDM) bi prema [9], bilo korištenje softverskih rješenja i drugih alata za pronalazak i manipulaciju podacima vezanim uz proizvode. Ti podaci su uobičajeno tehničke specifikacije proizvoda, specifikacije vezane uz proizvodnju i razvoj, te vrste materijala potrebnih za izradu proizvoda. Navedene specifikacije pohranjene su u primjerenim datotekama (kao što su CAD datoteke). PDM sustavi mogu se shvatiti kao dio šireg PLM sustava, koji bi trebao brinuti o cijelom životnom vijeku proizvoda, dok se PDM sustav koncentrira više na većinski inženjerske

djelatnosti. PDM sustavi koriste se za upravljanje i manipulaciju podacima vezanim uz proizvode i proizvodne procese.

Sustavi za upravljanje podacima o proizvodu (nadalje PDM sustavi) koriste se kao središnje spremište informacija vezanih uz procese i proizvode, te potpomažu u razmijeni podataka između svih korisnika sustava. U sustavu se bilježe podaci kao što su vlasnik datoteke (osoba koja je kreirala datoteku), verzije datoteke,te omogućuje se kontrola pristupa datotekama spremljenim u centralnu bazu. Tipične vrste informacija koje se spremaju u PDM sustav su: oznaka dijela, opis dijela, dobavljač ili proizvođač, mjerne jedinice, cijena i troškovi, tehnički crteži, popisi materijala. [9]

Prema [10], PDM je alat koji pomaže inženjerima i drugima uključenim u razvoj proizvoda u upravljanju podacima i proizvodnim procesima vezanim za proizvod. Ovi sustavi trebali bi pohranjivati i omogućiti razmjenu velikog broja podataka i informacija vezanih za konstruiranje, proizvodnju.

Opseg PDM-a obuhvaća u pravilu nekoliko područja, koja su uglavnom vezana uz inženjersku djelatnost, a prikazana su prema [10], na slici 6.

Slika 6. Područja PDM-a

Neki od sustava, koji se prema [10] pretežno koriste u ovom dijelu Europe su:

- PTC: Windchill PDMLink i PRO/Intralink, od čijeg cjelokupnog sustava je dio i CAD/CAM/CAE sustav Pro/ENGINEER Wildfire.
- Dassault Systemes: Enovia Solutions i SmarTeam – u vlasništvu IBM-a, PDM sustavi se integriraju u vrlo poznati CAD/CAM sustav Catia, no također vrlo dobro funkcioniraju i samostalni, budući da podržavaju i konkurentne CAD sustave, kao što je Pro/ENGINEER

- Autodesk: Vault i Streamline – iako Autodesk zapravo i nije trenutno previše prisutan na tržištu PDM sustava, vrlo je poznat po svojim sustavima široke namjene kao što su AutoCAD, Mechanical Desktop i 3DStudio, što je naznaka kako će se i oni sve više probijati i okupirati tržište PDM sustava.

3.2.1 FUNKCIJE PDM SUSTAVA

Svaki PDM sustav posjeduje različite funkcije, no svi oni imaju barem tri skupine funkcija, a to su prema [10]:

- Funkcije trezora podataka
- Korisničke funkcije
- Pomoćne funkcije

U navedene funkcije pripadaju funkcije kao što su:, trezor podataka, upravljanje razvojnim procesom, upravljanje strukturom proizvoda, klasifikacija komponenti, pregledavanje i odobravanje, upravljanje konfiguracijom i promjenama, upravljanje projektima, te upravljanje tokovima procesa.

Trezor podataka i kontrola pristupa– kao što je i prije navedeno, osnovna funkcija PDM sustava je da služi kao elektronički trezor podataka (primarno o proizvodu). U trezoru se podaci pohranjuju i dostavljaju korisniku uz kontroliran pristup. Trezori moraju biti u mogućnosti spremiti velike količine različitih podataka, a ukoliko PDM sustav podržava više vrsta podataka, više različitih informacija moguće je u njega spremiti.

U trezor se prema [10], pohranjuju dva osnovna tipa podataka:

- Podaci o proizvodu poput računalnih modela, tehničke dokumentacije, analitičkih podataka, korisničkih uputa, i sl.
- Podaci o podacima, koji zapravo pobliže opisuju podatke o datotekama upravljanim u PDM sustavu. To su dodatne informacije o proizvodu, koji nisu sadržane u drugim datotekama. Ovakvim podacima također se može unaprijediti pretraživanje trezora podataka, budući da informacije o proizvodu mogu biti opisne i proizvoljne, te se mogu adekvatnim opisivanjem stvoriti grupe sličnih podataka.

Upravljanje razvojnim procesom – pomaže u kontroli procedura za upravljanje podacima vezanim uz proizvod, te osigurava mehanizme potrebne u interakciji sustava i korisnika. Omogućuje izvršavanje promjena u konfiguraciji proizvoda, definiranju dijelova, te promjeni verzija dokumenata, tj. kontrolira i prati promjene na dokumentima,

te ih upisuje i sprema kao druge inačice prijašnjih dokumenata. Također, ova funkcija prati i status dokumenata, te ga čini nedostupnim ukoliko se dokument trenutno mijenja od strane nekog korisnika sustava. [10]

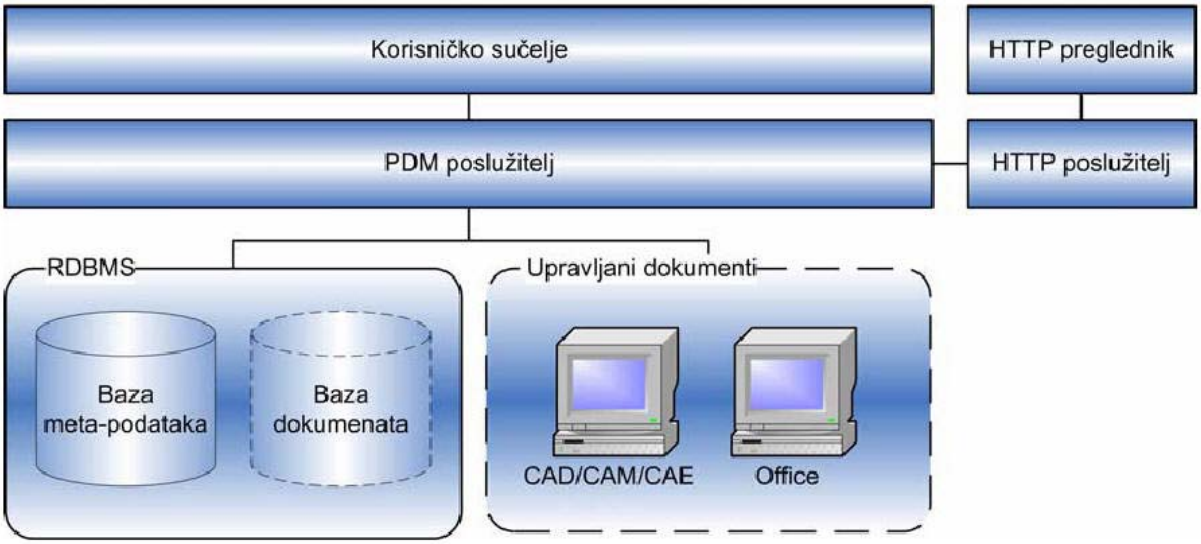
Upravljanje strukturom proizvoda – funkcija PDM sustava nije da primarno generira strukturu proizvoda. Za to su zaduženi specijalizirani CAD sustavi pomoću kojih se proizvodi konstruiraju (tj. izrađuju se 3D virtualni modeli) temeljeno na značajkama, kao što su npr. provrt, skošenje, i sl. Spomenuti CAD sustavi sami podržavaju hijerarhijski prikaz značajki, konstrukcije geometrije i dijelova sklopa, te se one zapravo „uvoze“ iz CAD sustava u PDM sustav. Osim hijerarhijskog prikaza pomoću stabla, kakav se koristi u CAD sustavima, također se u PDM sustavu primjenjuje i tablični prikaz strukture proizvoda, gdje su u tablici navedeni barem ime pojedinog elementa, njegova identifikacijska oznaka, te količina.[10]

Klasifikacija komponenti – PDM sustavi većinom nude već pripremljene procedure za razvrstavanje standardnih, kao i vlastitih kreiranih komponenti. Prilikom isporuke PDM sustava korisniku (tvrtci), katalozi komponenti su prazni, te se od korisnika očekuje njihovo popunjavanje, što se čini kao dodatan posao prilikom implementacije PDM sustava, no svakako je najbolje za korisnika da sustave kategorizira na sebi poznat način.

Ostale funkcije – osim osnovnih funkcija, PDM sustavi većinom nude i mnoge dodatne funkcije. Od bitnijih funkcija koje su sastavni dio PDM sustava, uključene su funkcije za komunikaciju, prijenos podataka i administraciju sustava. Kako PDM sustavi teže vrsti komunikacije preko globalne mreže, sve više omogućuju pristup sustavu preko Interneta, što omogućava korisniku da pregledava dokumente sa udaljenih lokacija, bez posjedovanja klijenta PDM sustava. Još jedna bitna funkcija PDM sustava je podržavanje više različitih vrsta CAD dokumenata, što omogućava korisniku koji ne posjeduje prikladan CAD sustav da pregledava i donekle mijenja određeni CAD dokument.

3.2.2 ARHITEKTURA PDM SUSTAVA

Prema [10], PDM sustavi su izvedeni na dvoslojnoj arhitekturi tipa klijent-poslužitelj, s težnjom prema većem raslojavanju, a slika 7., [10] prikazuje primjer takve arhitekture.



Slika 7. Arhitektura PDM sustava, [10]

Kao što je na slici 7 prikazano, PDM poslužitelj koristi podatke spremljene u bazu, koja je relacijska baza podataka, a također i nove podatke koji se dodaju u PDM sustav sprema u istu bazu. Neki od poznatih i robusnijih tehnologija za izradu relacijskih baza podataka su Oracle, MS SQL Server i IBM DB2. No, osim ovih, sve više dobivaju na popularnosti i besplatne tehnologije kao što su MySQL i PostgreSQL, koje ipak nisu uvijek jednako moćne kao ranije navedene komercijalne tehnologije.

U bazu dokumenata i podataka se najčešće binarno pohranjuju podaci i dokumenti stvoreni u CAD sustavima, ili u uredskim aplikacijama.[10] No, ideja ovog rada je da se u PDM sustave spremaju dokumenti koji prate i bilježe tijek promišljanja i odlučivanja prilikom procesa konstruiranja, kako bi se nadogradile mogućnosti PDM sustava i pružilo korisnicima detaljniji uvid u proces konstruiranja i razloge koji leže iza pojedinog izvedenog rješenja. U bazi meta-podataka čuvaju se ranije spomenuti podaci o podacima, koji uključuju putanje do dokumenata, kao što će biti slučaj za dokumente čija izrada je cilj ovog rada.

Svi dokumenti vezani uz projekte pohranjene u PDM sustavu (tj. u bazi podataka PDM sustava), trebali bi biti spremani u bazu, kako bi bili dostupni svim relevantnim korisnicima, te da ne dolazi do miješanja verzija i sadržaja dokumenata prilikom promjena, ukoliko korisnici drže podatke izvan dosega PDM sustava.

3.2.3 INTEGRACIJA PDM I CAD SUSTAVA

Prema [10] se navodi da se integracija PDM i CAD sustava ostvaruje na dva načina, tj. kao integracija podataka, te integracija korisničkog sučelja. Obje vrste integracije se također ostvaruju na nekoliko načina.

Integracija podataka, [10]:

- preko ručnog unosa podataka
- razmjenom podataka putem datoteka
- posebne baze podataka, koje prate unose datoteka
- zajednička, dijeljena baza podataka

Integracija korisničkog sučelja, [10]:

- PDM sustav je zadužen za pokretanje CAD sustava i povezivanje CAD datoteka
- u CAD sustavu postoje neke funkcije PDM sustava
- postoji zasebno sučelje koje nije ni sučelje PDM, ni CAD sustava
- izvedeno je potpuno integrirano korisničko sučelje

S obzirom na ostvarenu integraciju PDM i CAD sustava, prema [10], PDM sustavi su:

Začahureni – PDM sustav je samostalna aplikacija koja ostvaruje vrlo nisku razinu integracije sa bilo kojim drugim aplikacijama. U ovakvoj izvedbi nema prijenosa podataka o podacima između PDM baze i ostalih aplikacija, no postoji mogućnost pretraživanja i pokretanja CAD aplikacija, te spremanja u PDM bazu, prilikom čega korisnik mora dodati podatke o podacima.

Sučeljeni – ovakav PDM sustav zajedno sa CAD sustavom razmjenjuje podatke bez posredovanja korisnika, a PDM sučelje se nalazi unutar CAD sustava, čime je ostvarena jednosmjerna komunikacija ova dva sustava. Procedure PDM sustava se pozivaju iz izbornika CAD sustava.

Integrirani – PDM sustav je potpuno integriran svojim sučeljem u CAD sustav, te su sve procedure i funkcije PDM sustava dostupne unutar CAD sustava. Kod ovakvog PDM sustava prisutna je automatska razmjena podataka između dva sustava.

Hibridni – većina PDM sustava su zapravo hibridni sustav koji se najčešće u potpunosti integriraju u „svoj“ CAD sustav, a za druge CAD sustave se primjenjuje začahureni, ili sučeljeni oblik, ili njihova kombinacija. Takav je u sustav SmarTeam, tvrtke Dassault Systems korišten u primjeru ovog rada.

4 PLM/PDM SUSTAV SMARTEAM

SmarTeam sustav je u početku bio proizvod nezavisne tvrtke „SmarTeam“, koja je osnovana 1995. godine. No, godine 2006. SmarTeam je postao dio Dassault Systems, te je tako i PDM sustav postao dio Dassault-ova proizvoda ENOVIA. ENOVIA SmarTeam proizvodi većinom za ciljanu skupinu korisnika imaju manje i srednje tvrtke, što je razlika prema ostalim proizvođačima PLM sustava, koji većinom ciljaju na velike korporacije (kojima se više isplati uvođenje PLM/PDM sustava). [11]Trenutna inačica sustava ENOVIA je V6, koja pokušava ispuniti zamisli PLM 2.0 koncepta. No, u radu je kao PDM sustav korišten SmarTeam V5R17, koji će biti pobliže opisan u nastavku.

4.1 SMARTEAM V5R17

SmarTeam V5R17 je u suštini spremnik i trezor podataka, ponajprije CAD podatka, no također se može koristiti za spremanje podataka uredskih aplikacija, slika i slično. SmarTeam omogućuje reviziju i oslobađanje (pravo na korištenje) dokumenata vezanih za razvoj proizvoda i proizvodnju, gdje sustav prati sve promjene dokumenata spremljenih u trezor. Također, upotrebom korisničkih ovlasti, kontrolira pristup podacima i projektima unutar sustava. [12]

SmarTeam sustav sastoji se od nekoliko povezanih organizacijskih i funkcijskih komponenata: „DB Server“ (baza podataka), „Vault Server“ (trezor podataka), „Core Services“ (osnovni servisi za upravljanje sustavom), „Workflow Server“ (poslužitelj koji upravlja podacima vezanim uz tijek procesa), „Viewer Server“ (poslužitelj za pregledavanje CAD datoteka), „SmarTeam Editor“ (sadrži mnoštvo funkcija za mijenjanje i upravljanje sustavom), „SmarTeam Web Editor Server“ (poslužitelj za pristup sustavu preko Internet servisa), „SmarTeam CAD Integration“ (omogućuje hibridnu integraciju sustava u neke CAD sustave kao što su Catia, Solidworks, SolidEdge, ProEngineer, AutoCAD,…), „License Use Management Server“ (poslužitelj koji omogućuje SmarTeam-u upravljanje licencama). U tablici 1. navedeni su softverski i hardverski zahtjevi za pojedinu komponentu sustava.[13]

Tablica 1. Zahtjevi komponenata SmarTeam sustava, [13]

Komponenta	Softverski zahtjev	Hardverski zahtjev
Vault Server	Windows Server 2003 SP1	2xCPU Xeon DP 3.6 GHz RAM: 2 GB
Core Services Server	Windows Server 2003 SP1 Standard Edition ili Enterprise Edition	2xCPU Xeon DP 3.6 GHz RAM: 2 GB
Workflow Server	Windows Server 2003 SP1 Standard Edition ili Enterprise Edition	-
SMARTEAM – Editor	-Windows 2000 Professional SP4. -Windows XP Pro SP2. -Windows Server 2003 SP1 Standard Edition ili Enterprise Edition.	-
SMARTEAM – Web Editor Server	-Windows Server 2003 SP1 Standard Edition ili Enterprise Edition. • Microsoft Internet Information Services 6.0	4xCPU, Xeon MP 3.0 GHz RAM: 4 GB RAM
SMARTEAM – Community Workspace	-Windows Server 2003 SP1 Standard Edition ili Enterprise Edition. -Microsoft Internet Information Services 6.0	-
Viewer Server	-Windows Server 2003 SP1 Standard Edition ili Enterprise Edition. -Microsoft Internet Information Services 6.0	-
Database Server	-Oracle 10g -DB2 8.2 -MS-SQL 2005	

Kao što je vidljivo iz prethodne tablice, komponente koje se bave posluživanjem korisnika i čuvanjem podataka potrebno je učitati na poslužiteljske sustave, s adekvatnom hardverskom podrškom. Klijentske komponente, kao što je SmarTeam Editor moguće je učitati na većinu operacijskih sustava Windows sučelja. Kako za potrebe ovog rada, i

prikazivanja primjera u SmarTeamu nije bilo moguće učitati čitav sustav, sa svim komponentama, učitane su samo one koje rade i dostupne su za operacijski sustav Windows XP.

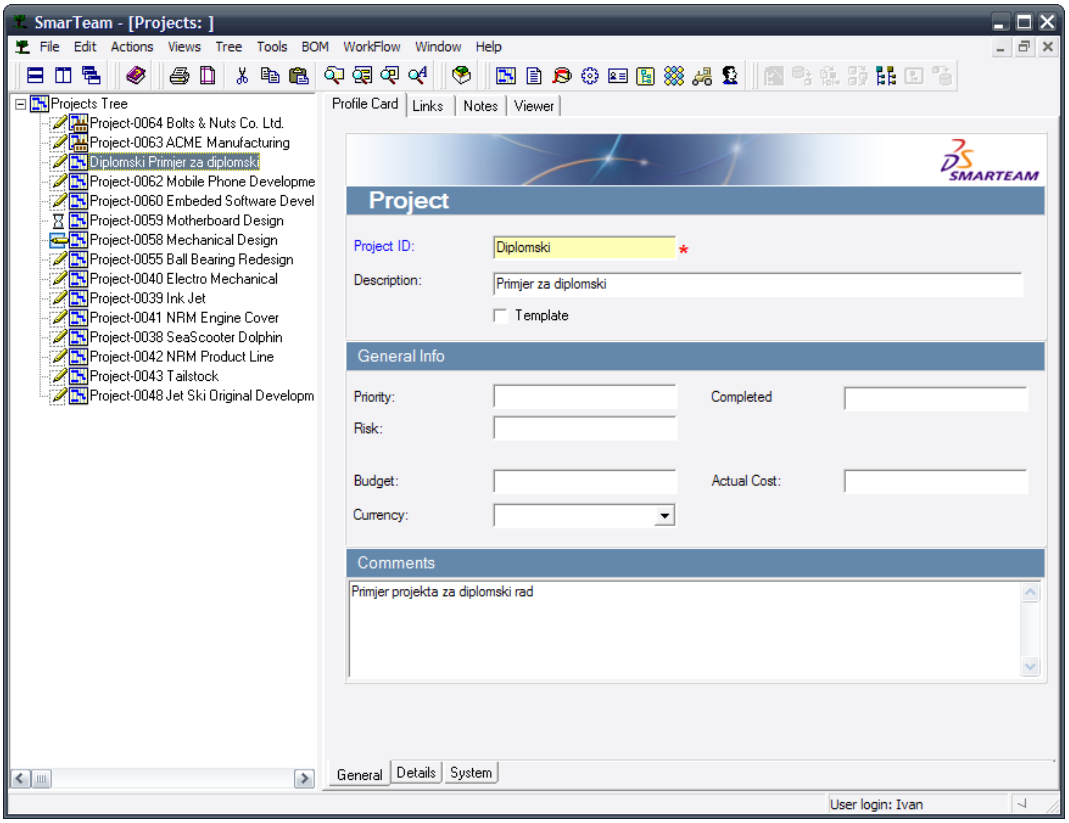
4.1.1 UČITAVANJE SMARTEAM SUSTAVA

SmarTeam u svrhu demonstracije nudi korisniku učitavanje dijelova sustava kao samostalne komponente *(eng. standalone)*, pa je tako SmarTeam Editor učitan na računalu sa Windows XP operacijskim sustavom i MS SQL 2005 Express bazom podataka, koja je neophodna za rad čak i samostalne komponente ovog sustava. Osim modula za upravljanje sustavom (*Editor*), također je u svrhu prikaza rada na primjeru učitana i demonstracijska baza. Na slici 8. prikazana je skupina komponenata koja je učitana za potrebe ovog rada.

Slika 8.Skupina učitanih komponenata

4.1.2 SMARTEAM EDITOR

Nakon povezivanja sa MS SQL 2005 Express bazom, i učitavanja skupine komponenata prikazanih na slici 8., korisničko sučelje SmarTeam Editora sa učitanom demonstracijskom bazom projekata prikazano je na slici 9..



Slika 9.Korisničko sučelje SmarTeam Editor-a

Osim sučelja prikazanog na slici 9., uz Editor se također učitava i skupina korisnih alata za uređivanje i manipulaciju korisnicima, klasama, strukturom upravljanih baza dokumenata, uvoz i izvoz dokumenata, alat za kreiranje vlastitih korisničkih naredbi i slično. Popis nekih alata i kratko pojašnjenje prikazano je u tablici 2..[14]

Tablica 2. Popis administratorskih alata SmarTeam-a, [14]

Alat	Opis
Authentication Manager	Omogućava mijenjanje postavki vezanih uz informacijske protokole
Database Connection Manager	Pomaže pri spajanju SmarTeama sa željenom bazom podataka
Export	Omogućava izvoz podataka iz SmarTeam baze, kako bi se mogli pohraniti u nekoj drugoj bazi
Flowchart Designer	Omogućava izradu tokova procesa
Form Designer	Omogućava mijenjanje postojećih formi klasa SmarTeam-a, ili kreaciju novih

	jednostavnih formi
Import	Omogućuje uvoz vijednosti potrebnih za kreaciju novih objekata u SmarTeam bazi
The Menu Editor	Omogućuje promjenu izgleda i alatnih traka korisničkog sučelja
Script Maintenance	Omogućuje stvaranje vlastitih funkcija
Data Model Designer	Omogućuje mijenjanje strukture baze podataka i dodavanje novih klasa podataka
User Maintenance	Omogućuje dodavanje korisnika, te izradu i manipulaciju korisničkim ulogama i grupama.
Web Form Designer	Omogućava izradu formi za Internet preglednike

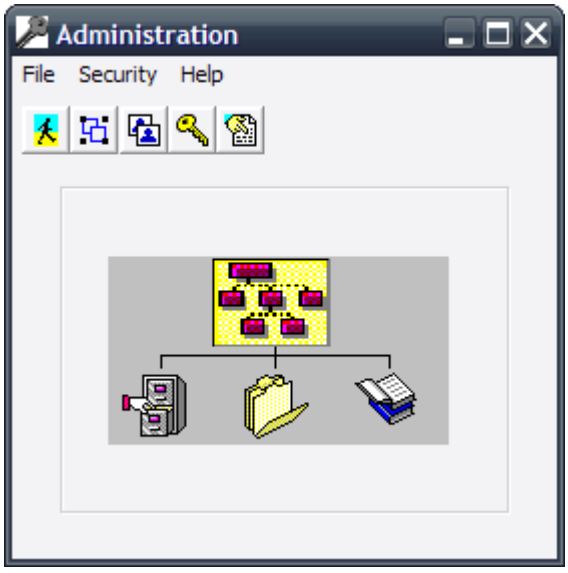
Osim navedenih alata, također unutar samog Editora postoje dodatni alati, [14]:

- Application Setup – omogućava određivanje vanjskih aplikacija za pojedine tipove podataka. Ovaj alat korišten je u primjeru rada, te je su u nastavku ukratko objašnjeni koraci za dodavanje novih tipova podataka i vanjskih aplikacija.
- Vault Maintanance – omogućuje definiranje trezora s obzirom na projekte unutar SmarTeam-a.
- File Explorer – je SmarTeam-ov interni preglednik radnog prostora.

Od navedenih alata, za potrebe ovog rada korišteni su „User maintanance“, za kreiranje korisničke strukture sustava, te „Application Setup“, za dodavanje nove vrste dokumenata koja je nastala kao rezultat rada.

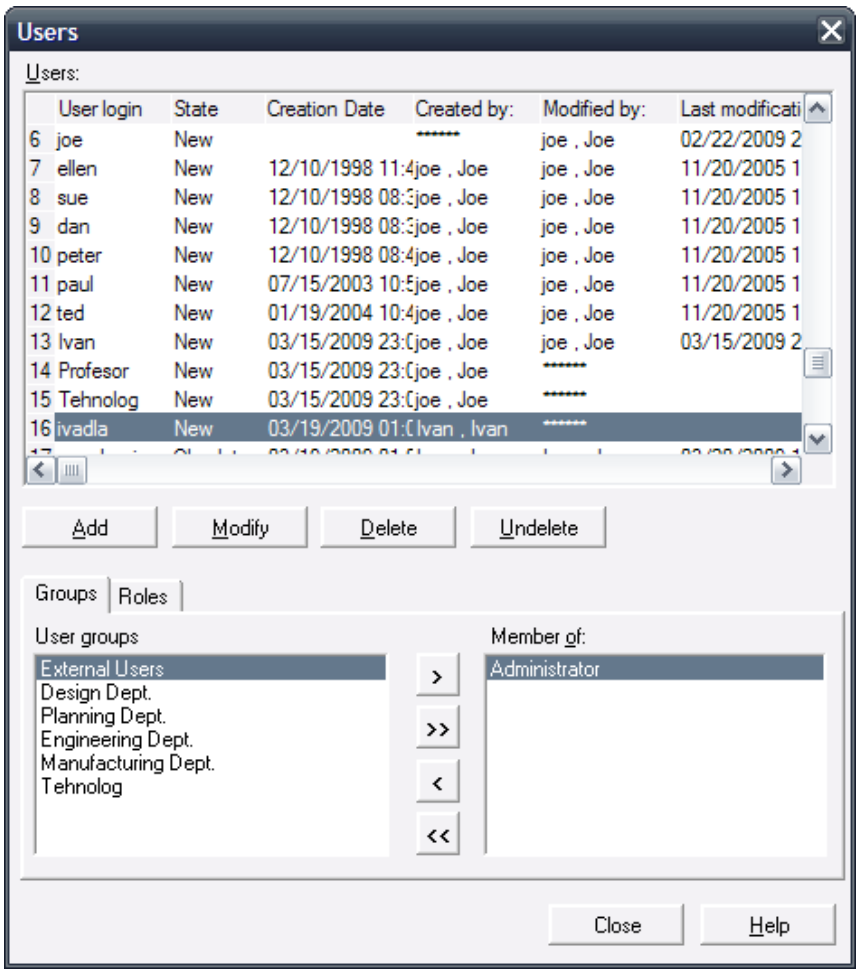
4.1.3 KREIRANJE KORISNIČKE STRUKTURE

Struktura korisnika SmarTeam sustava može se definirati u vanjskom alatu „User Maintanance“, prikazanom na slici 10.. Također, osim izrade novih korisničkih računa, moguće je definirati grupe korisnika i uloge korisnika.



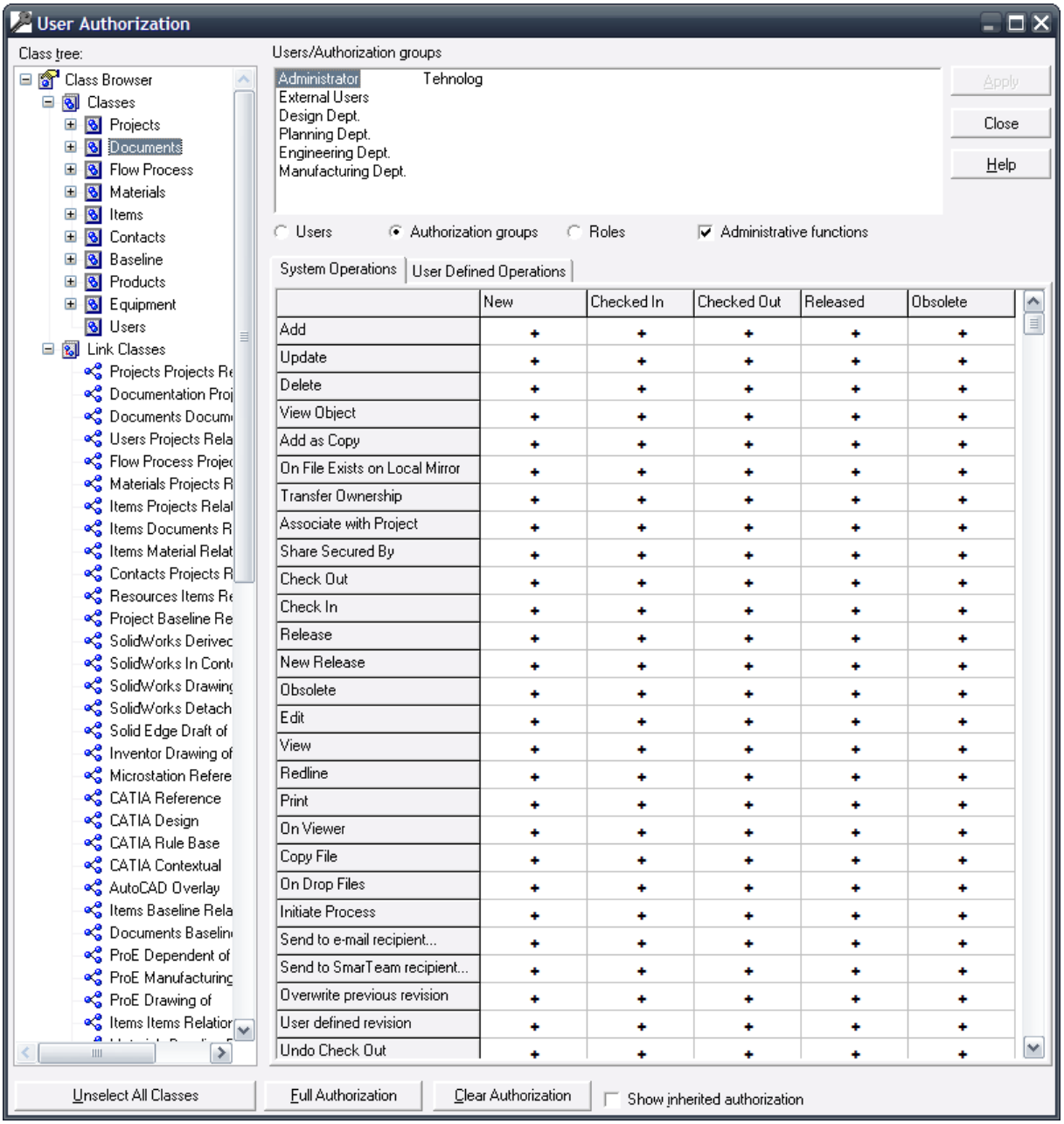
Slika 10. Korisničko sučelje alata za manipulaciju korisnicima sustava

U navedenom alatu, moguće je pregledati sve korisnike sustava, njihova prava i privilegije, odrediti korisničke grupe i uloge, kao što je prikazano na slici 11.



Slika 11. Prikaz korisnika i sučelja za promjenu uloga u korisničkih skupina

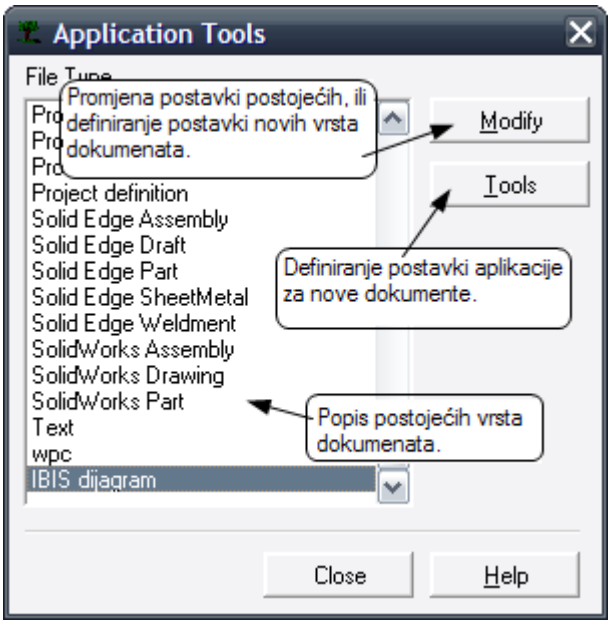
Također, bitna stvar kod korištenja ovog alata je mogućnost zabrane pristupa pojedinim funkcijama SmarTeam sustava. Tako npr. administrator sustava ima pravo pristupa svim funkcijama, dok vanjski suradnik na projektu, tj. osoba koja ne sudjeluje direktno u razvoju na projektu može imati samo pristup funkcijama za pregledavanje, ali ne i mijenjanje sadržaja. Korisničke privilegije i kontrola pristupa moguće je također ovisno o klasama unutar SmarTeama, što je za ulogu administratora i klasu dokumenata prikazano na slici 12.



Slika 12. Prikaz ovlaštenja s obzirom na ulogu korisnika i klasu

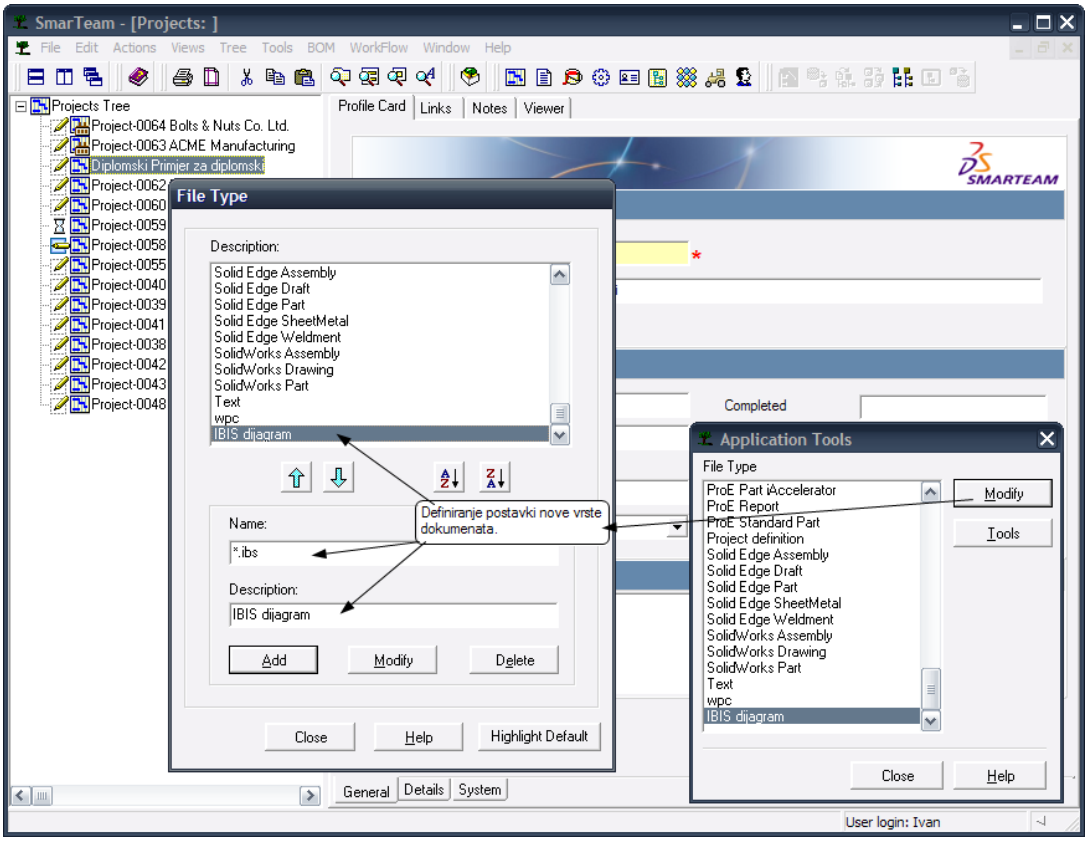
4.1.4 DODAVANJE NOVE VRSTE DOKUMENATA

U SmarTeam, nova vrsta dokumenata najjednostavnije se može prilagoditi za korištenje unutar alata „Application Setup“, prikazanog na slici 13.

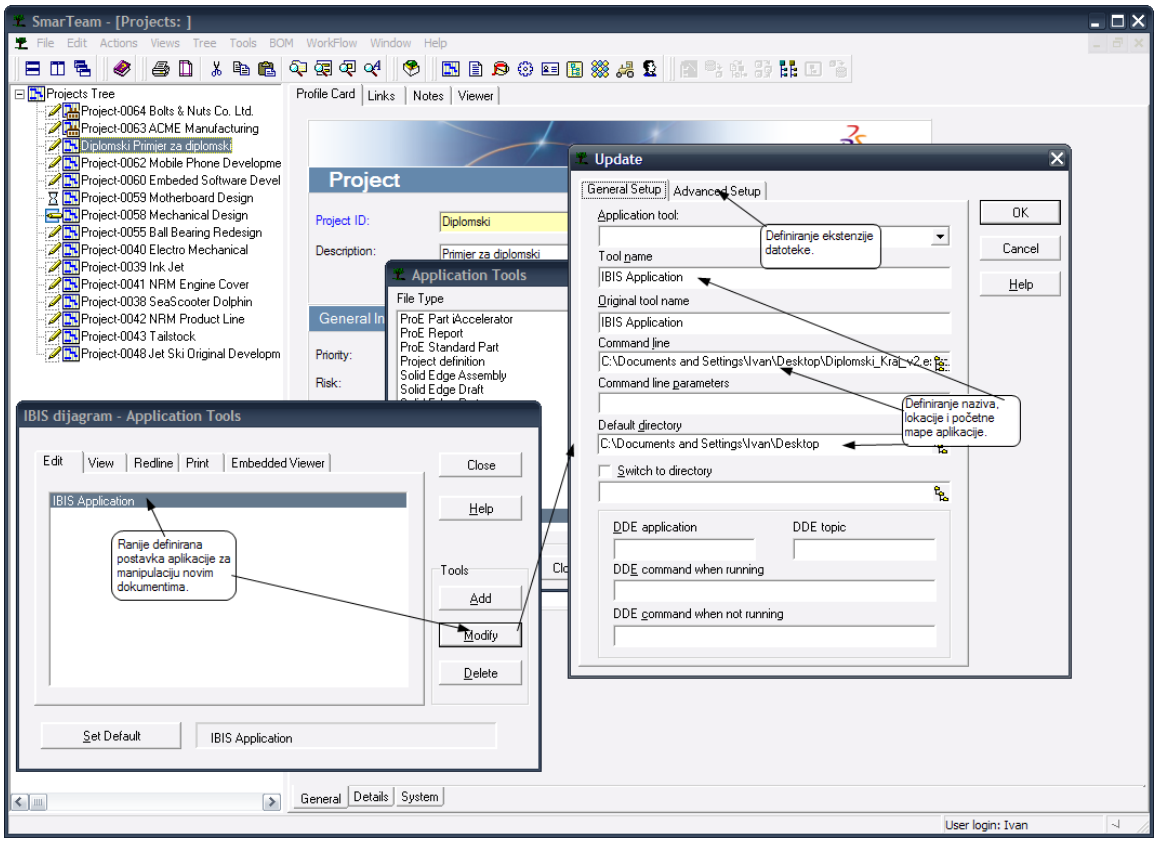


Slika 13. Korisničko sučelje za definiranje postavki novih vrsta dokumenata

Na slici 13, vidljivo je kako su na početnom sučelju prikazani svi tipovi podataka koje SmarTeam prepoznaje, tj. za koje zna što s njima treba učiniti, ukoliko ih korisnik želi pregledati. Na slici 13 vidljiva je zadnja vrsta dokumenata „IBIS dijagram“, koja je već dodana u SmarTeam, te su posložene postavke za njezinu manipulaciju. Razlog naziva „IBIS dijagram“ bit će vidljiv dalje u radu. Također je na slici 13. vidljivo gdje se mijenjaju postavke postojećih, ili definiraju nove vrste dokumenata, te gdje se definiraju postavke aplikacije za manipulaciju novim dokumentima. Korisničko sučelje u slučaju definiranja nove vrste dokumenata prikazano je na slici 14., dok je ono za definiranje aplikacije za manipulaciju prikazano na slici 15.. Vrsta dokumenta vidljiva na ovim slikama i njen naziv bazira se na jednoj od metoda za prikaz logičke podrške tijeku konstruiranja, tj. sustavima za bilježenje tijeka promišljanja i odlučivanja. Neki od tih sustava predstavljeni su u sljedećem poglavlju.



Slika 14. Definiranje postavki nove vrste dokumenata



Slika 15. Definiranje aplikacije za manipulaciju novim dokumentima

5 SUSTAVI PRIKAZIVANJA LOGIČKE PODRŠKE KONSTRUIRANJU

5.1 LOGIČKA PODRŠKA

Termin *logička podrška procesu konstruiranja* je zapravo slobodan prijevod engleskog termina *Design Rationale*. Isto bi se također moglo protumačiti kao *rezoniranje u konstruiranju*. Ukratko, logička podrška bi se prema [15], mogla definirati kao *teoretsko objašnjenje konstruiranja*. No, ideja je svakako da je logička podrška zapravo znanje i promišljanje korišteno tijekom procesa konstruiranja, koje se zapravo i ne bilježi. Sudionici konstrukcijskog tima često navedu samo rješenje problema, no ne i tijek promišljanja, ili ideja koji je vodio do rješenja.

Prethodno navedenu definiciju upotpunio je Jintae Lee (1997.), te je logičku podršku objasnio prema [15]: „*logička podrška konstruiranju nisu samo razlozi donošenja pojedine odluke, već i obrazloženje tih razloga, razmotrene alternative rješenja, te pripadajuća argumentirana diskusija koja je dovela do rješenja*“.

Naravno, osim objašnjenja pojma logičke podrške koji je predložio Lee, postoje i druge definicije,koje su zapravo objašnjenja slična onome koje je predložio Lee, a neka od njih su prema[15], da logička podrška konstruiranju:

- prikazuje elemente promišljanja koji su uloženi za konstruiranje proizvoda, (Shum, 1993)
- su navodi o promišljanju koji su temeljni za proces konstruiranja, te oni objašnjavaju i opravdavaju donesene odluke, (Fischer, 1995.)
- je informacija koja objašnjava zašto je proizvod strukturiran na način na koji je, te zašto posjeduje karakteristike koje posjeduje, (Conklin, 1995.)

Bez obzira koje objašnjenje za proces logičke podrške konstruiranju se odabere, iz svakog se da iščitati da ona sadrži informacije o uzrocima problema, promišljanjima

vezanim uz probleme, generiranim idejama za rješavanje problema, te barem naznakama toka misli u pronalasku problema.

Slijedeći bilo koje objašnjenje, logička podrška ima potencijal da se koristi na više načina, a neke od njih su prema [16], definirali Burge i Brown (1998.), te su predložili kako se logička podrška može koristiti za:

- Kontrolu konstruiranja – za provjeravanje da li su donesene odluke i konstruiran proizvod upravo ono što se namjeravalo postići, tj. da se može vratiti u ranije korake u procesu konstruiranja i potvrditi ispravnost donesenih odluka.
- Vrednovanje konstruiranja – logička podrška se koristi za ocjenjivanje diskutiranih alternativa, što može nakon pregleda polučiti i bolje rezultate od očekivanih.
- Podržavanje konstruiranja – gdje se koristi za lakše uočavanje potrebnih promjena konstrukcije, najčešći slučaj je prilikom konstruiranja sličnih proizvoda.
- Ponovnu upotrebu konstrukcije – uz eventualne manje izmjene, na raspolaganju je cijeli proces od problem do rješenja, ukoliko je isti bio i ranije korišten.
- Učenje konstruiranja – pri čemu tijek promišljanja iskusnijih konstruktora omogućava lakše učenje novih konstruktora. Također, omogućava osobama koje nisu po struci upoznati sa metodama konstruiranja da imaju uvid u tijek događanja i procese prilikom konstruiranja.

Pregledom načina na koje se logička podrška može koristiti, prema [17], neko od korisnih uloga njenog korištenja bile bi:

- Praćenje tijeka odluka.
- Komuniciranje između projekata.
- Strukturiranje i analiza novih konstrukcijskih problema.
- Zadržavanje dosljednosti prilikom donošenja odluka.
- Dodatni element u bazi znanja kroz ponovnu uporabu postojeće logičke podrške.
- Praćenje napretka u tekućim projektima i konstrukcijskim problemima, te prepoznavanje neriješenih problema i pitanja.
- Olakšano i ubrzano integriranje novih članova konstrukcijskog tima.

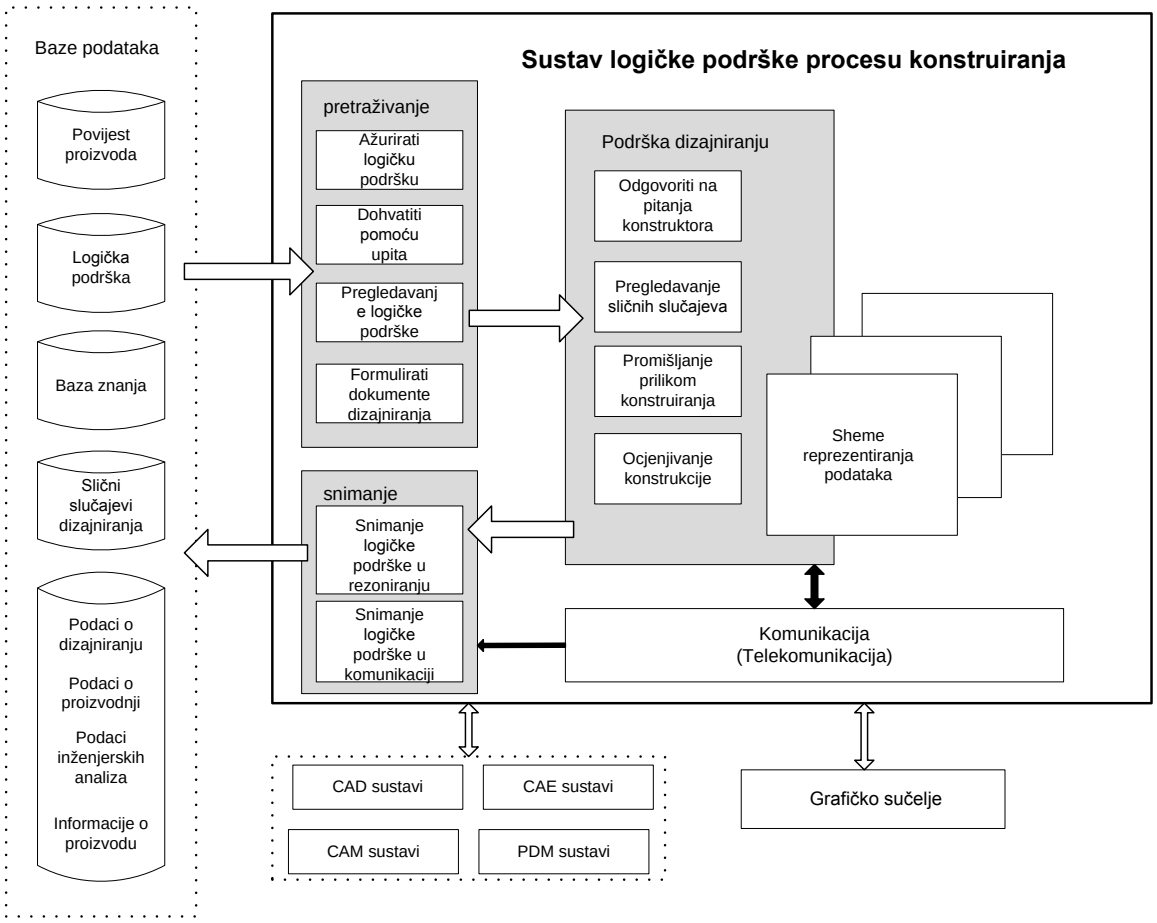
Kako bi se sam tijek promišljanja i odlučivanja mogao koristiti, potrebno ga je na neki način i prikazati, tj. predstaviti konstruktorima. Za tu svrhu postoje načini prikaza logičke podrške procesu konstruiranja, koji su prema [16], po stupnju formalnosti podijeljeni u tri grupe:

- *Neformalni* – rezoniranje se bilježi korištenjem tradicionalnih metoda i alata, tj. korištenjem alata za obradu teksta, snimanjem zvuka, ili videa, ili već pomalo zastarjela metoda bilježenja ručno papirom i olovkom (što je ipak u nekim slučajevima najbolje i najbrže rješenje, no ono bi se svakako za daljnju upotrebu trebalo prenijeti u neki drugačiji oblik zapisa).
- *Formalni* – ovaj način prikaza nalaže da se tijek promišljanja bilježi u točno određenom i definiranom formatu koji računalo može interpretirati i razumjeti.
- *Poluformalni* – je pokušaj povezivanja prednosti jednog i drugog ranije navedenog načina, tj. prikupljene i zabilježene informacije trebale bi biti takve da ih je moguće interpretirati računalom, no način njihovog bilježenja ne bi trebao biti rigorozno definiran, kako bi se ipak dopustila sloboda razmišljanja kod korisnika. Ovakav način zapravo omogućuje korisniku da ne razmišlja o formatu zapisa previše, već da se usredotoči na primaran posao, što je promišljanje. Korisnik kod ovakvog načina može koristiti unaprijed postavljene elemente, za koje korisnik intuitivno zna o čemu se radi kada koristi element. Osim korištenja predefiniranih oblika, korisnik svakom elementu dodjeljuje vlastito objašnjenje. Ovakav način prikaza najprikladniji je za korištenje u timskom okruženju zbog naravi njegovog korištenja i slobode koju pruža, a opet ga je moguće računalno obraditi i pohraniti.

Za efikasno bilježenje nekog od načina prikazivanja logičke podrške, potrebno je izraditi sustav, a pristupi i smjernice za izradu tog sustava ukratko su nadalje objašnjeni.

5.2 SUSTAVI ZA LOGIČKU PODRŠKU

Prilikom izrade sustava za logičku podršku procesu konstruiranja potrebno je obratiti pozornost na ranije navedenu slobodu korištenja i mogućnost manipulacije računalom. Što bi značilo da članovi konstrukcijskog tima moraju imati slobodu razmišljanja i diskutiranja o projektu, u okvirima sustava za logičku podršku, tj. korištenjem prikaza logičke podrške. Kako bi se tijek promišljanja mogao i spremati, te kasnije dohvaćati, generalna arhitektura takvog sustava trebala bi biti prema slici 16. [18]



Slika 16. Generalna arhitektura sustava za logičku podršku konstruiranju, [18]

Trenutni sustavi logičke podrške konstruiranju još uvijek nisu na razini sustava kakav je prikazan na slici 16., no ipak, postoje neki pristupi za izradu, koji pokušavaju dokučiti kako sustav izvesti da bude upotrebljiv i koristan, a ne samo još jedna prepreka u procesu konstruiranja.

Pristupi za izradu i razvoj ovakvog sustava prema, [18] podijeljeni su na:

- *Pristupi orijentirani na procese (eng. process-oriented)* – koriste se u dinamičkim okolinama za prikaz povijesnog puta do stvaranja proizvoda. Većina trenutnih sustava je orijentirana na procese, a ovakav pristup proizašao je iz sustava za prikaz argumentiranja IBIS[8], koji će dalje u radu biti pobliže pojašnjen.
- *Pristupi orijentirani na značajke (eng. feature-oriented*) - koriste se u područjima sa relativno visokom razinom standardizacije za logički prikaz

[8] IBIS (eng. Issue Based Information System) = informacijski sustav koji se bazira na pitanjima (predmetu rada)

proizvoda, kako bi se slijedila rigorozna pravila procesa konstruiranja. Pristup kreće od trenutnog stanja konstrukcije određenog proizvoda, te se obično razvija u kontekstu postavljenih zadaća, koristeći empirijske studije. Sustavi kreirani ovim pristupom sadrže baze znanja, te imaju tendenciju korištenja automatskog promišljanja, tj. zaključivanja na temelju unesenih podataka. Baze znanja se u ovom slučaju koriste kao podrška sustavu za automatizirano generiranje logičke podrške, te su prikazi svakako više formalne prirode, nego što su kod pristupa orijentiranog na procese. Ovakvim pristupom nije moguće bilježiti apstraktne informacije, već samo one predefinirane sustavom.

Na način pristupa razvoju sustava logičke podrške također utječe i spoznaja da li će se sustav integrirati sa nekim drugim inženjerskim sustavima, kao što su to CAD sustavi.

Za sustave logičke podrške procesu konstruiranja također je bitno snimanje, tj. spremanje tijeka promišljanja, kako bi se tijek razmišljanja mogao kasnije nastaviti i koristiti. Prema [18], postoje također dvije vrste metoda snimanja tijeka promišljanja:

- *Metode koje zahtijevaju akciju korisnika (eng. user-intervention)* – ovim metodama snimaju se informacije o samim odlukama, vremenu donošenja odluka, korisniku koji je donio pojedinu odluku i razlogu donošenja odluke, tj. dokumentira se povijest konstruktorovih aktivnosti.
- *Automatizirane metode (eng. automatic)* – kako bi metode bile automatizirane, pretpostavlja se da postoji način i metode za snimanje komunikacije između članova tima i sustava za podršku procesa konstruiranja. Iz snimljene komunikacije se kasnije može pregledati tijek promišljanja i odlučivanja.

Nakon spremanja tijeka promišljanja, sustavi logičke podrške moraju biti u mogućnosti spremljeno znanje i vratiti, tj. pronaći i dohvatiti. Naravno, bitno je pronaći relevantno znanje za trenutni projekt, a ne bilo koje iz baze. Tako prema [18], postoji nekoliko metoda pronalaženja, a to su:

- *Pregledavanje arhiviranih slučajeva* – korisnici zapravo sami pretražuju ranije spremljene informacije korištenjem jednostavnih poveznica između tih informacija. Kod procesno orijentiranog pristupa, ovakav način pregledavanja arhiviranih slučajeva omogućava i vraćanje u ranije korake i događaje procesa konstruiranja.
- *Korištenje upita za pronalazak pravih informacija* – mnogo je efikasnije od pregledavanja svih spremljenih informacija, no korisnik mora biti upoznat sa parametrima pretraživanja.

- *Automatizirano pronalaženje* – pretpostavlja postojanje umjetne inteligencije i ekspertnih sustava u svrhu prepoznavanja konteksta konstruiranja i predlaganja ranije završenih slučajeva. Sličan učinak može se postići pravilnim i opsežnim označavanjem dovršenih projekata.

5.3 SHEME I MODELI ZA PRIKAZIVANJE LOGIČKE PODRŠKE KONSTRUIRANJU

Postoji mnogo modela i shema za prikazivanje tijeka promišljanja, no bitno je znati odabrati i koristiti pravu i prikladnu. Najranija shema, tj. model prikazivanja tijeka promišljanja i odlučivanja prema [19], je ranije spomenuti IBIS sustav, kojeg su 1970. godine opisali Horst W. J. Rittel i Werner Kuntz u svom radu „Issues as elements of information systems". Neke od shema za prikazivanje tijeka promišljanja i odlučivanja u konstruiranju su prema [18]:

- *IBIS (eng Issue Based Information System)* – pokušava zabilježiti probleme koji se javljaju u procesu konstruiranja. Ključni problemi naglašavaju se kao *pitanja*, tako da se u obliku pitanja postavljaju problemi na koje su odgovor *ideje* . *Ideje* se postavljaju kao potencijalno rješenje problema postavljenog *pitanjem*. Kao obrazloženje, ili način ostvarivanja *ideja* postavljaju se *argumenti*. *Argumenti* mogu biti pozitivan, ili negativan odgovor i obrazloženje *ideja*. Nakon argumentiranja, svaka *ideja* može biti prihvaćena, ili odbačena. Sve ovo moguće je prikazati IBIS sustavom, što je također jednostavno i intuitivno za upotrebu od strane korisnika. No, IBIS sustavi ne snimaju eksplicitno kriterije kod odlučivanja, te se zanemaruju kompliciranije relacije između pojedinih problema. Na temelju IBIS sustava razvijeni su i drugi sustavi kao što je gIBIS[9], koji će naknadno biti ukratko objašnjen.[19]
- *PHI[10] (eng. Procedural Hierarchy of Issues)* – također nastao na bazi IBIS sustava, a povećava njegov doseg proširenjem koncepta problema, uz to mijenja strukturu koja povezuje IBIS-ove probleme, ideje i argumente.[18]
- *DSA[11] (eng*. Design Space Analysis*)* – postavlja proizvod za koji se vrši proces konstruiranja u prostor mogućnosti, tj. mogućih stanja, čime se pokušava objasniti i pronaći razlog zašto je od svim mogućnosti odabrano dobiveno rješenje. Sustav koristi poluformalnu notaciju nazvanu QOC[12] za prikazivanje prostora okruženja

[9] gIBIS (eng. Graphical Issue Based Infromation System) = grafički informacijski sustav koji se bazira na pitanjima (predmetu rada)

[10] PHI (eng. Procedural Hierarchy of Issues) = proceduralna hijerarhija problema

[11] DSA (eng. Design Space Analysis) = analiza prostora konstruiranja

[12] QOC (eng. Questions, Options, Criteria) = pitanja, mogućnosti, kriterij

proizvoda, koristeći tri komponente: *pitanja*, *opcije*, i *kriterije*. U ovom sustavu *pitanja*, koja su zapravo ekvivalentna onima u IBIS sustavu, predstavljaju glavni problem. Za pitanja se postavlja prostor mogućnosti, tj. *opcije*, koje traže rješenja na postavljeni problem. Nakon postavljenog prostora mogućnosti koriste se *kriteriji* za ocjenjivanje i uspoređivanje *opcija*. [19]

- *DRL*[13] *(eng. Decision Representation Language)* – je ekspresivan jezik za prezentaciju odluka, tj. stanja koje okružuje odluke. U njemu se prezentiraju kvalitativni elementi u važni u odlučivanju. Tako su prema [19], osnovni elementi odlučivanja kod ovog sustava *problem odlučivanja*, *alternative*, *tvrdnje*, i *ciljevi*.
- *FR*[14] *(eng. Functional Representations)* – ova shema ima za cilj prikazati funkciju proizvoda, ili poželjne i zahtijevane funkcije proizvoda. U prikazu se bilježe informacije o tome kako i na koje načine proizvod zadovoljava postavljene funkcijske zahtjeve. [18]

Od navedenih shema prikaza, prema [19], najvažnije su IBIS, PHI (koji se zapravo poistovjećuje s QOC) i DRL, koji su uspoređeni u tablici 3.

Tablica 3. Usporedba najvažnijih shema prikazivanja logičke podrške, [19]

	IBIS	QOC	DRL
Osnovni entiteti	Problem	Pitanje	Problem odlučivanja
	Ideja	Opcija	Alternativa
	Argument		Tvrdnja
		Kriterij	Cilj
Prednosti	Jednostavna notacija	Eksplicitno prikazuje relacije kriterija	Ekspresivna reprezentacija; Snimanje kompleksnih relacija
Ograničenja	Nema prostora kriterija	Ne može prikazati mnoge aspekte argumenata	Ne podržava generiranje alternativa

Kako se dokumenti za praćenje i pohranjivanje tijeka promišljanja i odlučivanja u ovom radu zapravo baziraju na IBIS sustavu, on će u nastavku biti bolje pojašnjen.

[13] DRL (eng. Decision Representation Language) = jezik za prezentaciju odluka

[14] FR (eng. Functional Representations) = funkcijska prezentacija, tj. prezentacija funkcija

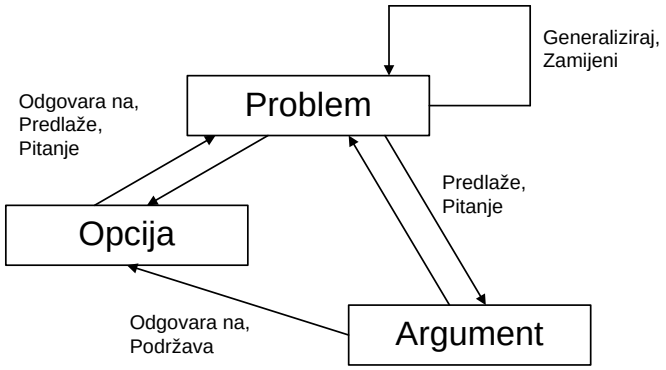
5.4 IBIS SUSTAV

U procesu konstruiranja u timu članovi tima u međusobnoj interakciji izmjenjuju informacije, a također, prema [20], odvijaju se još neke kategorije izmjene informacija, a to su: izmjena informacija sa stručnjacima povezanim s pojavljenim problemom, akumuliranje informacija iz dokumentacije i stručnih radova, te informacije dobivene od krajnjeg korisnika proizvoda koji se konstruira. IBIS sustav svojim entitetima omogućava članovima konstrukcijskog tima da kvalitetno raspravljaju i bilježe svoja razmišljanja. U IBIS sustavu, nastali problemi generiraju ideje, koje se obrazlažu argumentima. U svakom trenutku, ukoliko je dovoljno dobro objašnjeno i članovi tima su suglasni, argumenti, ideje, ili problemi mogu postati odbačeni, ili prihvaćeni.

U IBIS-u su problemi, tj. *pitanja* osnovica daljnje rasprave i tijeka promišljanja, a problemi bi prema [20], trebali:

- imati formu pitanja
- biti posljedica kontroverznih i kontradiktornih izjava
- biti specifični za svaku pojedinu situaciju
- prilikom rješavanja biti osporavani, izbjegavani, zamjenjivani i odbacivani

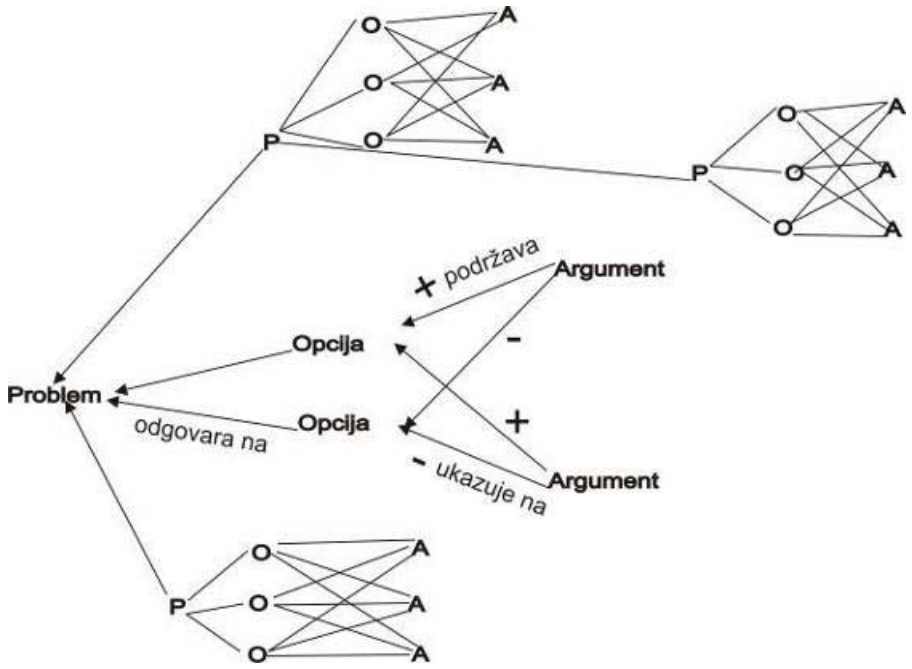
Osnovni IBIS sustav je zapravo bio rječnik, te su se morali poštivati neki principi kako bi sustav funkcionirao kako je bilo zamišljeno. Elementi takvog sustava su prema [20], *teme*, koje služe kao gruba organizacijska načela; *pitanja o problemu*, koja traže nekontroverznu informaciju kao odgovor, te *modeli problema*, koji odgovaraju najčešće znanstvenim modelima koji se bave čitavim klasama sličnih problema.



Slika 17. Shema osnovnih elemenata IBIS-a, [21]

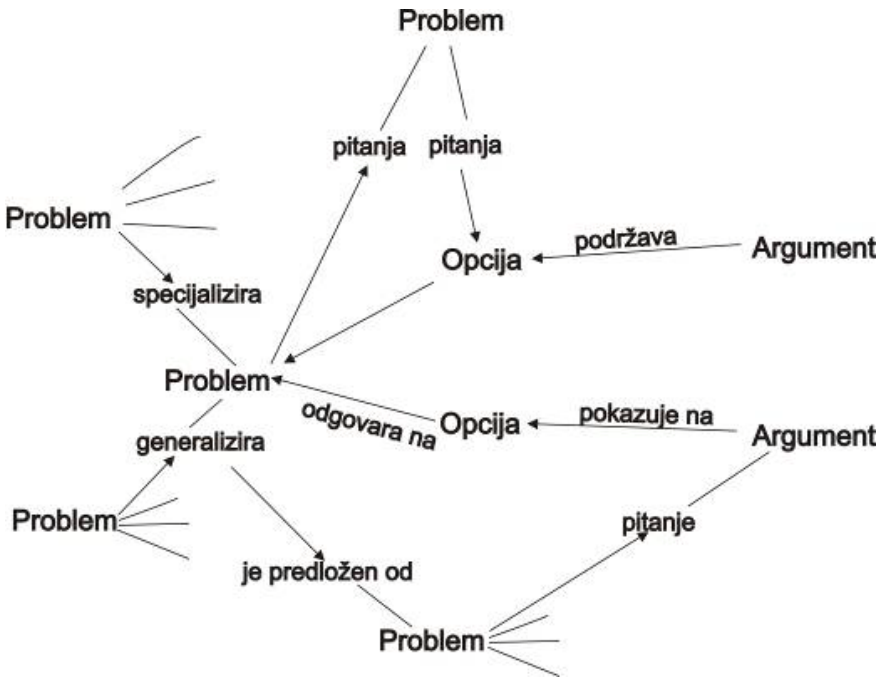
U današnjim IBIS sustavima, na osnovu prve ideje, moguće je odrediti tri ključna entiteta. Prema [21], to su *problemi (eng. Issues)*, *opcije (eng. Positions)* i *argumenti (eng.*

Arguments). Takvi entiteti povezuju se usmjerenim relacijama, s kratkim opisom prirode relacije, kao što je: *podržava, pokazuje na, zamjenjuje,* i sl. Takav osnovni IBIS sustav, sa tri ključna entiteta zamišljen je da se koristi na način prikazan shemom na slici 17. Naravno, način bilježenja tijeka promišljanja i problema koji se javljaju u procesu konstruiranja više se ne može predočiti samo tim osnovnim elementima, no, kad bi se ti elementi povezali relacijama i proširili, dobio bi se sličan graf promišljanja kao što je prikazan na slici 18.. [21]



Slika 18. Primjer grafa dobivenog korištenjem osnovnog IBIS-a, [21]

Graf na slici 18. predstavlja samo grafički prikaz zapisa osnovnog IBIS sustava, koji je bio zapravo rječnik pojmova. No, stvarna nadogradnja sustava događa se pojavom gIBIS sustava. gIBIS je u primjenu stavio grafički prikaz entiteta IBIS sustava i relacija između njih. gIBIS se u počecima koristio u hipertekstnim sučeljima,te se zbog toga mogao koristiti u mrežnoj okolini, što je i danas ideja vodilja za sustave praćenja i bilježenja tijeka promišljanja i odlučivanja. Ključni Entiteti gIBIS sustava jednaki su kao i oni IBIS sustava, tj. *problemi*, *opcije* i *argumenti*, zajedno sa relacijama, koje se u slučaju gIBIS-a prikazuju grafičkim prikazom, uz tekstualno objašnjenje relacija. Primjer sheme grafa dobivenog korištenjem gIBIS-a prikazan je na slici 19..[22]



Slika 19. Shema prikaza gIBIS grafa, [22]

Kao i svaki realni sustav IBIS također ima svoje prednosti i nedostatke. Prema [23], neke važnije i češće zamijećene prednosti IBIS sustava su da IBIS pruža korisne metode za opisivanje procesa konstruiranja, ne samo šturim riječima, nego uključujući i raspravu i odluke vezane uz zahtjeve konstrukcije. Također, kao prednost je navedena jasna struktura i značenje ključnih elemenata sustava, gdje relacijski prikazana evolucija problema, ideja i argumenata daje dobar uvid u tijek promišljanja i odlučivanja. Vjerojatno najbitnija prednost IBIS sustava je ta što pruža vrlo bitne podatke potrebne za olakšavanje procesa konstruiranja na novim konceptima i proizvodima, te pojednostavljuje procese rekonstruiranja i modifikacija proizvoda. Kao nedostaci se navode relativno mala povezanost problema opisanih sustavom sa vanjskim problemima, budući da se u IBIS-u rješavaju specifični problemi, nema dovoljno dobrog uvida u cijelu situaciju procesa konstruiranja kompleksnih proizvoda, već samo naznake ostalih problema. Još jedan problem, koji nije specifičan samo za IBIS sustave je da se javljaju poteškoće prilikom traženja specifičnih informacija iz baze, za potrebe nekog projekta. No, navedeni problemi se u sve većoj mjeri rješavaju korištenjem novih tehnologija i modifikacijama osnovnih elemenata sustava. IBIS danas postoji kao ideja začetnik sustava, ali više ne i kao sustav. Današnji sustavi prerastaju pojam osnovnog IBIS-a, no svakako, i njihovi elementi nalaze uporište u prvom i osnovnom IBIS modelu.

Na temelju IBIS sustava je također zamišljeno praćenje i spremanje tijeka promišljanja i odlučivanja u obliku elektroničkog dokumenta, što je cilj ovoga rada. Dalje u radu, osnovni elementi programskog sustava temelje se na prikazanom IBIS modelu.

6 ZAHTJEVI ZA PROGRAMSKI SUSTAV

Nakon proučavanja područja i informacija iznesenih ranije u ovom radu, cilj je osmisliti programski sustav za bilježenje i pohranjivanje tijeka promišljanja i odlučivanja u obliku dokumenta, koji će se moći kasnije koristiti, te umetnuti kao sastavni dio nekog PDM sustava. Takav dokument ima značajan utjecaj na razvoj proizvoda i proces konstruiranja, jer pruža prednosti ranije navedene u radu i ubrzava sam proces konstruiranja, što ima za rezultat kraće vrijeme razvoja proizvoda, koje je kao što je ranije u radu prikazano, veliki potencijalni generator troškova.

Programski sustav za grafički prikaz i računalnu pohranu tijeka promišljanja i odlučivanja pri timskom razvoju proizvoda potrebno je koncipirati prema poznatom IBIS sustavu. Točnije, rezultirajući dokument trebao bi imati osnovne elemente po uzoru na ključne entitete IBIS sustava. Programski sustav bi trebao biti u mogućnosti prepoznavati strukturu korisnika. No, ukoliko se on koristi uz PDM sustav, struktura korisnika se definira u PDM sustavu,kao što je to pokazano ranije u radu, a u programskom sustavu za izradu dokumenata koji prikazuju tijek promišljanja i odlučivanja, ta struktura korisnika se samo provjerava. No, prvo je bitno definirati osnovne entitete sustava, po uzoru na IBIS sustave.

6.1 ELEMENTI GRAFIČKOG PRIKAZA

Prema [24], osnovni elementi sustava, po uzoru na IBIS sustave su *problem*, *ideja* i *argument*. Svaki od tih elemenata može poprimiti obilježja karakteristična za vrstu elementa.

Tako *problem* može biti: *aktivan, prihvaćeni*, ili *odbačeni*. U smislu karakteristike elementa, *aktivni problem* označava problem koji je potrebno riješiti, tj. za koji je potrebno generirati ideje, te argumentirati njihovo izvođenje i rješavanje. Nakon generiranja ideja i argumenata, *aktivni problem* može indirektno uzrokovati nove probleme. *Prihvaćeni problem* označava riješeni problem, no, samo nakon prihvaćanja i usklađivanja mišljenja vezanih uz generirane ideje i argumente. *Odbačeni problem*

označava problem koji više ne postoji, tj. u daljnjem procesu konstruiranja, ranije označeni i navedeni problem je na neki drugi način riješen, ili više nije relevantan za rješavanje konstrukcijskog problema.

U smislu karakteristike elementa, *ideja* može poprimiti obilježja: *aktivna, prihvaćena i neprihvaćena*. Na taj način, *aktivna ideja* označava ideju o kojoj se trenutno raspravlja, a nastala je kao potencijalno rješenje problema. *Prihvaćena ideja* označava ideju koja je dovoljno dobro argumentirana i izvediva, a kao njen rezultat omogućeno je rješavanje problema. *Neprihvaćena ideja* označava ideju koja nije dobila dovoljnu podršku u argumentima, te se nije ispostavila kao dobro potencijalno rješenje problema.

Argument može poprimiti obilježja *aktivni argument ZA, aktivni argument PROTIV, odbačeni argument ZA* i *odbačeni argument PROTIV Aktivni argument ZA* tako označava argument koji podržava predloženu ideju, te nudi način izvršavanja i realiziranje te ideje, kao potencijalnog rješenja problema. *Aktivni argument PROTIV* označava argument koji ne podržava predloženu ideju, tj. obrazlaže i navodi razloge zašto ona nije dobar put ka potencijalnom rješenju. *Odbačeni argument ZA* označava argument koji podržava generiranu ideju, no više ne utječe na dobivanje rješenja, a takav može postati ukoliko je generirana neka nova ideja koja zamjenjuje ranije argumentiranu ideju. *Odbačeni argument PROTIV* označava argument koji ne podržava generiranu ideju, no više nije relevantan za daljnji tijek procesa konstruiranja, budući da je generirana druga ideja koja nudi bolje potencijalno rješenje problema. Ovi *odbačeni* argumenti ostaju u prikazu tijeka promišljanja u svrhu lakšeg razumijevanja procesa i tijeka konstruiranja, te načina dolaska do rješenja. Oni mogu biti ključan faktor u prepoznavanju problema u procesu konstruiranja, te približiti članovima tima način razmišljanja konstruktora. U tablici 4 su prema [24], prikazani modificirane grafičke oznake elemenata prikaza.

Tablica 4. Grafičke oznake elemenata prikaza

Grafička oznaka	Element prikaza
	Aktivni problem
	Prihvaćeni (riješeni) problem
	Odbačeni (nepostojeći) problem
	Aktivna ideja

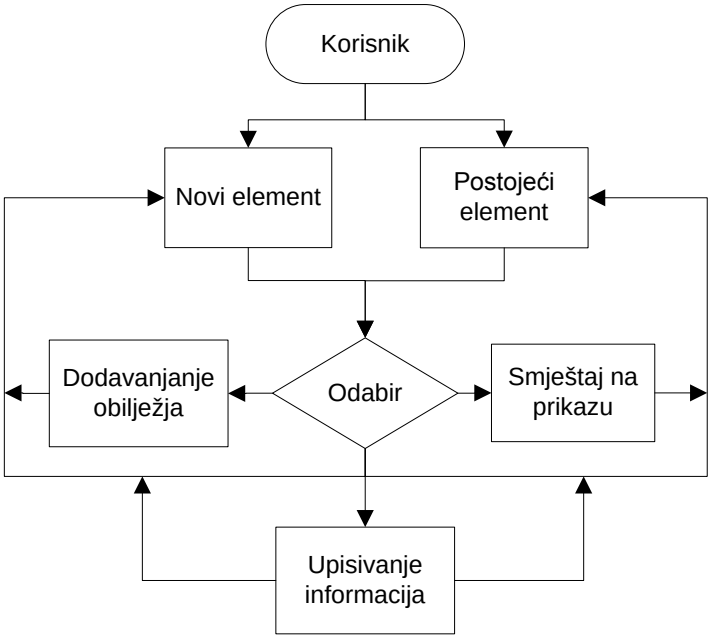
Grafička oznaka	Element prikaza
	Neprihvaćena (odbačena) ideja
	Prihvaćena ideja
	Aktivni argument ZA
	Aktivni argument PROTIV
	Odbačeni argument ZA
	Odbačeni argument PROTIV

6.2 FUNKCIJE GRAFIČKOG PRIKAZA

Nakon navedenih osnovnih elemenata grafičkog prikaza, potrebno je osmisliti i realizirati funkcije koje omogućuju jednostavnu manipulaciju i kreiranje dokumenta tijeka promišljanja. Tako, s obzirom na koncept IBIS sustava, prema [24], korisnik programskog sustava trebao bi biti u mogućnosti odabrati željeni element, te ga proizvoljno smjestiti u dokumentu. Nakon smještaja elementa, potrebno je korisniku pružiti mogućnost upisivanja teksta, kako bi se element mogao pojasniti i predstaviti što on označava. Kao što je ranije navedeno, svaki element ima nekoliko svojih karakteristika, tj. obilježja. Kako bi se izbjeglo gomilanje predefiniranih elemenata, potrebno je korisniku omogućiti dodavanje različitih obilježja elementa na već postojećem i postavljenom elementu. Svaki korisnik koji ima pravo mijenjati dokument, trebao bi u svakom trenutku biti u mogućnosti promijeniti sadržaj elementa, pomicati element unutar dokumenta, dodijeliti mu njegova specifična obilježja, te povezati elemente koji proizlaze iz prethodnog elementa.

Kao dodatno pojašnjenje određenog elementa, korisno je imati mogućnost postavljanja slike, ili skice, koja predočava problem, podržava ideje, ili pojašnjava argumente. Iako ovo nije osnovni element IBIS sustava, svakako omogućuje bolji uvid u cjelokupan tijek promišljanja i pojašnjava razloge donošenja određenih odluka.

Osnovna ideja funkcioniranja programskog sustava, tj grafičkog prikaza i njegovih elemenata prikazana je na slici 20..



Slika 20. Shema funkcija na grafičkom prikazu

6.3 POVEZIVANJE ELEMENATA ZAPISA SA VANJSKIM DATOTEKAMA

Kako bi se sustavu povećala funkcionalnost, zanimljivo je omogućiti povezivanje dijelova dokumenata sa vanjskim datotekama koje pobliže opisuju elemente. Kao što je slučaj i sa slikama, ni ovo nije osnovni element IBIS sustava, no čak i više od dodane slike može pojasniti razloge donošenja odluka, ili poduprijeti navedene argumente u tijeku promišljanja. Za povezivanje dijelova dokumenta potrebno je predefinirati još jedan element, *poveznicu*, koja se može dodavati uz ostale osnovne elemente. Takva poveznica može se koristiti za sve vrste dokumenata, za koje postoji adekvatna aplikacija na operativnom sustavu, a primjer takvih dokumenata su dokumenti uredskih aplikacija i CAD sustava.

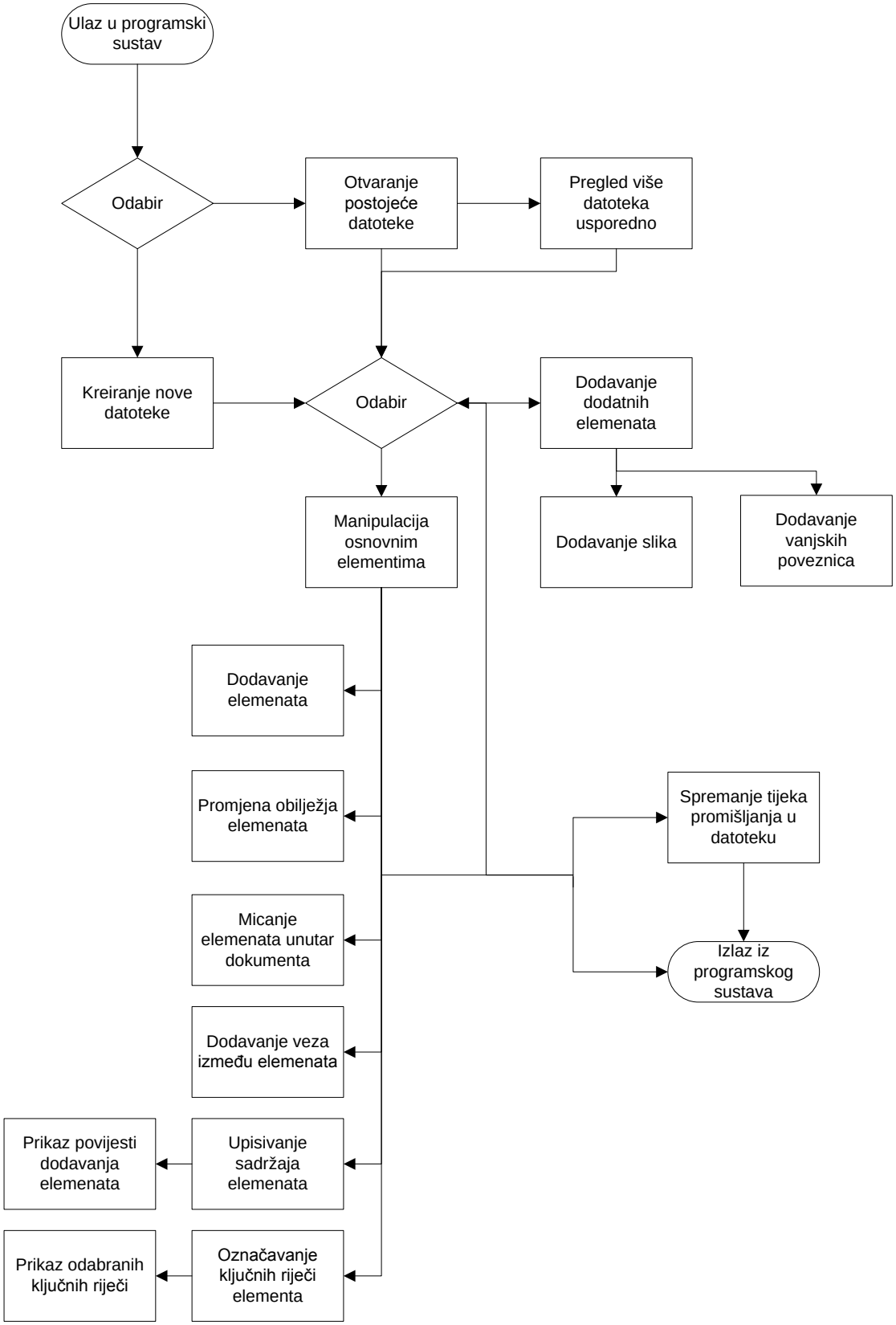
6.4 INDEKSIRANJE ELEMENATA GRAFIČKOG PRIKAZA U SVRHU PRETRAŽIVANJA

Za potrebe pretraživanja baze pohranjenih dokumenata tijeka promišljanja i odlučivanja, potrebno je osmisliti način indeksiranja sadržaja, tj. informacija unutar elemenata. Kako u ovom radu to nije primarni cilj, bit će predložen jedan od načina.

Korisnici, dodaju elemente, smještaju ih u dokumentu, dodaju im relacije, te eventualno dodaju slike, ili poveznice sa vanjskim dokumentima. Nakon toga, potrebno je omogućiti pronalaženje sadržaja određenog elementa, te označavanje njegovog sadržaja kao ključne riječi. Na taj način, sadržaj odabranog elementa može se pohranjivati u posebnu, ili zajedničku bazu zajedno sa vezom na dotični dokument. Kada bi se kasnije željelo pronaći određeni dokument, osim pomoću naziva dokumenta, to bi bilo moguće i pomoću ključnih riječi specifičnih za pojedini dokument. Na taj način, konstruktori dobivaju uvid u kojim projektima su se rješavali slični problemi, što može poslužiti za generiranje ideja za rješavanje postojećih problema.

6.5 FUNKCIJE PROGRAMSKOG SUSTAVA

Uz ranije navedene zahtjeve i funkcije grafičkog prikaza, cijeli programski sustav trebao bi biti u mogućnosti pohraniti dinamički dodane elemente grafičkog prikaza tijeka promišljanja u obliku datoteke. Osim pohranjivanja, korisnik mora imati mogućnost ponovnog otvaranja pohranjene datoteke i naknadnog mijenjanja njenog sadržaja. Za potrebe ovog rada, programski sustav sprema grafički prikaz u obliku datoteke na odabrano mjesto na tvrdom disku računala, te nije direktno integriran u PDM sustav kako bi se mogao koristiti i bez njega. Shema funkcija i mogućnosti sustava prikazana je na slici 21..

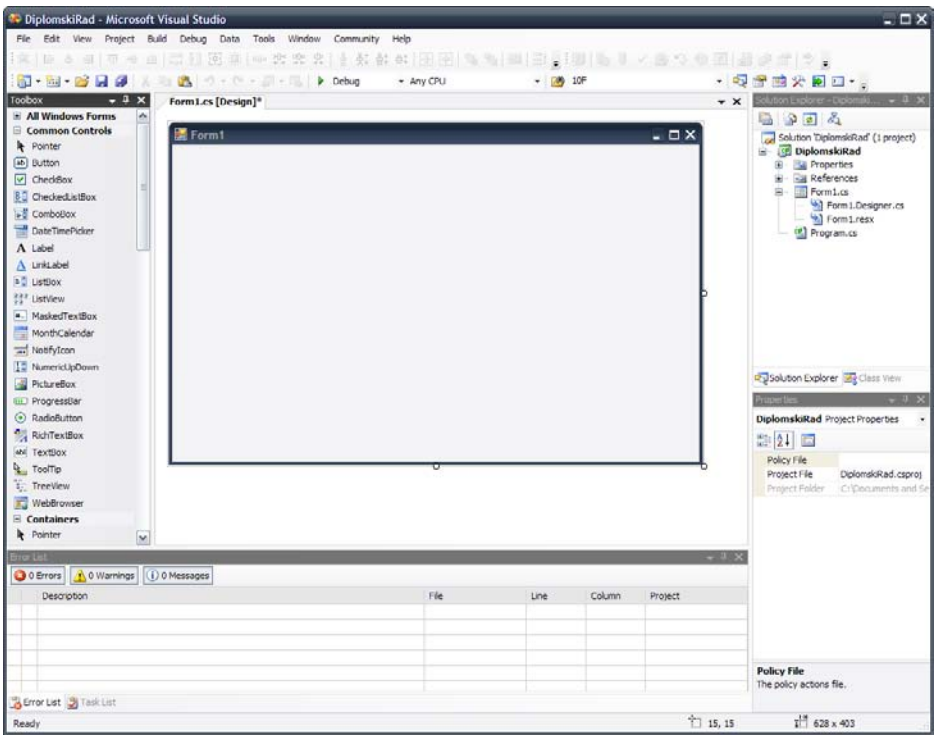


Slika 21. Funkcije i mogućnosti programskog sustava

7 IZRAĐENI PROGRAMSKI SUSTAV

7.1 ALAT ZA IZRADU

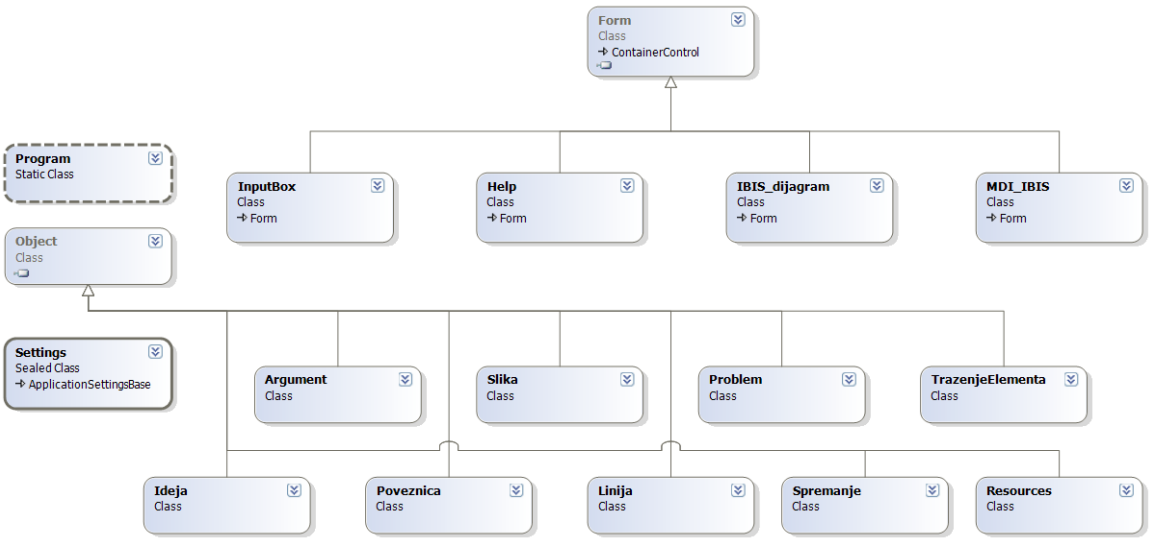
Programski sustav realiziran je upotrebom alata za objektno programiranje *C Sharp,* ili, kao što se uobičajeno piše *C#*, koji je dio Microsoftovog *Visual Studio 2005* paketa programerskih alata. C# je programski jezik razvijen od strane Microsoft Corporation, kao dio *.NET* inicijative. Prema [25], C# je objektno orijentiran jezik za opću namjenu, koji vuče korijene iz poznatijih programskih jezika C++ i Java, koji su utjecali na njegov razvoj. Ovaj programski jezik odabran je zbog ponešto iskustva u radu sa njime, te je smatrano kako se pomoću njega mogu izvesti zahtijevane funkcije. Na slici 22. prikazano je početno sučelje ovog alata. U izradi programskog sustava kao pomoć je korištena literatura [26],[27],[28] i [29].



Slika 22. Početno sučelje C# alata

7.2 POJAŠNJENJE DIJELOVA I FUNKCIJA PROGRAMSKOG SUSTAVA

Programski sustav realiziran je na načina da postoji glavna forma, i ostale forme, koje predstavljaju dokumente. Između njih je ostvarena veza *roditelj-dijete*, tj. glavna forma kontrolira sve dokumente koji se kreiraju i otvaraju unutar nje. Sustav je realiziran na način da su pojedini elementi definirani unutar vlastitih klasa, a metode kojima se ostvaruju željene funkcije su osim u svojim klasama, definirane unutar formi *roditelj* i *dijete*. Na slici 23. prikazane su klase definirane u okviru programskog sustava, bez metoda i svojstava koje one sadrže.



Slika 23. Klase definirane u programskom sustavu

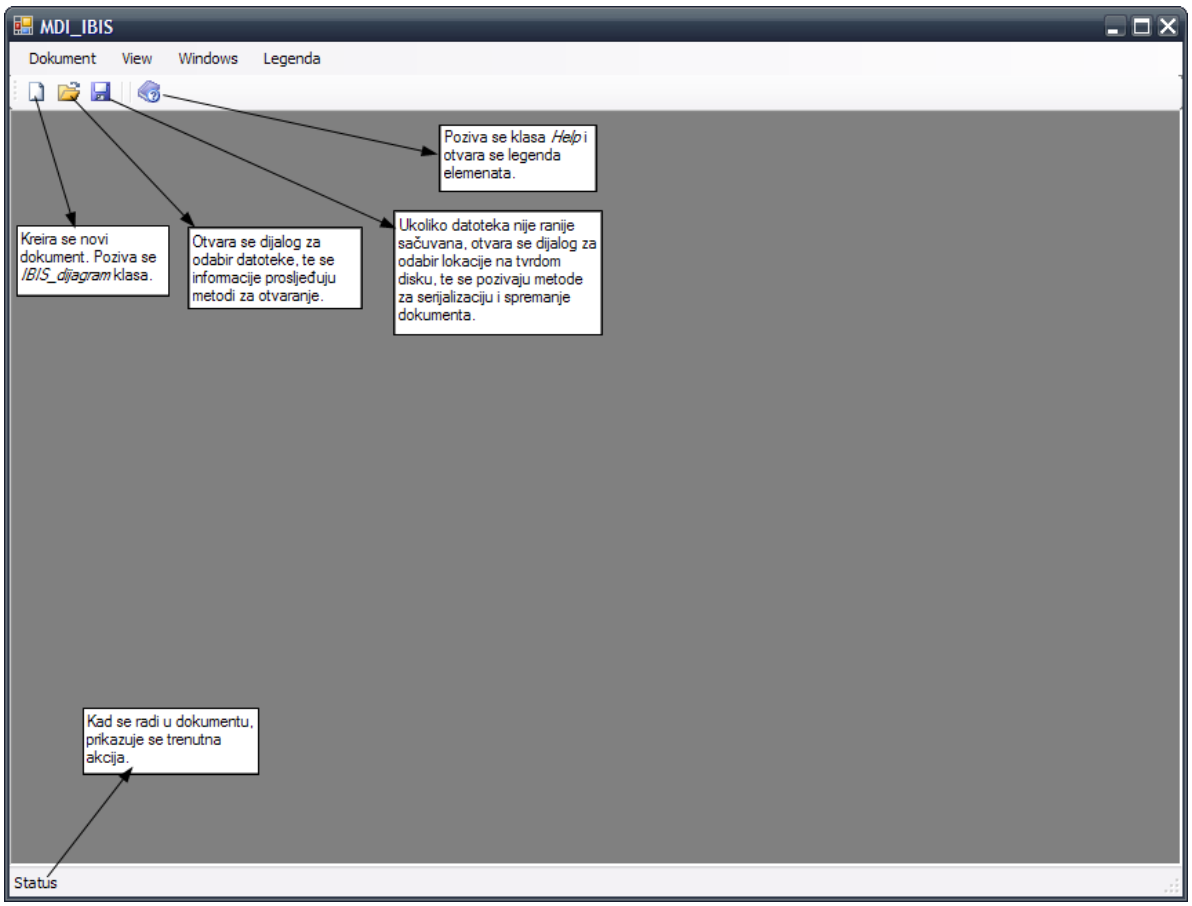
U nastavku će u radu biti pojašnjene pojedine klase koje predstavljaju elemente grafičkog prikaza s obzirom na svoju ulogu u prikazu, i metode koje one koriste za ostvarivanje funkcija.

7.2.1 DIJELOVI SUSTAVA

Programski sustav sastoji se od više dijelova, a osnovica je glavna forma, unutar koje se prikazuju svi dokumenti (IBIS dijagrami). Osim glavne forme, u sustavu postoji *IBIS_dijagram* forma,koja predstavlja dokument koji se kreira. On kao dijelove sadrži osnovne elemente (*problem, ideja, argument*) i dodatne elemente (*slika, vanjska poveznica*). Svi navedeni dijelovi bit će ukratko opisani u nastavku.

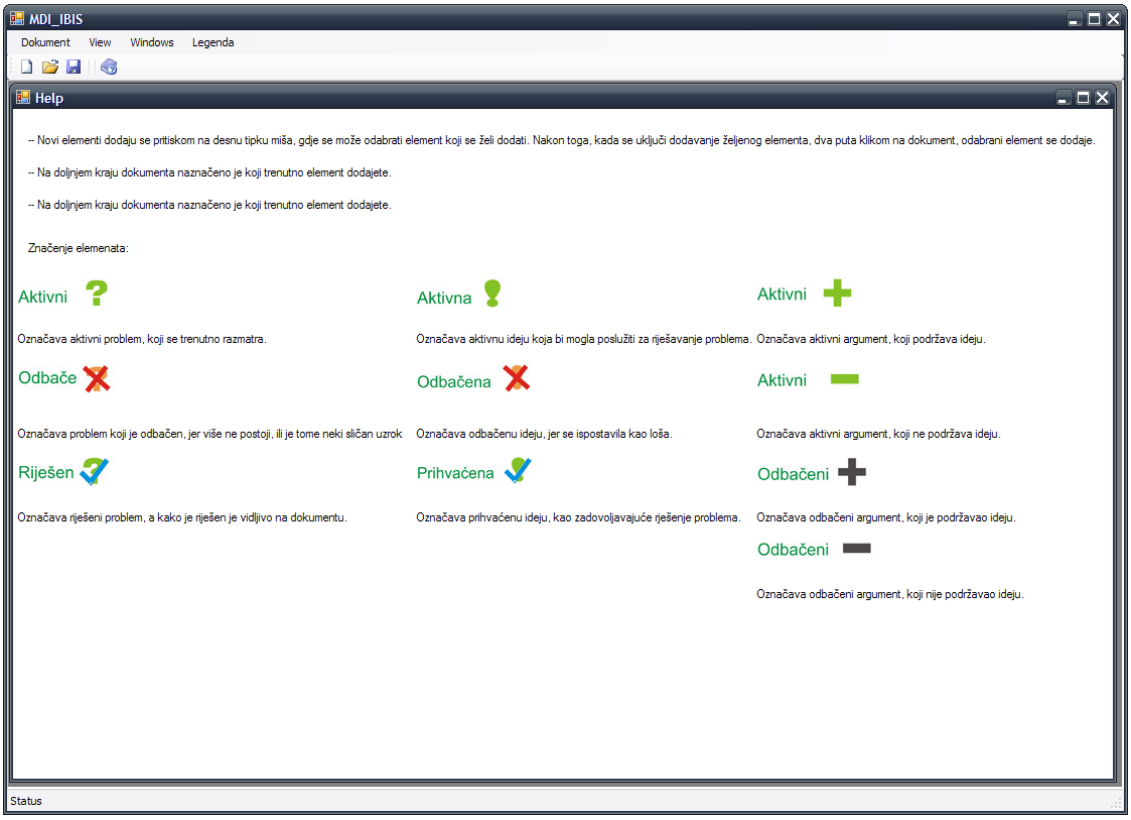
7.2.1.1 Glavna forma

Ova forma sadrži metode koje pozivaju neke definirane klase i metode za spremanje, otvaranje i razmještaj dokumenata unutar glavne forme. Programski kod forme sadržan je u klasi „MDI_IBIS“, a na slici 24 prikazan je izgled forme, sa kratkim pojašnjenjima funkcija.



Slika 24. Glavna forma

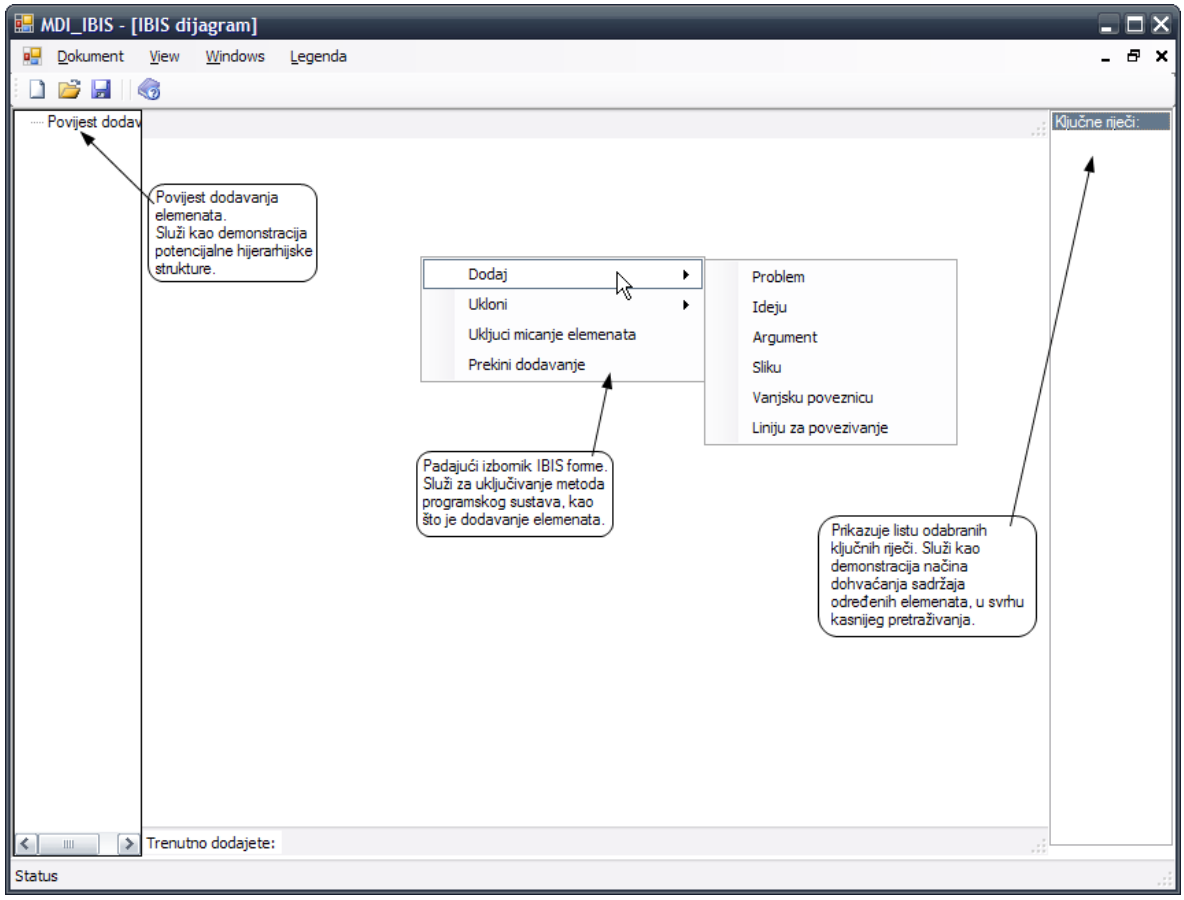
Unutar glavne forme prikazuju se i otvaraju svi *IBIS_dijagram* dokumenti, otvaraju se postojeće datoteke, omogućuje spremanje dijagrama u datoteku i omogućuje razmještaj otvorenih datoteka. Osim navedenog, iz glavne forme moguće je otvoriti *legendu* u kojoj je ukratko objašnjeno na koji način se manipulira elementima, te što predstavljaju grafičke oznake pojedinog elementa. Budući da sama forma legende nije od velike važnosti, njezin izgled prikazan je u ovom dijelu zajedno sa glavnom formom na slici 25.



Slika 25. Glavna forma s formom *Help* (legenda)

7.2.1.2 IBIS dijagram forma

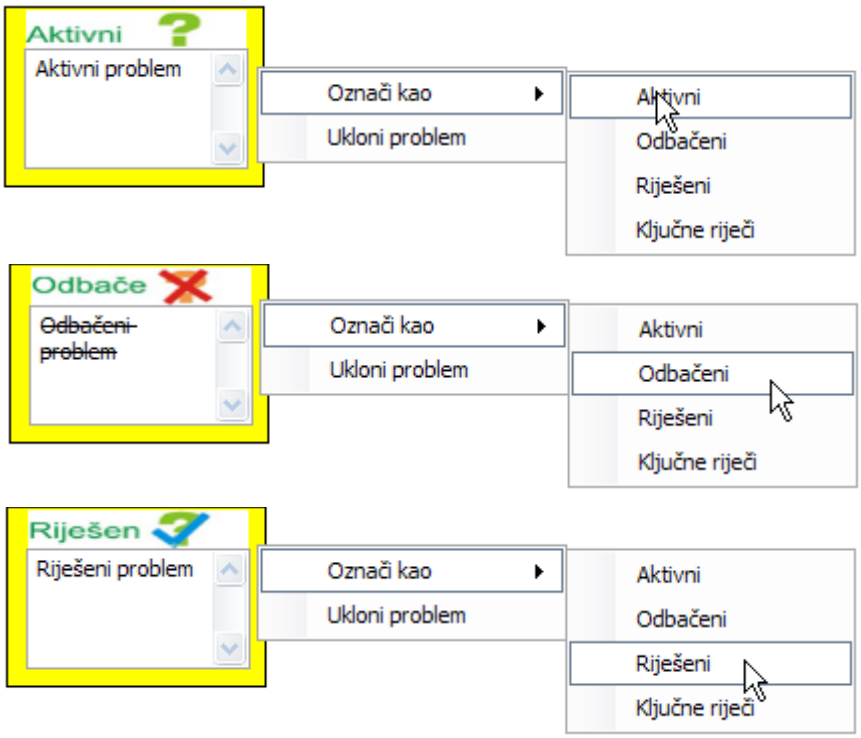
IBIS dijagram predstavlja dokument koji se kreira i u kojem se bilježi tijek promišljanja konstruktora. Datoteke koje se pohranjuju su zapravo IBIS forme, koje sadrže elemente i funkcije za manipulaciju tim elementima. Prilikom kreiranja dokumenta, forma je potpuno prazna, bez kontrola. Tome je tako zbog toga što se željelo ostaviti što više prostora za snimanje samog tijeka promišljanja na dokumentu, a elementi se dodaju i manipuliraju korištenjem padajućih izbornika, o čemu će biti više riječi kasnije. Programski sustav prepoznaje kada korisnik želi dodati novi element, ili uključiti neku od funkcija sadržanih u kôdu IBIS forme, te tada nudi odgovarajući padajući izbornik. Na slici 26. prikazana je *IBIS_dijagram* forma unutar glavne forme programskog sustava, sa pripadajućim padajućim izbornikom. Na slici 26. e također vidljivo da na formi sa svake strane postoji stupac. U lijevi stupac pohranjuje se vrsta i sadržaj elemenata, što služi kao demonstracija potencijalne hijerarhijske strukture elemenata, dok se trenutno može koristiti kao pregled povijesti dodavanja i upisivanja sadržaja u elemente. U desnom stupcu na formi pohranjuju se odabrane ključne riječi, što je u slučaju ovog rada također demonstracija načina na koji se sadržaj točno određenih elemenata može preuzeti, a kasnije iskoristiti kao ključne riječi za potrebe pretraživanja.



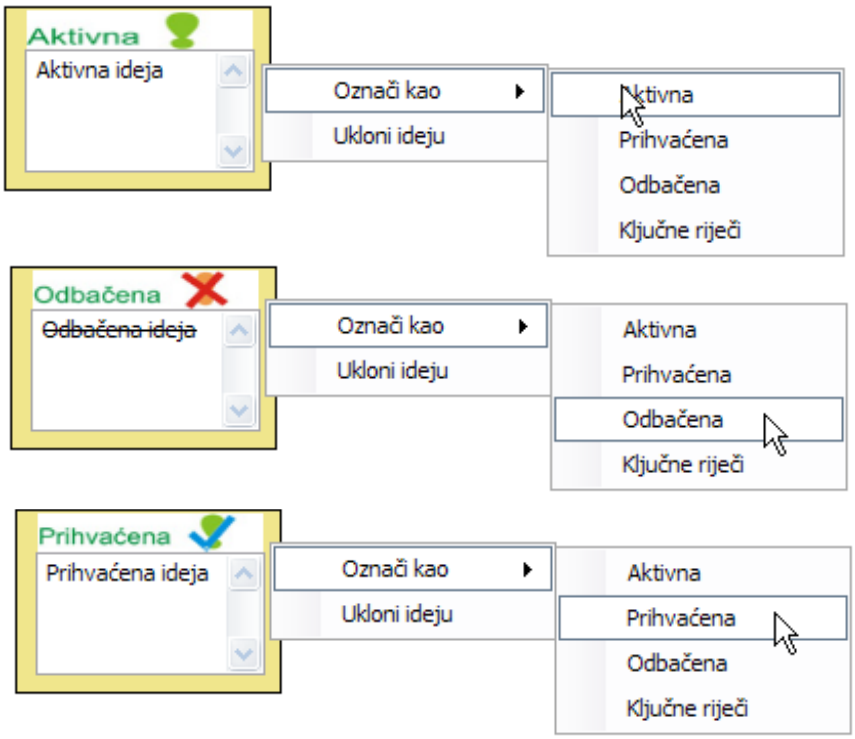
Slika 26. IBIS forma - dokument

7.2.1.3 Osnovni IBIS elementi

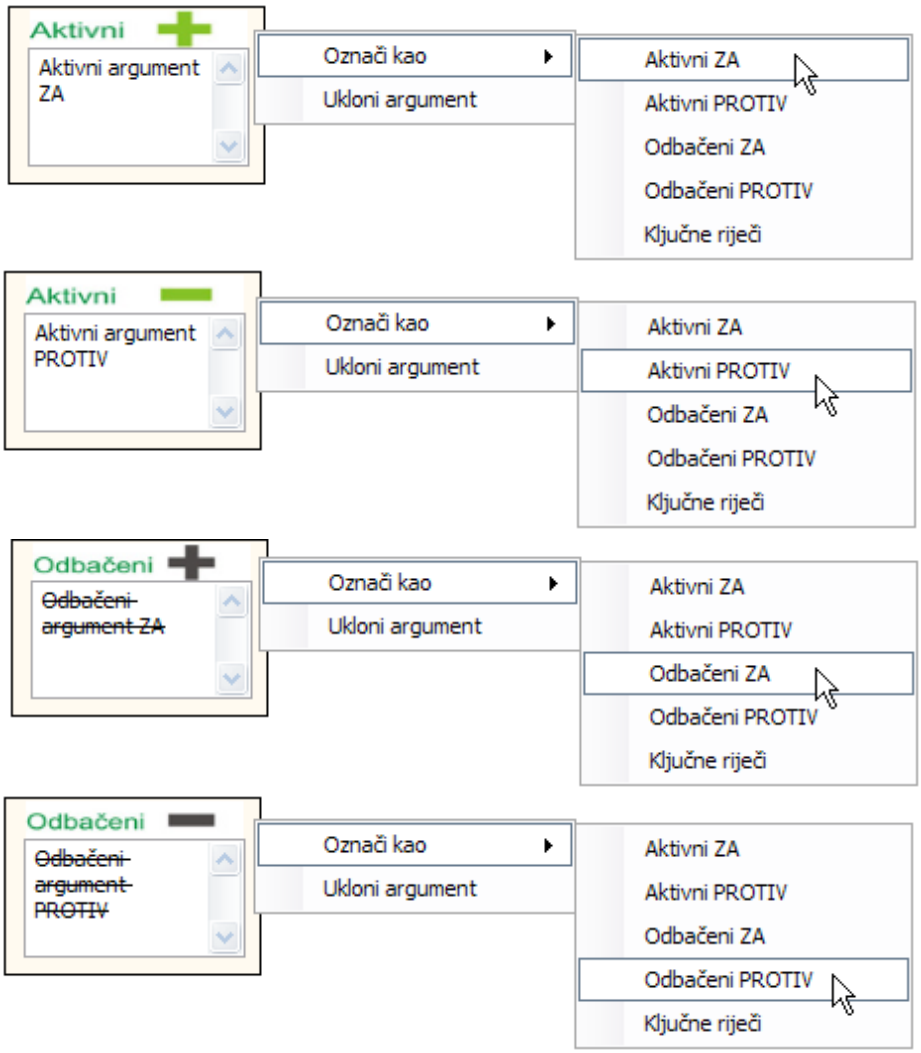
Svi elementi na *IBIS dijagram* formu dodaju se korištenjem padajućeg izbornika pritiskom desne tipke miša. Isto tako, za promjenu obilježja elementa, prilikom pritiska desne tipke miša programski sustav prepoznaje vrstu elementa te nudi pripadajući izbornik za promjenu obilježja. Na slici 27. prikazan je osnovni element *problem* sa pripadajućim padajućim izbornikom i mogućim obilježjima, na slici 28. prikazan je osnovni element *ideja* sa pripadajućim padajućim izbornikom, dok je na slici 28. prikazan elemen *argument* zajedno sa svojim padajućim izbornikom i mogućim obilježjima. U osnovne elemente prikaza mogla bi se ubrajati i linija za povezivanje elemenata, no kako ona zapravo nije poseban element, bit će vidljiva samo u primjeru. Linija za povezivanje dodaje se uključivanjem metode na IBIS formi i odabirom elemenata koji se žele povezati.



Slika 27. *Problem* sa mogućim obilježjima



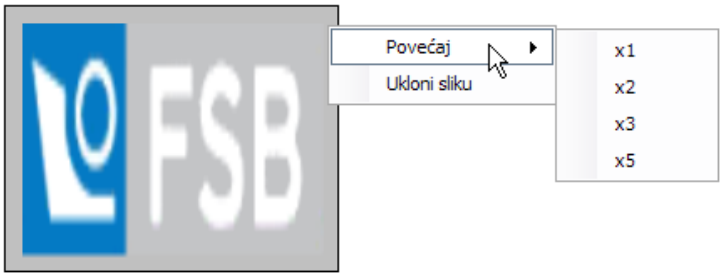
Slika 28. *Ideja* sa mogućim obiljžjima



Slika 29. *Argument* sa mogućim obilježjima

7.2.1.4 Dodatni elementi

Osim osnovnih IBIS elemenata u programskom sustavu postoje i ranije spomenuti elementi *slika* i *vanjska poveznica*, koji su zajedno sa pripadajućim padajućim izbornicima prikazani na slikama 30. i 31.



Slika 30. Primjer elementa *slika*

Slika 31. Primjer elementa *vanjska poveznica*

7.2.2 FUNKCIJE PROGRAMSKOG SUSTAVA I METODE IZVOĐENJA

Važnije funkcije programskog sustava su:

- Kreiranje novih dokumenata
- Dodavanje elemenata IBIS dijagrama
- Povezivanje elemenata
- Promjena obilježja elemenata
- Dohvaćanje sadržaja elemenata
- Micanje elemenata
- Uklanjanje elemenata
- Spremanje dokumenata
- Otvaranje dokumenata

Metode koje izvršavaju zadane funkcije bit će ukratko objašnjene u nastavku, s naznakama klasa uz koje su vezane i za važnije funkcije pojašnjenjem dijela programskog kôda, a sve klase s metodama i varijablama koje su zadužene za izvođenje funkcija programskog sustava prikazane su na slici 32.. Budući da funkcije povezivanja elemenata, promjena obilježja, dohvaćanja sadržaja, micanja i uklanjanja elemenata koriste sličnu metodu pronalaska točnog elementa, one će biti zajedno objašnjene. Na slici 32. klasa *IBIS_dijagram* zbog svoje veličine prikazana je u dva dijela.

Slika 32. Pregled svih korištenih klasa i metoda

7.2.2.1 Kreiranje novih dokumenata

Metoda za kreiranje novih dokumenata nalazi se unutar klase *MDI_IBIS*, a nakon što se pokrene, poziva se klasa *IBIS_dijagram* u kojoj se inicijaliziraju spremnici elemenata i varijable potrebne za kasniju manipulaciju elementima grafičkog prikaza.

Osnovni kôd za kreiranje novog dokumenta je:

```
IBIS_dijagram childForm = new IBIS_dijagram();
childForm.MdiParent = this;
childForm.Text = nazivIBISdokumenta;
childForm.Show();
```

prilikom čega se inicijaliziraju spremnici koji se koriste za pohranjivanje podataka o elementima:

```
spremnikArgumenti = new ArrayList();
spremnikIdeje = new ArrayList();
spremnikProblemi = new ArrayList();
spremnikPoveznica = new ArrayList();
spremnikSlika = new ArrayList();
spremnikLinijaTemp = new ArrayList();
spremnikLinija = new ArrayList();
```

7.2.2.2 Dodavanje elemenata IBIS dijagrama

Za dodavanje elemenata zadužen je kôd unutar klase *IBIS_dijagram* koji poziva željenu klasu elementa i potom prikazuje na dokumentu. Prilikom dodavanja elemenata, uključuje se metoda za dodavanje željene vrste elementa, a element se smješta ne proizvoljno mjesto na dokumentu. Osnovni kôd za dodavanje elementa na primjeru dodavanja *argumenta* je:

```
if (dodajArgument==true & e.Button == MouseButtons.Left)
      {
          Argument argument = new Argument();
          argument.Location = e.Location;

          argument.ArgumentClass(this.Controls);

          argument.mArgumentTxtBox.TextChanged += txt_TextChanged;
          argument.mArgumentTxtBox.Leave += txt_Leave;
          argument.mArgumentPanel.MouseClick += mPanel_MouseClick;
          argument.mArgumentPanel.MouseDown += mPanel_MouseDown;
          argument.mArgumentPanel.MouseUp += mPanel_MouseUp;
          argument.mArgumentPanel.MouseMove += mPanel_MouseMove;
```

```
            argument.Text = argument.mArgumentTxtBox.Text;
            argument.mArgumentTxtBox.Enabled = true;
            argument.TxtState = argument.mArgumentTxtBox.Enabled;

            spremnikArgumenti.Add(argument);
        }
```

Prilikom dodavanja novog elementa, uz njega se dodaju također i metode za kontrolu događaja nad elementom, kao što je pritisak tipke miša.

7.2.2.3 Metode promjene obilježja, povezivanja, dohvaćanja, micanja i uklanjanja elemenata

Različito kod ovih metoda je ponajprije način uključivanja. Dok se metode povezivanja elemenata i micanja elemenata uključuju izravno iz padajućeg izbornika dokumenta, metode promjene obilježja, dohvaćanja sadržaja elementa i metoda uklanjanja elementa specifične su za svaku vrstu elemenata, tj. aktiviraju se posebno za svaku vrstu elemenata.

Zajedničko svim navedenim funkcijama je da moraju prvo točno pronaći i ustanoviti koji element je trenutno potreba. Kako bi se to ispravno i izvelo, brine se više metoda, no osnova je prolazak kroz članove dodane u inicijalizirane spremnike i pronalazak člana koji sustavu šalje neku informaciju. Skup metoda za traženje točnog elementa nalazi se u klasi *TrazenjeElementa*, a na primjeru *argumenta* osnovni kôd za traženje je:

```
for (int index = 0; index < sprem.Count; index++)
{
       Argument trenutniArgumentListe = (Argument)sprem[index];

       if ((object)trenutniArgumentListe.mArgumentPanel == sender)
       {
       trenutniIndexArgumenta = index;
       postojiArgument = true;
       }
}

if (trenutniIndexArgumenta == -1)
return;

trenutniArgument = (Argument)sprem[trenutniIndexArgumenta];
```

Kada je pronađen trenutni element (u ovom slučaju *argument*), pokreću se ostale metode, ovisno o tome koja je prije procesa traženja bila uključena. Kako bi navođenje primjera osnovnog kôda za svaku pojedinu funkciju bilo bespotrebno gomilanje podataka, cijeli kôd dodan je kao prilog radu.

7.2.2.4 Spremanje dokumenata

Glavna metoda za spremanje dokumenata nalazi se u klasi *Spremanje*, gdje se prikupljeni podaci serijaliziraju i pohranjuju u datoteku u binarnom obliku. Kako bi spremanje uopće bilo moguće, svaki objekt i metoda pojedine klase moraju biti prilagođeni za serijalizaciju. No, iako se svi objekti klase označe kao serijalizabilni, C# nije u mogućnosti provesti serijalizaciju nad objektima kao što su kontrole i svojstva kontrola. Zbog toga je potrebno u binarnu datoteku serijalizirati varijable i objekte koji opisuju elemente koje želimo pohraniti, kako bi se oni kasnije mogli prilikom otvaranja generirati u obliku u kojem su spremljeni. Podaci koji se spremaju pohranjeni su u generičku listu koja je popunjena listama svih dodanih elemenata, te se tada vrši serijalizacija generičke liste u binarnu datoteku. Budući da je dokument koji se u programskom sustavu izrađuje zasnovan na IBIS sustavima, pridodana mu je ekstenzija **.ibs,* koja podsjeća na naziv sustava logičke podrške. Osnovni kôd za spremanje u obliku binarne datoteke je:

```
if (filename != null)
{
Stream saveStream = File.Create(filename);

   try
   {

       IFormatter sFormatter = new BinaryFormatter();
       sFormatter.Serialize(saveStream, glavniSpremnik);
   }
   catch (IOException ex)
   {
       MessageBox.Show(ex.Message);
   }
   finally
   {
       saveStream.Close();
   }
}
```

7.2.2.5 Otvaranje dokumenata

Otvaranje pohranjenih dokumenata moguće je postići na dva načina. Prvi i jednostavniji (što se programerske strane tiče) je otvaranje iz glavne forme. Tada se nad pohranjenim dokumentom vrši deserijalizacija metodama koje se nalaze u klasi *IBIS_dijagram*. Nakon deserijalizacije dobiva se ranije pohranjena generička lista koja sadrži podatke o svim elementima koji bi se ponovno trebali prikazati na dokumentu. No, za prikazivanje svakog ranije pohranjenog elementa potrebno je iz generičke liste kreirati

liste elemenata, iz kojih se tada na osnovu spremljenih informacija generiraju elementi na dokumentu. Osnovni kôd za deserijalizaciju pohranjene generičke liste je

```
if (filename != null)
{
   Stream openStream = File.Open(filename, FileMode.OpenOrCreate);

   try
   {
      IFormatter oFormatter = new BinaryFormatter();
      deSpremnik =
      (List<ArrayList>)oFormatter.Deserialize(openStream);
   }
   catch (IOException ex)
   {
      MessageBox.Show(ex.Message);
   }
   finally
   {
      openStream.Close();
   }

   dokumentSacuvan = true;
   filenameIsti = filename;
   postojiDatoteka = true;

   OnDeserialization(sender, e);
}
```

Kao što je vidljivo iz priloženog kôda, nakon deserijalizacije generičke liste poziva se metoda „*OnDeserialization*“, koja je zadužena za daljnje izvlačenje informacija iz generičke liste i kasnije generiranje elemenata u dokumentu.

Ukoliko se dokument otvara izvan glavne forme programskog sustava, tj. popularno dvostrukim klikom na već spremljeni dokument, u klasi *Program* provjerava se koji je dokument zatražio otvaranje programskog sustava, te se tada informacije o datoteci šalju dalje prema glavnoj formi, što u osnovi obavlja sljedeći dio kôda:

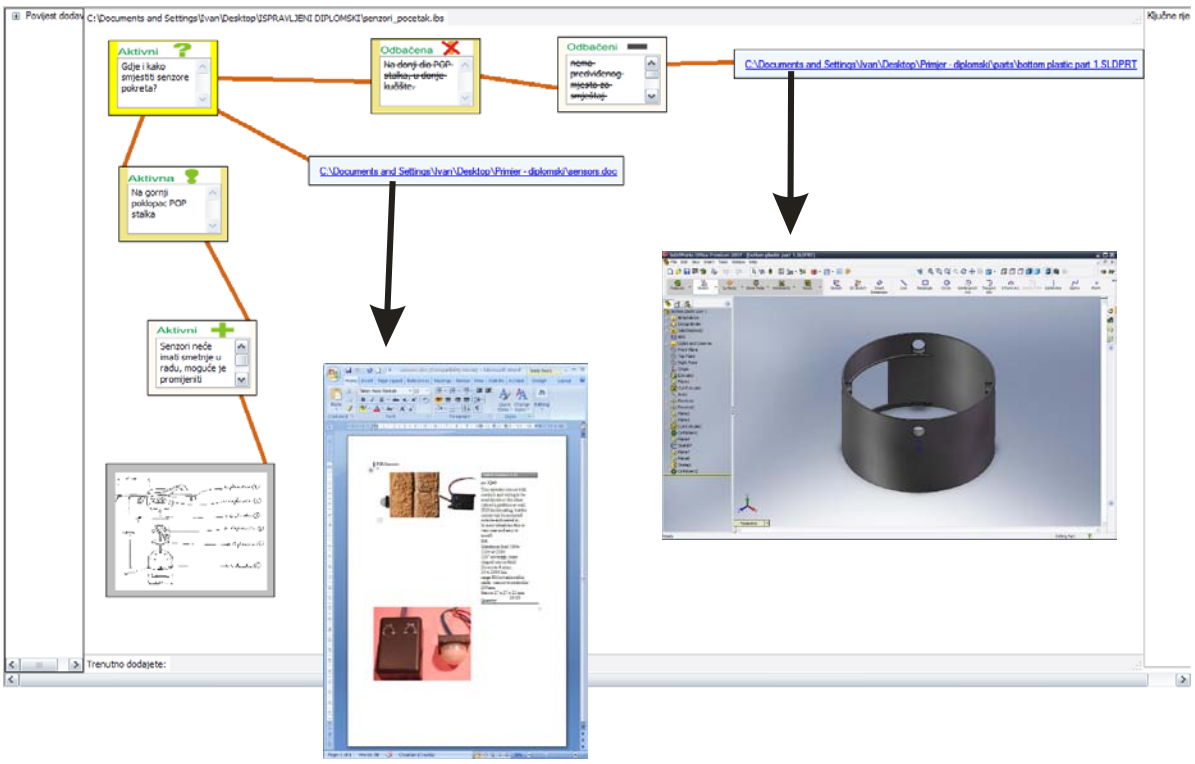
```
string odabraniFile;

if (args.Length > 0)
{
   odabraniFile = Convert.ToString(args[0]);
}
else
{ odabraniFile = null; }

Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);

MDI_IBIS app = new MDI_IBIS();
app.OdabraniFile = odabraniFile;
Application.Run(app);
```

Ukoliko je na glavnoj formi primijećeno da je otvaranje datoteke zatraženo izvan programskog sustava, kreira se novi dokument kojem se prosljeđuju informacije o dokumentu. Nakon toga, metodama za učitavanje datoteke sličnim ranije navedenima učitavaju se elementi datoteke. Dio kôda koji je primarno zadužen za provjeravanje i prosljeđivanje informacija o datoteci koja je zatražila otvaranje programskog sustava je:

```
if (OdabraniFile != null)
{
   string Odabrani = OdabraniFile;
   IBIS_dijagram childForm = new IBIS_dijagram();
   childForm.MdiParent = this;
   childForm.odabrani = Odabrani;
   childForm.Text = Odabrani;
   childForm.Show();
}
```

8 PRIMJER KORIŠTENJA REALIZIRANOG SUSTAVA

Za primjer korištenja realiziranog programskog sustava preuzet je problem iz EGPR[15] projekta, iz godine 2007. Dokumenti korišteni u primjeru djelo su grupe 2, koje sam bio član. Jedan od problema koji se javio bio je vezan uz smještaj senzora pokreta. Bilo je potrebno odabrati vrstu senzora i smjestiti je na prikladno mjesto na POP[16] stalku. Nakon dodavanja glavnog problema *„gdje i kako smjestiti senzore pokreta“*, moguće je generirati neke ideje koje je potrebno argumentirati. Uz problem je dodana poveznica s vanjskom datotekom koja opisuje vrstu senzora, kako bi se dobilo uvid u zahtjeve za konstrukciju. Uz argumente su dodane poveznica s vanjskom datotekom i slika, koje pobliže pojašnjavaju postavljene argumente, što je prikazano na slici 33.



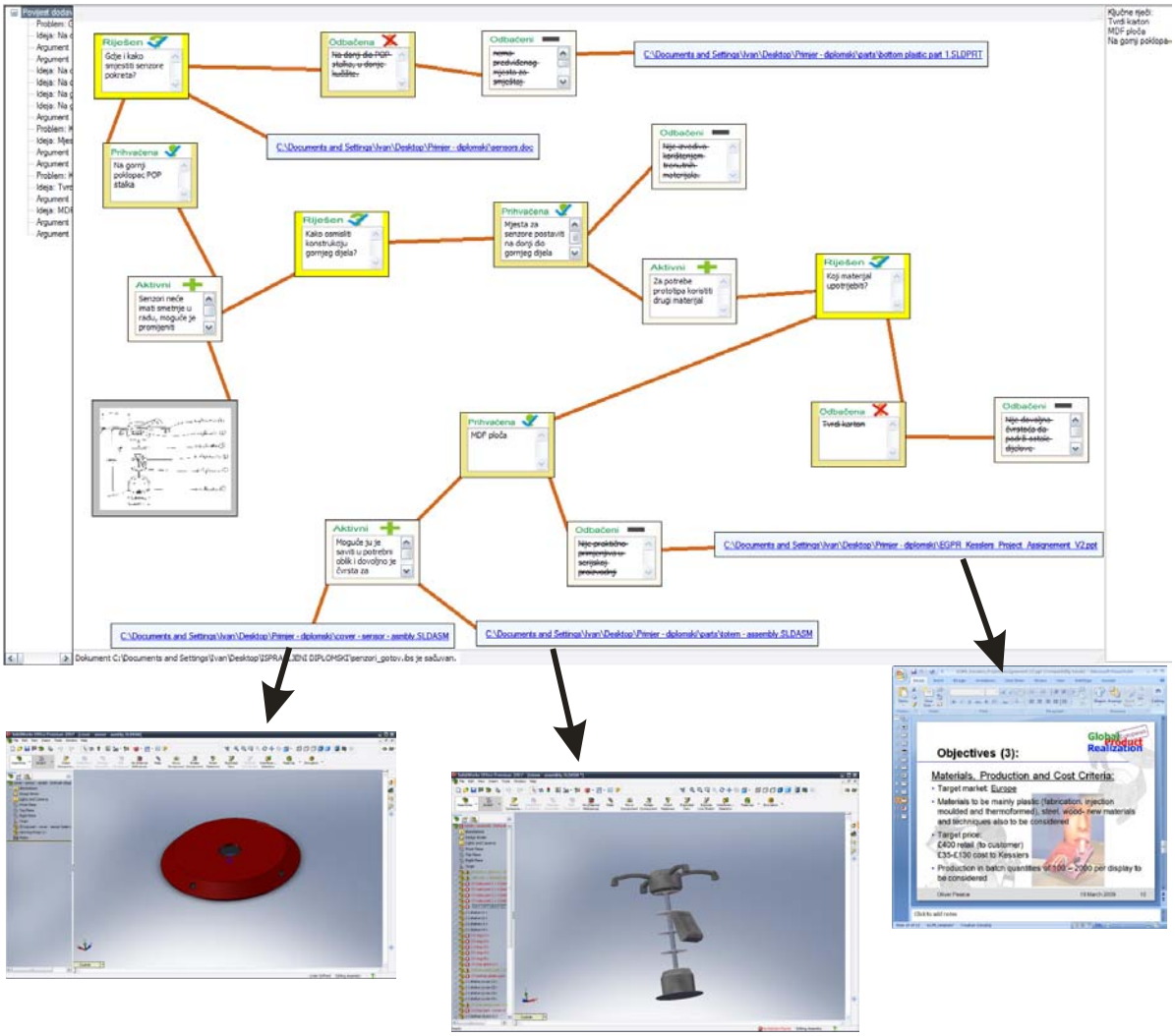
Slika 33. Primjer dodanih elemenata

[15] EGPR (eng. European Global Product Realisation) =

[16] POP (eng. Point of Purchase) = prodajno mjesto, stalak s proizvodima

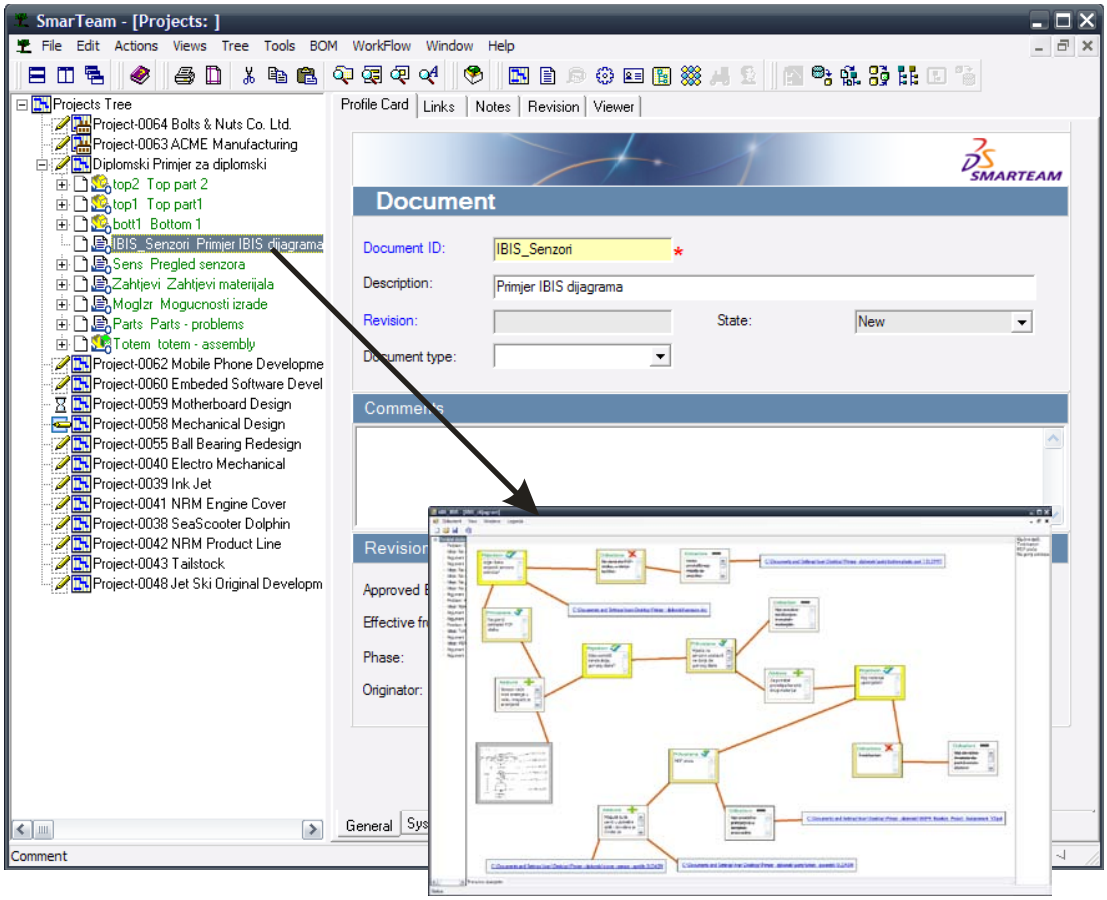
Na slici 33 prikazani su i vanjski dokumenti koji se pritiskom na poveznicu unutar IBIS dokumenta prikazuju u svojim aplikacijama.

Daljnjim dodavanjem elemenata IBIS sustava nastaje konačni dijagram tijeka odlučivanja. Na slici 34. prikazan je cjelokupni tijek promišljanja i odlučivanja za odabrani primjer. Na slici 34. također su prikazane vanjske datoteke otvorene u svojim aplikacijama pritiskom na odgovarajuću poveznicu.



Slika 34. Gotov IBIS dokument za odabrani primjer

U svakoj fazi izrade IBIS dijagrama moguće je pohraniti dokument i dodati ga u korišteni PDM sustav. U slučaju ovog primjera završeni IBIS dijagram dodan je u PDM sustav kao nova vrsta dokumenata unutar projekta u kojem su i ostali relevantni dokumenti. Primjer projekta s dodanim datotekama, zajedno sa IBIS dijagramom prikazanje na slici 35.



Slika 35. IBIS dijagram pokrenuti iz SmarTeam PDM sustava

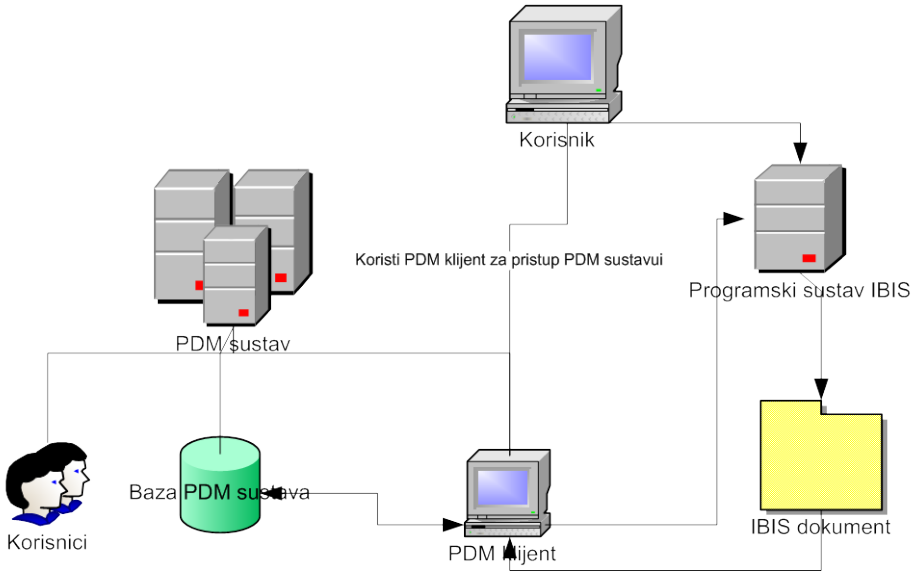
Kako bi se IBIS dijagram mogao otvoriti iz PDM sustava, potrebno je dodati poveznice na novu vrstu dokumenata u PDM sustav, kao što je to ranije u radu prikazano. Ukoliko se tada unutar PDM sustava želi otvoriti IBIS dijagram, on se otvara u izrađenom programskom sustavu, kao što je prikazano na slici 35., te je spreman za daljnje mijenjanje i ponovno spremanje.

9 MOGUĆNOSTI PROMJENE I UNAPRJEĐENJA PROGRAMSKOG SUSTAVA

9.1 TRENUTNO STANJE PROGRAMSKOG SUSTAVA

Budući da je sustav izrađen u alatu C# programskog sustava Visual Studio 2005,koji radi unutar *.NET 2.0* softverskog okvira, kako bi sustav funkcionirao potrebno je na računalu na kojem se pokreće imati učitan isti taj softverski okvir. Iako se sustav mogao izraditi da bude direktno pokretan unutar PDM sustava, budući da je cilj ovoga rada stvoriti osnovnu podlogu za daljnju izradu sustava za praćenje i pohranjivanje tijeka promišljanja i odlučivanja, ostvareni programski sustav trenutno je samostalna aplikacija koja nije direktno integrirana unutar PDM sustava. Na taj način, moguće je testirati i mijenjati programski sustav bez posjedovanja PDM sustava, ili klijenta PDM sustava na računalu. Realiziran programski sustav u stanju je u obliku datoteke pohraniti IBIS dijagrame, te nakon toga učitati pohranjene datoteke, koje su nakon toga ponovno spremne za mijenjanje i dorađivanje. U realiziranom sustavu također su postavljeni preduvjeti za kreiranje baze za pretraživanje korištenjem ključnih riječi,te generiranje hijerarhijske strukture osnovnih elemenata.

U trenutnom stanju, korisnik se na PDM sustav spaja pomoću PDM klijenta. Isti korisnik, ukoliko na računalu ima postavljen programski sustav za IBIS dijagrame, koristi taj sustav za kreiranje novih IBIS dokumenata. Nakon toga, korisnik preko PDM klijenta dodaje kreirani IBIS dokument u bazu podataka PDM sustava. Prilikom otvaranja IBIS dokumenta iz PDM sustava otvara se programski sustav IBIS kojeg korisnik mora posjedovati na računalu, te se interpretira i prikazuje dokument. Odnos realiziranog sustava prema PDM sustavu korištenom u primjeru shematski je prikazan na slici 36.



Slika 36. Shema odnosa IBIS i PDM sustava

9.2 MOGUĆNOSTI DALJNJEG RAZVOJA

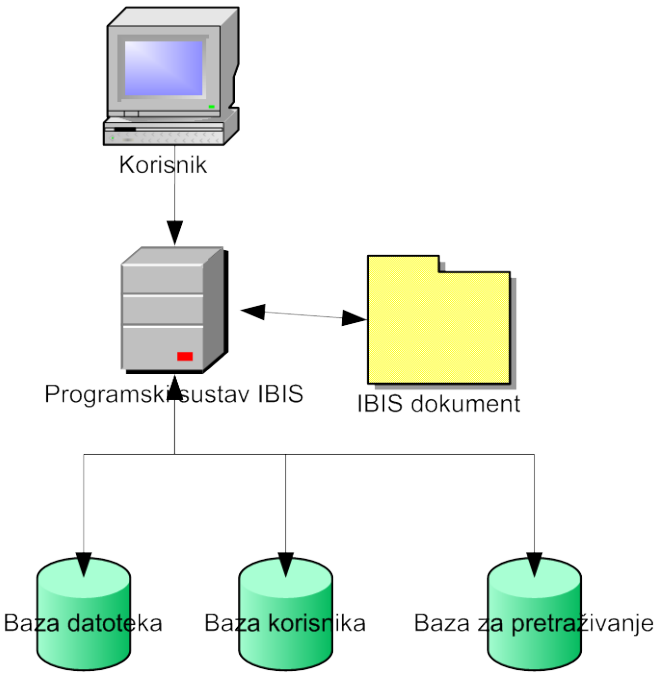
Kod odabira smjera daljnjeg razvoja programskog sustava moguće je krenuti u dva smjera. Prvi bi bio jednostavniji, i možda u ovoj fazi logičniji, razvoj samostalnog programskog sustava, dok je drugi pokušaj implementacije programskog sustava kao sastavni dio PDM sustava, kao što je to slučaj sa nekim CAD alatima za modeliranje.

9.3 RAZVOJ SAMOSTALNOG SUSTAVA

Za daljnji razvoj samostalnog sustava, osim nekih dorada na trenutnom programskom sustavu, potrebno je projektirati i adekvatne baze podataka. Tada bi bila potrebna baza korisnika, baza dokumenata, i baza za pretraživanje. Pomoću baze korisnika mogla bi se izraditi proizvoljna struktura korisnika, čime bi se omogućilo provjeravanje uloge korisnika unutar IBIS sustava. Na taj način mogla bi se kreirati struktura korisnika koji mogu:

- Mijenjati cijele dokumente.
- Mijenjati samo neke elemente.
- Samo dodavati nove elemente, ali ne i mijenjati postojeće.
- Dodavati samo posebne elemente, bez mogućnosti mijenjanja postojećih.
- Samo pregledavati postojeće dokumente.

Za potrebe pretraživanja baze IBIS dokumenata može se koristiti predložen model ključnih riječi, gdje se tada u tablici baze za pretraživanje pohranjuju ključne riječi zajedno sa pokazivačem na dokument od kojeg potječu. Shema takvog sustava prikazana je na slici 37..



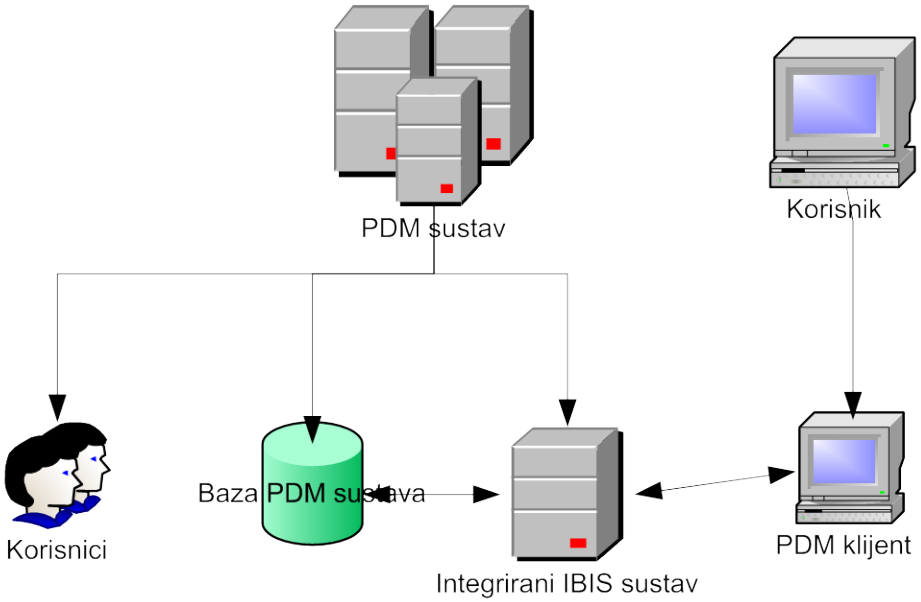
Slika 37. Samostalni programski sustav IBIS

9.4 INTEGRIRANI IBIS SUSTAV

Ukoliko bi se u daljnjem razvoju krenulo u smjeru integriranja IBIS programskog sustava unutar PDM sustava, javljaju se novi izazovi i problemi. Tada je potrebno dobro proučiti metode i mehanizme koje PDM sustav koristi i provjeriti da li je uopće u njega moguće dodati dokumente čiji elementi se dinamički kreiraju. Također, jedan od problema koji je postao očit tijekom izrade ovog rada je kako preuzeti informacije o korisniku da one budu dostupne unutar dokumenta, kako bi se zadovoljila ograničenja dodavanja i mijenjanja elemenata s obzirom na ulogu korisnika. Zasad, PDM sustavi ograničavaju korisnika prilikom pristupa dokumentu na način da on, ili može mijenjati dokument, ili ga može samo pregledavati. Takav način ograničavanja korisnika za IBIS dokumente nije dovoljan, te bi se trebale kreirati procedure i metode unutar PDM sustava kojima bi se zapravo izrađivao dokument, slične metodama predstavljenim u ovom radu.

Ukoliko bi se riješili navedeni problemi ovaj pristup daljnjem razvoju svakako bi bio mnogo bolji od prethodno navedenog. Budući da današnji PDM sustavi već posjeduju module pomoću kojih se mogu unutar PDM sustava modelirati proizvodi, očigledno je da

se teži prema objedinjavanju inženjerskih alata u jednom sustavu. Shema takvog sustava prikazana je na slici 38.



Slika 38. IBIS sustav integriran u PDM sustav

10 POVEZIVANJE REALIZIRANOG SUSTAVA S OSTALIM DIJELOVIMA SUSTAVA UPRAVLJANJA ZNANJEM

Kao što je vidljivo iz prethodnih poglavlja u radu, najvažniji način povezivanja realiziranog sustava bilo bi povezivanje sa PDM sustavom. No, zanimljivije je razmisliti na koji način bi se sam dokument i njegovi sastavni elementi mogli povezati sa dijelovima sustava upravljanja znanjem.

Zanimljiva mogućnost, koja je prezentirana u programskom sustavu *DRed* koji se trenutno još uvijek razvija na sveučilištu Cambridge, je međusobno povezivanje *DRed* dokumenata (koji su također nastali prema konceptu IBIS sustava). Neke značajke spomenutog *DRed* alata opisali su u svom radu Bracwell, Gourtovaia, Wallace i Clarkson, [30]. Kako prema [30], IBIS dokumenti mogu postati vrlo veliki, čime postaju nepregledni za korisnika i komplicirani za snalaženje, ovo povezivanje nije zamišljeno na način da se otvaraju posebni dokumenti, već na način da se stvori prostor znanja konstrukcijskog odlučivanja. Na dokumentu prema [30], postoje tzv *tuneli* prema određenim mjestima na drugom IBIS dokumentu, čime se ostvaruje jednostavan i brzi pregled sličnih problema. Svaki dokument tako bi omogućio stvaranje *tunela* na proizvoljnim mjestima prema elementima drugih IBIS dokumenata.

Još jedna zanimljiva mogućnost koju su Bracwell, Gourtovaia, Wallace i Clarkson predstavili i opisali u svom radu [30], je povezivanje elemenata IBIS dijagrama sa dijelovima ostalih dokumenata koji se koriste u sustavima upravljanja znanjem. Prema [30],za ostvarivanje veze koristile bi se dvosmjerne veze, što znači da bi označeni element pokazivao na određeno mjesto u vanjskom dokumentu, dok bi to isto mjesto u vanjskom dokumentu pokazivalo na specifično mjesto u dotičnom IBIS dokumentu. Na taj način bila bi ostvarena dvostruka, tj. obostrana veza iz svakog dokumenta, što bi olakšalo snalaženje u zbrci koja često nastaje gomilanjem dokumenata.

11 ZAKLJUČAK

Cilj ovog rada bio je izraditi programski sustav koji će omogućiti prikaz i pohranjivanje tijeka promišljanja i odlučivanja pri timskom razvoju proizvoda. Kao model načina prikaza korišten je IBIS sustav logičke podrške konstruiranju, a rezultat rada je programski sustav koji omogućuje izradu IBIS dijagrama, njihovo pohranjivanje u obliku datoteke, te kasnije ponovno otvaranje i korištenje.

Sustavi logičke podrške konstruiranju, točnije, dijagrami temeljeni na IBIS sustavu svakako su u stanju smanjiti, ili čak eliminirati potrebu za pisanjem izvještaja procesa konstruiranja, što je već samo po sebi velika ušteda vremena. Pisani izvještaji mogu biti mnogo kompliciraniji od dijagrama u kojem su elementi zaključivanja povezivani istim slijedom kojim su i donošeni. Kao što je vidljivo iz ranije spomenutog *DRed* alata, ovakva vrsta dokumenata ima vrlo velike mogućnosti za povezivanje raznih elemenata sustava upravljanja znanjem, te postoji veliki potencijal za stvaranje tzv. *prostora informacija*, koji bi povezivao ukupno konstrukcijsko znanje pojedine tvrtke, ili institucije. Naravno, pod ovim se podrazumijevaju činjenice i zapisane informacije, dok inovativnost i intuicija za rješavanje određenih problema ostaje stvar pojedinih konstruktora.

U radu je prikazana mogućnost korištenja kreiranih datoteka u okviru PDM sustava, što se zapravo nameće kao prirodni okoliš za ovakve dokumente. Iako trenutno PDM sustavi ne manipuliraju vrstama datoteka kakva je predložena u radu, njihovo bi korištenje pridonijelo skraćivanju vremena konstruiranja, a time i razvoja proizvoda. No, činjenica je da se sustavi logičke podrške u procesima konstruiranja još uvijek vrlo malo koriste, te se sporo napreduje u njihovoj implementaciji. Razlog tome može biti vrlo široko područje koje se pokušava pokriti, ili jednostavno komplicirana implementacija, što je često slučaj prilikom implementacije PDM i PLM sustava. Najveći problem prilikom implementacije takvih i sličnih sustava je kako naučiti ljude da koriste sve funkcije sustava, koji je u suprotnom gotovo uvijek preskup, te opravdava svoju cijenu samo ako ga se koristi u potpunosti. Na iste probleme mogla bi naići implementacija i integracija sustava logičke podrške.

Izrađeni programski sustav moguće je koristiti za jednostavnije stvarne primjere (poput studentskih projekata), no kako ne postoji izgrađena baza oko sustava, njegovo samostalno korištenje ne bi imalo veliki utjecaj na konstrukcijske timove u proizvodnim tvrtkama. Naravno, kako je sustav samo osnova za eventualno neke kasnije istraživačke radove, postoje mnoga moguća poboljšanja, počevši od samog grafičkog prikaza elemenata, koji nikako ne moraju izgledati kako je prikazano u ovom radu. Bitno je samo da elementi imaju svoja vizualna, lagano uočljiva obilježja kako bi omogućili jednostavno snalaženje u potencijalno vrlo velikim dijagramima.

Iako se možda doima jednostavno, kreiranje i pohranjivanje dinamičkih elemenata, i kreiranje grafova povezanih elemenata to svakako nije. Zbog toga je za eventualni daljnji razvoj potrebno dobro poznavanje programskih jezika i teorije vezane uz konstruiranje i upravljanje znanjem.

12 LITERATURA

[1] Štorga M.: Predavanja iz kolegija „Razvoj proizvoda“, FSB, Zagreb, 2004.

[2] Štorga M.: „Sustav za razmjenu i upravljanje informacijama o proizvodu“, magistarski ad, FSB, Zagreb, 2002.

[3] http://en.wikipedia.org/wiki/New_product_development (16.03.2009.)

[4] Pavković N.: „Objektno orijentirani pristup modeliranju procesa konstruiranja“, doktorska disertacija, FSB, Zagreb, 2000.

[5] Herold Z.: „Strukturiranje baze znanja u procesu konstruiranja“, doktorska disertacija, FSB, Zagreb, 1997.

[6] Pavković N.: Predavanja iz kolegija „Upravljanje konstrukcijskim uredom“, FSB, Zagreb, 2006.

[7] http://en.wikipedia.org/wiki/Product_lifecycle_management (16.03.2009.)

[8] Štorga M.: Predavanja iz kolegija „Informacijski modeli proizvoda“, FSB, 2007.

[9] http://en.wikipedia.org/wiki/Product_Data_Management (16.03.2009.)

[10] Galeta T.:“Dijeljenje podataka o proizvodu kroz računalne sustave za upravljanje resursima poduzeća“, Strojarski fakultet, Slavonski Brod, doktorska disertacija, 2005.

[11] Kurland R.: „ENOVIA SmarTeam Engineering Express: A review by TechniCom“, Rockway, NJ, SAD,2008. (www.technicom.com)

[12] http://www.coe.org/Collaboration/DiscussionForum/ActiveDiscussions/tabid/210/forumid/8/postid/109748/view/topic/Default.aspx (16.03.2009.)

[13] „SmarTeam Deployment Guide“, dokumentacija sustava SmarTeam

[14] „SmarTeam Editor Administration Guide“, dokumentacija sustava SmarTeam

[15] http://www.edc.ncl.ac.uk/highlight/rhmay2007.php (16.03.2009.)

[16] http://en.wikipedia.org/wiki/Design_Rationale (16.03.2009.)

[17] Simon Buckingham Shum: „Design Argumentation as Design Rationale“, Knowledge Media Institute, The Open University Milton Keynes, MK7 6AA, U.K (http://kmi.open.ac.uk/people/sbs/research/DR.html) (16.03.2009.)

[18] Xiaochun Hu, Jun Pang, Yan Pang, Michael Atwood, Wei Sun, William C. Regli: „A survey on design rationale: representation, capture and retreival“, Geometric and Intelligent Computing Laboratory, Department of Mathematics and Computer Science, Drexel University, Philadelphia, PA 19104 (gicl.cs.drexel.edu/wiki-data/images/c/c6/Asme_detc00_dfm_14008.pdf) (16.03.2009.)

[19] S. C-Y. Lu, F. Udwadia, B. Burkett, i J. Cai: „Design Rationale for Collaborative Engineering Design in the Socio-Technical Framework“, Engineering School, University of Southern California, Los Angeles, SAD (http://impact.usc.edu/cerl/Task2/task2-2-3/DesignRationale.pdf) (16.03.2009.)

[20] Werner Kunz i Horst W. J. Rittel: „Issues as Elements of Information Systems“, 1970.

[21] Simon Buckingham Shum i Nick Hammond: „Argumentation-Based Design Rationale: What Use at What Cost?“, Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, U.K (http://kmi.open.ac.uk/people/sbs/research/DR.html) (16.03.2009.)

[22] Simon J. Shum, 1991.: „A Cognitive Analysis of Design Rationale Representation“, Chapter 2:“Approaches to Representing Design Rationale“, Department of Psychology, University of York (http://kmi.open.ac.uk/people/sbs/research/phd/phd.html) (16.03.2009.)

[23] Lemai Nguyen i Paul A. Swatman: „Complementary Use of *ad hoc* and *post hoc* Design Rationale for Creating and Organising Process Knowledge“, School of Management Information Systems, Deakin University, Victoria, Australia (www.deakin.edu.au/buslaw/infosys/docs/workingpapers/archive/working-papers-99/99-20-nguyen.pdf) (16.03.2009.)

[24] Vadla I.: „Grafičko sučelje programskog sustava za prikaz i pohranu tijeka rezoniranja i odlučivanja“, završni rad, FSB, 2008.

[25] http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29 (16.03.2009.)

[26] MSDN, Visual Studio 2005 dokumentacija

[27] http://www.switchonthecode.com/tutorials/csharp-tutorial-serialize-objects-to-a-file (16.03.2009.)

[28] http://www.c-sharpcorner.com/UploadFile/yougerthen/102162008172741PM/1.aspx (16.03.2009.)

[29] http://www.java2s.com/Code/CSharp/CatalogCSharp.htm (16.03.2009.)

[30] Bracewell R., Gourtovaia M., Wallace K. i Clarkson J.:“Extending design rationale to capture an integrated design information space“, University of Cambridge, UK, 2007.