

TEHNIČKO VELEUČILIŠTE U ZAGREBU ZAGREB UNIVERSITY OF APPLIED SCIENCES

PROGRAMSKO INŽENJERSTVO I INFORMACIJSKI SUSTAVI

dr. sc. Željko Kovačević
Edmond Krusha, struč.spec.ing.techn.inf.

Zagreb, 2022

PRIRUČNICI TEHNIČKOG VELEUČILIŠTA U ZAGREBU
MANUALIA POLYTECHNICI STUDIORUM ZAGRABIENSIS

Željko Kovačević
Edmond Krusha

PROGRAMSKO INŽENJERSTVO I INFORMACIJSKI SUSTAVI

Priručnik na kolegijima 'Programsko inženjerstvo u otvorenim sustavima', 'Projektiranje i oblikovanje informacijskih sustava' te 'Modeliranje poslovnih procesa' koji se održavaju u sklopu nastave na diplomskom stručnom studiju informatike Tehničkog veleučilišta u Zagrebu

Zagreb, 2022.



TEHNIČKO VELEUČILIŠTE U ZAGREBU ZAGREB UNIVERSITY OF APPLIED SCIENCES

izdavač	Tehničko veleučilište u Zagrebu Vrbik 8, Zagreb
za izdavača	mr. sc. Goran Malčić
autori	dr. sc. Željko Kovačević Edmond Krusha, struč.spec.ing.techn.inf.
recenzenti	dr. sc. Ksenija Klasić, prof. v. š. dr. sc. Milorad Nikitović, prof. v .š.
vrsta djela	Priručnik Objavljivanje je odobrilo Stručno vijeće Tehničkog veleučilišta u Zagrebu odlukom broj: 3595-2 red. 11/20 od 17. studenog 2020
ISBN	978-953-7048-93-8

SADRŽAJ

1. Metode razvoja softvera 11
1.1. Metoda vodopada 11
1.2. Agilne metode 14
1.2.1. Scrum 15
1.2.2. Kanban 25
1.2.3. Scrumban 34
1.2.4. Scrum of Scrums 34
2. Testiranje softvera 37
2.1. Načela i metode testiranja 37
2.1.1. Ručno i automatizirano testiranje 39
2.1.2. Testiranje crne kutije 41
2.1.3. Testiranje bijele kutije 43
2.1.4. Testiranje crne ili bijele kutije – što odabrati? 46
2.1.5. Testiranje sive kutije 46
2.2. Standardi 48
3. Dokumentacija 53
3.1. Tipovi dokumentacije 53
3.1.1. Sistemska dokumentacija 53
3.1.2. Korisnička dokumentacija 55
3.2. Generatori dokumentacije 56
4. Uvod u alate Enterprise Architecture 61
5. Rad s dijagramima – Dijagram aktivnosti 63
5.1. Kako do dijagrama aktivnosti? 64
5.2. Temeljni elementi 64
5.3. Primjer dijagrama aktivnosti 67
5.4. Rad s dijagramima – Dijagram klasa 68
5.5. Kako do dijagrama klasa? 69
5.6. Korištenje dijagrama klasa 69
5.7. Primjer dijagrama klasa 69
6. Rad s dijagramima – Organizacijska shema 71
6.1. Izrada organizacijskog grafikona 71
6.2. Organizacijski dijagram – hijerarhijska struktura 73

6.2.1. Kako do dijagrama? 74
7. Rad s dijagramima – Dijagram slučajeva korištenja 75
7.1. Primjer dijagrama 78
8. Rad s dijagramima – Dijagram toka podataka 79
8.1. Primjer dijagrama toka podataka 80
9. Rad s dijagramima – Dijagram slijeda 83
9.1. Primjer dijagrama slijeda 86
10. Generiranje dokumenata i koda 87
10.1. Generiranje dokumentacije 87
10.2. Generiranje izvornog koda 89
11. Prvi koraci 95
11.1. Važno je znati 95
11.2. Što se promijenilo? 96
12. Pokretanje Enterprise Architecta 99
12.1. Korištenje probnog izdanja 99
13. Izrada modela 101
13.1. Odabir perspektive 101
13.2. Korištenje uzorka 102
13.3. Project Browser 103
14. Rad s dijagramima 107
15. Izgradnja modela – osnovne vještine 113
15.1. Korištenje perspektiva i uzoraka modela 121
16. Prilagodba radnog prostora 129
16.1. Promjena vizualnog stila 129
16.2. Postavljanje određenih glavnih postavki 131
16.3. Rad s vrpcom 132
16.3.1. Razmještaj s prozorima 134
16.3.2. Vrpce ukratko 135
16.3.3. Prozor za portale 135
16.3.4. Perspektive (Perspective) 136
16.3.5. Kolaboracija 137
16.4. Tipke prečaca 138
17. Dostupni resursi 141
17.1. Help – pomoć 141

17.2. EA Primjeri 142
18. Pogovor 147
19. Popis slika 149
20. Popis primjera i tablica 153
21. Reference 155

Uvodna riječ

Ovaj priručnik pisan je za potrebe studenata diplomskog stručnog studija informatike na Tehničkom veleučilištu u Zagrebu. Sadržaj priručnika obuhvaća sadržaje kolegija 'Programsko inženjerstvo u otvorenim sustavima', 'Projektiranje i oblikovanje informacijskih sustava' te 'Modeliranje poslovnih procesa', a namijenjen je i svima, koji iz navedenih područja žele steći dodatna znanja.

Sadržaj priručnika obuhvaća područja programskog inženjerstva koja se bave metodama razvoja, testiranja i dokumentiranja softvera. Također, obuhvaća i područja iz projektiranja i oblikovanja informacijskih sustava te modeliranja poslovnih procesa, s naglaskom na korištenje alata Enterprise Architect. Priručnik je pisan s namjerom da studentima u što većoj mjeri, vizualno i kroz praktične primjere, demonstrira danas aktualne metode razvoja i upravljanja softverskim projektima, proces njihovog testiranja i dokumentiranja. Također, detaljnije je opisan i rad u alatu Enterprise Architect koji se višenamjenski može koristiti u gotovo svim fazama razvoja softvera, bilo da je riječ o vizualizaciji procesa i programskog koda, generiranju dokumentacije itd.

Ovim putem autori se također žele zahvaliti svima koji su doprinijeli kvaliteti sadržaja ovog priručnika, svojim prijedlozima ili primjedbama, a to su mnogobrojni kolege iz struke, recenzenti te studenti Tehničkog veleučilišta u Zagrebu.

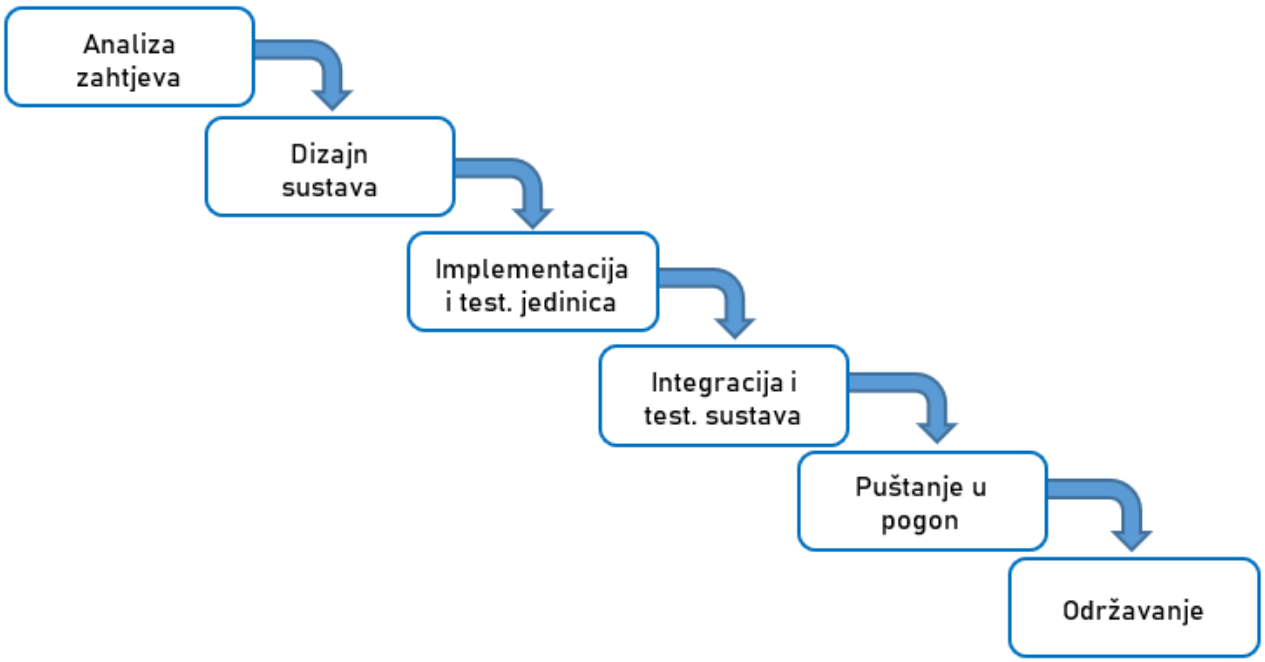
Autori
Tehničko veleučilište u Zagrebu, 2022.g.

1. Metode razvoja softvera

Da bi softver najbolje funkcionirao, potrebno je odabrati prikladnu metodu po kojoj će se proizvesti. Odabir metode ovisi o vrsti proizvoda, broju osoba koje će se njime služiti, načinu na koji te osobe najbolje funkcioniraju, kako bi se postigla što veća učinkovitost i sličnim čimbenicima koji utječu na njegov razvoj. Metode razvoja softvera mogu biti metoda vodopada, agilne metode, rapidne aplikacije (RAD), spiralne, ekstremno programiranje, DevOps i sl. U ovom priručniku prikazuju se metoda vodopada i agilne metode.

1.1. Metoda vodopada

Razvojnu metodu zvanu Metoda vodopada (engl. *waterfall*) prvi je objasnio Winston W. Royce 1970. godine [1] i to je ujedno i najstarija metoda. Za ovaj model karakteristična je njegova sekvencijalnost i linearnost. Tek kada je jedna faza u potpunosti završena, prelazi se na iduću fazu i tako do kraja ciklusa razvoja softvera. Kao što prikazuje Slika 1., grafički prikaz modela nalikuje vodopadu, a rad na izradi softvera prikazan je kao tok. Zbog tih karakteristika, ovaj model naziva se vodopadom [2].



Slika 1. Prikaz hijerarhijske strukture vodopada [2]

Model se sastoji od 6 faza.

1. **Analiza zahtjeva**

   U ovoj fazi sastaju se naručitelj softvera i razvojni tim, kako bi se što preciznije i detaljnije definiralo korisnikove zahtjeve, potrebe, funkcionalnosti i probleme koje će se novim proizvodom riješiti. Zahtjevi se prikupljaju i specificiraju. Na temelju ove faze nastaje dokument koji služi za razvoj softvera, ali i kao svojevrstan ugovor za rješavanje eventualnih spornih situacija.

2. **Dizajn sustava**

   Ovo je faza planiranja proizvoda. Pomaže specificirati potrebne hardverske komponente. Zahtjevi definirani u prethodnoj fazi pretvaraju se u strukturu koja je pogodna za implementaciju u nekom od programskih jezika. Rezultat ove faze je arhitektura sustava.

3. **Implementacija i testiranje jedinica**

   Implementacija se sastoji od izrade programskog koda, odnosno pretvaranja dizajna u softver. Sustav se implementira po manjim cjelinama – jedinicama. Svaka od njih integrirana je u iduću fazu. Izrađuju se testne studije koje predviđaju kako će razvoj proizvoda teći, nedostaci i greške koje se mogu ukloniti u ranijoj fazi razvoja, i slične predvidive poteškoće koje je važno ukloniti što ranije, kako se na završetku razvoja proizvoda ne bi moralo vraćati na prethodne faze. Krajnji rezultat ove faze je skup modula koji su testirani kao jedinice.

4. **Integracija i testiranje sustava**

   Sve jedinice, odnosno moduli, koje su izrađene u prethodnoj fazi, integriraju se u jedan sustav. Integracija je inkrementalna, odnosno provodi se u nekoliko koraka. Nakon integracije, pristupa se testiranju cjelokupnog sustava. Testiranjem se ispituje zadovoljava li sustav zahtjevima i potrebama naručitelja. Testiranje može biti funkcionalno i nefunkcionalno, a izvode ga naručitelj softvera i proizvođač, odnosno razvojni tim. Ako proizvod nije zadovoljio testiranje, vraća se u prethodnu fazu.

5. **Puštanje u pogon**

   Ova faza provodi se nakon što su rezultati svih testiranja pozitivni. Može se provesti u jedinicama, a u konačnici program se isporučuje naručitelju ili se lansira na tržište.

6. **Održavanje**

   Dobivaju se povratne informacije od naručitelja i izvršava se održavanje softvera. Vrste održavanja su: [3]

   1. Usavršavanje je najčešći tip održavanja, a sastoji se od nadogradnje funkcionalnosti, izdavanja novih inačica proizvoda.
   2. Korektivno održavanje obuhvaća ispravljanje grešaka i nedostataka koje nisu otkrivene za vrijeme razvojne faze.
   3. Adaptivno održavanje odnosi se na prilagodbu promjenama uvjeta, poput prijenosa softvera u novo okruženju (primjerice, na drugi uređaj, OS), prilagodbe novim poslovnim ili zakonodavnim pravilima (nove odredbe o zaštiti privatnosti) i sl.
   4. Preventivno održavanje najmanje je zastupljeno, a odnosi se na sprječavanje pojavljivanja predvidivih grešaka.

   Rezultat provedbe ove faze izrade softvera su izmjene.

Metoda vodopada je jednostavna i lagana za razumijevanje i upotrebu. Lagan je za primjenu zbog njegove sekvencijalnosti – tek kada je jedna faza završena, prelazi se na sljedeću i ne dolazi do preklapanja. Zahtjevi su vrlo jasni pa je lako odrediti zadatke i faze, ulazne i izlazne kriterije. Dobro je dokumentiran te pogodan za manje programe, jer je u njima lako pratiti faze. Također je dobar i za programe s jasno definiranim zahtjevima.

Međutim, metoda vodopada ima i svoje nedostatke. Faze se izvršavaju jedna za drugom, a model ne dopušta istovremeno provođenje faza, što je najčešće slučaj u stvarnosti, kako bi se smanjili troškovi i vrijeme razvoja. U slučaju bilo kakvih problema, moramo se vratiti nazad u jednu od iteracija (mnogo mogućih ponavljanja iteracija – dugo vrijeme do izdavanja softvera).

Softver je gotov tek u kasnoj fazi. Nema izrade prototipa niti ispravaka grešaka tijekom ranijih faza razvoja, a to uzrokuje visoki rizik i neizvjesnost, pa su problemi i rizici s ovim pristupom vidljivi tek u kasnoj fazi izrade programa. Testiranje se provodi tek u završnoj fazi razvoja, što je neekonomično financijski i vremenski. Metoda vodopada nije pogodna

za kompleksne i dugotrajne projekte jer nije fleksibilan i ne omogućuje naknadne promjene u korisničkim zahtjevima.

1.2. Agilne metode

Agilne metode razvoja softvera predstavljaju pristup koji u prvi plan stavlja korisnika. Iterativne su, fleksibilne i iziskuju manji budžet. Provode se česta testiranja i prilagodbe te se potiče timski rad. Korisnik je uključen u razvoj proizvoda tijekom cijelog razvojnog ciklusa svojim prijedlozima za izmjene dijelova kojima nije u potpunosti zadovoljan i testiranjem. [4] Zbog toga se greške rano otkriju i brže isprave. Korisnik je zadovoljan gotovim proizvodom koji je završen ranije i financijski je povoljniji od izrade putem klasičnih metoda (poput vodopada).

Agilni razvoj odnosi se na sve postupke koji su u skladu s Agilnim manifestom [5]. Manifest je donijela (Utah, 2001.) ekipa od sedamnaest programera kao odgovor na nezadovoljstvo sporom procedurom razvoja softvera, koja je dovela do mnogih otkazanih projekata. Sastoji se od četiri temeljne vrijednosti te dvanaest principa [6].

Temeljne vrijednosti podrazumijevaju davanje prednosti:

1. pojedincima i interakciji u odnosu na procese i alate,
2. funkcionalnom softveru u odnosu na opsežnu dokumentaciju,
3. suradnji s korisnikom, i
4. odgovoru na promjene u odnosu na unaprijed zacrtani plan.

Principi Agilnog manifesta [5]:

1. Najvažniji prioritet je ispunjavanje zahtjeva korisnika tako da se kvalitetan softver počne isporučivati rano i kontinuirano,
2. Treba prigrliti zahtjeve za izmjenama, čak i u kasnoj fazi razvoja,
3. Učestalu isporuku radne verzije softvera, u razmacima od dva tjedna do dva mjeseca, po mogućnosti sa što manjim vremenskim razmakom,
4. Članovi timova moraju surađivati svakodnevno, tijekom izvedbe projekta,
5. Projekte treba graditi oko motiviranih pojedinaca. Osigurati im potrebnu okolinu, podršku i povjerenje da će dobro odraditi posao,

6. Najučinkovitiji način prijenosa informacija razvojnom timu i unutar njega je razgovor licem u lice,
7. Funkcionalan softver primarno je mjerilo napretka,
8. Agilni procesi promoviraju održivi razvoj. Sponzori, programeri i korisnici trebaju biti u mogućnosti stalno držati korak.,
9. Neprestano posvećivanje pažnje tehničkoj izvrsnosti i dobrom dizajnu podižu razinu agilnost,
10. Neophodna je jednostavnost – umjetnost povećavanja količine posla kojeg ne treba obaviti,
11. Najbolje arhitekture, zahtjevi i dizajn proizlaze iz timova koji su sami zaduženi za organiziranje svog posla, te
12. Tim sam propitkuje na koje načine može biti učinkovitiji i usklađuje svoje djelovanje

Najpoznatiji principi razvoja projekata, odnosno softvera, nastalih u okviru agilnih metodologija su Scrum i Kanban.

1.2.1. Scrum

Naziv „Scrum“ nije kratica, iako se često piše svim velikim slovima. On se prvi puta pojavljuje 1986. u članku *The New New Product Development Game* (autori: Hirotaka Takeuchi i Ikujiro Nonaka). Scrum (skraćeno od *scrummage*) je termin koji se koristi u ragbiju, a upotrijebljen je u članku kako bi se naglasila važnost timskog rada u razvoju složenih proizvoda. [7]

Scrum je razvojni okvir koji omogućava rješavanje složenih, dinamičnih problema te istovremeno isporučivanje najkvalitetnijih proizvoda. [8] Scrum nije postupak, tehnika ili metoda. On je razvojni okvir koji uključuje razne postupke i tehnike. Omogućava jasan uvid u razvoj projekta, kako bi se kontinuirano moglo raditi na njegovom unapređenju, kao i unapređenju tima i radne okoline.

Scrum se pokazao posebno učinkovitim kod iterativnog i inkrementalnog prijenosa znanja. Naširoko se primjenjuje prilikom izrade proizvoda i usluga, kao i kod upravljanja organizacijom koja ih izrađuje. Scrum koristi preklapajući model gdje se sve radi pomalo, kroz cijeli tok razvoja projekta [9]. Bit Scruma je u malom timu ljudi. On je samostalan,

fleksibilan i prilagodljiv. Sastoji se od timova i njima pripadajućih uloga, događaja, artefakata i pravila. Jednostavan za razumjeti, ali težak za savladati.

1.2.1.1. Uloge u scrum timu

Izraz „Scrum Team“ odnosi se na Scrum Mastera, vlasnika proizvoda i razvojni tim. Svi članovi tima imaju svoju definiranu ulogu u vođenju projekta. [10] *Scrum Master* [11] je voditelj tima i najbolje poznaje Scrum. Odgovoran je za besprijekorno odvijanje procesa. On:

1. podučava vlasnika proizvoda kako maksimalno unovčiti svoje ulaganje i postići svoje ciljeve kroz Scrum,
2. unapređuje razvojni tim, potičući kreativnost i osnaživanje,
3. unapređuje produktivnost razvojnog tima,
4. unapređuje inženjersku praksu i pomagala kako bi svako poboljšanje u funkcionalnosti bilo moguće preslikati i na druge segmente, te
5. ažurira podatke o napretku tima kako bi ono bilo vidljivo svima uključenima u proces.

Scrum Master je stalno u toku sa stanjem projekta, fleksibilan je i sposoban riješiti eventualne teškoće, te se brine da ništa ne ometa razvojni tim u njegovom radu. On potiče razvojni tim na samostalnost i donošenje odluka, osnažuje ga i minimalno nadzire.

Vlasnik proizvoda tijesno surađuje s razvojnim timom. On određuje i postavlja zahtjeve, definira ključne značajke proizvoda te određuje redoslijed implementacije segmenata proizvoda. Odgovoran je za viziju proizvoda te odlučuje kada nastaviti s njegovim razvojem, a kada ga pustiti na tržište. Vlasnik održava popis zadataka koje treba ispuniti, određuje raspored puštanja u pogon gotovog proizvoda i donosi završnu odluku o tome zadovoljava li proizvod inicijalno zacrtane zahtjeve i potrebnu kvalitetu prije puštanja u pogon, odnosno prodaju.

Osim Scrum Mastera i vlasnika, za izradu proizvoda potreban je razvojni tim. On samostalno donosi odluke kako raščlaniti izradu proizvoda u manje zadatke, za koje tada treba zadužiti pojedine članove razvojnog tima. Idealna veličina tima je tri do devet članova, najčešće sedam, ukoliko mogućnosti to dopuštaju. Više od toga otežava komunikaciju, a manji broj članova tima dovodi do loše produktivnosti.

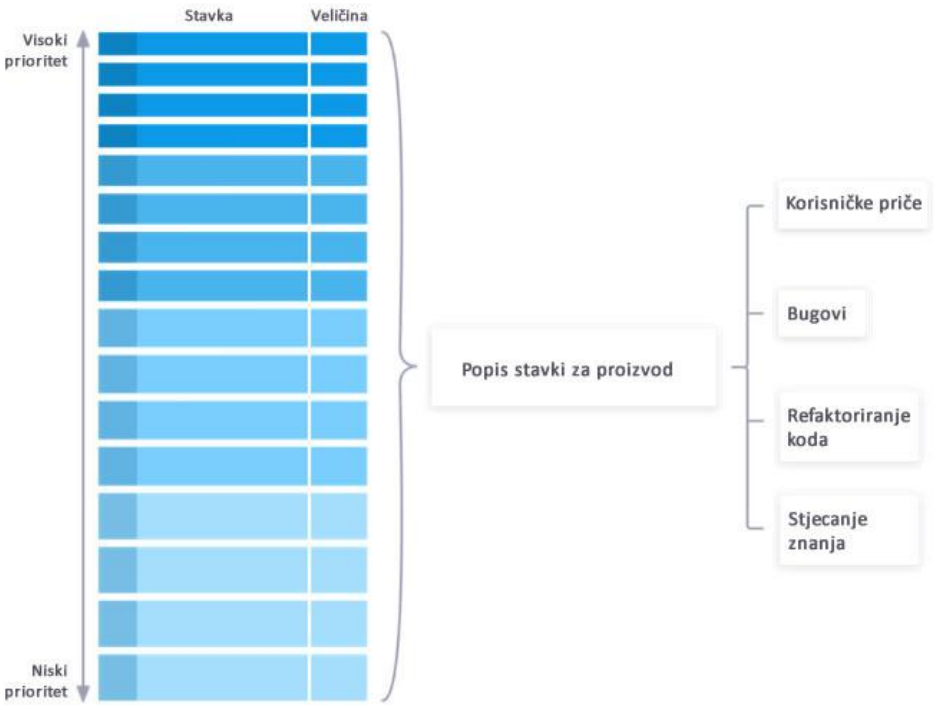
1.2.1.2. Artefakti

U arheologiji artefakti (engl. *artifacts*) predstavljaju predmete koje je izradio čovjek, dok u latinskom jeziku ta riječ označava, u grubom prijevodu, umjetna tvorevina. Stoga, artefakte možemo definirati kao ljudski proizvod, pomagalo za rješavanje problema ili umjetničko djelo [12]. Artefakti u Scrumu omogućavaju transparentnost, mogućnost uvida i prilagodbe [8]. Za artefakte, kao i za Scrum općenito, karakteristična je dinamičnost. Promjene u zahtjevima, uvjetima na tržištu ili tehnologiji i sl. uzrok su i promjena u artefaktima i samom razvoju proizvoda i projekta. Nekoliko je vrsta artefakata [13]:

Vizija proizvoda definira dugoročni cilj proizvoda ili projekta. Mora biti jasno definirana, kratka i precizna.

Cilj ciklusa bit će postignut tijekom ciklusa primjenom (implementacijom) popisa stavki za proizvod.

Popis stavki za proizvod odgovara na pitanje „Što je sljedeće najvažnije za izraditi?“ Sastoji se od dinamičnog popisa svega potrebnoga za izradu proizvoda ili projekta te za njegovu prilagodbu u kasnijim inačicama. Stavke na popisu poredane su po prioritetu. Za popis stavki za proizvod (eng. *Product Backlog*), uključujući sadržaj, dostupnost i redoslijed stavki, odgovoran je vlasnik proizvoda [8].



Slika 2. Popis stavki za proizvod [14]

Popis stavki za proizvod je prikaz posla koji razvojni tim planira izvršiti tijekom ciklusa i pripada isključivo razvojnom timu. Vlasnik proizvoda može utjecati na razvojni tim pomažući mu u razumijevanju i dogovaranju načina implementacije, ali ljudi koji izvode radove daju konačnu procjenu.

Rafiniranje stavki s popisa (eng. *Product Backlog Refinement*) provodi se neprekidno tijekom cijelog životnog ciklusa samog popisa. Stavkama se dodaju detalji, procjene i redoslijed, one se pregledavaju i mijenjaju. Vlasnik može ažurirati stavke na popisu kad god smatra da je to korisno i uskladiti ih prema vlastitim potrebama [8].

Popis stavki za ciklus (eng. *Sprint Backlog*) možemo reći da se sastoji od dviju stavki – „Što“ i „Kako“. To je u suštini plan razvojnog tima o načinu isporuke inkrementa proizvoda i ostvarivanju cilja ciklusa [12]. Sastoji se od stavki s popisa za proizvod koje su odabrane za taj ciklus, plana za isporuku inkrementa proizvoda i ostvarivanja cilja ciklusa. To je predviđanje razvojnog tima o funkcionalnostima koje će biti u sljedećem inkrementu te posla nužnog za isporuku funkcionalnosti u "gotov" inkrement.

Popis stavki za ciklus dovoljno je detaljan plan da bi se promjene o napretku razumjele tijekom dnevnog sastanka. Stavke na popisu se neprekidno ažuriraju u skladu s izvođenjem poslova. Samo razvojni tim može promijeniti popis stavki za ciklus tijekom ciklusa. Popis stavki za ciklus je svima vidljiva slika posla u realnom vremenu kojeg razvojni tim planira dovršiti tijekom ciklusa, i pripada isključivo razvojnom timu. U bilo kojoj fazi provedbe ciklusa, popis stavki za ciklus svima pruža transparentni uvid u preostali dio posla, omogućavajući tako upravljanje napretkom. Tijekom svakog dnevnog sastanka, razvojni tim prati stanje popisa [8].

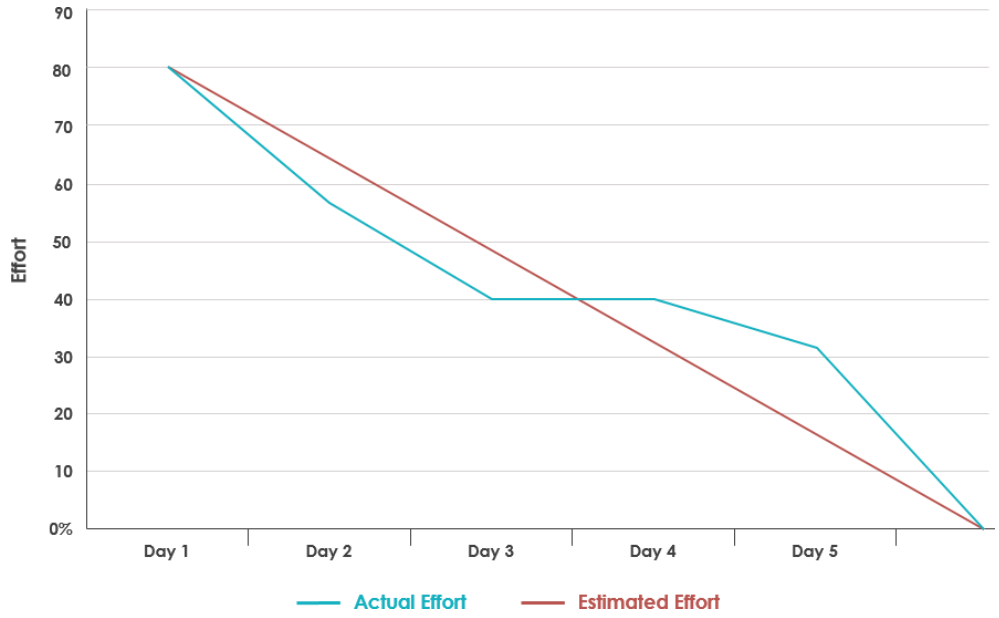
Inkrement je zbroj svih stavki s popisa za proizvod, koje su dovršene tijekom svih ciklusa te vrijednost inkremenata svih prethodnih ciklusa. Na kraju svakog ciklusa, novi inkrement mora biti „gotov“. Da bio „gotov“, mora zadovoljavati definiciju „gotovoga“ i mora biti u upotrebljivom stanju, neovisno o odluci vlasnika kada će pustiti proizvod na tržište. Scrum tim izrađuje zajedničku definiciju „gotovoga“. Ukoliko na proizvodu ili projektu radi više razvojnih timova, oni svi imaju zajedničku definiciju „gotovoga“, završenoga posla. Definicija „gotovog“ pruža transparentnost razine kvalitete koja je dovoljna da bi se Inkrement pustio u pogon [12].

Definicija „gotovog“ služi i članovima razvojnog tima kako bi znali koliko stavki s popisa za proizvod mogu odabrati tijekom planiranja ciklusa. Svrha svakog ciklusa je isporučiti funkcionalnosti koje se mogu pustiti u rad i koje su „gotove“. Svaki Inkrement je dodatak na sve ranije inkremente, detaljno je testiran, kako bi svi Inkrementi radili zajedno [8].

Padajući *ili silazni* grafikon *(u daljnjem tekstu: Burn-down)* pružaju vremenski pregled napretka tijekom provedbe projekta, odnosno smanjenja opsega posla kako odmiče vrijeme. Po završetku zadataka, *burn-down* grafikon se svede na nulu. Njegova svrha je pomoći da ciklus proizvede odgovarajući rezultat u zadanome vremenu. Može se koristiti kod bilo kakvih projekata kod kojih se prati napredak kroz vrijeme.

Burn-down grafikon je jednostavan i svakodnevno se ažurira. Stoga pruža brzi uvid u preostali dio posla i dobar je za njegovu vizualizaciju. Grafikon prikazuje planirani (crvenom bojom) i stvarni (plavom bojom) ritam izvršavanja posla (Slika 3). Smješten je na svima vidljivo mjesto. Tijekom planiranja ciklusa izrađuje se idealni *burn-down* grafikon. Tijekom ciklusa, svaki član tima preuzima zadatke s popisa stavki za ciklus i radi na njima. Na kraju

dana članovi tima ažuriraju broj sati potreban za završetak zadataka. Na taj način grafikon se ažurira na dnevnoj bazi.



Slika 3. Burn-down grafikon [15]

Ako tim zaključi da na popisu stavki za ciklus postoji prevelika količina stavki, pomoću *burn-down* grafikona određuju koje od tih stavki ne mogu izvršiti u zadanom ciklusu te ih vraćaju na popis stavki za proizvod. *Burn-down* grafikon može se pogrešno koristiti kao oblik izvještaja o napretku i biti prepreka u produktivnosti tima. U tom slučaju Scrum Master ga treba ukloniti [16].

1.2.1.3. Događaji u Scrumu

Propisani događaji se u Scrumu koriste kako bi se postigla redovitost i umanjila potreba za sastancima koji u Scrumu nisu definirani. Oni obuhvaćaju aktivnosti poput planiranja ciklusa, održavanja dnevnih sastanaka, pregleda i osvrta na ciklus itd. (Slika 4). Svaki događaj ima ograničeno maksimalno vremensko trajanje.



Slika 4. Događaji u Scrumu [11]

Svaki od događaja formalna je prilika za kontrolu i prilagodbu. Ovi događaji posebno su osmišljeni za omogućavanje kritične transparentnosti i kontrole. Ako se bilo koji od ovih događaja preskoči, rezultat je smanjenje transparentnosti i gubitak prilike za kontrolu i prilagodbu [8].

Srce Scruma je ciklus (engl. *Sprint*). Vremenski je ograničen na trajanje od 2 do 4 tjedna jer bi u predugim ciklusima moglo doći do promjene definicije onoga što je potrebno napraviti, a provedba može postati kompleksna i rizična. Ciklus traje onoliko dugo koliko se predviđa da neće doći do promjena u njemu. Najbolje je da ciklusi imaju uvijek isto trajanje tijekom razvoja projekta. Kada se planira promjena, prelazi se na novi ciklus. Svaki ciklus može se smatrati pod-projektom. On ima cilj, definiciju onog što se treba napraviti, dizajn, prilagodljiv plan vođenja posla i na kraju, proizvod koji je rezultat rada [8].

Ciklus se može otkazati prije isteka predviđenog vremena, za što je ovlašten samo vlasnik. On se otkazuje ako više nema smisla nastaviti s radom. Budući da ciklusi traju najduže mjesec dana, otkazivanje je rijetko potrebno. Ako dođe do otkazivanja ciklusa, sve dovršene i "gotove" stavke s popisa se pregledavaju. Sve nedovršene stavke se vraćaju natrag u popis stavki koje proizvod treba imati.

Na početku svakog ciklusa održava se sastanak za planiranje ciklusa (engl. *Sprint Planning*), u kojem sudjeluju svi članovi Scrum tima. Na njemu se dogovara koje će se

stavke s popisa za proizvod preuzeti za rad u ciklusu. Vlasnik objašnjava koje su stavke najvažnije za razvoj projekta, odnosno koji je cilj ciklusa. Razvojni tim odabire količinu zadataka koju smatra realnom za ciklus te preuzima zadatke s popisa za proizvod, na popis za ciklus koji se razrađuje na manje cjeline. Tim utvrđuje koji su zadaci prioritetni i kako će ih izvršiti. Dužina sastanka ovisi o dužini ciklusa. On smije trajati najviše osam sati i to za ciklus od 30 dana. Za kraće cikluse i trajanje sastanka je kraće [16].

Planiranje ciklusa ne traje duže od osam sati za jednomjesečni ciklus. Scrum Master osigurava da se planiranje ciklusa dogodi i da sudionici razumiju njegovu svrhu. Također, on podučava članove tima kako da održe planiranje u dogovorenim vremenskim okvirima. Do kraja planiranja ciklusa razvojni tim treba biti u mogućnosti objasniti vlasniku proizvoda i Scrum Masteru kako namjeravaju raditi kao samoorganizirajući tim, da bi postigli cilj ciklusa i kreirali očekivani inkrement. Rezultati planiranja su cilj ciklusa i popis za ciklus.

Dnevni sastanak (eng. *Daily Scrum*) održava se svaki dan u isto vrijeme i na istom mjestu. Sastanci traju 15-ak minuta, a tog vremenskog ograničenja se lakše pridržavati ako svi prisutni stoje tijekom sastanka. Održavaju se ispred Scrum ploče (Slika 5, Slika 6). Dnevni sastanak je interni sastanak članova razvojnog tima. Ako su prisutne i druge osobe, Scrum Master osigurava da ga one ne ometaju.

Članovi tima raspravljaju o učinjenome prethodnog dana, planovima za tekući i o teškoćama s kojima se susreću. Od pomoći mogu biti popis zadataka u ciklusu i *burn-down* grafikon ciklusa. Odgovara se na tri pitanja: 1. Što ste radili jučer?; 2. Što ćete raditi danas?; 3. Postoje li kakve prepreke? Teme koje iziskuju dodatnu pozornost mogu se raspravljati nakon formalnog dijela sastanka [16]. Cijeli razvojni tim ili samo neki članovi često se sastaju odmah nakon dnevnog sastanka, kako bi detaljnije prilagodili plan za ostatak posla u ciklusu. Dnevni sastanci nisu namijenjeni rješavanju problema [9].

Dnevni sastanak poboljšava komunikaciju i interakciju, čini druge sastanke nepotrebnima, te unaprijed prepoznaje situacije koje bi inače prerasle u probleme. Odluke se donose brzo i poboljšava se razina znanja kod članova razvojnog tima. Ovo je ključni sastanak za kontrolu i prilagodbu [8]. Ovi sastanci sušta su suprotnost staroj praksi individualnog rada. Scrum se temelji na samostalnosti svakog člana tima, koji pridonose timskom radu [11].

Pregled ciklusa (eng. *Sprint Review*) se održava na kraju svakog ciklusa. Cilj mu je kontrola inkrementa i, ako je potrebno, prilagodba popisa stavki za proizvod. Na sastanku sudjeluje Scrum tim i ostale osobe koje pozove vlasnik, primjerice dionike i krajnje korisnike. Predstavljaju se stavke s popisa stavki za proizvod koje jesu gotove i koje nisu. Raspravlja se o tome što se izvršilo bez ikakvih teškoća, koji su se problemi pojavili te jesu li riješeni i kako. Stavke koje nisu dovršene, vraćaju se na popis stavki za proizvod i vlasnik ih kategorizira prema prioritetu za buduće cikluse. Raspravlja se i o tome što će se sljedeće raditi tako da je ovaj sastanak i svojevrsna priprema za planiranje ciklusa. Pregledava se vremenski plan, budžet i prolaze se ostala pitanja koja se pojave tijekom rada na inkrementu [11] [16].

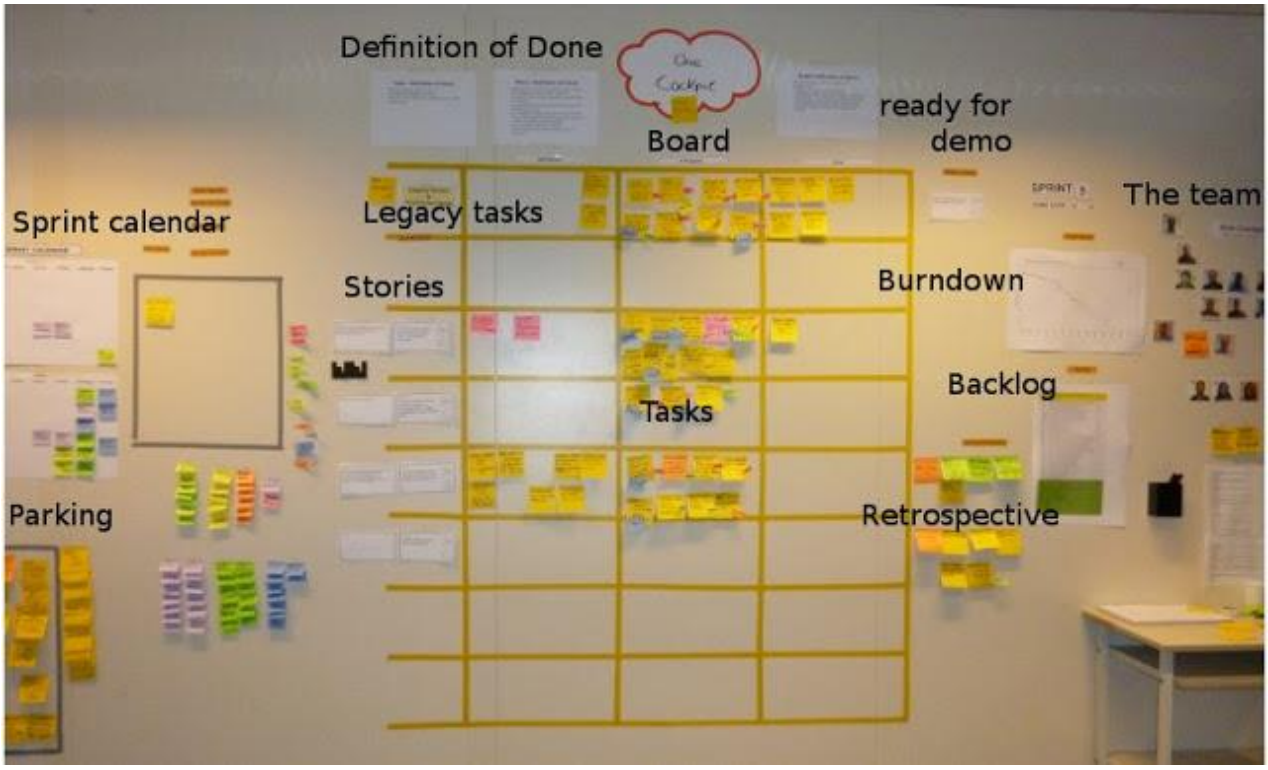
Rezultat pregleda ciklusa je prilagođeni popis stavki za proizvod, koji istovremeno definira moguće stavke za sljedeći ciklus. Popis stavki se može i općenito prilagoditi tako da oslikava novonastale prilike [8]. Sastanak bi trebao biti neformalan i demonstracija postignutoga, a ne šturo izvještavanje. Pripreme za sastanak trebale bi trajati oko dva sata. Pregled ciklusa traje ne duže od četiri sata, za jednomjesečne cikluse. Za kraće cikluse, ovaj sastanak je obično kraći.

Osvrt na ciklus (engl. *Sprint Retrospective*) odvija se na kraju svakog ciklusa, na sastanku u trajanju do 3 sata za mjesečne cikluse. Odvija se nakon pregleda ciklusa, a prije planiranja sljedećeg ciklusa. U njega je uključen cijeli Scrum tim, prema potrebi i ostali. Raspravlja se o tome na čemu žele: 1. prestati raditi i 2. nastaviti raditi. Tim retrospektivno sagledava svoj rad i postignutu kvalitetu proizvoda kako bi ih unaprijedio u budućim ciklusima i prilagodio definiciju „gotovog“.

Za ovakvu vrstu sastanaka potrebno je stvoriti atmosferu psihološke sigurnosti kakva se ne nalazi u većini organizacija. Bez osjećaja sigurnosti na sastancima na kojima se retrospektivno analizira rad, mogla bi se izbjegavati neugodna pitanja ili skrenuti u smjer okrivljavanja i netrpeljivosti. Osjećaju sigurnosti nikako ne doprinosi prisustvo osoba koje donose evaluaciju izvedbe, niti ljudska sklonost donošenju preuranjenih zaključaka i prijedloga. Još jedna prepreka osjećaju sigurnosti je zemljopisna raspršenost članova tima iz razloga što se ne poznaju dovoljno. Sve su to prepreke koje se mogu ukloniti [11]. Rezultati osvrta na ciklus su prepoznata poboljšanja koja će se uvesti u rad tijekom idućeg ciklusa [8].

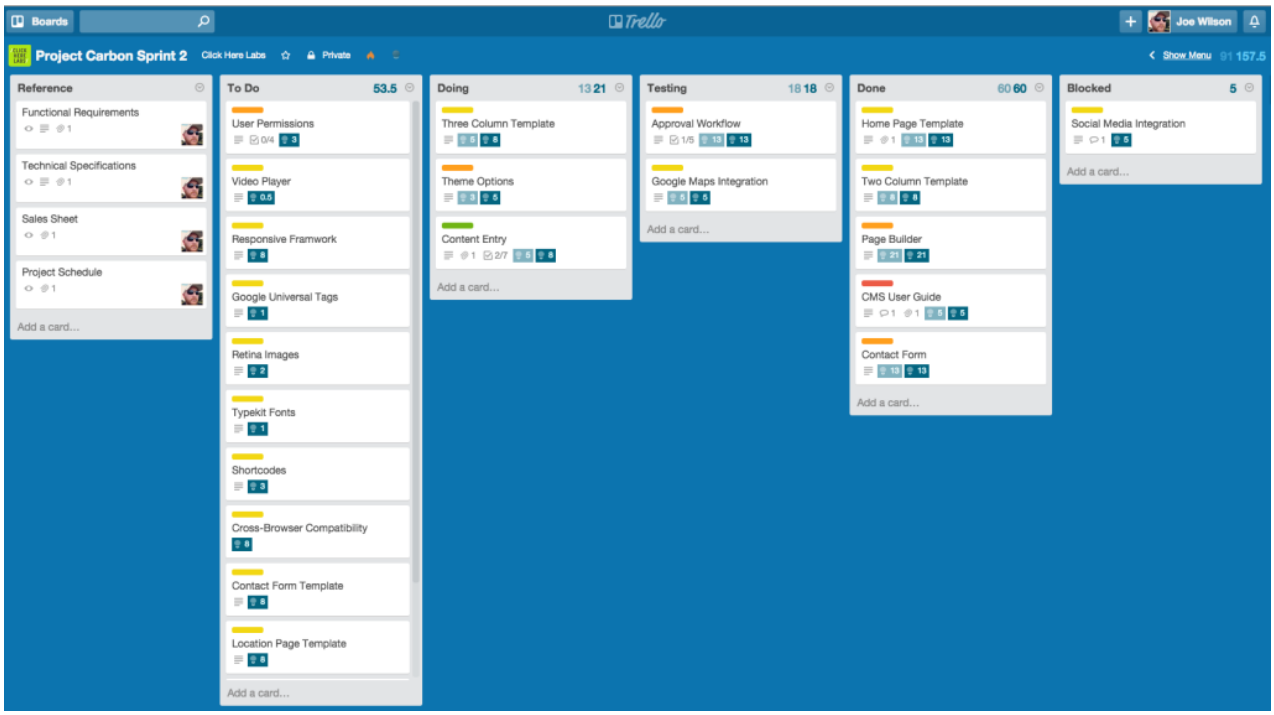
Pročišćavanje (Rafiniranje) stavki s popisa za proizvod (engl. *Product Backlog Refinement*) neprekidno se provodi, kako bi se unaprijedio sam popis stavki za proizvod. Popis stavki često je predugačak ili nejasan, pa ga je stoga dobro revidirati prije početka svakog ciklusa. Velike stavke razrađene su u manje i pojašnjene su. Tijekom rafiniranja može se odrediti i vrijeme potrebno za dovršetak svake od stavki i tehničke informacije, koje će vlasniku pomoći pri određivanju stupnja važnosti [11].

Stavkama se dodaju detalji, procjene i redoslijed, nakon čega se stavke ponovo pregledavaju i eventualno mijenjaju. Međutim, vlasnik može ažurirati stavke na popisu kad god smatra da je to korisno i prema vlastitim potrebama [8]. Rafiniranje stavki s popisa neprekidno se provodi, tijekom cijelog životnog ciklusa samog popisa.



Slika 5. Primjer Scrum ploče sa zadacima [17]

Za upravljanje Scrum-om već postoji mnoštvo programskih rješenja, a trenutno jedno od najpopularnijih je Trello (Slika 6).



Slika 6. Primjer digitalne Scrum ploče [18]

Niti jedno rješenje nije idealno za sve. Tako su prednosti Scruma njegova transparentnost svim članovima tima u svakom trenutku, veća kvaliteta proizvoda i njegova stabilnost. Mane su odvojeni timovi, nedostatak specijalizacije unutar timova te moguće teškoće kada timovi ovise o drugima, kako bi završili proizvod.

1.2.2. Kanban

Riječ „kanban“ potječe iz japanskog jezika i u doslovnom prijevodu označava „vidljivu karticu“. Kanban je metoda upravljanja razvoja proizvoda. Razvila ga je Toyota 1940-ih godina, uvodeći model proizvodnje „upravo na vrijeme“ (eng. *just in time, JIT*), odnosno proizvodnje na zahtjev. U središtu ovog načela nalazilo se smanjenje nepotrebnog posla bez smanjenja produktivnosti, pružajući maksimalnu vrijednost kupcu, bez stvaranja dodatnih troškova [19].

Naglasak u Kanban metodi je na neprekidnoj isporuci, na način koji ne opterećuje razvojni tim. I u ovoj agilnoj metodi izražena je suradnja timova. Scrum i Kanban polaze od postavke *Prestani počinjati, počni završavati!* Kanban promovira neprekidnu suradnju i potiče na aktivno i stalno učenje. To je metoda upravljanja tokom rada koja pomaže u vizualizaciji rada i omogućavanju najbolje moguće učinkovitosti.

Kanban se temelji na trima glavnim načelima [20]:

1. *Vizualizirati današnji rad* jer vizualni prikaz svih stavki u međusobnom odnosu na Kanban ploči može biti vrlo informativan.
2. *Ograničiti količinu zadataka na kojima se radi* (eng. *Work In Progress, WIP*) kako bi se izbjegla preopterećenost.
3. *Unaprijediti tok* – tek po završetku zadatka kreće se s radom na sljedećoj stavki najvišeg prioriteta.

Kanban se sastoji od četiri ključne komponente [21]:

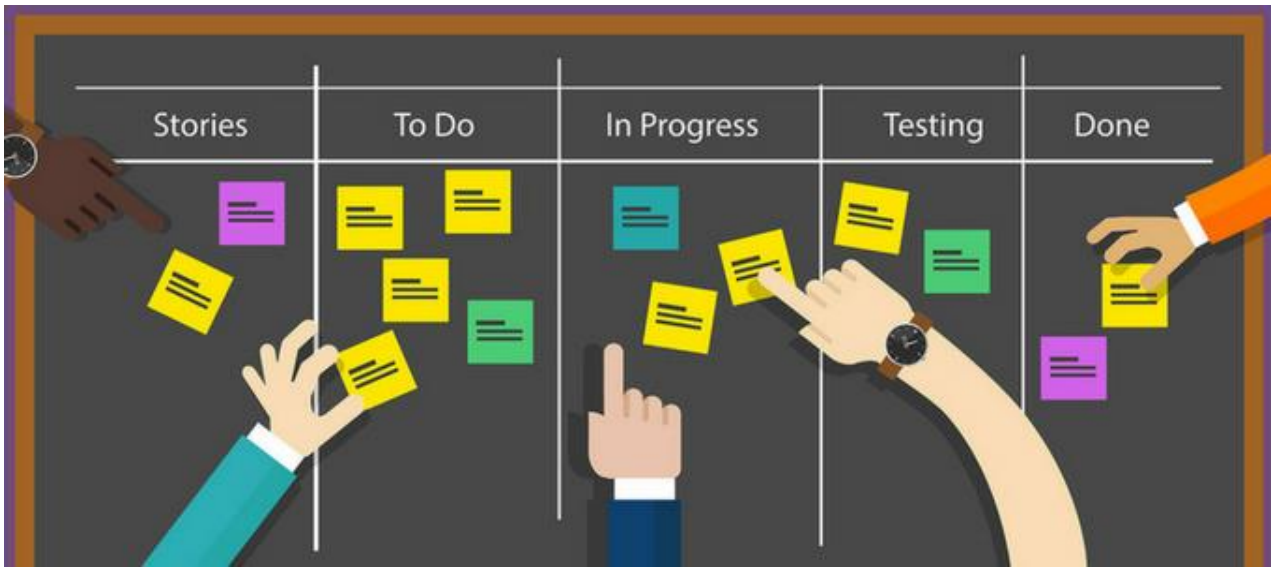
1. Kanban karte,
2. Kanban stupci,
3. WIP, te
4. Kanban staze sa zadacima.

Vlasnici, voditelji projekta i bilo tko s odlučujućom ulogom, sastavljaju opsežan popis stavki za rad. Vizualizacija napretka priče ključna je za transparentnost Kanbana. Priča se kreće kroz stupce i/ili retke. Svaki stupac/redak ima ograničen broj stavki. Kada se dosegne maksimum, nove se stavke ne mogu više dodavati u taj stupac/redak sve dok se u njemu ne oslobodi mjesto. Primjerice, ako jedna stavka čeka nečije odobrenje, nove stavke se ne mogu dodavati sve dok se ne završi stavka na čekanju i ukloni iz stupca. Na taj način postiže se ograničenje rada u provedbi. U čistom Kanbanu nema ciklusa koji iziskuju novu iteraciju nakon određenog vremena. U Kanbanu se proizvod ili projekt, odnosno njihove kontinuirane inačice, puštaju u rad čim su dovršeni. Prednost Kanbana je u brzom usklađivanju poslovnog toka s promjenama te u minimaliziranju posla koji ne daje rezultate. Idealan je za rad gdje često dolazi do promjena prioriteta proizvoda ili projekta [20].

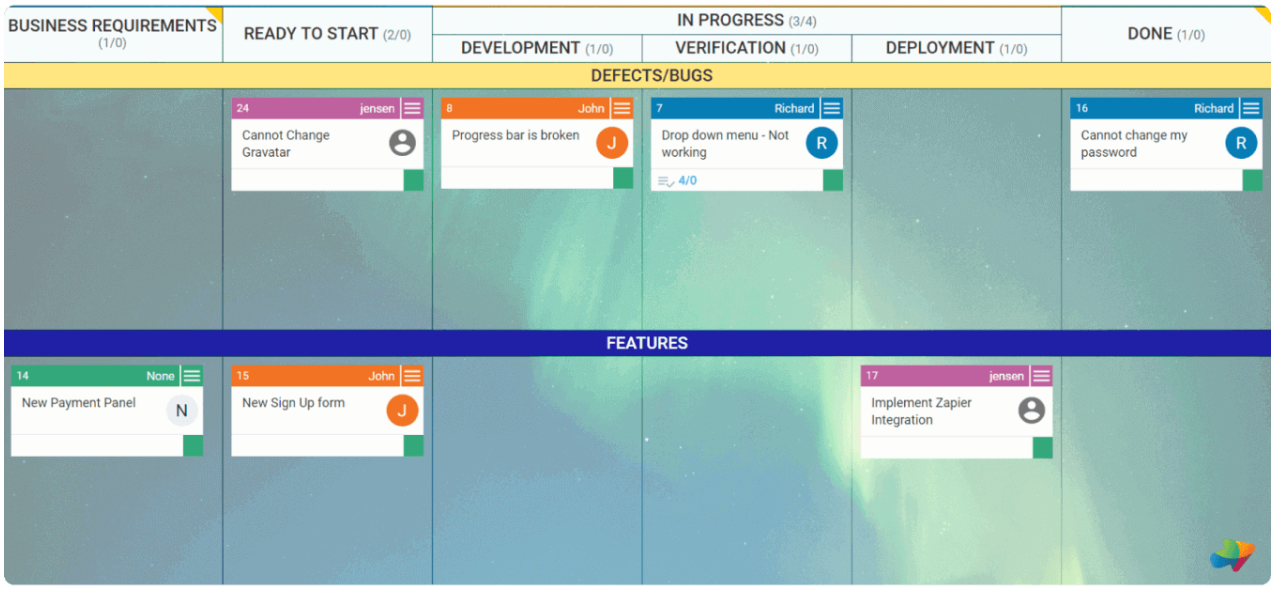
Mnoge organizacije uvode Kanban kako bi uvele red u poslovanju. On pomaže napraviti što više posla u što manje vremena i sa što manje troška. Kanban ploča (Slika 7, Slika 8, Slika 9) je izvor informacija i središnje mjesto okupljanja tima. Svi su upućeni u posao i znaju što trebaju raditi. U radu s pločom vidljivo je na kojim je mjestima previše kartica jer je taj dio zatrpan zadacima, te se odmah pristupa rješenju tog problema.

Kanban je fleksibilan. On potiče kontinuirani napredak, ništa radikalno i revolucionarno, već razvojno. Kanban ne forsira puštanje proizvoda u rad, već je orijentiran na prilagodbu

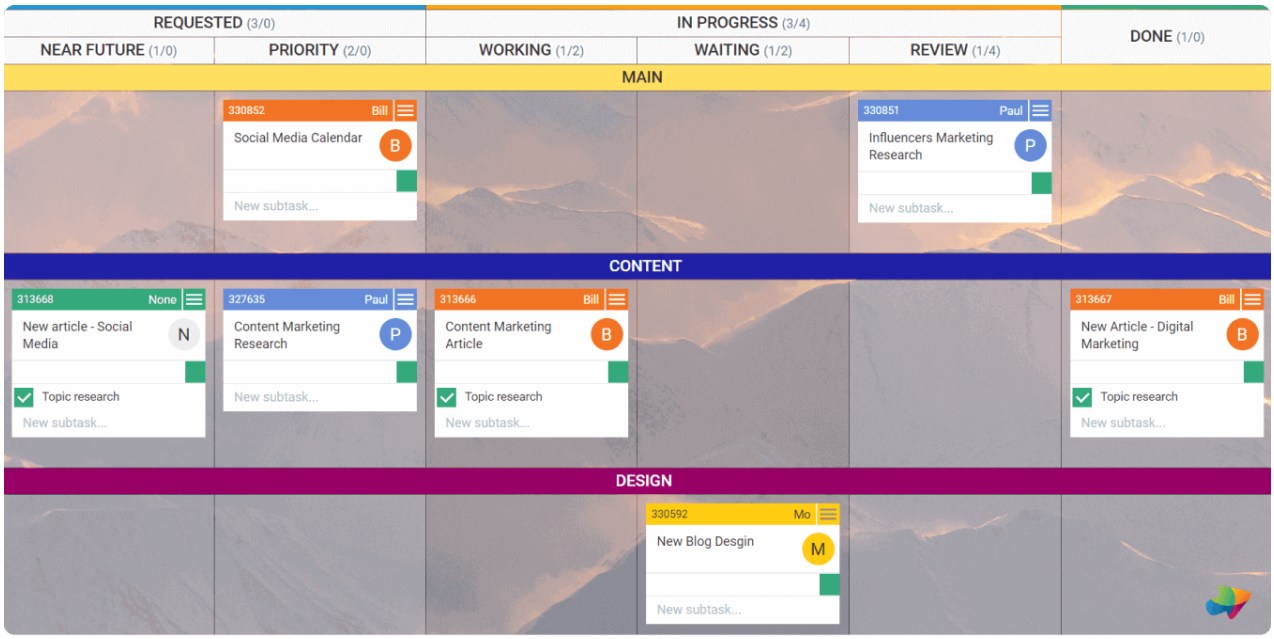
korisničkim potrebama. Članovi tima usredotočeni su na završetak zadatka na kojem rade. Tek nakon toga prelaze na novi zadatak. Ovo je postignuto na način da se ograničava WIP (engl. *work in progress*), odnosno 'radovi u tijeku'. Članovi tima surađuju dok rade na ograničenoj količini zadataka i tako izbjegavaju *multitasking* i povećavaju svoju produktivnost [20].



Slika 7. Kanban ploča [22]

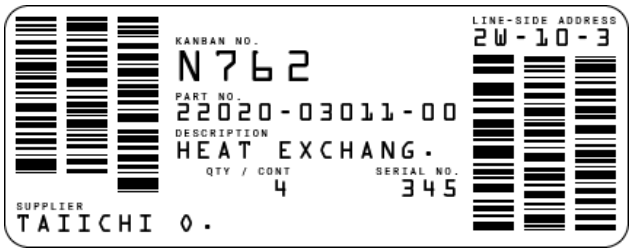


Slika 8. Predložak digitalne Kanban ploče za razvoj softvera [23]



Slika 9. Predložak digitalne Kanban ploče za marketing [23]

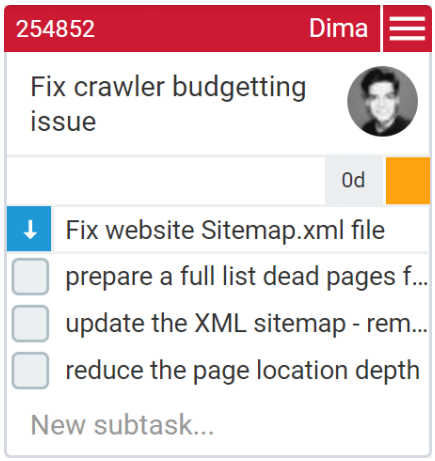
Kanban kartica vizualno predstavlja stavku posla koja je u fazi zahtjeva ili se na njoj radi. Ona se nalazi u središtu Kanban metode. Sadrži vrijedne informacije o zadatku i njegovom statusu kao što su npr. kratke informacije o zadatku, odgovorna osoba, rok i sl. Njezina svrha je vizualni prikaz životnog ciklusa zadatka, od trenutka kada je on stvoren do njegovog izvršenja. One su središte informacija, smanjuju potrebu za sastajanjem, unapređuju transparentnost radnog procesa. Kao što je već spomenuto, broj kartica na ploči treba biti ograničen.



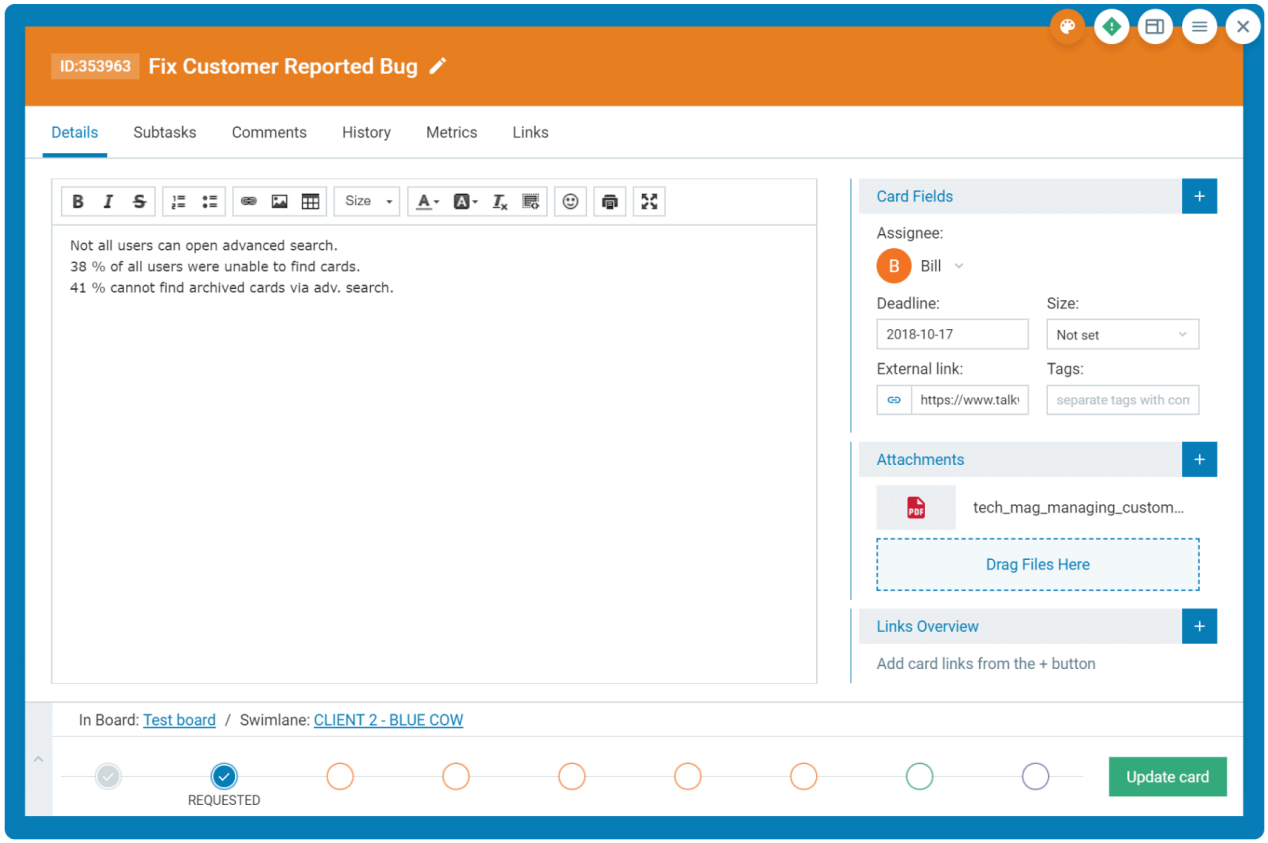
Slika 10. Fizička Kanban kartica [24]

Slika 10 prikazuje papirnatu Kanban karticu koju je koristila Toyota. Danas naravno da se koriste suvremene, digitalne kartice (Slika 11, Slika 12) koje sadrže daleko više informacija, lakše im se pristupa lokalno i udaljeno. Digitalne Kanban kartice imaju prednju stranu i poleđinu. Na prednjoj strani nalaze se glavne informacije o zadacima, poput opisa, naziva, osobe zadužene za izvršavanje zadatka, prioritetima i pod-zadacima. Ovo su podaci koji

su vidljivi na Kanban ploči. Na poleđini se mogu prikupljati važni statistički podaci tijekom vremenskog toka, mogu se ostavljati komentari, dodati datoteke i internetske poveznice. Ovaj dio kartice vidljiv je kada se ona otvori [24].



Slika 11. Primjer prednje strane digitalne Kanban kartice [24]



Slika 12. Primjer poleđine digitalne Kanban kartice [24]

Digitalne Kanban kartice mogu se prilagoditi potrebama korisnika. Svaki član tima koji ima pristup Kanban ploči može prilagoditi način pregleda kartica tako da su odjednom vidljive

sve informacije o zadatku ili samo dio njih. Programska rješenja za Kanban nude razne predloške za razne vrste poslova. Kanban se sastoji od četiriju temeljnih načela i šest praksi [19]:

Načelo 1: Započeti s onim što je poznato

Kanban je fleksibilan i u njega se lako može integrirati postojeće tokove rada. Pritom neće doći do neželjenih izmjena, ali ukazat će se na neke teškoće koje se trebaju riješiti. Kanban se može uvesti postupno na sve vrste organizacija jer ne postoji potreba za unošenjem velikih promjena već na samom početku.

Načelo 2: Odluka o postupnoj, razvojnoj promjeni

Kanban je osmišljen kako bi njegova primjena izazvala minimalan otpor i stoga potiče postupne promjene tekućim procesima. Općenito, u Kanbanu se ne provode temeljne promjene jer su uzrok strahu od nepoznatoga.

Načelo 3: Poštovati aktualne postupke, uloge i zaduženja

Kanban shvaća vrijednost tekućih postupaka, uloga i zaduženja te da ih treba sačuvati. On ne brani promjenu, ali ju isto tako ne promiče kao univerzalno rješenje za sve. Kanban promiče postupnu promjenu, bez izazivanja straha od same promjene.

Načelo 4: Poticaj ljudima da budu vođe na svim razinama

Ovo načelo nas podsjeća da primjeri najboljeg vođenja dolaze od ljudi koji su na prvoj liniji svojih timova. Svi sudionici trebaju imati na umu neprekidan razvoj kako bi postigli optimalnu izvedbu na razini tima, odjela, tvrtke.

Iako je najvažniji korak samo donošenje odluke za upotrebu metode Kanban, posebna je pozornost potrebna kod njegove primjene. Stoga je David Anderson prepoznao šest suštinskih praksi koje moraju biti prisutne kako bi se Kanban uspješno primijenio.

Praksa 1: Vizualizirati tok rada

Prvi korak je vizualizirati tok rada. Tek kada se uoči kako trenutno funkcionira tok rada, možemo ga unaprijediti potrebnim prilagodbama. Za vizualizaciju je potrebna ploča s karticama (jap. *Kanban*) i stupcima. Svaki stupac na ploči predstavlja korak u toku rada, a svaka kartica stavku na kojoj se radi. Kada počnemo raditi na određenoj stavci, preuzmemo ju iz stupca *Za napraviti* (engl. *To Do*), a kada završimo posao premjestimo ju u stupac *Završeno* (engl. *Done*).

Praksa 2: Ograničiti količine posla na kojem se radi

Kanban primarno služi tome da se istovremeno radi na izvedivom broju aktivnosti. Stoga nije preporučljivo započeti raditi na prevelikom broju procesa istovremeno. Ako nema ograničenja količine posla, to nije Kanban. Kartice iz stupca *Za napraviti* treba preuzimati samo kada ima mjesta za njih. Na ovaj način lako će se uočiti eventualni problem i riješiti ga.

Praksa 3: Upravljati tokom

Sama ideja Kanbana leži u glatkom toku, odnosno kretanju stavki kroz proizvodni proces. Idealan tok je brz i gladak. Stoga, upravljanje tokom je upravljanje poslom, kako bi se on što prije završio, a ne upravljanje ljudima tako da im se, na tradicionalan način, neprekidno daje posla kako bi bili zauzeti i imali privid produktivnosti.

Praksa 4: Jasno definirati politiku procesa

Ne može se usavršiti nešto što je nepoznato. Stoga se proces moram jasno definirati na svima vidljivom mjestu. Kada svi budu upoznati s glavnim ciljem, moći će donositi odluke o potrebnim promjenama, kako bi se kretalo u pravom smjeru.

Praksa 5: Povratne informacije

Da bi se došlo do njih, potrebni su redoviti, odnosno dnevni sastanci timova, zbog usklađivanja u radu i razmjene znanja. Sastanci su brzi (10-15 minuta) i zbog toga se na njima stoji, kao i u Scrumu, a održavaju se ispred Kanban ploče. Svaki član tima izvještava prisutne što su postigli prethodni dan i na čemu će raditi taj dan. Postoje i sastanci za pregled usluga, operativni pregled i pregled rizika. Učestalost održavanja ovih sastanaka ovisi o mnogim čimbenicima, a ono što im je zajedničko jest da su redoviti, zakazani za stalni termin, izravni i nikad duži nego što je neophodno (do sat vremena, ovisno o veličini tima i dnevnom redu).

Praksa 6: Unapređivati zajednički

Neprekidno usavršavanje i održive promjene unutar organizacije mogu se postići zajedničkom vizijom bolje budućnosti i kolektivnog shvaćanja prepreka koje je potrebno savladati. Timovi moraju biti upoznati s cijelim poslovnim procesom kako bi izgradili kolektivno shvaćanje problema i predložili korake k unapređenju.

Dobro je krenuti od najjednostavnije Kanban ploče na kojoj se nalaze tri stupca: *Zahtjevi*; *U postupku*; *Gotovo*. Kada je ploča dobro osmišljena, potrebno ju je koristi prema pravilima Kanbana. Ona je okupljalište informacije u stvarnom vremenu i ističe teškoće u sustavu i bilo kakve prepreke u toku posla [19].

1.2.2.1. Scrum ili Kanban?

Odluka o izboru Scruma ili Kanbana može biti teška. Za odabir odgovarajuće metode važno je znati koje su im sličnosti, ali i poznavati njihove razlike. Scrumu i Kanbanu je zajednički fokus na završavanju proizvoda ili projekta što prije, uz puštanje u pogon funkcionalnosti odmah po njihovom završetku. Neke timove će unaprijed zacrtani vremenski okvir obeshrabrivati, dok će drugima biti motivacija za produktivnost. Odabir će se donijeti na temelju organizacijskih i kulturoloških prilika, te odnosa među timovima.
Dvije glavne razlike su da se Scrum usredotočuje oko vremenskog okvira trajanja ciklusa, dok je kod Kanbana naglasak na kontinuiranom puštanju u rad. Čim je jedna značajka, projekt ili priča gotova, prelazi se na slijedeću. Scrum je više kruć od Kanbana i kod planiranja sastanaka i dodijeljenih uloga članova tima. Primjerice, sastanak za planiranje ciklusa može trajati do osam sati (ovisno o dužini ciklusa) i jedan je od glavnih nedostataka Scruma. Kanban je puno ležerniji. Od članova timova očekuje se neprestano usavršavanje procesa na kojima rade te kontinuirano puštanje u rad proizvoda, odnosno njihovih značajki [21].

Tablica 1. Prikaz ostalih glavnih razlika između Scruma i Kanbana. [20], [21]

Scrum	**Kanban**
Zadaci se izvršavaju u ciklusima koji su vremenski ograničeni na dva do četiri tjedna. Naglasak je na ciklusima.	Ne radi se u vremenski ograničenim ciklusima. Timovi odabiru zadatke s popisa prema prioritetu. Naglasak je na brzini.
Proizvod se pušta u pogon nakon jednog ili više ciklusa.	Proizvod se kontunuirano pušta u pogon kada god je za to spreman.
Nije dozvoljeno unošenje izmjena tijekom provedbe ciklusa.	Proces se može prilagoditi izmjenama u bilo kojem trenutku.

Uloge članova tima su Scrum master, vlasnik i razvojni tim.	Nema zadanih uloga članova Kanban tima.
Veliki je naglasak stavljen na ravnomjernu funkcionalnost cijelog tima. Timovi nemaju specijalizaciju.	Članovi tima mogu biti stručnjaci za određeno područje i odabirati zadatke u skladu s time. Međutim, previše specijaliziranih članova tima umanjit će njegovu učinkovitost.
Naglasak je na dnevnim sastancima, planiranju ciklusa i osvrtima na ciklus.	Naglasak je na neprekidnom usavršavanju procesa, ali bez standardiziranih, redovitih sastanaka.
Prikladniji za situacije gdje se poslovi mogu grupirati u blokove, prema prioritetu.	Prikladniji za okolinu gdje dolazi do varijabilnosti.
Članovi tima rade na zadacima za koje su zaduženi unutar pojedinog ciklusa. Kada je ciklus završen, na Scrum ploči više nemaju zadatke koji čekaju svoj dovršetak.	Članovi tima imaju određen maksimalan broj zadataka na kojima istovremeno rade. Kada se završi jedan zadatak, preuzima se drugi, bez čekanja na kraj ciklusa. Na taj način postiže se tok, koji je karakterističan za Kanban.

Pri odabiru je potrebno uzeti u obzir sklonosti članova tima i njihove značajke. Dobro je postaviti pitanje jesu li timovi skloniji rigidnijem (Scrum) ili ležernijem pristupu (Kanban). Ukoliko je timu potreban netko tko će biti između njih i viših instanci, dobro je imati posrednika - Scrum Mastera i vlasnika.

Odabir pristupa nije konačan, može se krenuti s jednim pristupom da bi se shvatilo da on nije optimalan te prijeći na drugu metodu. Dakako, postoji i mogućnost hibridne metodologije – kombinacije elemenata i Scruma i Kanbana, koji najbolje odgovaraju produktivnosti članova tima. Ono što se nikako ne preporuča je forsiranje jednog od pristupa. Agilna metodologija zalaže se za neprestanu prilagodbu, kako proizvoda, odnosno projekata, tako i same metodologije rada [21].

1.2.3. Scrumban

Kao što se može zaključiti iz samog naziva ove agilne metodologije, radi se o hibridu Scruma i Kanbana. On je odgovor potrebi za smanjenjem količine posla na kojem se istovremeno radi, odnosno potrebi za preuzimanjem određene količine zadataka prema prioritetu, u zadanom trenutku. Na taj način članovi tima imaju fleksibilnost reagirajući na potrebu za promjenom i nisu preopterećeni količinom zadataka. Scrumban preuzima strukturu Scruma te fleksibilnost i vizualni aspekt Kanbana. Scrumban može biti koristan i kao prijelazna faza iz Scruma u Kanban.

Elementi Scruma koji su uključeni u Scrumban su [25]:

1. iterativno planiranje u učestalim vremenskim razmacima, s osvrtima i pregledima
2. odluka koliko posla se može obaviti u ciklusu, ovisno o složenosti posla i dužini ciklusa
3. postavljanje prioriteta prema potraživanju – radi se točno ono što se traži
4. odgovarajuća razina analize prije nego što se krene na idući korak

Elementi Kanbana koji su uključeni u Scrumban su [25]:

1. Preuzimanje zadataka s popisa poslova prema prioritetu i neprekidni tok rada
2. ograničenje količine preuzetih zadataka
3. uloge članova tima nisu jasno definirane
4. analiza i planiranje provode se neposredno prije izvođenja posla, a ne na početku dužeg vremenskog razdoblja (ciklusa)
5. nedostaci i mogućnosti poboljšanja vidljivi su iz vizualnog prikaza toka rada

Scrumban je idealan za timove kojima je potrebna struktura Scruma i flesibilnost Kanbana ili za timove koji žele preći na Kanban.

1.2.4. Scrum of Scrums

Timovi u Scrumu najučinkovitiji su kada ih čini tri do devet osoba. Ako je manje članova tima, ne mogu završiti posao predviđen za ciklus niti mogu kvalitetno surađivati. Ako ih je previše, samoorganizacija, koja je karakteristična za Scrum, je praktički nemoguća. [26]

No, što se događa ako razvoj proizvoda ili projekta iziskuje rad od čak stotinjak osoba koji ne rade samo u različitim uredima, već čak i u različitim zemljama? Ako je tim prevelik, jednostavno se oformi više Scrum timova. Svaki od njih funkcionira na način da imaju svoje cikluse, popise stavki za proizvod i svoje dnevne sastanke. Da bi svi timovi radili usklađeno, održavaju se sastanci – Scrum of Scrums (SOS, Meta-Scrum). Za razliku od sastanaka članova Scrum tima, ovom sastanku ne prisustvuju svi članovi već predstavnici (jedan ili više) svakog Scrum tima [27]. Sastanci se održavaju dnevno, ili rjeđe, ovisno o potrebi. Na sastanku sudjeluju predstavnici svakog tima, a dnevni red je isti kao i na dnevnom Scrum sastanku, odnosno odgovara se na tri slična pitanja:

- Što ste radili od prošlog sastanka što bi moglo utjecati na druge timove?;
- Što će vaš tim raditi do idućeg sastanka što bi moglo utjecati na druge timove?;
- Postoje li kakve prepreke u vašem radu u čemu drugi tim može pomoći?

Sastanak traje 15 minuta i na sastanku se iz toga razloga stoji [28]. Na sastancima sudjeluje tehničko osoblje, odnosno članovi razvojnog tima. Sudjelovati može i Scrum Master, ali tek kao druga osoba iz tima.

2. Testiranje softvera

2.1. Načela i metode testiranja

Zbog kompleksnosti, niti jedan softver ne može se izraditi bez ikakvih grešaka. Testiranje softvera postupak je kojim se procjenjuje zadovoljava li gotovi proizvod zahtjeve koji su prvobitno definirani te otkriva prisutnost grešaka. Testiranje osigurava isporuku kvalitetnog proizvoda na vrijeme. Pod kvalitetnim proizvodom podrazumijeva se proizvod koji je u realnoj mjeri bez grešaka, isporučen na vrijeme, zadovoljava unaprijed postavljene zahtjeve i očekivanja te ga se može održavati. [29] Testiranjem se također mogu uočiti funkcionalnosti koje nedostaju. Prema standardu ANSI/IEEE 1059, testiranje je postupak analize softvera kako bi se otkrila odstupanja između postojećeg stanja i postavljenih zahtjeva [30].

U najvećoj mjeri, testiranjem se mjere specifikacije, funkcionalnosti i performanse softvera [31]. Više je razloga za testiranje. Primjerice, isplativost. Ako se potkradu greške, teže ih je detektirati i ukloniti, što povećava trošak proizvodnje. Nekada se tijekom ispravljanja grešaka pojave nove. Zato ih je potrebno pronaći tijekom ranijih faza razvoja softvera, odnosno uvesti testiranje nakon završetka svake faze.

Zadovoljstvo korisnika krajnji je cilj svakog proizvoda, a testiranjem se poboljšava korisničko iskustvo. Sigurnost je vjerojatno jedan od najosjetljivijih razloga testiranja jer nitko ne želi kupiti riskantan proizvod. Također, i kvaliteta je još jedan od kriterija koji mora biti zadovoljen kako bi se potencijalni kupac odlučio na kupovinu [30].

Kod testiranja softvera važno je doći do optimalnih rezultata bez odudaranja od cilja. Da bismo bili sigurni da primjenjujemo dobru strategiju testiranja potrebno je slijediti sedam načela testiranja, a oni su [32]:

Načelo 1. Iscrpno testiranje je nemoguće

Iscrpno testiranje je testiranje svih funkcionalnosti, korištenjem svih mogućih kombinacija ulaznih podataka. Ono je nemoguće jer takvo testiranje traje iznimno dugo vremena i nije isplativo. Umjesto toga, u obzir se uzimaju određeni (prihvatljivi) rizici koji mogu nastati u radu softvera.

Načelo 2. Klasteri pogrešaka

Pogreške nisu ravnomjerno raspoređene već se pojavljuju u klasterima. Prema Peretovom načelu, većina pogrešaka (80%) je grupirana u manjem broju modula (20%), što osoba koja izvršava testiranje može predvidjeti prema svome iskustvu. Preostalih 20% pogrešaka nalazi se u ostalih 80% modula.

Načelo 3. Paradoks pesticida

Kada se kukce tretira istim pesticidom, nakon nekog vremena steknu otpornost na njega. Isto se događa i sa softverom. Nakon što se softver neprekidno testira istim testnim slučajevima, nakon nekog vremena neće se više pronaći greške. Da bi se tome doskočilo, potrebno je redovito revidirati i ažurirati testne slučajeve.

Načelo 4. Testiranje ukazuje na nedostatke

Svrha testiranja je pronaći greške. Ako ih osobe koje testiraju softver ne mogu pronaći, to ne znači da ih nema.

Načelo 5. Izostanak greške - zabluda

Može se dogoditi da je softver koji je 99% bez grešaka i dalje neiskoristiv. Razlog tome može biti što je testiran za krivu svrhu. To dovodi do idućeg načela – testiranja u ranim fazama razvoja.

Načelo 6. Rano testiranje

Testiranje je dobro provesti u ranijim fazama razvoja proizvoda jer je tada jeftinije uočiti i ispraviti greške.

Načelo 7. Testiranje ovisi o kontekstu

Testiranje se razlikuje ovisno o kontekstu softvera, odnosno njegovoj namjeni.

Testiranje se može podijeliti na dvije metode [30], [31]. Prva metoda, *statičko testiranje ili verifikacija* provjerava dokumentaciju i datoteke te odgovara na pitanje *Izrađujete li proizvod pravilno*? Aktivnosti koje ona uključuje su inspekcije, pregledi (eng. *reviews*) i prolazi (eng. *walkthroughs*). Druga metoda, *dinamičko testiranje ili validacija* odnosi se na testiranje stvarnog proizvoda, brine se da softver odgovara postavljenim zahtjevima te odgovara na pitanje *Izrađujete li pravi proizvod*?

2.1.1. Ručno i automatizirano testiranje

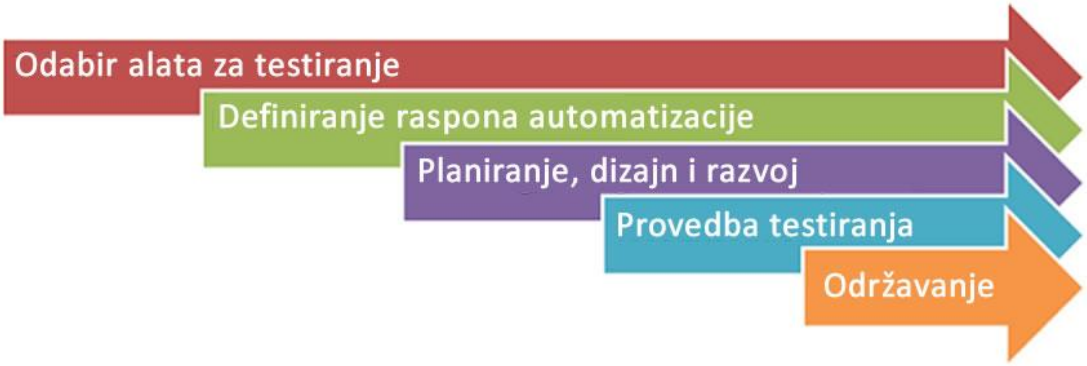
Testiranje se može podijeliti u dva osnovna tipa - ručno i automatizirano. [31] Ručno testiranje izvodi se bez ikakve automatizacije ili skripte. Tester preuzima ulogu krajnjeg korisnika te pronalazi nepredviđeno ponašanje softvera i greške. Unutar ručnog testiranja postoje faze poput testiranje jedinica, integracijsko, sistemsko testiranje te testiranje prihvaćanja korisnika(eng. *user acceptance testing*).

Mogu se koristiti planovi, studije i scenariji. Ručno testiranje uključuje i istraživačko testiranje budući da se softver istražuje u potrazi za greškama. [31] Sve nove aplikacije moraju se ručno testirati prije automatiziranog testiranja, a ono je zahtjevnije od automatiziranog testiranja. Međutim, i neophodno je zbog pretpostavke da niti jedno testiranje nije moguće biti u potpunosti automatizirano.
Vrste ručnog testiranja su:

- **Testiranje crne kutije** (detaljnije se obrađuje u ovom priručniku)
- **Testiranje bijele kutije** (detaljnije se obrađuje u ovom priručniku)
- **Testiranje jedinica**
  Testira se individualna jedinica softvera, odnosno komponenta, ili skupina jedinica koje su u međusobnom odnosu.
- **Sistemsko testiranje**
  Testira se softver u više operativnih sustava. Spada u Black Box tehniku. Ne testira se rad proizvoda, već samo očekivani ulaz i izlaz. Sastoji se od sigurnosnog testiranja, testiranja oporavka, stres testiranja (testiranje u nepogodnim okolnostima) i testiranja performansi (učinkovitost i brzina).
- **Integracijsko testiranje**
  Testira se povezanost ili prijenos podataka između nekoliko komponenti. Cilj je izgraditi strukturu programa koristeći testirane komponente softvera. Podvrste su: odozgo prema dolje, odozdo prema gore i sendvič (kombinacija).
- **Testiranje prihvaćanja od strane korisnika**
  Nužno da bi se proizvod isporučio i naplatio. Podvrste su alfa, beta i gama testiranje.

Automatizirano testiranje izvodi se upotrebom skripte koju je tester napisao ili nekog drugog softvera za testiranje. Podrazumijeva automatizaciju ručnog postupka, a koristi se

za ponovno testiranje koje je prvobitno izvedeno ručno, ali puno brže i više puta. Uz regresijsko testiranje, automatizirano testiranje se koristi i za testiranje aplikacije od trenutka učitavanja, testiranjem performansi i slabih točki. Neke od prednosti automatiziranog testiranja su povećani raspon testiranja, veća točnost, ušteda vremena i novca, u odnosu na ručno testiranje. Naročito je pogodno u situacijama kada je ručno testiranje nepouzdano, primjerice, kod testiranja višejezičnih proizvoda. Ne iziskuje ljudski nadzor pa se može provoditi tijekom noći i daleko je brže od ručnog testiranja. Obavezno se provodi za testiranje koje je dugotrajno, dosadno ili teško za provesti. Ipak, nije pogodno za testiranje novih proizvoda ili ako ručno testiranje nije provedeno barem jednom, kod proizvoda koji se neprestano mijenjaju te kod povremenih i neformalnih testiranja.



Slika 13. Koraci u automatskom testiranju [33]

Neke od prednosti automatiziranog testiranja su sljedeće: 70% je brže od ručnog testiranja, široko područje primjene, pouzdani rezultati, dosljednost, ušteda vremena i novca, poboljšana točnost i učinkovitost, skripte za testiranje mogu se ponovno koristiti, te tijekom provedbe testiranja nije potrebna ljudska intervencija [33].

Automatizirane vrste testiranja su:

- **Dimno testiranje**

  Ovim testiranjem se ispituje je li proizvod spreman za daljnja testiranja. Zove se dimno testiranje jer se ispituje je li se proizvod *zapalio* ili *zadimio* tijekom inicijalnog puštanja u rad.
- **Jedinično testiranje**

  Najčešće automatiziran postupak testiranja kojim se provjerava da li se pojedini dijelovi programskog koda (jedinice) izvršavaju kao što je to i predviđeno.
- **Integracijsko testiranje**

Pojedini dijelovi aplikacije (moduli) se međusobno kombiniraju i testiraju kao cjelina.

- **Funkcionalno testiranje**
  Tip testiranja crne kutije koji za određene ulazne podatke provjerava da li su dobiveni odgovarajući izlazni podaci.
- **Regresijsko testiranje**
  Nakon dodavanja novih promjena provodi se ponovno testiranje prethodno implementiranih funkcionalnosti kako bi se uvjerili da one i dalje rade.
- **Testiranje pogonjeno podacima**
  Parametrizirano testiranje gdje se ulazni i izlazni testni podaci čitaju iz tablice.
- **Testiranje crne kutije** (detaljnije se razrađuje u nastavku)

Razlike između ručnog i automatiziranog testiranja mogu se svesti na neke osnovne: [34]

- Ručno testiranje podrazumijeva ljudski faktor, dok automatizirano testiranje koristi alate.
- Ručno testiranje iziskuje vještinu, puno vremena i sredstava. Automatizirano štedi vrijeme, sredstva i iziskuje manji broj ljudi. Bilo koja vrsta aplikacije može se testirati ručno, a neke su čak i pogodnije za to.
- Automatizirano testiranje pogodno je samo za stabilne sustave i najčešće se koristi za regresijsko testiranje.
- Ručno testiranje može postati ponavljajuće i dosadno pa dolazi do grešaka.

Treba naglasiti da testiranje nije 100% pouzdano. Testiranje ne jamči da grešaka nema već da one nisu uočene. Nemoguće je otkriti sve moguće nedostatke u softveru, a svako naknadno dodavanje novih funkcionalnosti zahtjeva ponovno testiranje svih dijelova aplikacije koji koriste novu funkcionalnost. U praksi, najčešće će biti potrebna kombinacija ručnog i automatiziranog testiranja kako bi u što boljoj mjeri pokušali pronaći nedostatke i greške u softveru.

2.1.2. Testiranje crne kutije

Testiranje crne kutije (engl. *black box testing*) provodi se iz korisničke perspektive. Provodi se na razini korisničkog sučelja, a tester nema pristup programskom kodu. Naziva se još i testiranje po načelu zatvorene kutije, odnosno podatkovno testiranje. Provode ga krajnji

korisnici, testeri i programeri, a temelji se na unaprijed postavljenim zahtjevima, odnosno ispitivanju jesu li oni udovoljeni.

Testiranje crne kutije pomaže u pronalasku propusta u mogućnostima koje softver pruža. On pruža pregled rada softvera i njegovog izlaza. Može se koristiti za testiranje cjelokupnog razvojnog procesa, ali i pojedinih dijelova kao što su jedinice, moduli, integracija, sustav i regresija. Provjerava se izvršavaju li korisnici ispravne operacije, interakciju sustava i ulaza, vremenski odaziv, korištenje struktura podataka, teškoće s korisničkim sučeljem, s iskoristivošću, performansama, naglim prekidima u radu, nemogućnost pokretanja, zatvaranje i slično. Ova metoda općenito je korisna za ispitivanje upotrebljivosti proizvoda [35].

Više je pristupa *black box* testiranju. Kod ručnog testiranja, tester iz aspekta korisnika provjerava korisničke podatke i poruke o greškama. Prilikom automatiziranog testiranja, snima se interakcija sustava i korisnika kako bi se otkrilo greške, dok tijekom dokumentacijskog testiranja tester provjerava ulaz i izlaz.

Više je vrsta testiranja crne kutije, pa se navode samo najvažnije**. Funkcionalno testiranje** ispituje funkcionalne zahtjeve sustava i provode ga testeri za softver. **Nefunkcionalno testiranje** odnosi se na nefunkcionalne zahtjeve kao što su performanse, skalabilnost i iskoristivost. **Regresijsko testiranje** provodi se nakon popravaka koda, održavanja i ažuriranja inačica kako bi se ispitalo jesu li naknadne izmjene uzrokovale probleme s postojećim kodom i funkcionalnostima [36].

Testiranje se provodi prema unaprijed zadanim koracima [36].

1. **Ispitivanje zahtjeva i specifikacija aplikacije**
   U ovom koraku od velike je pomoći dokument (engl. *software requirement specifications document)* koji opisuje te zahtjeve, odnosno što softver radi i što se od njega može očekivati.
2. **Odabir valjanog ulaza** (scenarij pozitivnog testiranja) kako bi se testirao softver. Odabiru se i nevaljani ulazi kako bi se testiralo hoće li ih se moći prepoznati kao nevaljane.
3. **Utvrđivanje očekivanog izlaza za sve ulaze**
4. **Izrada testnih slučajeva za sve odabrane ulaze**

5. **Provedba testiranja**
6. **Usporedba stvarnih izlaza s očekivanima**
7. **Ponovno testiranje** (provodi se ukoliko se pojave greške ili neočekivano ponašanje).

Testiranje crne kutije ima svoj životni ciklus (engl. *software testing life cycle – STCL*). U početnoj fazi nalaze se *zahtjevi*, odnosno njihovo prikupljanje. Dalje, *planiranje testiranja i analiza,* faza je u kojoj se određuje tip testiranja, a ima ih preko stotinu, te se izrađuje plan testiranja kojim se utvrđuju mogući rizici i načini njihovog savladavanja. Sljedeća faza je *dizajn* tj. izrada testnih slučajeva i skripti na temelju dokumenta koji definira zahtjeve softvera iz početne faze ciklusa. Posljednja faza životnog ciklusa testiranja softvera je sama provedba testiranja. Ukoliko se otkriju kakve greške, nakon njihovog popravka ponovno se pristupa testiranju [36].

Prednosti testiranja crne kutije u odnosu na ostale metode su:

1. Tester ne mora imati nikakve tehničke vještine da bi ispitao sustav. Umjesto toga, važno je da radi svoj posao iz korisničke perspektive.
2. Testiranje se provodi nakon razvoja softvera i neovisno je o njemu.
3. Sagledava se šira slika cjelokupnog softvera.
4. Testni slučajevi mogu se generirati prije razvoja softvera i neposredno nakon specifikacije.
5. Ova metoda testiranja srodna je agilnoj.

Najveća mana testiranja crne kutije je nemogućnost testera da vidi programski kod kojim su implementirane određene funkcionalnosti aplikacije. Zbog toga je svakako poželjno pristupiti i testiranju bijele kutije kako bi tester uvidom u programski kod i dizajn aplikacije mogao bolje ispitati i rubne slučajeve i ponašanja u njenom radu.

2.1.3. Testiranje bijele kutije

Za razliku od prethodnog pristupa, pri testiranju bijele kutije (engl. *white box testing)*, tester zna na koji način funkcionira proizvod te ima pristup programskom kodu, a provodi se na način koji omogućava provedbu svih unutarnjih radnji prema specifikacijama. Ispituje se unutarnja struktura softvera, njegov dizajn i kod. Neophodno je poznavanje unutarnjeg

rada proizvoda i provode ga testeri i programeri, ne krajnji korisnici. Testiranjem se provjerava provode li se operacije prema specifikacijama. Naziva se još i prozirna, otvorena, staklena kutija, strukturalno testiranje i testiranje koda. Naziv dolazi od koncepta kutije, odnosno vanjske ljuske, koja je prozirna pa se kroz nju vidi unutarnji kod.

Testiranje bijele kutije prikladno je za kritične sustave i njegove komponente. Primjerice, za pitanja sigurnosti, loše strukturirane putanje u kodu, uvjetovanu funkcionalnost, za rad pojedinih naredbi, objekata i funkcija na individualnoj bazi. [37] Testiranje se može provesti na razini sustava, integracije ili na jediničnoj razini. Ispituju se svi mogući scenariji i zato je ono nezaobilazno. Testiranje na razini koda omogućava ispitivanje svih mogućnosti i kombinacija. Uključuje ispitivanje unaprijed definiranih ulaza i izlaza da bi se otkrilo izlaze koji se razlikuju od očekivanih tj. kako bi na taj način uočili greške.

Glavna tehnika testiranja bijele kutije je analiza pokrivenosti koda (engl. *code coverage analysis*). Ona otkriva dijelove programa koji nisu pokriveni testnim slučajevima i tada se izrađuju testni slučajevi. Nekoliko je tehnika za to [38]:

- **Područje djelovanja naredbe**
  Provjerava se može li se svaka linija koda izvesti barem jednom.
- **Područje grananja (odluke)**
  Za svaku moguću putanju (*if*, *else* i druge uvjetne naredbe) testira se provodi li se barem jednom. Dvije su moguće grane – True i False, pa tako i dva moguća testna uvjeta.
- **Područje putanje**
  Ispituju se sve putanje programa, odnosno da su sve prohodne barem jednom. Ova tehnika je korisna kod složenih programa.
- **Područje uvjetovanja, višestrukog uvjetovanja, funkcija itd.**

Svaka od tehnika ima svoje prednosti i zajedno pokrivaju gotovo sve dijelove koda. Prve dvije tehnike pokrivaju 80-90% koda i dovoljne su za pouzdanost testiranja. Greške koje se otkrivaju u testiranju bijele kutije najčešće su logičke greške koje nastaju prilikom implementiranja funkcija i metoda. Otkrivaju se i greške u dizajnu koje su nastale zbog razlike u toku između programa i njegove primjene. Primjećuju se i greške u gramatici i sintaksi koda.



Slika 14. Vrste testiranja bijele kutije [38]

Testiranje bijele kutije provodi se u sljedećim koracima [39]:

1. **Odabir funkcionalnosti, komponente ili programa za testiranje**
   Iako je moguće testirati čitav sustav, najbolje je testirati što manje jedinice (module) softvera. Testiraju se svi mogući scenariji za pojedine mogućnosti, a to je lakše učiniti temeljito kada se radi o manjoj cjelini. WBT provodi se najčešće jer su pojedini segmenti kritični i temeljitim ispitivanjem jamči se da rade kako se od njih očekuje. Kada se završi testiranje najmanjeg modula, prelazi se na sljedeći.
2. **Ucrtavanje svih mogućih putanja u grafikon toka**
   U ovoj fazi definiramo sve putanje koje se mogu testirati za određenu značajku, komponentu ili modul. To olakšava pisanje testnih slučajeva za svaki od njih. Za izradu grafikona toka potrebno je konzultirati korisničke slučajeve, specifikacije softvera, tehničke specifikacije, pseudo kod.
3. **Identifikacija svih mogućih putanji iz grafikona toka**
   Sve što je u ovoj fazi potrebno je prepoznati sve moguće putanje i njihove kombinacije te mjesta gdje dolazi do ispadanja (engl. *drop-off*).
4. **Pisanje testnih slučajeva za svaku pojedinu putanju na grafikonu toka**
5. **Provedba i ponavljanje**
   Sada je sve spremno za provedbu testiranja bijele kutije na određenom sustavu, komponenti ili modulu. Pripremljeni su grafikon toka i testni slučajevi neophodni za testiranje.

Prednosti testiranja bijele kutije su njegova temeljitost i pokrivanje cijelog koda, a rezultat je optimizirani kod bez grešaka i nepotrebnih linija koda. Za razliku od testiranja crne kutije,

ne zahtijeva sučelje. Lako ga je automatizirati i moguće ga je početi provoditi u ranim fazama razvoja programa. Nedostaci su njegova složenost, dugotrajnost i skupoća. Osobe koje vrše testiranje trebaju dobro poznavati kod i programski jezik. Također, najčešće je nemoguće izvršiti testiranje svake postojeće putanje rada softvera, zbog čega je nemoguće temeljito testirati velike sustave. Unatoč tome, testiranje bijele kutije je učinkovito prilikom testiranja najvažnijih putanja i struktura [38].

2.1.4. Testiranje crne ili bijele kutije – što odabrati?

Testiranje crne kutije provodi se iz korisničke perspektive, dok se testiranjem bijele kutije ispituje programski kod. Ovisno o sustavu, može se preporučiti testiranje bijele ili crne kutije, što često ovisi i o osobnim željama. Međutim, u stvarnosti je najbolje koristiti oba pristupa kako bismo doista bili sigurni u vjerodostojnost ishoda. Testiranje crne kutije usredotočuje se na iskustvo krajnjeg korisnika, odnosno kupca proizvoda. Istovremeno, testiranje svake pojedine linije koda otkriva greške koje se ne bi otkrile testiranjem iz korisničke perspektive. Također, otkrivaju se sporne situacije koje se smatraju iznimkama, za koje su inače mali izgledi da se pojave, ali ipak postoje [39].

2.1.5. Testiranje sive kutije

Kombinacija testiranja crve i bijele kutije naziva se, očekivano, testiranje sive kutije (eng. *gray* ili *grey box testing*). Pruža djelomičan uvid u unutarnju strukturu softvera, te je najpogodnije za web aplikacije i za funkcionalno testiranje.

Metoda testiranja sive kutije sastoji se od tri koraka. Najprije s provodi testiranje bijele kutije kako bi se stekao osnovni uvid u razumijevanje unutarnjih značajki softvera. Slijedi dizajniranje i definiranje testnih slučajeva, kako bi se obuhvatio svaki pojedini aspekt softvera, nakon čega se provodi testiranje crne kutije.

Za provedbu testiranja sive kutije nije neophodan uvid u kod. Testni slučajevi mogu se izraditi na temelju algoritma, arhitekture i drugih metoda, a testiranje se provodi u sljedećim koracima [40]:

1. Odabir i identifikacija ulaznih elemenata testiranja crne i bijele kutije.
2. Identifikacija najvjerojatnijih izlaza testiranja crne i bijele kutije.

3. Identifikacija svih ključnih putanja za dovršetak testne faze.
4. Identifikacija pod-funkcija za provođenje testiranja na dubljoj razini.
5. Identifikacija ulaza za pod-funkcije
6. Identifikacija vjerojatnih izlaza za pod-funkcije
7. Provedba testnih slučajeva za pod-funkcije.
8. Verifikacija prikladnosti ishoda.
9. Ponavljanje koraka 4 i 8.
10. Ponavljanje koraka 7 i 8.

Nekoliko je tehnika provođenja testiranja sive kutije [41]:

Testiranje matrice (eng. *matrix testing*) definira sve korištene varijable softvera. Ukoliko ne odgovara očekivanju, čitljivost (eng. *readability*) i brzina softvera bit će ugroženi. Testiranjem matrice uklanjaju se nepotrebne varijable na način da se definira koje se varijable koriste.

Regresijsko testiranje (eng. *regresion testing*) provodi se nakon dorada gotovog i već testiranog softvera, poput novih inačica, ispravljanja grešaka i sl. Prilikom ispravaka grešaka, softver može dobro funkcionirati u tom dijelu, ali isto tako može uzrokovati probleme u nekom drugom dijelu softvera i stoga je ponovno, regresijsko testiranje nezaobilazno.

Testiranje ortogonalnih nizova (eng. *orthogonal array testing, OAT*) namijenjeno je testiranju što većeg raspona koda, ali i grafičkog sučelja. Sastoji se od niza vrijednosti gdje su varijable predstavljene stupcima, a testni slučajevi redcima.

Testiranje obrazaca (eng. *pattern testing*) koristi zapise prethodnih analiza defekata sustava. Razlog tome je što se u tim zapisima mogu naći podaci o teškoćama koje su riješene. Primjenjiv je samo na proizvode koji imaju stariju inačicu, upravo zbog korištenja zapisa iz prethodnih testiranja.

Prednosti testiranja sive kutije su njegovo provođenje i iz perspektive korisnika i zato je lakše uočiti propuste zbog kojih softver može biti meta provala. Dopušta testerima

postavljanje prioriteta u testiranju ovisno o tome što je ciljani sustav, dok samo testiranje se usredotočuje na problematični dio i time je brže i jeftinije.
Nedostaci su nedostatak pristupa kodu zbog čega može doći do propusta u detekciji grešaka. Ako je programer već proveo slično testiranje, testiranje sive kutije je nepotrebno. Također, ovaj tip testiranja nije pogodan za testiranje algoritama. [40]

Imajući na umu sve prednosti i nedostatke, testiranje sive kutije je svakako preporučljivo jer pronalazi nedostatke u zaštiti proizvoda od zlonamjernih napada i može biti presudno za zaštitu od virtualnog kriminala.

2.2. Standardi

Standardi su dogovoreni propisi koje izdaju ovlaštena tijela za regulaciju i ujednačavanje postupaka, pa su stoga vrlo bitni i u postupku provedbe testiranja. Standardi pružaju zajedničke usuglašene smjernice i upute koje se temelje na uzajamnom shvaćanju i koriste se kao pomagala za mjerljivost. Omogućuju interaktivnost, upućuju na poštivanje s mjerodavnim zakonima i pravilima, uvode inovativne proizvode na tržište, te teže pružiti interoperabilnost između postojećih proizvoda, procesa i usluga.

Međunarodne standarde donosi nekoliko organizacija.

1. Međunarodna organizacija za standardizaciju (***International Standardization Organization, ISO***), sa sjedištem u Ženevi sastoji se od 164 države članice i donijela je preko 20.000 standarada.
2. Institut inženjera elektrike i elektronike (***Institute of Electrical and Electronic Engineers, IEEE***), s uredom u New Yorku i New Jerseyju, SAD, djeluje u preko 160 zemalja te donosi standarde iz elektrike, elektronike u računalnih znanosti.
3. Međunarodna elektrotehnička komisija (***Commission électrotechnique internationale, IEC***) osnovana je 1906. godine i čini ju 86 zemalja članica. Donosi standarde elektrotehnike - iz svih električkih, elektroničkih i srodnih tehnologija.

Stup bilo kojeg standarda sačinjavaju [42]:
Komunikacija: normira se zajednička terminologija koja organizacijama olakšava komunikaciju i suradnju.

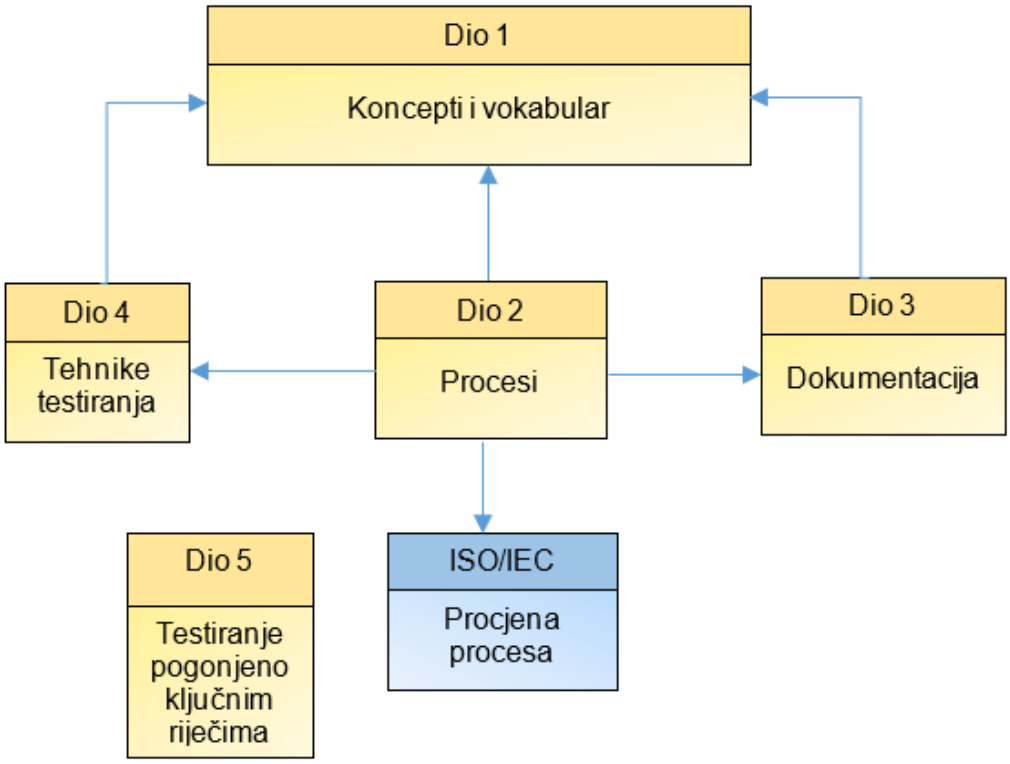
Certifikacija/sheme usklađenosti: certifikacija je postupak potvrđivanja da je određeni proizvod zadovoljio testove kontrole kvalitete i performansi. Standard koji regulira certifikaciju je ISO/IEC 17065.

Smjernice za vrednovanje (eng. benchmarking) primjera dobre prakse: Mjere se vrijeme, kvaliteta i cijena.

Važnost interoperabilnosti i dosljednosti: interoperabilnost označava temeljnu sposobnost sustava za lako povezivanje i međusobnu komunikaciju, čak i kada se radi o sustavima koje su razvile potpuno različiti proizvođači, iz različitih zemalja. S druge strane, dosljednost označava proizvod koji je dosljedan u raznim okruženjima.

Za testiranje softvera danas se koristi međunarodni standard ISO/IEC/IEEE 29119. Primjenom pojedinog standarda proizvod je ujednačen s ostalim takvim proizvodima proizvedenim bilo gdje u svijetu, u skladu s istim standardom. Ukoliko se radi o testiranju softvera, usuglašenost s međunarodnim standardom jamči da je on testiran pod jednakim uvjetima. ISO/IEC 29119 je standard za testiranje softvera i odnosi se na sve faze razvojnog ciklusa softvera koju provodi bilo koja organizacija. U ovoj seriji nalazi se pet standarada (Slika 15), a to su:

1. ISO/IEC 29119 -1:2013 Koncepti i definicije,
2. ISO/IEC 29119-2:2013 Postupci testiranja,
3. ISO/IEC 29119-3:2013 Dokumentacija o testiranju,
4. ISO/IEC 29119-4:2015 Tehnike testiranja, te
5. ISO/IEC 29119-5:2016 Testiranje pogonjeno ključnim riječima



Slika 15. Struktura standarda ISO/IEC 29119 [43]

ISO/IEC 33063:2015 - Standard propisuje model procjene procesa za testiranje softvera donesen u svrhu podrške standarda ISO/IEC 29119-2:2013).

ISO/IEC 20246:2017 - Propisuje standarde za provođenje statičkog testiranja (ISO/IEC 29119-5:2016) i bavi se rezultatima pregleda radnog materijala.

ISO/IEC 9126 - Da bi se utvrdila kvaliteta softvera, ovaj standard se bavi modelom kvalitete te vanjskom i unutarnjom metrikom. Opisuje funkcionalnost, pouzdanost, iskoristivost, učinkovitost, održivost i prenosivost.

ISO/IEC 25000:2005 - Pruža smjernice za zahtjeve za kvalitetan softver i njegovu evaluaciju (engl. *Software Quality Requirements and Evaluation, SQuaRE*). Sastoji se od pod-dijelova:

ISO 2500n – Upravljanje kvalitetom

ISO 2501n – Model kvalitete

ISO 2502n – Mjerljivost kvalitete

ISO 2503n – Preduvjeti za kvalitetu

ISO 2504n – Evaluacija kvalitete

ISO/IEC 12119 - Standard se odnosi na paket softvera isporučen kupcu, odnosno zahtjevima paketa i uputama za testiranje isporučenih paketa u odnosu na zahtjeve.

Tablica 2. Standardi testiranja softvera

Standard	Opis
IEEE 829	Standard za format dokumenata u raznim fazama testiranja softvera.
IEEE 1061	Metodologija utvrđivanja zahtjeva za kvalitetu, identifikaciju, primjenu, analizu i validaciju procesa te rezultate mjerljivosti kvalitete.
IEEE 1059	Vodič za verifikaciju softvera i planove validacije.
IEEE 1008	Standard za jedinično testiranje.
IEEE 1012	Standard za verifikaciju i validaciju softvera.
IEEE 1028	Standard za inspekcije softvera.
IEEE 1044	Standard za klasifikaciju anomalija.
IEEE 1044-1	Vodič kroz klasifikaciju anomalija.
IEEE 830	Vodič kroz razvoj specifikaciju zahtjeva sustava.
IEEE 730	Standard za planove osiguravanja kvalitete.
IEEE 1061	Standard za metriku i metodologiju kvalitete softvera.
IEEE 12207	Standard za procese i podatke životnog ciklusa softvera.
BS 7925-1	Rječnik nazivlja korištenog u testiranju softvera.
BS 7925-2	Standard za testiranje komponenti softvera.

Osim prethodno opisanih, postoje još i mnogi drugi standardi koji se odnose na testiranje softvera, a neke od njih prikazuje i **Error! Reference source not found.** [44].

3. Dokumentacija

3.1. Tipovi dokumentacije

Dokumentacija se može podijeliti u dvije glavne skupine: dokumentaciju proizvoda, koja opisuje nastali proizvod, i dokumentaciju procesa. Dokumentacija proizvoda opisuje proizvod i daje upute o rukovanju, a može se podijeliti u dvije podskupine: sistemska i korisnička dokumentacija [45].

3.1.1. Sistemska dokumentacija

Sistemsku dokumentaciju čine dokumenti koji opisuju sustav i njegove dijelove te pojašnjava temeljnu tehnologiju. Sastoji se od dokumenta zahtjeva, opisa dizajna, arhitekture i koda, validacijskih dokumenata, informacija o verifikaciji i testiranju i vodiča za pomoć.

Dokument zahtjeva pruža informacije o funkcionalnosti sustava, odnosno o tome što proizvod treba činiti. U njemu se nalaze poslovna pravila, korisničke priče, korisnički slučajevi itd. Dokument treba sažeto i jasno opisati svrhu proizvoda, njegove značajke, funkcionalnosti i način ponašanja. Prilikom sastavljanja dokumenta zahtjeva važno je uzeti u obzir:

Uloge i zaduženja – Na početku dokumenta potrebno je navesti osobe koje sudjeluju u provedbi projekta, a to su vlasnik, članovi tima i dionici, Na taj način svaki član tima upoznat je sa svojim zaduženjima i ciljevima.

Ciljevi tima i poslovni cilj – Kratak i jasan opis najvažnijih ciljeva.

Pozadina i strateško uklapanje – Sadrži kratko pojašnjenje strateškog cilja i razlog zašto se razvija proizvod. Kako strateški cilj svojim djelovanjem utječe na razvoj proizvoda i kako je ono usklađeno s ciljevima tvrtke.

Pretpostavke - Izrada popisa tehničkih i poslovnih pretpostavki.

Korisničke priče – Pisani su dokumenti iz gledišta korisnika softvera, koji su potrebni za projekt. To je kratki opis postupaka korisnika i rezultata koji se žele postići.

Korisnička interakcija i dizajn - Na web stranicu s dokumentacijom potrebno je navesti razradu dizajna.

Pitanja - Potrebno je dokumentirati sva pitanja s kojima se susreće tim prilikom razvoja proizvoda. Također, izraditi i popis poslova na kojima se trenutno ne radi ali se planira uskoro. To će olakšati organiziranje timskog rada i postavljanje prioriteta.

Sve navedeno bit će lakše razumljivo ako se koriste linkovi i sidra, jer će olakšati navigaciju po dokumentu, te grafički prikazi, jer olakšavaju usvajanje informacija.

Dokumentacija arhitekture softvera navodi samo najvažnije odluke u arhitekturi. Sadrži:

Predložak dizajna dokumenta: potrebno je postići dogovor oko sadržaja dokumenta o dizajnu arhitekture, prije njegove izrade i koristiti zadani predložak za ucrtavanje arhitekturnih rješenja.

Načela arhitekture i dizajna: potrebno je naglasiti temeljnu arhitekturu i principe dizajna koji će se koristiti tijekom izrade proizvoda.

Opis korisničkih priča: opisom se povezuju korisničke priče i srodni poslovni procesi, bez navođenje tehničkih detalja.

Detalji rješenja: potrebno je opisati rješenje koja su razmotrena prilikom navođenja planiranih usluga, modula, komponenti i njihove važnosti.

Predstavljanje rješenja pomoću dijagrama: potrebno je odabrati dijagrame koji će olakšati razumijevanje strukture i načela dizajna.

Dokument o kodu opisuje način rada koda, te je prvenstveno namijenjen developerima. Treba sadržavati programske okvire koji su korišteni u proizvodu, način povezivanja podataka, uzorke dizajna s primjerima, sigurnosne mjere, druga načela i ostale sadržaje koji se smatraju važnima.

Najučestalije vrste dokumentacije o testiranju su strategija, plan, specifikacija slučajeva i unaprijed definirana lista (popis) testova. Strategija testiranja opisuje pristup testiranju. Daje informacije o strukturi, potrebnim resursima i prioritetima prilikom testiranja. Obično je statička jer se odnosi na cjelokupni razvojni raspon. Plan testiranja opisuje što se treba testirati u kojem trenutku. Treba sadržavati popis stavki za testiranje, metode testiranja, vremenske okvire, uloge i zaduženja, dok specifikacije testnih slučajeva popisuju kojim postupcima se verificira svaka od značajki. Konačno, unaprijed definirana lista testova predstavlja popis testova koji se trebaju provesti u određeno vrijeme. Govori koja su testiranja dovršena, a koja su neuspjela.

3.1.2. Korisnička dokumentacija

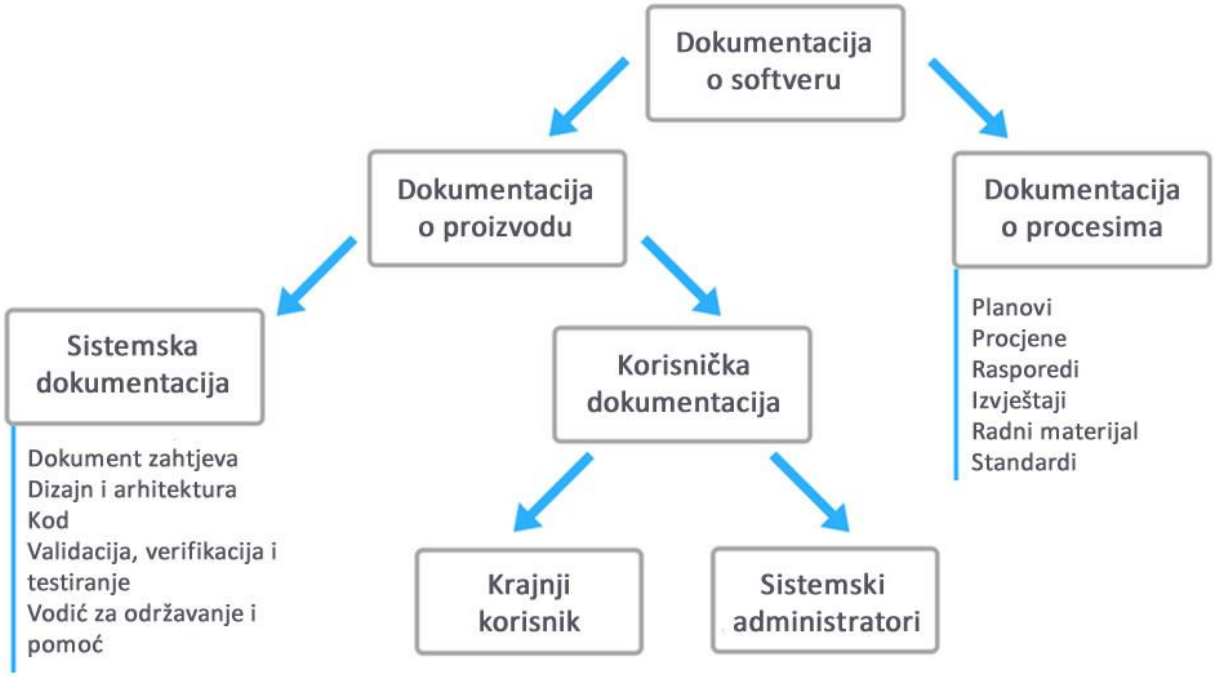
Korisničku dokumentaciju čine upute o radu namijenjene krajnjim korisnicima i sistemskim administratorima pa treba biti prilagođena razini vještine korisnika. Dokumentacija za krajnje korisnike sažeto opisuje kako proizvod može riješiti njihove poteškoće. Kada se radi o online dokumentaciji, ona treba sadržavati učestala pitanja, video upute, ugniježdenu pomoć, portale za pomoć itd., a kako bi se alati za pomoć ažurirali i unapređivali, potrebno je neprekidno prikupljati povratne informacije krajnjih korisnika.

Primjerice, dokumentacija procesa odnosi se na sve aktivnosti vezane uz razvoj proizvoda. Sastoji se od projektnih planova, rasporeda testiranja, izvještaja, standarda, bilješki sa sastanaka, pa čak i poslovne korespondencije. Ona iziskuje planiranje i papirologiju tijekom i nakon razvoja proizvoda, a omogućava bolju organizaciju i planiranje procesa. Najčešće vrste su:

Planovi, procjene i rasporedi - Obično nastaju prije početka razvoja proizvoda, a mogu se prilagođavati tijekom izrade softvera.

Izvještaji i metrika - Izvještaji prate korištenje vremena i kadra tijekom razvoja softvera. Mogu se izrađivati na dnevnoj, tjednoj i mjesečnoj bazi.

Uvijek treba napisati odgovarajuću količinu dokumentacije. Ona ne smije biti štura jer će doći do grešaka i smanjene učinkovitosti tijekom cjelokupnog postupka razvoja softvera. Ukoliko je dokumentacija preopširna, doći će do ponavljanja. Dokumentacija nikad nije konačna, već se uvijek nadopunjuje.



Slika 16. Vrste dokumentacije **[45]**

Dokumentacija je rezultat zajedničkog napora svih članova tima. Svi koji su na bilo koji način sudjelovali u razvoju proizvoda svojim će komentarima i idejama doprinijeti sastavljanju dokumentacije. Preporučljivo je unajmiti profesionalnog sastavljača dokumentacije koji ima tehničko znanje potrebno za razumijevanje procesa i iskustvo u ovom poslu. Poželjno je da ta osoba bude smještena u uredu gdje se procesi odvijaju, kako bi postao dio tima i tako što kvalitetnije izvršio svoj zadatak. Kvalitetno sastavljena dokumentacija je precizna, jasna i relevantna. Nije potrebno izraditi sve vrste dokumentacije (Slika 16), već samo onu kojom će se to postići.

3.2. Generatori dokumentacije

Kako bi se olakšalo pisanje dokumentacije na tržištu su dostupni razni generatori dokumentacije namijenjeni programerima i krajnjim korisnicima. Primjerice, Doxygen je 1997. godine je u programskom jeziku C++ razvio Dimitri van Heesch. Naziv je nastao od riječi doc(s) – dox i gen(erator). Doxygen se koristi prvenstveno za rad s programskim jezikom C++, ali podržava i jezike poput C, Objective-C, C#, PHP, Java, Python, IDL, Fortran, VHDL, Tcl i donekle i D. Radi na operativnim sustavima Windows, iOs i na većini Lynuxa. Dokumentaciju generira iz komentara unutar koda.

Doxygen se može koristiti u tri svrhe:

1. Njime se generira online dokumentacijski preglednik u HTML-u, kao i referentne upute za korištenje u Latexu. Postoji i podrška za generiranje dokumentacije u RTF formatu, PostScript, PDF s hipervezama, komprimirani HTML itd.
2. Doxygen se može konfigurirati za izvlačenje strukture koda iz nedokumentiranih izvorišnih datoteka. On može i vizualno prikazati odnose među raznim elementima putem grafova koji prikazuju ovisnosti, dijagrama nasljeđivanja i dijagrama suradnji koji se generiraju automatski.
3. Doxygen se može koristiti i za generiranje uobičajene dokumentacije. [46]

Primjer 1. Doxygen - dokumentiranje klase ExampleClass [47]

```
class ExampleClass{
public:
    /// Create an ExampleClass
    ExampleClass();
    /// Create an ExampleClass with lot's of intial values
    ExampleClass(int a, float b);
    /// This method does something
    void DoSomething();

    /** This is a method that does so
      * much that I must write an epic
      * novel just to describe how much
      * it truly does. */
    void DoNothing();

    /** \brief A useful method.
      * \param level an integer setting how useful to be
      * \return Output that is extra useful
      *
      * This method does unbelievably useful things.
      * And returns exceptionally useful results.
      * Use it everyday with good health.
      */
    void* VeryUsefulMethod(bool level);
private:
    const char* fQuestion; ///< the question
    int fAnswer;           ///< the answer
```

```
}; // end of class ExampleClass
```

Primjer 1 prikazuje kako je potrebno komentirati C++ izvorni kod da bi naknadno iz njega mogli automatski generirati željenu dokumentaciju.

JavaDoc također generira dokumentaciju na temelju komentara unutar koda, ali za programski jezik Java. Dokumentacija se generira u formatu HTML. Dvije su vrste komentara u JavaDocu: komentari na razini klase i komentari na razini članova (atributa i metoda). [48] Komentari na razini klase pružaju opis klase, a najčešće još definiraju i autora. Smješteni su točno iznad koda koji deklarira klasu, a izgledaju na sljedeći način (Primjer 2).

Primjer 2. JavaDoc - dokumentiranja izvornog Java koda

```
/**
* This is a class comment describing the ExampleClass class.
*
* @author  FirstName SecondName
* @version 1.0
* @since   Date
*/
Class ExampleClass{
      public static void main(String[] args) {
            // ...
         }
}
```

Komentari na razini članova opisuju polja i metode. Komentari metoda mogu sadržavati tagove kojima se opisuje parametre i povratnu vrijednost (Primjer 3).

Primjer 3. JavaDoc - komentari na razini članova klase

```
public class Skladiste {

   /**
    * Količina mlijeka
    */
   private int kolMlijeka;
   /**
```

```
     * Definira količinu mlijeka u skladištu
     *
     * @param int nova količina mlijeka (lit)
     */
    public void setKolMlijeka(int novaKolicina) {
       this.kolMlijeka = novaKolicina;
    }
}
```

Neki od češće korištenih tagova u JavaDocu su:

Tag	Opis	Sintaksa
@author	Autor dokumenta	@author <ime>
@param	Parametar metode	@param <ime>
@return	Tip povratne vrijednosti	@return <opis>
@throws @exception	Ovo su sinonimi, Iznimke koje se mogu „baciti“ unutar funkcije	@throws <tip> @exception <tip>
@version	Inačica	@version <inačica>

Tri su tipa komentara u JavaDocu:

1. /* tekst */
   Kompajler ignorira tekst između ovih znakova.
2. //tekst
   Kompajler ignorira sve iza kosih crta do kraja retka.
3. /** dokumentacija */
   Između znakova se nalazi dokumentacija i koristi se za automatsko generiranje komentara iz više redaka (za razliku od //tekst). Primjerice,

Ovisno o korištenoj platformi, programskom jeziku i razvojnom okruženju, moguće je koristiti i druga programska rješenja za generiranje dokumentacije koda. Neka od njih su i Sphinx, Dr. Explain, PHPDocumentator itd.

4. Uvod u alate Enterprise Architecture

U prethodna tri poglavlja obrađena je problematika programskog inženjerstva i projektiranja informacijskih sustava. Ovo poglavlje donosi prikaz praktične primjene alata Enterprise Architect-a, koji omogućava pojedincima i timovima stvaranje različitih dijagrama koji olakšavaju analizu, strategiju, projektiranje i planiranje. Uporabom jednostavnih dijagrama, informacije se pojednostavljuju, te ih može razumjeti svaki korisnik. Aplikacija je do sada 15 godina kontinuirano razvijana i poboljšavana, kako bi mogla zadovoljiti potrebe programera, poslovnih analitičara, arhitekata, testera, voditelja projekata, dizajnera i drugih.

Enterprise Architect je sofisticirana i fleksibilna platforma Enterprise Architecture, koja se može koristiti i kao repozitorij arhitekture i kao alat za upravljanje arhitektonskim projektima. Može se koristiti tokom cijelog životnog ciklusa postavljanja arhitektonskog programa ili ureda za planiranje, upravljanje, razvoj i dokumentiranje arhitekture do upravljanja projektima provedbe, koji koriste arhitektonske rezultate. Alat se može koristiti s bilo kojim pojedinačnim ili kombinacijom arhitektonskih okvira, procesa i jezika predstavljanja. Postoji širok raspon sadržaja i alata koji omogućuju arhitektu da radi, koristeći svoje preferirane metode, kao što su WordProcessor, Spreadsheet prikazi, dijagrami, matrice odnosa ili niz drugih jezgri i proširenih značajki.

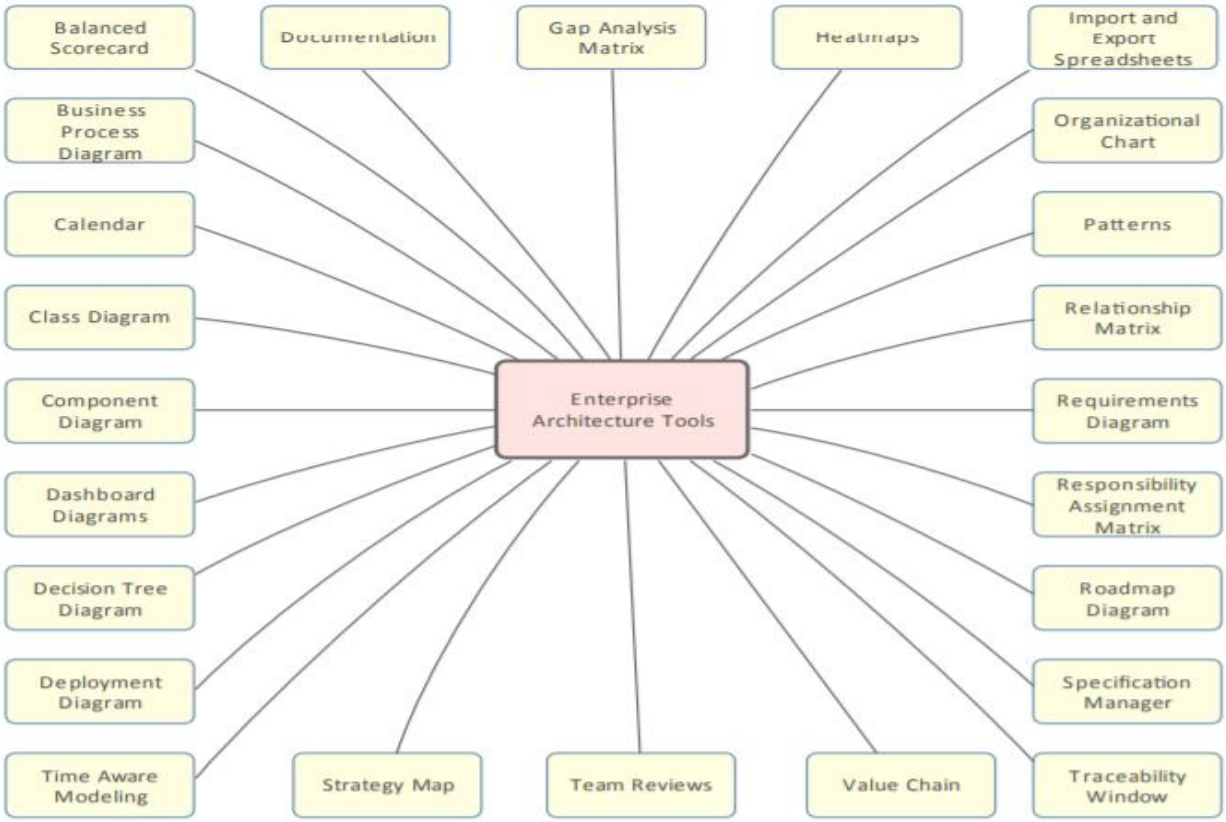
Kroz cijelu poznatu povijest ljudskog roda, opstanak i prosperitet pojedinaca i društvenih zajednica, između ostaloga, uvelike je ovisio i o mogućnosti razmjene informacija između udaljenih skupina ljudi. Brzom i pravodobnom razmjenom informacija, odnosno dostavom podataka, omogućeno je i olakšano koordiniranje aktivnosti u korist cjelokupne zajednice. Nastankom Interneta, napredak u komunikaciji je krenuo nevjerojatnom brzinom. Nove tehnologije donijele su, brzinom svjetlosti, novosti u hardveru i softveru. Svaka aplikacija donosi razne novosti, nova sučelja, nove korisne dodatke i mogućnosti i to sve u cilju olakšanja i ubrzanja rada korisnika. Jedan od tih produkata je i Enterprise Architect.

Enterprise arhitektura (EA) se kao pojam pojavljuje već u kasnim osamdesetima, međutim ni danas nema jasnu definiciju, niti jasne granice. Iz povijesnih razloga, treba spomenuti Zachmanov okvir za EA. Naime, 1987. John A. Zachman je u IBM-ovoj publikaciji “IBM Systems Journal“ objavio članak Okvir za

arhitekturu informacijskih sustava (A Framework for Information Systems Architecture) i tako postavio temelje Enterprise arhitekture u formi kako je danas gledamo. Prihvaćanje Enterprise arhitekture, u prvom redu zahtijeva organizacijske promjene, i to ne samo osnivanje odjela za EA, nego i ispravno povezivanje poslovnih sektora, preko EA, do informatike. Logično je zaključiti da uvođenje EA nije jednostavan zadatak i da će zahtijevati ozbiljno promišljanje, razradu i kvalitetnu implementaciju.

Slika 17 prikazuje ključne alate Enterprise Architecture koji se mogu koristiti za planiranje, stvaranje, upravljanje i dokumentiranje arhitekture. Iako su to primarni alati, postoji niz drugih alata opisanih u poglavlju Dodatni alati za Enterprise arhitekture.

Izvor: Sparx Systems

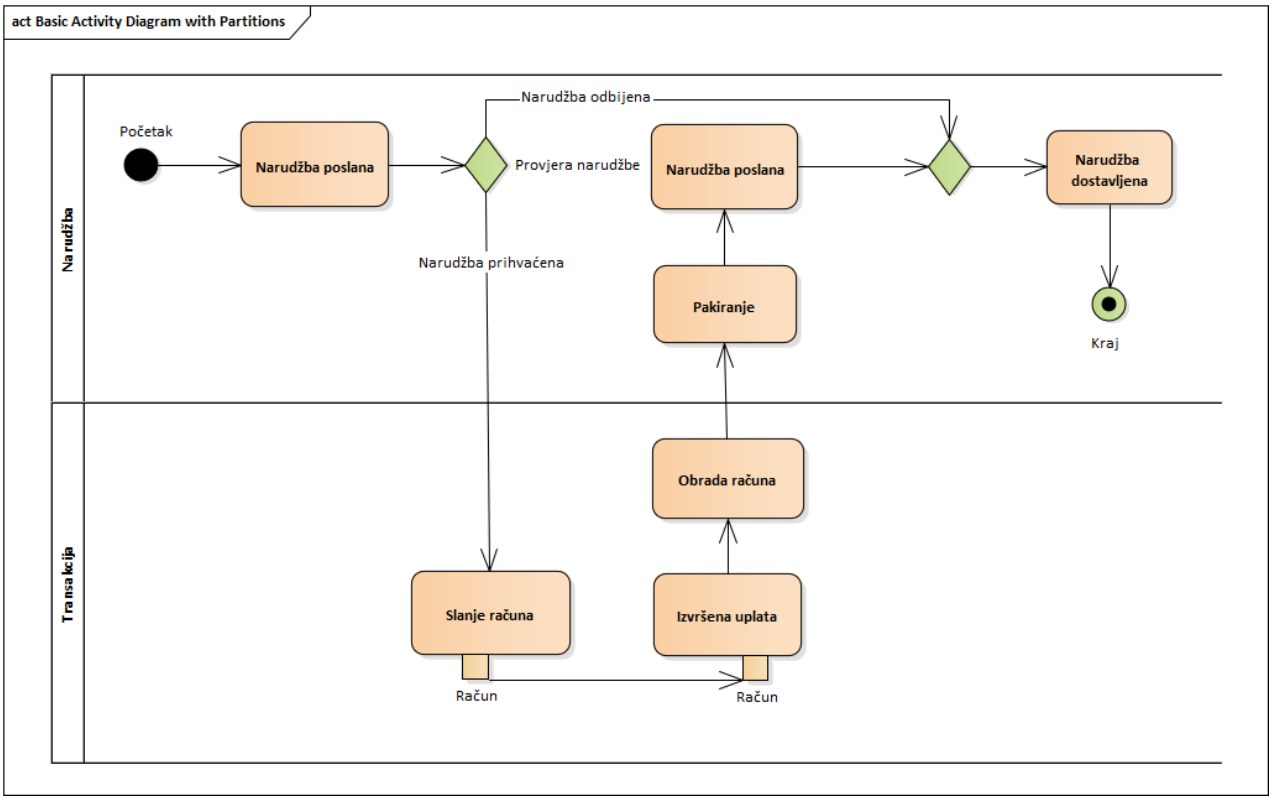


Slika 17. Mind mapa

5. Rad s dijagramima – Dijagram aktivnosti

Dijagrami aktivnosti koriste se za modeliranje ponašanja sustava i pokazuju način na koji su ta ponašanja povezana u ukupnom toku sustava (to jest, interakcije dinamičkih elemenata). Logički putovi koje slijedi proces, temeljeni na različitim uvjetima, istodobnoj obradi, pristupu podacima, prekidima i drugim razlikovanjima logičkog puta, koriste se za konstruiranje procesa, sustava ili postupka. Elementi dijagrama aktivnosti i priključci, generiraju se sa stranica "Aktivnost" u **alatu Diagram Toolbox** .

Dijagram aktivnosti je jedan od Unified Modeling Language (UML) Bihevioralnih dijagrama koji se mogu koristiti za modeliranje procesa ili algoritma, kao niz koraka. To je sofisticiranija verzija njezina bliskog rođaka u dijagramu toka. Dijagrami aktivnosti mogu se koristiti za modeliranje poslovnih procesa, kao UML alternativu dijagramu BPMN poslovnih procesa i imaju istu sposobnost stvaranja hijerarhije aktivnosti u pregledniku projekata.



Slika 18. Primjer dijagrama aktivnosti

Dijagrami aktivnosti prikazuju proceduralnu logiku, poslovni proces i radni tok aktivnosti koje se obavljaju u sustavu korak po korak. Naglasak kod ovih dijagrama jest na jednostavnosti i poslovnim operacijama koje se odvijaju slijedno, jedna za drugom.
Dijagram aktivnosti koristi se tako da daje pogled na sustav sa više razine. Prilagođena je više za poslovne korisnike, a manje za tehničke korisnike. Fokus je na razumijevanju poslovne logike i poslovnih procesa, a ne na implementaciji detalja.

5.1. Kako do dijagrama aktivnosti?

Dijagrami aktivnosti prikazuju funkcionalnost softverskog sustava iz perspektive unutrašnjosti sustava. Pritom, ovi dijagrami ne prikazuju niti sudionike niti vanjsko sučelje prema krajnjim korisnicima. Budući da razrađuju ponašanje sustava u smislu aktivnosti i prijelaza između stanja, svrstavaju se u dinamičke dijagrame.

Kako do dijagrama aktivnosti	Vrpca: Dizajn> Dijagram> Umetni> UML ponašanje> Alatna traka preglednika aktivnosti projekta: Nova ikona dijagrama> UML ponašanje> Kontekstni izbornik preglednika aktivnosti projekta | Dodaj dijagram ...> UML ponašanje> Aktivnost

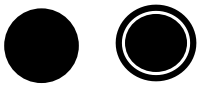
Dijagrami aktivnosti prikazuju funkcionalnost softverskog sustava iz perspektive unutrašnjosti sustava. Pritom, ovi dijagrami ne prikazuju niti sudionike niti vanjsko sučelje prema krajnjim korisnicima. Budući da razrađuju ponašanje sustava u smislu aktivnosti i prijelaza između stanja, svrstavaju se u dinamičke dijagrame.

Dijagrami aktivnosti prikazuju proceduralnu logiku, poslovni proces i radni tok aktivnosti koje se obavljaju u sustavu, korak po korak. Naglasak kod ovih dijagrama jest na jednostavnosti i poslovnim operacijama koje se odvijaju slijedno, jedna za drugom.

5.2. Temeljni elementi

Temeljni elementi dijagrama aktivnosti, koje EA koristi su:

1. Početno i konačno stanje



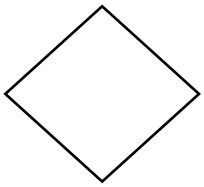
2. Simbol aktivnosti, obilježava se zaobljenim pravokutnikom. Aktivnosti mogu biti slijedne ili paralelne.



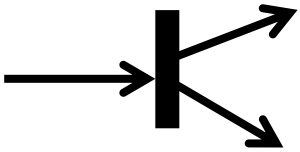
3. Prijelaz (tok) između aktivnosti, označava se strelicom.

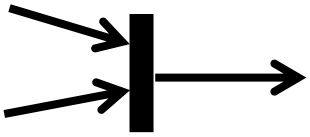


4. Simbol odluke ili uvjetnog grananja, označena dijamantnim oblikom.

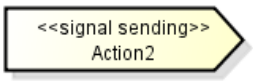


5. Račvanje i skupljanje, označava se crnom i podebljanom linijom.





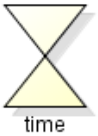
6. Signal (događaj) pokazuje da na neku aktivnost djeluju određeni događaji iz vanjskog procesa. Takva aktivnost konstantno „sluša“ signale, a dijagram definira kako aktivnost reagira.
   a. Šaljući signal



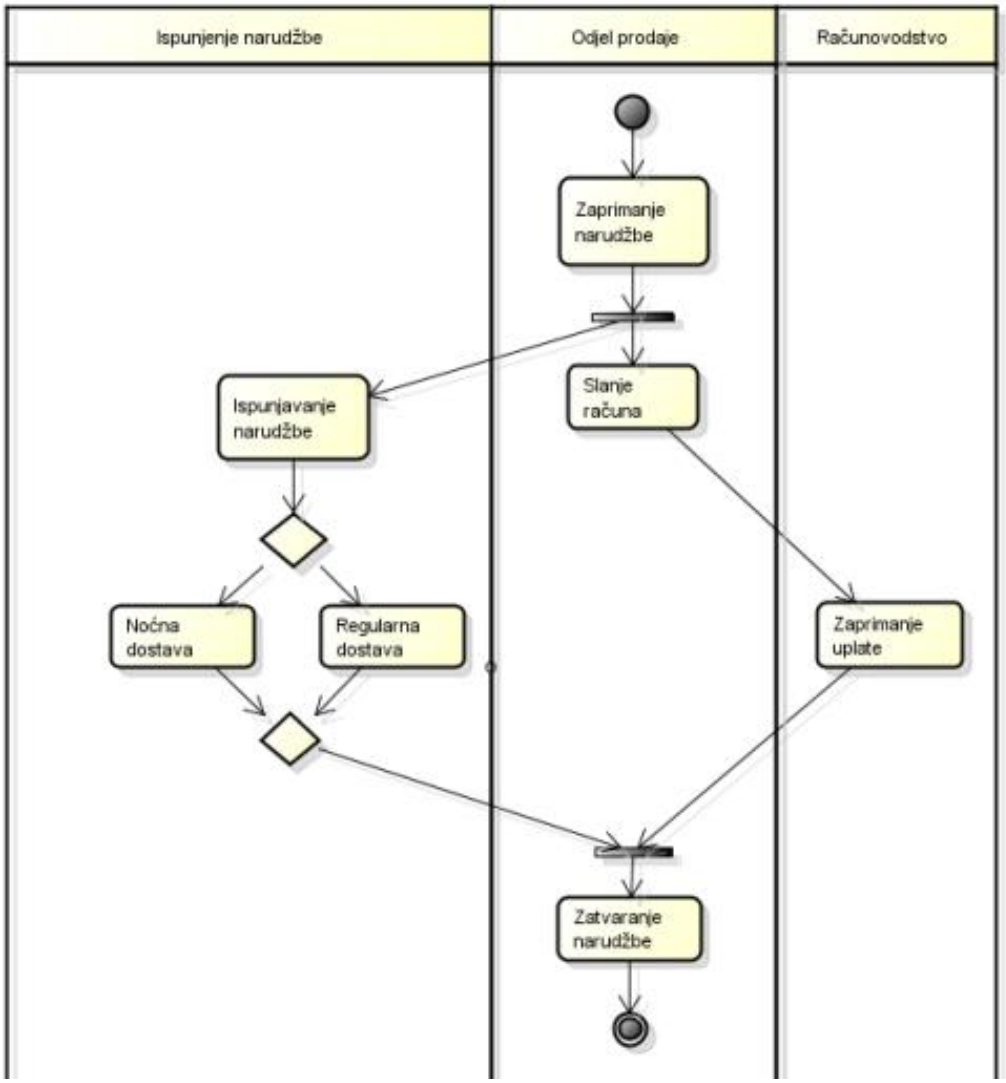
   b. Primajući signal



   c. Vremenski signal



7. Plivajuće staze, kod kojih su aktivnosti raspoređene u uspravne ili vodoravne u dijagramu. Izuzetak je organizacijska, kada se želi istaknuti koji od dijelova sustava što izvodi. Aktivnosti u ovom slučaju pripadaju stazama, a tokovi podataka mogu prelaziti iz jedne staze u drugu.

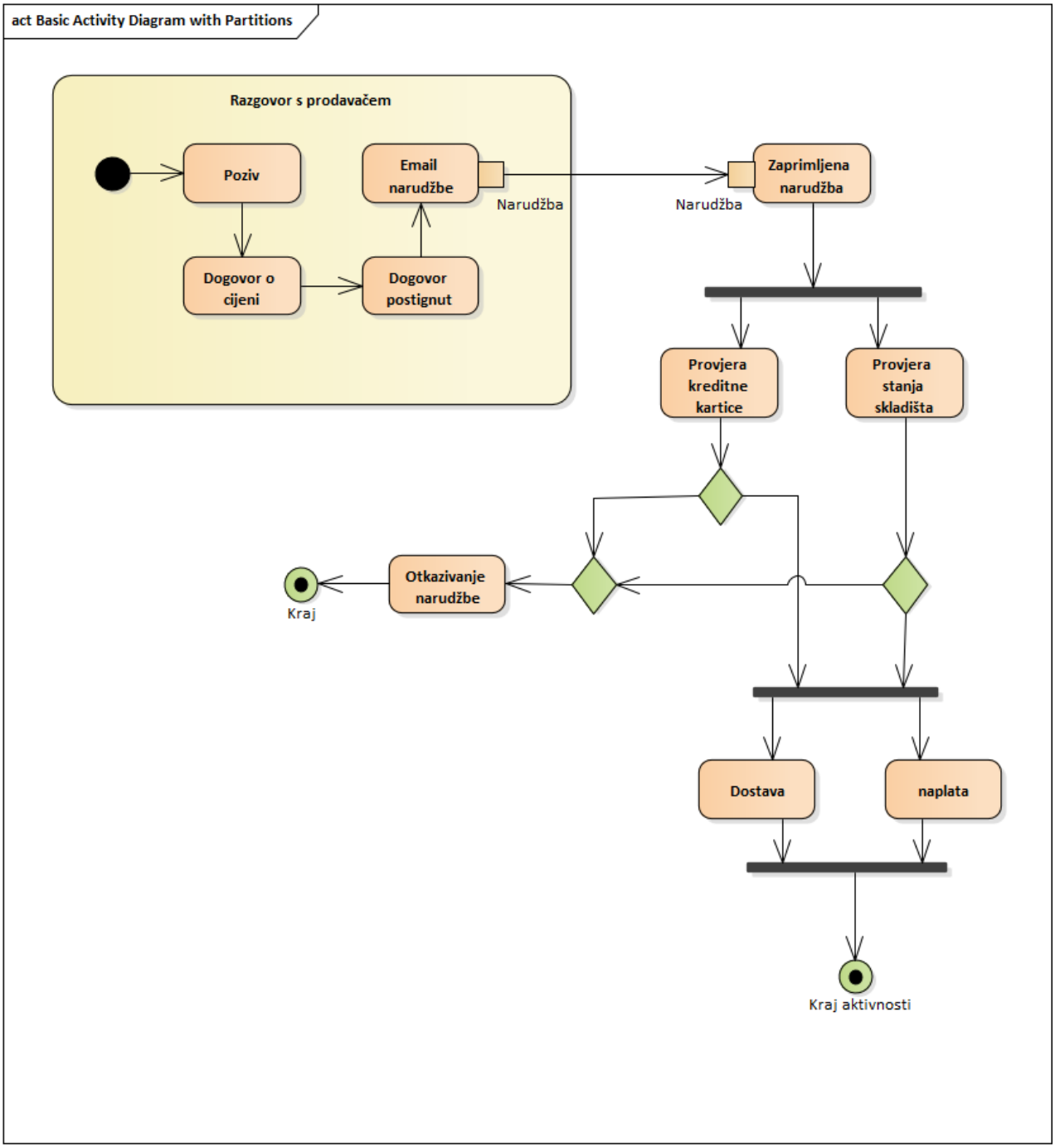


Primjer:

Odjel prodaje nekog poduzeća zaprima narudžbu kupca. Prvi korak je ispunjavanje narudžbenice i slanje računa kupcu. Prilikom ispunjavanja narudžbenice, kupac odabire način dostave koji može biti noćni (prioritetni) ili regularni. Nakon nekog vremena od slanja računa, poduzeće zaprima uplatu kupca. Zaprimanjem uplate i slanjem robe, narudžba se završava.

5.3. Primjer dijagrama aktivnosti

Ovaj dijagram prikazuje neke dijelove dijagrama aktivnosti, uključujući aktivnosti, akcije, početne čvorove, završne čvorove i točke odlučivanja.



Slika 19. Primjer dijagrama aktivnosti

5.4. Rad s dijagramima – Dijagram klasa

Dijagram klase bilježi logičku strukturu sustava - Klase - i stvari koje čine model. To je statički model, koji opisuje ono što postoji, kakve atribute i ponašanje ima, a ne kako se nešto radi. Na dijagramu klase mogu se ilustrirati odnosi između klasa i sučelja, pomoću generalizacija, agregatima i asocijacija, koje su vrijedne u odražavanju nasljeđivanja, sastava ili upotrebe, odnosno veza.

Dijagram klase je jedan od Unified Modeling Language (UML) strukturnih dijagrama koji se mogu koristiti za modeliranje širokog raspona stvari. To je dijagram opće namjene, za modeliranje entiteta u poslovnim i tehničkim domenama, uključujući pojmove i koncepte, poslovna pravila i mogućnosti u XML i shemama baze podataka.

1. prikazuje sustav pomoću klasa (razreda) i relacija među klasama
2. statički dijagram, što znači da ne uzima u obzir vremensku komponentu sustava, već dio ili cijeli sustav kakav postoji u nekom trenutku
3. pripada strukturnoj skupini UML dijagrama

Dozvoljene vrijednosti kod višestrukosti veza (multiplicity):

1	= točno 1 pojedinac
n_1	= bilo koji točno određen broj (0, 1, 5, 23)
n_1.. n_2	= između n_1 i n_2 (5..8 → 5, 6, 7 ili 8)
n_1..n	= između n_1 i više pojedinaca
n_1...n_2, n_3	= kombinacija (4..7,9 → 4, 5, 6, 7 ili 9)
n..*	= n ili više pojedinaca, neograničeno (5..* → 5 ili više)
0..* ili * ili n	= više pojedinaca, neograničeno

Ako višestrukost nije naznačena, smatra se vrijednost 1 (točno 1 pojedinac).

5.5. Kako do dijagrama klasa?

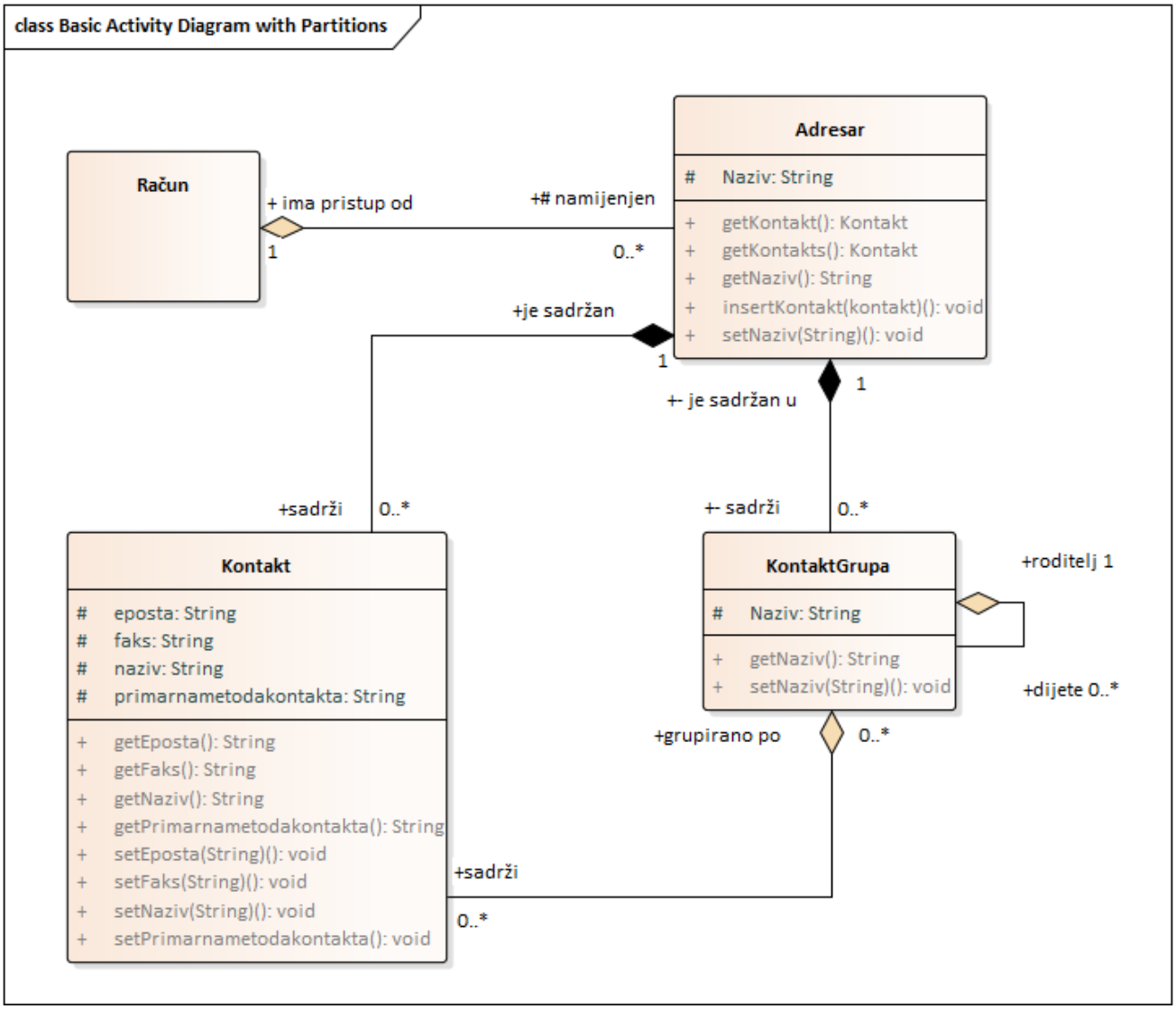
Kako do dijagrama klasa	Vrpca: Dizajn> Dijagram> Umetni> UML strukturna> Alatna traka preglednika projekta klase: Nova ikona dijagrama> UML strukturna> Kontekstni izbornik preglednika projekta klase | Dodaj dijagram ...> UML Strukturno> Klasa

5.6. Korištenje dijagrama klasa

Dijagram klase može se koristiti u svim slučajevima kada je potrebno logičko ili strukturalno prikazivanje sustava. Primjenjiv je za modeliranje poslovnih i tehničkih koncepata i može se koristiti za modeliranje informacija i struktura, kao što su XML i sheme baza podataka.

5.7. Primjer dijagrama klasa

U ovom primjeru dijagrama klasa postoje dva oblika odnosa agregacije: Blijedi obrazac označava da račun klase koristi Adresar, ali ne mora ga nužno sadržavati. Tamni obrazac složenog agregiranja označava vlasništvo ili ograničenje ciljnih klasa (na dijamantnom kraju) izvora Klasa.



Slika 20. Primjer dijagrama klasa

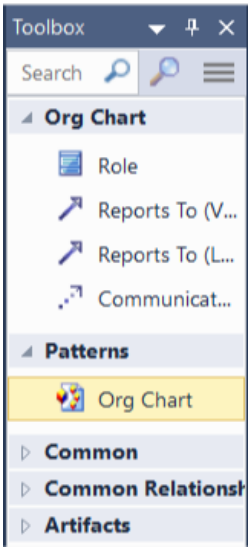
6. Rad s dijagramima – Organizacijska shema

Organizacijska shema (Org Chart) je grafički prikaz strukture organizacije, uz opisivanje uloge i odgovornosti svakog člana osoblja. Enterprise Architect pruža Org Chart Pattern, kako bi ubrzao stvaranje organizacijskih karata.

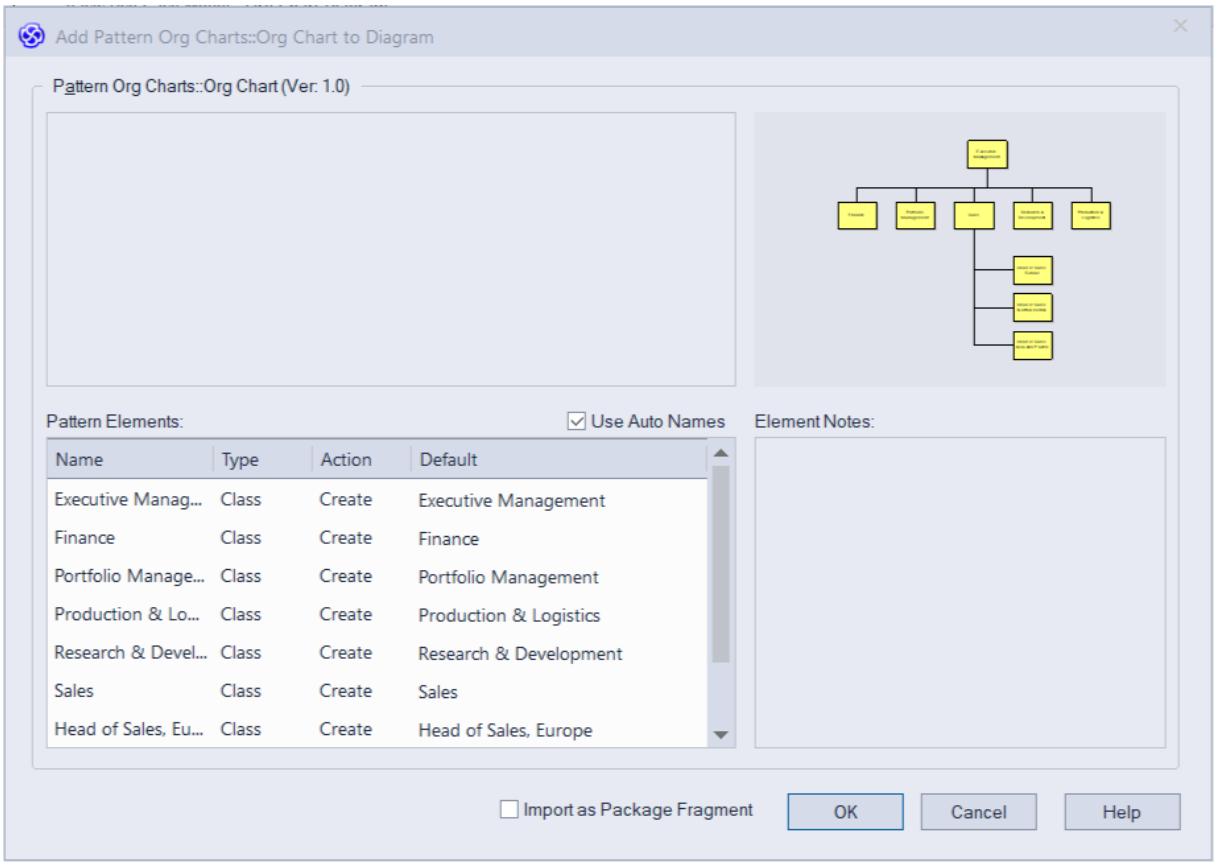
Prilikom modeliranja strukture organizacije, posve je uobičajeno koristiti organizacijsku shemu za opisivanje strukture, u smislu odnosa između osoblja ili odjela. Svaki pravokutnik predstavlja osobu, položaj ili odjel, dok svaki konektor predstavlja hijerarhijski odnos ili strukturu zapovijedanja. Enterprise Architect može pohraniti bilješke i povezani dokument za svaku poziciju, kako bi pružio više detalja od tradicionalnih Org grafikona.

6.1. Izrada organizacijskog grafikona

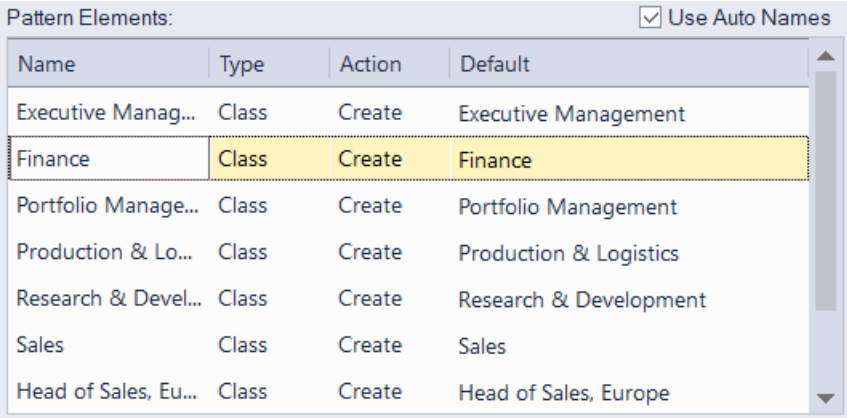
Ovaj primjer ilustrira koliko je lako modificirati osnovni uzorak, tako da odgovara individualnim potrebama. Stvara se novi paket pod nazivom "Org grafikon". Desnom tipkom miša treba kliknuti Paket, a zatim odabrati "Dodaj dijagram". Stvorena je nova organizacijsku tablicu pod nazivom 'Struktura tvrtke' i treba kliknuti gumb U redu . Iz Toolboxa odabrati uzorak 'Org Chart' i povući ga na dijagram. Treba izmijeniti elemente uzorka odabirom gumba za elipsu, pokraj svake uloge.



Slika 21. Org chart toolbox



Slika 22. Dodavanje org chart uzorka

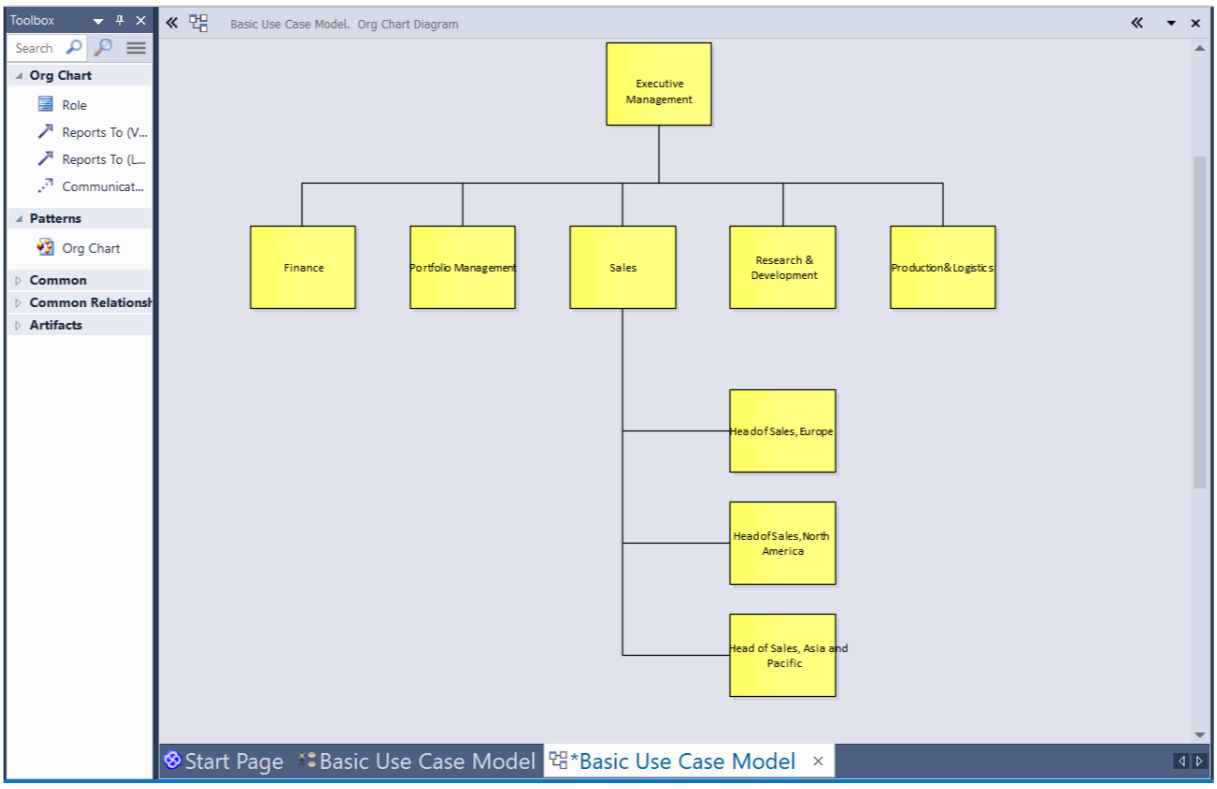


Slika 23. Odabir uzorka

6.2. Organizacijski dijagram – hijerarhijska struktura

Organizacijski dijagram je moćan alat koji predstavlja vizualni prikaz, opisuje uloge, odgovornosti i strukturu izvješćivanja organizacije. To je od neprocjenjive vrijednosti za Business Analyste jer omogućuje vizualizaciju i razumijevanje s kim trebaju komunicirati tijekom obavljanja raznih zadataka. Organizacijski grafikon u Enterprise Architect je fleksibilan i može se koristiti za stvaranje bilo koje vrste predstavljanja, uključujući korištenje slika, koje predstavljaju timove ili pojedince.

Organizacijska shema prikazuje unutarnju strukturu organizacije ili tvrtke. Zaposlenici i radna mjesta su predstavljeni kvadratima ili drugim oblicima, ponekad uključujući fotografije, kontaktne informacije, veze e-pošte i stranice, ikone i ilustracije.



Slika 24. Organizacijska shema unutarnje strukture tvrtke

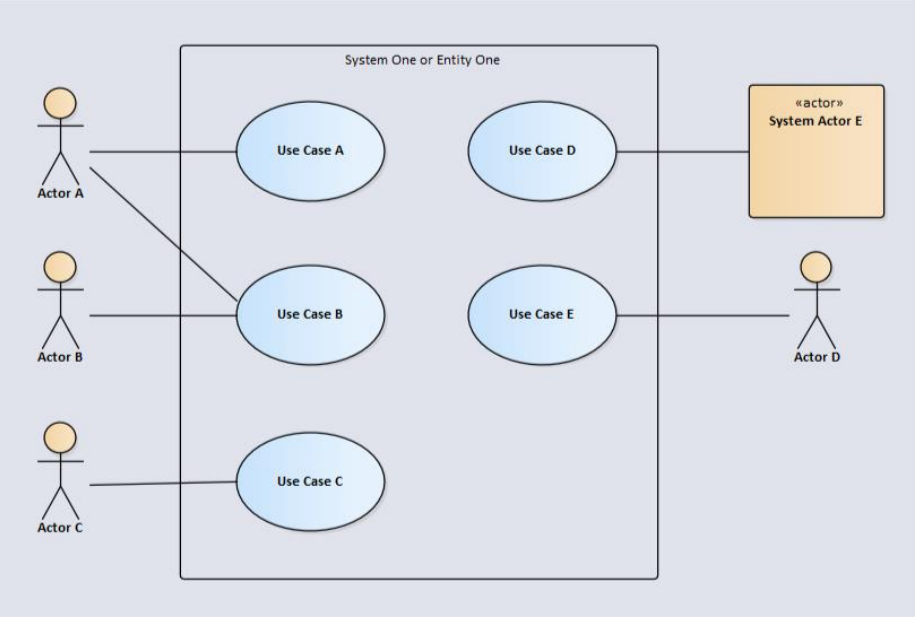
6.2.1. Kako do dijagrama?

Kako do dijagrama	Vrpca: Dizajn> Dijagram> Umetanje> Strateško modeliranje> Alatna traka preglednika projekata Org Chart: Nova ikona dijagrama> Strateško modeliranje> Kontekstni izbornik preglednika projekata Org Chart | Dodajte dijagram ...> Strateško modeliranje> Org Chart

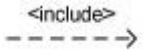
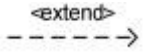
7. Rad s dijagramima – Dijagram slučajeva korištenja

Dijagrami slučajeva korištenja prikazuju ponašanje sustava, dijelova sustava ili konkretne klase, na način vidljiv korisniku sustava. Ponašanje sustava opisano je pomoću ciljeva i sudionika, koji predstavljaju apstrakciju korisnika sustava. Ovi dijagrami opisuju samo poglede na ponašanje sustava, sa strane korisnikove percepcije te ne opisuju kako je funkcionalnost izvedena unutar sustava.

Dijagrami slučajeva su statički UML dijagrami, pripadaju skupini dijagrama ponašanja jer modeliraju moguće ponašanje korisnika sustava. Kreiraju se u ranim fazama oblikovanja (najčešće kao prvi dijagram). Dijagrami slučajeva korištenja daju grubi prikaz **što** sustav radi, a ne **kako** radi.

Aspekt	Opis
Dijagram slučaja korištenja	Dijagram slučaja korištenja jedan je od Unified Modeling Language (UML) Bihevioralnih dijagrama koji se mogu koristiti za opisivanje ciljeva korisnika i drugih sustava koji su u interakciji sa sustavom koji se modelira. Koriste se za opisivanje funkcionalnih zahtjeva sustava, podsustava ili entiteta i predstavljaju jednostavnu, ali uvjerljivu sliku o tome kako će se sustav koristiti. 

<table>
<tr><td></td><td>Obično se koriste zajedno s poslovnim zahtjevima i zahtjevima sudionika na višoj razini i često se nadopunjuju nizom nefunkcionalnih zahtjeva.</td></tr>
<tr><td>Kako do dijagrama slučaja korištenja</td><td>Vrpca: Dizajn> Dijagram> Umetni> UML ponašanje ponašanja> Koristi slučaj
Alatna traka preglednika projekta : Nova ikona dijagrama> UML ponašanja> Koristi slučaj
Kontekstni izbornik preglednika projekata | Dodaj dijagram ...> UML ponašanja> Koristi slučaj</td></tr>
<tr><td>Korištenje dijagrama slučaja korištenja</td><td>Dijagram Use Case koristi se za opisivanje ciljeva koje korisnici ili drugi sustavi žele postići u interakciji sa sustavom. On uvijek opisuje cilj iz perspektive sudionika, a ako se dodaju detalji o slučaju korištenja, cilj će biti opisan većom preciznošću.
Slučajevi korištenja često će poslužiti kao osnova za definiranje ispitnih slučajeva.</td></tr>
<tr><td>Opcije za dijagram slučaja korištenja</td><td>Za predstavljanje različitih dijelova sustava ili paketa slučajeva korištenja, moguće je stvoriti bilo koji broj dijagrama Use Case. Dijagrami mogu biti jednostavni ili se mogu strukturirati primjenom više dodatnih konektora, kao što su Include, Extend i Generalization. Veza između sudionika i slučaja korištenja prikazuje se ravnom crtom, bez strelica i može biti između:
• sudionika i slučaja korištenja
• dva ili više slučaja korištenja
• dva ili više sudionika
<table><tr><td>Tip veze</td><td>Grafički prikaz</td><td>Opis</td></tr><tr><td>Pridruživanje (asocijacija)</td><td></td><td>Asocijacijom se povezuju sudionici sa slučajevima korištenja u kojima sudjeluju.</td></tr></table></td></tr>
</table>

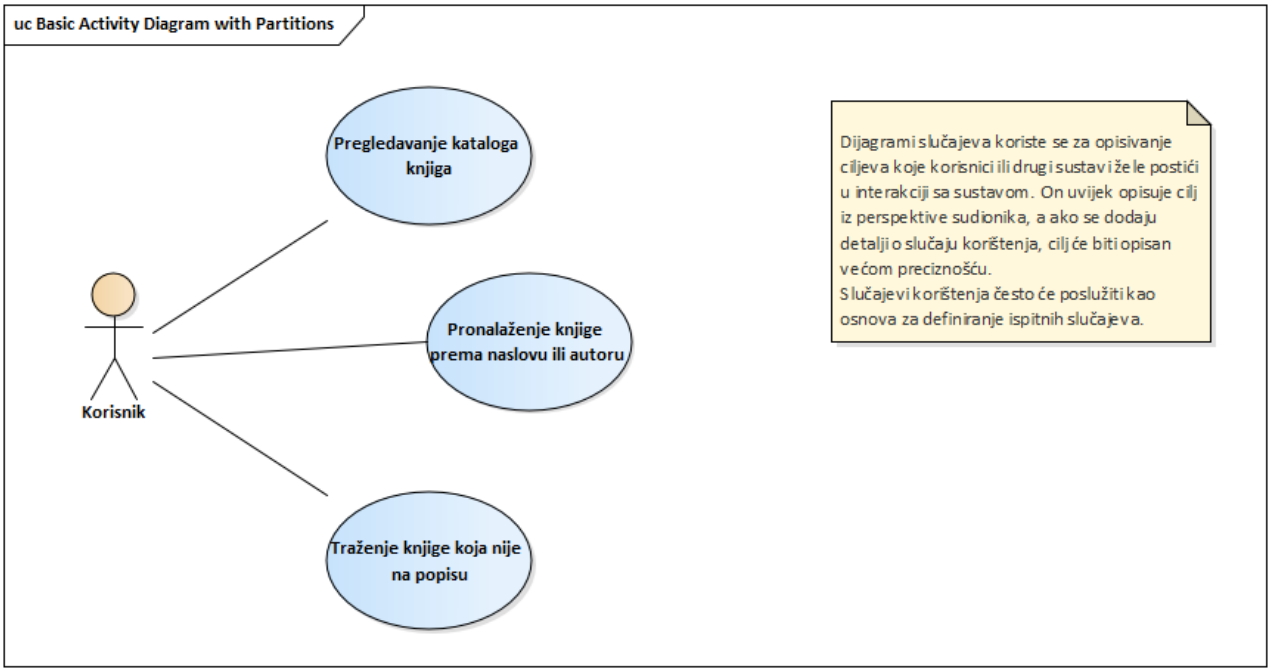
		Kod asocijacije, moguće je definirati njenu višestrukost
Generalizacija		Generalizacijom se povezuju dva sudionika ili dva slučaja korištenja. U slučaju da se generalizacijom povezuju dva sudionika, specifičniji sudionik preuzima sve uloge apstraktnijeg, uz dodatak novih uloga. Kod korištenja generalizacije između dva slučaja korištenja, specifičniji proširuje odnosno mijenja funkcionalnosti apstraktnijeg.
Uključivanje		Vezom uključivanja se povezuju dva slučaja korištenja na način da jedan slučaj u toku svog izvođenja u potpunosti izvede uključeni slučaj korištenja.
Proširenje		Vezom proširenja se povezuju dva slučaja korištenja, pri čemu jedan proširuje funkcionalnost drugog. Proširenje se ostvaruje ako je zadovoljen određeni uvjet, definiran u točki proširenja.

Može se uključiti granica sustava koja se koristi za imenovanje sustava, podsustava ili entiteta o kojem se raspravlja; Sudionici leže izvan granice i slučajeva korištenja iznutra.

	Dijagrami slučaja korištenja mogu se koristiti za prikaz načina na koji je slučaj upotrebe povezan s drugim elementima u sustavu, uključujući elemente za nadogradnju, kao što su zahtjevi i elementi down-stream, kao što su komponente. Dijagram Use Case (kao bilo koji dijagram) može se promatrati kao popis elemenata, što olakšava rad sa svojstvima elementa.

7.1. Primjer dijagrama

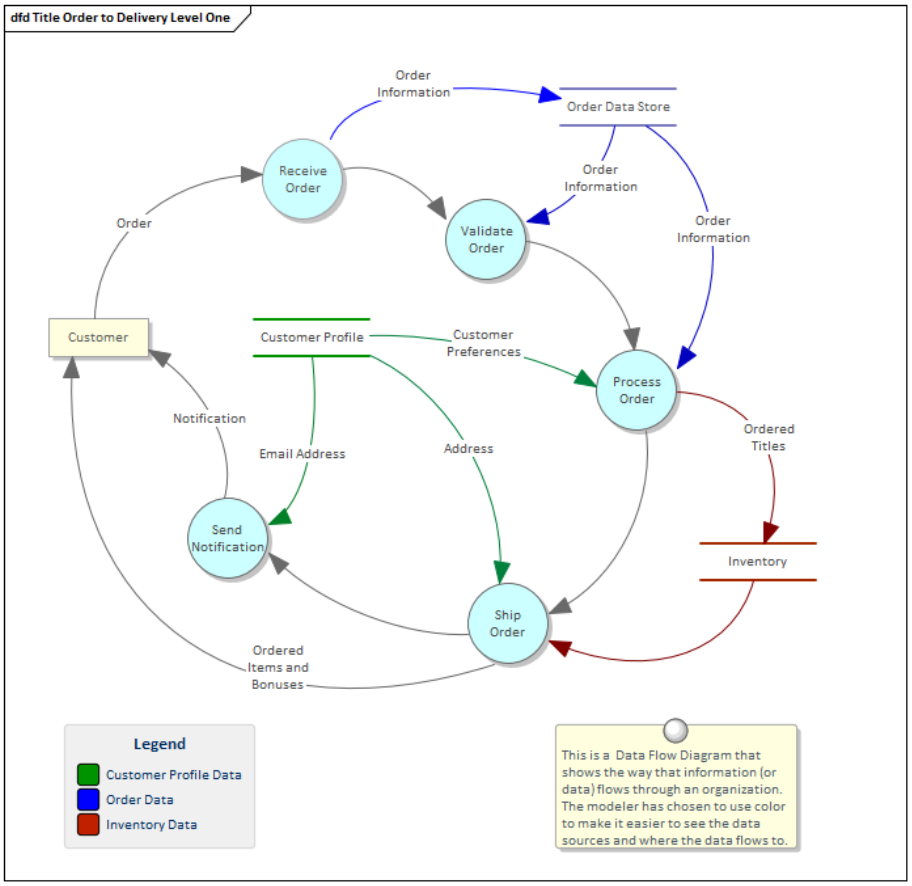
Dijagram na slici 25. prikazuje neke značajke dijagrama slučaja korištenja.

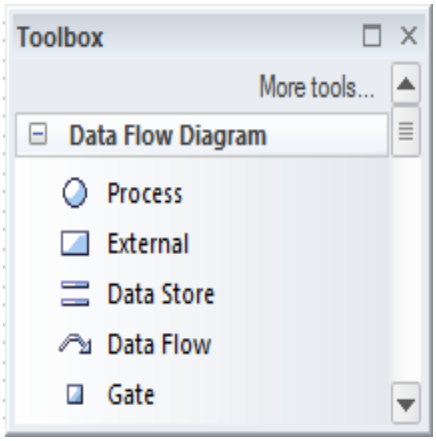


Slika 25. Primjer dijagrama slučaja korištenja

8. Rad s dijagramima – Dijagram toka podataka

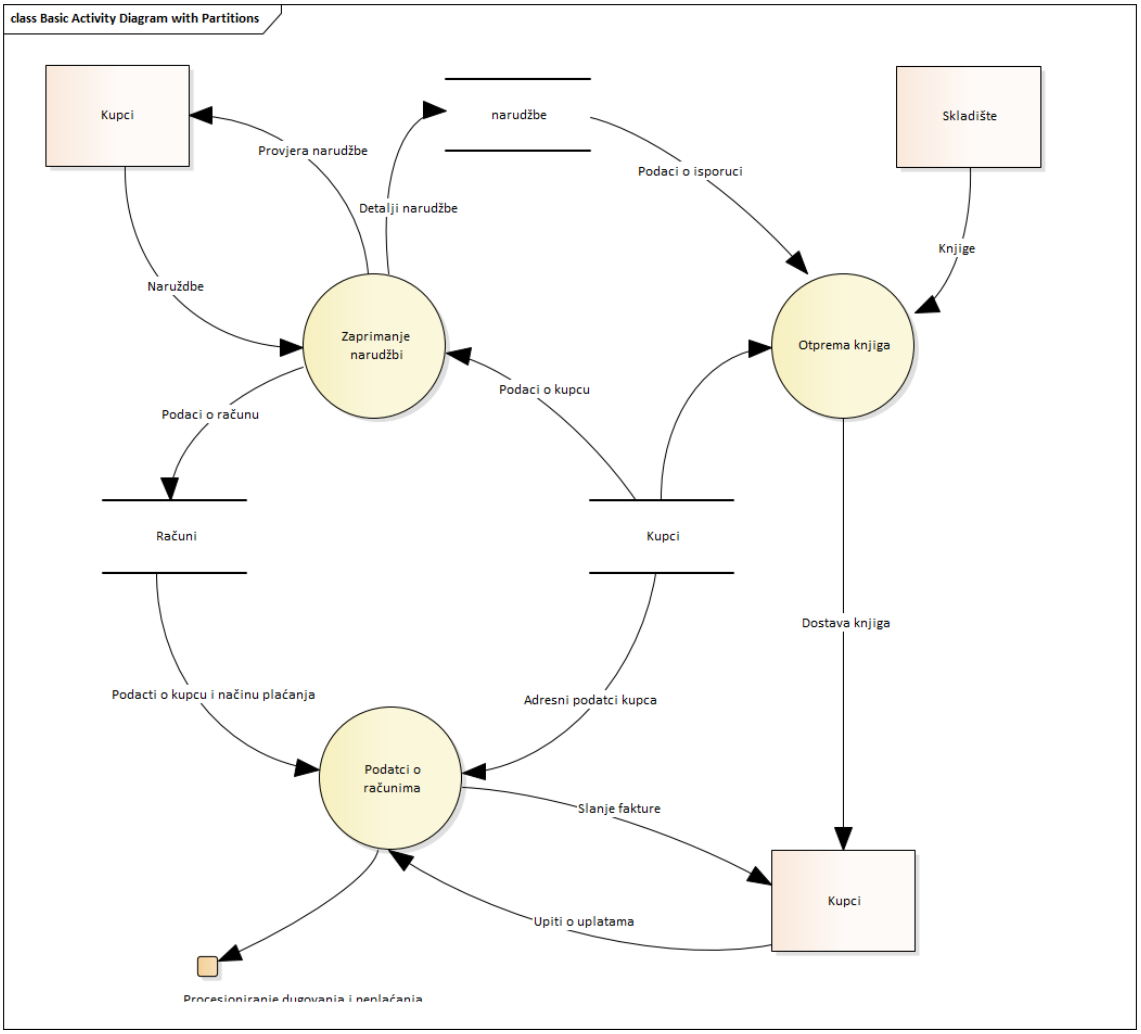
Dijagram toka podataka (DFD) dio je metode analize i projektiranja strukturiranih sustava, a koristi se za prikaz načina na koji podaci protječu kroz sustav, uključujući procese, baze podataka i entitete koji se odnose na podatke.

Aspekt	Opis
Dijagram tijeka podataka	 Hijerarhija dijagrama uobičajeno se kreira tako da započinje s „Kontekst dijagrama“, za koji se kaže da je na 'razini 0' u hijerarhiji.
Kako do dijagrama toka podataka	Vrpca: Dizajn> Dijagram> Umetni> Dijagrami **toka podataka** > Dijagram **toka podataka** Alatna traka **preglednika projekta** : Nova ikona dijagrama> Dijagrami protoka podataka> Dijagram toka podataka Kontekstni izbornik preglednika projekata | Dodaj dijagram ...> Dijagrami toka podataka> Dijagram toka podataka

Korištenje dijagrama toka podataka	Dijagram protoka podataka može se koristiti za stvaranje pogleda na sustav na razini podataka i procesa na bilo kojem nivou. Dok BPMN dijagram poslovnih procesa nije usredotočen na podatke, dijagram toka podataka jest usredotočen na podatke i pokazuje koji procesi troše, proizvode i pohranjuju podatke. Dijagrami su korisni za analitičara podataka, koji treba znati kako se podaci kreću kroz sustav, iz kontekstne razine, gdje se kreću između vanjskih entiteta i kroz sustav, na najnižu razinu.
Opcije za dijagram toka podataka	 Dijagram protoka podataka (kao i bilo koji dijagram), može se promatrati kao popis elemenata, što olakšava rad sa svojstvima elemenata. Dijagrami Filtri se također mogu koristiti prilikom predstavljanja dijagrama, kako bi se skrenula pozornost na dijelove dijagrama, a mogu se prikazati kao ručno nacrtani ili u stilu bijele ploče, promjenom svojstava dijagrama.

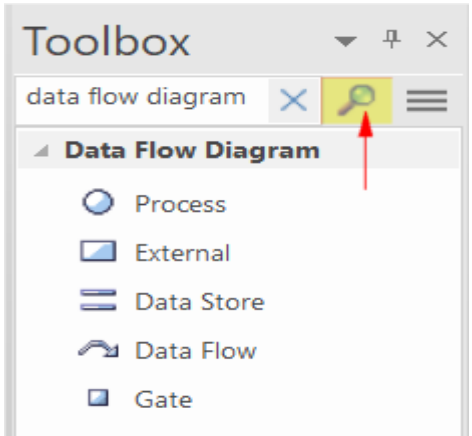
8.1. Primjer dijagrama toka podataka

Kada se povuku na dijagram protoka podataka, elementi i odnosi imaju pojave prikazane slikom 26.

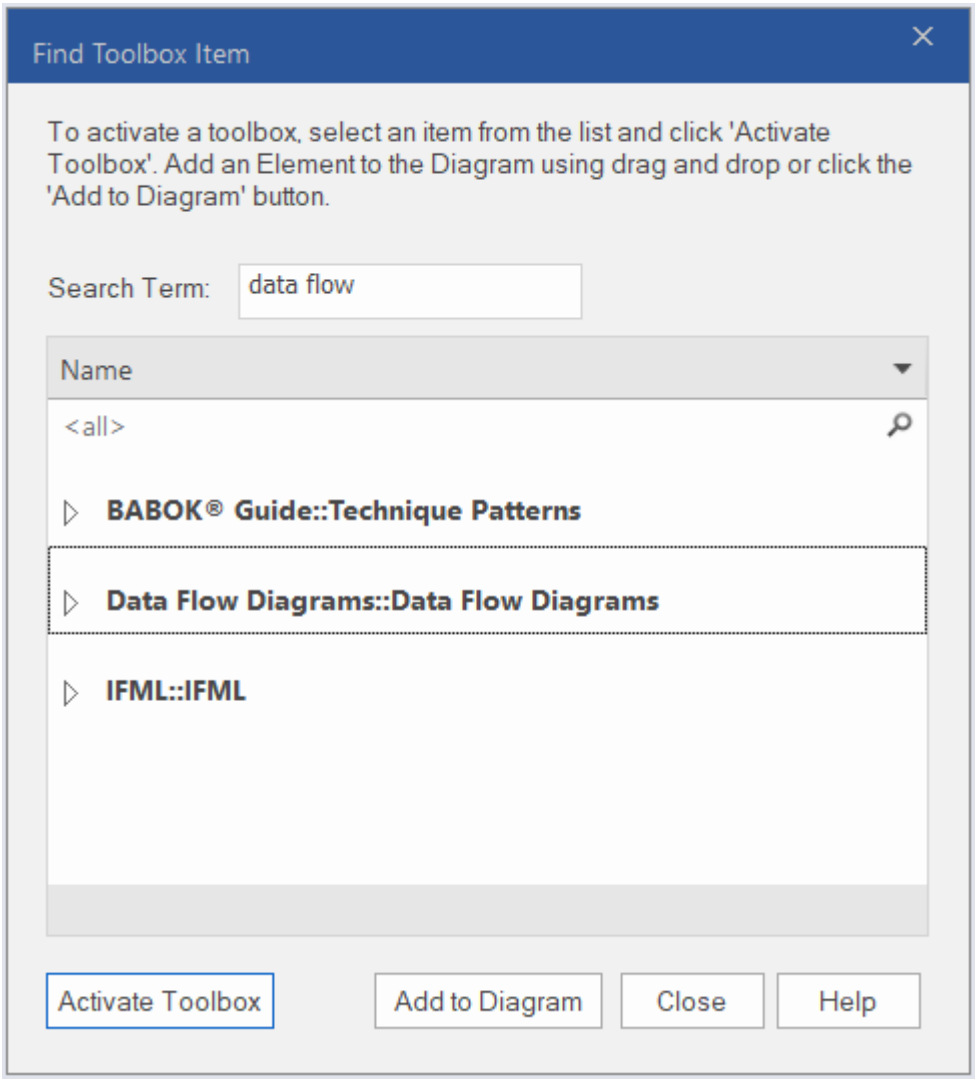


Slika 26. Primjer dijagrama toka podataka

Ovaj dijagram se može također dohvatiti preko Toolboxa na način prikazan sljedećom slikom.



Treba kliknuti na tražilicu i nakon toga dobit ćete ekran 'Find Toolbox Item'.



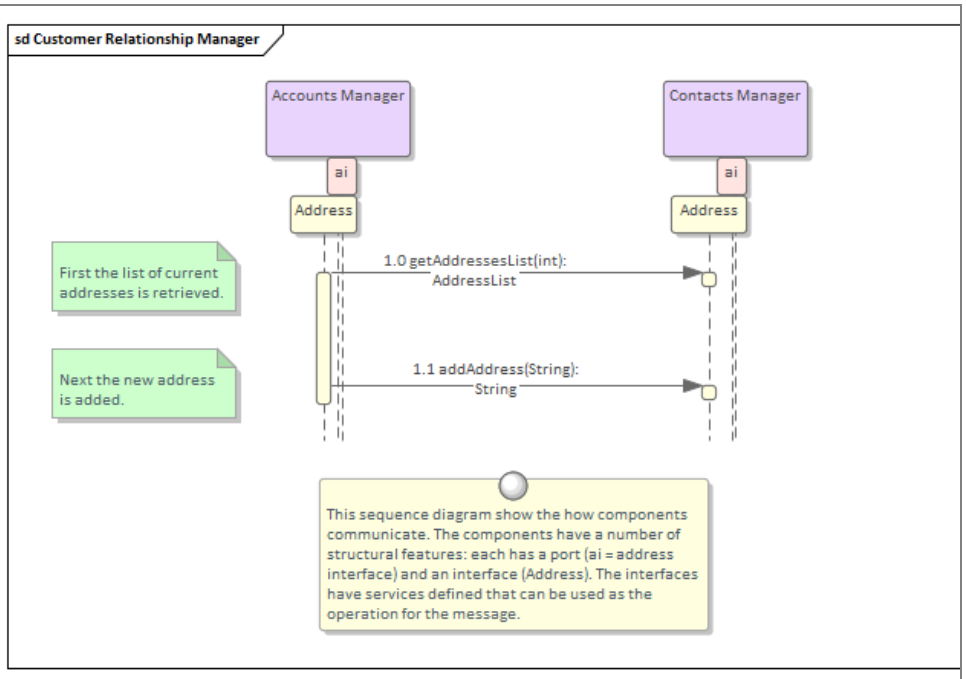
Nakon toga potrebno je upisati pojam koji želite pronaći u polje 'Search Term', u ovom slučaju Data Flow Diagrams. U polju 'Name' pojavit će se veliki izbor dijagrama. Upisom potrebnog dijagrama izbor će se suziti na traženi dijagram.

Sljedeći korak je klik na Activate Toolbox.

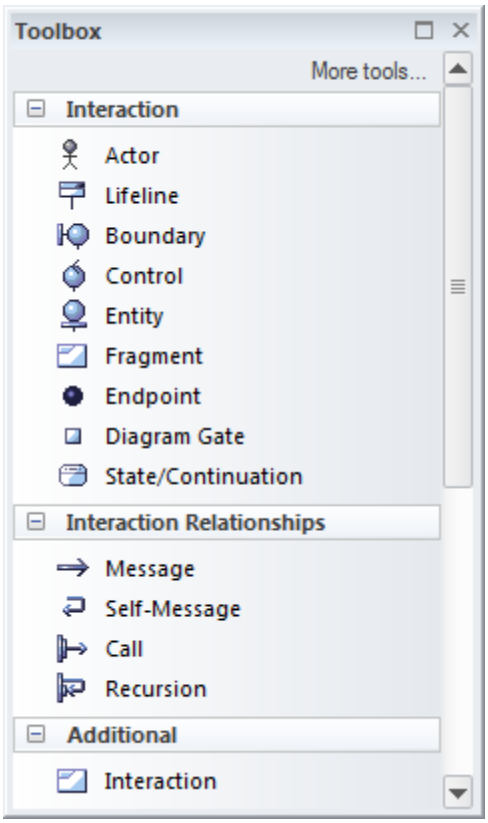
9. Rad s dijagramima – Dijagram slijeda

Dijagram slijeda (Sequence Diagram) prikazuje dinamičku suradnju između određenog broja objekata. Važan aspekt ovog dijagrama je da prikazuje slijed poruka koje se šalju između objekata. Također, prikazuje i interakciju između objekata, nešto što se događa u određenoj točki izvršenja sustava. Dijagram se sastoji od određenog broja objekata prikazanih okomitom linijom života. Vrijeme prolazi silazno u dijagramu i prikazuje izmjenu poruka između objekata u vremenu prema slijedu funkcija. Poruke su prikazane strelicama između okomitih linija života. Specifikacija može biti prikazana kao ograničenje u dijagramima, a komentari mogu biti dodani u skriptu kao margina dijagrama. Dijagram slijeda prikazuje interakcijski fragment. Ti fragmenti mogu staviti operatore u gornji kut, što pokazuje specijalno rukovanje za sekciju.

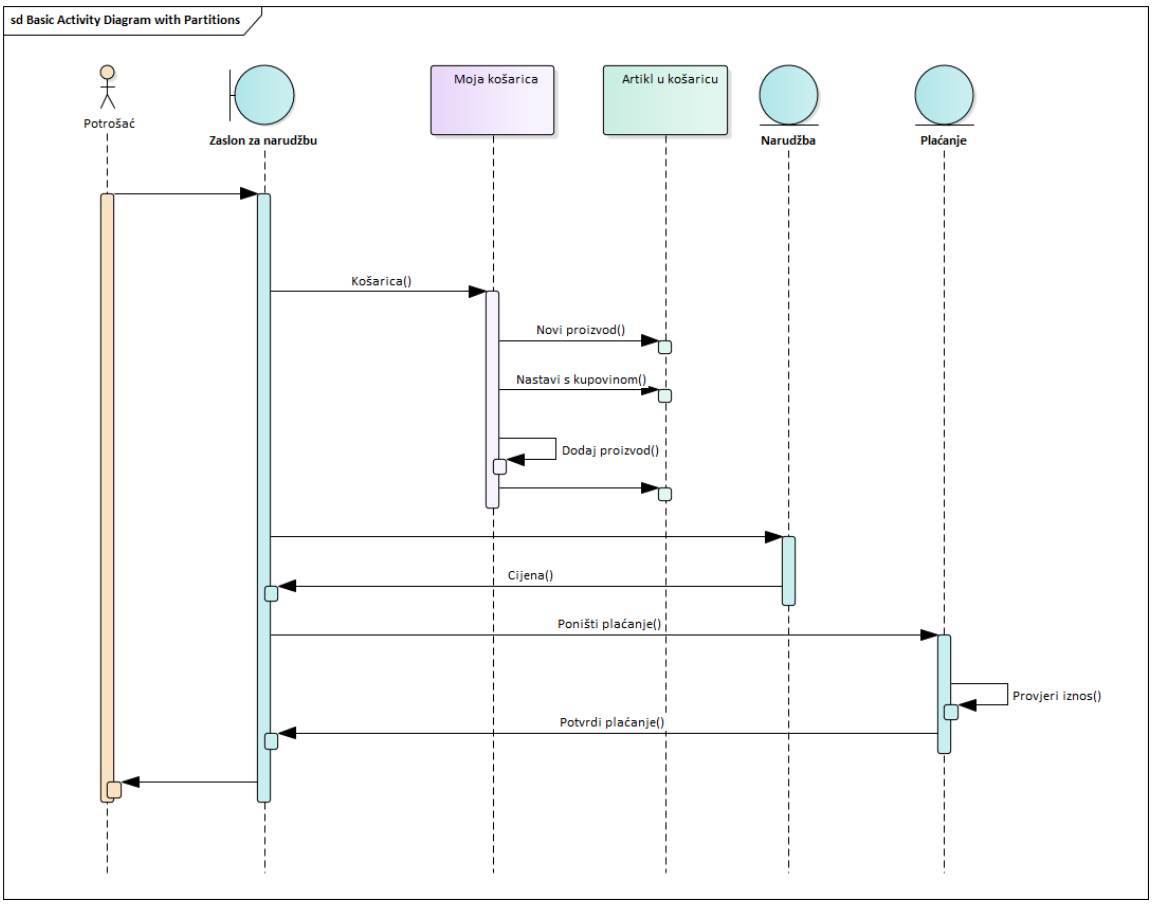
Aspekt	Opis
Uvođenje dijagrama slijeda	Dijagram sekvenci je jedan od Unified Modeling Language (UML) Bihevioralnih dijagrama koji se mogu koristiti za definiranje vremenskog tijeka poruka između elemenata. Češće se koriste za tehničke opise sustava, iako ih mogu koristiti i poslovni i tehnički korisnici. Na okomitoj osi vrijeme je prikazano konvencionalno, a elementi (Klasifikatori ili instance) su postavljeni vodoravno, na samom vrhu dijagrama. Uobičajeno, glumac (koji predstavlja ulogu korisnika) pokreće protok, a poruke se prenose između elemenata koji pokazuju kako je rezultat postignut.

	
Kako do dijagrama slijeda	Vrpca: Dizajn> Dijagram> Umetni> UML ponašanje> Slijed Alatna traka **preglednika projekta** : Nova ikona dijagrama> UML ponašanje> Slijed Kontekstni izbornik preglednika projekata | Dodaj dijagram ...> UML ponašanje> Slijed
Korištenje dijagrama slijeda	Da bi se dobio rezultat, dijagram sekvence može se koristiti za modeliranje bilo kojeg skupa poruka, koje teku između dva ili više elemenata. Uobičajenije se koriste od strane tehničkih modelara, da bi prikazali kako se poruke šalju između aplikacijskih komponenti i sučelja te da bi se postigao ishod. Posebno su korisni kada služe za opisivanje složenih protokola ili interakcija podsustava koje je teško razumjeti, osim ako nisu prikazane grafički. Dijagrami sekvenci također se mogu kreirati automatski iz tragova stogova, kao način vizualizacije funkcioniranja sustava.
Opcije za dijagram slijeda	Elementi koji se pojavljuju u dijagramima sekvence mogu biti ili klasifikatori, kao što su Klase i Komponente ili instance, kao što su Objekti i Elementi komponente. Izbor će ovisiti o tome što modelar želi predstaviti. Da bi dijagram bio čitljiviji, redoslijed objekata i

poruka može se promijeniti, a bilješke se mogu koristiti za objašnjavanje dijelova dijagrama. Također, mogu se dodati i fragmenti, kako bi predstavili kontrolne strukture koje djeluju na skupinu poruka.



9.1. Primjer dijagrama slijeda



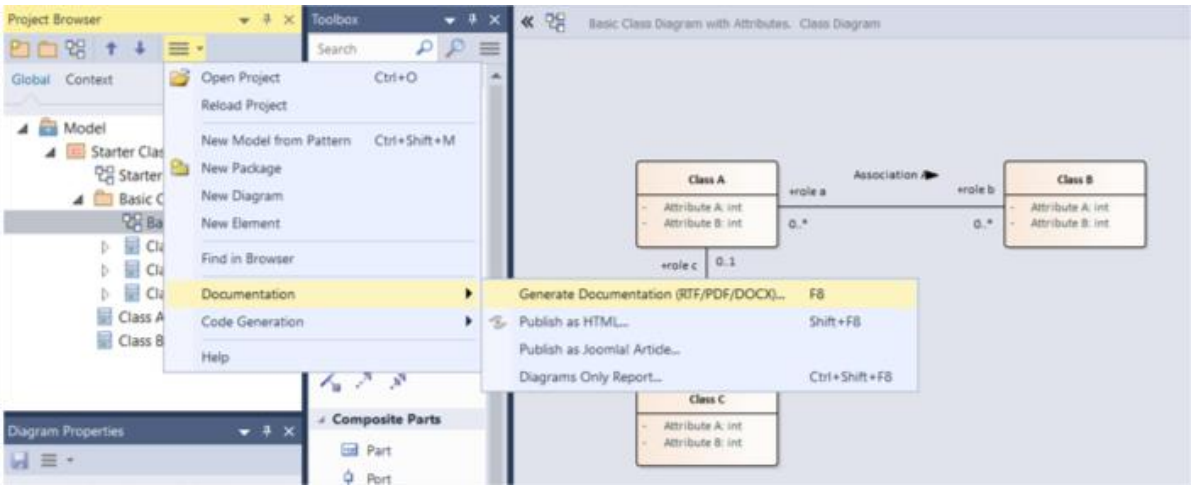
Slika 27. Primjer dijagrama slijeda

10. Generiranje dokumenata i koda

Sljedeća mogućnost koju pruža Enterprise Architect je mogućnost generiranja dokumentacije i to u nekoliko ekstenzija, kao što su RTF, PDF, DOCX, itd.

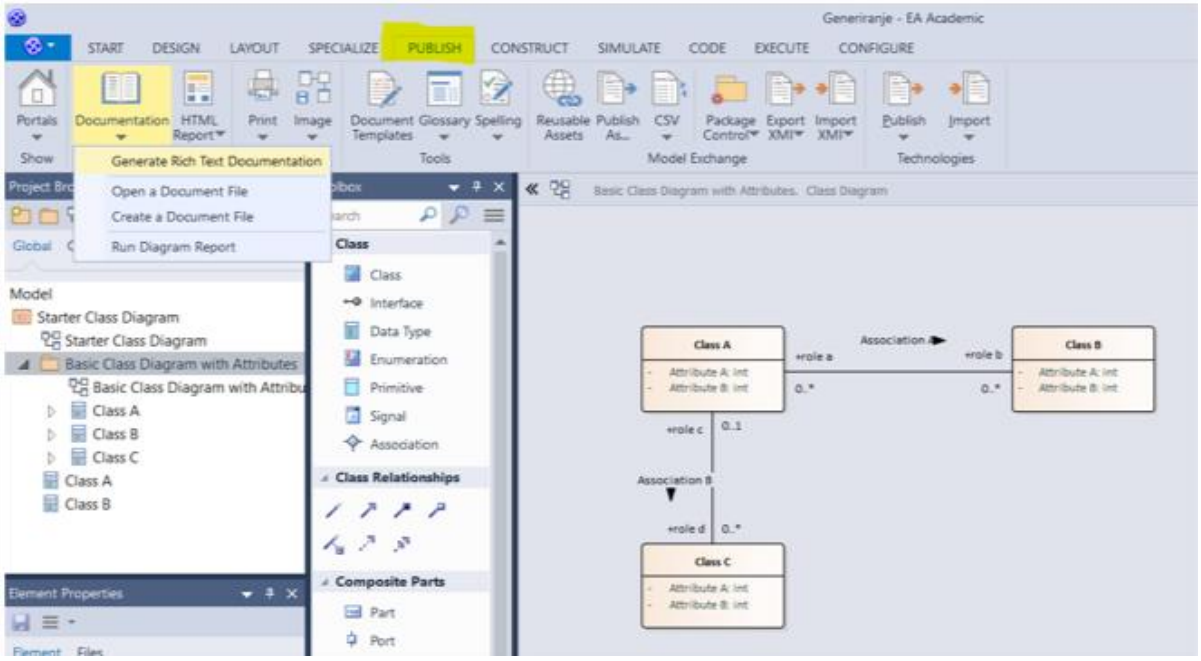
10.1. Generiranje dokumentacije

Jedan od načina generiranja dokumentacije prikazuje Slika 28.



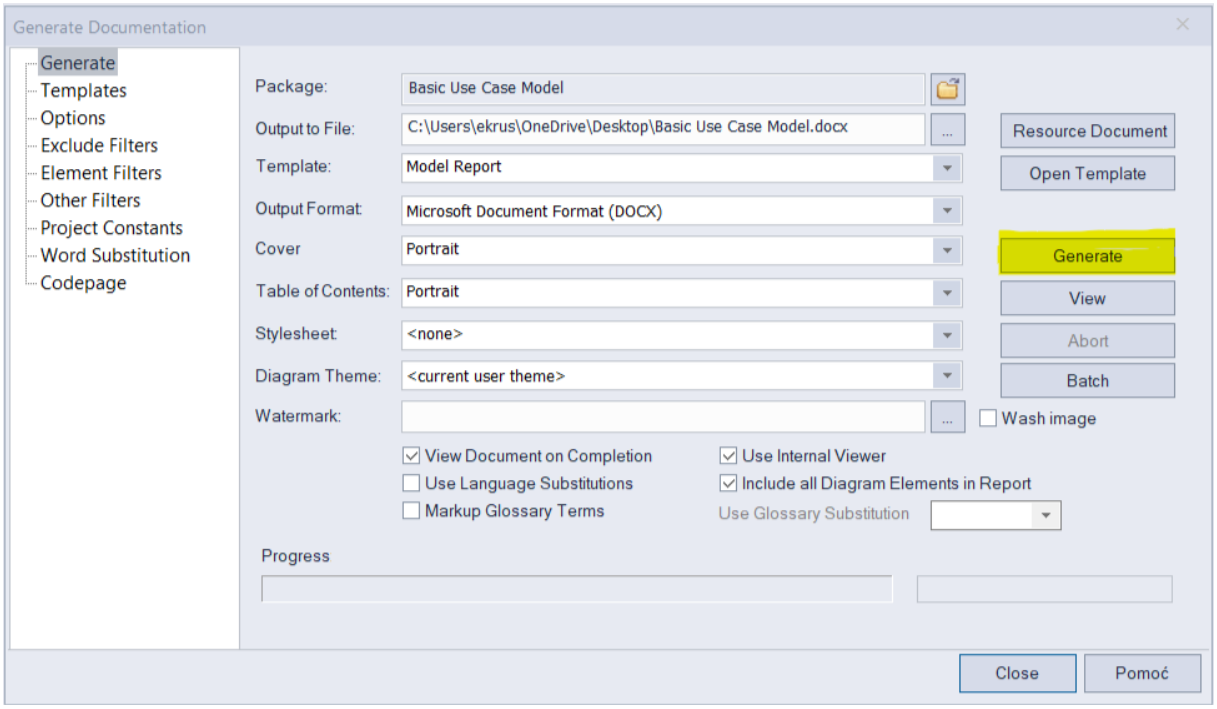
Slika 28. Generiranje dokumentacije

Osim odabira naredbe s trake '**Project Browsera**', postoji i mogućnost korištenja tipkovnice. Za generiranje dokumentacije može se koristiti gumb F8 ili preko kartice 'Publish', slika 29.



Slika 29. Drugi načini generiranja – preko kartice 'Publish' ili gumba F8

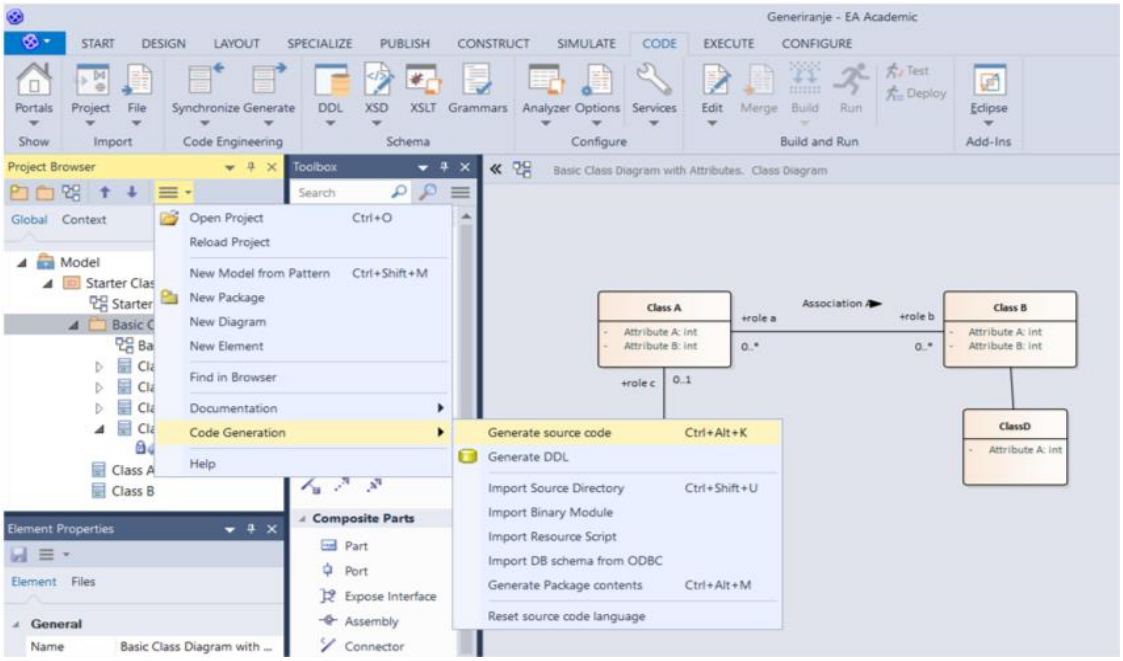
Nakon odabira dobit će se novi ekran za generiranje dokumenta.



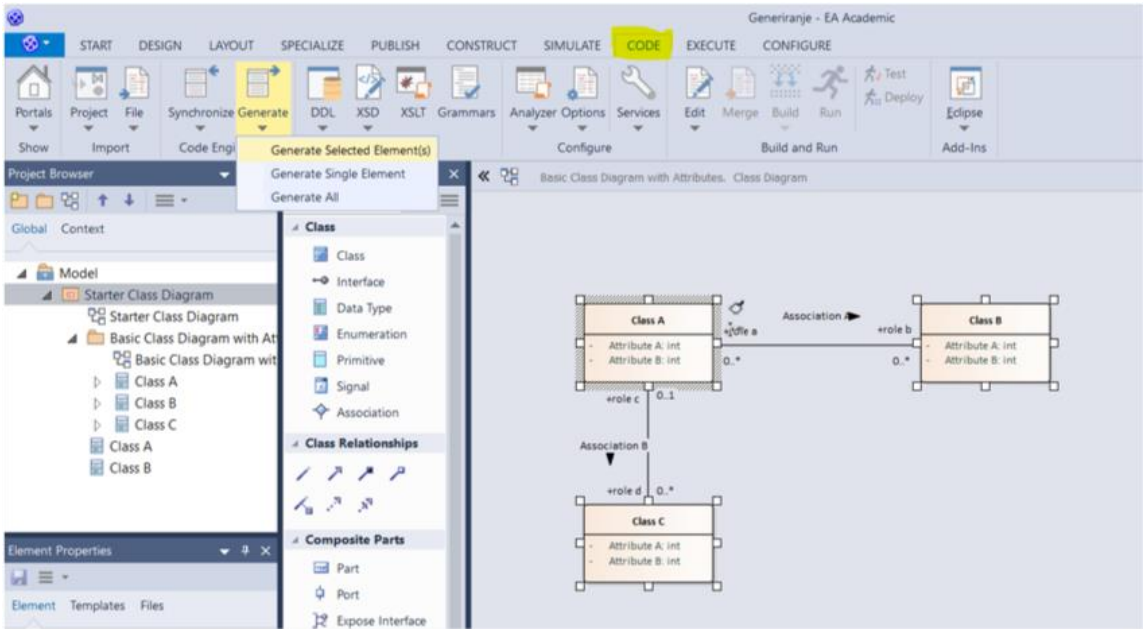
Slika 30. Nastavak rada s čarobnjakom za generiranje dokumenata

10.2.Generiranje izvornog koda

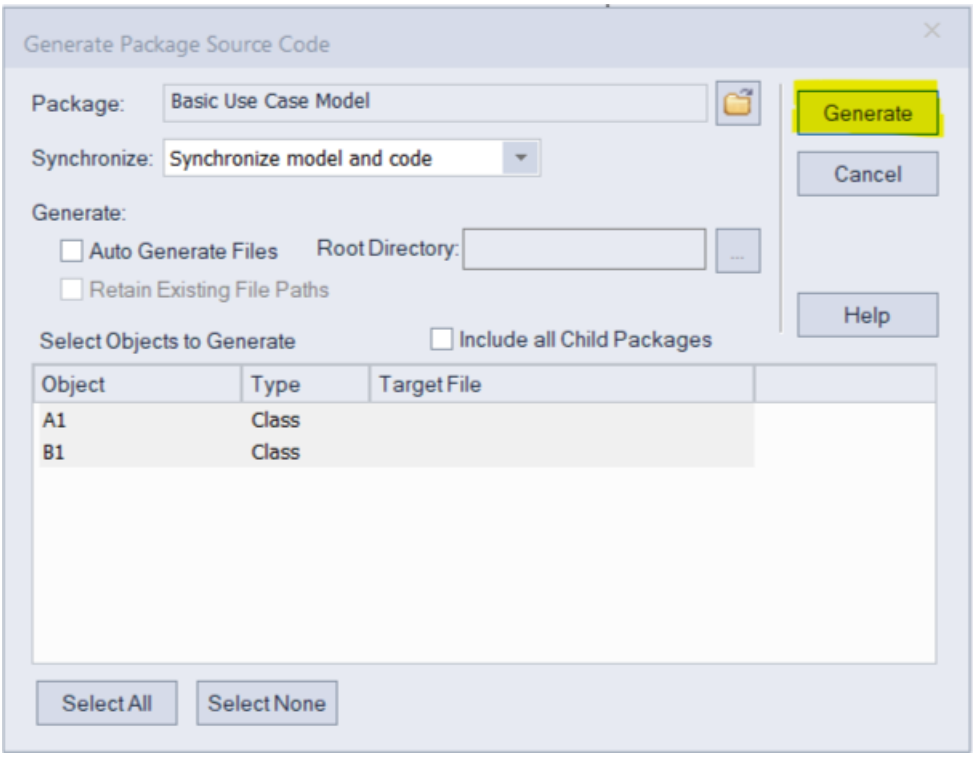
Generiranje izvornog koda znači da je korisnik stvorio UML dijagrame, koji imaju neki određeni model podataka, a da ga Enterprise Architect izvodi (eng. Derivation) iz dijelova dijagrama ili cijelog dijagrama u izvorni kod za softverski sustav.



Slika 31. Generiranje izvornog koda

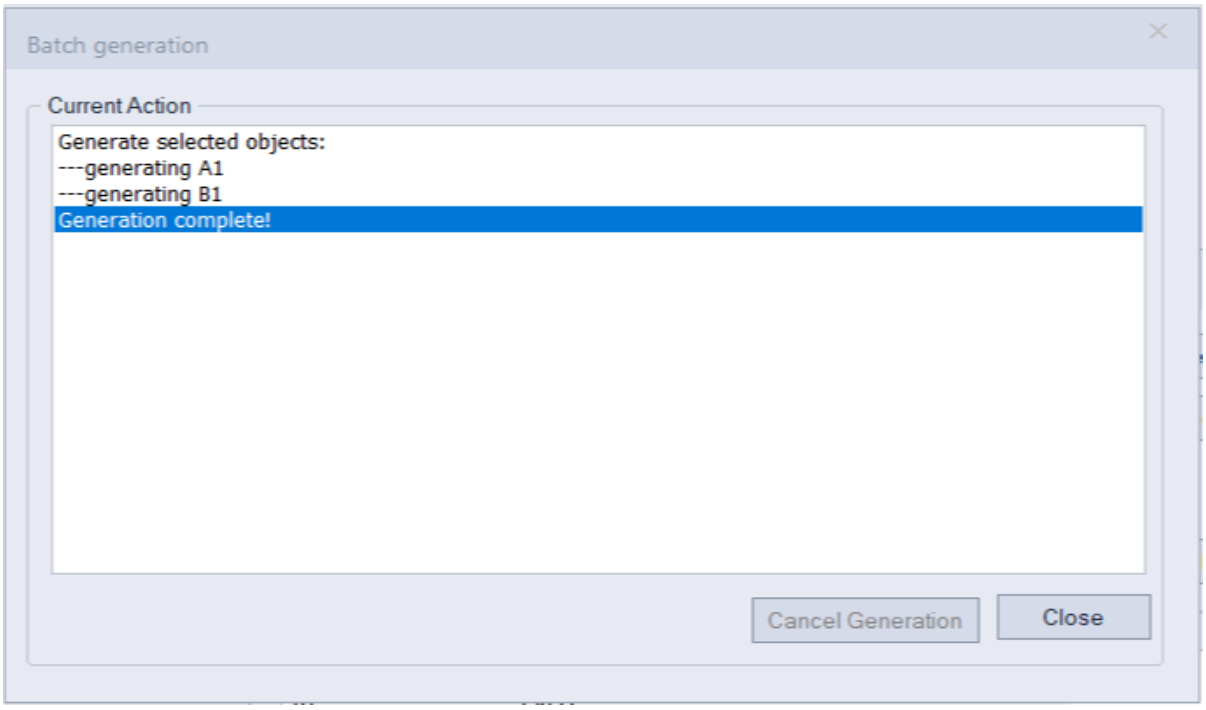


Slika 32. Drugi načini generiranja – preko kartice 'Code' ili gumba F8



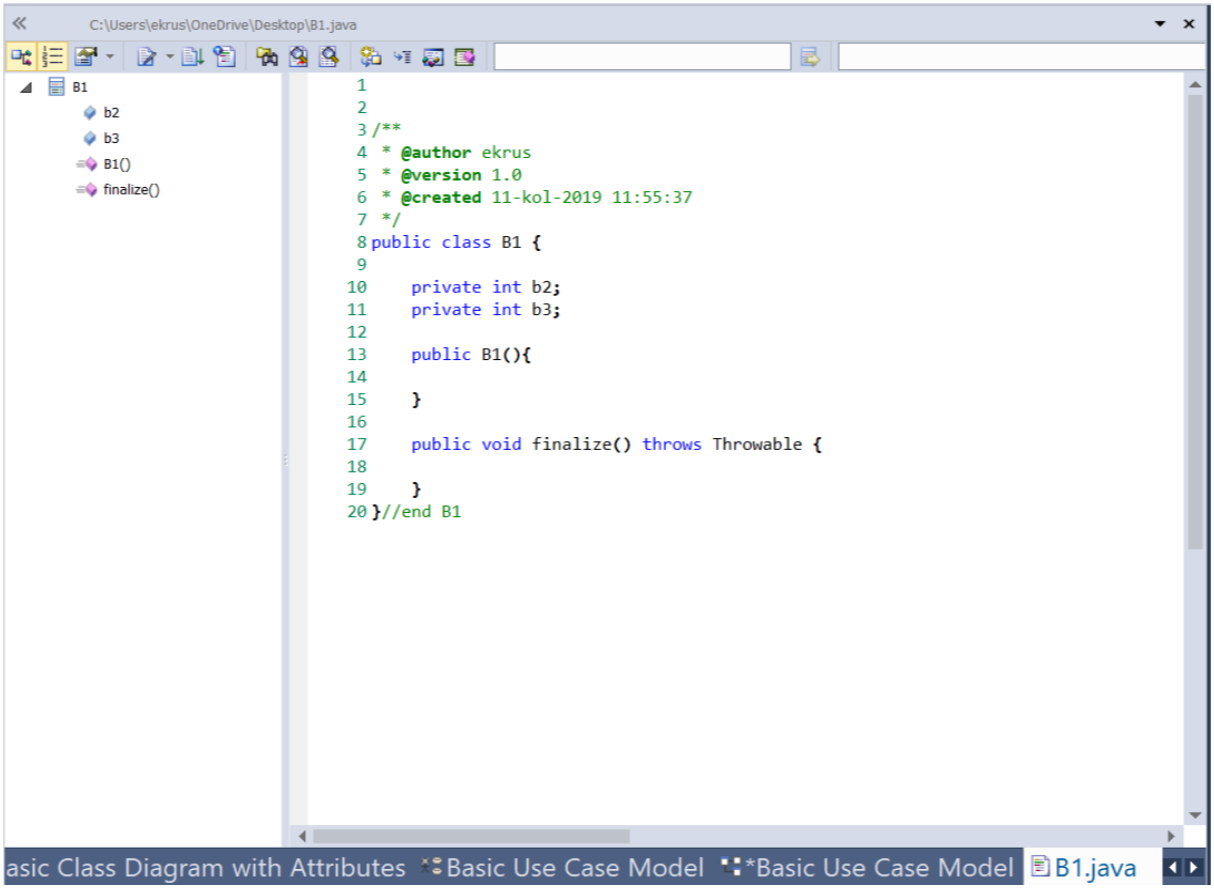
Slika 33. Odabir entiteta

Nakon odabira Entiteta za spremanje, slijede prikazi spremanja pojedinačnih Entiteta. Svaki pojedinačni Entitet moći će se spremati toliko puta, koliko će biti izrađenih Entiteta.



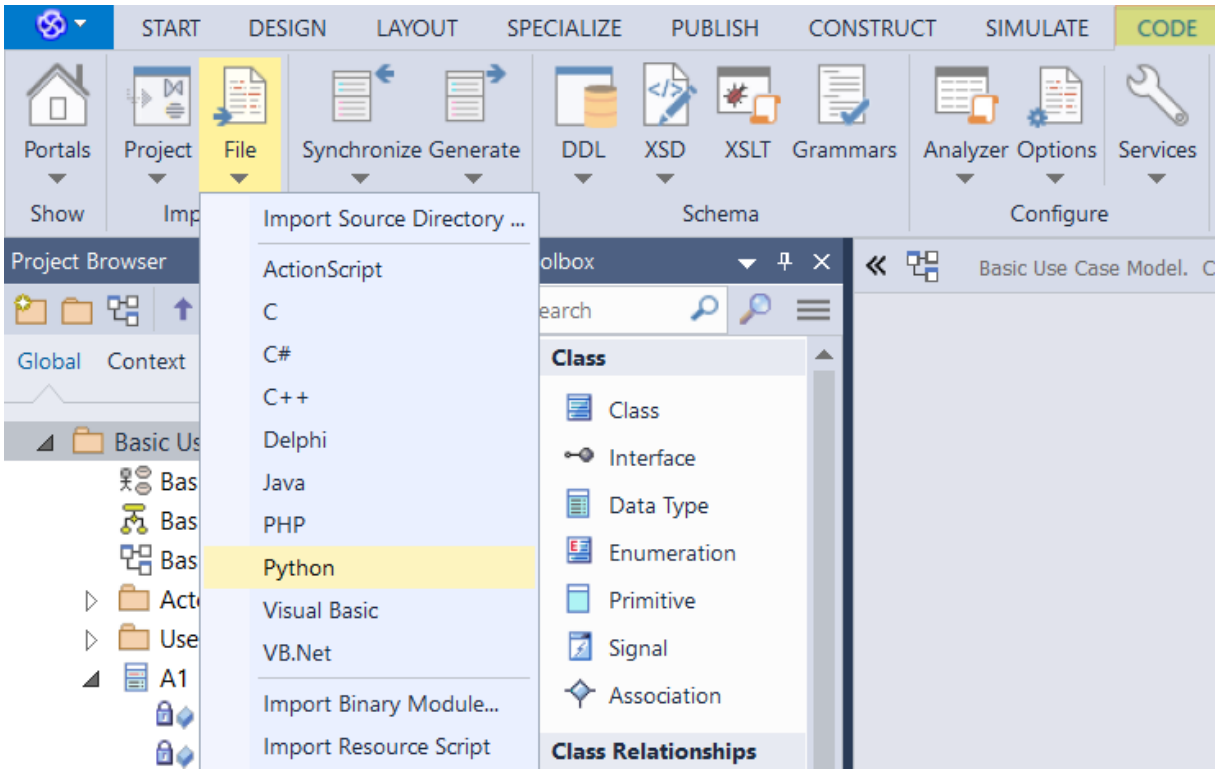
Slika 34. Batch obrada - generiranje izvršeno

Nakon što su generirani objekti snimljeni i smješteni na neko odabrano mjesto na računalu, na sljedećoj slici pojavljuje se generirani kod.



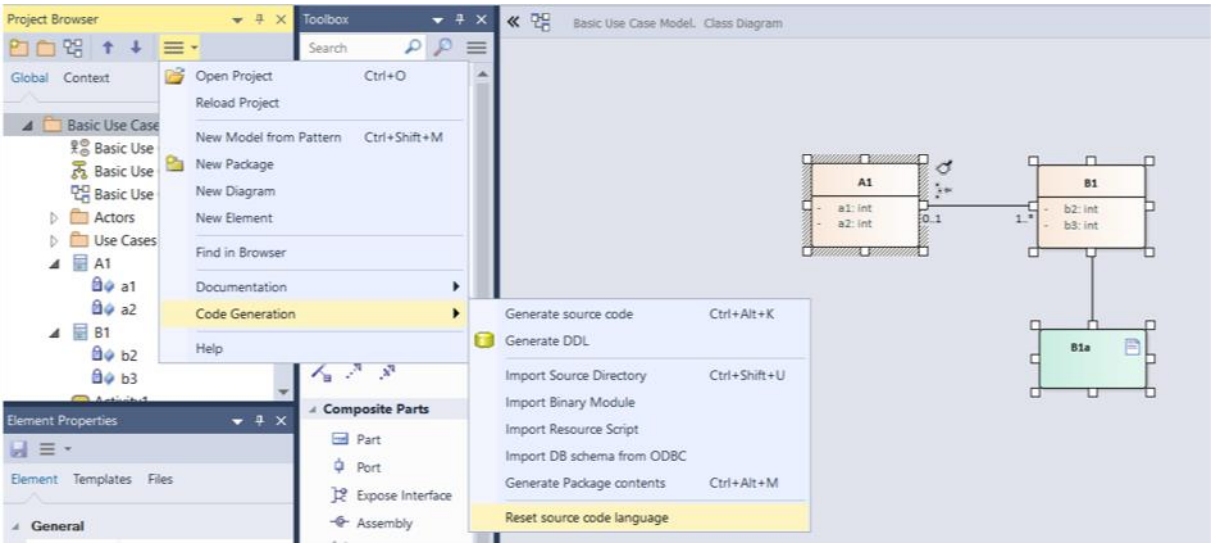
Slika 35. Primjer generiranog koda

Odabir programskih jezika moguće je odabrati preko kartice 'Code' i naredbe 'File', a primjer je prikazan na slici 36.

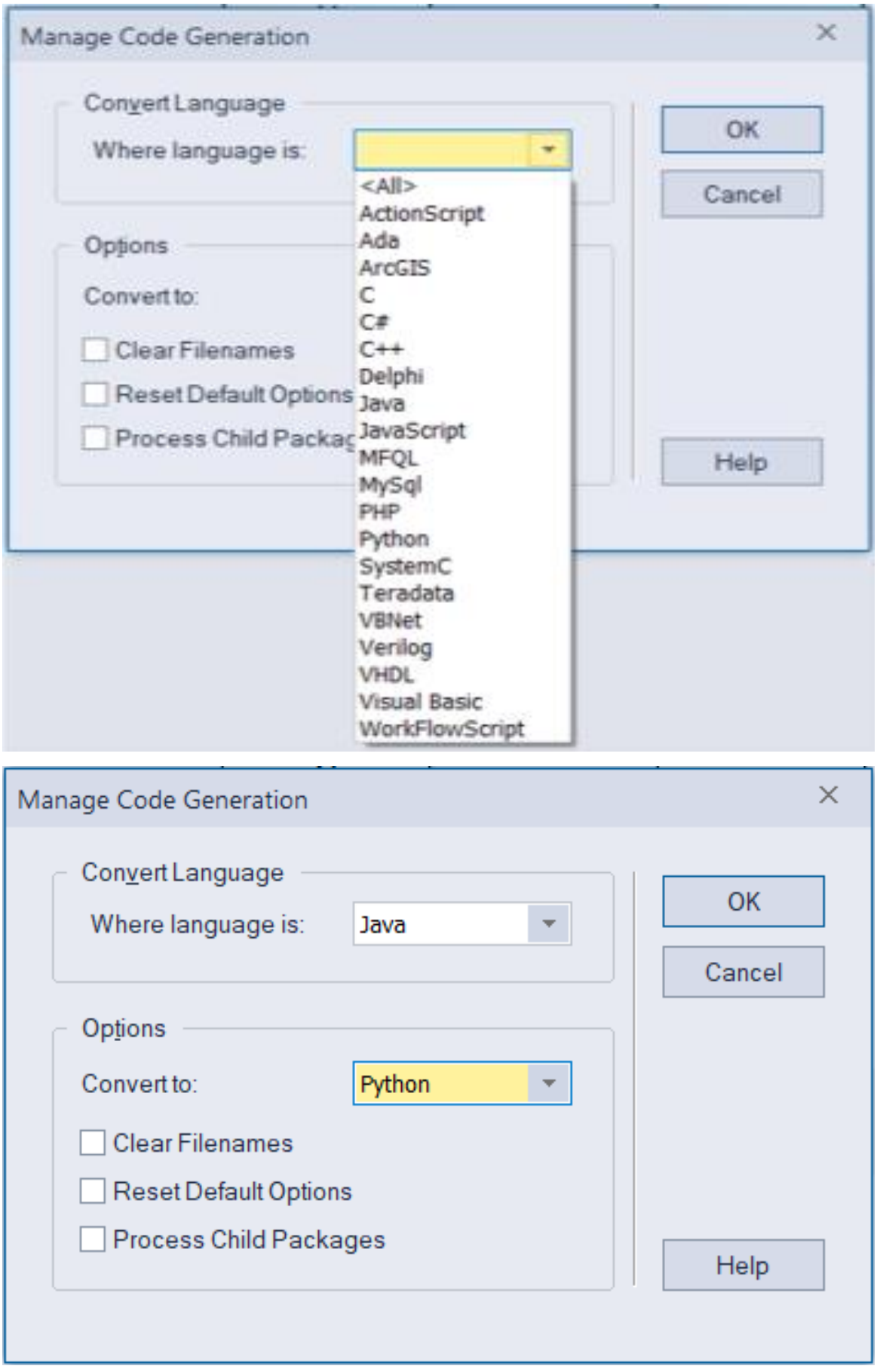


Slika 36. Odabir programskog jezika

Druga mogućnost je putem opcije 'Project Browser-a'. Nakon odabira 'Code Generation', dobit ćete padajući izbornik. Sljedeći korak je odabir opcije 'Reset source code language'.



Slika 37. Zamjena ili ponovno namještanje programskih jezika



Slika 38. Upravljanje generiranjem koda

11. Prvi koraci

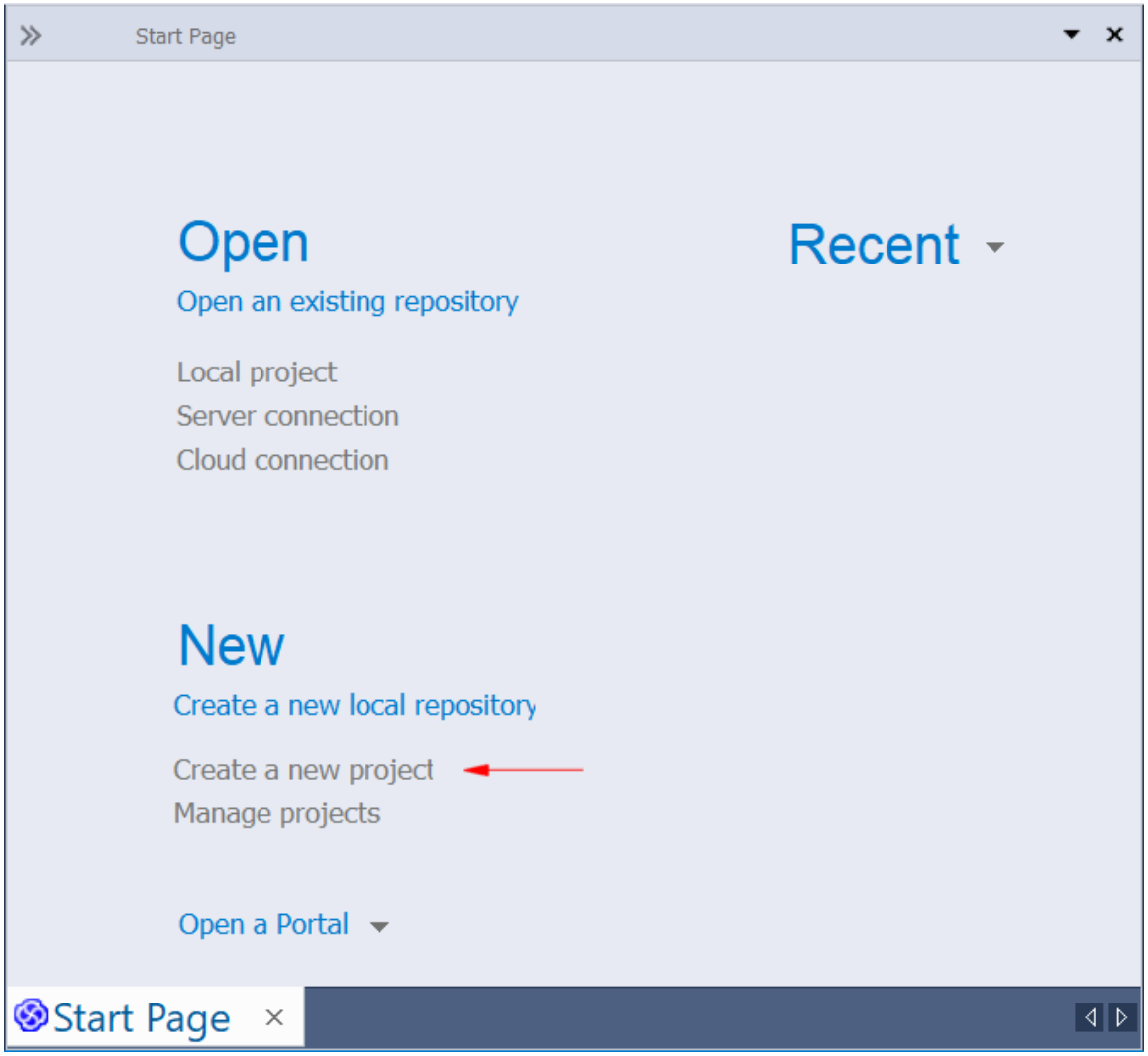
I putovanje od tisuću kilometara započinje prvim korakom. (Lao Tse – kineski filozof)

11.1. Važno je znati

Nakon pokretanja Enterprise Architect-a, prije nego što se počne modelirati i kreirati dijagrame, **ako nema** odabranog **trenutnog spremišta**, treba otvoriti postojeći model ili stvoriti novi. Ako se prvi put radi s Enterprise Architectom i nema niti jedan postojeći model za upotrebu, treba stvoriti novo spremište, na temelju datoteka kojima se možete eksperimentirati i igrati, dok se uči o modeliranju i korištenju Enterprise Architecta.

Izrada nove datoteke spremišta.

Korak	Opis akcije
•	Pritisnuti **Ctrl + N,** što je tipka prečaca za stvaranje novog spremišta, temeljenog na datoteci.
•	U odjeljku „novi” na Enterprise Architect **Početak stranice**, kliknuti na opciju „Stvori novi projekt”.

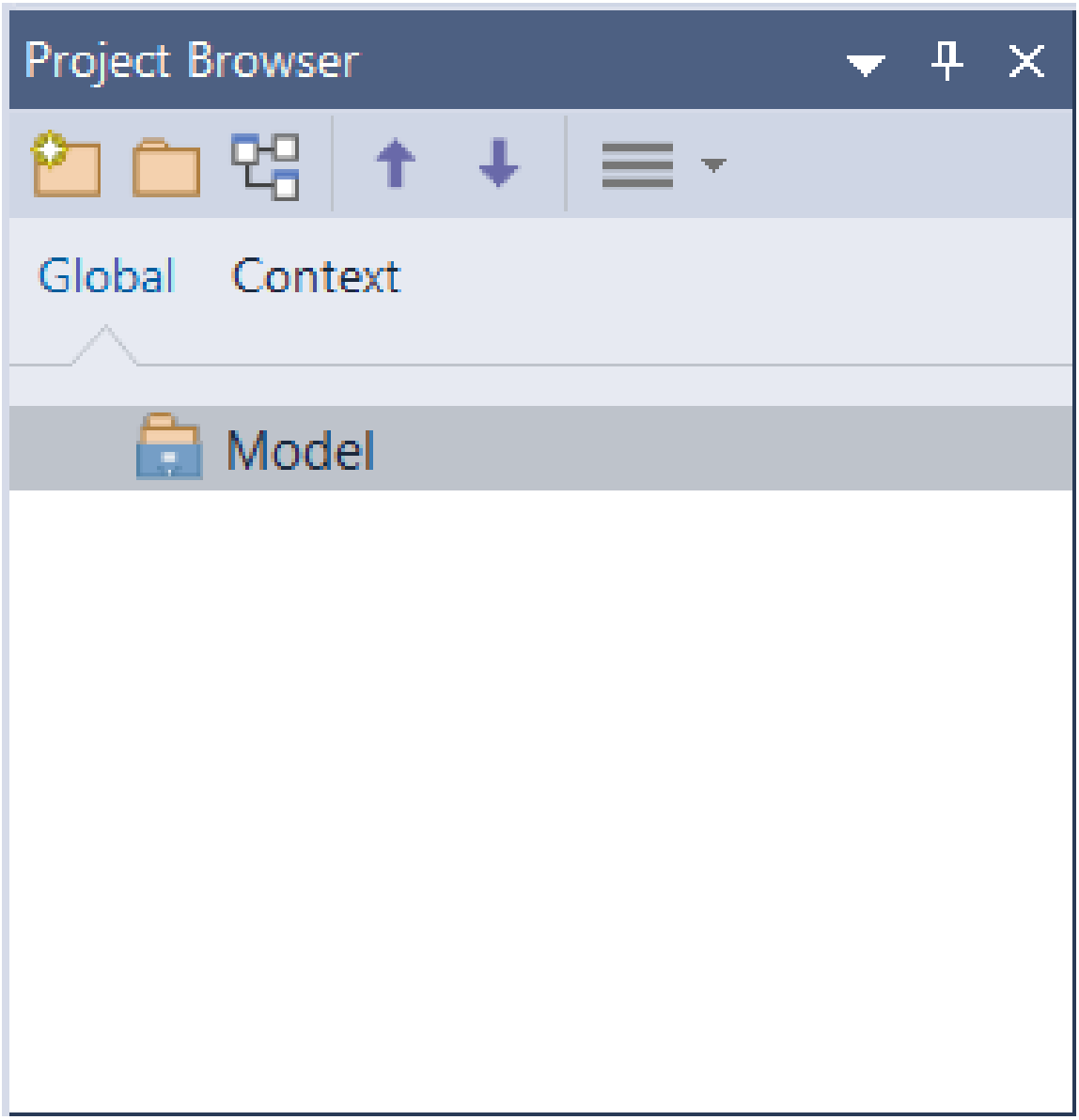


Slika 39. Odabrati naziv datoteke i lokaciju za spremište

Traži se da se odabere naziv datoteke i mjesto za novo spremište. Nakon što se odabere ime i lokacija, potrebno je kliknuti na **gumb Spremi** - Enterprise Architect će izraditi novi prazan model, kojim se počinje raditi.

11.2.Što se promijenilo?

Iako se čini da se nije mnogo toga promijenilo, kreirano je novo radno spremište u koje se mogu početi dodavati novi elementi i dijagrami. Osnovno što se treba primijetiti jest da **preglednik projekta** sada ima novi korijenski element, pod nazivom 'Model', koji je polazna točka iz koje će se izrađivati vlastiti pojedinačni repozitorij.



Slika 40. Stvaranje korijenskog elementa

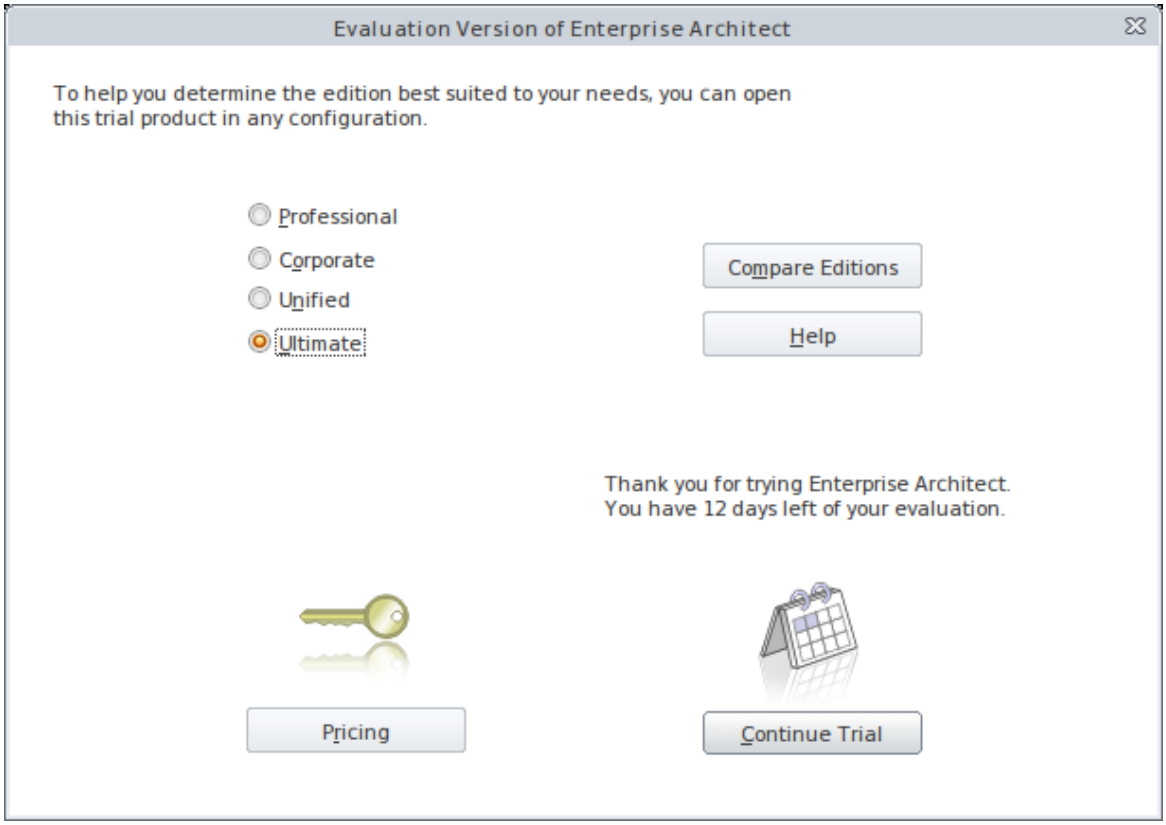
Do sada je pokrenuta probna verzija, stvoreno novo spremište i učitano to spremište u radni prostor Enterprise Architect.

12. Pokretanje Enterprise Architecta

Sada je vrijeme za pokretanje Enterprise Architecta i pronalaženje načina za korisničko sučelje. Potrebno je napraviti nekoliko izbora, kako bi se probno izdanje prilagodilo svojoj svrsi. Pri tome, treba imati na umu da sve odluke koje se donose nisu konačne i da se mogu promijeniti ponovnim pokretanjem Enterprise Architecta ili korištenjem različitih alata za prilagodbu. Uz pretpostavku da je instaliran Enterprise Architect, sada se može kliknuti na ikonu radne površine (ako je dostupna) ili pronaći prečac u izborniku Start, ako je to potrebno.

12.1. Korištenje probnog izdanja

Tijekom procesa otvaranja, pojavit će se dijalog.

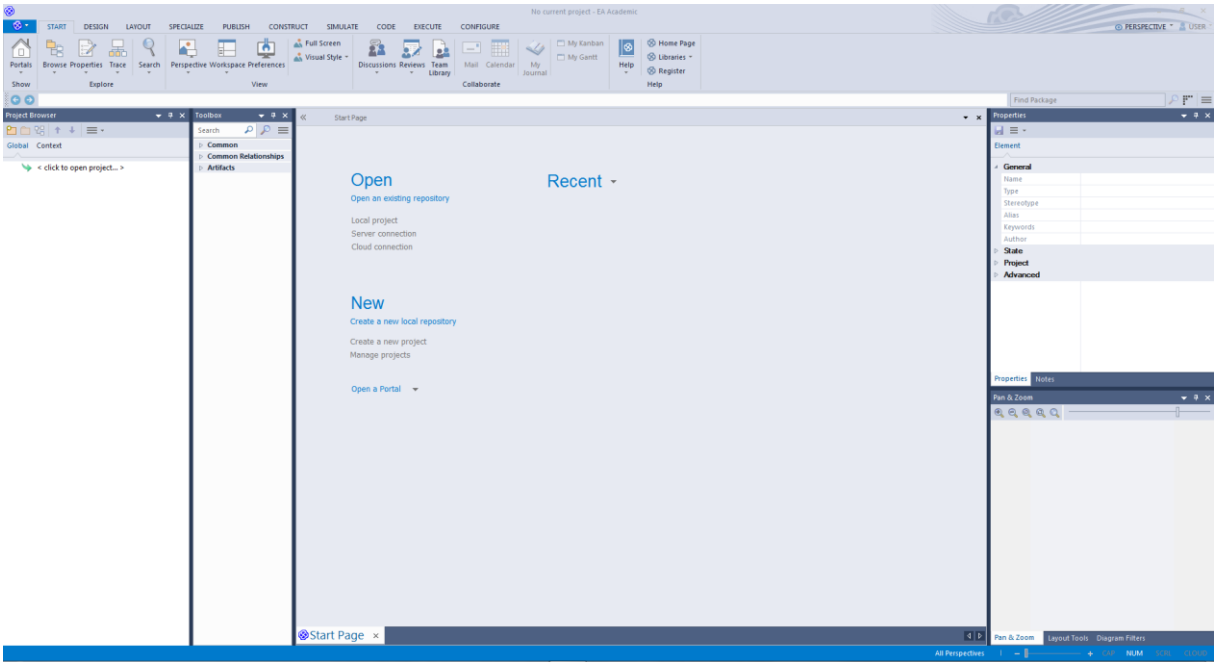


Slika 41. Odabir EA verzije

Ovaj dijalog potvrđuje da se počelo koristiti probno izdanje za 30-dnevno probno razdoblje koje pruža mogućnost pristupa web-mjestu za pregled cijena i opcija kupnje proizvoda. Prikazuje se i skup radio gumba, s kojih se odabire izdanje Enterprise

Architecta koje se želi isprobati. Klikom na **gumb Usporediti izdanja,** dobivaju se **i**nformacije o tome što uključuje svako od različitih izdanja, dostupno na web-lokaciji.

Kako bi se moglo istraživati i uspoređivati različita izdanja onoliko puta, koliko je potrebno tijekom probnog razdoblja, ovaj će se dijaloški okvir prikazati svaki put kada se otvori probna verzija. Treba kliknuti na radio gumb za bilo koje izdanje i na **gumb Nastavi probnu verziju**; prikazat će se glavni radni prostor Enterprise Architect-a.



Slika 42. Glavni radni prostor Enterprise Architect-a

Ovaj radni prostor može se prilagoditi tako da poprimi drugačiji izgled i sadržaj, kako bi maksimalno podržao radne zadatke.

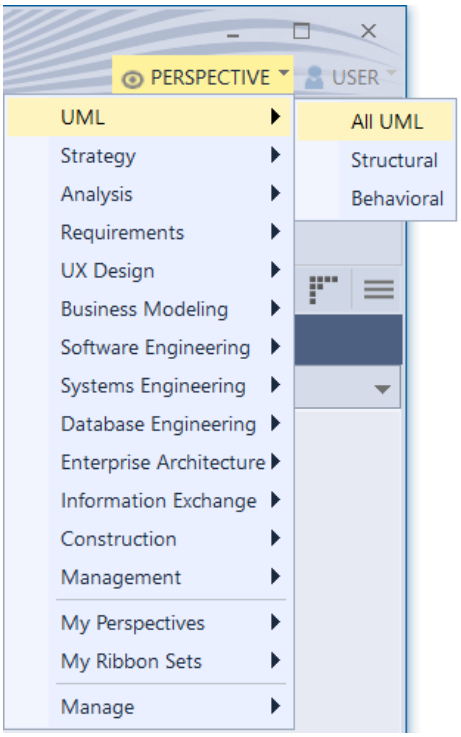
13. Izrada modela

Sada, kada je stvoren novi model i učitan u radni prostor, sljedeći cilj je **dodati** novi model, modelu. Iako postoji mnogo načina za dodavanje sadržaja, za ovaj prvi primjer koristit će se prikladan prečac i dodavati sadržaj modela pomoću uzorka modela. Da bi se to učinilo, potrebno je sljedeće:

- Odabrati odgovarajuću perspektivu
- Odabrati uzorak modela
- Upoznati se sa sadržajem uzorka
- Dodati sadržaj uzorka u spremište
- Pronaći novi sadržaj u **pregledniku projekata** i istražiti nove dijagrame.

13.1. Odabir perspektive

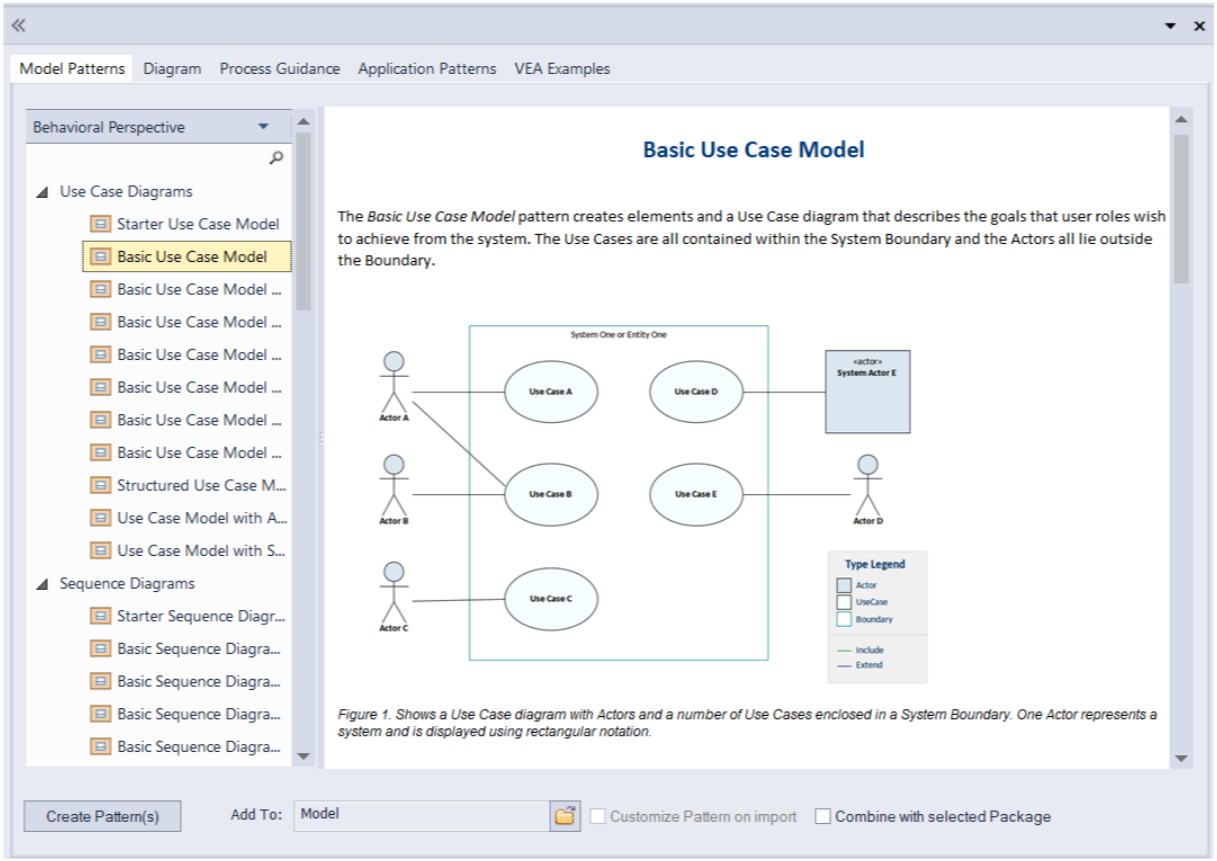
Kao što je već spomenuto, Perspektiva je ograničenje koje se postavlja na radni prostor, samo za prikaz ograničenog skupa tehnologija. Perspektivu se može odabrati na više načina, no za ovu šetnju koristit će se padajući izbornik u gornjem desnom kutu korisničkog sučelja. Cilj je stvoriti jednostavan model primjera upotrebe, koji se kasnije može proširivati i prilagođavati. Kada se pronađe izbornik **Perspective** (Perspektiva), treba kliknuti na njega i odabrati 'UML | Sve UML-ove, kao što je prikazano na slici 43.



Slika 43. Izbornik perspektiva

13.2.Korištenje uzorka

Na taj način će se **Čarobnjak modela** pojaviti u središnjem području dizajna i unaprijed postaviti tako da prikazuje sve UML obrasce modela (a ima ih dosta!) Potrebno je pomicati se prema dolje, dok se ne pronađe odjeljak "Koristi slučaj" i odabrati "Osnovno" 'Koristiti model slučaja'. Primjer je prikazan na slici 44.



Slika 44. Osnovni model slučaja korištenja

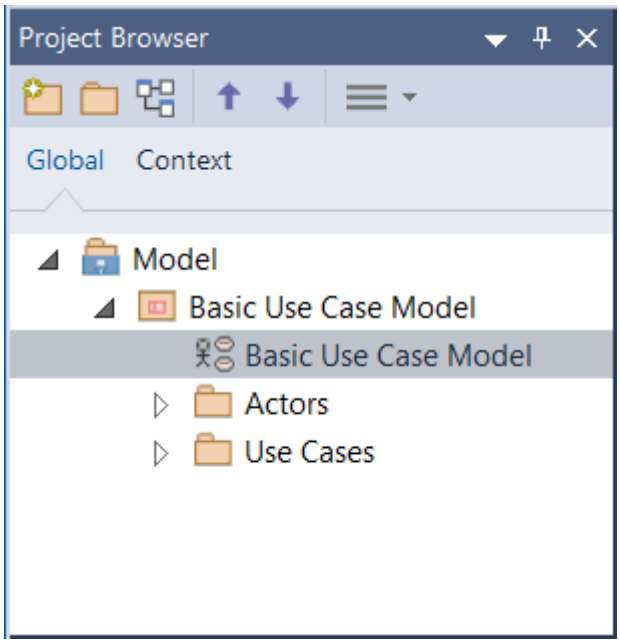
Treba imati na umu da desna strana ima neke vrlo korisne informacije o uzorku, uključujući sliku koja pokazuje što će se dobiti, raspravu o tome kada se treba koristiti ovaj uzorak, dodatne reference i drugi materijal. Iako su neki pojmovi i reference možda nepoznati u ovoj fazi, bilo bi dobro pročitati ih, kako bi se upoznalo s onim što se dodaje novom modelu.

Kada se završi s čitanjem, treba kliknuti gumb **Izradi uzorak (a)** 

koji se nalazi u donjem lijevom kutu čarobnjaka za model. Enterprise Architect sada će stvoriti nove elemente modela, dijagrame i pakete, kako bi se napravila kopija ovog uzorka, da bi se moglo prilagoditi i igrati.

13.3.Project Browser

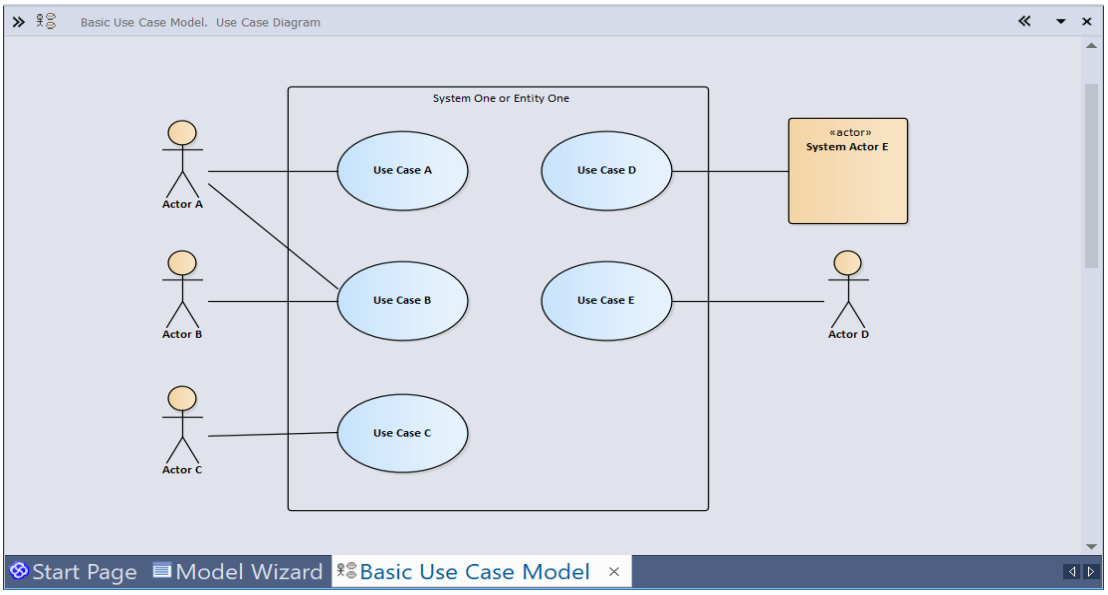
Sada, kada je stvoren naš prvi model sadržaja, treba vidjeti što se dodalo. Da bi se to učinilo, treba obratiti pozornost na preglednik projekata.



Slika 45. Stvaranje prvog modela sadržaja

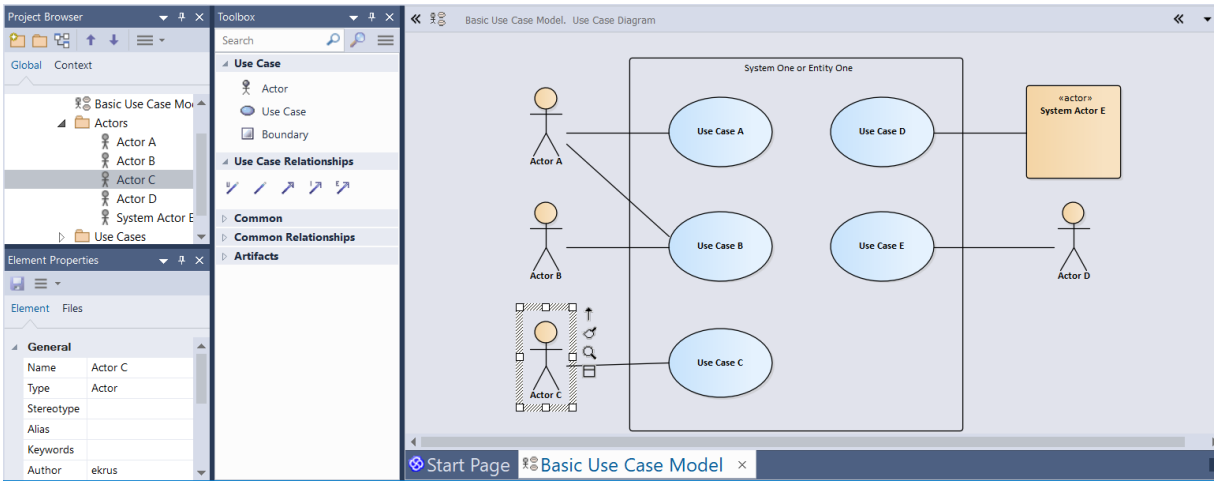
Kao što se može vidjeti na slici 45., u model je dodan novi sadržaj. To uključuje paket i dijagram na najvišoj razini, te neke slučajeve aktera i korištenja (unutar paketa "Sudionik - Actors" i "Slučaj korištenja - Use Case").

Potrebno je dva puta kliknuti na vezu dijagrama "Osnovni model slučaja korištenja", kako bi se pojavio grafički prikaz onoga što već postoji. Trebao bi se pokazati dijagram na slici 46.



Slika 46. Grafički model 'Osnovnog modela slučaja korištenja'

U ovom dijagramu može se vidjeti nekoliko sudionika i slučajeva korištenja, ako se klikne na jedan od sudionika, a zatim pritisne **Alt + G** , ili desnom tipkom miša i odabere 'Pronađi'. U pregledniku projekata iz kontekstnog izbornika, paket 'Sudionici' je proširen i element odabran u dijagramu, bit će odabran u pregledniku projekata, kao što je prikazano na slici 47.



Slika 47. Primjer odabranog elementa u dijagramu

Kao što se može vidjeti, u pregledniku projekata nalazi se element na razini modela, koji ima grafički prikaz u dijagramu. Napominje se kako se jedan element modela može po želji smjestiti u više dijagrama, te da će se promjena imena ili bilješki ili drugih svojstava elemenata odmah odraziti na svim dijagramima, u kojima je element postavljen.

14. Rad s dijagramima

Dijagrami su vizualni prikaz modela i osiguravaju jedan ili više prozora u strukturi i ponašanju modela. Učenje rada s dijagramima bitan je dio modeliranja s Enterprise Architectom. Prikazat će se neke od najosnovnijih načina korištenja dijagrama za pregled i izgradnju modela.

Otvaranje dijagrama

Ako se želi pregledati ili raditi s postojećim dijagramom, može ga se otvoriti dvostrukim klikom na njegovu ikonu, u **pregledniku projekata** . Za ovaj kratki uvod, trebalo bi pronaći dijagram Osnovni model slučaja upotrebe, koji je stvoren ranije i dvaput kliknuti na njegovu ikonu, da bi se otvorio. Ako je već otvoren, bit će odabran i postati prozor fokusa.

Odabir i pomicanje elemenata

Sada kada je dijagram otvoren, potrebno je kliknuti na elemente, da bi ih se odabralo. Nakon odabira, može se kliknuti i povući za pomicanje, desnim klikom odabrati radnje iz kontekstnog izbornika, promatrati ili raditi sa svojstvima elementa u različitim prozorima svojstava i / ili mijenjati izgled elementa i druge vizualne karakteristike. Važno je znati da se također može kliknuti na povezivanje veza između elemenata i isto tako urediti i pregledati svojstva, izgled i pozicioniranje.

Vrpca rasporeda

Treća kategorija vrpce je 'Layout' i glavna je vrpca namijenjena manipuliranju izgledom dijagrama. Postoje naredbe za promjenu tema dijagrama, postavljanje izgleda elemenata, poravnavanje i oblikovanje elemenata, dijagrama izgleda i sadržaja dijagrama filtra. Mnogi od tih alata detaljno su opisani u opciji pomoć (kartica START - Help). Kako bi se najbolje iskoristili dijagrami, poželjno je odvojiti vrijeme za čitanje o dostupnim alatima i radnjama.

Osnovno uređivanje

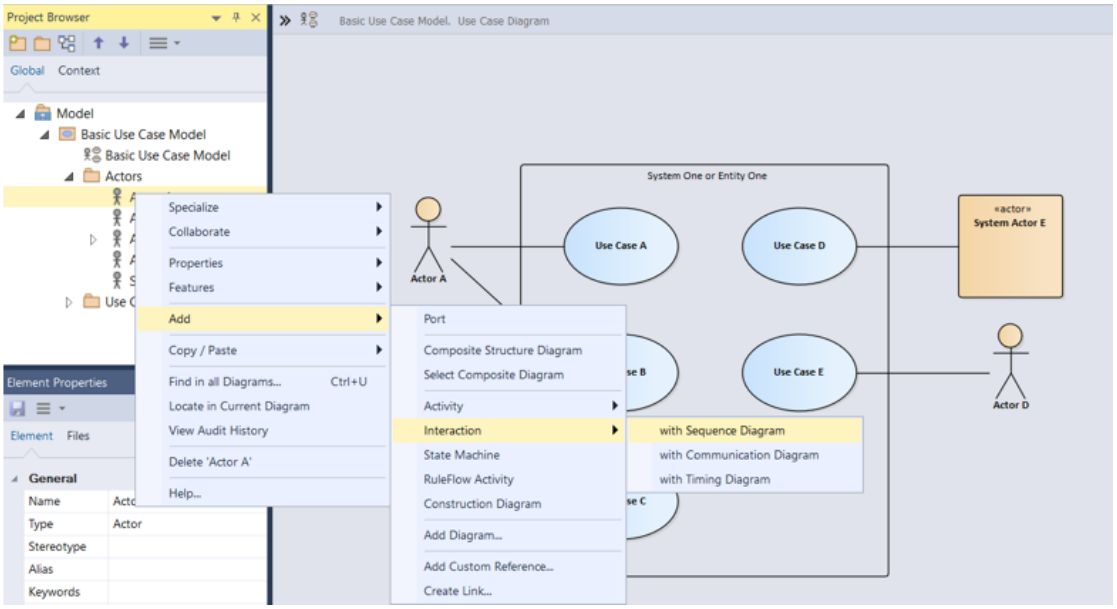
Dodavanje dijagrama scenarija slučaju korištenja

Sada, kada su dijagrami usvojeni, može se dodati novi i izgraditi jednostavan dijagram sekvenci koji pokazuje tijek akcija i odgovora između sudionika i slučaja korištenja. Prvo

se mora odabrati slučaj korištenja u pregledniku projekata, koji će sadržavati novi dijagram sekvence (ili interakcija).

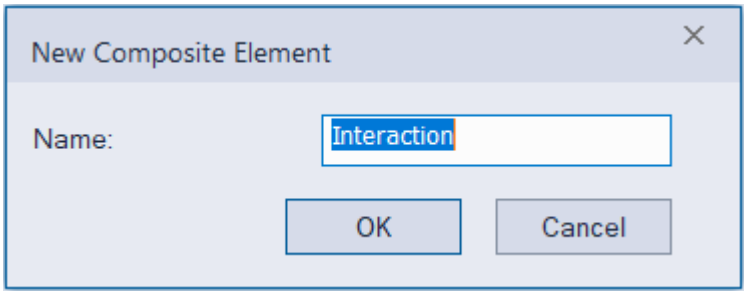
Stvaranje dijagrama

Za početak, treba kliknuti na "Koristi slučaj A" u pregledniku projekata, da bi ga se odabralo. Sada desnom tipkom miša treba kliknuti na njega, da bi se prikazao kontekstni izbornik i odabrao 'Dodaj - Add| Interakcija | i **dijagramom slijeda** '. Potrebno je imenovati novu interakciju i kliknuti **gumb 'OK'** .



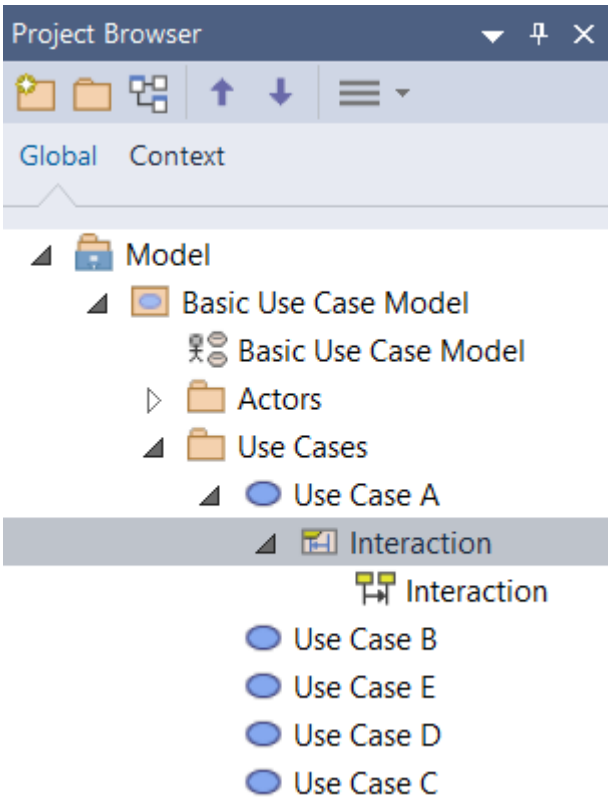
Slika 48. Dodavanje dijagrama scenarija slučaju korištenja

Nakon odabira dobit će se 'New composite Element'.



Slika 49. New composite element

Nakon toga, treba kliknuti na gumb 'OK'.

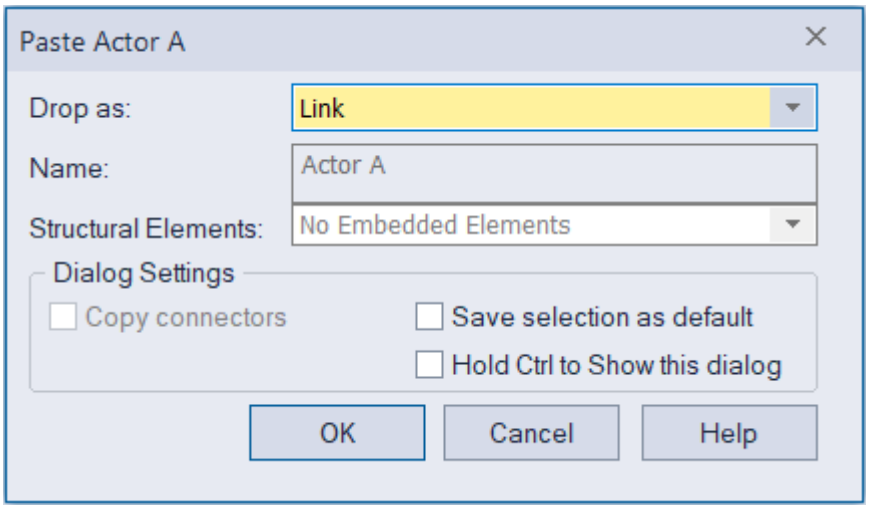


Slika 50. Preglednik projekta - Interaction

Treba podsjetiti da postoji novi spremnik Interakcija, s praznim dijagramom slijeda. Dvaput kliknuti na ikonu dijagrama sekvence (Interaction), da bi se to učinio novim zadanim radnim prostorom. Tako će se dobiti prazan radni prostor dijagrama a alatni okvir će se promijeniti, kako bi se prikazali elementi koji se obično koriste u dijagramima sekvenci. Prvo, iskoristit će se model i postaviti link na relevantnog sudionika na dijagram.

U originalnom dijagramu, **Use Case A** je bio u odnosu prema **sudioniku A**, pa je potrebno pronaći sudionika A u pregledniku projekata i povući ga i ispustiti na novi dijagram.

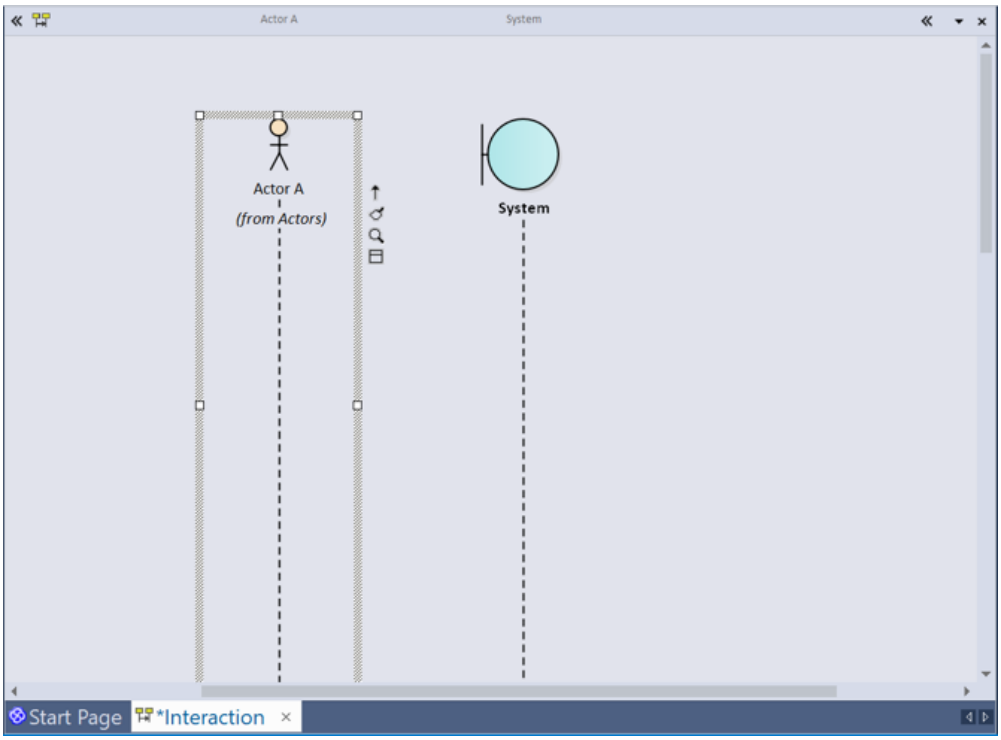
U dijaloškom okviru, koji se pojavi, treba odabrati "Veza", kao vrstu potrebnog položaja i kliknite gumb Uredu (OK).



Slika 51. Nova vizualna reprezentacija sudionika A

Na taj način je stvorena i postavljena nova vizualna reprezentacija sudionika A, na dijagram sekvence, kao Lifeline.

Zatim, mora se dodati granični element, da bi se predstavio sustav koji se modelira i način kako će komunicirati s našim sudionikom. Na **alatnoj** traci treba kliknuti **ikonu Boundary** (**Granica),** a zatim kliknuti desno od životne linije za sudionika. Sada bi se trebao prikazati dijagram kao na slici 52.

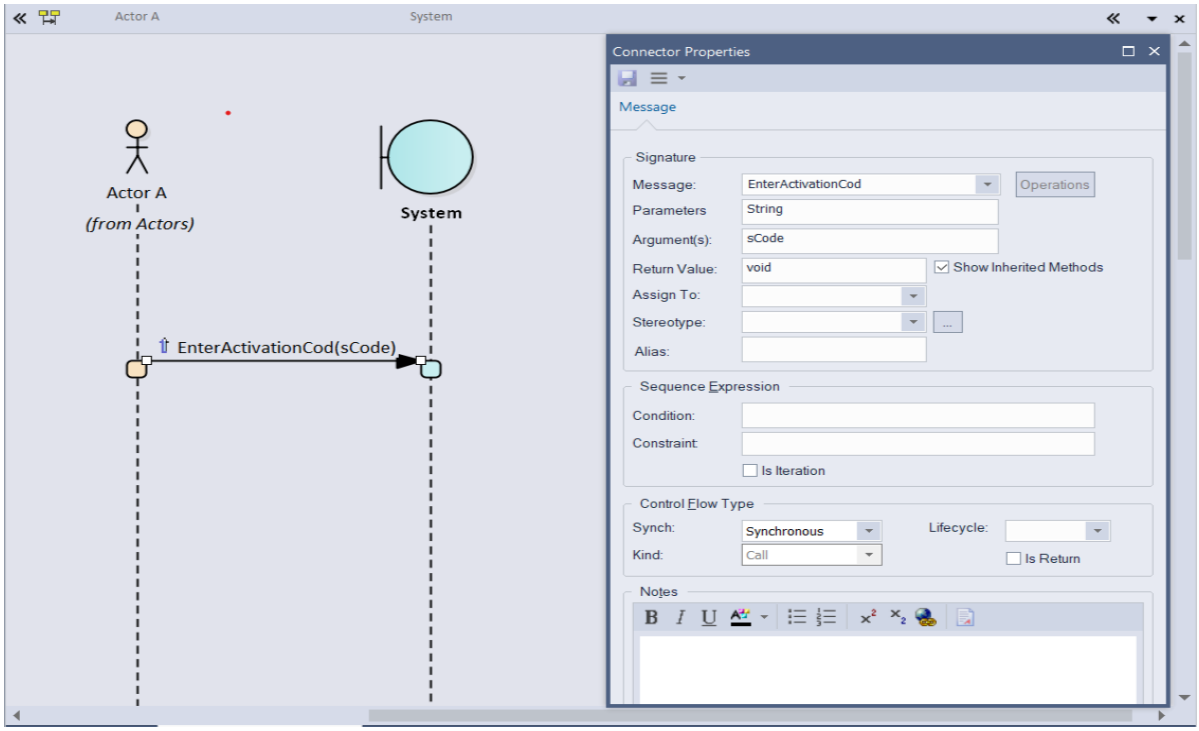


Slika 52. Dodavanje graničnog elementa

Sada se dobio sudionik i sustav, ali želi se predstaviti i tijek akcije i odgovora između dva entiteta. Tijek akcije i odgovora može se modelirati, koristeći poruke između njih.

Treba kliknuti natrag na Ljestvicu sudionika A i potražiti gumb Quicklink - ovo je mala 'strelica gore' koja se nalazi na desnom rubu odabranog elementa i vidljiva je samo kada je element odabran.

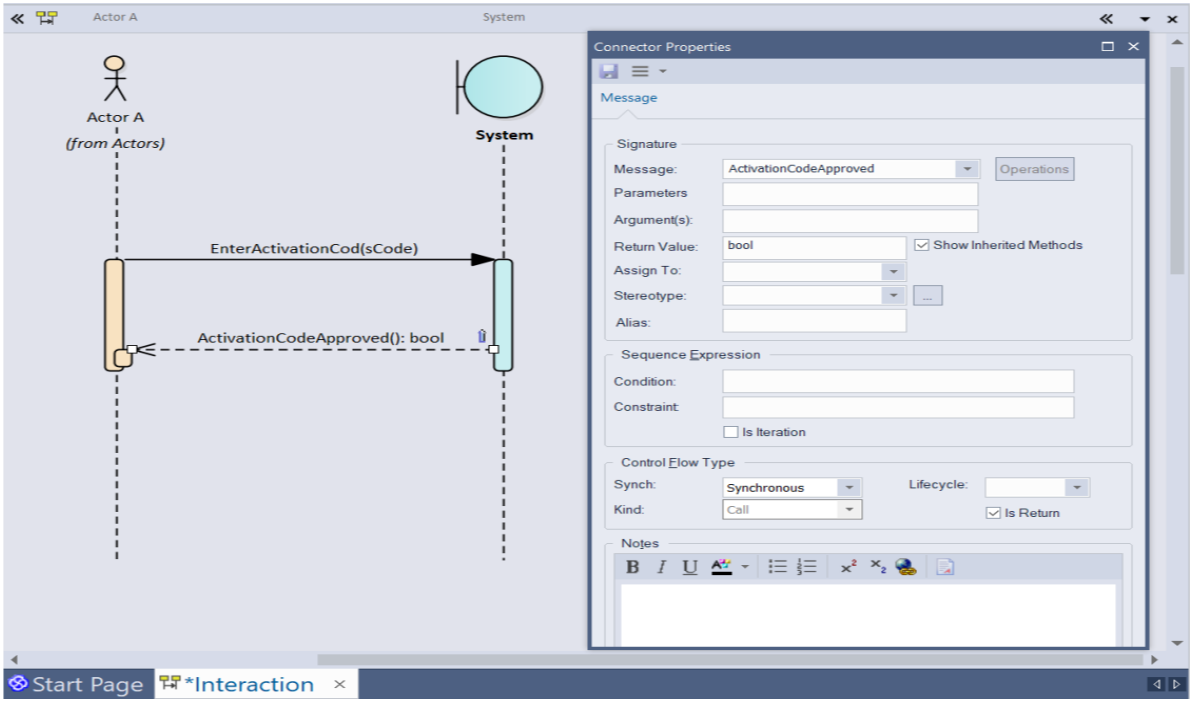
Sada treba kliknuti i držati gumb Quicklink (↑), a zatim povući do Lifeline sustava i otpustiti. Ovaj postupak stvara novu poruku u modelu, između sudionika A i sustava. Treba odabrati novu poruku na dijagramu i imati na umu da se **prozor Svojstva** prebacuje na 'Svojstva konektora', omogućujući uređivanje detalja novog konektora. Nakon što se unese nekoliko pojedinosti za poruku i klikne **gumb Spremi**, radni prostor dijagrama trebao bi izgledati kao na slici 53.



Slika 53. Veza između dva entiteta

Povratna poruka može se stvoriti na sličan način, tako da se klikne na System, a zatim povuče gumb Quicklink preko u Actor A Lifeline. Potrebno je odabrati novu poruku na dijagramu i ispunite pojedinosti u prozoru Connector Properties. U odjeljku "Kontrola

protoka (Control Flow Type)" u prozoru Svojstva treba odabrati opciju "Is Return". Dijagram bi sada bi trebao prikazati povratnu poruku, kao na slici 54.



Slika 54. Prikaz povratne poruke

15. Izgradnja modela – osnovne vještine

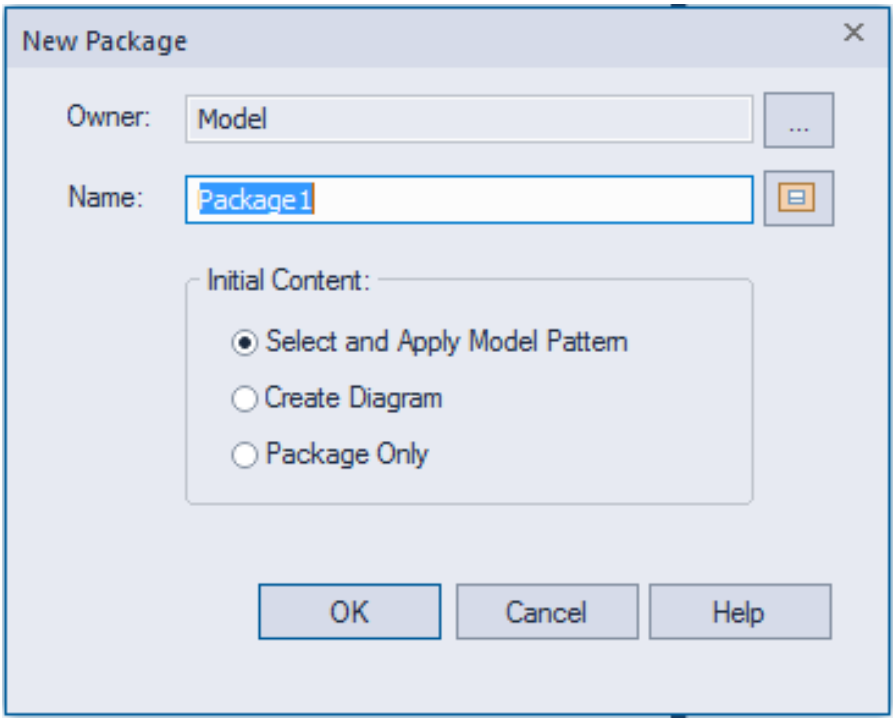
Vrpca dizajna

Traka za dizajn grupira zajedno naredbe koje se koriste za kreiranje i modificiranje stavki modela, kao što su elementi (Element), dijagrami (Diagram), paketi (Package) i sl. To omogućuje pristup brojnim prozorima, poput **prozoru Svojstva**, **Mogućnosti prozora**, kao i naredbe za pregledavanje sadržaja paketa i dijagram sadržaj, u obliku popisa ili prikaza karte.

Prije stvaranja novih elemenata i konektora na dijagramu, koristit će se vrpca Design za izvođenje nekih osnovnih zadataka modeliranja, kao što je kreiranje novog paketa i kreiranje novog dijagrama unutar paketa. U ovom primjeru stvorit će se jednostavan model klase.

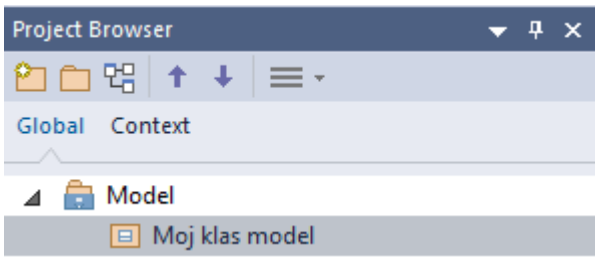
Izrada paketa

Ako se klikne na korijenski model modela u **pregledniku projekata, a** zatim na panelu "Paket" dizajnerske vrpce, treba odaberati "Umetni (Insert) > Umetni jednostavan paket (Insert a Simple Package)", stvorit će se novi paket pod korijenskim čvorom modela. Otvorit će se dijaloški okvir "Novi paket (New Package)". Ovdje se može unijeti naziv za novi paket i može se odabrati samo stvaranje paketa, kreirati paket s dijagramom ili odabrati uzorak modela za umetanje u novi paket, slika 55.



Slika 55. Stvaranje novog paketa

Za ovaj pregled, treba odaberati 'Samo paket', unijeti naziv paketa, kao što je 'Moj klas model – My class model', a zatim kliknuti **gumb 'OK'** . Trebao bi se vidjeti novi paket u pregledniku projekata, kao dijete korijenskog čvora modela.

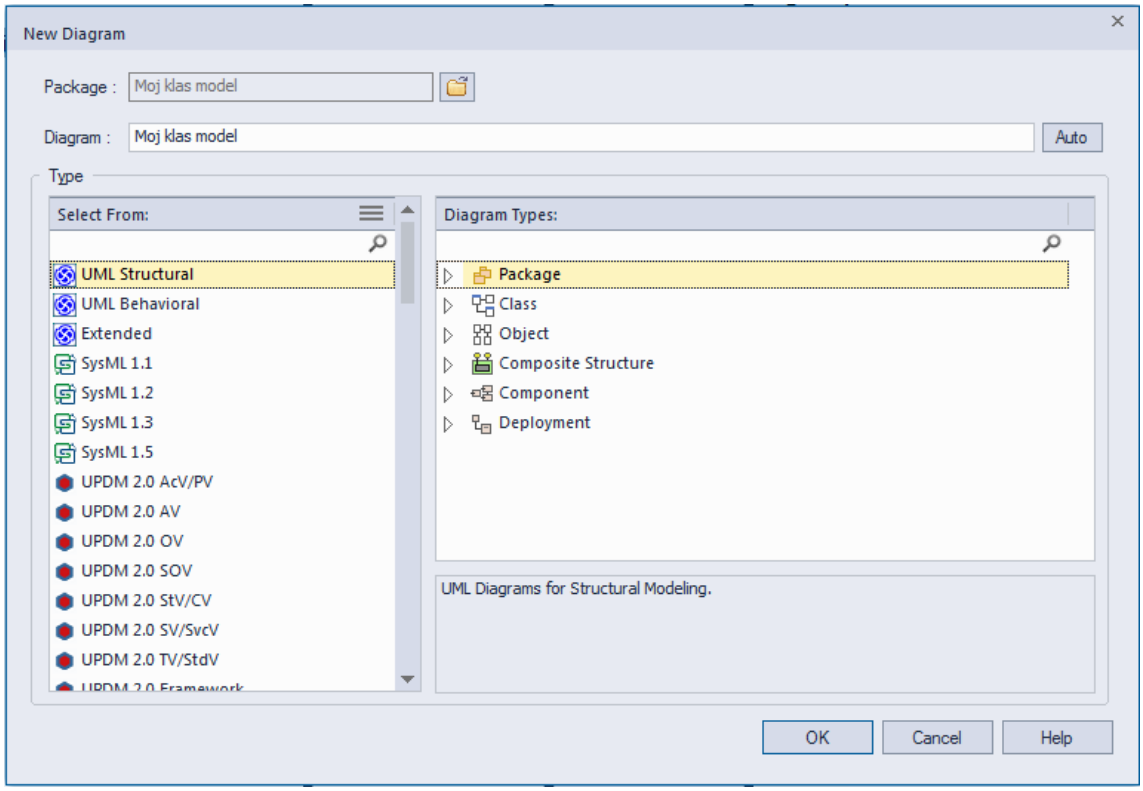


Slika 56. Moj klas model

Izrada dijagrama

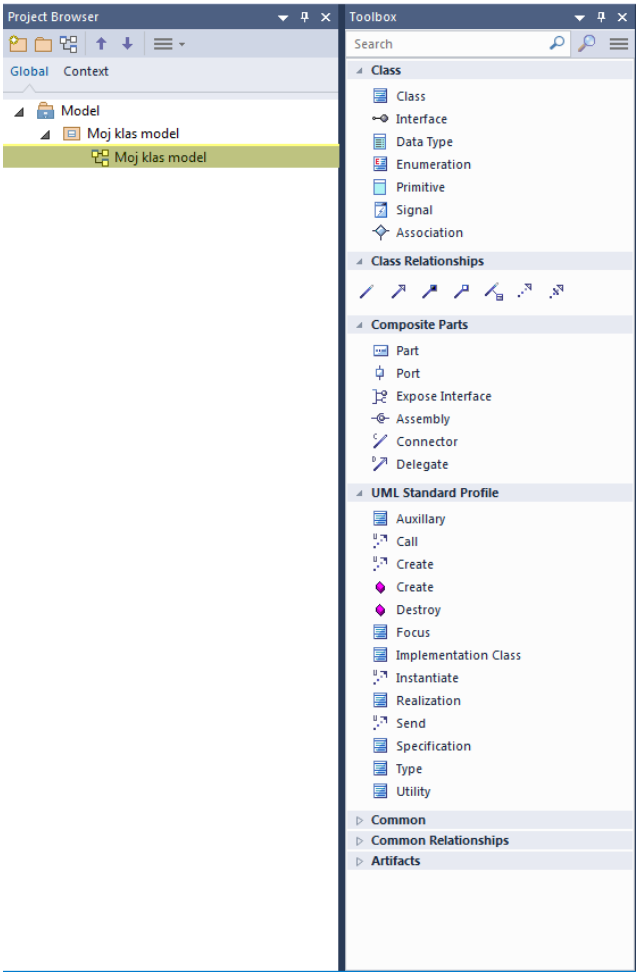
Sada sljedi izrada dijagrama klase unutar novog paketa.

Potrebno je kliknuti na Paket u pregledniku projekata, a zatim na ploči Dizajn, vrpce Dijagram' odabrati 'Umetni (insert)'. Pojavit će se dijalog "Novi dijagram (New Diagram)".



Slika 57. Izrada dijagrama klase unutar novog paketa

Zadani naziv za dijagram je isto ime kao i njegov nadređeni paket; treba ostaviti tako kako piše. U odjeljku "Vrsta (Select From)" treba odabrati "UML Structural" i vrstu dijagrama (Diagram Types) "Klasa (Class)". Kliknuti gumb 'U redu (**OK)**'. Novi dijagram klase pojavit će se u pregledniku projekata i otvorit će se u radnom prostoru glavnog dijagrama.



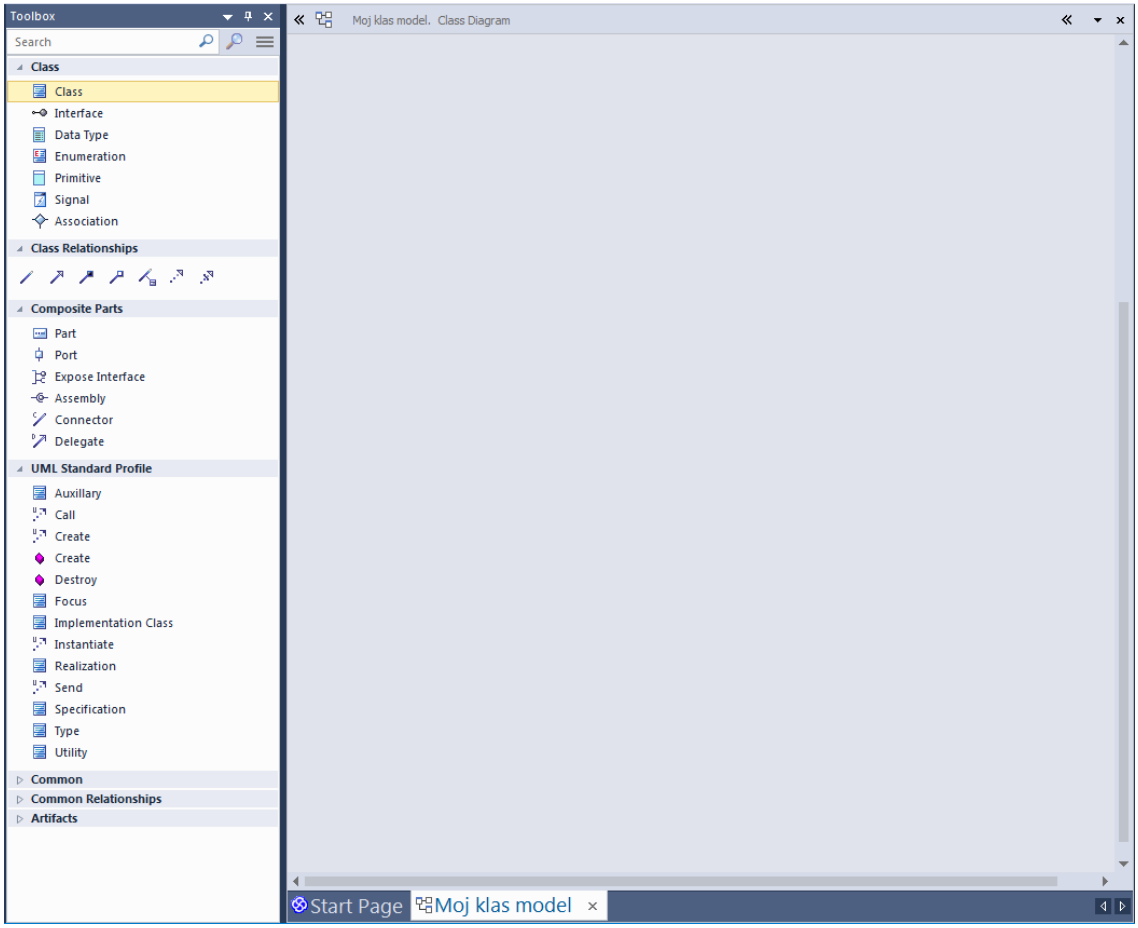
Slika 58. Stvaranje novog dijagrama klase

Kutija s alatima dijagrama

Kutija s alatima dijagramima (Toolbox) sadrži paletu elemenata i konektora koji se mogu odabrati i ispustiti na trenutno otvoren dijagram. Na taj način se stvara novi element ili priključak odabranog tipa, unutar modela i prikazuje na dijagramu, gdje je ispušten. Alatni okvir prikazuje skup elemenata i konektora, koji se obično koriste s trenutnim tipom dijagrama. Dok se otvaraju dijagrami različitih vrsta, ovaj skup se mijenja.

Sada se treba otvoriti alatnu traku dijagrama klikom na gumb 'Alatni okvir (Toolbox)', na grupu naredbi 'Dijagram', na vrpci Dizajn. Alatni okvir prikazan je na lijevoj strani radnog prostora dijagrama.

Slika 59. prikazuje novi, prazan dijagram klase i pored njega alatni okvir, koji prikazuje elemente i konektore prikladne za rad s dijagramima klase.

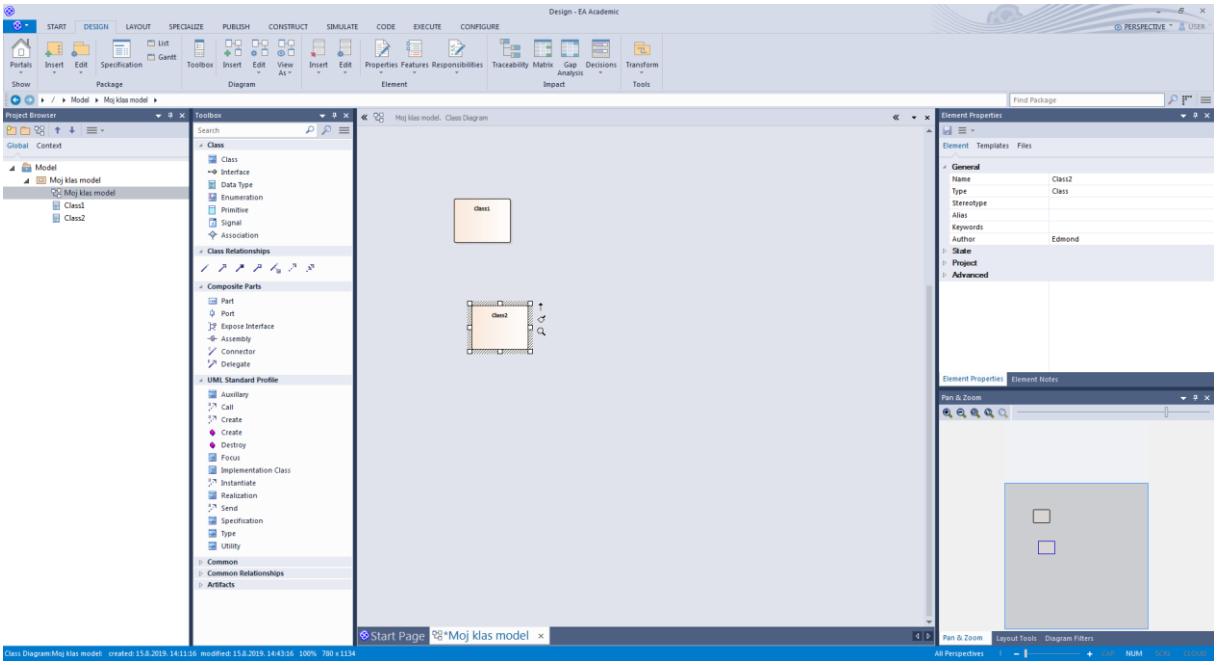


Slika 59. Radna površina i kutija s alatima za izradu dijagrama klasa

Dodavanje elemenata dijagramu klase

Pomoću alata za dijagrame, dodaju se elementi u dijagram klase.

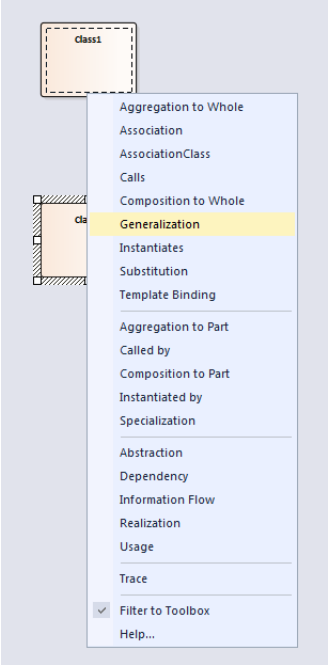
Potrebno je odabrati element Class u Toolboxu, a zatim kliknuti na prazan dijagram. Nova klasa pod nazivom 'Class1' je stvorena u modelu i postavljena na otvoreni dijagram. Naziv klase 'Class1' primijenjen je prema zadanim postavkama. Može se unijeti novo ime u Klasu na dijagramu, a kasnije i promijeniti naziv, pomoću prozora Svojstva elementa. Druga klasa na dijagramu može se napraviti tako da se ponovno odabere 'Class' u alatu i povuče na dijagram. Radni prostor sada bi trebao izgledati kao na slici 60.



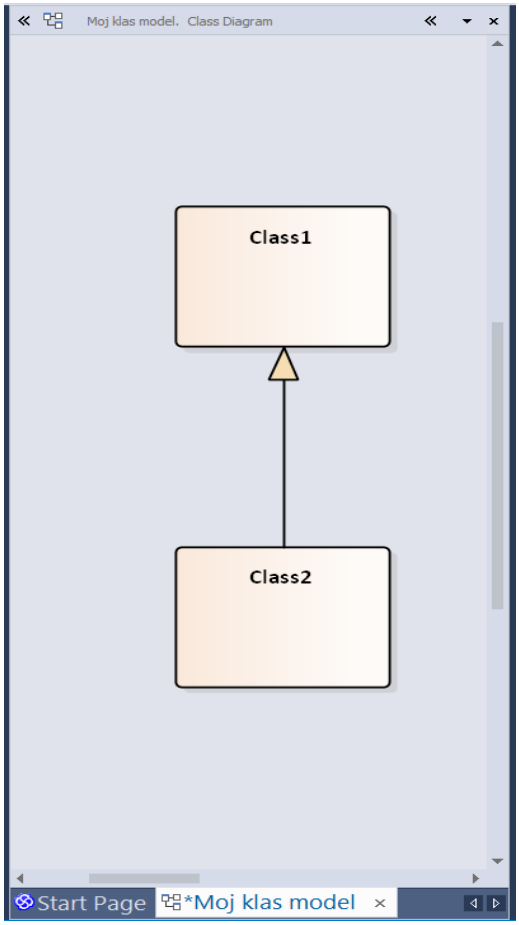
Slika 60. Dodavanje elemenata dijagramu klase

Da bi se stvorila veza između dva elementa na dijagramu, jednostavno se odabere vrsta odnosa u alatu Diagram Toolbox, a zatim povuče linija iz jednog elementa na dijagramu, u drugi. Većina odnosa je usmjerena, pa se povlačenje veze iz klase 1 u klasu 2 razlikuje od povlačenja veze iz klase 2 u klasu1.

Treba kliknuti na opciju Generalization u Opciji alata za dijagrame, a zatim na dijagramu povući liniju iz Class2 u Class1.



Stvoreni odnos ukazuje da je Class1 generalizacija Class2.

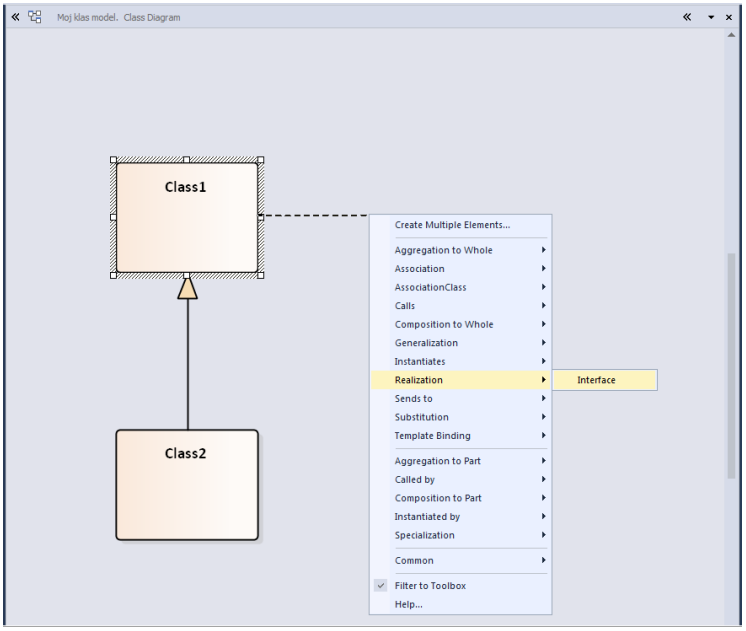


Slika 61. Prikaz stvorenog odnosa koji ukazuje da je Class1 generalizacija Class2

Dodavanje elemenata pomoću alata za brzo spajanje

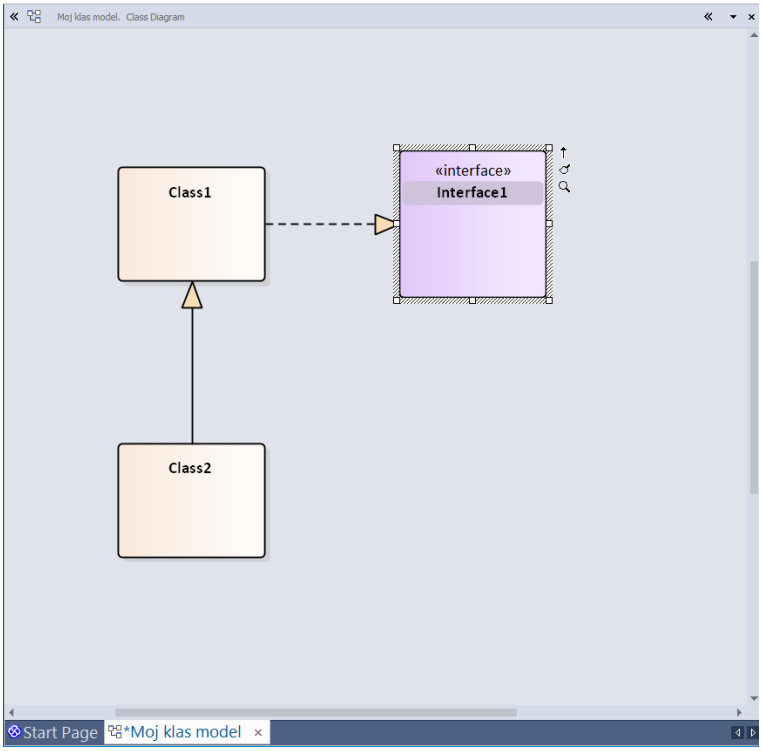
Ukratko je obrađen Quicklinker (↑) u *radu s dijagramima*, gdje su stvorene veze između dva elementa u obliku poruka Interakcija. Sada ćese koristiti Quicklinker (↑) za stvaranje novog elementa u isto vrijeme, kao i stvaranje novog odnosa iz postojećeg elementa.

U dijagramu 'Moj klas model' treba kliknuti na Class1, da bi ga se odabraoi. Strelica Quicklinker (↑) trebala bi biti vidljiva pokraj gornjeg desnog kuta Class1. Treba kliknuti i povući strelicu Quicklinker (↑) na desno, otpustiti tipku miša u praznom prostoru na dijagramu. Ova radnja prikazuje skočni izbornik koji traži da se odabere vrsta odnosa, a zatim vrsta elementa. Iz izbornika najprije treba odabrati 'Realizacija (Realization)' a nakon toga, 'Sučelje (Interface)'.



Slika 62. Odabir vrste odnosa i vrstu elemenata

Rezultirajući konektor i element prikazani su na slici 63.



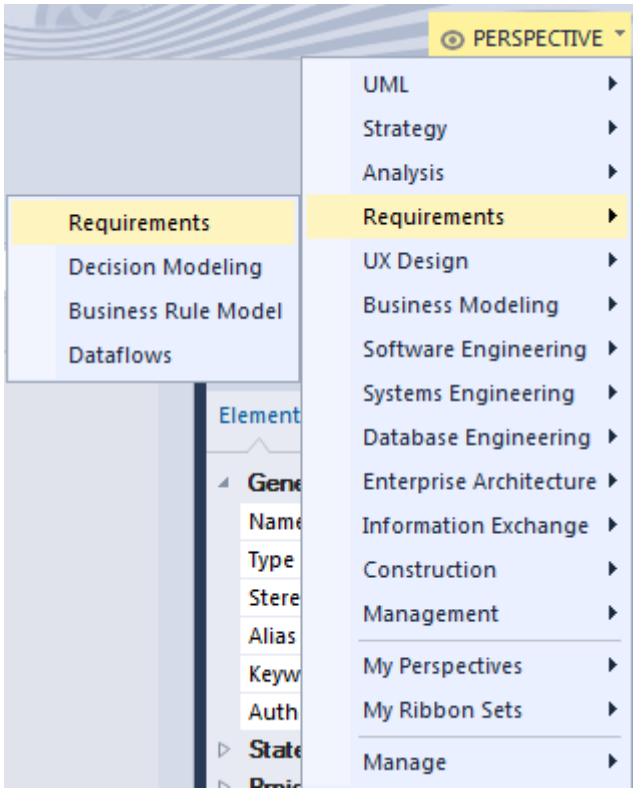
Slika 63. Rezultirajući konektor i element

15.1. Korištenje perspektiva i uzoraka modela

Enterprise Architect nudi širok raspon različitih perspektiva, što odgovara raznim vrstama modeliranja koje možete izvesti. Ranije je spomenuto kako je perspektiva ograničenje postavljena na korisničko sučelje, samo za prikaz određene tehnologije ili skupa tehnologija. Koristi se za smanjenje složenosti i pomaže za fokusiranje na zadatak.

Dodavanje modela zahtjeva

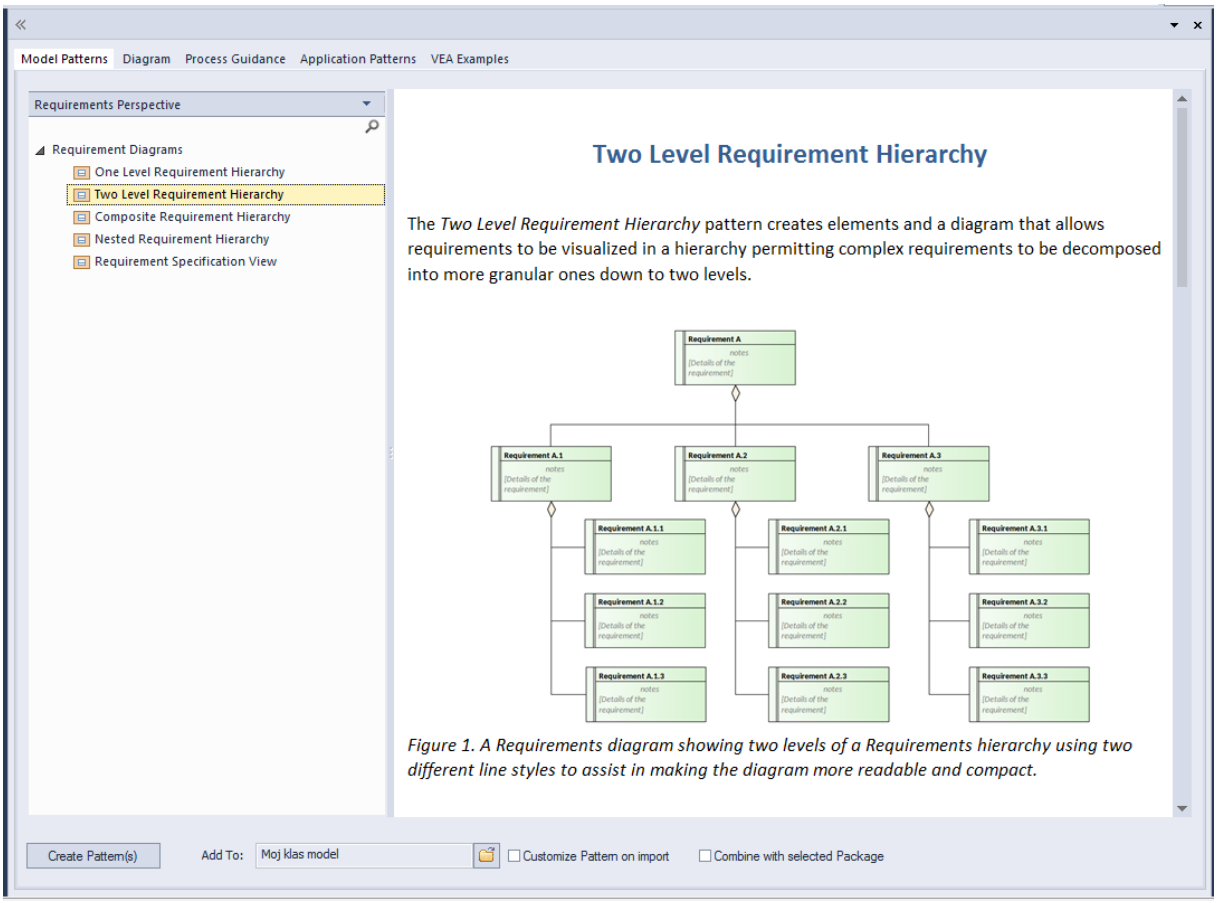
Prikazuje se izgradnja jednostavnog modela zahtjeva. Prvo se odabere jedna od perspektiva zahtjeva. Za primjer, iz izbornika Perspektive, treba odabrati 'Zahtjevi (Requirements) | Zahtjevi (Requirements).



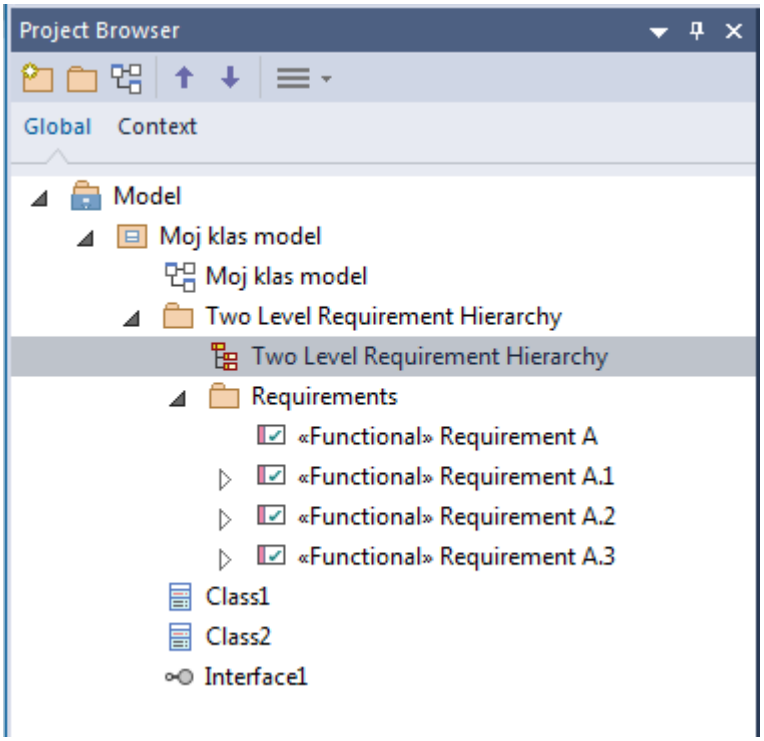
Slika 64. Izgradnja modela zahtjeva

Nakon odabira Perspektive zahtjeva, **Čarobnjak modela** se prikazuje u glavnom području radnog prostora, nudeći samo izbor osnovnih obrazaca modela Zahtjeva. Najprije treba odabrati gdje će se u modelu dodati novi sadržaj. Treba kliknuti na ikonu mape, pored polja "Dodaj u:" na dnu stranice čarobnjaka modela, zatim odabrati korijenski model modela i kliknuti gumb U redu. Nakon toga, odabrati uzorak 'Hijerarhija zahtjeva na dvije razine (Two Level Requirement Hierarchy)' i kliknuti **gumb Stvori uzorak** 

.



Odabrani uzorak modela dodaje se našem modelu u korijenski model modela.

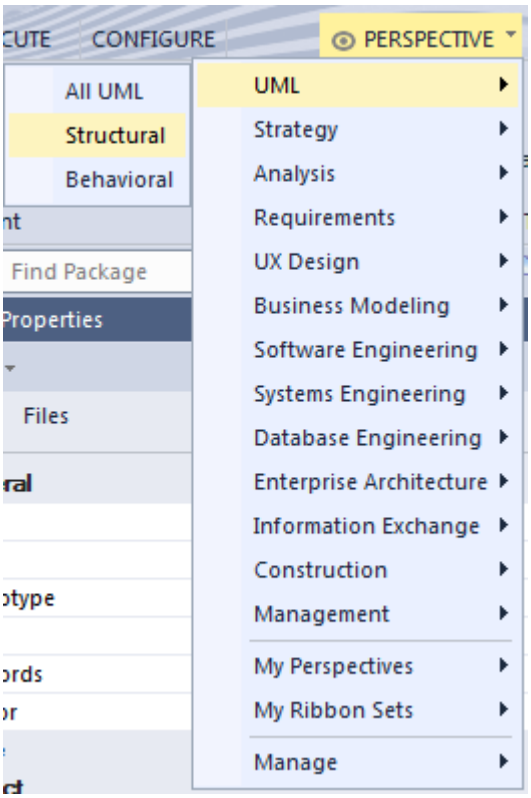


Slika 65. Dodavanje uzorka u korijenskom modelu modela

Da bi se otvorio dijagram u radnom prostoru glavnog dijagrama, potrebno je dvaput kliknuti na poveznicu dijagrama u pregledniku projekata . Nakon toga može se odabrati elemente na dijagramu (ili u pregledniku projekata) i koristiti prozor Svojstva elementa, da bi ih se prilagodilo vlastitoj upotrebi, preimenovanjem, unosom opisa u **prozoru Napomene** ili postavljanjem drugih svojstava prema potrebi.

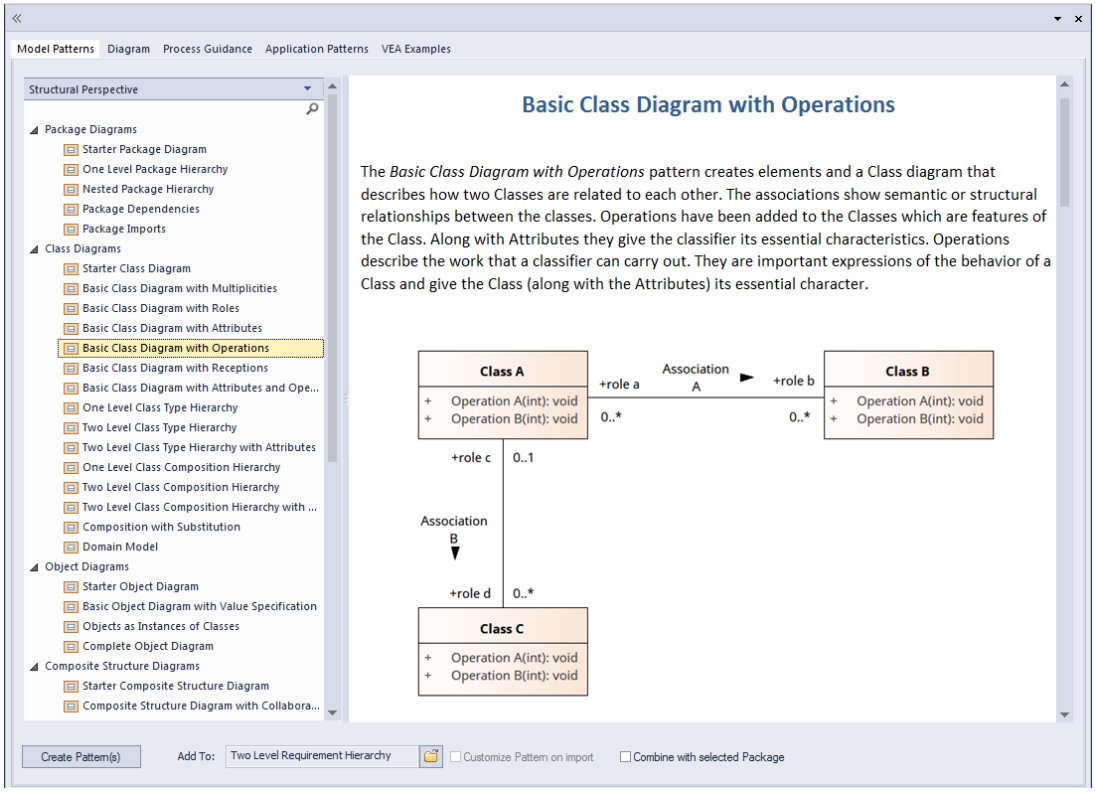
Dodavanje modela klase

U projekt se dodaje jednostavan model klase, slijedeći proceduru sličnu onoj koja je upravo korištena. U izborniku Perspektive treba odabrati 'UML | Strukturna".



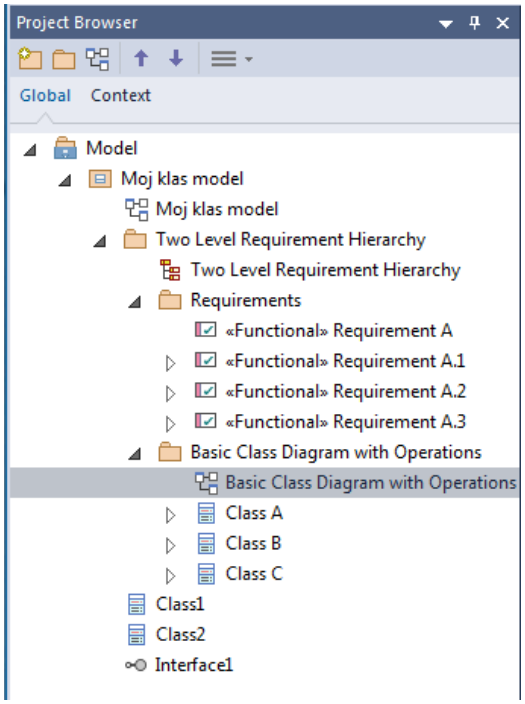
Slika 66. Odabir iz izbornika 'Perspektive'

Prikazuje se čarobnjak za model koji prikazuje strukturalnu perspektivu. Potrebno je pomicati se kroz popis, da bi se pronašao dijagram klase, a zatim kliknuti na 'Osnovni **dijagram klase** s operacijama'. Treba kliknuti ikonu mape pored polja 'Dodaj u:' i odabrati korijenski čvor modela kao mjesto na kojemu se može dodati sadržaj novog modela. Na kraju, kliknuti gumb Stvori uzorak, kako bi se projektu dodao sadržaj novog modela.



Slika 67. Basic Class Diagram with Operations

Odabrani uzorak modela dodaje se vašem projektu; Paketi, dijagrami i elementi prikazani su u pregledniku projekata.

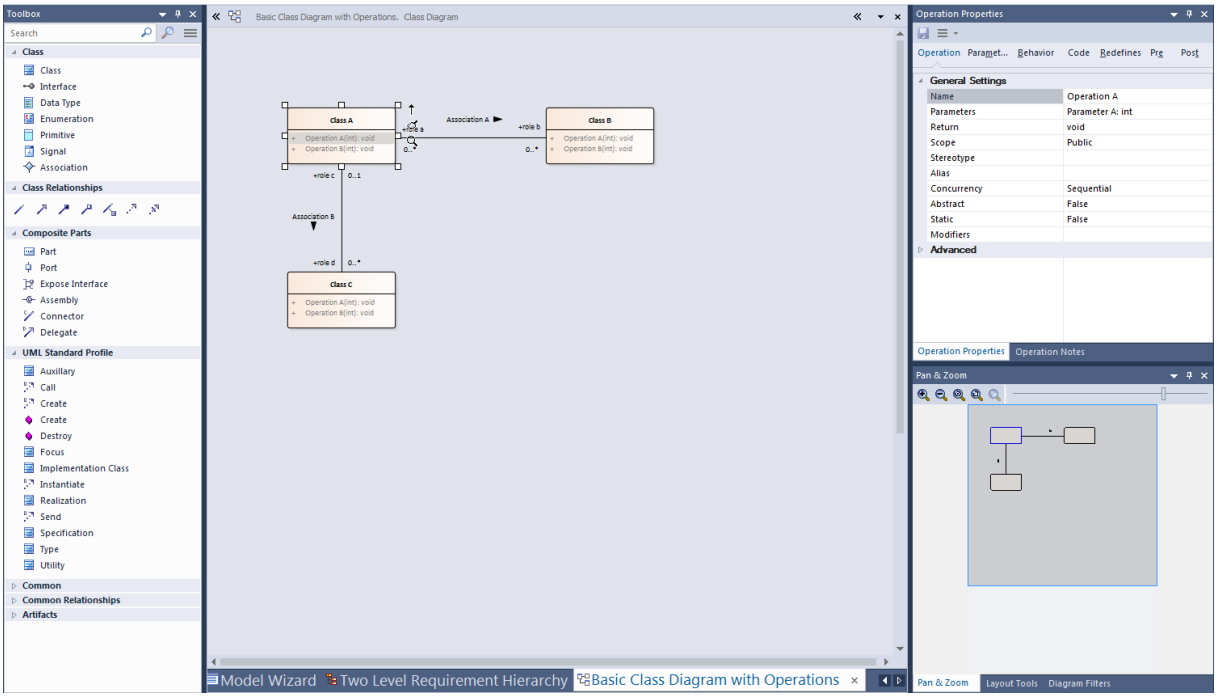


Slika 68. Prikaz dodanog uzorka modela vašem projektu

Potrebno je dvaput kliknuti na dijagramsku vezu 'Osnovni dijagram klase s radom' u pregledniku projekata. Time se otvara dijagram u glavnom radnom prostoru. Sada će se koristiti prozor Mogućnosti za pregled operacija klase A. na ploči "Istraživanje", na traci Start odabrati stavku izbornika 'Svojstva> Značajke'. Prikazuje se prozor Mogućnosti; treba odabrati karticu "Operacije" u ovom prozoru. Nakon toga, kliknuti na klasu A u dijagramu (ili u pregledniku projekata). Detalji o njegovom radu prikazuju se u prozoru, čim se odabere klasa A, Svojstva. Ovdje se može izmijeniti postojeća operacija promjenom njezina imena, povratnog tipa ili čak popisa parametara. Kada se odabere operacija na ovom popisu, prozor Properties (Svojstva) prelazi u Operation Properties (Svojstva operacije) i prikazuje detalje odabrane operacije.

Također, može se koristiti prozor Mogućnosti da bi se definirale nove operacije, jednostavnim upisivanjem imena na popis na kojem piše " *Nova operacija ...* ".

Ova slika prikazuje prozor Značajke kada je na dijagramu odabrana klasa A i prozor Svojstva operacije, kada je u prozoru Svojstva odabran rad B.



Slika 69. Prikaz prozora 'Značajke'

Prikazane su neke jednostavne operacije modeliranja, kako bi se dobilo osnovno razumijevanje o tome na koji način se mogu koristiti perspektive i obrasci modela za stvaranje struktura modela, koje se mogu koristiti kao početna točka za stvaranje vlastitih modela.

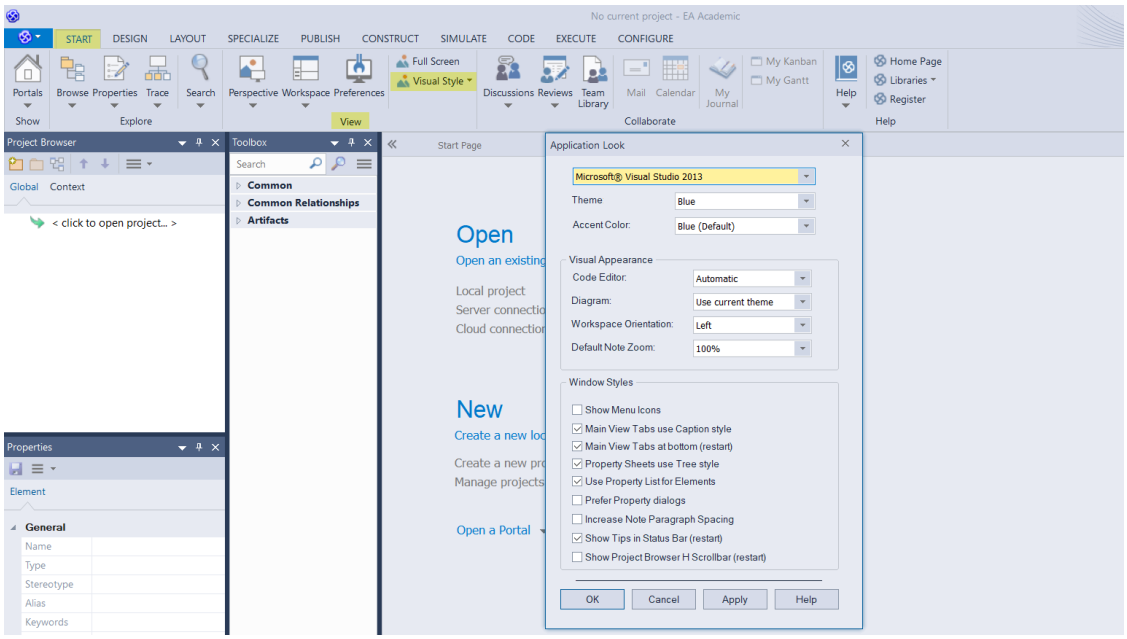
Također je demonstrirano korištenje Project Browser-a za navigaciju kroz model, koristeći Diagram Toolbox za dodavanje novog sadržaja svakom pojedinom modelu ili modelima putem dijagrama i korištenje različitih prozora Properties za pregled i modificiranje svojstava različitih elemenata modela.

16. Prilagodba radnog prostora

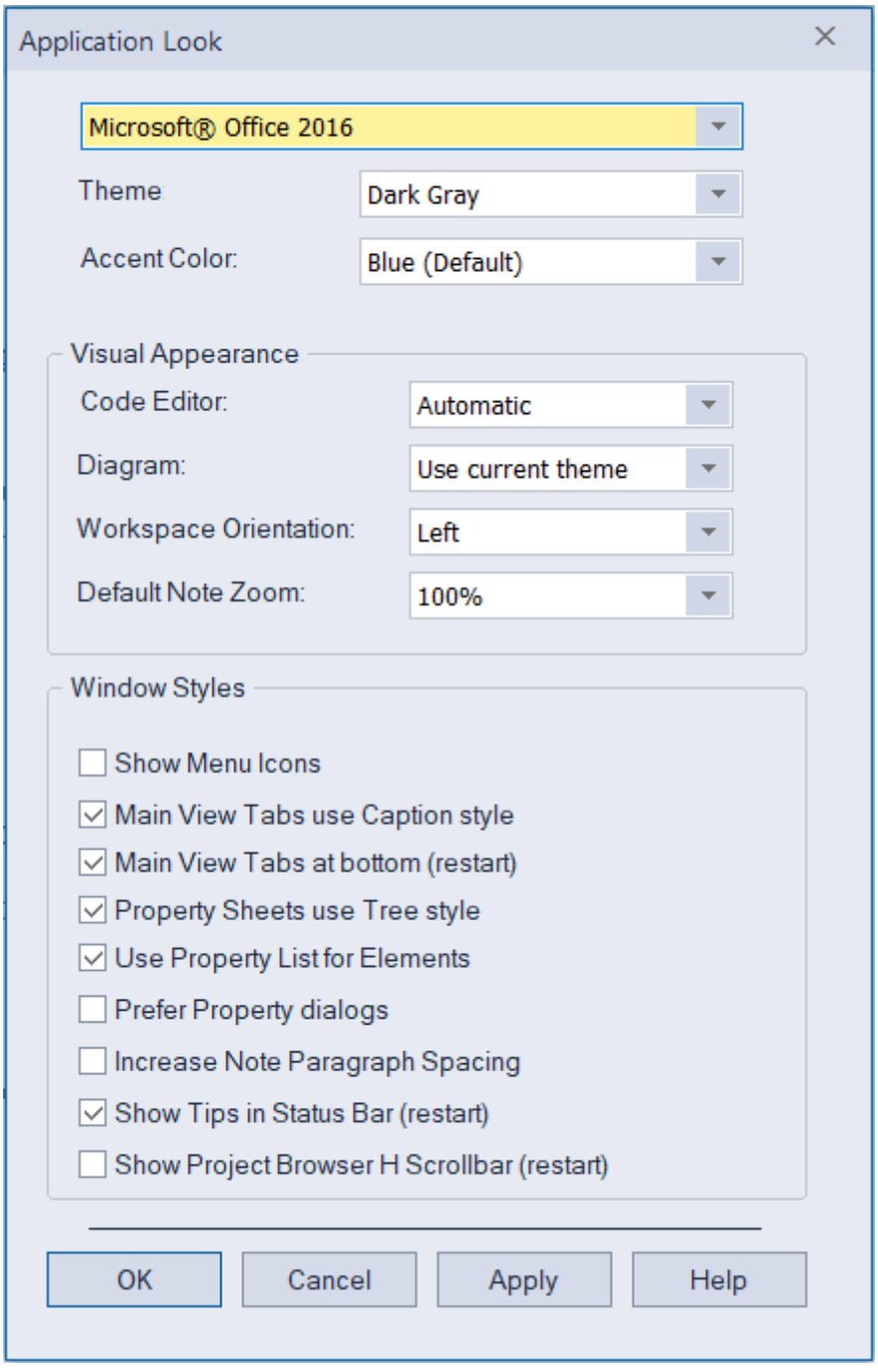
Kada se prvi put otvori Enterprise Architect, vidi se zadani izgled korisničkog sučelja. Izgled sučelja može se promijeniti na mnogo načina, skrivati prozore koji nisu potrebni, organizirati dijaloge u najprikladnijoj konfiguraciji, mijenjati boje zaslona i tekstove, i još mnogo toga, kako bi izgled odgovarao konkretnim potrebama, osobnim preferencijama i zadacima koji se obavljaju.

16.1.Promjena vizualnog stila

Ilustracije korisničkog sučelja koje se vide u vodiču, imaju određenu boju i izgled - ili vizualni stil. Ovaj stil može se promijeniti u jednu od nekoliko ponuđenih alternativa, pomoću dijaloškog okvira 'Pogled aplikacije' (odabrati opciju 'Start> View> Visual Style> Visual Style').

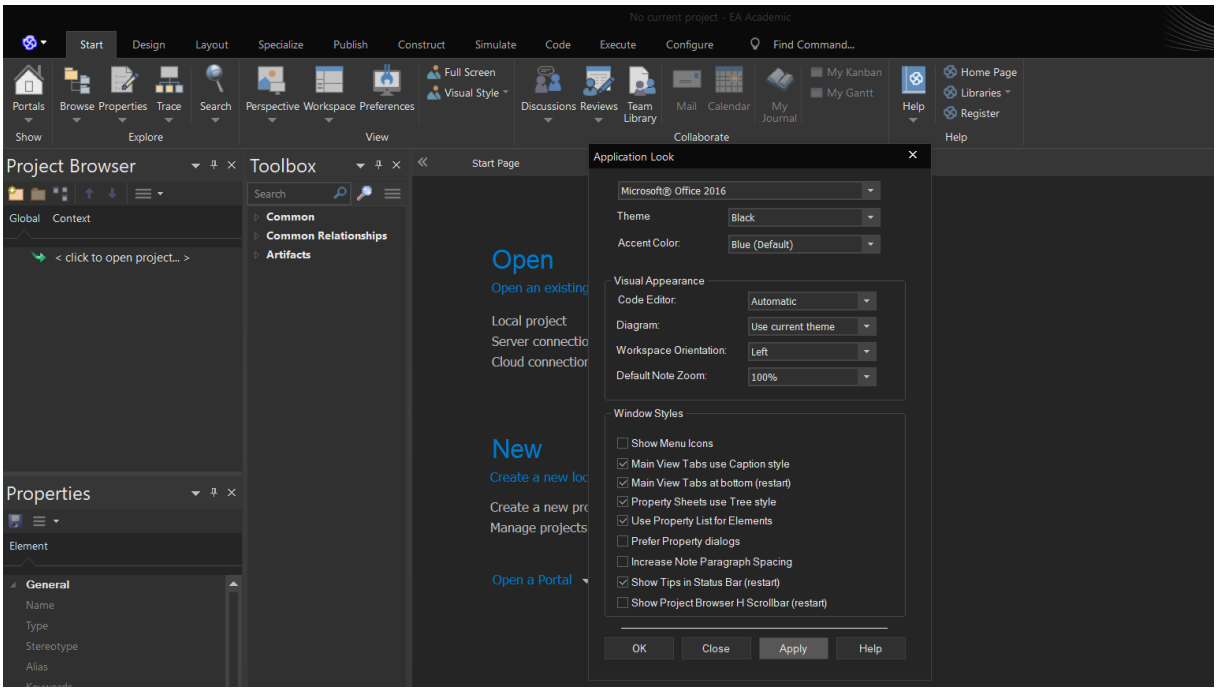


Slika 70. Promjena vizualnog stila



Slika 71. Izgled aplikacije

Prva tri polja u dijalogu imaju najveći vizualni učinak. Može se eksperimentirati s promjenom vrijednosti, kliknuti na **gumb Primijeni** i vidjeti učinak bez zatvaranja dijaloga. Treba imati na umu da za neke od permutacija ne treba imati vrijednosti u sva tri polja. Primjer kada je za temu odabrana crna boja:

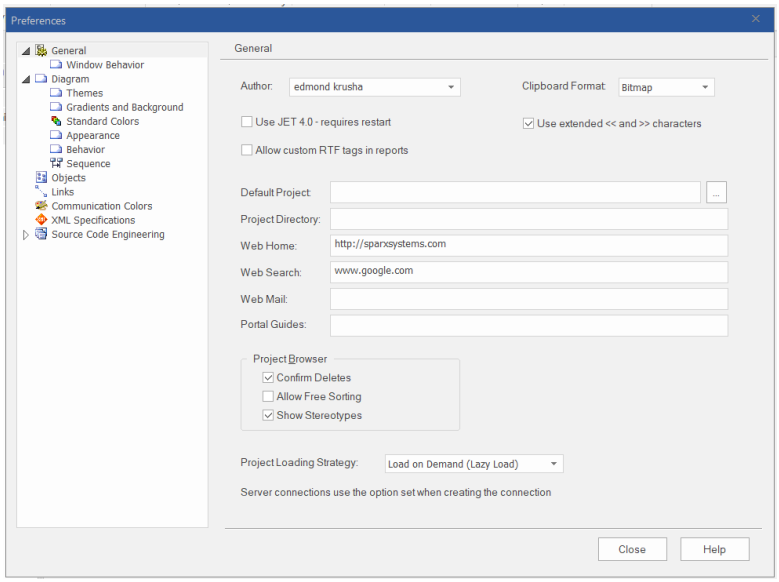


Slika 72. Prikaz aplikacije odabirom crne boje

Zadane vrijednosti prikazane su na slici, tako da ih se može vratiti ako je to potrebno.

16.2.Postavljanje određenih glavnih postavki

Postavke su aspekti sustava koji su personalizirani, tako da svaki projekt koji se otvori u Enterprise Architectu na radnoj stranici i pod korisničkim ID-om, izgleda i ponaša se na isti odabrani način. Svaki korisnik može vidjeti isti projekt drugačije, u skladu s vlastitim željama. Odabir dijaloškog okvira "Preferences" (odaberite opciju "Start> View> Preferences") prikazan na slici 73.



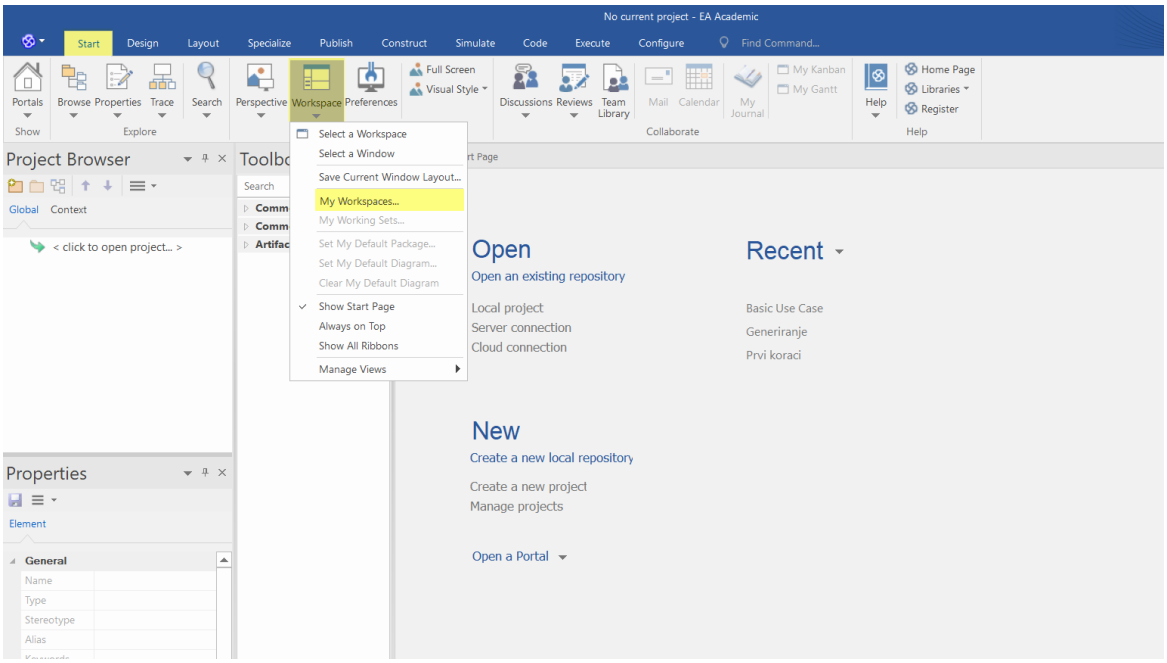
Slika 73. Odabir dijaloškog okvira 'Preferences'

Kao što se može vidjeti, postoje mnoge stranice opcija koje se mogu postaviti, iako nisu važne baš sve opcije. Kako bi se pregledale dostupne opcije, potrebno je kliknuti na nazive stranica s lijeve strane dijaloškog okvira. Ako se želi objašnjenje postavke, treba kliknuti **gumb Pomoć,** da bi se prikazao opis svake postavke na stranici.

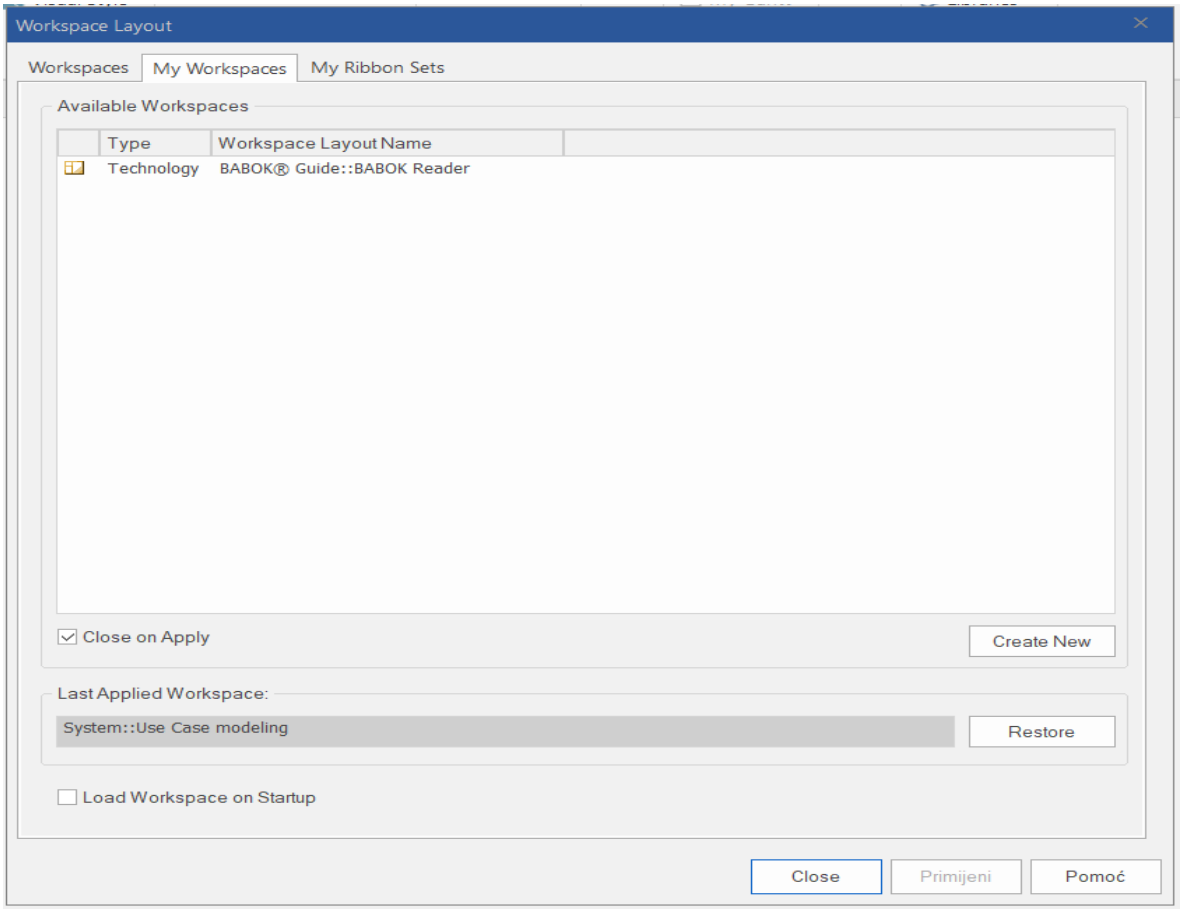
Najvidljivije postavke su boje objekta na dijagramima. Nakon otvaranja dijagrama, promijene se postavke na stranici "Dijagram> Standardne boje" dijaloškog okvira. Svaka postavka ima mogućnost povratka na zadanu boju, tako da se mogu isprobati sve vrste kombinacija, bez gubitka standardnih postavki.

16.3.Rad s vrpcom

Vrpce na vrhu radnog područja zajedno omogućuju pristup gotovo svakom alatu i postrojenju Enterprise Architecta, ali budući da nije vjerojatno da će se koristiti svaki alat i mogućnost, treba razmisliti o prilagodbi traka, kako bi se isključile stvari koje nisu potrebne. Najprije se može prikazati kartica "My Ribbon Sets" u dijaloškom okviru "Workspace Layout" (odabrati opciju "Start> View> Workspace> My Workspaces" i kliknuti na karticu "My Ribbon Sets").



Slika 74. Odabir Workspaces-a



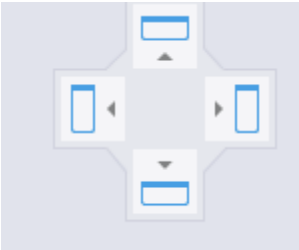
Slika 75. Izgled radnog prostora

Da bi se upisao naziv za vlastiti prikaz vrpce, treba kliknuti na **gumb Novo** . Vidjet će se da su sve vrpce sada navedene na ploči 'Konfiguriraj vrpce', a ako se proširi ime vrpce, prikazat će se i dijelovi te vrpce. Za svaku vrpcu i odjeljak koje nije potreban, treba izbrisati odabrani potvrdni okvir, tako da vrpca ili odjeljak budu isključeni iz trake s trakama.
Da bi se ažurirao prikaz na vrpci, treba kliknuti gumb Primijeni. Ako se želi ponovno postaviti zadani puni skup vrpci, treba kliknuti na pozadinu panela 'Stvori skup vrpce' i **gumb Poništi** .

16.3.1. Razmještaj s prozorima

Kada se prvi put otvori Enterprise Architect, različiti prozori nalaze se u položaju oko zaslona. Ovi prozori - i drugi koji se otvore - ne moraju ostati u tim položajima, može se jednostavno kliknuti na naziv prozora i povući prozor na prikladnije mjesto na zaslonu. Alternativno, može se **podići** prozor točno tamo gdje se stavi, tako da ostaje tamo, otvoren, na mnogim radnim sjednicama.

Dok se pomiče prozor, ikona 'kompasa' prikazuje se u središtu zaslona, a na rubu se prikazuje ikona 'točke kompasa'.

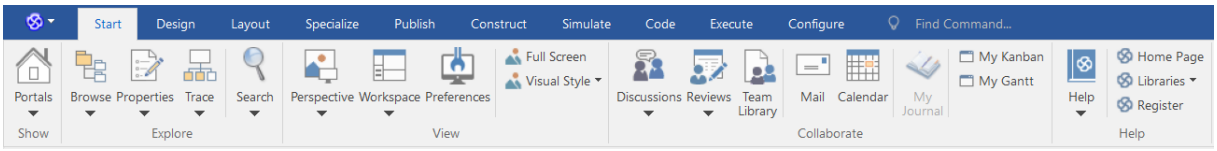


Kada se prozor pomakne preko točke kompasa, područje zaslona postaje plavo. To je područje koje će prozor zauzeti kada ga se ispusti i dok se ne pristane. Zatim se može prilagoditi rubove prozora, tako da odgovaraju odabranom prostoru.

Ako postoji još jedan prozor u području pristajanja, drugi kompas se prikazuje tamo, s kvadratom u sredini. Točke kompasa usidre povučeni prozor iznad, ispod ili na stranu drugog trenutnog prozora, dok ga kvadrat čini karticom istog okvira kao i taj prozor.

16.3.2. Vrpce ukratko

Vrpce, prikazane na vrhu radnog područja, omogućuju pristup gotovo svakom alatu i objektu u Enterprise Architectu. Svaka vrpca podržava domenu rada, kao što je oblikovanje komponenti modela i strukture. Podijeljena je u segmente, kako bi se alati i objekti grupirali prema strukturi ili funkciji koju podržavaju.

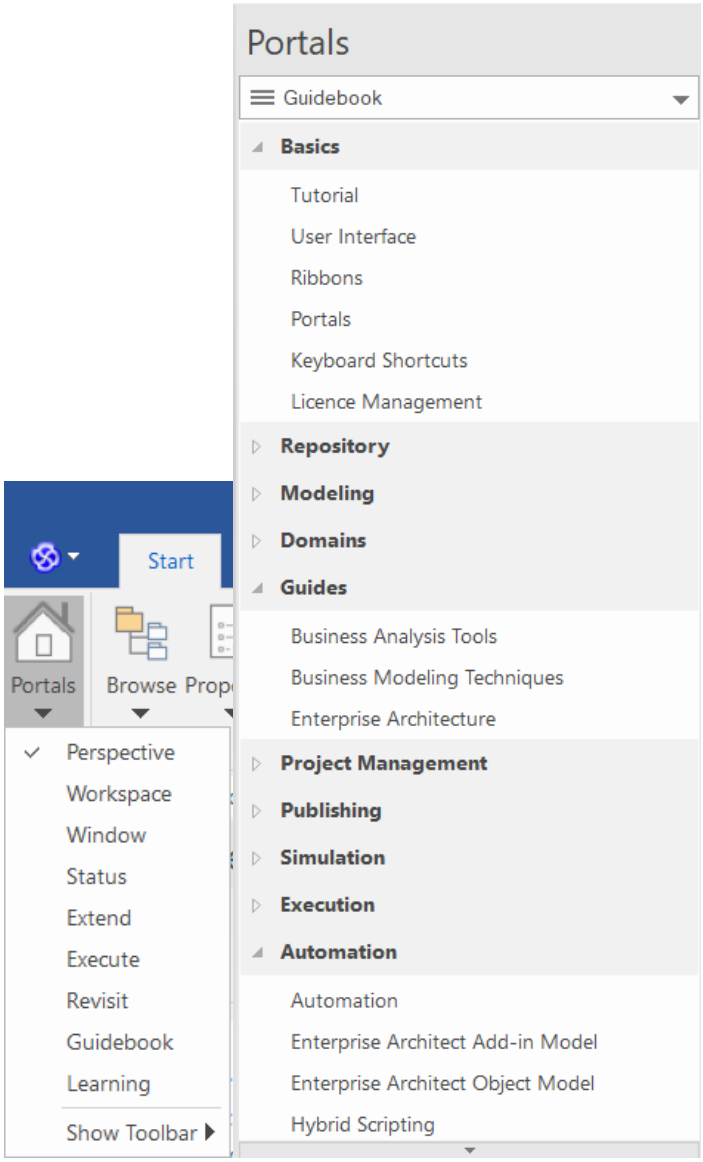


Ovaj primjer je vrpca Start, koja omogućuje pristup alatima za organiziranje posla i radnog područja. Potrebno je obratiti pozornost opcijama na ploči "Istraživanje (Explore)", koja pruža alate za gledanje onoga što je u vašem modelu i gdje su određeni objekti. Ti alati uključuju:

- Preglednici - poput prozora **preglednika projekta**, prozora **Element Browser** ili Resources
- Svojstva - kao što je **prozor Svojstva** i prozori s bilješkama, srodnim dokumentima, značajkama i ograničenjima
- Mogućnosti praćenja za ispitivanje i praćenje veza između parova elemenata i nizova elemenata
- Model Search (Pretraživanje modela) pruža mnoga unaprijed definirana pretraživanja za lociranje i označavanje različitih objekata u modelima

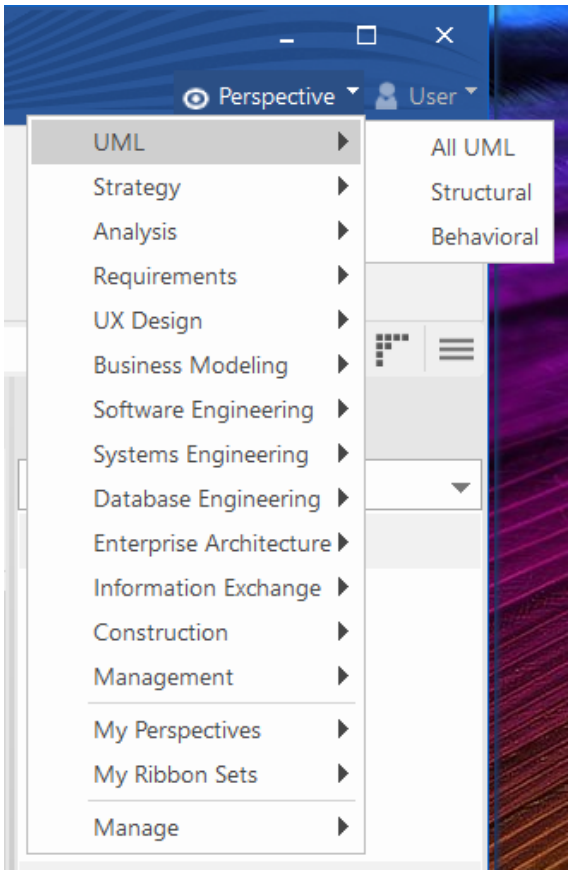
16.3.3. Prozor za portale

Prozor Portali prikazuje jedan od devet Portala i pruža mogućnost prebacivanja na drugi Portal. Portali identificiraju zadatke u određenim područjima rada i osiguravaju veze na prozor ili dijalog za izvršavanje svakog zadatka. Vodič i portali za učenje pružaju veze do informacija na mreži koje opisuju značajku Enterprise Architecta i kako raditi s njom. **Prozor Portali** prikazuje se odabirom opcije 'Prikaži> Portali' na lijevom kraju trake.



16.3.4. Perspektive (Perspective)

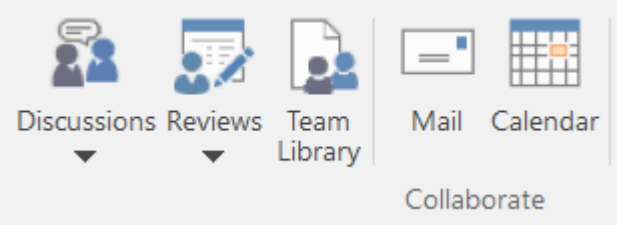
Perspektive poboljšavaju objekte za razvoj modela u Enterprise Architectu, kako bi podržali određenu tehnologiju ili domenu. Najvidljiviji učinci postavljanja Perspektive su da je sadržaj **Alata** za **dijagrame** ograničen na one skupove elemenata i konektora koji se koriste u domeni, a modeli Uzorci prilagođene su za generiranje dijagrama i / ili struktura modela točno one vrste za modeliranje u odabranom području. Može se prikazati i odabrati Perspektive s padajućeg popisa 'Perspective' u gornjem desnom kutu zaslona, kako je prikazano na slici 76.



Slika 76. Perspektive

16.3.5. Kolaboracija

Ako radi član tima, trebat će kontaktirati kolege, od brzog pitanja do popisa formalnih komentara za pregled. Bez obzira na vrstu poruke koju se želi poslati, Enterprise Architect ima mogućnost stvaranja i isporuke te poruke. Može se pristupiti objektima putem segmenta vrpce 'Start> Suradnja', odabirom opcija "Rasprave", "Recenzije" ili "Pošta". Opcija Rasprave daje značajku Chat.



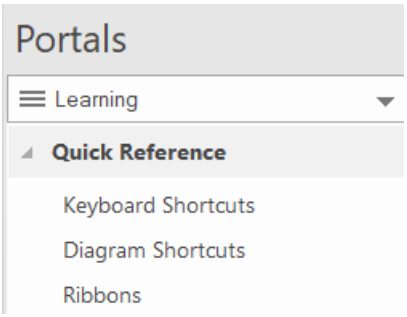
Slika 77. Kolaboracija

(treba imati na umu da uređaji Mail i Chat djeluju s korisničkim ID-ovima, koji se ne mogu postaviti u probnom izdanju. Prozori se prikazuju, ali se ne može poslati poruku.)

16.4. Tipke prečaca

Mnogi sadržaji i operacije u Enterprise Architectu mogu se odabrati pritiskom tipki na tipkovnici, kao i odabirom opcije vrpce ili izbornika.

Za popis tipkovnih prečaca koje se mogu koristiti, treba pristupiti portalu za učenje (odabrati opciju "Prikaži> Portali> Učenje") i pogledati teme *Tipkovni prečaci* i *prečaci dijagrama* pod naslovom "Brze preporuke".



Primjeri:

Otvaranje projekta

Prečac	Operacija
CTRL + N	*Izrada novog projekta*
CTRL + O	*Otvaranje postojećeg projekta*
CTRL + SHIFT + F11	*Ponovo učitavanje trenutnog projekta*

Dodavanje objekata projektu

Prečac	Operacija
CTRL + SHIFT + M	*Dodavanje novoga modela projektu pomoću čarobnjaka za model*
CTRL + INSERT	*Dodavanje novoga dijagrama paketu*
CTRL + M	*Dodavanje novoga elementa u paket*
SHIFT + F3 ILI CTRL + KLIK	*Stvaranje novog elementa iste vrste kako i zadnji*
CTRL + T	*Dodavanje označene vrijednosti elementu*
CTRL + SHIFT + F9	*Dodavanje atributa elementu*
CTRL + SHIFT + F10	*Dodavanje operacije elementu*
CTRL + F11	*Dodavanje drugih vrsta značajki elementu*

Izrada dokumentacije

Prečac	Operacija
F8	*Stvaranje dokumenata i izvještaja*
SHIFT + F8	*Stvaranje WEB izvješća u HTML-u*
CTRL + SHIFT + F8	*Stvaranje izvješća "Samo dijagrami"*

Izvođenje Code Inženjeringa

Prečac	Operacija
CTRL + P	*Uređivanje predložaka za generiranje koda*
ALT + 7	*Otvaranje prozora izvorni kod*
CTRL + ALT + Y	*Otvaranje izvornog direktorija*
CTRL + ALT + K	*Stvaranje izvornog koda paketa*
….....	

Ostale prečice preko portala (Portals > Learning > Keyboard Shortcuts).

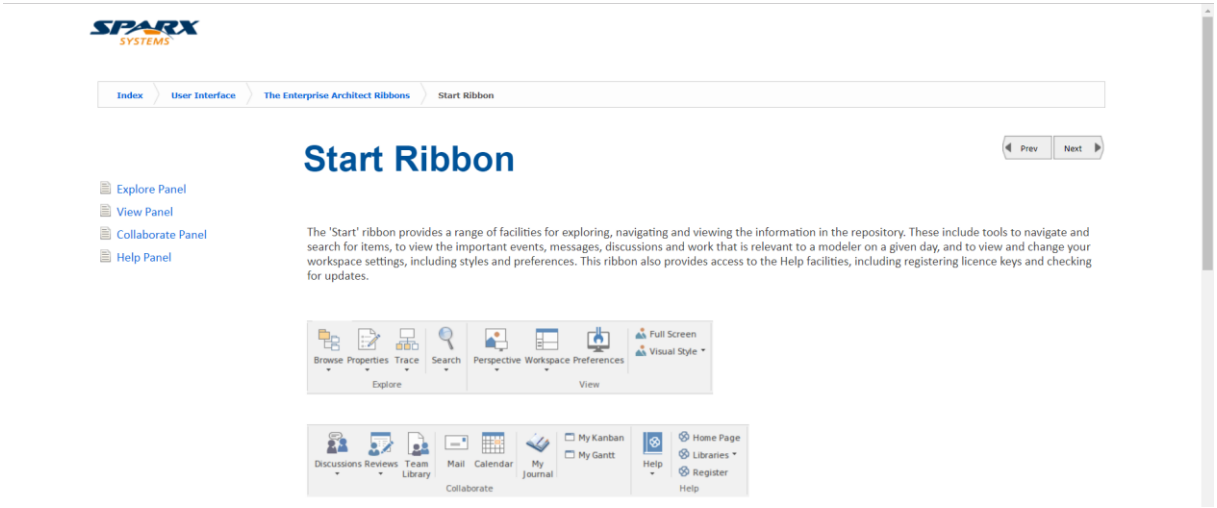
17. Dostupni resursi

17.1.Help – pomoć

U bilo kojem trenutku, ako se žele informacije o svrsi zaslona ili kako ga dovršiti, može se odabrati jedna od ovih opcija, da bi se prikazala tema pomoći, koja opisuje taj zaslon:

- Pritisnite **F1**
- Kliknite na ikonu u zaglavlju zaslona ili
- Kliknite **gumb Pomoć** na dnu zaslona

Ili se može pristupiti pomoći odabirom opcije "Start> Pomoć> Pomoć> Otvori sustav pomoći".

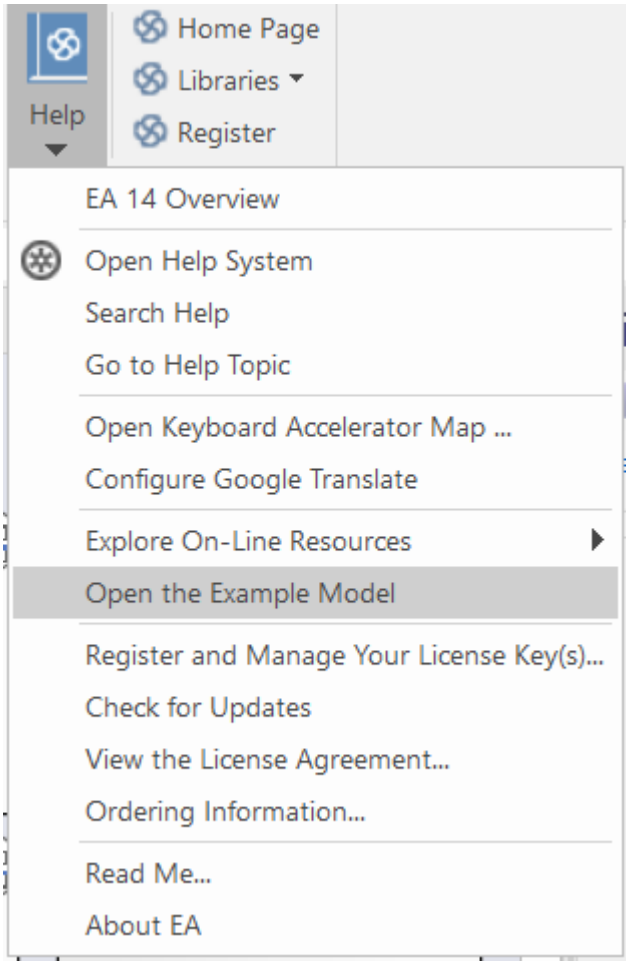


Slika 78. Otvaranje sustava pomoći

Svaka tema pomoći prikazuje se kao web-stranica. Može se kliknuti na popis tema s lijeve strane stranice, da bi se istražile srodne i / ili detaljnije teme, kako bi se pristupilo prema više konceptualnih tema ili Pomoć na drugom predmetnom području.

17.2.EA Primjeri

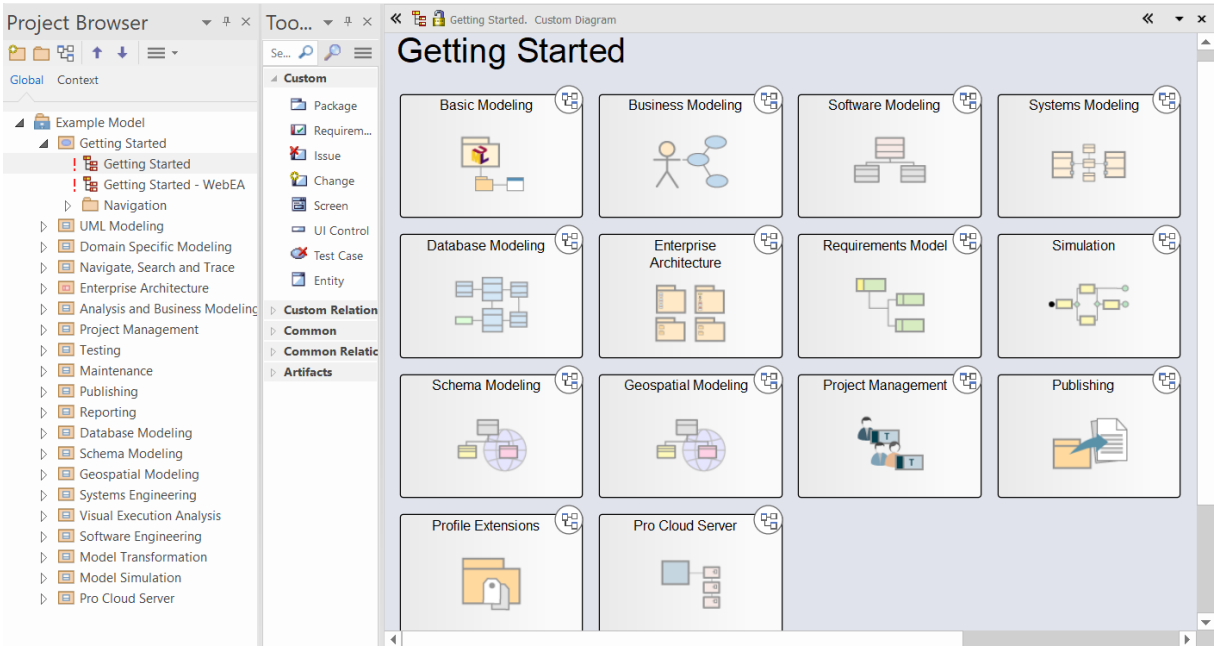
Primjer modela je skup fragmenata modela, koji su objedinjeni u jedan model, kako bi ilustrirali mnoge različite načine korištenja Enterprise Architecta u širokom rasponu domena modela. Model Primjera ne predstavlja model radnog sustava, već daje uzorke iz mnogih različitih sustava. Primjer modela nalazi se u sklopu instalacije proizvoda Enterprise Architect i može ga se otvoriti na ploči 'Pomoć', na vrpci 'Start', odabirom 'Pomoć | Otvorite primjer modela ".



Slika 79. Otvaranje Example modela

Kada se učita model EAExample, učitava se njegov zadani dijagram 'Getting Started'. Dijagram "Početak rada" sadrži niz veza na druge dijagrame koji se odnose na različite domene modeliranja, kao što su poslovno modeliranje ili modeliranje sustava, kao i neke od zadataka koji bi se mogli obaviti u Enterprise Architect-u, kao što su upravljanje projektima i objavljivanje.

Dvostrukim klikom na poveznicu dijagrama, otvorit će se dijagram. Sljedeća razina dijagrama obično sadrži opis i neke upute o tome kako pristupiti povezanim značajkama u programu Enterprise Architect te daljnje veze na primjere modela za to područje interesa.

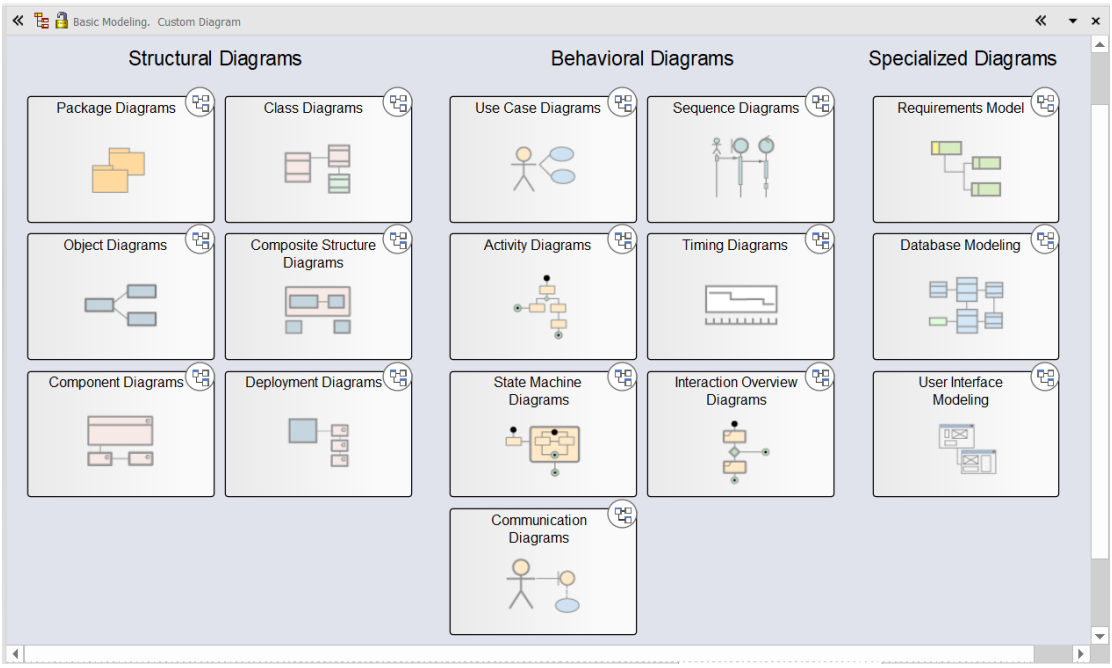


Slika 80. Getting Started (početak rada)

Kao primjer, pokušajte dvaput kliknuti na 'Basic Modeling'. To će vas dovesti do sljedećih primjera dijagrama:

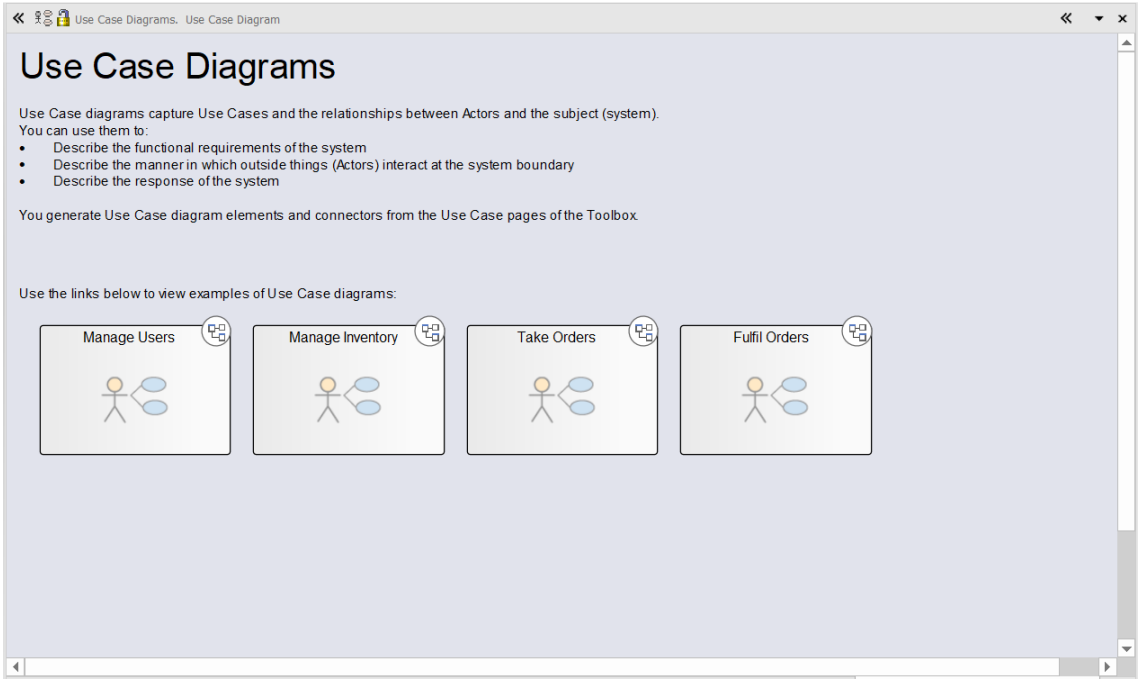
1. Structural Diagrams,
2. behavioral Diagrams i
3. specialized Diagrams

Unutar kojih možete odabrati veliki izbor ponuđenih dijagrama.



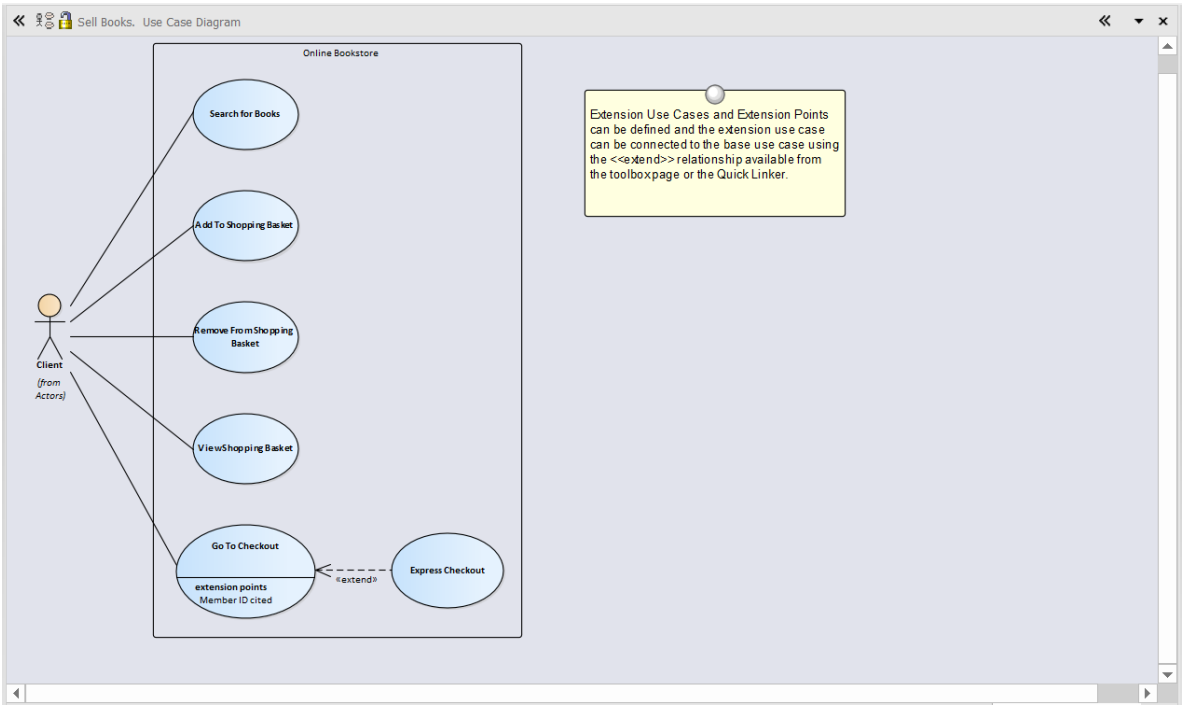
Slika 81. Basic Modeling - izbor dijagrama

Daljnjim odabirom, primjerice 'Use Case Diagrams' dobiju se sljedeće ponuđene opcije:



Slika 82. Use Case - ponuđene opcije

Dvostrukim klikom na opciju 'Take Orders' dobije se ponuđena i riješena opcija dijagrama korištenja 'Preuzimanje narudžbe'.



Slika 83. Preuzimanje narudžbe - Take Orders

Kao što se može uočiti, veliki je izbor ponuđenih opcija, te se mogu odabrati samo opcije koje su potrebne.

18. Pogovor

Problematiku obrađenu u ovom priručniku treba i dalje dopunjavati i osuvremenjivati svim spoznajama iz područja ‘Programskog inženjerstva', 'Projektiranja i oblikovanja informacijskih sustava' te 'Modeliranja poslovnih procesa' koji se održavaju u sklopu nastave na diplomskom stručnom studiju informatike Tehničkog veleučilišta u Zagrebu, u skladu s prilagodbom istoimenog nastavnog programa i kontinuiranog praćenja razvoja korištenog grafičkog alata. Metode i tehnike za projektiranje i razvoj softvera se i dalje razvijaju a u ovom priručniku su obrađene one koje su do sada polučile pozitivne rezultate u praksi. Njihova prednost je i u tome da se mogu primijeniti uz upotrebu raznih grafičkih alata od kojih je u ovom priručniku obrađena primjena Enterprise Architecta koji se koristi u nastavi na Tehničkom veleučilištu u Zagrebu.

Enterprise Architect je grafički alat s iznimnim mogućnostima i bogat skup značajki, kao pomoć pri upravljanju informacijama i inovacijama u današnjem složenom i zahtjevnom okruženju. Ovaj grafički alat 15 je godina kontinuirano razvijan i poboljšavan, kako bi mogao zadovoljiti potrebe programera, poslovnih analitičara, arhitekata, testera, voditelja projekta, dizajnera i drugih.

Enterprise Architect, kao grafički alat postao je nezaobilazan u procesima provođenja poslovne vizije i strategije, kao i učinkovite promjene poduzeća stvaranjem, komuniciranjem te poboljšanjem ključnih zahtjeva, načela i modela koji opisuju buduće stanje poduzeća i omogućuju njihovu evoluciju i transformaciju. Ovaj grafički alat u procesima transformacije podrazumijeva analizu i dizajn poduzeća u svojim sadašnjim i budućim stanjima sa strateškog, organizacijskog i tehnološkog gledišta. Navedeni grafički alat pomaže u poboljšanju organizacijske učinkovitosti, djelotvornosti i agilnosti poslovnih sustava.

I na kraju, može se reći da primjena ovoga grafičkog alata, koji se koristi na vježbama radi izvršavanja zadaća u izradi projekta i savladavanja gradiva, studentima donosi koristi na njihovim sadašnjim i budućim poslovnim izazovima, stručnim i znanstvenim radovima. Ako ovaj priručnik pomogne u razumijevanju obrađenog gradiva, objašnjenog i uspješno uputi čitatelje kako konkretno primijeniti sve ono što je u njemu obrađeno, autori su postigli željeni cilj.

19. Popis slika

Slika 1. Prikaz hijerarhijske strukture vodopada [2] 11
Slika 2. Popis stavki za proizvod [14] 18
Slika 3. Burn-down grafikon [15] 20
Slika 4. Događaji u Scrumu [11] 21
Slika 5. Primjer Scrum ploče sa zadacima [17] 24
Slika 6. Primjer digitalne Scrum ploče [18] 25
Slika 7. Kanban ploča [22] 27
Slika 8. Predložak digitalne Kanban ploče za razvoj softvera [23] 27
Slika 9. Predložak digitalne Kanban ploče za marketing [23] 28
Slika 10. Fizička Kanban kartica [24] 28
Slika 11. Primjer prednje strane digitalne Kanban kartice [24] 29
Slika 12. Primjer poleđine digitalne Kanban kartice [24] 29
Slika 13. Koraci u automatskom testiranju [33] 40
Slika 14. Vrste testiranja bijele kutije [38] 45
Slika 15. Struktura standarda ISO/IEC 29119 [43] 50
Slika 16. Vrste dokumentacije [45] 56
Slika 17. Mind mapa 62
Slika 18. Primjer dijagrama aktivnosti 63
Slika 19. Primjer dijagrama aktivnosti 67
Slika 20. Primjer dijagrama klasa 70
Slika 21. Org chart toolbox 71
Slika 22. Dodavanje org chart uzorka 72
Slika 23. Odabir uzorka 72
Slika 24. Organizacijska shema unutarnje strukture tvrtke 73
Slika 25. Primjer dijagrama slučaja korištenja 78
Slika 26. Primjer dijagrama toka podataka 81
Slika 27. Primjer dijagrama slijeda 86
Slika 28. Generiranje dokumentacije 87
Slika 29. Drugi načini generiranja – preko kartice 'Publish' ili gumba F8 88
Slika 30. Nastavak rada s čarobnjakom za generiranje dokumenata 88
Slika 31. Generiranje izvornog koda 89
Slika 32. Drugi načini generiranja – preko kartice 'Code' ili gumba F8 89

Slika 33. Odabir entiteta 90
Slika 34. Batch obrada - generiranje izvršeno 90
Slika 35. Primjer generiranog koda 91
Slika 36. Odabir programskog jezika 92
Slika 37. Zamjena ili ponovno namještanje programskih jezika 92
Slika 38. Upravljanje generiranjem koda 93
Slika 39. Odabrati naziv datoteke i lokaciju za spremište 96
Slika 40. Stvaranje korijenskog elementa 97
Slika 41. Odabir EA verzije 99
Slika 42. Glavni radni prostor Enterprise Architect-a 100
Slika 43. Izbornik perspektiva 102
Slika 44. Osnovni model slučaja korištenja 103
Slika 45. Stvaranje prvog modela sadržaja 104
Slika 46. Grafički model 'Osnovnog modela slučaja korištenja' 104
Slika 47. Primjer odabranog elementa u dijagramu 105
Slika 48. Dodavanje dijagrama scenarija slučaju korištenja 108
Slika 49. New composite element 108
Slika 50. Preglednik projekta - Interaction 109
Slika 51. Nova vizualna reprezentacija sudionika A 110
Slika 52. Dodavanje graničnog elementa 110
Slika 53. Veza između dva entiteta 111
Slika 54. Prikaz povratne poruke 112
Slika 55. Stvaranje novog paketa 114
Slika 56. Moj klas model 114
Slika 57. Izrada dijagrama klase unutar novog paketa 115
Slika 58. Stvaranje novog dijagrama klase 116
Slika 59. Radna površina i kutija s alatima za izradu dijagrama klasa 117
Slika 60. Dodavanje elemenata dijagramu klase 118
Slika 61. Prikaz stvorenog odnosa koji ukazuje da je Class1 generalizacija Class2 119
Slika 62. Odabir vrste odnosa i vrstu elemenata 120
Slika 63. Rezultirajući konektor i element 121
Slika 64. Izgradnja modela zahtjeva 122
Slika 65. Dodavanje uzorka u korijenskom modelu modela 123
Slika 66. Odabir iz izbornika 'Perspektive' 124

Slika 67. Basic Class Diagram with Operations 125
Slika 68. Prikaz dodanog uzorka modela vašem projektu 126
Slika 69. Prikaz prozora 'Značajke' 127
Slika 70. Promjena vizualnog stila 129
Slika 71. Izgled aplikacije 130
Slika 72. Prikaz aplikacije odabirom crne boje 131
Slika 73. Odabir dijaloškog okvira 'Preferences' 132
Slika 74. Odabir Workspaces-a 133
Slika 75. Izgled radnog prostora 133
Slika 76. Perspektive 137
Slika 77. Kolaboracija 137
Slika 78. Otvaranje sustava pomoći 141
Slika 79. Otvaranje Example modela 142
Slika 80. Getting Started (početak rada) 143
Slika 81. Basic Modeling - izbor dijagrama 144
Slika 82. Use Case - ponuđene opcije 144
Slika 83. Preuzimanje narudžbe - Take Orders 145

20. Popis primjera i tablica

Primjer 1. Doxygen - dokumentiranje klase ExampleClass [47] 57
Primjer 2. JavaDoc - dokumentiranja izvornog Java koda 58
Primjer 3. JavaDoc - komentari na razini članova klase 58

21. Reference

[1] W. W. Royce, »Managing the development of large software systems,« *Proceedings of IEEE WESCON,* br. 26, pp. 328-338 , 1970.

[2] D. W. W. Royce, »Managing the development of large software systems,« *Proceedings of IEEE WESCON 26 ,* pp. 1-9, 8 1970.

[3] D. Thakur, »Types of Software Maintenance,« Computer Notes, [Mrežno]. Available: http://ecomputernotes.com/software-engineering/types-of-software-maintenance. [Pokušaj pristupa 30 08 2019].

[4] Cprime, »What is Agile? - What is Scrum? - Agile FAQ's,« Cprime, [Mrežno]. Available: https://www.cprime.com/resources/what-is-agile-what-is-scrum/. [Pokušaj pristupa 30 08 2019].

[5] K. Beck, M. Beedle, A. van Bennekum i et.al, »Manifesto for Agile Software Development,« 2001. [Mrežno]. Available: http://agilemanifesto.org. [Pokušaj pristupa 01 09 2019].

[6] K. Beck, M. Beedle, A. van Bennekum i et.al, »Principles behind the Agile Manifesto,« [Mrežno]. Available: http://agilemanifesto.org/principles.html. [Pokušaj pristupa 01 09 2019].

[7] G. Verheyen, »Scrum is not an acronym,« 09 01 2014. [Mrežno]. Available: https://guntherverheyen.com/2014/01/09/scrum-is-not-an-acronym/. [Pokušaj pristupa 10 10 2019].

[8] J. S. Ken Schwaber, »Sveobuhvatni vodič kroz Scrum: pravila igre,« Scrum Guide, November 2017. [Mrežno]. Available: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Croatian.pdf#zoom=100. [Pokušaj pristupa 10 10 2019].

[9] M. Cohn, »Getting Agile with Scrum,« Mountain Goat Software, 6 6 2014. [Mrežno]. Available: https://www.mountaingoatsoftware.com/uploads/presentations/Getting-Agile-With-Scrum-Norwegian-Developers-Conference-2014.pdf. [Pokušaj pristupa 10 10 2019].

[10] M. James, »Scrum Methodology,« [Mrežno]. Available: http://scrummethodology.com/. [Pokušaj pristupa 03 09 2019].

[11] L. W. Michael James, »Scrum Reference Card,« CollabNet, Inc., [Mrežno]. Available: https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf. [Pokušaj pristupa 03 09 2019].

[12] P. Green, »Scrum Artifacts,« Scrum Alliance, 2017. [Mrežno]. Available: https://www.scrumalliance.org/learn-about-scrum/scrum-elearning-series/scrum-artifacts. [Pokušaj pristupa 10 10 2019].

[13] Visual Paradigm, »What are Scrum Artifacts?,« Visual Paradigm, [Mrežno]. Available: https://www.visual-paradigm.com/scrum/what-are-scrum-artifacts/. [Pokušaj pristupa 10 10 2019].

[14] »Product Backlog,« Infinity, [Mrežno]. Available: https://startinfinity.com/product-management-framework/product-backlog. [Pokušaj pristupa 19 10 2019].

[15] »What is Burndown Chart in Scrum?,« Visual Paradigm, [Mrežno]. Available: https://www.visual-paradigm.com/scrum/scrum-burndown-chart/. [Pokušaj pristupa 19 10 2019].

[16] L. W. Michael James, »Scrum Reference Card,« CollabN, 2017. [Mrežno]. Available: https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf. [Pokušaj pristupa 11 10 2019].

[17] Martial, »The Scrum board,« Agile but Pragmatic, 11 03 2012. [Mrežno]. Available: http://agilebutpragmatic.blogspot.com/2012/03/visual-wall.html. [Pokušaj pristupa 19 10 2019].

[18] J. Wilson, »Virtual Scrum Boards,« Click Here Labs, 19 05 2015. [Mrežno]. Available: https://clickherelabs.com/productivity/virtual-scrum-boards/. [Pokušaj pristupa 19 10 2019].

[19] »Kanban Explained for Beginners,« Kanbanize, [Mrežno]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban/. [Pokušaj pristupa 01 10 2019].

[20] »What Is Kanban? An Introduction to Kanban Methodology,« VersionOne CollabNet, [Mrežno]. Available: https://resources.collab.net/agile-101/what-is-kanban. [Pokušaj pristupa 01 10 2019].

[21] A. Fryrear, »Kanban System for Agile Marketing: A Beginner's Guide,« Workfront, [Mrežno]. Available: https://www.workfront.com/blog/kanban-system-for-agile-marketing-a-beginners-guide. [Pokušaj pristupa 01 10 2019].

[22] Remi, »Kanban Project Management Methodology (Easy Explanation),« Think Thyme, 09 05 2018. [Mrežno]. Available: https://thinkthyme.com/project-management/kanban-project-management-methodology. [Pokušaj pristupa 19 10 2019].

[23] »Kanban Software Guide | How to Choose the Best Tool for You,« Kanbanize, [Mrežno]. Available: https://kanbanize.com/kanban-resources/kanban-software/. [Pokušaj pristupa 19 10 2019].

[24] »What Is a Kanban Card? Details and Example,« Kanbanize, [Mrežno]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-card/. [Pokušaj pristupa 03 10 2019].

[25] »What is Scrumban?,« Planview LeanKit, [Mrežno]. Available: https://leankit.com/learn/agile/what-is-scrumban/. [Pokušaj pristupa 15 10 2019].

[26] J. Bowes, »Scrum in practice: the Scrum of Scrums meeting,« Manifesto, 05 01 2016. [Mrežno]. Available: https://manifesto.co.uk/scrum-of-scrums-meeting/. [Pokušaj pristupa 16 10 2019].

[27] »Scrum of Scrums. Work together with multiple Scrum teams.,« Micro Tool, [Mrežno]. Available: https://www.microtool.de/en/knowledge-base/what-is-the-scrum-of-scrums/. [Pokušaj pristupa 16 10 2019].

[28] »Scrum of Scrums,« Knowledgejut Tutorials, [Mrežno]. Available: https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-of-scrums. [Pokušaj pristupa 18 10 2019].

[29] »What is Software Quality?,« Try QA, [Mrežno]. Available: http://tryqa.com/what-is-software-quality/. [Pokušaj pristupa 19 10 2019].

[30] Rajkumar, »What Is Software Testing – Definition, Types, Methods, Approaches,« Software Testing Material, [Mrežno]. Available: https://www.softwaretestingmaterial.com/software-testing/. [Pokušaj pristupa 19 10 2019].

[31] »Software Testing | Basics,« Geeks for Geeks, [Mrežno]. Available: https://www.geeksforgeeks.org/software-testing-basics/. [Pokušaj pristupa 19 10 2019].

[32] »7 Software Testing Principles: Learn with Examples,« Gurru99, [Mrežno]. Available: https://www.guru99.com/software-testing-seven-principles.html. [Pokušaj pristupa 20 10 2019].

[33] »Automation Testing Tutorial: What is, Process, Benefits & Tools,« Guru99, [Mrežno]. Available: https://www.guru99.com/automation-testing.html. [Pokušaj pristupa 19 10 2019].

[34] »Manual Testing Tutorial for Beginners: Concepts, Types, Tool,« Guru99, [Mrežno]. Available: https://www.guru99.com/manual-testing.html. [Pokušaj pristupa 19 10 2019].

[35] »Black Box Testing – Understanding the Basics,« ReQtest, 08 08 2019. [Mrežno]. Available: https://reqtest.com/testing-blog/black-box-testing/. [Pokušaj pristupa 25 10 2019].

[36] »What is BLACK Box Testing? Techniques, Example & Types,« Guru99, [Mrežno]. Available: https://www.guru99.com/black-box-testing.html. [Pokušaj pristupa 25 10 2019].

[37] »What is WHITE Box Testing? Techniques, Example, Types & Tools,« Guru99, [Mrežno]. Available: https://www.guru99.com/white-box-testing.html. [Pokušaj pristupa 25 10 2019].

[38] »White Box Testing: A Complete Guide with Techniques, Examples, & Tools,« Software Testing Help, [Mrežno]. Available: https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/. [Pokušaj pristupa 25 10 2019].

[39] U. Eriksson, »White Box Testing – Step by Step Guide on Everything you Need to Know,« ReQtest, 28 11 2016. [Mrežno]. Available: https://reqtest.com/testing-blog/white-box-testing-example/. [Pokušaj pristupa 25 10 2019].

[40] »Gray Box Testing,« Veracode., [Mrežno]. Available: https://www.veracode.com/security/gray-box-testing. [Pokušaj pristupa 27 10 2019].

[41] »What is Grey Box Testing With Examples,« Testbytes, 18 04 2019. [Mrežno]. Available: https://www.testbytes.net/blog/grey-box-testing/. [Pokušaj pristupa 28 10 2019].

[42] L. Sharma, »Software Testing Standards : An Ultimate Guide [2019],« Tools QA, 08 07 2019. [Mrežno]. Available: https://www.toolsqa.com/software-

testing/istqb/software-testing-standards-an-ultimate-guide-2019/. [Pokušaj pristupa 1 11 2019].

[43] U. Eriksson, »ISO/IEC/IEEE 29119: Creating a standard approach to test software,« ReQtest, 07 08 2014. [Mrežno]. Available: https://reqtest.com/testing-blog/isoiecieee-29119-creating-a-standard-approach-to-test-software/. [Pokušaj pristupa 28 10 2019].

[44] »ISO/IEC/IEEE 29119 Software Testing,« ISO/IEC/IEEE 29119 Software Testing, 11 12 2018. [Mrežno]. Available: https://softwaretestingstandard.org/. [Pokušaj pristupa 1 11 2019].

[45] »Software Documentation Types and Best Practices,« Prototypr.io, 16 01 2018. [Mrežno]. Available: https://blog.prototypr.io/software-documentation-types-and-best-practices-1726ca595c7f. [Pokušaj pristupa 1 11 2019].

[46] »Doxygen,« Doxygen - About, [Mrežno]. Available: http://www.doxygen.nl/index.html. [Pokušaj pristupa 28 12 2019].

[47] MINOS Software, » How to document your code for doxygen,« 14 4 2005. [Mrežno]. Available: https://www-numi.fnal.gov/offline_software/srt_public_context/WebDocs/doxygen-howto.html. [Pokušaj pristupa 18 1 2020].

[48] D. H. S. H. Laurie Williams, »Javadoc,« North Carolina State University, 16 08 2004. [Mrežno]. Available: https://web.archive.org/web/20170613233020/http:/agile.csc.ncsu.edu/SEMaterials/tutorials/javadoc/#section2_0. [Pokušaj pristupa 28 12 2019].

[49] Rajkumar, »Seven Principles of Software Testing | Software Testing Material,« Software Testing Material, [Mrežno]. Available: https://www.softwaretestingmaterial.com/principles-of-software-testing-2/. [Pokušaj pristupa 20 10 2019].

[50] T. Murnane, »What’s the latest on ISO/IEC 29119 Software Testing?,« EuroSTAR Software Testing Conference, 08 10 2010. [Mrežno]. Available: https://conference.eurostarsoftwaretesting.com/whats-latest-isoiec-29119-software-testing-dr-tafline-murnane/. [Pokušaj pristupa 1 11 2019].