PRILOGA E: Osnovni pojmi UML notacije uporabljeni v metodologiji

Diagram primerov uporabe

Obnašanje obravnavanega sistema, t.j. funkcionalnost , ki jo mora sistem nuditi, dokumentiramo z modelom primerov uporabe. Ta vsebuje predvidene funkcije sistema (primeri uporabe), njegovo okolje (akterje) in odnose med primeri uporabe in akterji (diagrami primerov uporabe). Najvažnejša vloga modela primerov uporabe je komunikacija: omogoča izmenjavo mnenj o funkcionalnosti in obnašanju sistema med strankami oz. končnimi uporabniki in tistimi, ki ga načrtujejo. Razvoj modela primerov uporabe se začne z identifikacijo akterjev in osnovnih primerov uporabe sistema, v nadaljevanju obstoječim primerom uporabe dodamo bolj podrobne podatke in model po potrebi dopolnimo z dodatnimi primeri uporabe.

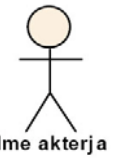
Akter

Akter (*actor*) je v splošnem lahko posamezni uporabnik sistema, naprava, podatkovna baza ali drug sistem, ki z opisovanim sistemom komunicira, izmenjuje podatke in iz njega pridobiva rezultate.
Akter predstavlja možno medsebojno vplivanje zunanjih uporabnikov in sistema. Posamezni fizični uporabnik lahko deluje v vlogi različnih akterjev sistema. Različni uporabniki lahko igrajo vlogo istega akterja in tako predstavljajo več pojavov iste definicije akterja. Na primer: ena oseba je lahko v trgovini blagajnik, v trenutku, ko kupuje artikle v tej trgovini, pa tudi kupec.

V fazi določanja akterjev sistema nam pomaga, če najdemo odgovore na naslednja vprašanja:

- Koga zanima določena zahteva, ki jo mora zadovoljiti sistem?
- Kdo v organizaciji bo uporabljal sistem?
- Kdo bo nekaj pridobil z uporabo sistema?
- Kdo bo sistem oskrboval s podatki, kdo uporabljal in kdo jih bo odstranjeval?
- Kdo bo podpiral in vzdrževal sistem?
- Ali sistem uporablja zunanje vire?
- Ali ena oseba igra več različnih vlog?
- Ali več ljudi igra isto vlogo?

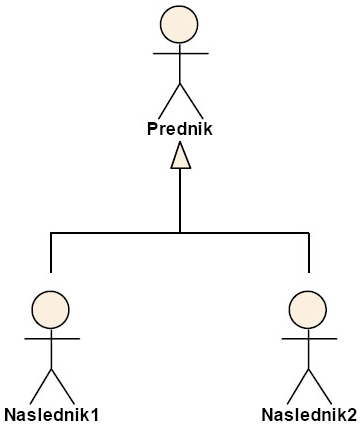
Notacija je prikazana na sliki 1: akterja prikažemo z likom možica, pod lik napišemo ime vloge akterja.



Slika 1: Notacija za akterja

Generalizacija: Situacijo, kjer ima več akterjev v okviru svoje lastne vloge neko skupno (splošnejšo) vlogo, opišemo z generalizacijo. V tem primeru se obnašanje, ki se nanaša na splošno vlogo, prikaže kot vloga splošnejšega akterja – prednika. Specifični akterji – nasledniki podedujejo lastnosti in operacije svojega prednika in jih tudi vsak na svoj način

razširijo - dodajo lastne ali (in) preoblikujejo podedovane lastnosti in operacije. Pojav akterja - naslednika se lahko vedno uporabi v primerih, kjer se pričakuje pojav akterja – prednika. Grafično se generalizacija prikaže s puščico z votlo konico, usmerjeno proti predniku (slika 2).



Slika 2: Notacija za generalizacijo akterjev

Primer uporabe

Primer uporabe (*use case*) je povezana enota funkcionalnosti, ki jo zagotavlja sistem ali podsistem in je izražena z zaporedji sporočil, izmenjanimi med sistemom in enim ali več uporabniki. Zaključi se z merljivim rezultatom, ki ima nek pomen za določenega akterja. Detajlni zaporedni opis primera uporabe vsebuje celotno obnašanje: glavni potek dogodkov, različne variacije normalnega obnašanja in vse izjemne situacije, do katerih lahko pride, vključno s pričakovanimi odzivi akterja na obnašanje sistema. Z uporabnikovega gledišča so izjemne situacije nenormalne, z gledišča sistema pa so to dodatne variacije, ki jih je potrebno opisati in tudi znati obvladati. V modelu je izvršitev vsakega primera uporabe neodvisna od drugih, vendar pa lahko pri izvedbi primerov uporabe pride do implicitnih odvisnosti med njimi zaradi skupnih objektov.
Notacija je prikazana na sliki 3: primer uporabe narišemo kot elipso, znotraj katere se nahaja ime primera uporabe.

Slika 3: Notacija za primer uporabe

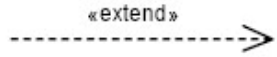
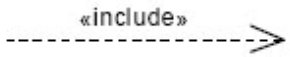
Pri določanju primerov uporabe sistema si lahko pomagamo z naslednjimi vprašanji:

- Kakšne so naloge vsakega akterja?
- Ali bo kateri od akterjev vnašal, shranjeval, brisal ali pregledoval informacije v sistemu?
- Kateri primer uporabe bo vnašal, shranjeval, brisal ali pregledoval te informacije?
- Ali bo moral kateri od akterjev obvestiti sistem o nenadnih zunanjih spremembah?
- Ali mora biti katerikoli od akterjev obveščen o določenih dogodkih v sistemu?
- Kateri primer uporabe bo podpiral in vzdrževal sistem?
- Ali primeri uporabe podpirajo vse funkcionalne zahteve sistema?

Odnosi med primeri uporabe in akterji

Primer uporabe lahko sodeluje v naslednjih vrstah relacij:

Preglednica 1: Vrste relacij med primeri uporabe

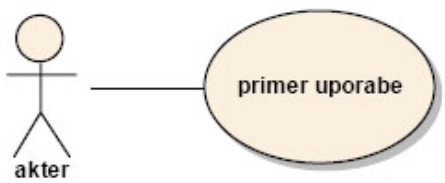
Relacija	Pomen	Notacija
asociacija	Komunikacijska povezava med akterjem in primerom uporabe, v kateri sodeluje.	
razširja (*extend*)	Primer uporabe na začetku puščice (lahko) razširi obnašanje primera uporabe na koncu puščice.	
vsebuje (*include*)	Primer uporabe na začetku puščice vsebuje obnašanje primera uporabe na koncu puščice.	
generalizacija primera uporabe	Relacija med splošnim in bolj specifičnim primerom uporabe, ki podeduje značilnosti splošnega primera uporabe in mu doda nove.	

Opis obsežnega primera uporabe lahko razstavimo na druge, bolj enostavne primere uporabe. (podobno kot lahko opis razreda definiramo na podlagi opisa nadrazreda). Primer uporabe lahko vključuje vedenje drugih primerov uporabe kot sestavne dele lastnega vedenja – to je relacija vsebovanja (*include*). Vsebovani primer uporabe ni specializacija osnovnega primera uporabe in osnovnega primera uporabe torej ne more zamenjati.
Primer uporabe lahko definiramo tudi kot razširitev osnovnega primera uporabe – temu rečemo relacija razširitve (*extend*). Obstaja lahko več razširitev osnovnega primera uporabe, ki se lahko tudi hkrati izvajajo.

Črtkani puščici, ki označuje relacijo »extend«, lahko pripnemo oznako, v kateri navedemo potrebne pogoje (*condition*) za »extend« primer uporabe in mesto v osnovnem primeru uporabe, kjer se začne izvajati »extend« primer uporabe (*extension point*).

Primer uporabe je lahko specializiran z enim ali več primeri uporabe – generalizacija primera uporabe. Vsak specifični primer uporabe lahko uporabimo v situacijah, v katerih se pričakuje nadrejeni primer uporabe. Puščica povezave, ki ponazarja generalizacijo, je usmerjena k nadrejenemu primeru uporabe.

Vsak primer uporabe ima primarnega akterja, ki od sistema zahteva, naj nudi neko uslugo. Običajno je pobudnik (iniciator) primera uporabe. Primer uporabe skuša doseči cilj, ki ga ima primarni akter. Sekundarni akterji so ostali akterji, s katerimi sistem komunicira tekom izvajanja primera uporabe.



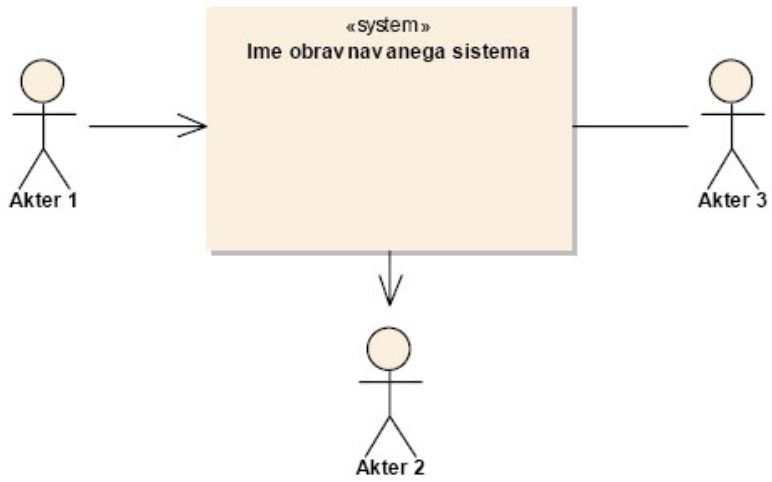
Slika 4: Relacija med akterjem in primerom uporabe

Vsebino primera uporabe lahko opišemo na več načinov: z neformalnimi tekstovnimi opisi, diagrami kolaboracije, diagrami zaporedij izvedbe ali diagrami aktivnosti.

Pregledni diagram primerov uporabe

Pregledni diagram primerov uporabe povzame medsebojno vplivanje akterjev in sistema. Pogosto ga imenujemo kontekstualni diagram (njegov namen je posplošen prikaz domene obravnavanega sistema). Prikazuje obravnavani sistem in vse njegove akterje, primeri uporabe ostanejo skriti. Ta diagram narišemo takoj, ko smo določili akterje in preden se lotimo določanja primerov uporabe.

Če je akter vedno iniciator primerov uporabe, v katerih sodeluje, ga s sistemom povezuje puščica, usmerjena proti sistemu. Če akter ni nikoli iniciator, naj bo puščica usmerjena proti njemu. Kadar nismo prepričani, puščico izpustimo (slika 5).



Slika 5: Primer preglednega diagrama primerov uporabe

Podroben opis primerov uporabe

Diagram s primeri uporabe in akterji shematično prikazuje zahteve, ki jih mora sistem zadovoljiti, vendar pa je za dejansko razumevanje delovanja sistema ta nivo preveč splošen. Pomembna dejstva, kot so npr., kdo je najpomembnejši akter ali kateri koraki so vsebovani v primeru uporabe, najlaže navedemo v obliki tekstovnega opisa, ki naj bi se izdelal za vsak primer uporabe. UML ne podaja strogih navodil, kaj točno mora vsebovati.

V preglednici 2 je prikazano, katere podrobnosti sem v praktičnem delu naloge vključila v podrobne opise primerov uporabe.

Preglednica 2: Oblika podrobnega opisa primera uporabe

Ime primera uporabe	**Ime**
Cilj	Mesto primera uporabe znotraj sistema in zakaj je primer uporabe pomemben.

Predpogoji	Kaj se mora zgoditi, preden se začne primer uporabe izvrševati.
Uspešni končni pogoji	Kakšni naj bi bili pogoji sistema, če se primer uporabe uspešno izvrši.
Neuspešni končni pogoji	Kakšni naj bi bili pogoji sistema, če se primer uporabe ne izvrši uspešno.
Primarni akterji	Glavni akterji, ki sodelujejo pri primeru uporabe. Večinoma so to akterji, ki sprožijo izvršitev primera uporabe ali pa ob izvršitvi direktno pridobijo podatke.
Sekundarni akterji	Akterji, ki sodelujejo pri primeru uporabe, a nimajo glavne vloge pri izvršitvi primera uporabe.
Glavni potek dogodkov	Pomembni koraki, ko se primer uporabe neovirano izvrši po najkrajši možni poti.
»Extend« primeri uporabe	Primeri uporabe, ki lahko pri določenih pogojih nadgradijo osnovni primer uporabe.
Alternativni poteki dogodkov	Opis katerihkoli alternativnih korakov glede na korake, opisane v glavnem poteku dogodkov (zaradi boljše preglednosti sem alternativne poteke opisala v ločeni tabeli).

Pri glavnem in alternativnih potekih dogodkov sem akterje izpisala poudarjeno svetlo modro, ključne glagole, ki predstavljajo aktivnosti akterjev, pa poudarjeno oranžno, kar naredi podrobne opise primerov uporabe, ki so osnova za izdelavo diagramov aktivnosti, bolj pregledne.

Diagram aktivnosti

Za razliko od razrednega diagrama, ki prikazuje, kdo (kateri razred ali razredi) je povezan (asociacija in generalizacija) s kom (ostali razredi), in kaj lahko vsak naredi (operacije posameznih razredov), se diagram aktivnosti osredotoči na tok obnašanja (ne zanima ga toliko, kdo ga izvaja).
Uporabljamo ga lahko za različne namene:

- prikaz zaporedja akcij tekom izvrševanja operacije,
- podroben vpogled v primer uporabe (za ta namen sem ga uporabila pri praktičnem delu diplomske naloge),
- modeliranje poslovnih procesov.

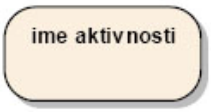
Osnovni elementi

Diagram aktivnosti vsebuje naslednje osnovne elemente :

Akcija (*action*) je osnovni gradnik obnašanja. Določimo lahko predpogoje in končne pogoje akcije, ki morajo biti izpolnjeni, preden (oz. potem, ko) se akcija izvede. V smislu programiranja je akcija atomaren in kratek izračun, običajno prireditev vrednosti spremenljivke ali preprost aritmetičen izraz. Lahko pa je tudi pošiljanje signala drugemu objektu, klic operacije, določitev izhodne vrednosti, ustvarjanje ali uničenje objekta. V smislu opisovanja splošnega procesa je akcija aktiven korak v izvrševanju tega procesa.
UML notacija za akcijo je zaobljen pravokotnik, znotraj katerega napišemo ime enostavnega obnašanja (slika 6).

Slika 6: Notacija za akcijo

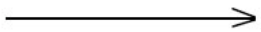
Aktivnost (*activity*) vsebuje zaporedje akcij in/ali drugih aktivnosti– z njo prikažemo niz povezanih akcij. Z aktivnostjo lahko (na nivoju objektno usmerjenega razreda) predstavimo metodo operacije ali pa naloge, ki tvorijo poslovni proces.
Na diagramu jo prikažemo kot zaobljen pravokotnik (slika 7), ki vsebuje ime aktivnosti (enako kot pri akciji), lahko pa tudi kompleksno zaporedje akcij, aktivnosti, objekte in krmilni tok. Prav tako je možen prikaz parametrov, predpogojev (*precondition*), končnih pogojev (*postcondition*) in lastnosti aktivnosti .



Slika 7: Notacija za aktivnost

Potrebna je pazljivost pri razlikovanju pojmov aktivnost in akcija, zlasti ker se v verziji 2.0 (za razliko od zgodnejših verzij) notaciji bistveno ne razlikujeta. Aktivnost je proces, ki se modelira (npr. pranje avtomobila), akcija pa korak znotraj aktivnosti (npr., miljenje, spiranje sušenje).

Krmilni tok (*control flow*) je usmerjena povezava med akcijami oz. aktivnostmi, ki s puščico ponazarja potek izvedbe.



Slika 8: Notacija za krmilni tok

Krmilni element (*control node*): krmilni tok vodimo med množico akcij in aktivnosti s krmilnimi elementi, ki jih je več vrst:

- **Začetni element (*initial*):** z njim začnemo niz aktivnosti in akcij, notacija je polni krog (slika 9).

Slika 9: Notacija za začetni element

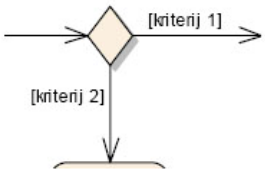
- **Končni element aktivnosti *(final activity)*:** z njim zaključimo vse krmilne tokove, prikazan je s koncentričnim krogom (slika 10).

Slika 10: Notacija za končni element aktivnosti

- **Končni element toka (*final flow*)** uporabimo takrat, ko želimo končati enega (in ne vseh) od tokov znotraj aktivnosti. Končni element toka nima nobenega vpliva na ostale tokove. Notacija je X znotraj majhnega kroga (slika 11).

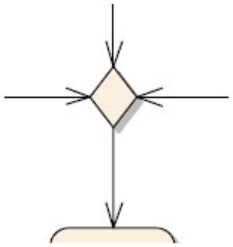
Slika 11: Notacija za končni element toka

- **Element odločitve (*decision*)** uporabimo takrat, ko želimo izvesti *if-then-else* selekcijo za potek izvedbe. Na ta način naredimo test, ki zagotavlja, da poteka krmilni oz. objektni tok samo po eni poti. Notacija je romb, ki ga povežemo z vsako od izhajajočih aktivnosti oz. akcij s puščico krmilnega toka. Na linijo krmilnega toka napišemo kriterij odločitve v oglatih oklepajih (slika 12).



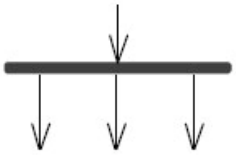
Slika 12: Notacija za element odločitve

- **Element združitve (*merge*):** z njim znova združimo različne izvedbene poti, ki so nastale z elementom odločitve (notaciji sta enaki, s tem da element združitve ne uporablja kriterijev odločitve v oglatih oklepajih – slika 13).



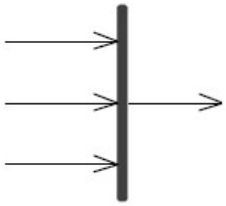
Slika 13: Notacija za element združitve

- **Element cepitve na vzporedne podaktivnosti (*fork*)** razcepi krmilni tok na dva ali več krmilnih tokov, ki se izvajajo neodvisno med seboj in vzporedno. Notacija je poudarjena črta (t.i. sinhronizacijska črta), na katero se na eni strani pripenja ena linija (krmilni tok), na drugi strani pa iz nje izhaja več linij -krmilnih tokov (slika 14).



Slika 14: Notacija za element cepitve na vzporedne podaktivnosti

- **Element združitve vzporednih podaktivnosti (*joins*)** je nasprotje predhodnega elementa – dva ali več krmilnih tokov sinhronizira v enega. Tudi tu je notacija sinhronizacijska črta (slika 15).

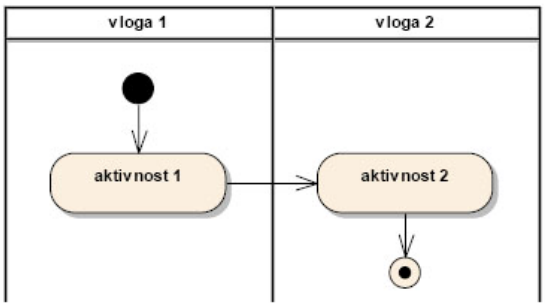


Slika 15: Notacija za element združitve vzporednih podaktivnosti

- **Povezovalni element** uporabimo, ko zmanjka prostora na diagramu in želimo nadaljevati krmilni tok na drugi lokaciji na diagramu ali na naslednji strani. Oznaka znotraj kroga nakazuje, da se tok nadaljuje tam, kjer se nahaja povezovalni element z enako oznako (slika 16).

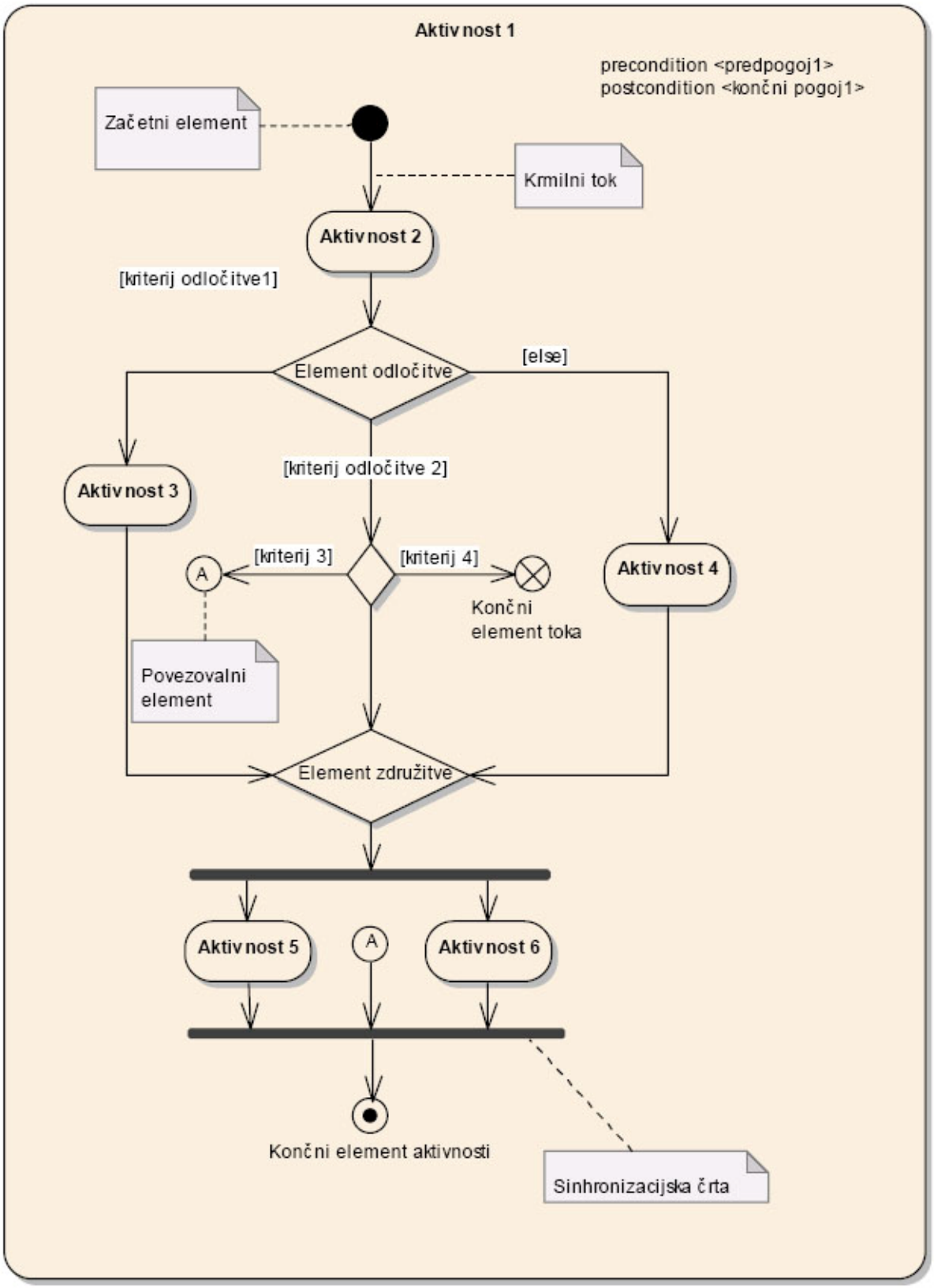
Slika 16: Notacija za povezovalni element

Steze (*swim lanes)* uporabimo, kadar želimo izrecno pokazati, kdo nosi odgovornost za posamezno aktivnost. Diagram aktivnosti vertikalno (lahko tudi horizontalno) razdelimo na vzporedne pasove – steze. Na vrhu (ali ob strani) vsake steze napišemo ime osebe, zaposlitveno vlogo ali organizacijsko enoto, ki je odgovorna za izvajanje aktivnosti na tem pasu. Aktivnosti in akcije pod odgovornostjo ene osebe (vloge, organizacijske enote) postavimo znotraj odgovarjajoče steze (slika 17).



Slika 17: Prikaz uporabe stez

Na sliki 18 je prikazana notacija za večino elementov diagrama aktivnosti.



Slika 18: Pregled notacije elementov diagrama aktivnosti

Viri:

- Chonoles, M.J., Schardt, J.A. 2003. UML 2 for Dummies. New York, Hungry Minds: 412 str.
- Eriksson, H.E., Penker, M., Lyons, B., Fado, D. 2004. UML™ 2 Toolkit. Indianapolis, Indiana, Wiley Publishing, Inc.: 511 str.
- Hamilton, K., Miles, R. 2006. Learning UML 2.0. Sebastopol, O'Reilly Media: 286 str.
- Rumbaugh, J., Jacobson, I., Booch, G. 2004. The Unified Modeling Language Reference Manual Second Edition. Boston, Addison Wesley: 752 str.
- Rumbaugh, J., Jacobson, I., Booch, G. 1999. The Unified Modeling Language Reference Manual. Boston, Addison Wesley: 568 str.
- Slokar, T. 2005. Objektno orientirano modeliranje informacijskega sistema operativnega procesa v centru vodenja elektrodistribucije. Magistrsko delo. Ljubljana, Univerza v Ljubljani, Ekonomska fakulteta: 132 str.
  http://www.cek.ef. uni-lj.si/magister/slokar546.pdf. (17.3.2006)
- UML Basics – The use case diagram. http://www-306.ibm.com/software/rational/uml/ (20.3.2006).