

Oblikovanje programske potpore

2012./2013. grupa P01

UML dijagrami

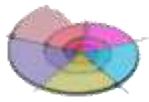
Prof.dr.sc. Vlado Sruk

Sveučilište u Zagrebu

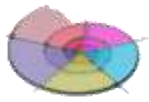
Fakultet elektrotehnike i računarstva

Zavod za elektroniku, mikroel., računalne i inteligentne sustave

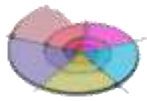




- Podsjetnik UML-a
 - obrasci uporabe
 - sekvencijski dijagrami
- UML dijagrami interakcija
 - dijagram kolaboracije
- UML dijagrami stanja



- Sommerville, I., ***Software engineering***, 8th ed., Addison-Wesley, 2007.
- Grady Booch, James Rumbaugh, Ivar Jacobson: ***Unified Modeling Language User Guide***, 2nd Edition, 2005
- OMG; ***OMG Unified Modeling Language, Superstructure Version 2.2*** ; URL:
www.omg.org/spec/UML/2.2/Superstructure/PDF
- Simon Bennett, John Skelton, Ken Lunn: ***Schaum's Outline of UML***, Second Edition, 2005



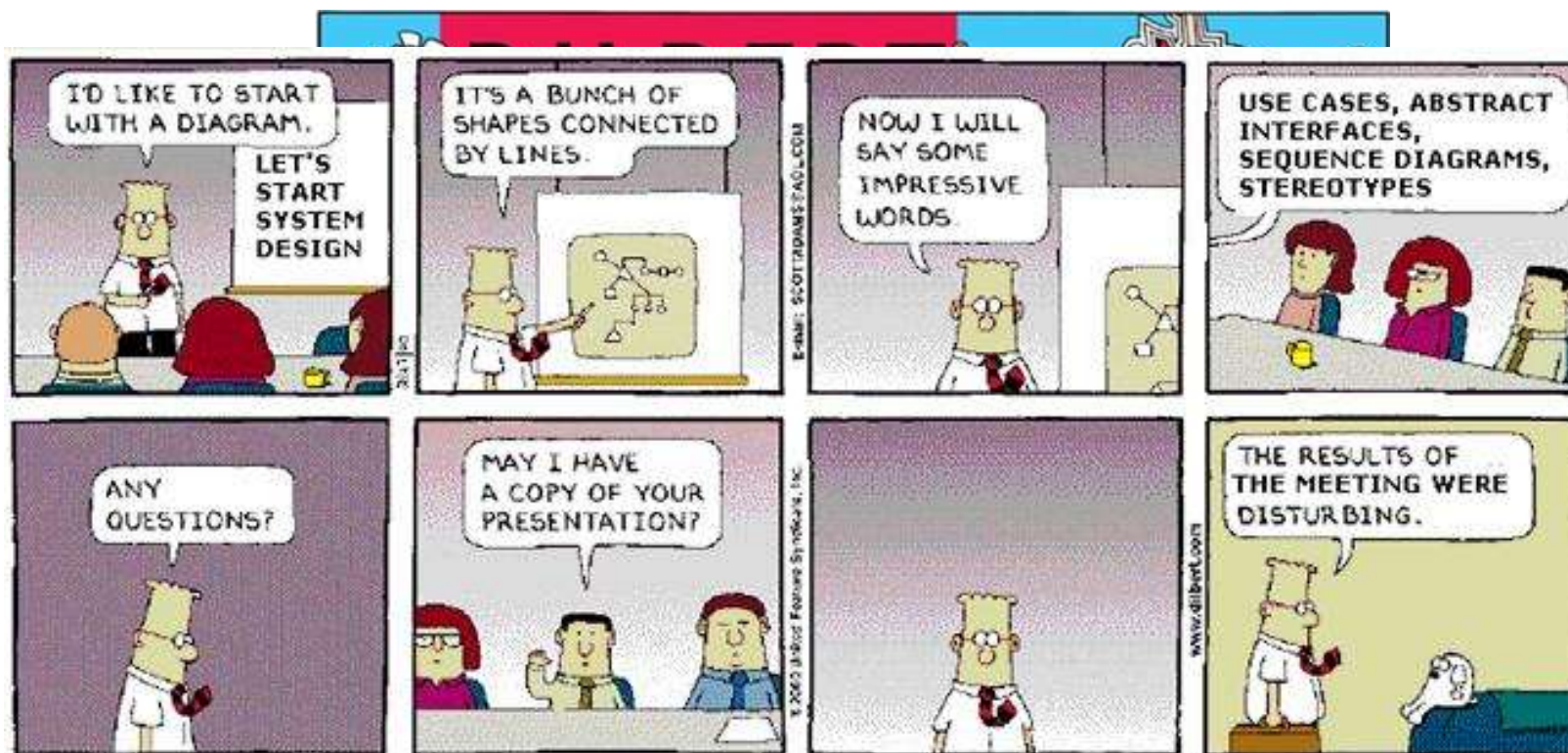
Unified Modeling Language



- Omogućava različite poglede na model
- 'de facto' standardni jezik programskog oblikovanja
- Omogućava
 - višestruke međusobno povezane poglede
 - poluformalnu semantiku izraženu kao metamodel
 - jezik za opis formalnih logičkih ograničenja



O UML-u



Copyright © 2000 United Feature Syndicate, Inc.



UML dijagrami

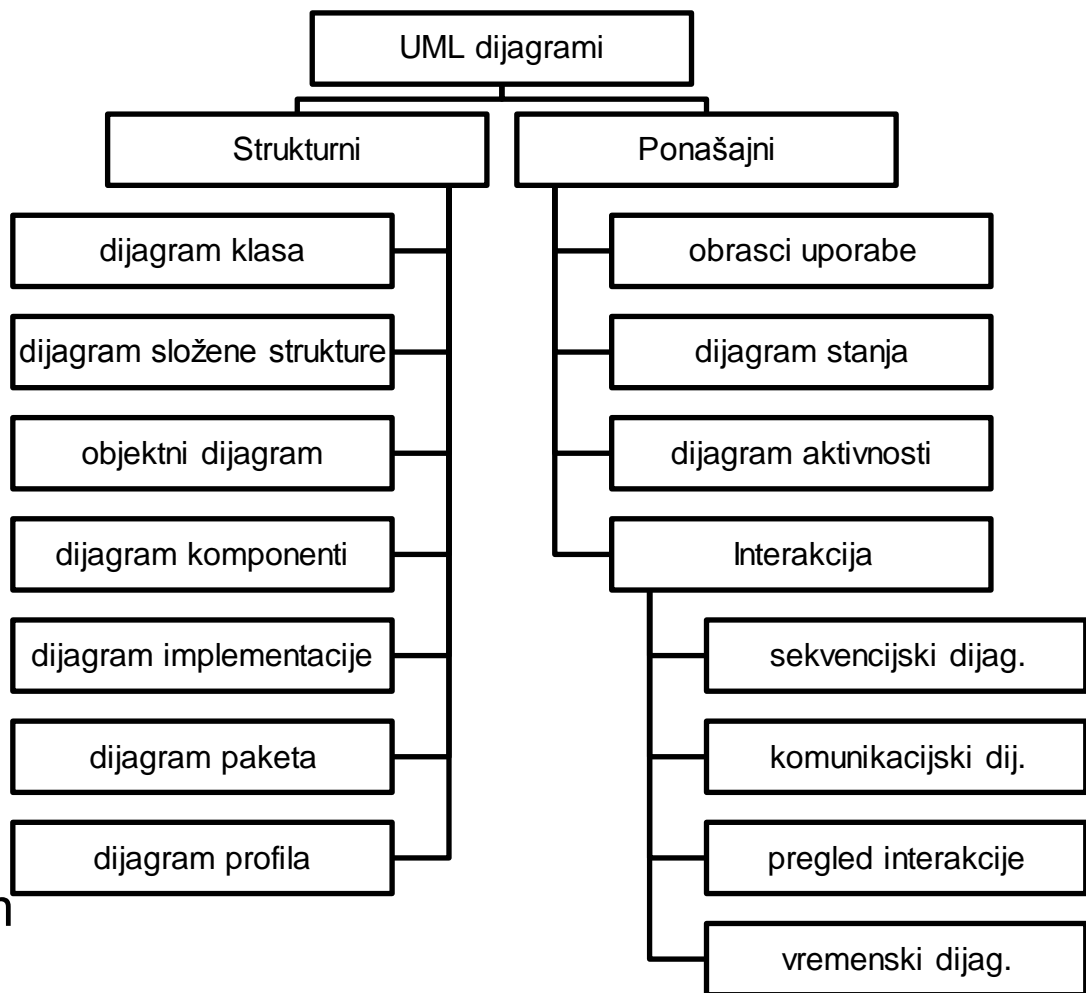


■ Pogled na model

- iz perspektive dionika
- djelomična reprezentacija sustava
- semantički konzistentan s ostalim pogledima

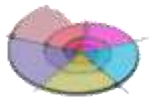
■ Specifičnost dijagrama

- svaki ima vlastitu sintaksu i semantiku
- evoluiraju tijekom procesa oblikovanja, te se mijenjaju donošenjem odluka o oblikovanju i proširuju novim detaljima





- **Obrasci uporabe** *engl. use case diagrams*
- **Sekvencijski dijagram** *engl. sequence diagrams*
- Dijagram komunikacije *engl. communication diagrams*
- Dijagram stanja *engl. state machine diagrams*
- Dijagram aktivnosti *engl. activity diagrams*
- Dijagram komponenti *engl. component diagrams*
- Dijagram implementacije *engl. deployment diagrams*
- Dijagram paketa *engl. package diagrams*
- Dijagram pregleda interakcije *engl. interaction overview diagrams*
- Vremenski dijagram *engl. timing diagrams*
- Dijagram profila *engl. profile diagram*
- Dijagram klasa *engl. class diagrams*
- Dijagram objekata *engl. object diagrams*
- Dijagram složene strukture *engl. composite structure diagrams*



- Bankomat
 - čitač mag. kartica
 - tastatura, zaslon
 - utor za umetanje omotnica
 - spremnik novac
 - printer za ispis potvrda
 - ključ za uključivanje/isključivanje
 - komunikacija s bankom



Opis rada



- Posluživanje jednog korisnika
- Ubacivanje kartice + identifikacija PIN-om, podaci se šalju banci na validaciju tijekom svake transakcije
- Korisnik može obaviti jednu ili više transakcija
- Kartica se zadržava u bankomatu sve dok korisnik obavlja transakcije, nakon završetka kartica se vraća (postoji iznimke)



Usluge bankomata



- Korisnik može podići novce s računa kartice. Podizanje novac odobrava banka.
- Korisnik može uložiti novac na račun kartice (gotovina/ček)
 - korisnik upisuje uloženi iznos
 - operator ručno verificira iznos
 - banka odobrava prihvatanje uplate
- Korisnik može prebacivati novce između računa
- Korisnik može pregledati stanje računa



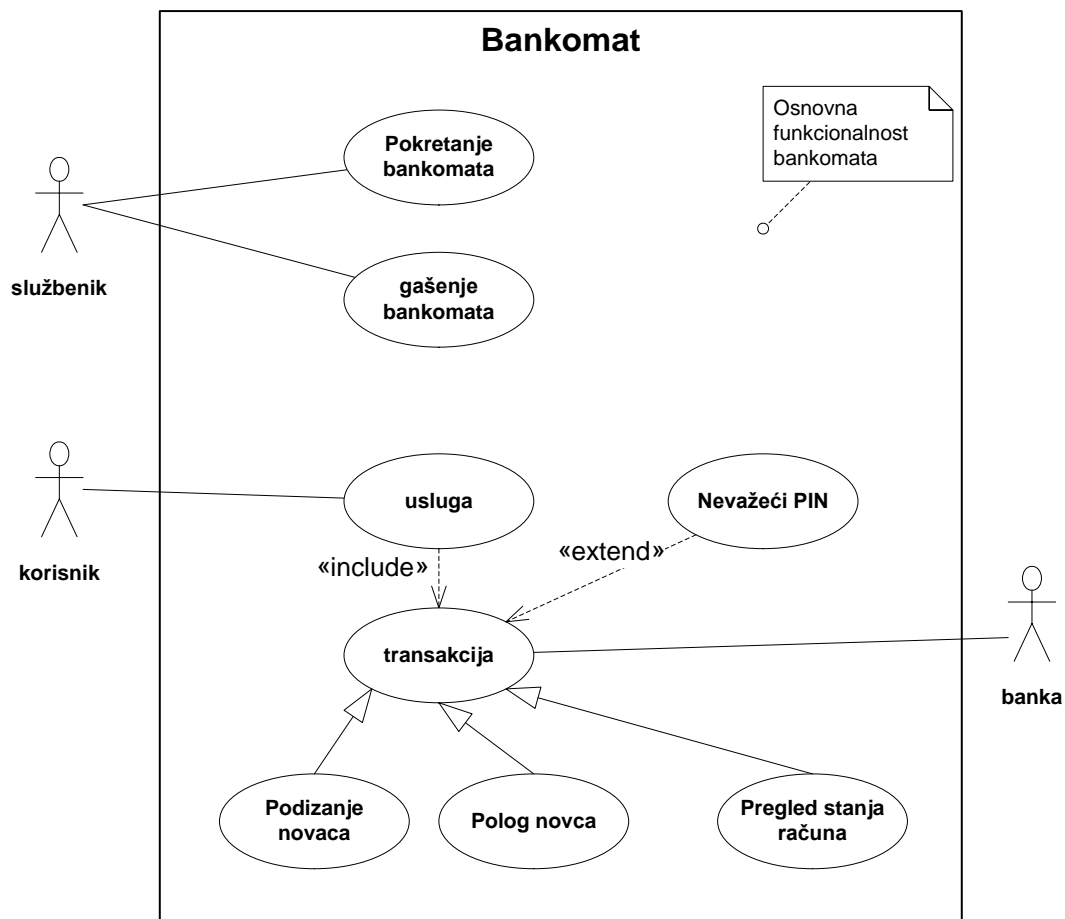
Usluge bankomata



- Korisnik može prekinuti započetu transakciju
- Bankomat komunicira s bankom za verifikaciju svake transakcije
- U slučaju unosa pogrešnog pina traži se ponovni unos. Nakon tri uzastopna pogrešna unosa kartica zadržava u bankomatu
- Bankomat ispisiuje poruke o pogreškama i pripadajuća objašnjenja prilikom neuspjelih transakcija i postavlja upit o slijedećoj transakciji
- Bankomat ispisiuje potvrdu svake transakcije
- Posluga bankomata može prekinuti rad s korisnicima
 - kada korisnik ne obavlja transakciju
 - kada je bankomat isključen moguće je umetnuti novac, ...
 - nakon uključivanja upisuje se iznos novca
- Bankomat interno zapisuje rad svih transakcija
 - nikada ne zapisuje PIN

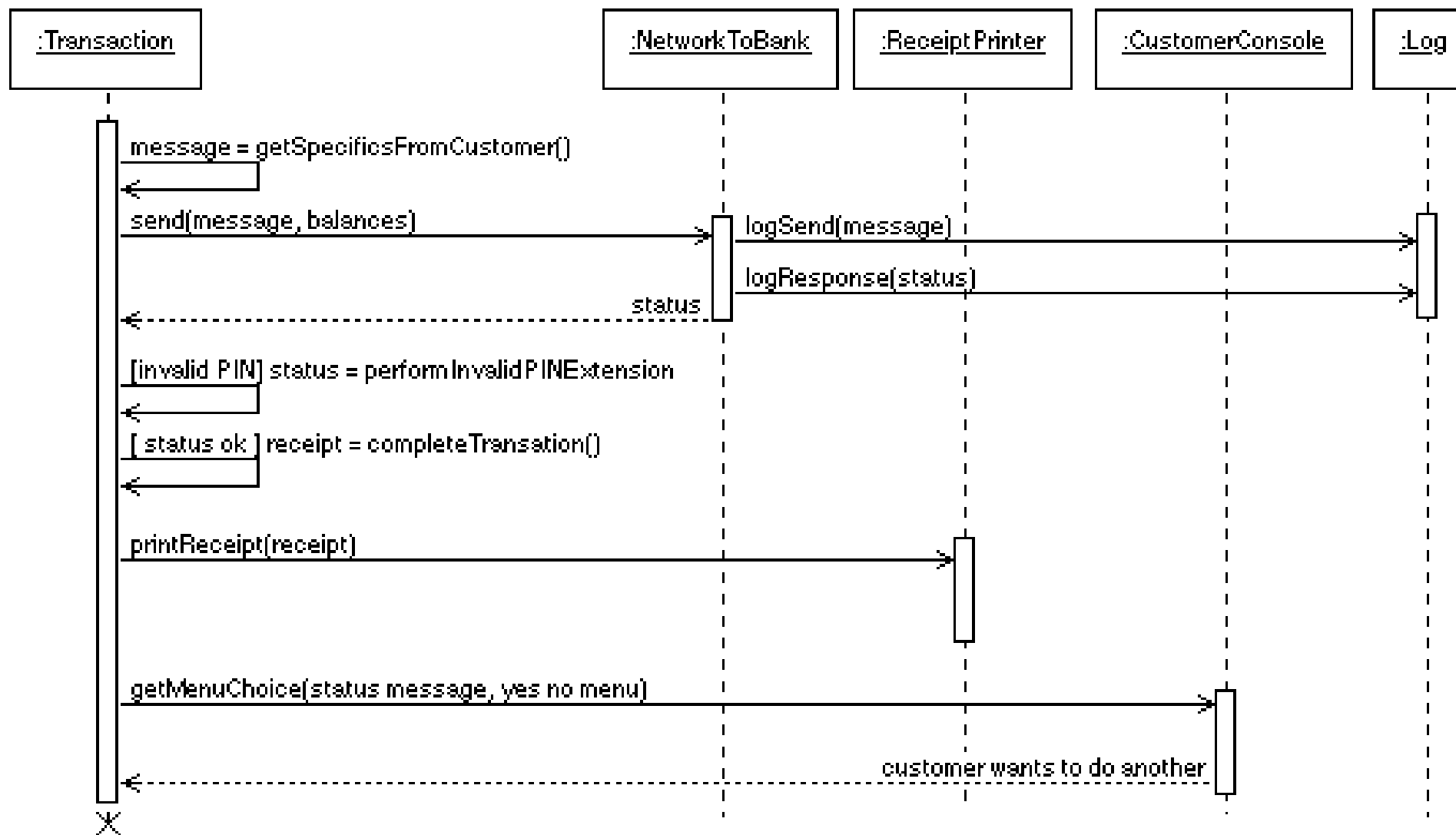


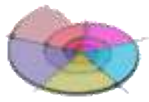
Obrazac uporabe



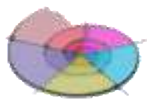


Sekvencijski dijagram transakcije





- Obrasci uporabe
- Sekvencijski dijagram
- ***Dijagram komunikacije***
- Dijagram stanja
- Dijagram aktivnosti
- Dijagram komponenti
- Dijagram razmještaja
- Dijagram paketa
- Dijagram pregleda interakcije
- Vremenski dijagram
- Dijagram profila
- Dijagram razreda
- Dijagram objekata
- Dijagram složene strukture



- Prikaz interakcija instanci modela
 - grafički prikaz instanci i podražaja
 - stvaranje i brisanje instanci
- Interakcije se modeliraju ako se želi:
 - specificirati kako instance uzajamno djeluju.
 - identificirati sučelja (*engl. interfaces*).
 - raspodijeliti zahtjeve (*engl. distribute requirements*).
- UML dinamički dijagrami interakcija:
 - sekvencijski – **vrijeme**
 - eksplicitno uređenje vremenskih odnosa između podražaja
 - modeliranje sustava za rada u stvarnom vremenu
 - **dijagram komunikacije/kolaboracije** – struktura
 - upotreba za opis struktura
 - usredotočeno na efekte instanci

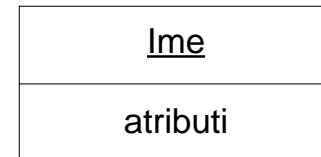


Osnovni elementi interakcija



■ Objekti – *engl. Objects*

- različite uloge



■ Veze - *engl. Associations*

- prikazuju povezanost objekata koji komuniciraju



■ Poruke – *engl. Messages*

- komunikacija između instanci
- struktura:

opis

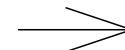


[broj_sekvence]{*[petlja]}{[uvjet]} [:]Ime(parametri) [: povratne vrijednosti]

[sequenceNumber]{*[loop]}{[condition]} {:} methodName(parameters)[:
returnValue]

- oznaka sekvence prema Deweyom sustavu. npr. 1.1, 1.2, 1.3
- asinkrone poruke

opis

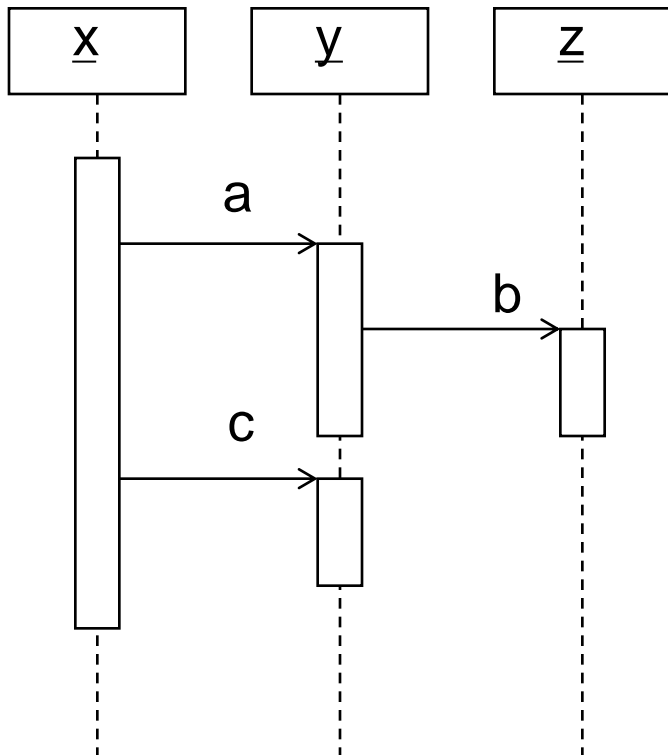




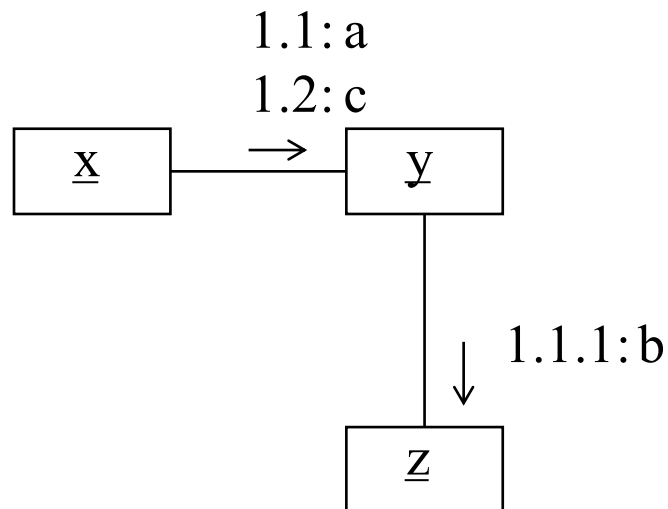
Dijagrami interakcija

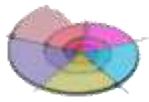


■ Sekvencijski dijagram



■ dijagram komunikacije





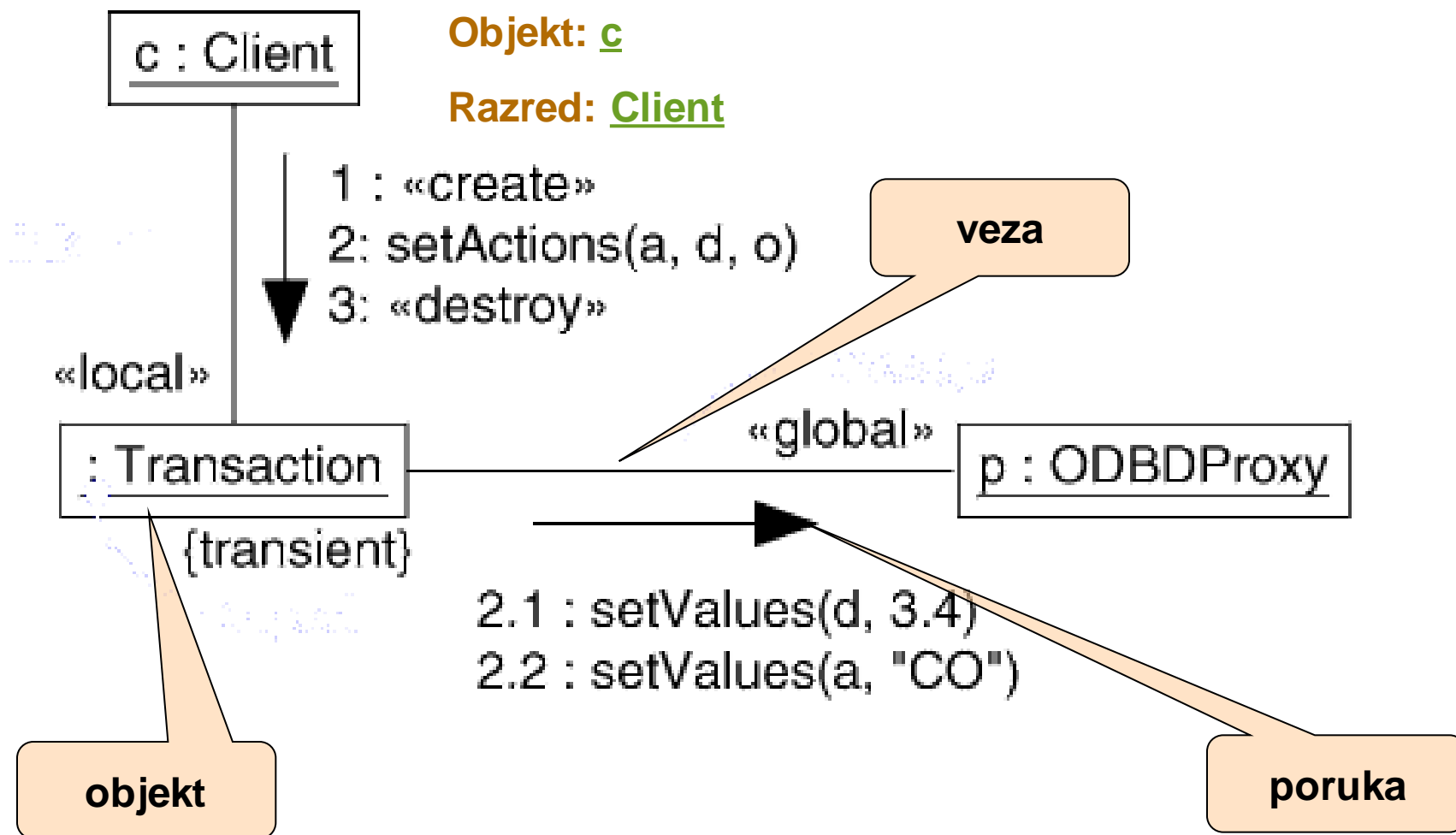
Dijagram komunikacije



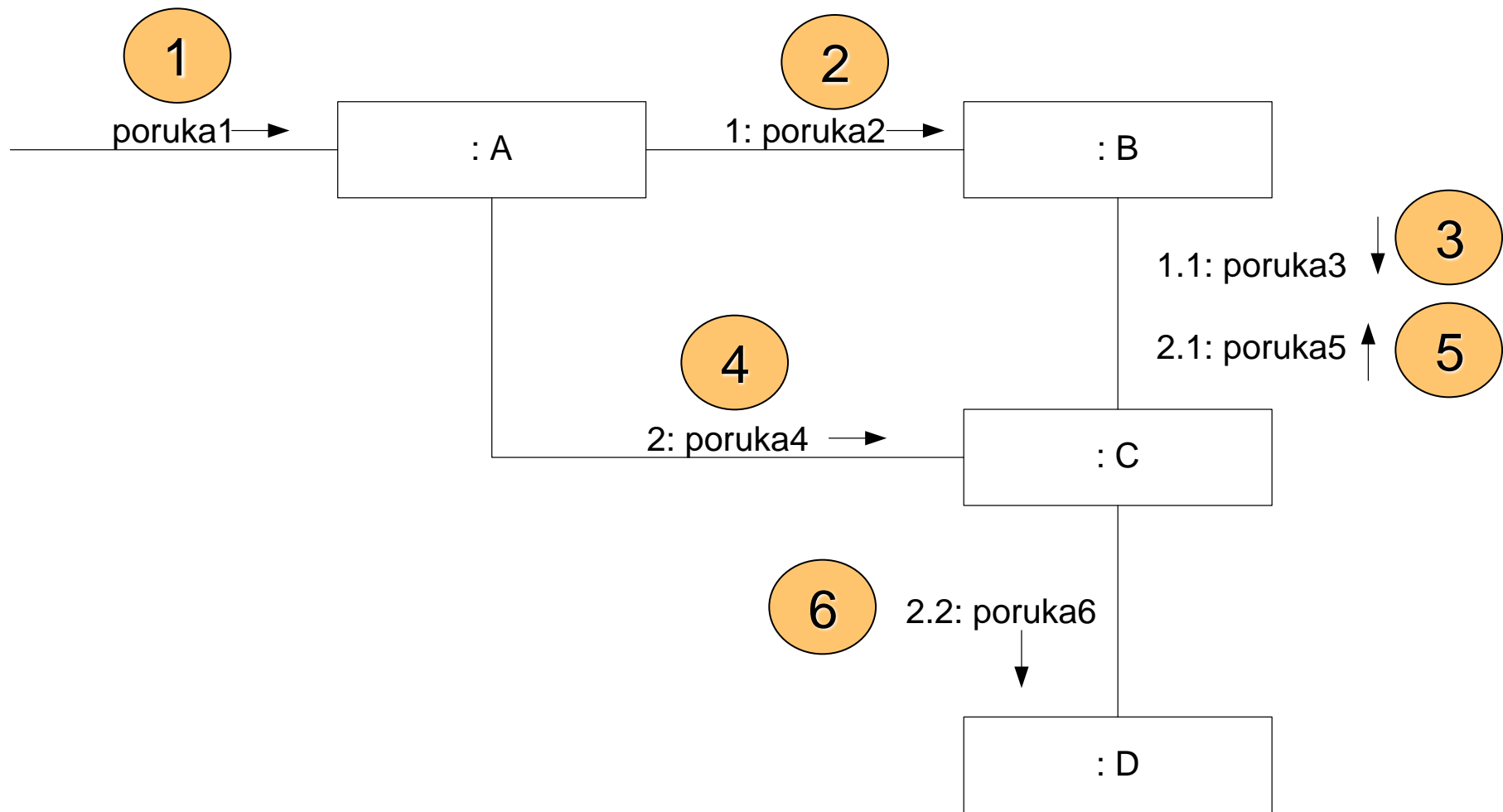
- *engl. Communication Diagrams*
- Prijašnje inačice = kolaboracijski dijagram, *engl. Collaboration Diagram*
- Obuhvaća dinamičko ponašanje
 - poruke – tko šalje kome; uređen redoslijed.
 - definira uloge instanci tijekom obavljanja nekog zadatka
 - ne prikazuje vremenske odnose.
- Modelira upravljački tok
 - prikaz koordinacije
 - nije izravno vidljiv
 - specificira tijek komunikacije između instanci tijekom kolaboracije



Primjer:



Primjer: Označavanje i redoslijed poruka





Primjer: Promjena rute leta



■ Aktori:

- putnik, baza računa klijenta (s planom puta), rezervacijski sustav avio kompanije.

■ Preuvjeti:

- putnik se prijavio na sustav i odabrao opciju “promjena leta”.

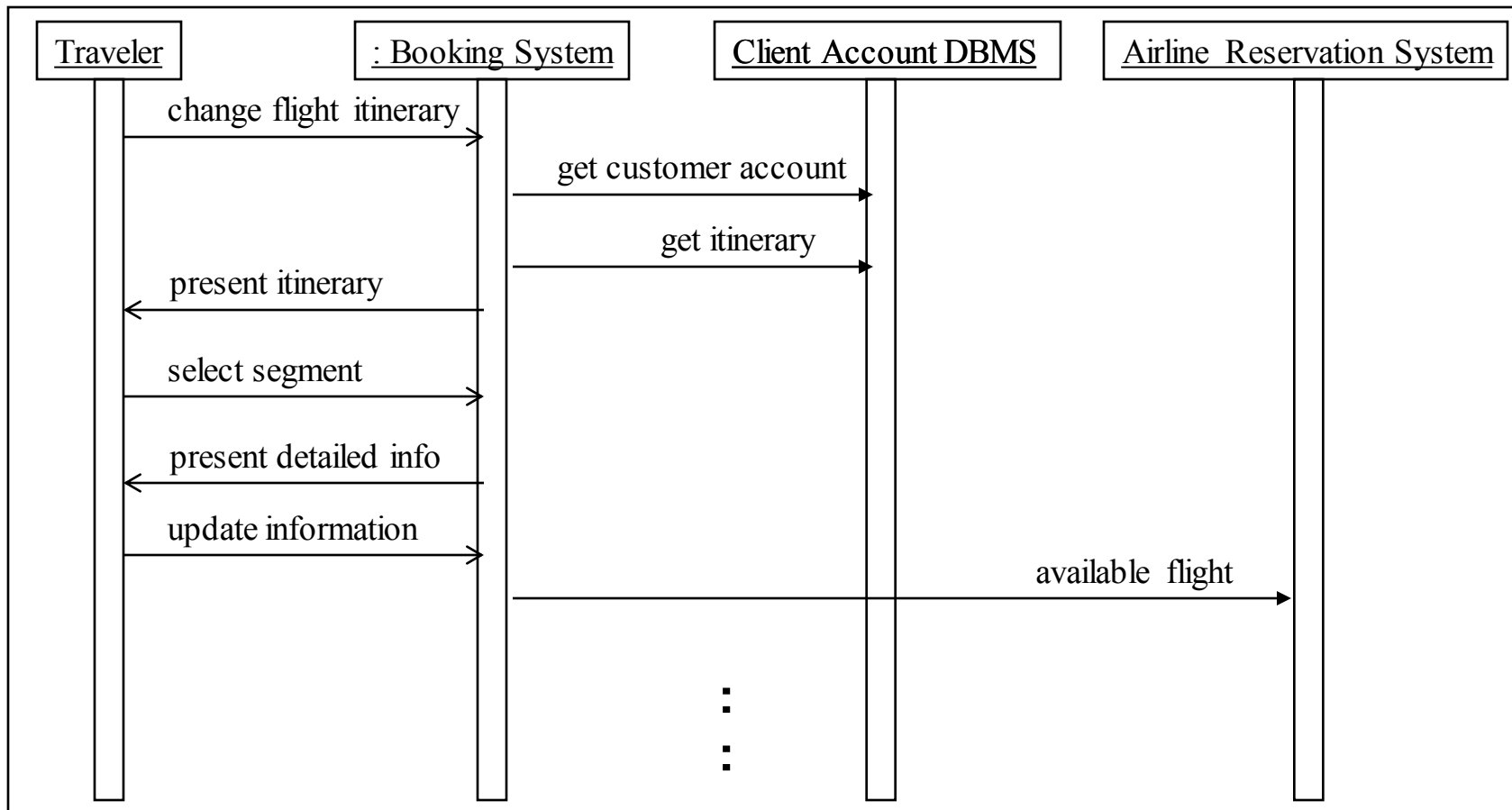
■ Temeljni tijek transakcija

- sustav dohvaća putnikov bankovni račun i plan puta iz baze.
- sustav pita putnika da odabere dio plana puta koji želi mijenjati; putnik selektira segment puta.
- sustav pita putnika za novi odlaznu i dolaznu destinaciju; putnik daje traženu informaciju.
- ako je let moguć, tada ...
- ...
- sustav prikazuje sažetak transakcije.

■ Alternativni tijek transakcija

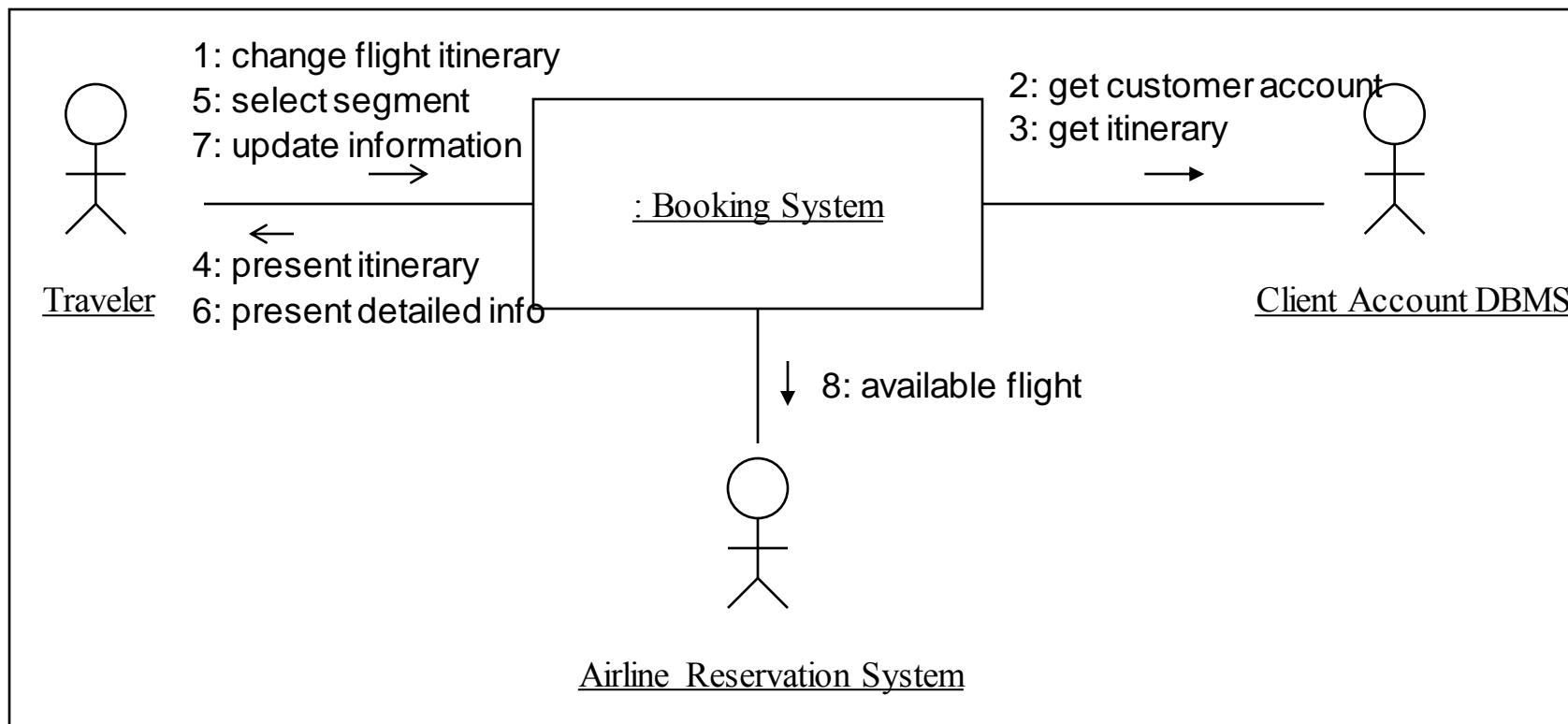
- ako let nije moguć, tada ...

Primjer: sekvencijski dij. promjene rute leta

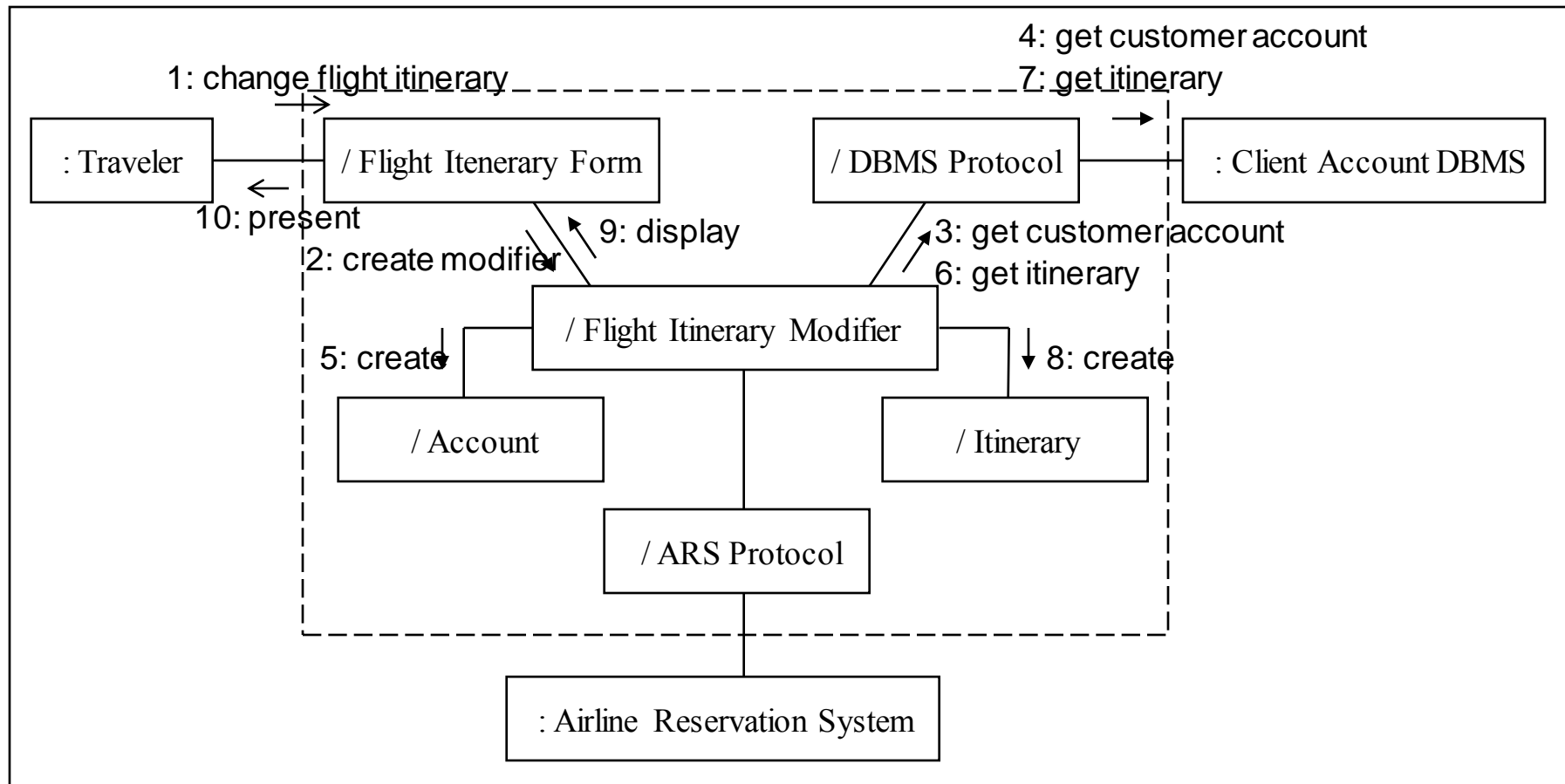




Promjene rute leta - dijagram komunikacije

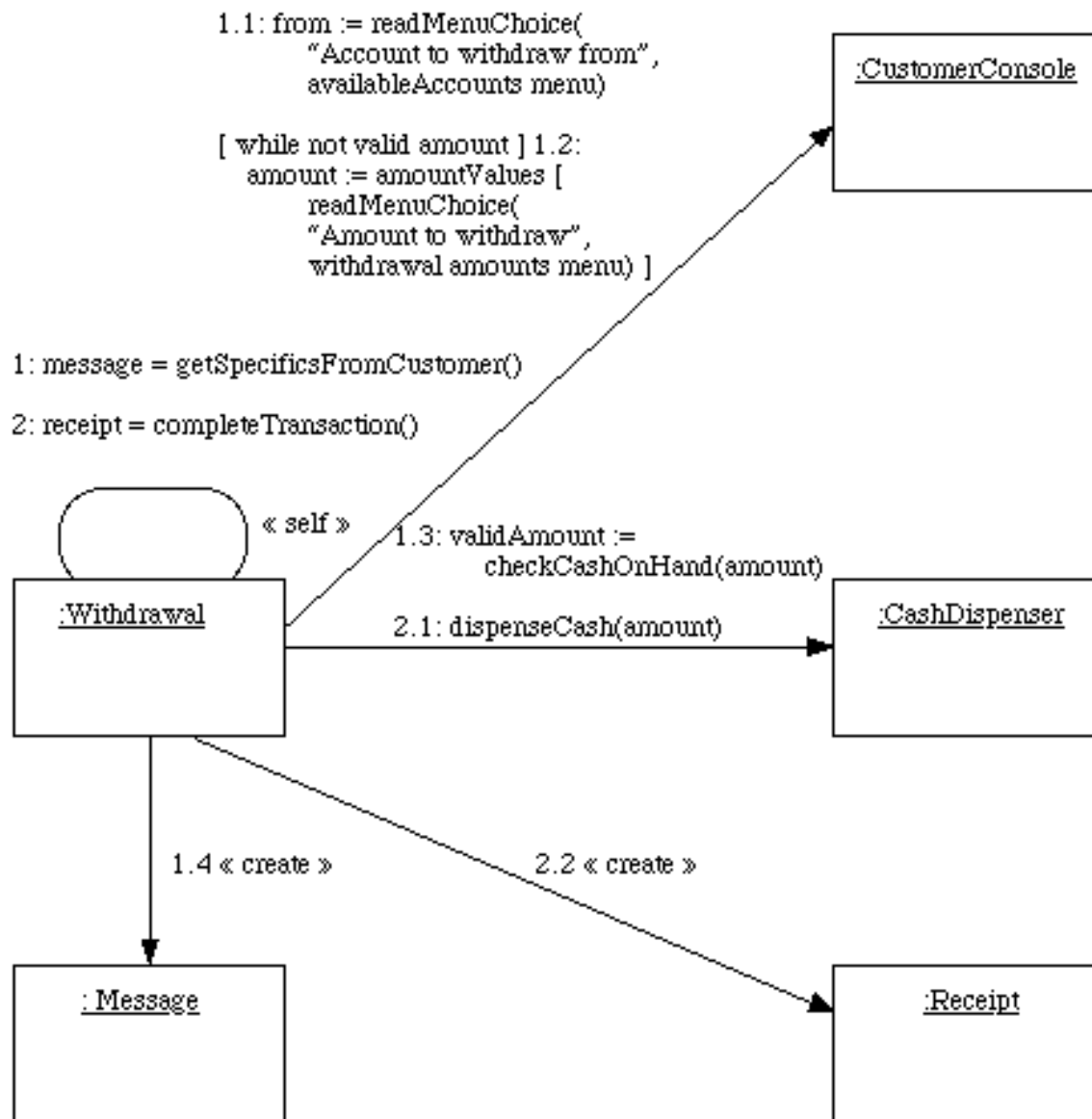


Promjene rute leta - dijagram komunikacije

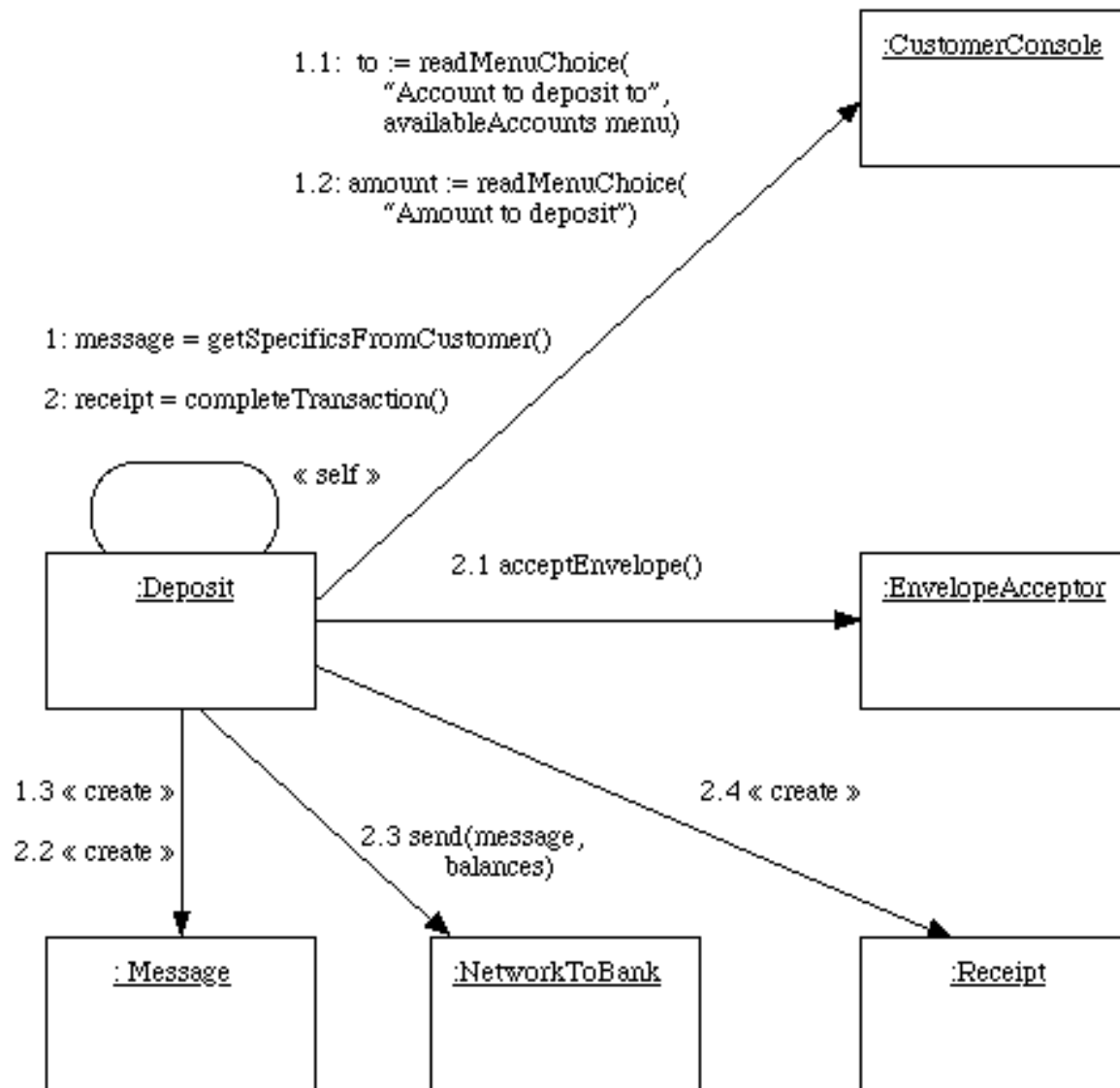




Primjer: Dijagram komunikacije



Primjer: Dijagram komunikacije pologa

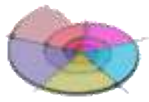




Preporuke



- Odaberite ime koje prikazuje namjenu
- Pri crtanju minimizirajte presijecanje linija
- Upotrijebite samo nužne elemente
- Važne elemente smjestite u centar dijagrama
- Započnite s porukom koja započinje interakciju i nastavite sistematično
- Jedan dijagram interakcije prikazuje samo jedan upravljački tok!!



- Obrasci uporabe
- Sekvencijski dijagram
- Dijagram komunikacije
- ***Dijagram stanja***
- Dijagram aktivnosti
- Dijagram komponenti
- Dijagram razmještaja
- Dijagram paketa
- Dijagram pregleda interakcije
- Vremenski dijagram
- Dijagram profila
- Dijagram razreda
- Dijagram objekata
- Dijagram složene strukture



- Nedostatak standardnih dijagrama stanja
 - velik broj stanja složenijih sustava, nepreglednost
- 1987. Harel, D.: Statecharts: A visual formalism for complex systems.
 - vizualni formalizam za opis stanja i prijelaza s naglaskom na modularnost, grupiranje, ortogonalnost, konkurentnost i poboljšanja
 - modeliranje jednog reaktivnog objekta
 - osnova UML dijagrama stanja s modifikacijom semantike i terminologije



Dijagrami stanja

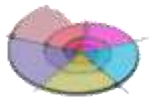
- *engl. State Machine Diagram*
- Opisuje dinamičko ponašanje jednog objekta u vremenu
 - pogodno za opis značajnijeg dinamičkog ponašanja objekta
 - objekt se promatra izolirano od ostalih
 - izlaz ne ovisi samo o trenutnim ulazima nego i o povijesti
 - pogodno za opis diskretnog ponašanja (*engl. discret-event*)
- Prikazuje sekvencu **stanja objekta** te **prijelaze** iz jednog stanja u drugo temeljene na **dogadjajima**
- Stanje objekta
 - opis stanja (okolnosti) u kojem se objekt nalazi kada zadovoljava određene uvjete
 - vrijednosti jednog ili više atributa objekta
 - u jednom stanju objekt može obavljati tri grupe aktivnosti:
 - **do** – aktivnosti koje se izvode za vrijeme dok je objekt u tom stanju
 - **entry** – aktivnosti koje se izvode pri ulasku u stanje
 - **exit** – aktivnosti koje se izvode izlasku iz stanja



- početno stanje – jedno
- krajnje stanje - može imati više njih

■ Prijelaz

- sve dozvoljene promjene stanja iz trenutnog u novo
 - novo stanje može biti to isto stanje
- inicirani su događajima i uvjetima
 - **[uvjet] događaj/akcija**
 - događaji:
 - interakcija
 - asinkroni prijem signala - primljene poruke – *engl. signal*
 - sinkroni poziv objekta – *engl. call*
 - vremenski – *engl. time*
 - proteklo vrijeme – istek vremenskog intervala
 - apsolutno vrijeme – def. trenutak, takt ...
 - ispunjeni uvjeti – *engl. change*
 - **moгу prenositi parametre!!**
- trajanje izvođenja prijelaza je 0 i ne može se prekinuti



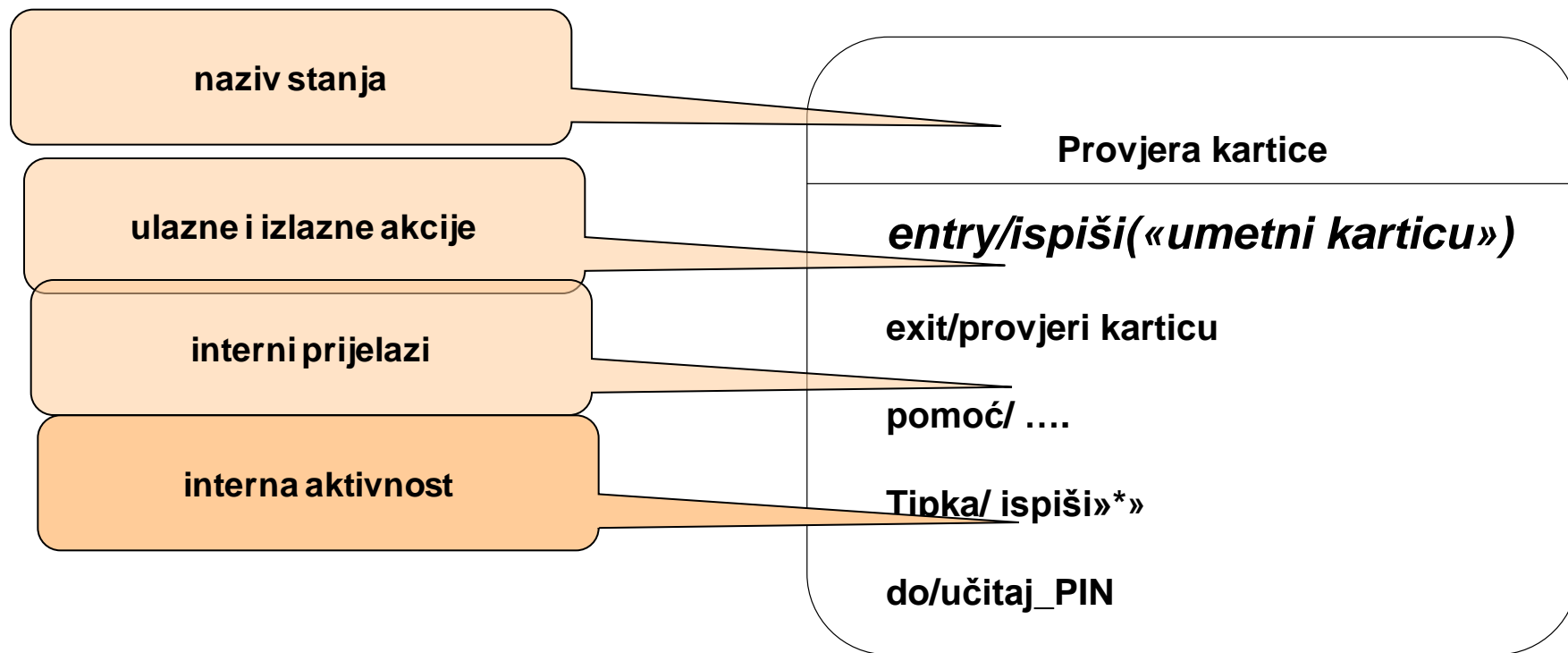
Aktivnost i akcije



- Aktivnost – *engl. Activity*
 - obavlja se sve dok je stanje aktivno
 - ovisi o dolaznim događajima
- Akcija – *engl. Action*
 - kratkotrajno, neprekidivo ponašanje



- Akcija:
 - naziv_događaja/Akcija
- Aktivnost
 - do/Aktivnost

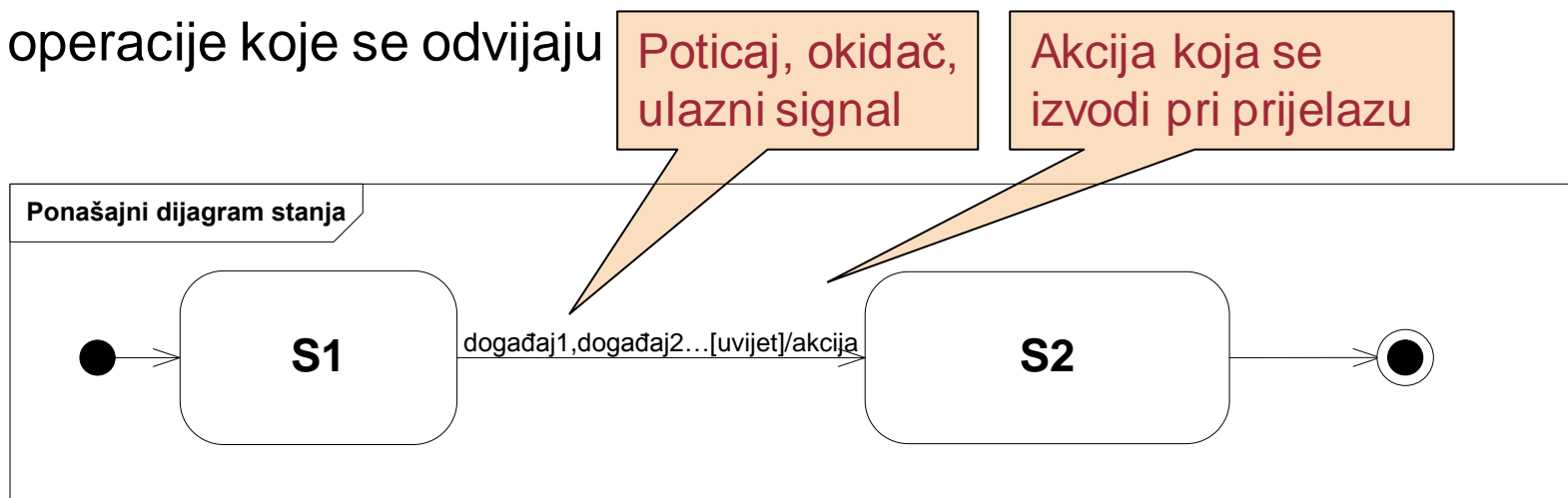




Prijelazi



- Ponašajni dijagrami- *engl. behavioral state diagrams*
- Događaji
 - vanjska ili unutarnja pojavljivanja događaja koja pokreću prijelaz
- Uvjet
 - izraz koji kada je ispunjen(istinit) dopušta odvijanje prijelaza (naravno uz pojavu događaja)
- Akcija
 - operacije koje se odvijaju

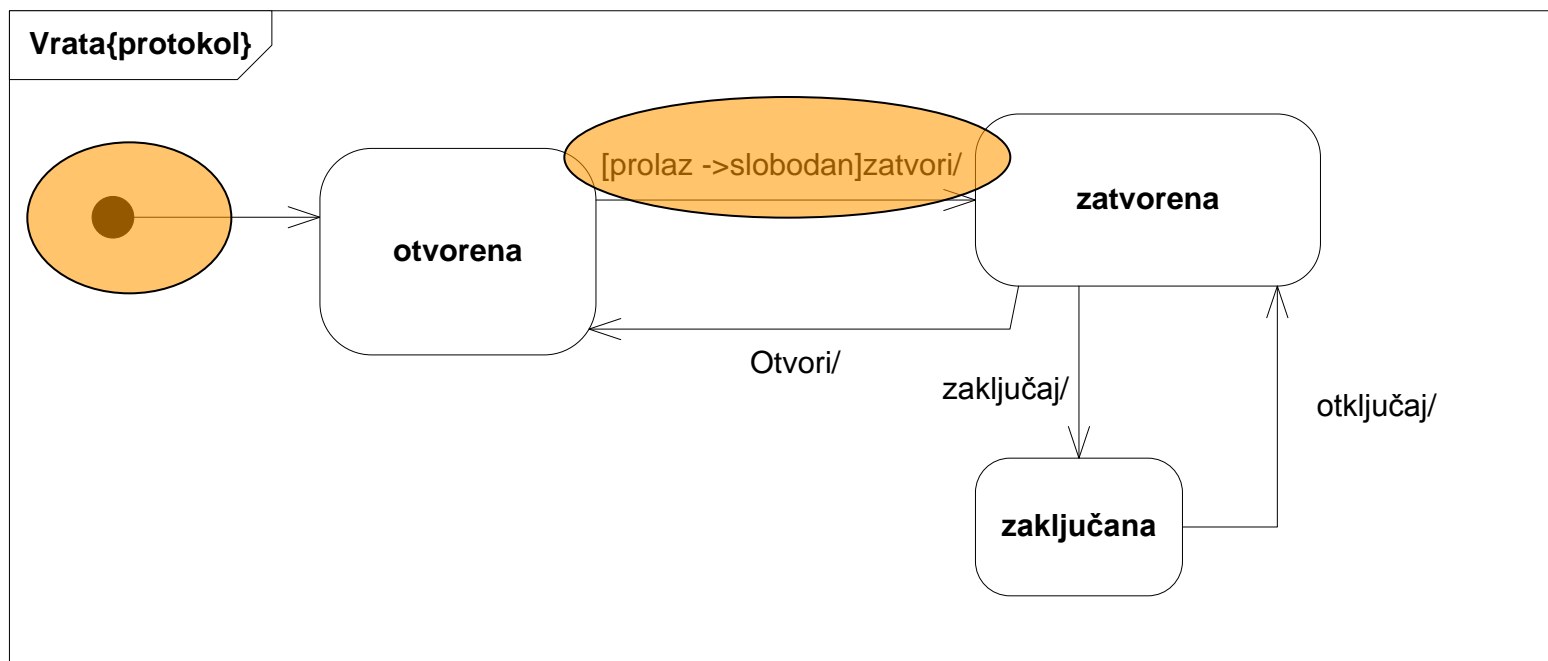




Primjer



■ Grafički prikaz ponašanja ulaznih vrata

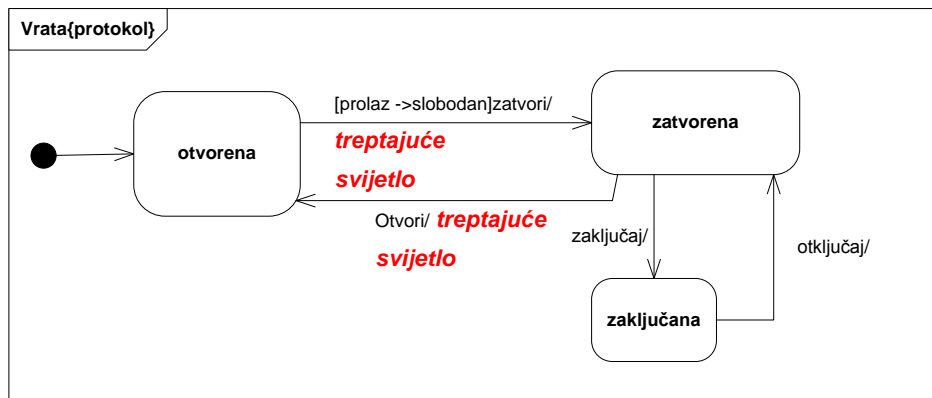




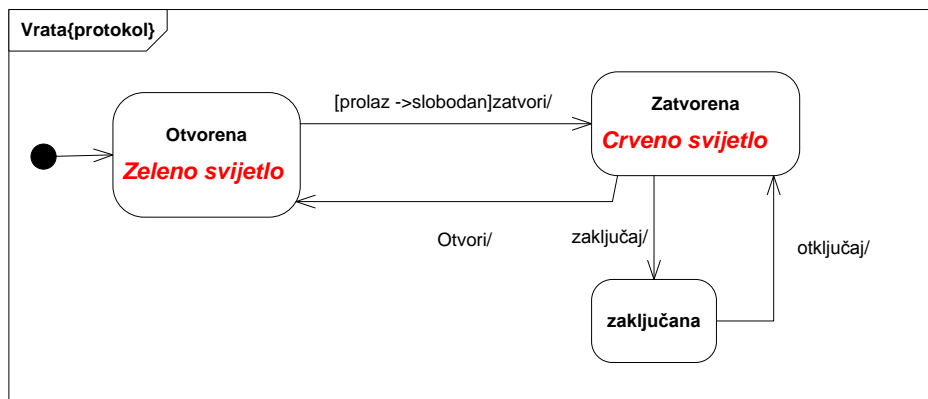
Promjena stanja i akcije

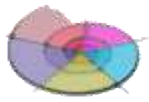


- Automati s izlazom - može generirati akciju
 - mealyev automat - izlaz je funkcija ulaza i stanja

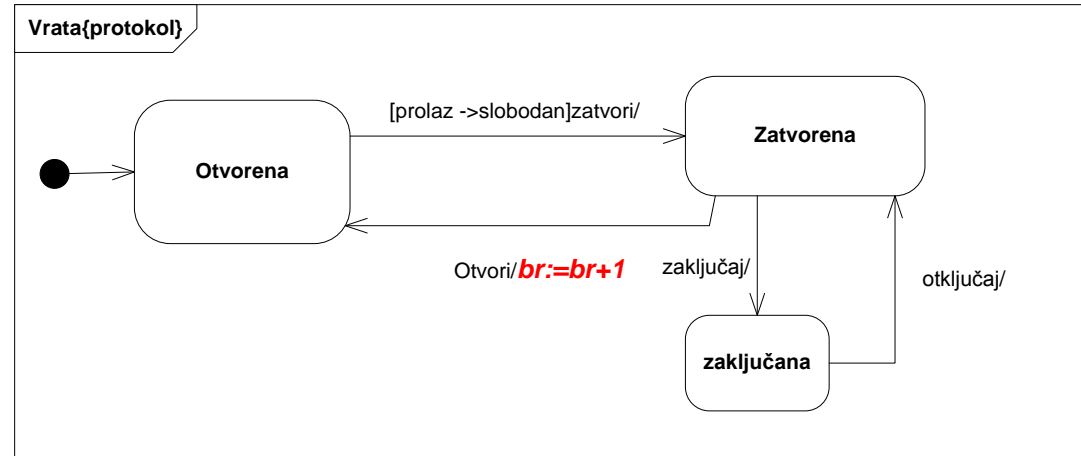


- mooreov automat - izlaz je funkcija stanja





■ dodatne varijable stanja



■ definiranje automata

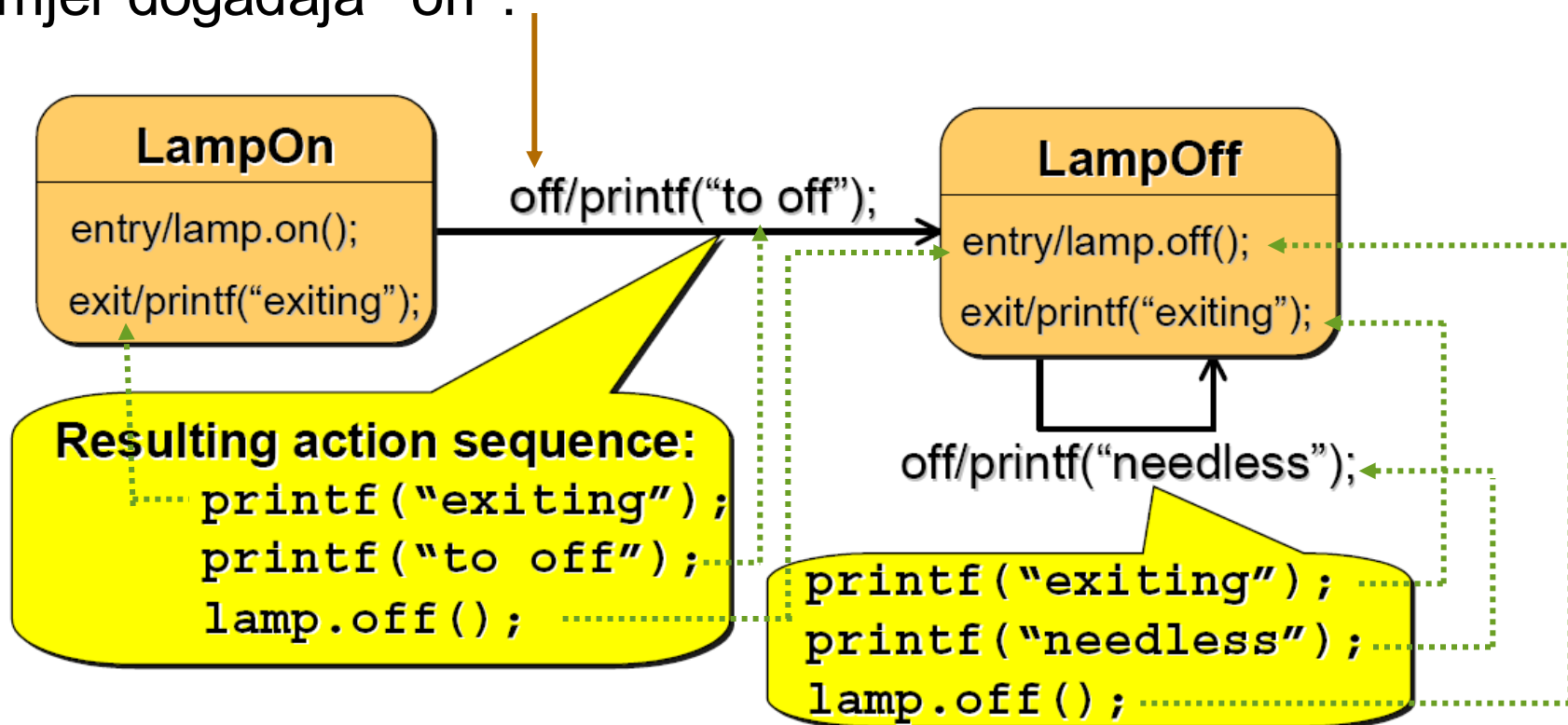
- konačan skup ulaznih signala
- konačan skup izlaznih signala
- konačan skup stanja
- skup prijelaza
 - Signali okidanja
 - Akcije koje se izvode pri prijelazu
- konačan skup proširenih varijabli stanja
- početno stanje
- skup konačnih stanja



Dijagrami stanja – slijed akcija



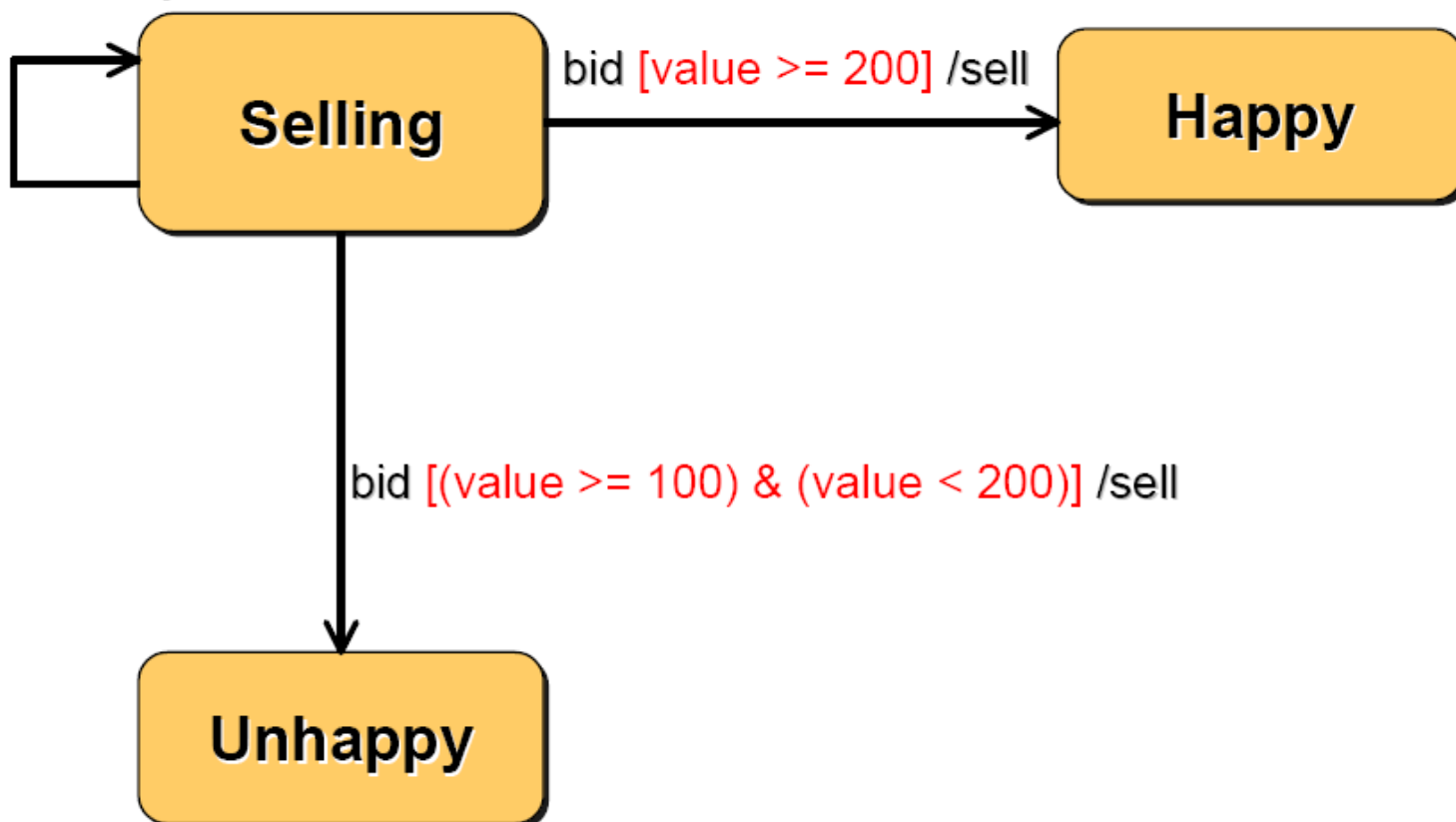
- Akcije pri izlazu iz stanja (izlazne akcije) izvode se prije akcije prijelaza.
- Akcije pri ulazu u stanje (ulazne akcije) izvode se nakon akcija prijelaza.
- Primjer događaja “off”:





- Uvjetno izvođenje prijelaza
- Npr: prodaja dionica kada je ispunjen uvjet

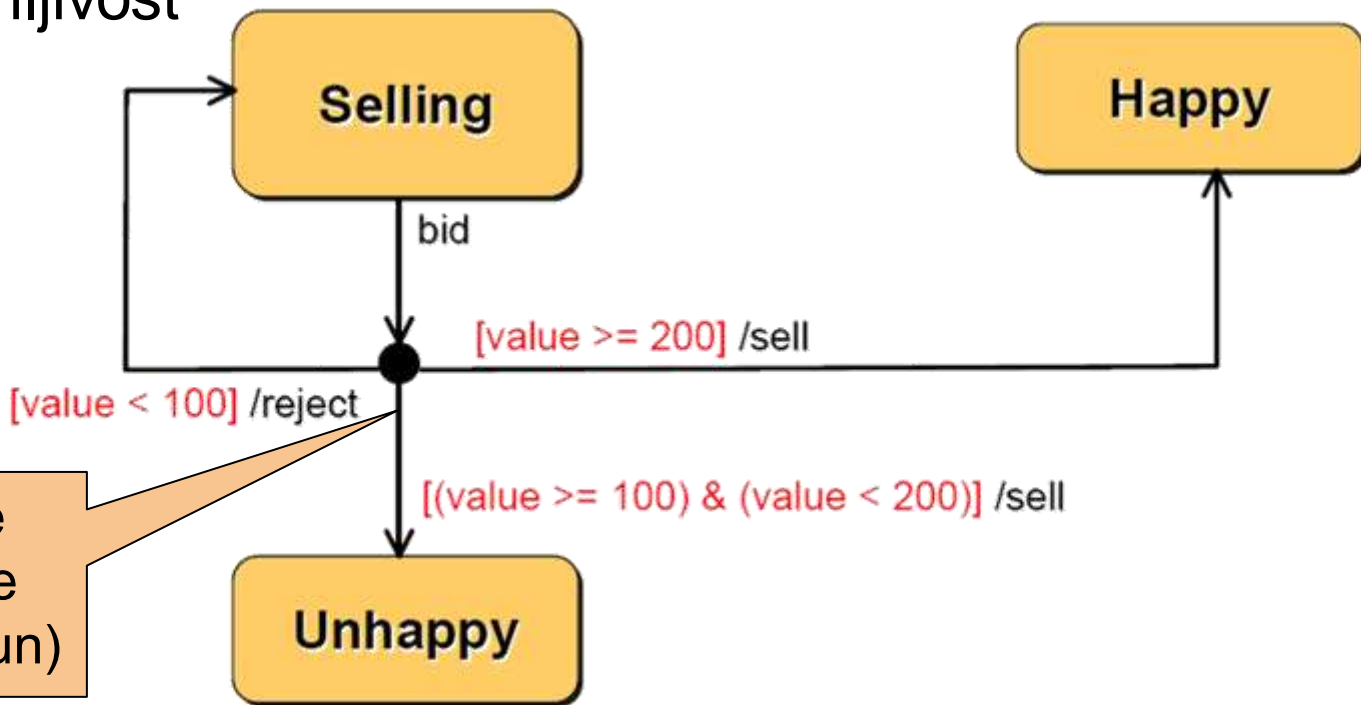
bid [value < 100] /reject





Uvjetna grananja

- Statičko uvjetovanje grananja
 - grafički prikaz odlučivanja
 - uvjeti su poznati prije grananja
- Pojednostavljivanje grafičkog prikaza
 - veća razumljivost

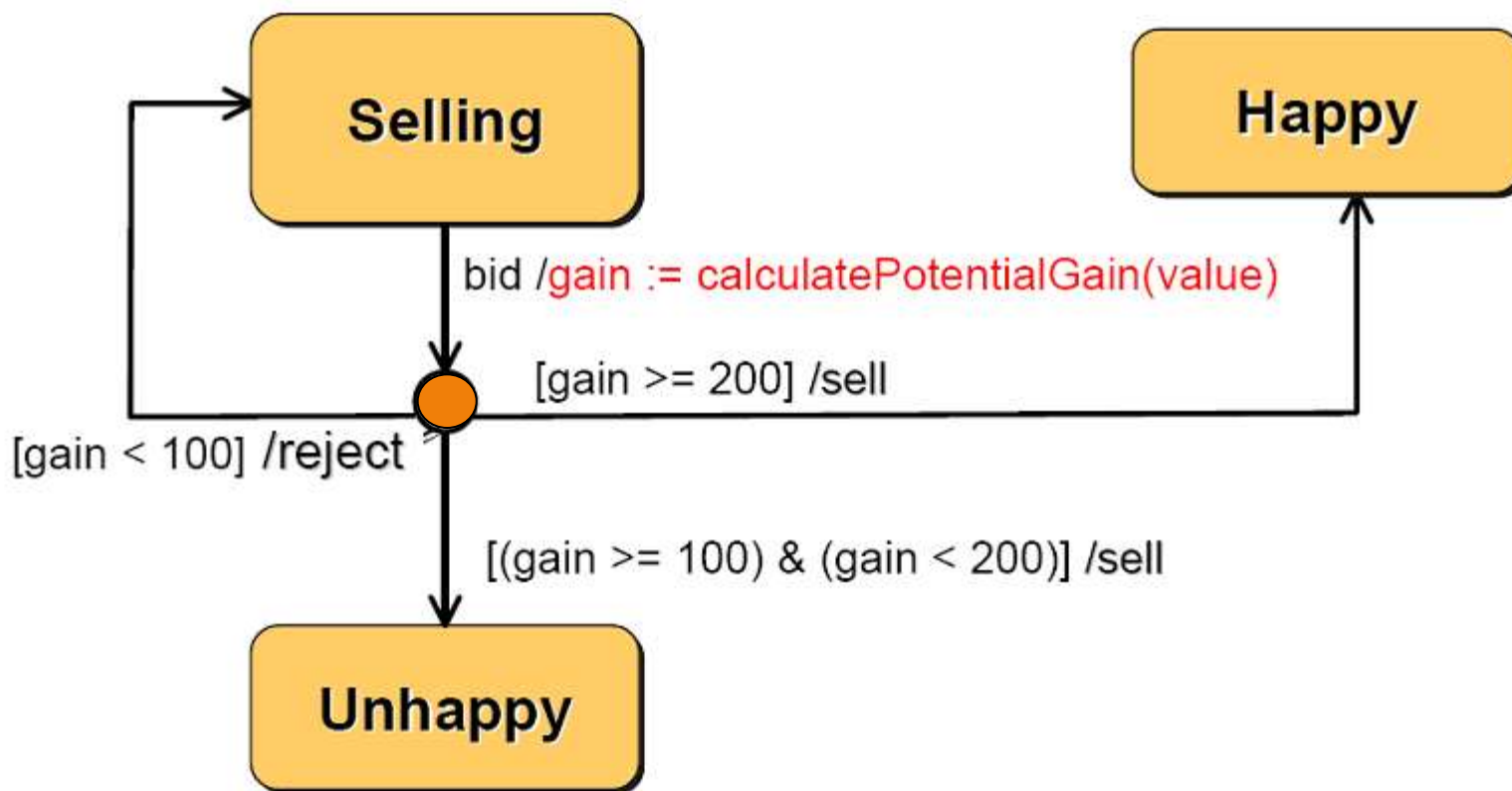




Dinamička uvjetna grananja



- Dinamičko uvjetovanje grananja
 - uvjeti se izračunavaju pri izlasku iz stanja.
 - Nisu poznati unaprijed





Pseudostanja UML dijagrama



- vrsta stanja u UML metamodelu koje predstavlja točke prijelaza unutar dijagrama stanja

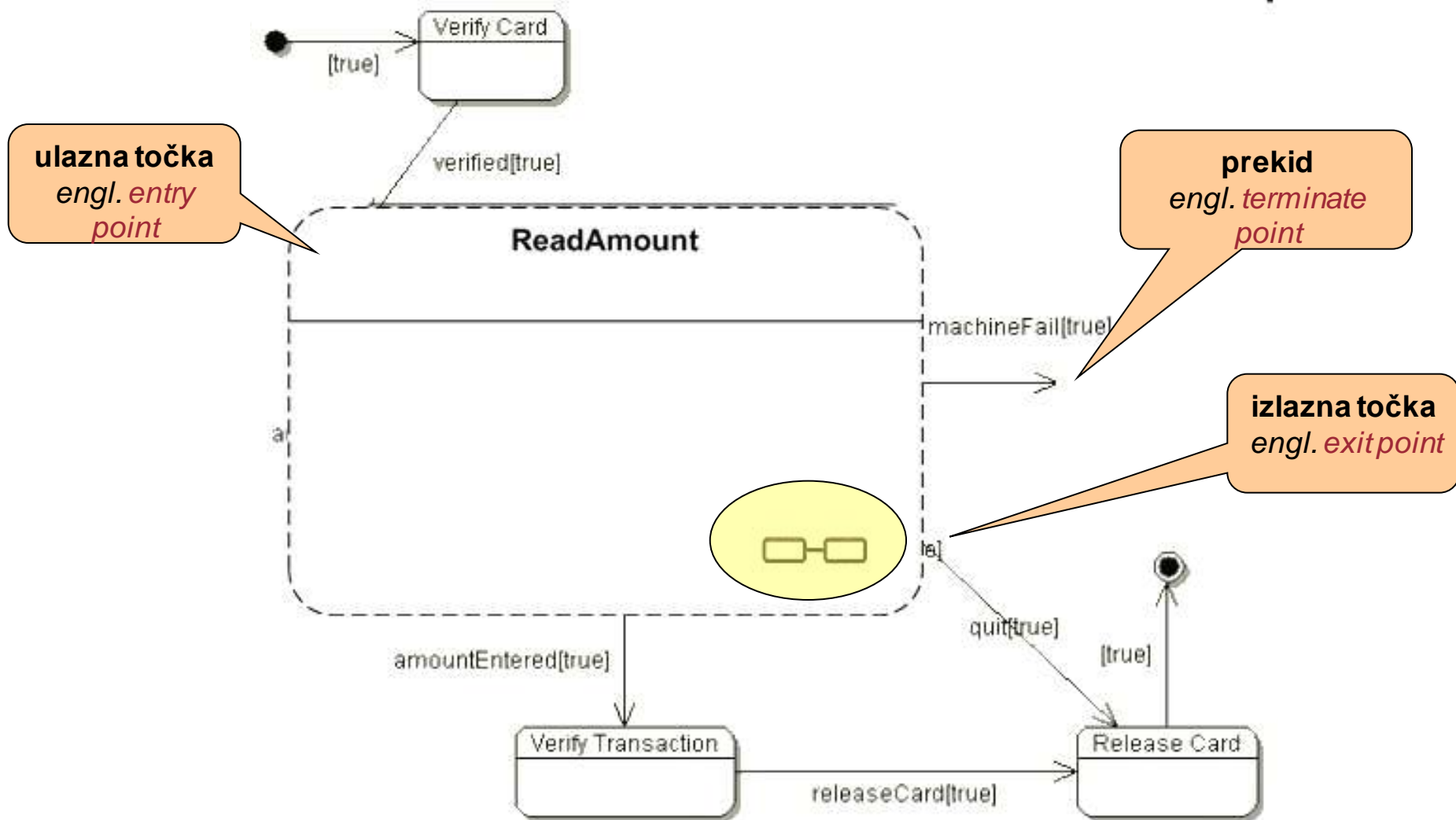
	Početno stanje - Initial State
	Završno stanje(a) - Final State
	Povijest - History
	Duboka povijest – Deep History
	Ulazna točka - Entry Point
	Izlazna točka - Exit Point
	Spajanje - Junction Pseudo-State
	Izbor-Choice Pseudo-State
	Završetak - Terminate Pseudo-State
	Grananje/Račvanje- Fork
	Spajanje/skupljanje - Join



Hijerarhija stanja

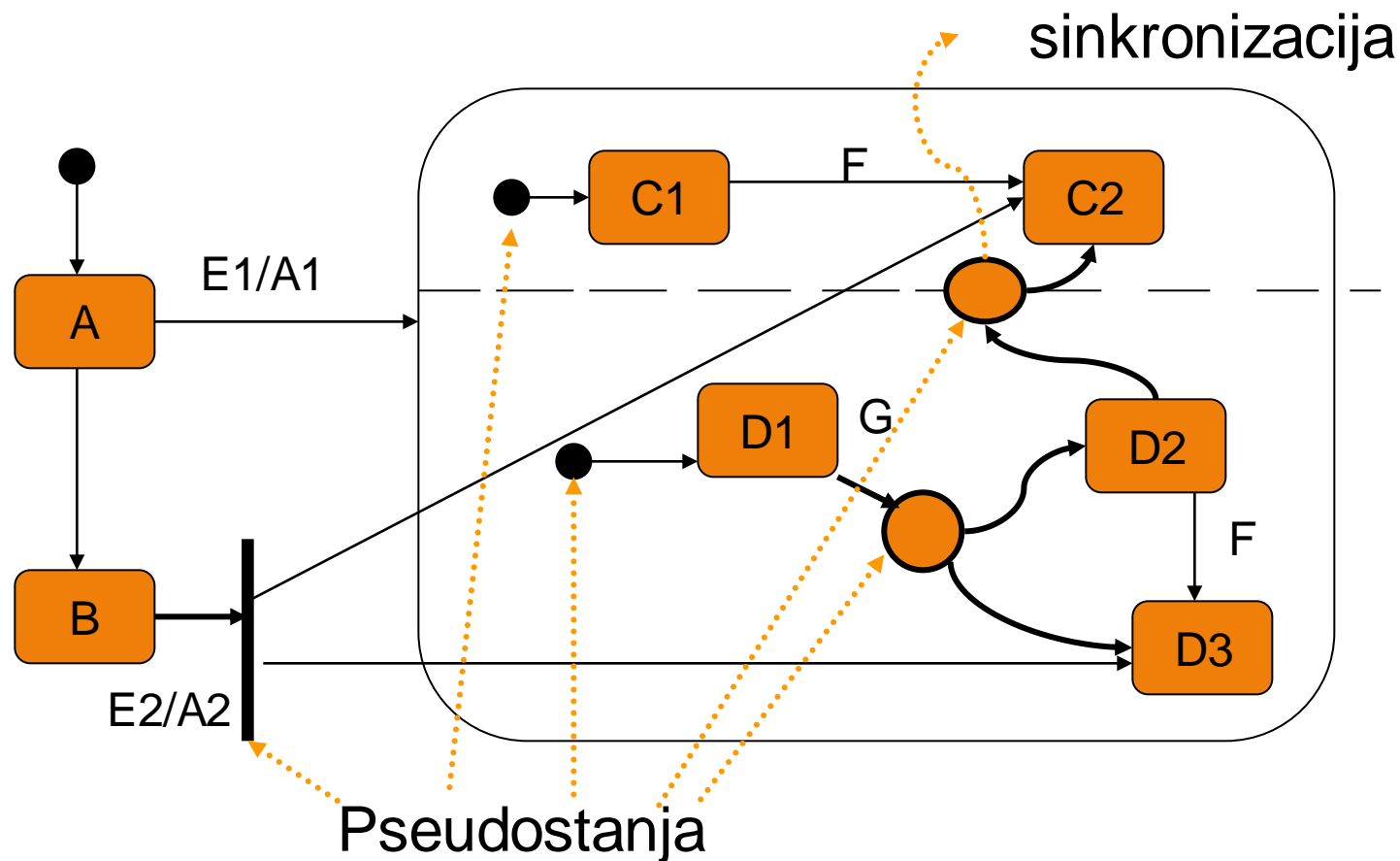


- Olakšava prikaz složenih problema





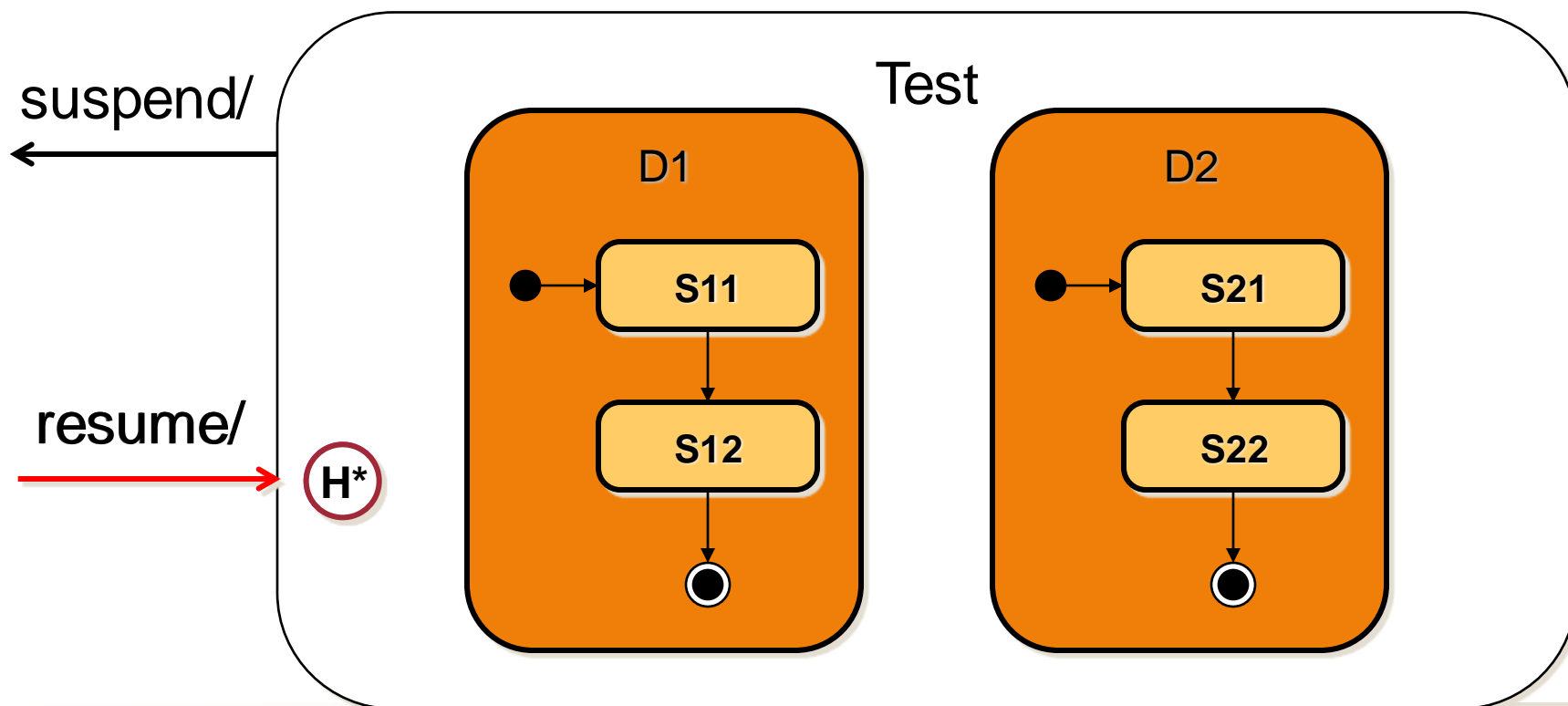
Primjer pseudostanja





■ Mogućnost povratka na prethodno stanje

- povijest – *engl. Shallow History (H)*
 - povratak na posljednje stanje na istoj razini
- duboka povijest - *engl. Deep History (H*)*
 - povratak na posljednje stanje (bez obzira na kojoj razini)

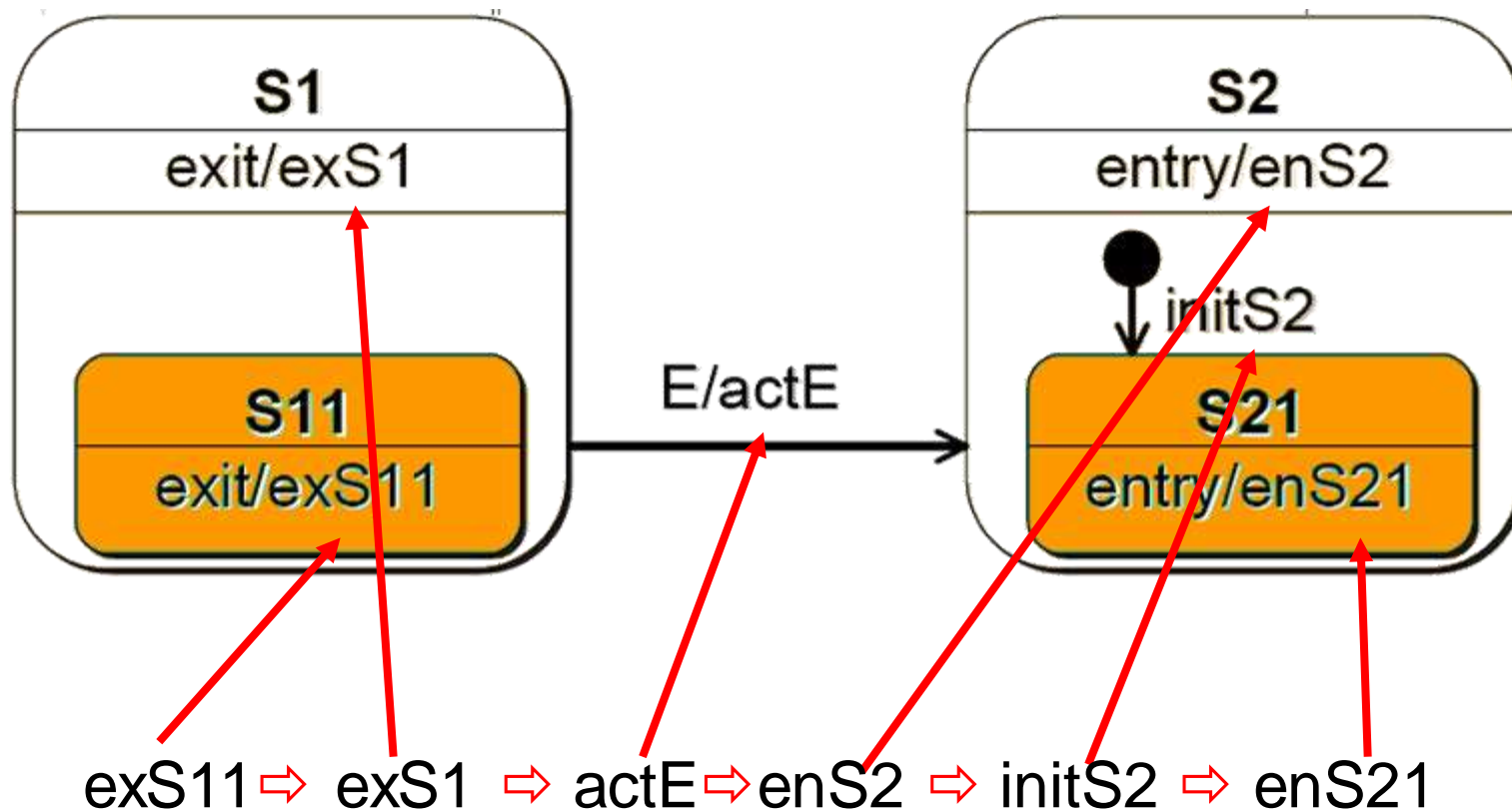




Prioritet akcija



- Ulaz: izvana prema unutra
- Izlaz: iznutra prema van

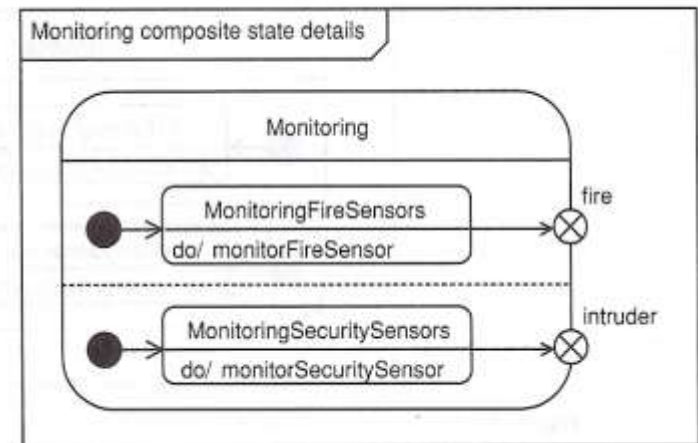
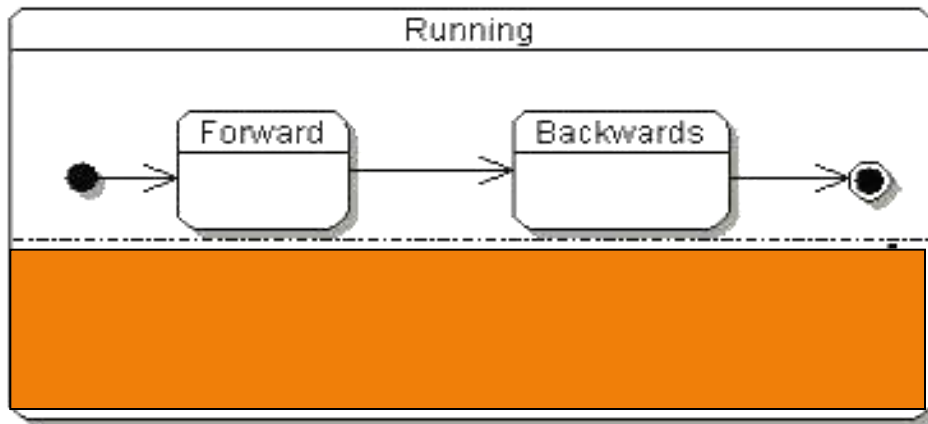




Konkurentna podstanja



- Jedno stanje može imati više konkurentnih podstanja
 - višestruka perspektiva jednog objekta
 - smanjuje broj stanja

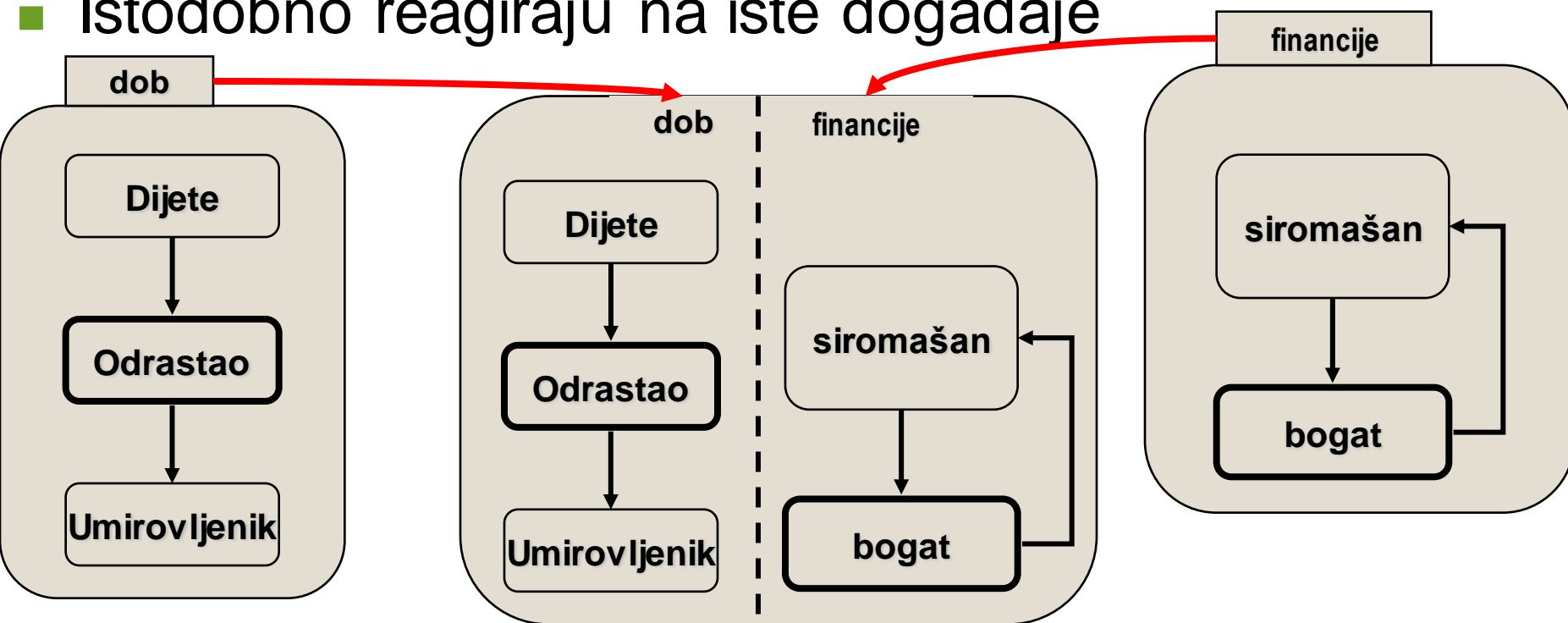




Paralelna stanja - Ortogonalnost



- Višestruka perspektiva istog objekta *engl. orthogonality*
 - paralelna stanja
- Istodobno reagiraju na iste događaje



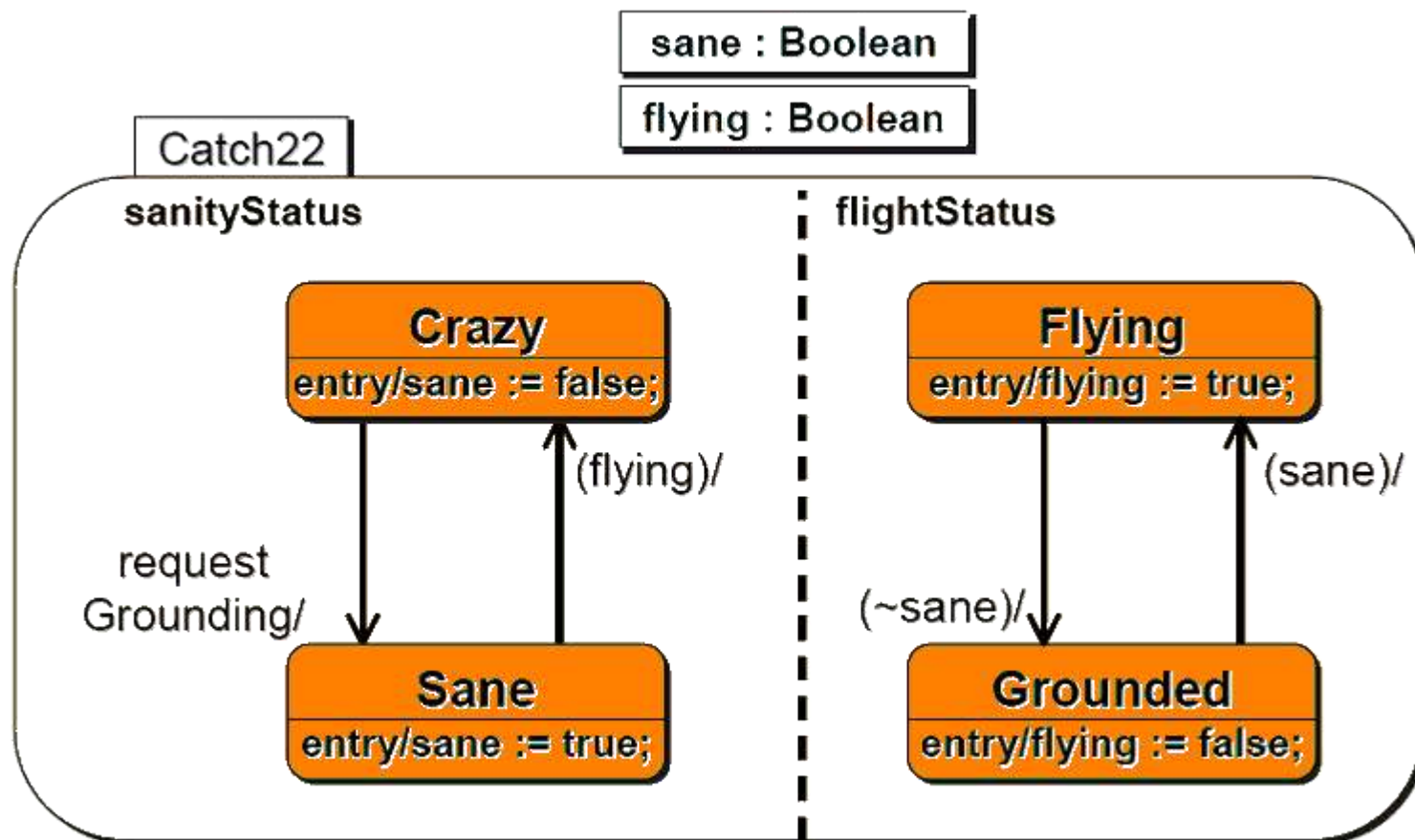
- Ne modelirati neovisne objekte primjenom ortogonalnih regija!!



Interakcija između područja



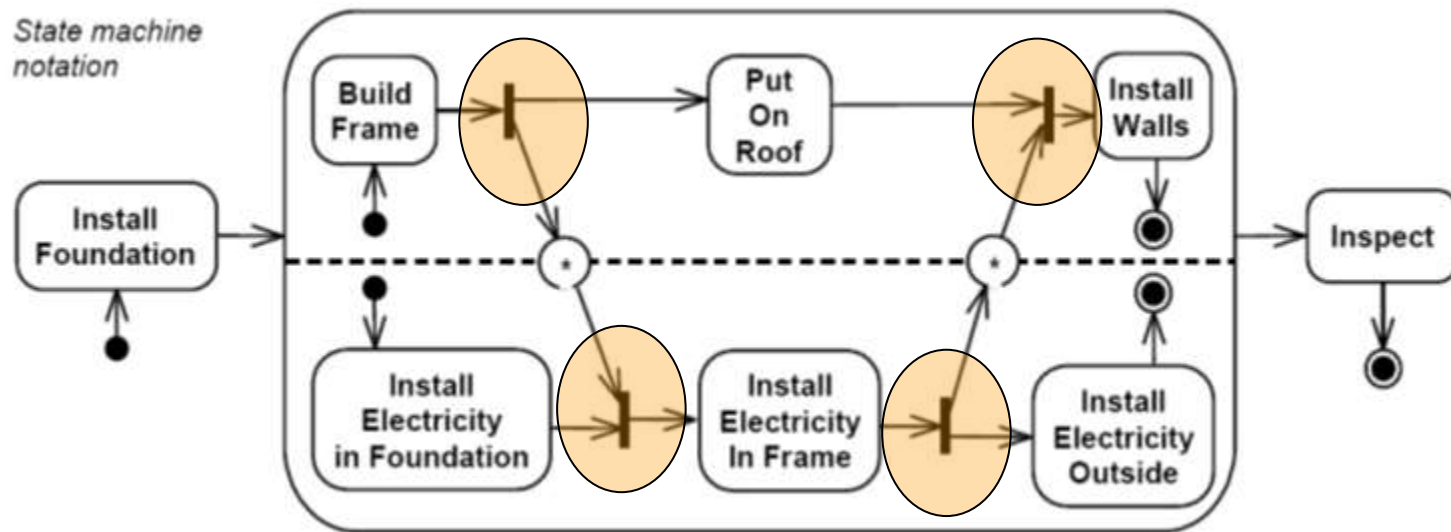
- Uporabom dijeljenih varijabli



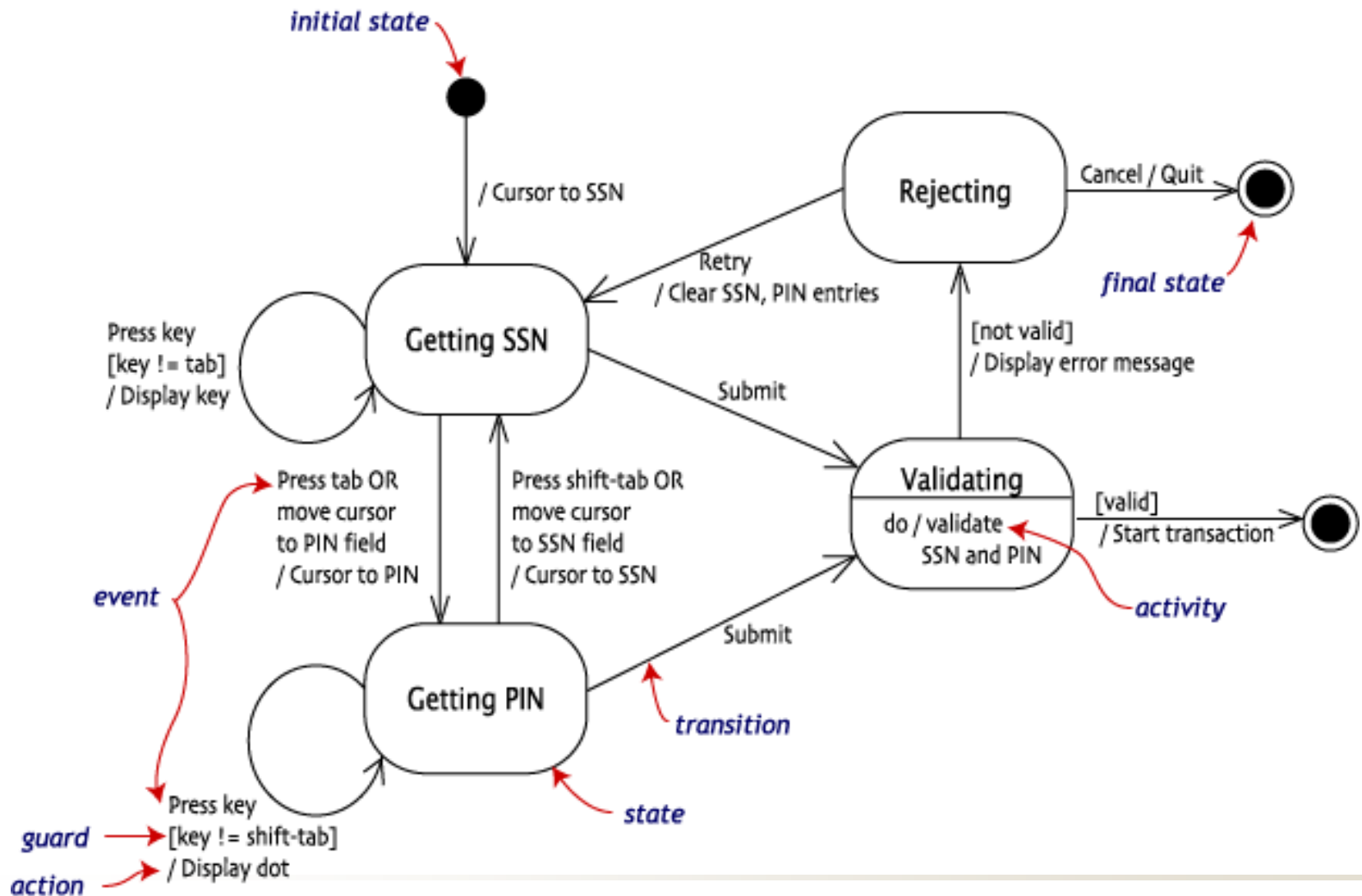


Račvanje i skupljanje

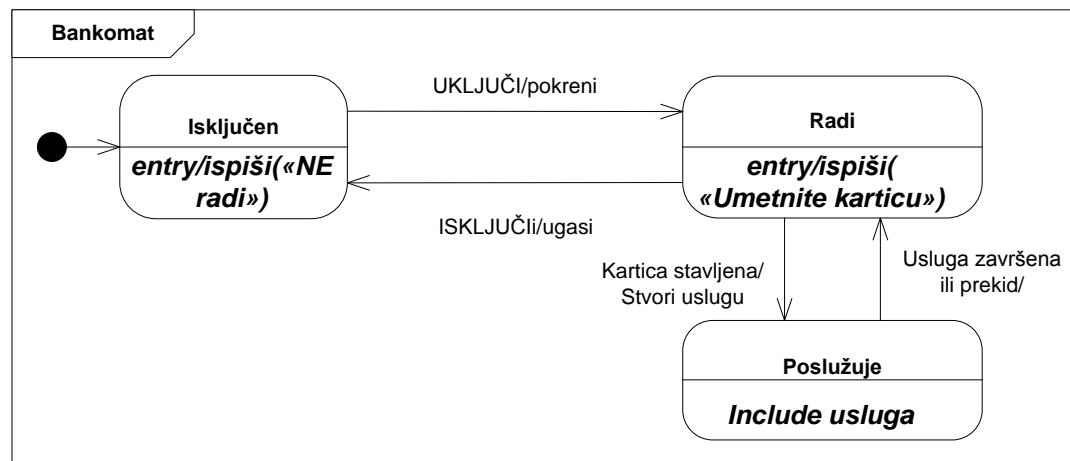
- Prikaz račvanja i skupljanja paralelnih prijelaza koristi se za opis prijelaza u i iz ortogonalnih paralelnih regija.



■ Prijava: unos OIB-a i PIN-a

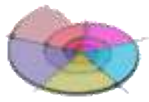


Primjer: Dijagram stanja bankomata





- Pogodni za modeliranje događajima poticanog ponašanja sustava (*engl. **event driven behavior***).
 - u objektno usmjerenoj arhitekturi - opis ponašanja aktivnih objekata.
 - Sustavi se prikazuju kao mreža međusobnog djelovanja (kolaboracije) strojeva stanja.
- U okviru UML jezika dijagrami stanja sadrže napredne značajke:
 - hijerarhijsko modeliranje stanja.
 - modeliranje paralelnih (ortogonalnih) stanja.
 - aktivnosti u stanjima.
 - akciju pri ulasku/izlasku iz stanja.
 - statičko i dinamičko uvjetno grananje.



UML dijagrami



- Obrasci uporabe
- Sekvencijski dijagram
- Dijagram komunikacije
- Dijagram stanja
- ***Dijagram aktivnosti***
- Dijagram komponenti
- Dijagram razmještaja
- Dijagram paketa
- Dijagram pregleda interakcije
- Vremenski dijagram
- Dijagram profila
- Dijagram razreda
- Dijagram objekata
- Dijagram složene strukture



Dijagrami aktivnosti



- *engl. Activity diagrams:*
- U prvoj inačici UML-a predstavljali su samo specifičan prikaz dijagrama stanja
- UML 2 definira novu semantiku zasnovanu na Petrijevim mrežama
 - povećana prilagodljivost modeliranju različitih tipova tijeka
 - jasno razdvajanje od dijagrama stanja
 - usmjereno na opis tijeka izvođenja i ponašanja sustava
- Namjena pobliže opisivanje obrazaca uporabe, dijagrama razreda, komponenti, sučelja i operacija
- Pogodni za:
 - prikaz proceduralnog tijeka procesa
 - modeliranje poslovnih zahtjeva (procesa)
 - prikaz paralelnosti



Primjena dijagrama aktivnosti



- Primjenjuju se za opis modela toka upravljanja (*engl. control flow*) ili toka podataka (objekata).
- Ne primjenjuju se za modeliranje događajima poticanog ponašanja.
- Tipična primjena: opis poslovnog modela u kojem želimo modelirati tijek zadataka i poslova.
- Temeljna razlika između dijagrama aktivnosti i komunikacijskih dijagrama te dijagrama stanja je način iniciranja pojedinog koraka a posebice kako koraci dobivaju ulazni signal ili podatke (u komunikacijskom dijagramu to su poruke, u dijagramu stanja to su događaji).
- U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog (neovisno da li su ulazi dostupni, ispravni ili potpuni u tom času).
 - tzv. “pull” način djelovanja (povlačenje).
- U modeliranju toka podataka (objekata) slijedeći korak se poduzima kada su svi ulazni podaci dostupni.
 - tzv. “push” način djelovanja (guranje).
- Dijagrami aktivnosti vrlo su slični dijagramima stanja, ali s drugačijom semantikom elemenata s kojima se modelira.



Primjena dijagrama aktivnosti



- Za opis upravljačkog i podatkovnog toka
- Analiza
 - oblikovanje tijeka obrazaca uporabe
 - oblikovanje tijeka između različitih obrazaca uporabe
- Oblikovanje:
 - detalja operacija
 - detalja algoritama
- Modeliranje poslovnih procesa



Elementi dijagrama aktivnosti



- Prilagođeni opisu tijeka upravljanja i podataka
- Čvorovi – *engl. activity nodes, nodes:*
 - čvor akcije – *engl. Action nodes*
 - nedjeljive aktivnosti
 - upravljački čvorovi – *engl. Control nodes*
 - upravljanje tijekom
 - objekti – *engl. Object nodes*
 - predstavljaju objekte u aktivnostima
- Veze – *engl. Activity edges*
 - upravljački tijek – *engl. Control flows*
 - predstavlja tijek upravljanja aktivnostima
 - tijek objekta – *engl. Object flows*
 - tijek objekta kroz aktivnosti



- Aktivnost – *engl. Activity*
 - obuhvaća više čvorova i veza koji predstavljaju odgovarajući slijed zadataka
- Akcija – *engl. Action*
 - kratkotrajno, neprekidivo ponašanje unutar čvora akcije
- Za sve aktivnosti i akcije mogu biti definirani odgovarajući uvjeti prije izvođenja (*engl. preconditions*) i nakon izvođenja (*engl. post conditions*)
- Značke – *engl. Tokens*
 - dio semantike bez grafičkog prikaza
 - značka može predstavljati:
 - upravljački tijek
 - objekt
 - podatak
 - kreću se od izvorišta prema odredištu vezama ovisno o:
 - ispunjenim uvjetima izvornog čvora
 - postavljenim uvjetima veza (*engl. Edge guard conditions*)
 - preduvjetima ciljnog čvora



- Modelira ponašanje kao niz akcija
- Izvođenje se modelira uporabom znački
 - proizvode ih čvorovi, kreću se po vezama

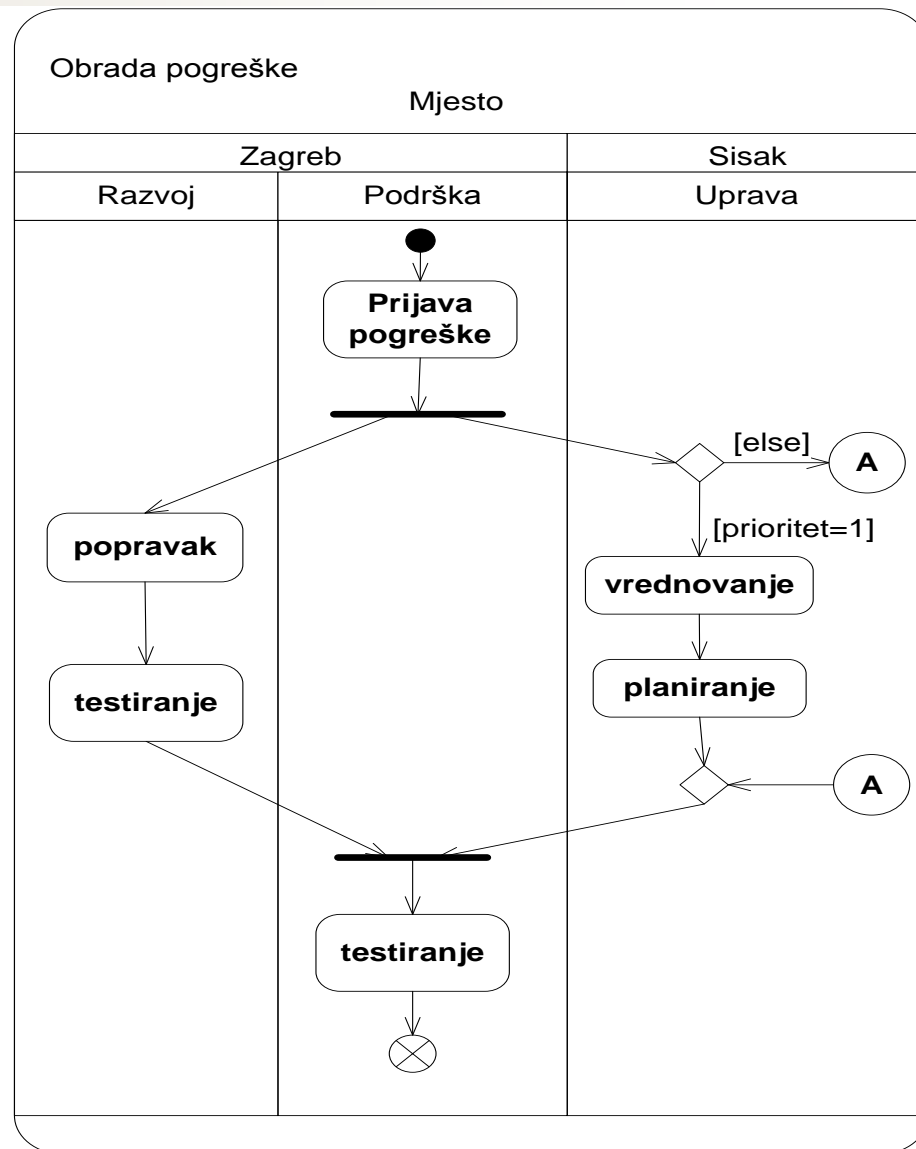


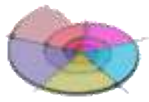


Podjela aktivnosti - particije



- Dijagram aktivnosti može biti podijeljen u particije (*engl. swim lanes*)
 - horizontalno, vertikalno, proizvoljno
 - hijerarhija
- Ne utječu tijek odvijanja aktivnosti
 - način grupiranja

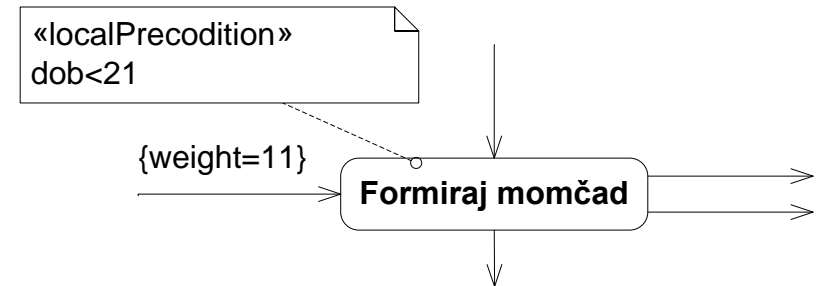
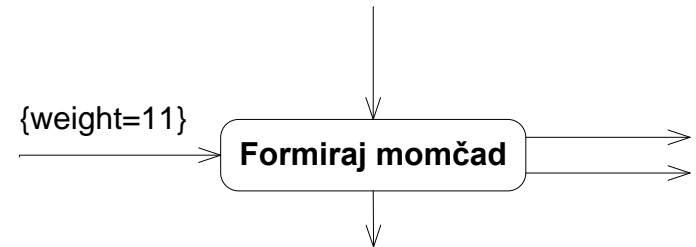




Čvor akcije



- Započinje izvođenje kada:
 - postoji odgovarajući broj znački na svim ulazima
 - zadovoljeni su svi lokalni preduvjeti akcije
- Nakon izvođenja provjerava se zadovoljavanje uvjeta te proslijeđuju značke na sve izlaze
- Tipovi:
 - obrade osnovnih operacija, pozivanje složenih aktivnosti ili ponašanja - *engl. call action*
 - komunikacijske akcije
 - slanje signala
 - manipuliranje objektima

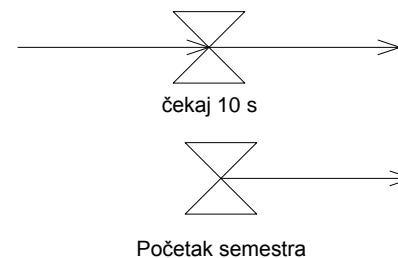
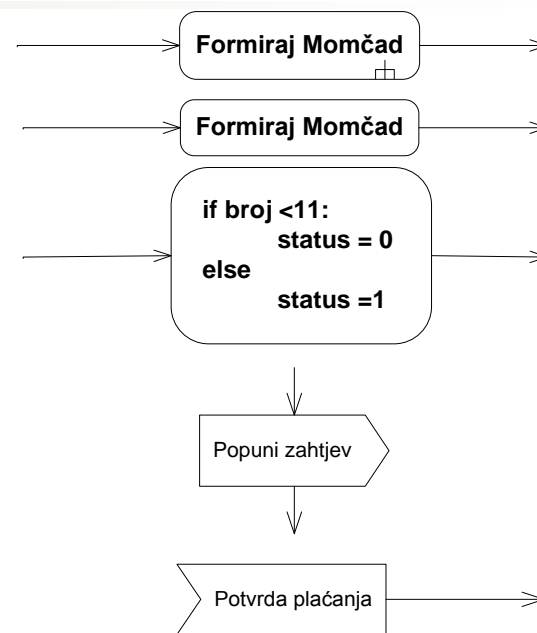




Tipovi čvorova akcije



- Obrada osnovnih operacija, pozivanje složenih aktivnosti ili ponašanja - *engl. call action*
- Slanje signala – *engl. send signal*
 - asinkroni signal
- Prihvatanje događaja – *engl. accept event*
 - čekanje na neki događaj
- Vremenski događaja – *engl. time event*
 - definirana vremenskim izrazom





Upravljački čvorovi



- Rukuju upravljanjem aktivnostima
 - početak i završetak
 - odluke
 - paralelnost

	Početni čvor - Initial node
	Završni čvor aktivnosti - Activity final node
	Završetak toka - Final flow node
	Čvor odluke - Decision node
	Spajanje čvorova - Merge node
	Grananje toka- Fork node
	Spajanje - Join node Sinkronizira više paralelnih tijekova

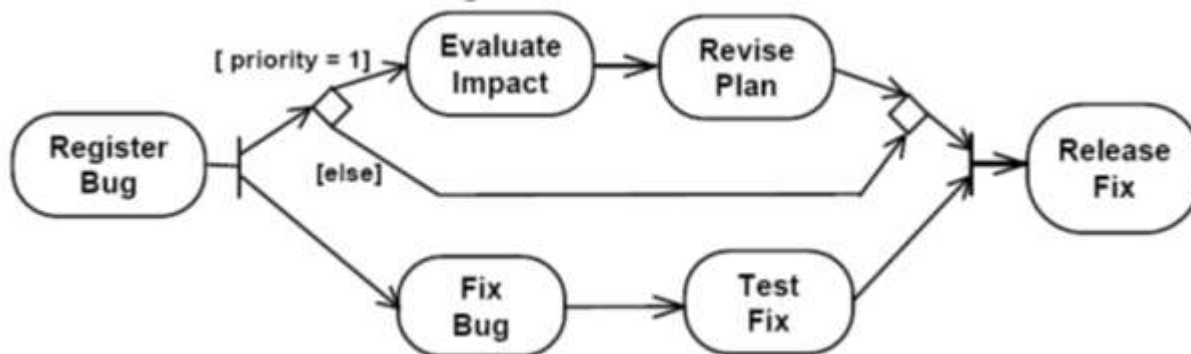


Koordiniranje koraka



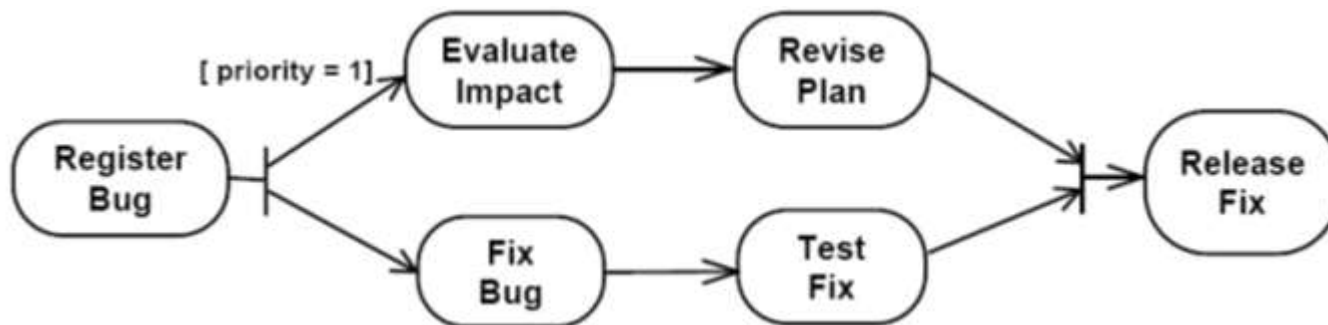
- Uvjetovanje toka aktivnosti može se izvesti na dva načina:

- u točkama odlučivanja



- na račvanju

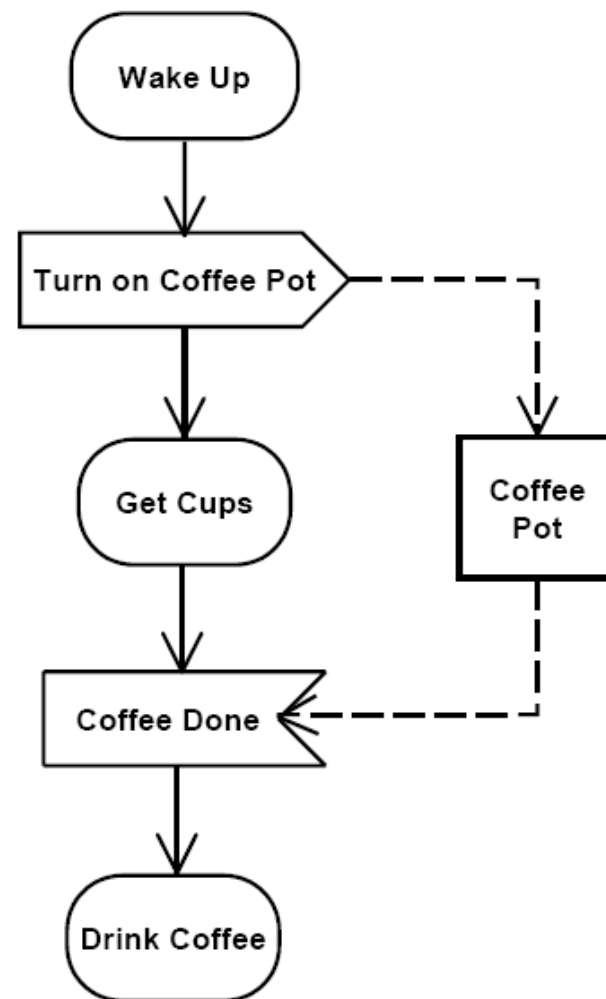
- sažetije





Primjer uporabe signala

- U dijagramu aktivnosti katkada je potrebno poslati ili primiti signal.
- Pošalji (*engl. **send***) signal preslikava se u prijelaz i akciju slanja signala.
- Primi (*engl. **receive***) signal preslikava se u stanje čekanja na signal bez akcije.

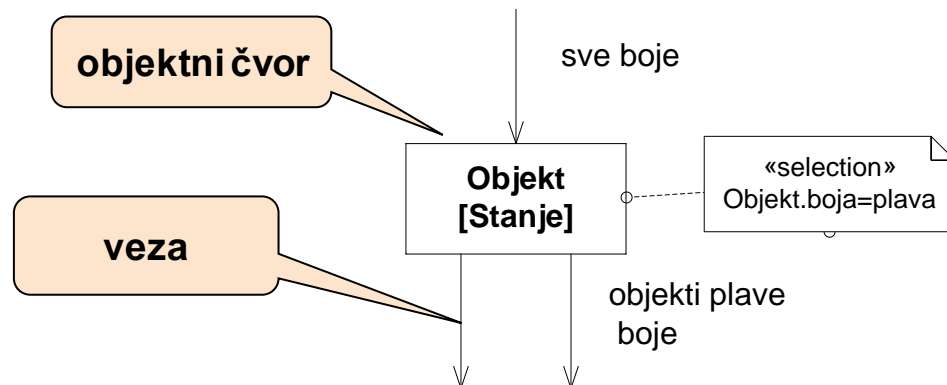




Objektni čvorovi

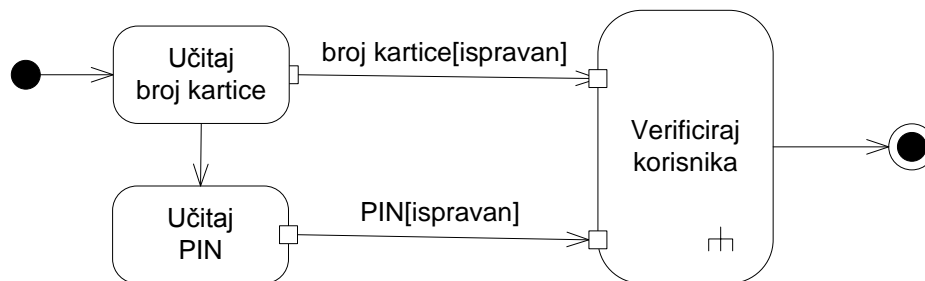


- Prikazuju podatke i objekte
 - objektni čvor ukazuje na raspoloživost u danoj točki aktivnosti
 - uobičajeno označeni imenom razreda i predstavljaju instancu
- Ulazne i izlazne veze predstavljaju tok objekta
 - opisuju kretanje objekta unutar aktivnosti
- Objekte stvaraju i upotrebljavaju akcijski čvorovi
- Objektne značke
 - kada objektni čvor primi značku, nudi ju na svim izlaznim vezama koje se natječu za značku. Značku dobiva prva veza koja ju je spremna prihvatiti
- Objektni čvorovi se ponašaju kao međuspremnici (*engl. buffer*)
 - pohranjuju objektne značke



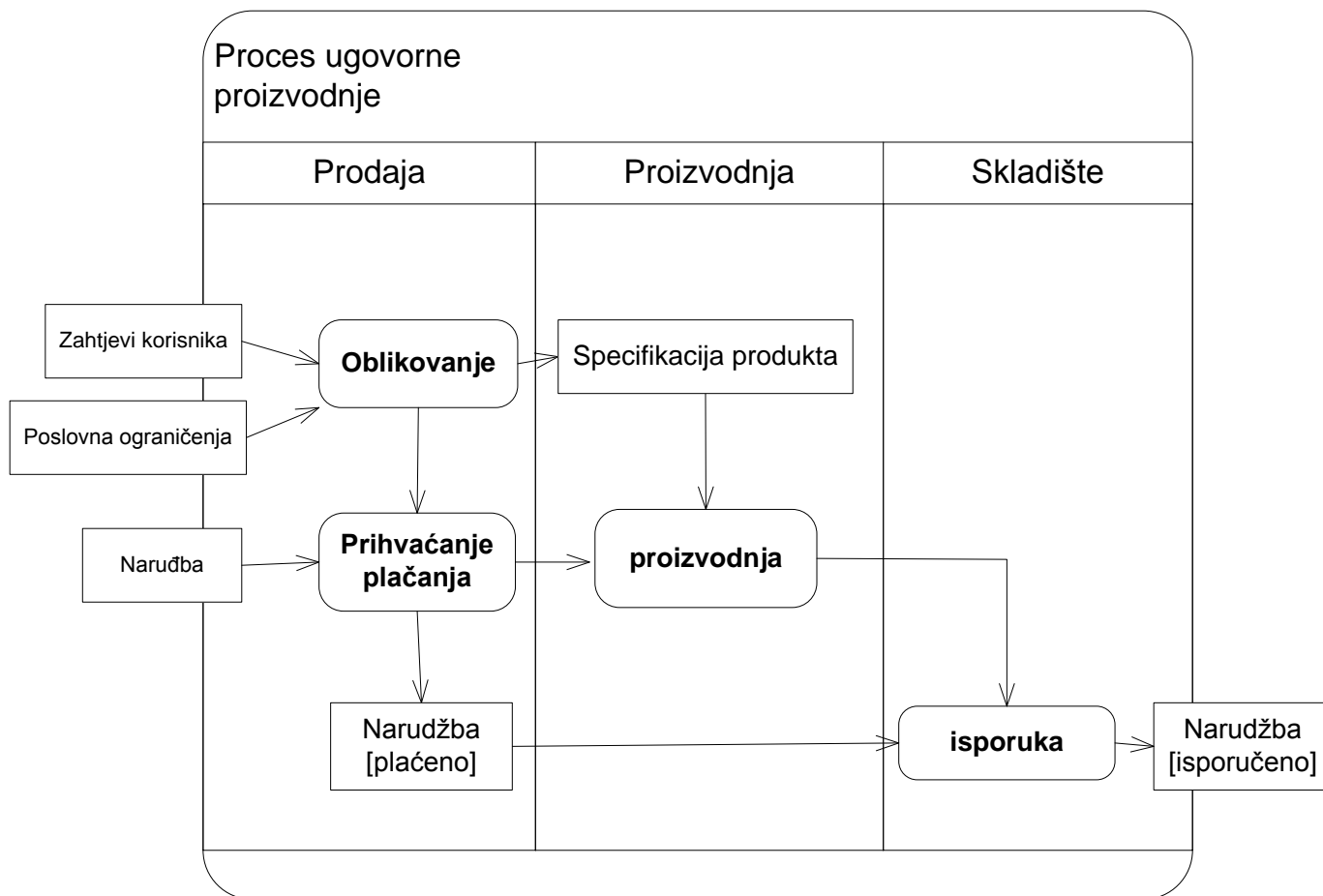


- *engl. Pins*
- Objektni čvor s jednim ulazom ili izlazom prema čvoru akcije
- Pojednostavljuju grafički prikaz dijagrama aktivnosti s većim brojem objektnih čvorova



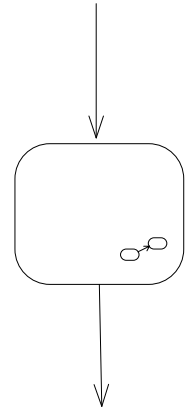


Primjer:





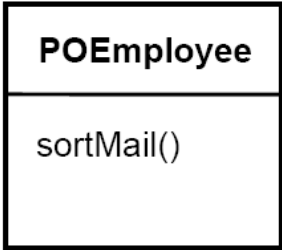
- *engl. Subactivity*
- Namjena funkcionalne dekompozicije
- Ugnježdjenje drugog dijagrama aktivnosti
- Ulaskom u podaktivnost pokreće se njegova početna akcija
- Po završetku podaktivnosti, nastavlja se izvođenje prethodne aktivnosti





Primjer:

- Primjer: Objekt iz razreda *POEmployee* izvodi operacije *sortMail()* i *deliverMail()*.

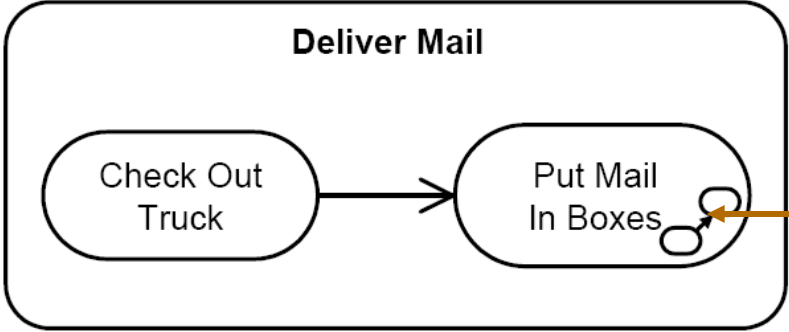


- Operacije opisujemo slijedom dviju aktivnosti:



ima podaktivnosti

- Aktivnost *Deliver Mail* ima podaktivnosti koje se razjašnjavaju novim dijagramom:

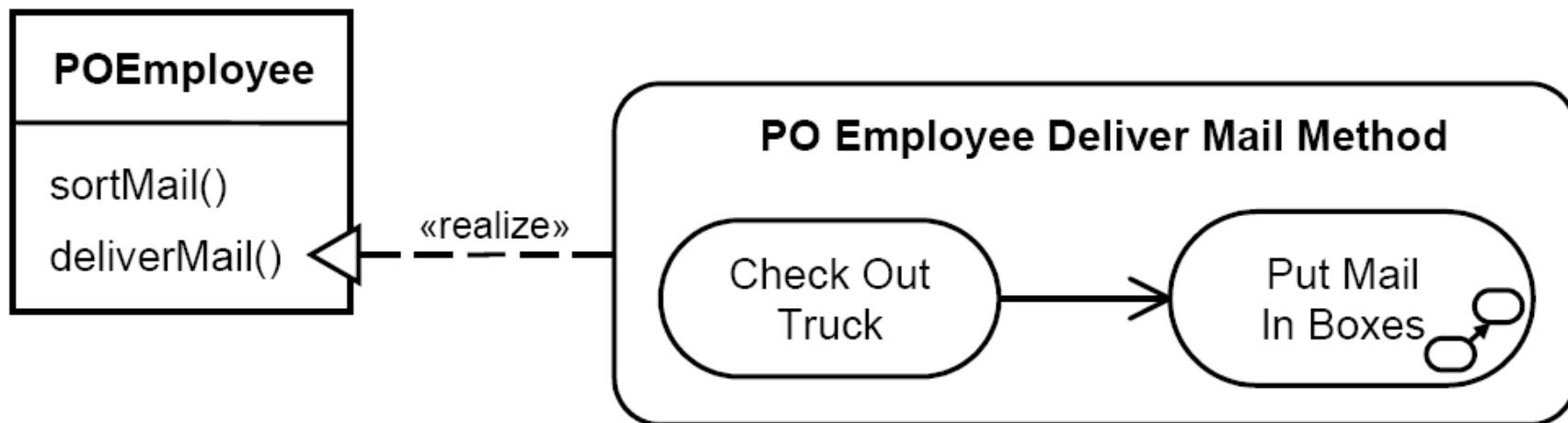


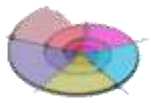
ima daljnje podaktivnosti



Dijagram aktivnosti kao “metoda”

- Programska potpora opisana dijagramima aktivnosti je potpuno objektno usmjerena ako svi dijagrami aktivnosti realiziraju metode (procedure) kao implementaciju pojedinih operacija.
- Na slici dijagram aktivnosti realizira (oznaka «*realize*»)
- Pokazuje se povezivanje dijagrama razreda i dijagrama aktivnosti.



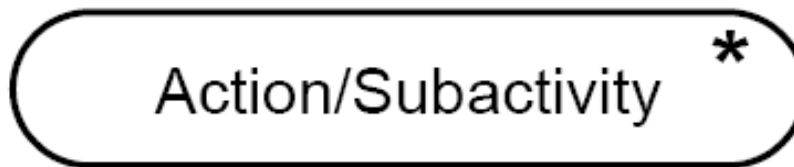


Dinamički paralelizam



- Neka aktivnost ili podaktivnost može se izvoditi više puta (paralelno).
- Broj izvođenja ovisi o rezultatu evaluacije nekog izraza tijekom izvođenja.
- Unutar simbola aktivnosti/podaktivnosti može se umetnuti UML oznaka brojnost koja određuje dopustiv maksimalan broj paralelnog izvođenja (npr. osiguranje od beskonačne petlje).
- Nakon završetka svih paralelnih aktivnosti inicira se prijelaz na novi korak.

ovdje 0..*, odnosno *

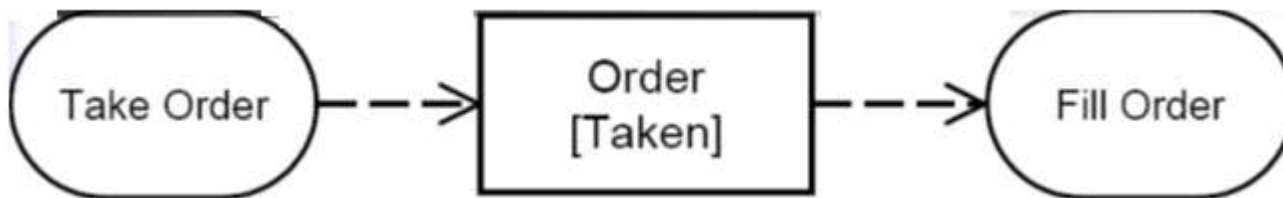
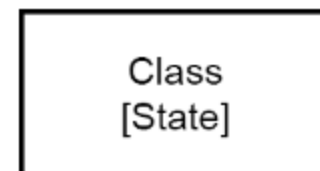


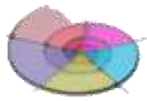


Objekt kao uvjet prijelaza



- Katkad je potrebno prijelaz s aktivnosti na drugu aktivnost uvjetovati postojanjem određene vrste objekta (npr. ulazno/izlaznih parametara).
 - u prijelaz se umeće simbol:
 - u tom stanju nema nikakve aktivnosti.
 - tok se nastavlja do slijedećeg koraka nakon zadovoljenja uvjeta.
- Npr. aktivnost *Take Order* mora generirati izlazni parametar (objekt određenog tipa), a aktivnost *Fill Order* mora imati ulazni parametar (objekt određenog tipa).
- U modeliranju uvjetnog prijelaza koriste se crtkane oznake.





- Pogodno za slučajeve:
 - kada ponašanje ne zavisi o velikom broju vanjskih događaja
 - postoje definirani koraci bez prekida
 - kada imamo tijek objekata između koraka
 - kada želimo pojasniti aktivnosti

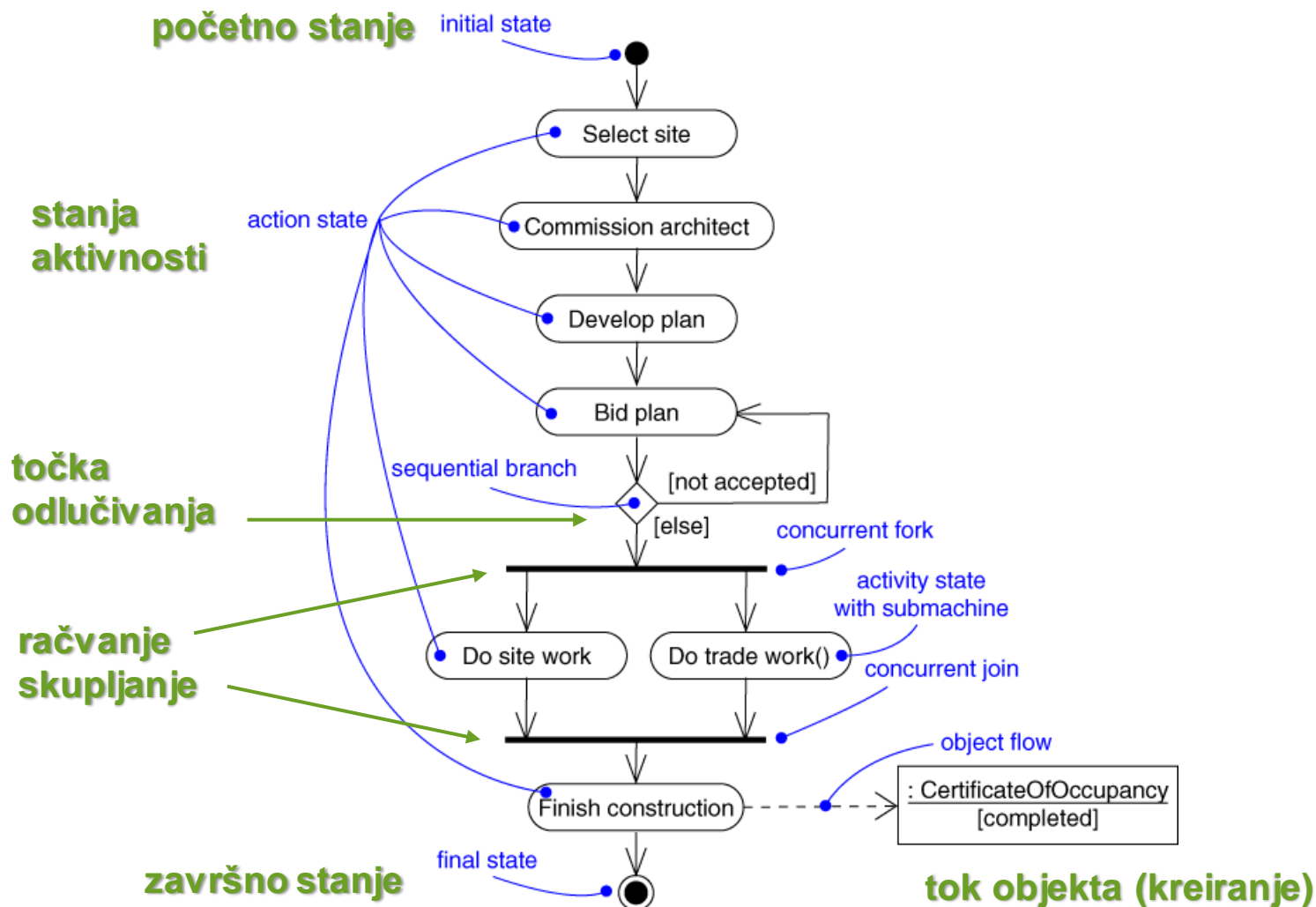


Primjer: bankomat





Primjer: gradnja kuće



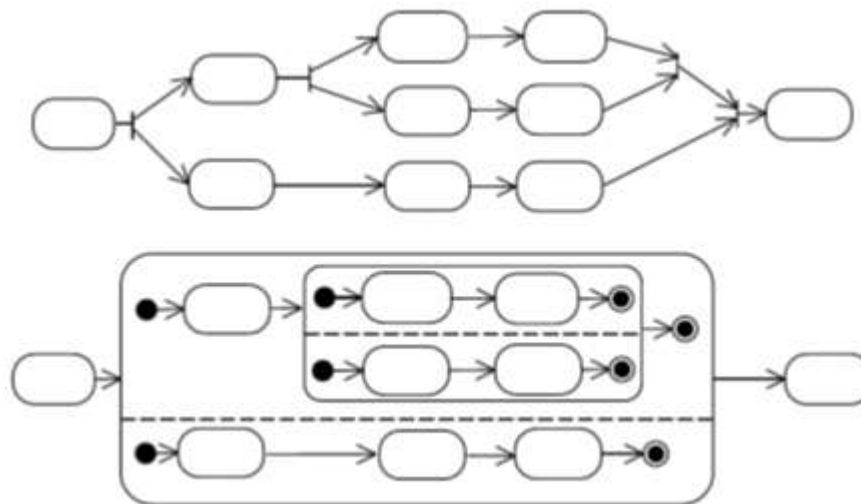


Dijagrami aktivnosti

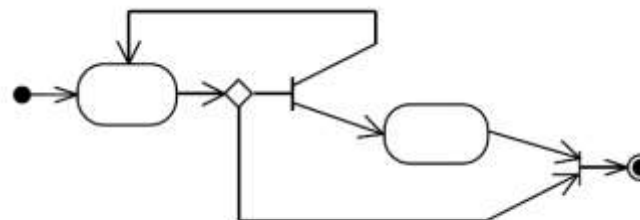


- Dijagrami aktivnosti naslijedili su iz dijagrama stanja potrebu za dobrom strukturom i ugnježđivanjem složenih stanja.

- Dobro ugnježđivanje:



- Loše ugnježđivanje:

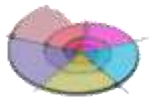




Dijagrami aktivnosti – sažetak



- Dijagrami aktivnosti koriste se za modeliranje ponašanja koje nije jako ovisno o vanjskim događajima. Primarno na modeliranje poslovnog procesa (a ne na npr. oblikovanje ugrađenih sustava zasnovanih na računalima).
- Dijagrami aktivnosti slični su dijagramima stanja (do UML v1.3 nisu bili posebno deklarirani, u UML v2.0 potpuno su odvojeni).
- Dijagrami aktivnosti posjeduju korake koji se izvode do završetka (nisu prekidani događajima).
- Dijagrami aktivnosti izrađuju se u slučaju kad je usredotočenost ponašanja na pojedinim aktivnostima i njihovom slijedu a ne na to koji objekti su odgovorni za te aktivnosti.
- Između koraka postoji tok objekata.
- Upravljački tok i podatkovni/objektni tok nemaju različitu već jedinstvenu semantiku.



- Obrasci uporabe
- Sekvencijski dijagram
- Dijagram komunikacije
- Dijagram stanja
- Dijagram aktivnosti
- ***Dijagram komponenti***
- Dijagram razmještaja
- Dijagram paketa
- Dijagram pregleda interakcije
- Vremenski dijagram
- Dijagram profila
- Dijagram razreda
- Dijagram objekata
- Dijagram složene strukture



Programske komponente



- *engl. software component*
 - “are binary units of independent production, acquisition, and deployment that interact to form a functioning system”, C. Szyperski
 - zamjenjivi, ponovo iskoristivi dijelovi koda.
- U programskom inženjerstvu zasnovanom na komponentama sustav se integrira
 - višestrukom uporabom postojećih komponentata,
 - uporabom komercijalnih, gotovih komponentata (*engl. commercial-of-the-shelf COTS*) ili
 - modificiranim komercijalnim komponentama (*engl. modified-of-the-shelf MOTS*).



Dijagrami komponenata



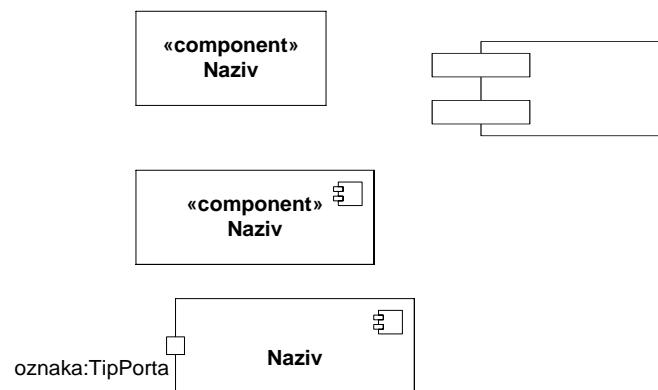
- Dijagrami komponenata predstavljaju statički pogled na sustav.
- Opisuju organizaciju i međuovisnost (fizičku strukturu) između implementacijskih komponenata programske potpore.
- Dijagrami komponenata dio su specifikacije arhitekture programske potpore.
- Dijagrame komponenata oblikuju arhitekti programske potpore i programeri.
- Vrste komponenata:
 - izvorni kod
 - binarni (objektni) kod
 - statičke ili dinamičke knjižnice programskih komponenata
 - izvršne (aka “also known as” exe) komponente programske potpore.
 - tablice
 - druge datoteke
 - baze podataka
 - ...



Dijagram komponenti



- *engl. Component Diagram*
- UML komponente predstavljaju **fizičke** modularne, zamjenjive jedinice s dobro definiranim sučeljem
 - obuhvaća neki sadržaj kojem se pristupa putem sučelja
 - definira ponašanje kroz ponuđena i zahtijevana sučelja
 - **razred** - logička apstrakciju
 - attribute i operacije
 - može im se pristupati izravno
 - **komponenta** - fizička stvar, fizičko obuhvaćanje logičkih apstrakcija
 - operacije
 - može im se pristupati samo kroz sučelja
- Namjena komponentnog dijagrama je prikaz komponenata, interne strukture i odnosa prema okolini
- Komponente mogu sadržavati druge komponente te na taj način prikazivati internu strukturu.
- Mogu imati definirane portove (*engl. port*)
 - točka interakcije komponente s okolinom
- Instanca komponente
 - ime_komponente: Tip_komponente

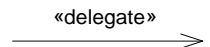


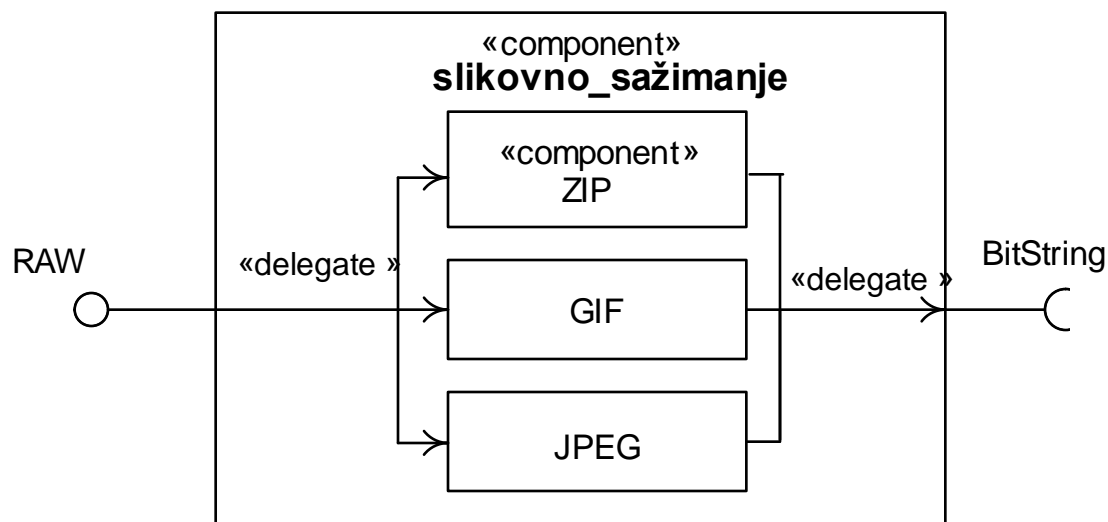


Sučelja/Konektori



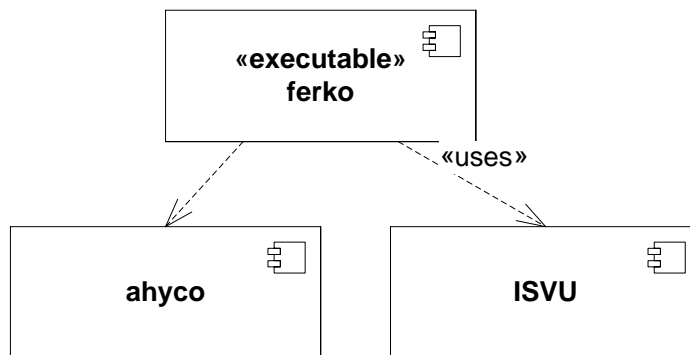
- *engl. Interface*
- Imenovan skup javno vidljivih atributa i apstraktnih operacija
 - implementaciju osigurava komponenta/razreda
- Komponente i klase ostvaruju sučelja
 - uključivanjem atributa i implementacijom operacija
- Tipovi sučelja:
- Ponuđeno/implementirano sučelje – *engl. provided interface*
 - ostvaruje ga razred ili komponenta
 - usluga koja se nudi
- Zahtijevano sučelje – *engl. required interface*
 - potrebno klasi ili komponenti
 - ono što je potrebno za njezin rad
- Tipovi konektora
- Spojnica - *engl. assembly connector*
 - povezuje dvije komponente
 - “ball-and-socket”, “lollipop”
- Delegacija - *engl. delegation connector*
 - povezuje sučelje komponente s internom strukturom







- *engl. dependency*
- Ova relacija između komponenti upotrebljava se kada jedna komponenta zahtijeva uporabu druge radi potpunog ostvarenja implementacije
- Predstavlja se crtkanom strelicom od ovisne komponente





■ Artefakti

- iz razvojnog procesa: izvorni kod, podaci ...
 - *engl. Source Component*
- binarne komponente za isporuku
 - *engl. Binary Component (object code file, static or dynamic libraries)*
- izvršne komponente
 - *engl. Executable component*

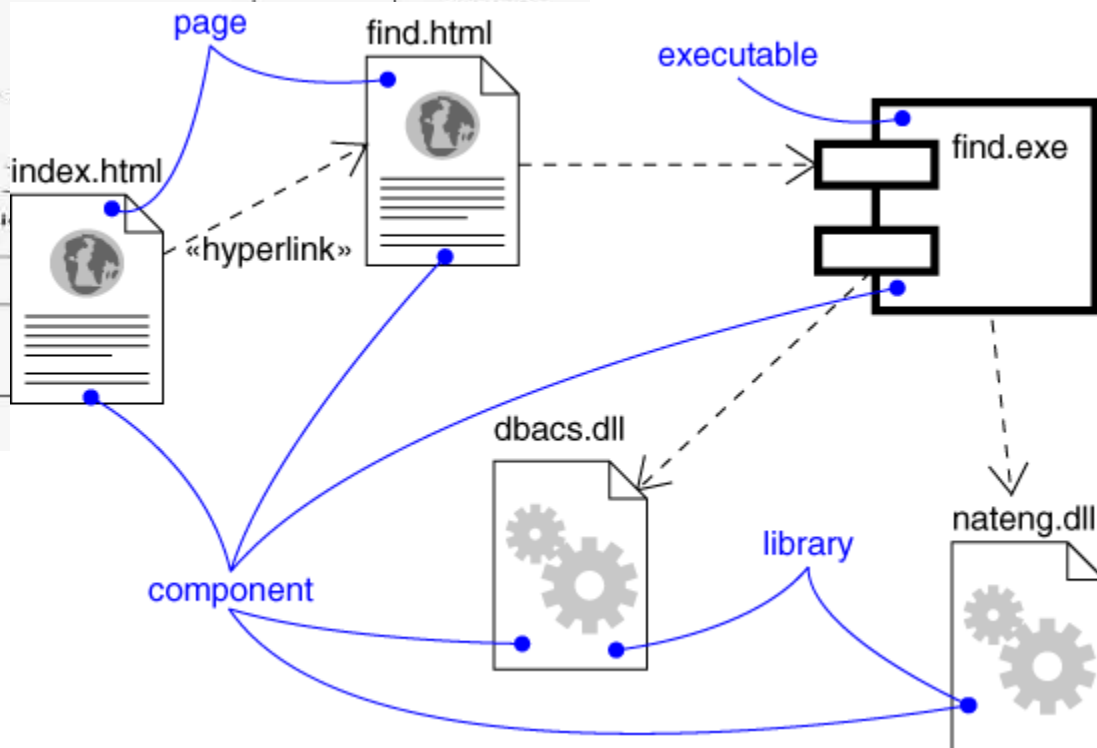
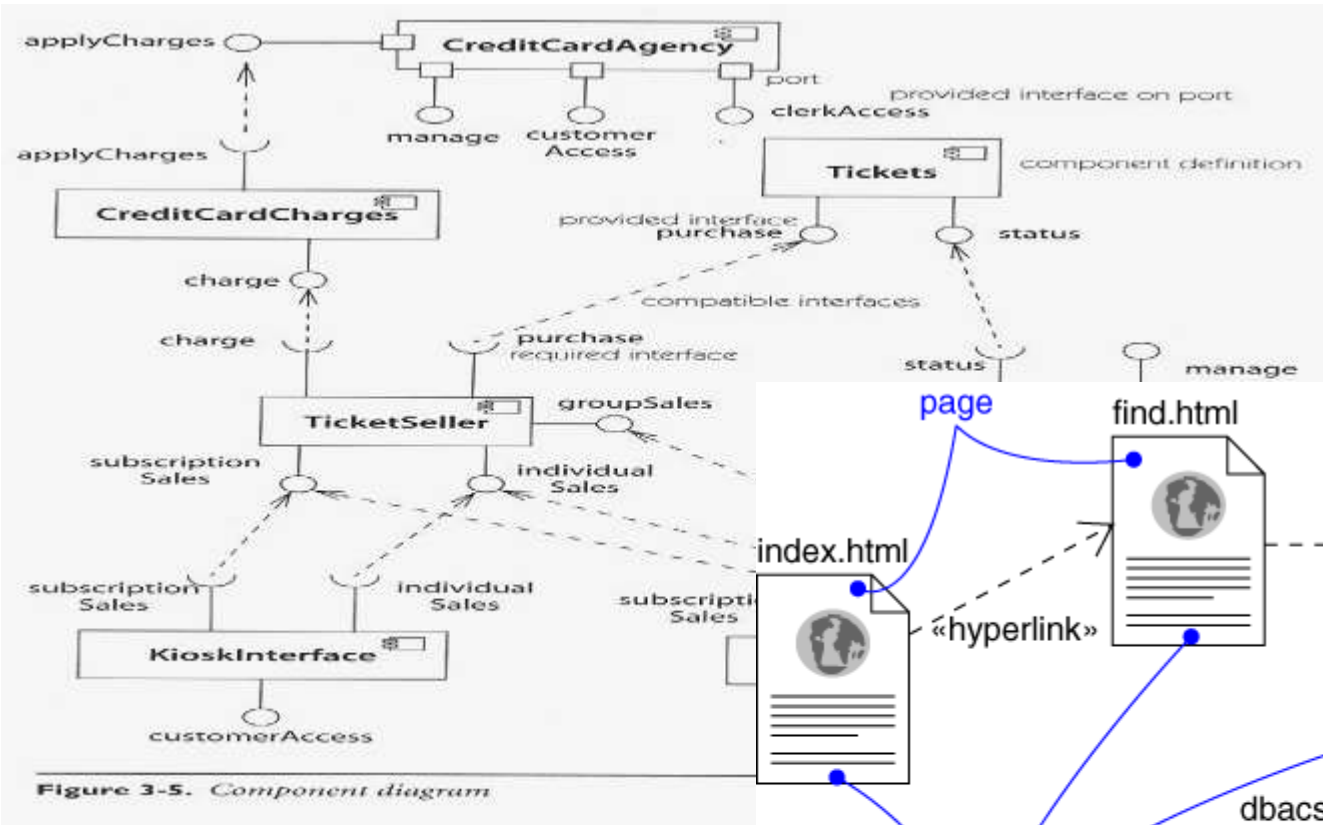
■ Stereotipi

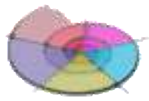
- *executable* - komponenta koja se može izvršavati
- *library* - statička ili dinamička biblioteka
- *file* - datoteka s proizvoljnim sadržajem
- *document* - dokument
- *script* - skript
- *source* - datoteka sa izvornim kodom

- Statički model programskih komponenti
 - ponovna uporaba i zamjena komponenti dijelova
- Oblikovanje programskih komponenti
 - arhitekturni model
 - detalji oblikovanja
 - odnos s okolinom
- Oblikovanje interne strukture komponenti



Primjer





UML dijagrami



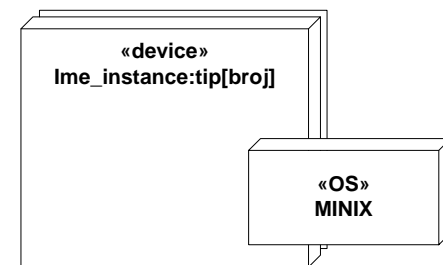
- Obrasci uporabe
- Sekvencijski dijagram
- Dijagram komunikacije
- Dijagram stanja
- Dijagram aktivnosti
- Dijagram komponenti
- ***Dijagram razmještaja***
- Dijagram paketa
- Dijagram pregleda interakcije
- Vremenski dijagram
- Dijagram profila
- Dijagram razreda
- Dijagram objekata
- Dijagram složene strukture



Dijagram razmještaja



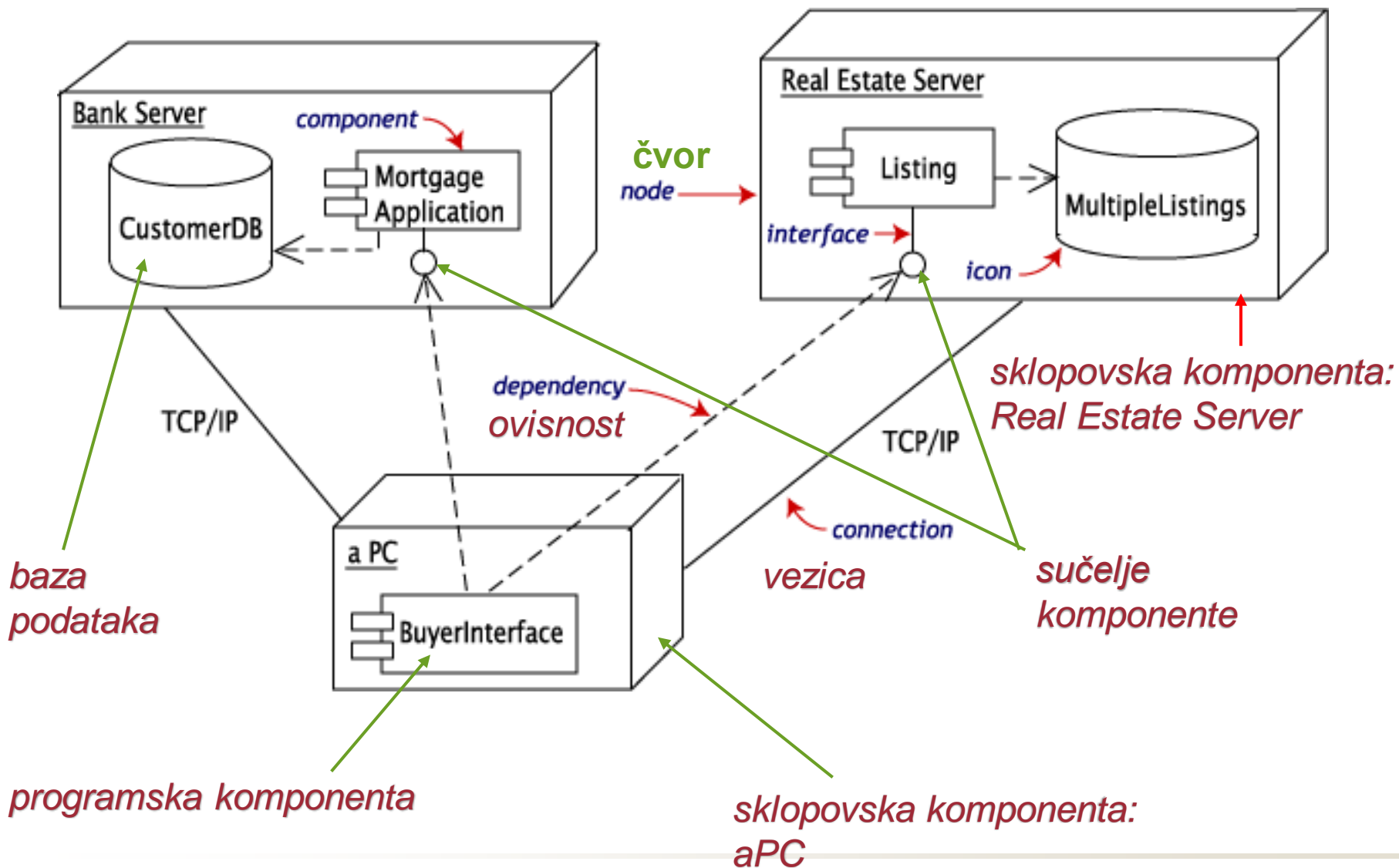
- *engl. Deployment diagram*
- opisuju topologiju sustava i usredotočeni su na odnos sklopovskih i programskih dijelova
 - ostvarenje u obliku koda i podataka koje se nalazi i izvršava na računalnim resursima
 - prikaz sklopovskih komponenti, komunikacijskih putova te smještaja i izvođenja programskih artifakta
 - naznaka gdje i kako implementirati komponente
- Sklopovske komponente:
- čvorovi – *engl. nodes*
 - uređaj – *engl. device*
 - stvarni i virtualni uređaji
 - procesna jedinka npr. računalo
 - oznaka «device»
 - okolina izvođenja – *engl. execution environment*
 - programski sustav npr. virtualni stroj, OS, interpreter
 - oznaka «execution environment»
- Spojevi – *engl. connections*
 - komunikacijski putovi
- Programske komponente (artifakti)
 - Komponente – implementirani moduli i podaci
 - Ovisnosti – *engl. Dependencies*
 - prikazuju odnos između komponenti



Vezice - punom crtom
Ovisnost - crtkanom strelicom

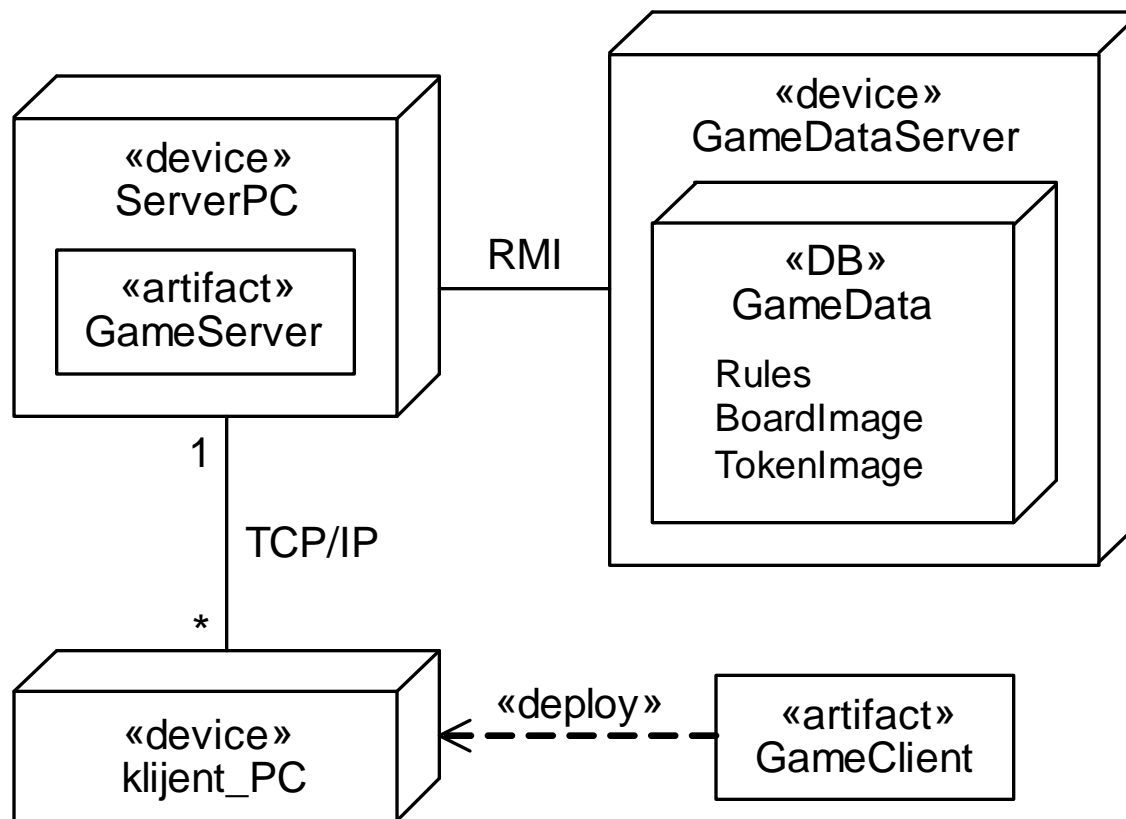


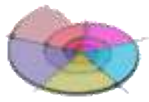
Primjer: kupnja nekretnine





Primjer:





UML dijagrami



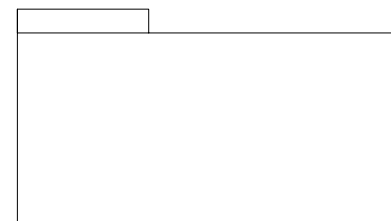
- Obrasci uporabe
- Sekvencijski dijagram
- Dijagram komunikacije
- Dijagram stanja
- Dijagram aktivnosti
- Dijagram komponenti
- Dijagram razmještaja
- ***Dijagram paketa***
- ***Dijagram pregleda interakcije***
- ***Vremenski dijagram***
- ***Dijagram profila***
- *Dijagram razreda*
- *Dijagram objekata*
- *Dijagram složene strukture*



Dijagram paketa

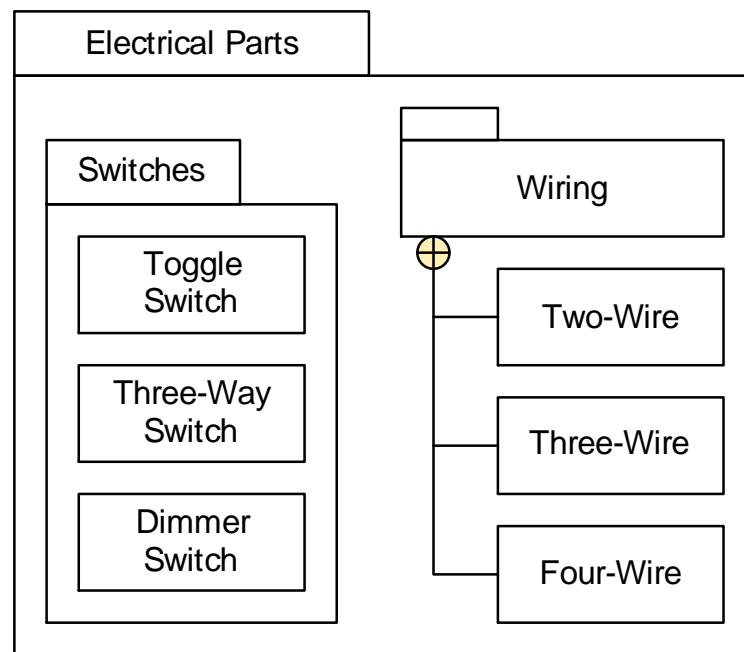
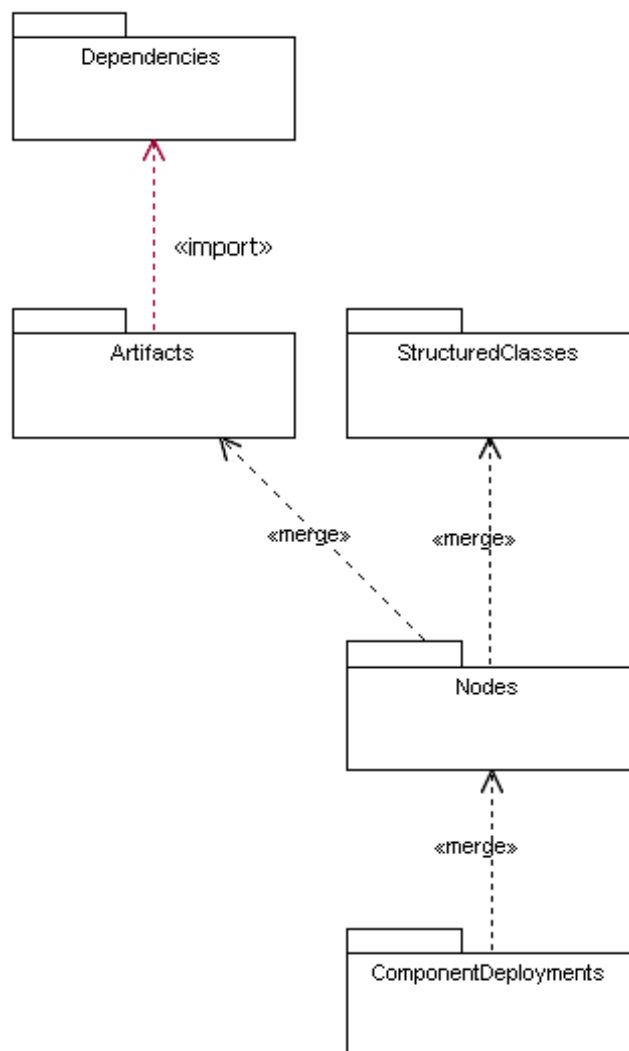


- *engl. Package Diagram*
- Paket je mehanizam organiziranja elemenata u grupe
 - nema utjecaj na izvođenje
- Dobro definiran paket povezuje semantički bliske elemente koji imaju tendenciju zajedničkih promjena
- U jednostavnim aplikacijama nema potrebe uvođenja paketa
- Dijagram paketa prikazuje dekompoziciju modela u organizirane grupe i njihove međuodnose
 - hijerarhija
- Vidljivost
- Moguća su tri nivoa vidljivosti paketa:
 - javni element (*engl. public*)
 - sadržaj vidljiv svim paketima koji koriste paket (+)
 - zaštićeni (*engl. protected*)
 - mogu ga vidjeti samo djeca tog paketa (#)
 - privatni (*engl. private*)
 - nije vidljiv izvan paketa (-)



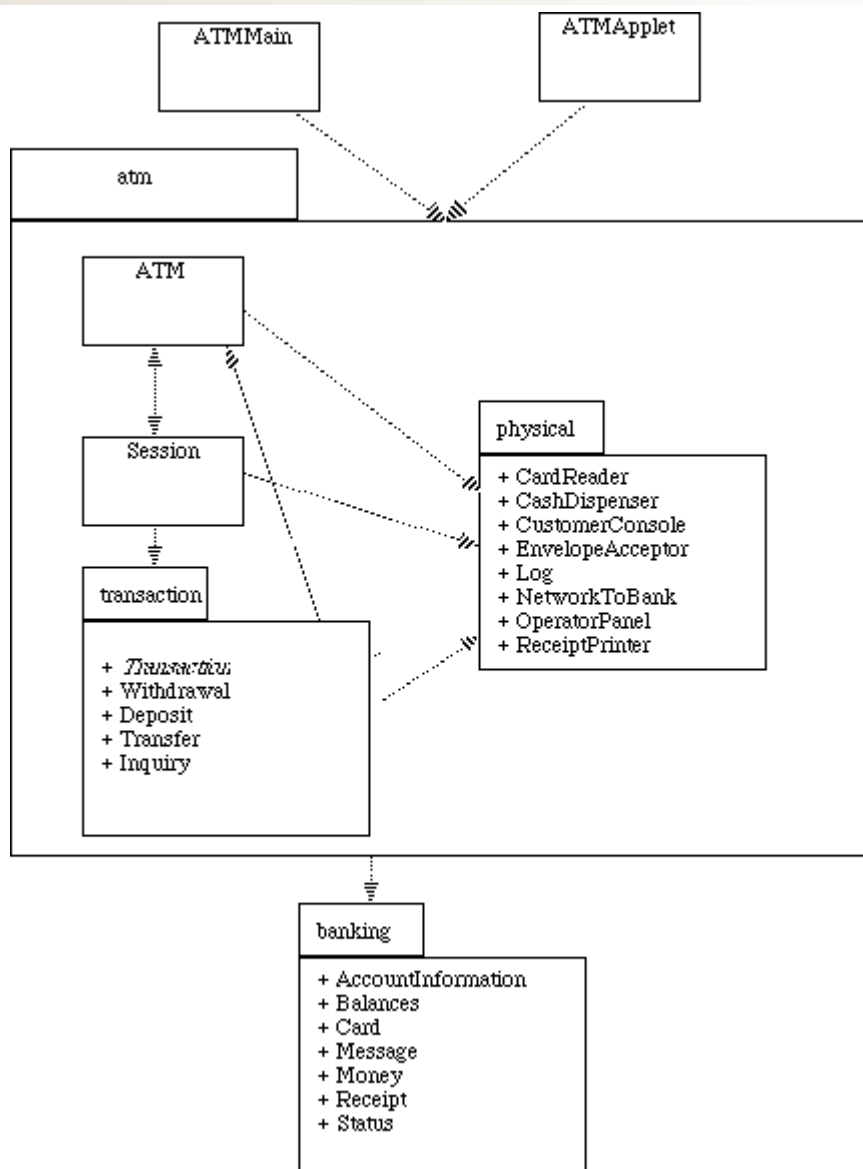


Primjer: Dijagrami paketa





Primjer: bankomat

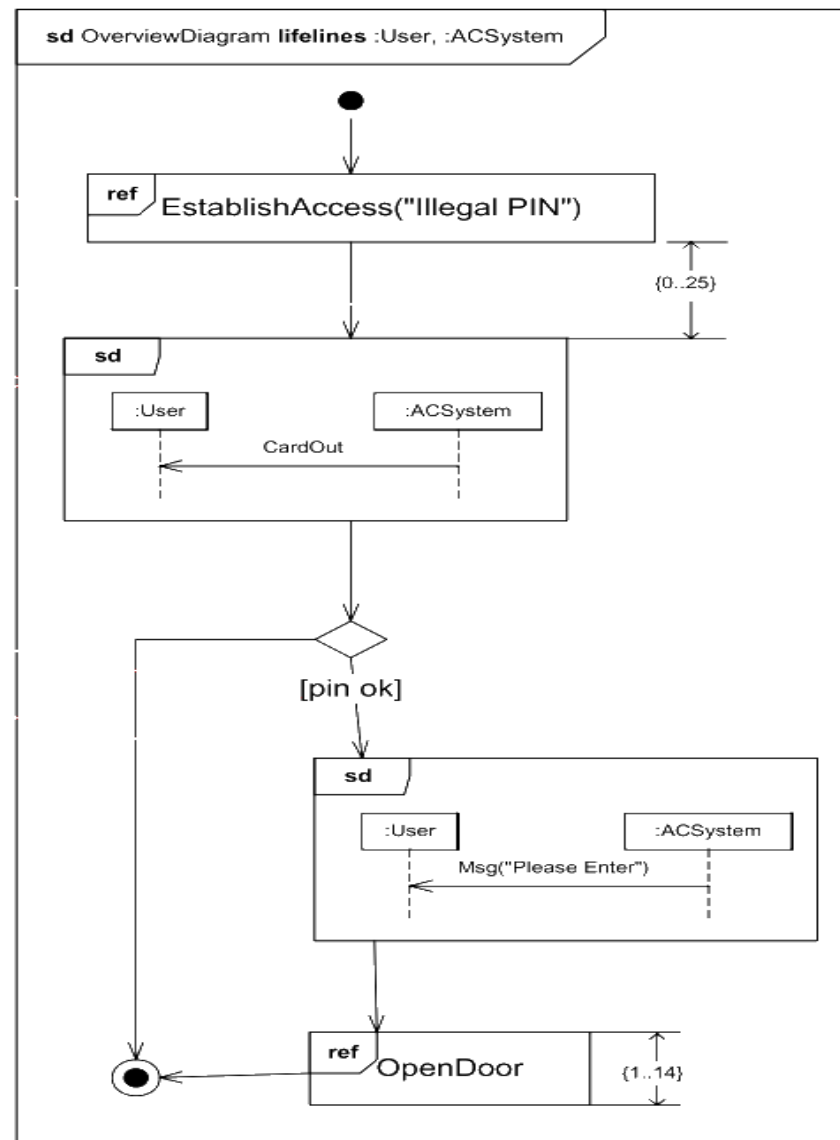




Dijagram pregleda interakcije



- *engl. Interaction Overview Diagram*
- Kombinacija dijagrama aktivnosti i sekvencijskog dijagrama
 - dijagram aktivnosti s dijelovima sekvencijskih dijagrama i kontrolom tijeka
 - notacija odluka i grananja iz dijagrama aktivnosti

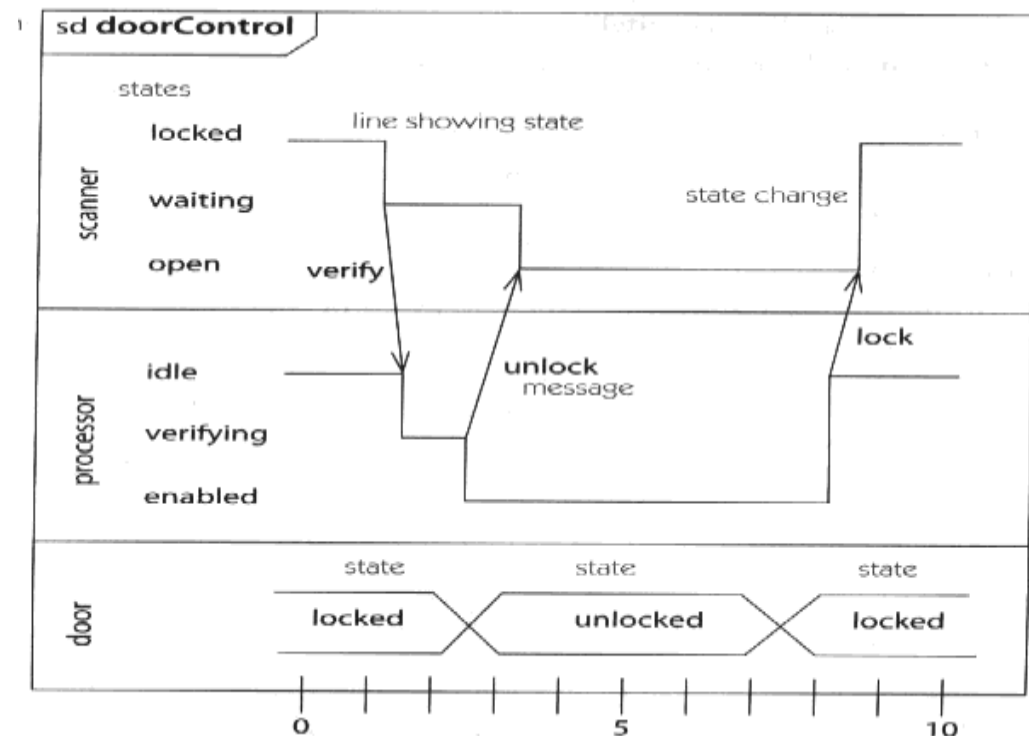




Vremenski dijagram



- *engl. Timing Diagram*
- Vrsta dijagrama interakcije
- Prilagođen za izričit prikaz stvarnih vremena
 - točniji zapis sekvencijskog dijagrama (vidljiv samo relativan odnos poruka)
 - prikazuje promjene stanja na liniji života u vremenu
- pogodan za primjenu u sustavima za rad u stvarnom vremenu (*engl. real-time applications*)





Dijagram profila



- *engl. Profile Diagram*
- Namjena proširenje jezika za stvaranje novih dijagrama
- Statički dijagrami strukture koji pokazuju proširenje postojećih ili nove elemente modeliranja.
- Omogućava definiranje korisničkih stereotipa, vrijednosti, ikona za specifična područja primjene, tehnologije ili metode.
- UML Profil
 - skup predefiniranih stereotipa
 - NE proširuje UML
 - pojednostavljuje primjenu UML-a u drugim domenama primjene



Primjeri UML profila








- UML profile for CORBA
 - UML profile for EJB (Enterprise Java Beans)
 - UML profiles for C++, or for Java,
 - UML profiles for specific RDB
-
- Više na:
http://www.omg.org/technology/documents/profile_catalog.htm



Primjeri UML profila

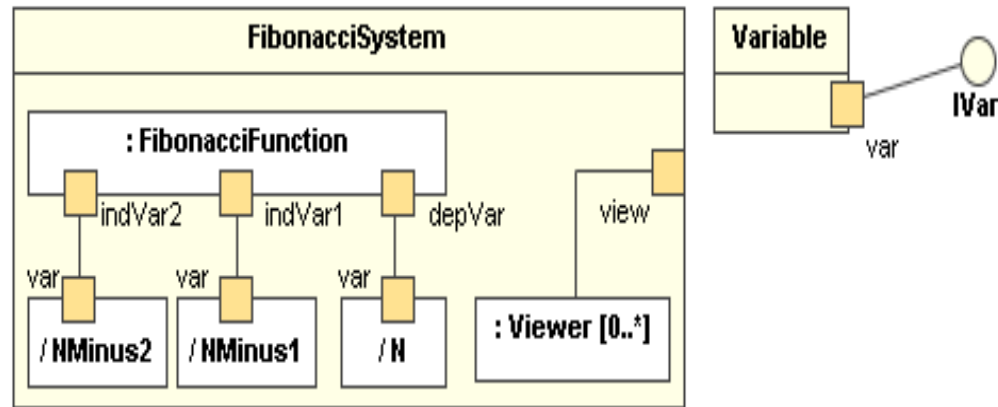


Concept	Mapping to existing UML	UML extension	Description
Vulnerability	Object	<<vulnerability>>  *{vulnerability}	This stereotype is used to distinguish objects containing vulnerabilities from normal objects. The tag vulnerability is used to specify the vulnerability. If more than one vulnerability is present, several vulnerability tags should be used (and numbered).
Unwanted incident	Message	<<unwanted incident>> 	This stereotype is used to distinguish unwanted incidents from normal messages.
Misuser	Actor instance	<<misuser>>  *{type} *{intension}	This stereotype is used to distinguish misusers from normal users (actors). The tag type refers to whether the misuser is insider or outsider. The tag intention refers to whether the misuse is intended or unintended.
Threat	Message	<<threat>> 	This stereotype is used to distinguish threats from normal and abnormal messages.
Asset	Object	<<asset>> 	Stereotype from CORAS UML profile.

Siv Hilde Houmb and Kine Kvernstad Hansen : **Towards a UML Profile for Security Assessment**

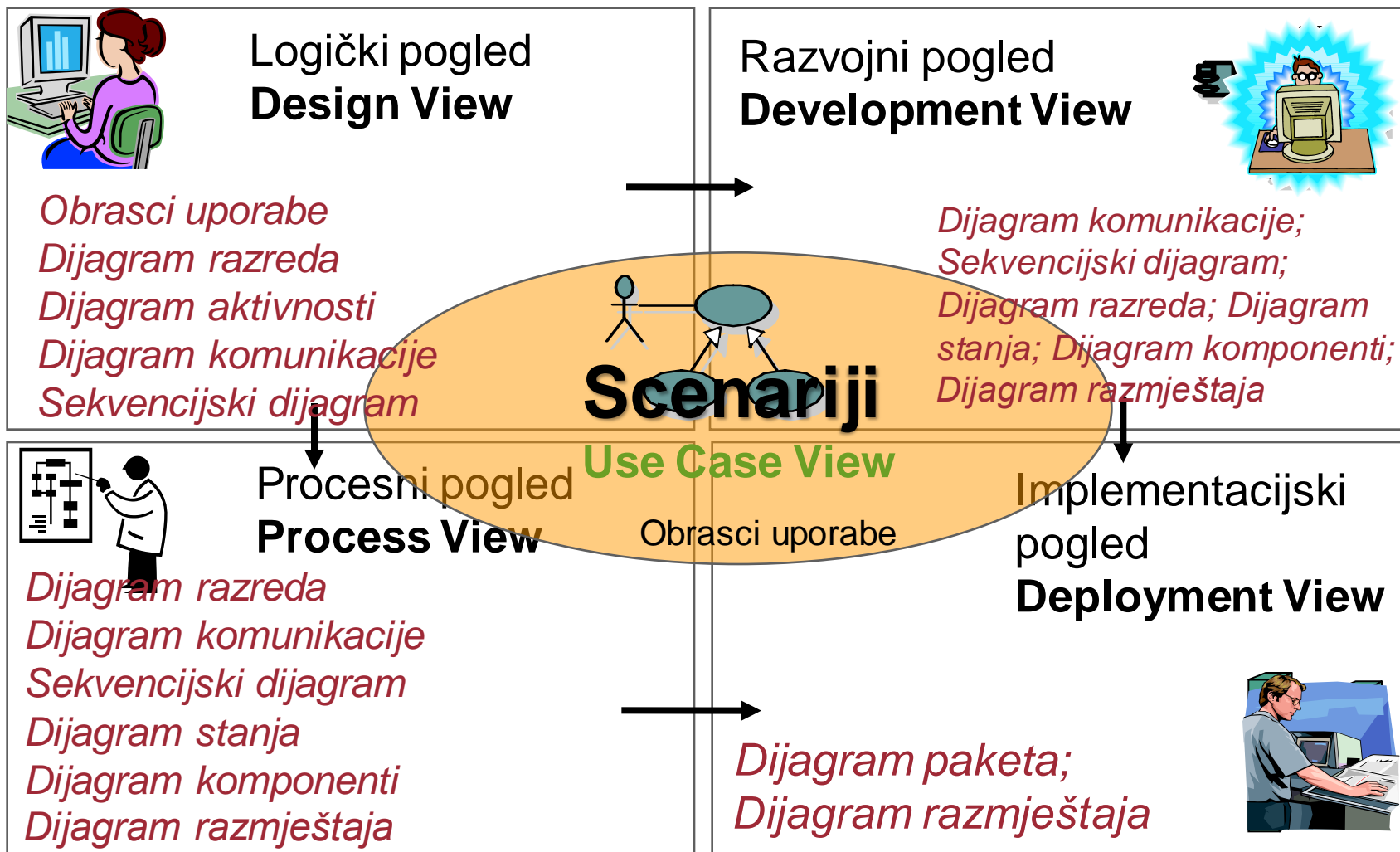
Dijagrami složene strukture razreda

- *engl. Composite Structure Diagram*
- Opisuje konfiguraciju i odnose dijelova koji zajednički ostvaruju neko ponašanje
- Opisuju internu strukturu razreda i kolaboracije koje ta struktura omogućuje.
 - uključuju interne dijelove i sučelja (*engl. ports*) za međusobnu interakciju dijelova i interakciju s vanjskim svijetom, te konektore.





Uporaba UML dijagrama





Diskusija

