

## PROCESI INŽENJERSTVA ZAHTJEVA

Generičke aktivnosti u procesu programskog inženjerstva:

- **Specifikacija** (temeljem analize zahtjeva)
- Razvoj i oblikovanje
- Validacija i verifikacija
- Evolucija

1

## PROCESI INŽENJERSTVA ZAHTJEVA

Cilj ove prezentacije je pokazati kako generirati i dokumentirati zahtjeve, posebice:

- Opisati temeljne aktivnosti u procesima i njihove odnose.
- Upoznati se s tehnikama za izlučivanje i analizu zahtjeva.
- Opisati validaciju zahtjeva i ulogu recenzenta.
- Analizirati upravljanje zahtjevima (*engl. requirements management*) kao potporu procesu inženjerstva zahtjeva.

2

## PROCESI INŽENJERSTVA ZAHTJEVA

Procesi koji su u uporabi u inženjerstvu zahtjeva **razlikuju se** ovisno o domeni primjene, ljudskim resursima i organizaciji koja oblikuje zahtjeve.

Nema jedinstvenog procesa !

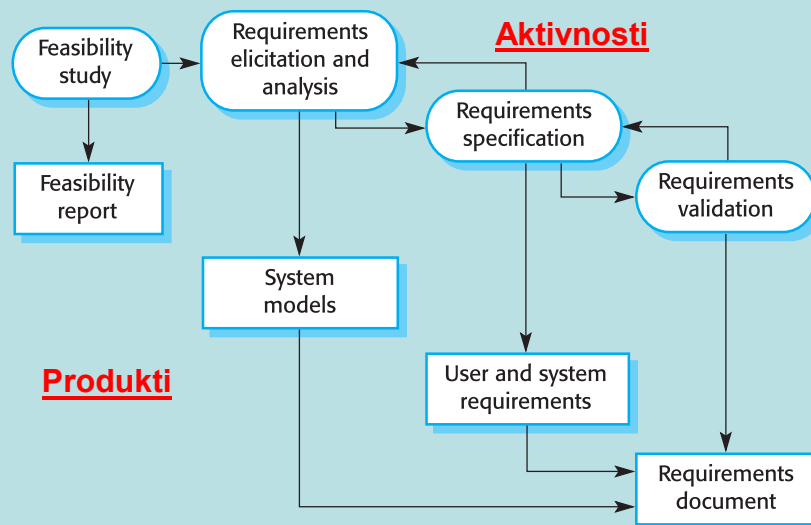
**Dva** su uobičajena **modela** procesa inženjerstva zahtjeva (klasični i spiralni).

Postoje neke **generičke aktivnosti** zajedničke svim procesima:

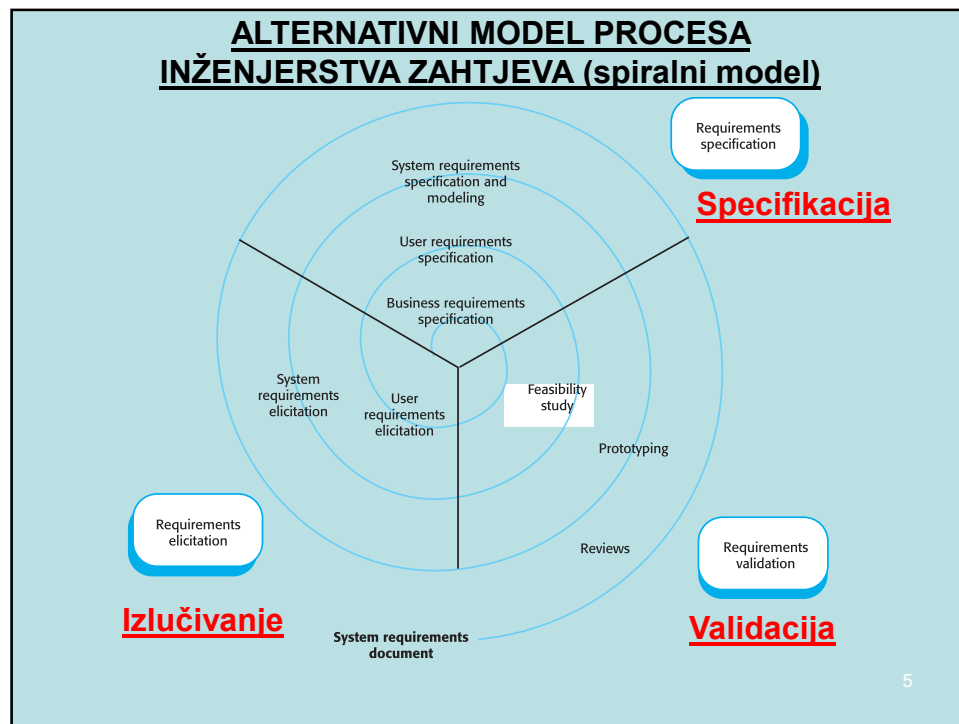
- **Studija izvedivosti** (*engl. feasibility study*)
- **Izlučivanje (otkrivanje) zahtjeva** (*engl. requirements elicitation*)
- **Analiza zahtjeva**
- **Validacija zahtjeva**
- **Upravljanje zahtjevima** (promjenama)

3

## KLASIČNI MODEL PROCESA INŽENJERSTVA ZAHTJEVA



4



### Spiralni model inženjerstva zahtjeva

- **Tro-stupanjska aktivnost** (specifikacija, validacija, izlučivanje).
- Promatra proces inženjerstva zahtjeva kroz **iteracije**.
- U svakoj iteraciji je **različit intenzitet aktivnosti** (u ranim iteracijama fokus na razumijevanju poslovnog modela, u kasnijim modeliranje sustava).
- Zahtjevi se u pojedinim iteracijama specificiraju s **različitom razinom detalja**.

## STUDIJA IZVEDIVOSTI (*engl. feasibility study*)

**Studija izvedivosti** na početku procesa inženjerstva zahtjeva određuje da li se predloženi sustav isplati (t.j. da li je vrijedan uloženi sredstava). Ulazna informacija je preliminarni skup zahtjeva poslovnog procesa.

To je kratka fokusirana studija koja **u pisanom dokumentu** provjerava i odgovara na pitanja:

- Da li sustav doprinosi ciljevima organizacije u koju se uvodi ?
- Da li se sustav može izvesti postojećom tehnologijom i predviđenim sredstvima ?
- Da li se predloženi sustav može integrirati s postojećim sustavima organizacije u koju se uvodi ?

7

## PROVEDBA STUDIJE IZVEDIVOSTI

Temelji se na određivanju **koje informacije su potrebne** za studiju, **prikupljanje** informacija i **pisanju izvješća**.

Pitanja za ljude u organizaciji:

- Što ako se sustav ne implementira ?
- Koji su trenutni problemi procesa organizacije ?
- Kako bi predloženi sustav pomogao u poboljšanju procesa ?
- Koji se problemi mogu očekivati pri integraciji novoga sustava ?
- Da li je potrebna nova tehnologija ili nove vještine ?
- Koje dodatne resurse organizacije traži implementacija novoga sustava ?

8

## IZLUČIVANJE I ANALIZA ZAHTJEVA

Poznato i kao izlučivanje (*engl. elicitation*) zahtjeva ili otkrivanje (*engl. discovery*) zahtjeva.

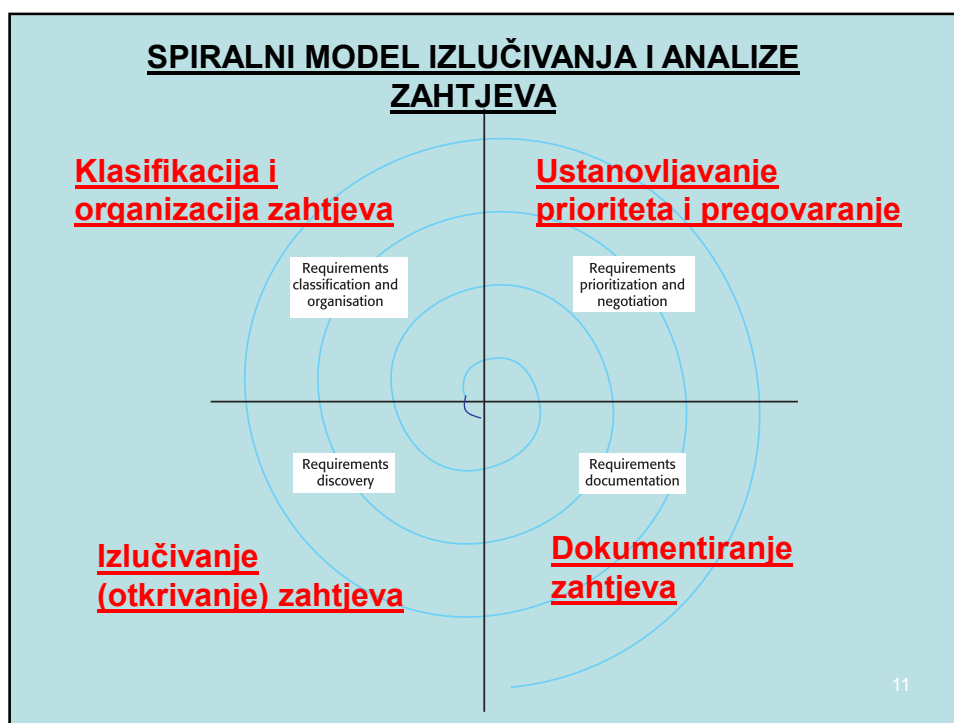
- Uključuje stručno tehnički obrazovano osoblje koje u zajedničkom radu s kupcima razjašnjava domenu primjene, definira usluge koje sustav treba pružiti, te određuje ograničenja u radu sustava.
- Može uključivati: krajnjeg korisnike sustava, rukovoditelje, inženjere uključene u održavanje sustava, eksperte domene primjene, predstavnike sindikata i sl.
- Svi se oni nazivaju **dionici** (*engl. stakeholders*).

9

## PROBLEMI U IZLUČIVANJU I ANALIZI ZAHTJEVA

- Dionici ne znaju što stvarno žele (teško artikuliraju zahtjeve, ili su zahtjevi nerealni obzirom na cijenu).
- Dionici izražavaju zahtjeve na različite, njima specifične načine (posjeduju implicitno znanje o svojem radu - domeni).
- Različiti dionici mogu imati konfliktne zahtjeve i izražene na različite načine.
- Organizacijski i politički faktori mogu utjecati na zahtjeve (npr. rukovoditelj traži da uvedeni sustav ima značajke koje povećavaju njegov utjecaj u organizaciji).
- Zbog dinamike ekonomskog i poslovnog okruženja zahtjevi se mijenjaju za vrijeme procesa analize.
- Uz promjenu poslovnog okruženja pojavljuju se novi dionici s novim, specifičnim zahtjevima.

10



## **AKTIVNOSTI U SPIRALI IZLUČIVANJA I ANALIZE ZAHTJEVA**

### **Izlučivanje (otkrivanje) zahtjeva**

Interakcija s dionicima s ciljem otkrivanja njihovih zahtjeva. Zahtjevi domene primjene se također definiraju na ovom stupnju. Izvori informacija su dokumenti, dionici, slični sustavi.

### **Klasifikacija i organizacija zahtjeva**

Grupiraju se srodni zahtjevi i organiziraju u koherentne grozdove (klustere).

### **Ustanovljavanje prioriteta i pregovaranje**

Zahtjevi se razvrstavaju po prioritetima i razrješuju konflikti.

### **Dokumentiranje zahtjeva**

Zahtjevi se dokumentiraju (formalnim i neformalnim dokumentima) i ubacuju u slijedeći ciklus spirale.

12

### PRIMJER DIONIKA ZA SUSTAV BANKOMATA:

- Bankovni klijenti
- Predstavnici drugih banaka
- Bankovni rukovoditelji (engl. *manageri*)
- Šalterski službenici
- Administratori baza podataka
- Rukovoditelji sigurnosti
- Marketing odjel
- Inženjeri održavanja sustava (sklopovlja i programske potpore)
- Regulatorna tijela za bankarstvo

13

### POGLEDI (engl. *viewpoints*)

Pogledi su način strukturiranja zahtjeva tako da oslikavaju perspektivu i fokus različitih dionika. Dionici se mogu razvrstati po različitim pogledima.

Ova više-perspektivna analiza omogućuje razrješavanje konflikata.

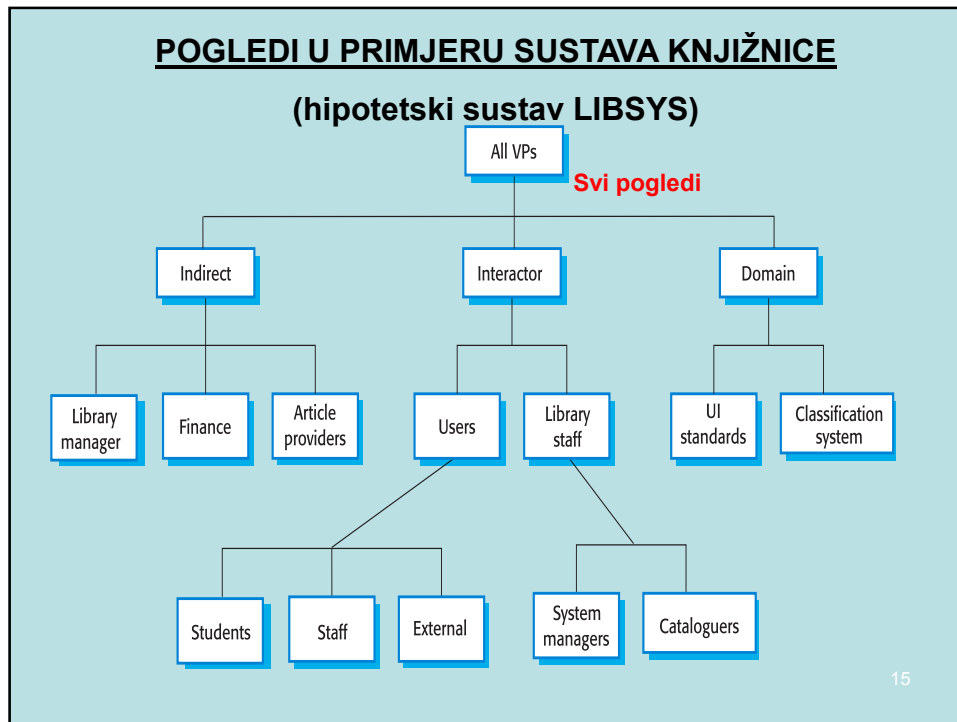
#### Tipovi pogleda

Pogledi interakcije (Ljudi i drugi sustavi koji izravno komuniciraju sa sustavom. Primjer bankomata: klijenti i korisnička baza podataka).

Indirektni pogledi (Dionici koji ne koriste sustav izravno, ali utječu na zahtjeve. Primjer bankomata: rukovoditelji i osoblje zaduženo za sigurnost).

Pogledi domene primjene (Karakteristike domene i ograničenja koja utječu na zahtjeve. Primjer bankomata: standardi u komunikaciji između banaka).

14



## INTERVJUIRANJE kao metoda izlučivanja zahtjeva

U formalnom i neformalnom intervjuiranju tim zadužen za inženjerstvo zahtjeva ispituje dionike o sustavu koji trenutno koriste te o novo predloženom sustavu.

### TIPOVI INTERVJUA:

Zatvoreni intervju u kojem se odgovara na skup prije definiranih pitanja.

Otvoreni intervju u kojem ne postoje definirana pitanja, već se niz pitanja otvara i raspravlja s dionicima.

U praksi intervjui ne daju dobre rezultate za zahtjeve domene primjene jer inženjeri zahtjeva ne razumiju specifičnu terminologiju domene, a eksperti domene toliko poznaju te zahtjeve da ih ne artikuliraju (misle da su svima razumljivi sami po sebi).

16



## SCENARIJI kao metoda izlučivanja zahtjeva

Scenariji su primjeri iz stvarnog života o načinu korištenja sustava. Tijekom izlučivanja zahtjeva dionici diskutiraju i kritiziraju scenarij.

Scenariji trebaju sadržavati:

- Opis početne situacije.
- Opis normalnog (standardnog) tijeka događaja.
- Opis što se eventualno može dogoditi krivo.
- Informaciju o paralelnim aktivnostima.
- Opis stanja gdje scenarij završava.

17

### Primjer scenarija za sustav LIBSYS (1):

#### Početno stanje

Korisnik je pristupio (logirao se) u LIBSYS sustav, i locirao časopis u kojem se nalazi željeni članak.

#### Normalan rad

Korisnik selektira članak za kopiranje. Sustav traži od korisnika informaciju o njegovim pravima (tip pretplate) ili načinu plaćanja. Opcije plaćanja su kreditna kartica ili račun organizacije koja ima pretplatu.

Sustav traži da korisnik potpiše formular o pravima na kopiranje i ostalim detaljima transakcije. To se upisuje u LIBSYS.

Formular o pravima na kopiranje se provjerava, i ako je OK članak u PDF formatu se prosljeđuje do korisničkog računala u sklopu LIBSYS sustava. Korisnik treba odabrati pisač i kopija članka se ispisiše. Ukoliko je članak tipa "samo za ispis", članak se briše sa korisničkog računala nakon što korisnik potvrdi da je ispis završen.

18

**Primjer scenarija za sustav LIBSYS (2):****Što može poći krivo**

Korisnik nije ispravno popunio formular o pravima na kopiranje. U tom slučaju formular se mora ponovo dati korisniku na ispravak. Ako je ponovljeni formular krivo ispunjen, zahtjev korisnika se odbacuje.

Sustav može odbaciti način plaćanja. Korisnikov zahtjev se odbacuje.

Prijenos članka na korisnikovo računalo nije ispravno izveden. Treba ponavljati dok prijenos ne bude uspješan ili dok korisnik ne prekine transakciju.

Članak nije moguće ispisati. Ako članak nije tipa "samo za ispis" drži ga se u radnom prostoru LIBSYS sustava, a u suprotnom članak se izbriše i korisnika se kreditira u visini cijene članka.

**Paralelne aktivnosti**

Prijenos i obrada zahtjeva drugih korisnika LIBSYS sustava.

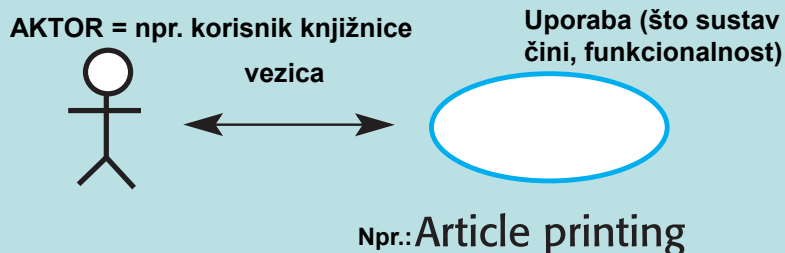
**Završno stanje**

Korisnik i dalje na sustavu. Ako je članak "samo za ispis" briše se.<sup>19</sup>

## **IZLUČIVANJE ZAHTEJEVA OBRASCIMA** **UPORABE** (engl. *use cases*)

**Obrasci uporabe** predstavljaju metodu preuzetu iz **UML standarda**. Temeljeni su **na ideji scenarija**. Skup obrazaca uporabe opisuje sve moguće interakcije sustava.

Uz obrasce uporabe, dodatno se mogu koristiti i drugi dijagrami iz UML sustava (npr. **sekvencijski dijagrami** kako bi se detaljno opisao dinamički tijek događaja.

**Temeljni simboli u dijagramu obrasca uporabe:**

20

## Object Modeling with OMG UML Tutorial Series

# Introduction to UML: Use Cases



Cris Kobryn

Co-Chair UML Revision Task Force  
November 2000



© 1999-2000 OMG and Contributors: Crossmeta, EDS, IBM, Enea Data, Hewlett-Packard, InLine Software, IntelliCorp, Kabira Technologies, Klasse Objecten, ObjectTime Ltd., Rational Software, Unisys

21






## What is use case modeling?

- use case model: a view of a system that emphasizes the behavior as it appears to outside users. A use case model partitions system **functionality** into transactions ('use cases') that are **meaningful to users** ('actors').

Obrasci uporabe dijele funkcionalnosti sustava u transakcije razumljive korisnicima.


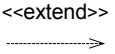

22

## Use Case Modeling: Core Elements

Construct	Description	Syntax
<b>use case</b>	A sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.	
<b>actor</b>	A coherent set of roles that users of use cases play when interacting with these use cases.	
<b>system boundary</b>	Represents the boundary between the physical system and the actors who interact with the physical system.	

23

## Use Case Modeling: Core Relationships

Construct	Description	Syntax
<b>association</b> (pridruživanje)	The participation of an actor in a use case. i.e., instance of an actor and instances of a use case communicate with each other.	
<b>extend</b> (opcijsko, specifično proširenje)	A relationship from an <i>extension</i> use case to a <i>base</i> use case, specifying how the behavior for the extension use case can be inserted into the behavior defined for the base use case.	
<b>generalization</b>	A taxonomic relationship between a more general use case and a more specific use case.	

24

## Use Case Modeling: Core Relationships (cont'd)

Construct	Description	Syntax
<b>include</b> (nužno sadrži)	An relationship from a <i>base</i> use case to an <i>inclusion</i> use case, specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.	<<include>> ----->

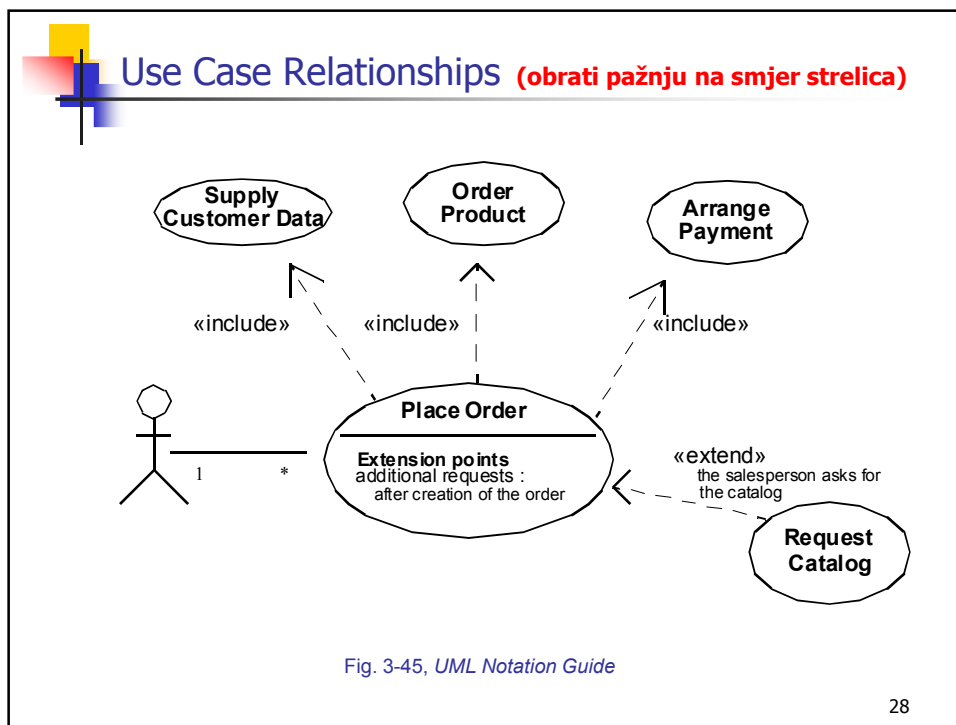
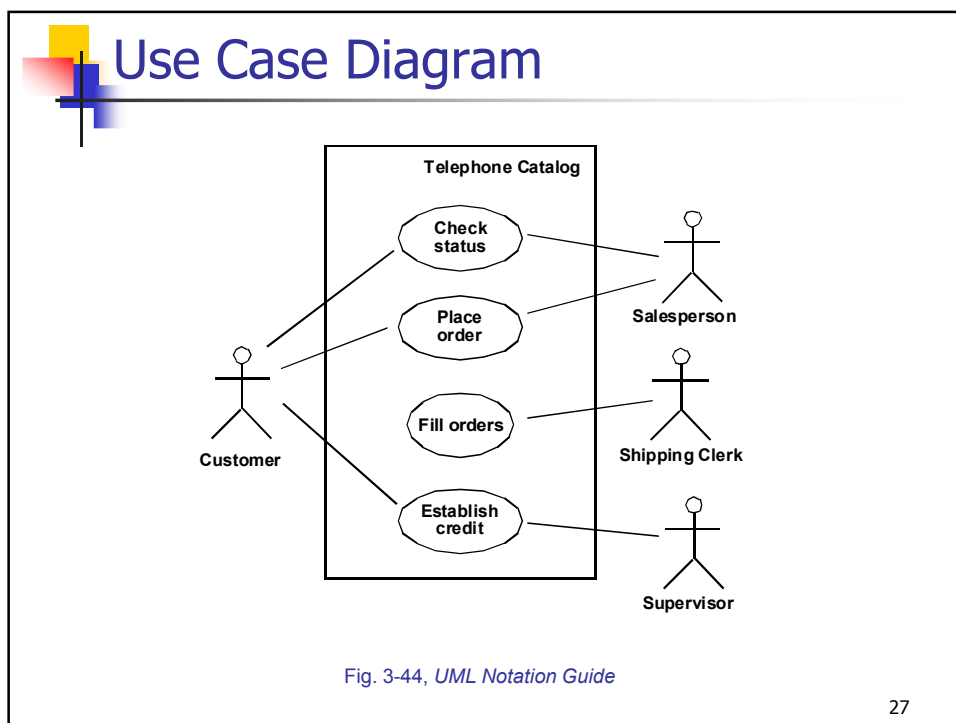
(crtkana ili puna  
strelica često  
nisu u  
konzistentnoj  
uporabi)

25

## Use Case Diagram Tour

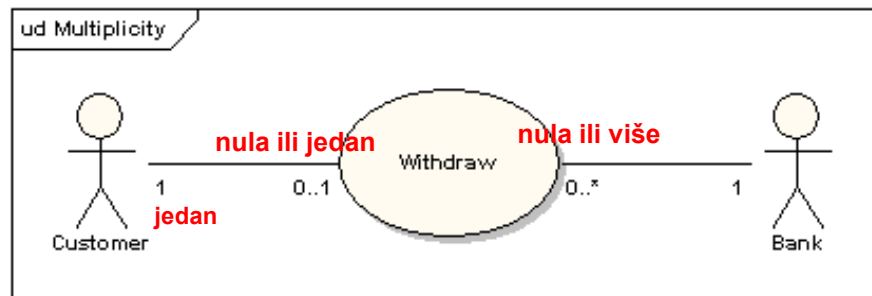
- Shows use cases, actor and their relationships
- Use case internals can be specified by text and/or **interaction diagrams**
- Kinds
  - use case **diagram**
  - use case **description** (t.j. tekstualni opis)

26



(višestrukost)

The uses connector can optionally have **multiplicity values** at each end, as in the following diagram, which shows a customer may only have one withdrawal session at a time, but a bank may have any number of customers making withdrawals concurrently.



29

## Actor Relationships

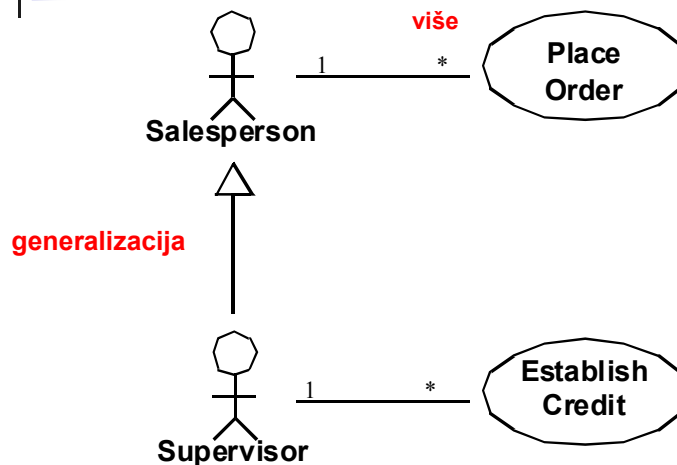


Fig. 3-46, UML Notation Guide

30

## Use Case Textual Description: Ex.:Change Flight

- **Actors:** traveler, client account db, airline reservation system
- **Preconditions:**
  - Traveler has logged on to the system and selected 'change flight itinerary' option
- **Basic course**
  - System retrieves traveler's account and flight itinerary from client account database
  - System asks traveler to select itinerary segment she wants to change; traveler selects itinerary segment.
  - System asks traveler for new departure and destination information; traveler provides information.
  - If flights are available then
  - ...
  - System displays transaction summary.
- **Alternative courses**
  - If no flights are available then ...

31

## When to model use cases

- Model **user requirements** with use cases.
- Model **test scenarios** with use cases.
- If you are using a use-case driven method
  - start with use cases and derive your structural and behavioral models from it.
- If you are not using a use-case driven method
  - make sure that your use cases are consistent with your structural and behavioral models.

32

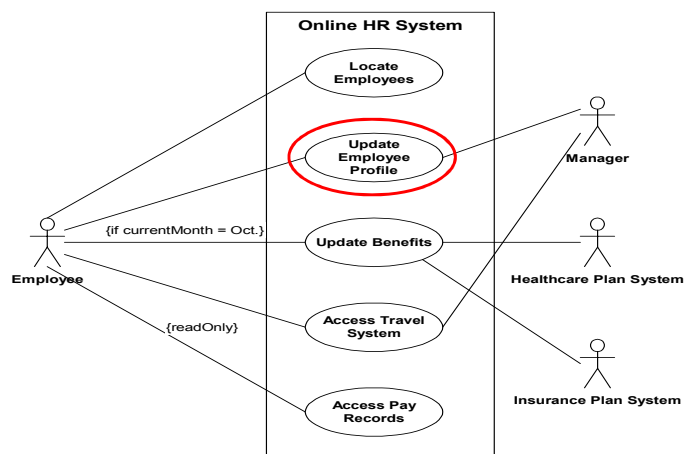


## Preporuke u oblikovanju obrazaca uporabe

- Svaki obrazac uporabe mora sadržavati značajan dio uporabe sustava i biti razumljiv ekspertima domene i programerima.
- Ako se obrasci uporabe definiraju tekstem sve imenice i glagole treba koristiti razumljivo i konzistentno kako bi se kasnije mogli definirati ostali (UML) dijagrami.
- Ako su obrasci uporabe nužni, koristi **<include>**.
- Ako su obrasci uporabe kompletirani a postoje opcije, koristi **<extend>**.
- Dijagram obrazaca uporabe treba sadržavati obrasce uporabe jednake apstraktne razine.
- Uključiti samo zaista potrebne aktore.
- Veliki broj obrazaca uporabe treba organizirati u pakete.

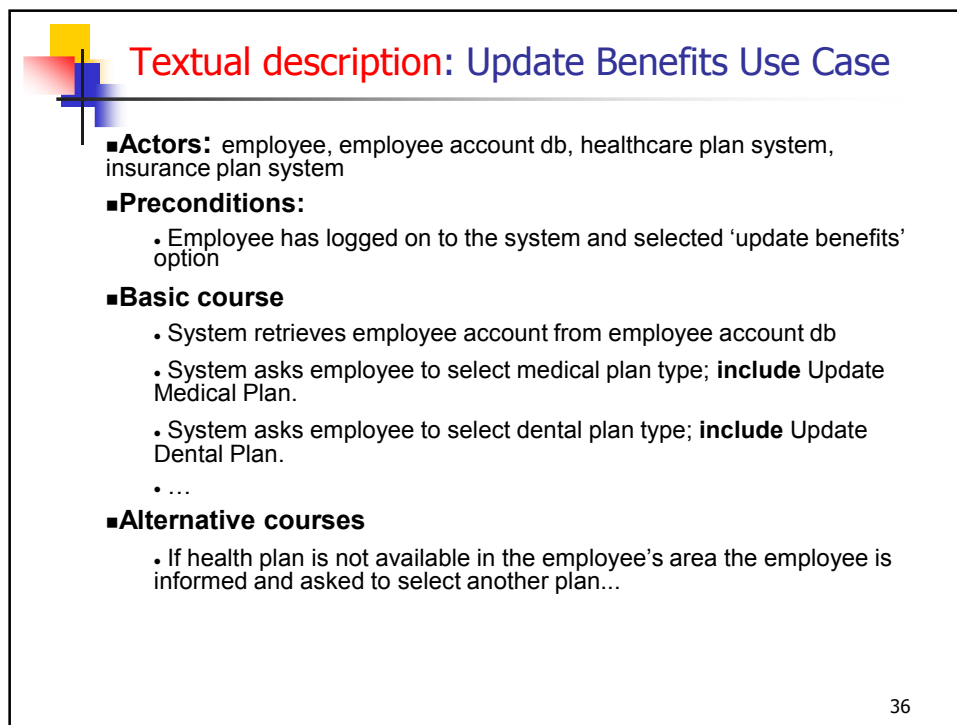
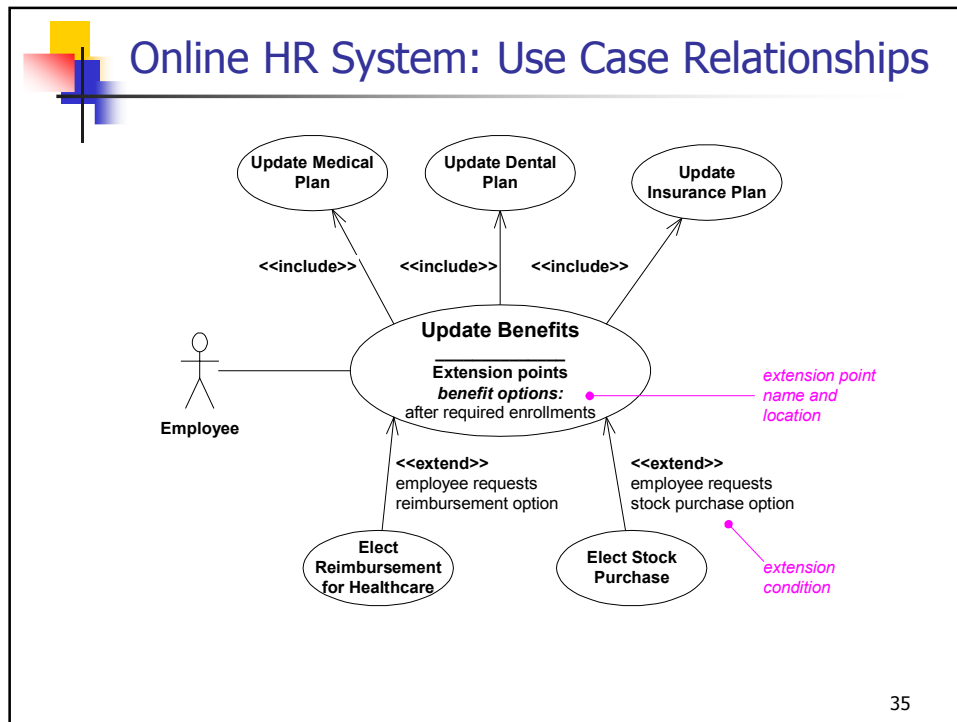
33

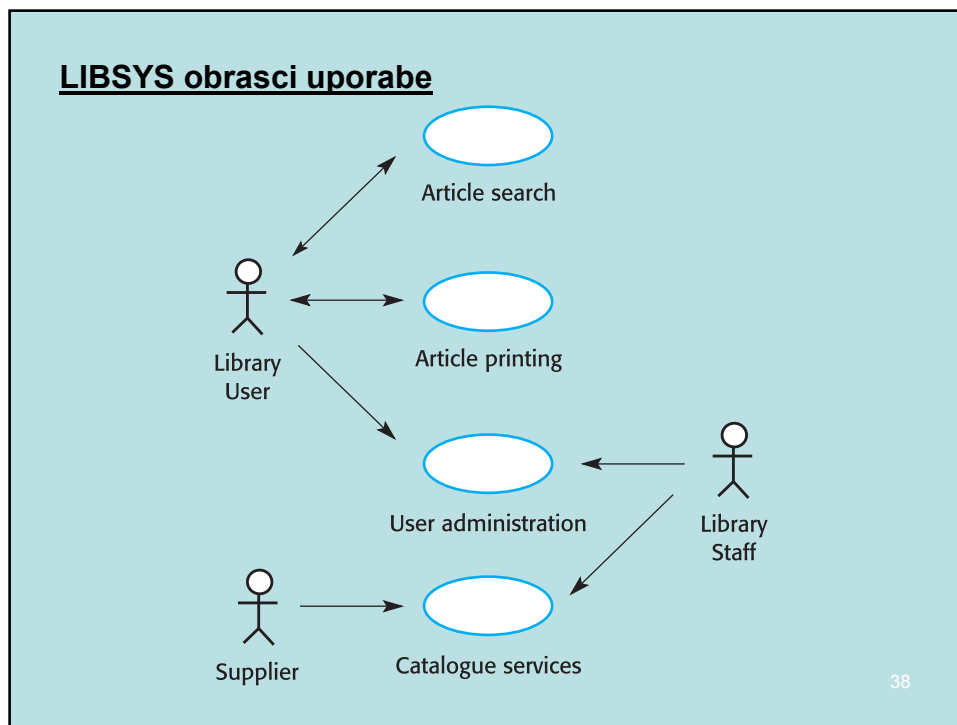
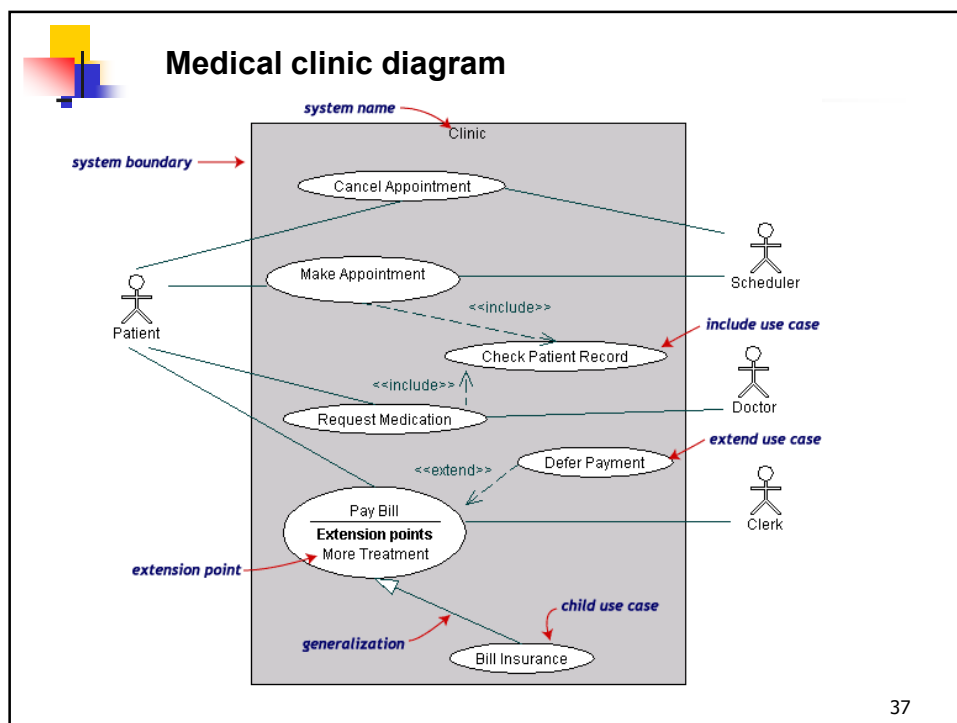
## Example: Online HR System - diagram



“on-line” sustav za upravljanje ljudskim resursima

34





## ZAKLJUČCI o UML obrascima uporabe kao metodi izlučivanja i analize zahtjeva

- UML dijagrami i tekstualni opisi obrazaca uporabe modeliraju funkcionalne zahtjeve sustava.
- Temeljeni su na ideji scenarija.
- Služe za izlučivanje zahtjeva prema pogledu interakcije.
- Pogodni su za modeliranje velikih i složenih programskih produkata.
- Jednostavni su za razumijevanje ali posjeduju i napredne značajke za ekspertne analitičare, arhitekte i programere.
- Specificiraju sustave neovisno o načinu implementacije.
- Mali skup konstrukcija (10% do 20%) koristi se u 80% do 90% mjesta u sustavu.

39

## Detaljniji razvoj i prikaz scenarija u izlučivanju, analizi i dokumentiranju zahtjeva:

### DINAMIČKE INTERAKCIJE U SUSTAVU

(modeliranje ponašanja, *engl. behavioral modeling*)

- Obrasci uporabe identificiraju individualne **interakcije** u sustavu.
- Dodatne informacije u inženjerstvu zahtjeva uz obrasce uporabe slijede iz UML dijagrama koji pokazuju **aktore** involvirane u interakciji, **entitete (objekte, instancije)** s kojima su u interakciji i **operacije** pridružene tim objektima.

To su UML dinamički dijagrami interakcija:

sekvencijski i kolaboracijski.

Izvori:

G. Overgaard, B. Selic, C. Bock, OMG, 2000.

40

## What are interactions?

- Interaction: a collection of communications between instances, including all ways to affect instances, like operation invocation, as well as creation and destruction of instances
- The communications are partially ordered (in time)

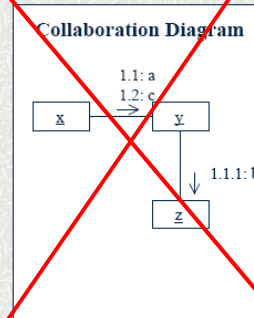
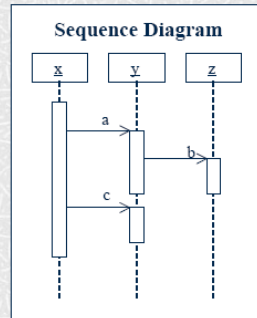
41

## When to Model Interactions

- To specify how the instances are to interact with each other.
- To identify the interfaces of the classifiers.
- To distribute the requirements.

42

## Interaction Diagrams



(nisu temeljni u  
inženjerstvu  
zahtjeva)

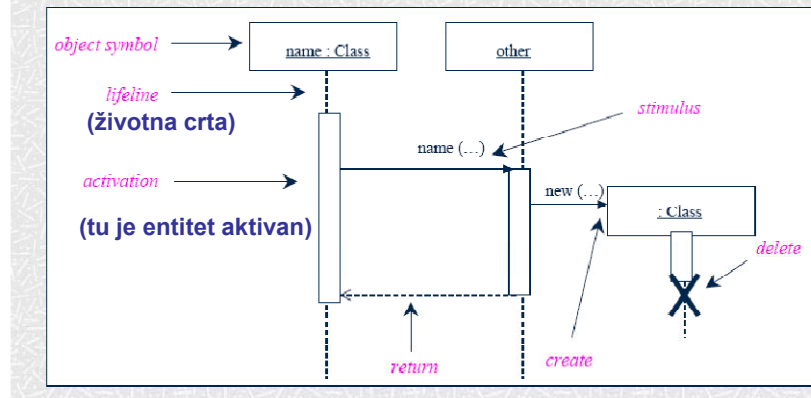
43

## Interaction Modeling Tips

- Set the context for the interaction.
- Include **only** those features of the instances that are relevant. **obilježja**
- Express the flow from left to right and from top to bottom.
- Put active instances to the left/top and passive ones to the right/bottom.
- Use sequence diagrams
  - to show the explicit ordering between the stimuli
  - when modeling real-time **(tada su obavezni)**

44

## Sequence Diagram



45



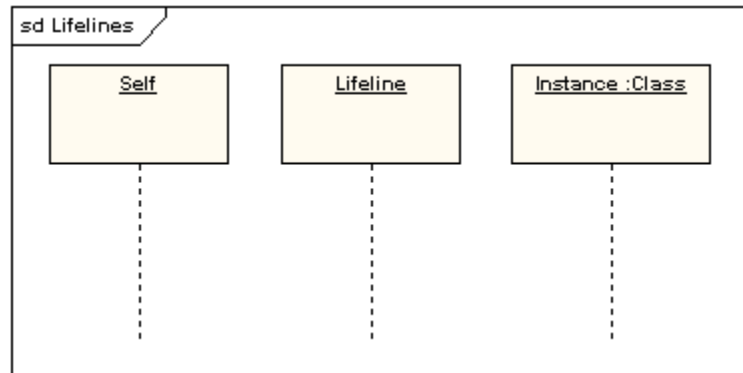
### Sequence Diagrams

A sequence diagram is a form of interaction diagram which shows objects as lifelines running down the page, with their interactions over time represented as messages drawn as arrows from the source lifeline to the target lifeline. Sequence diagrams are good at showing which objects communicate with which other objects; and what messages trigger those communications. Sequence diagrams are not intended for showing complex procedural logic.

46

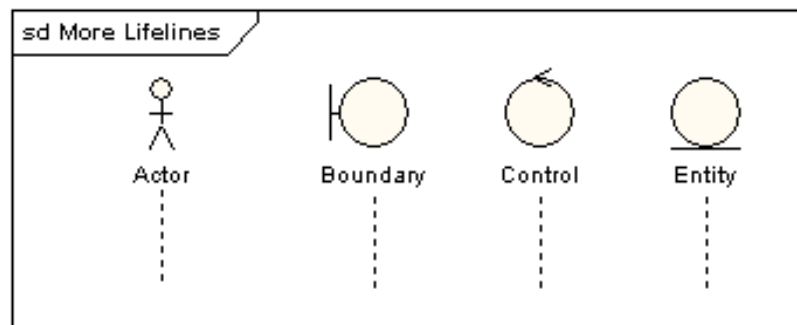
### Lifelines

A lifeline represents an individual participant in a sequence diagram. A lifeline will usually have a rectangle containing its object name.



47

Sometimes a sequence diagram will have a lifeline with an actor element symbol at its head. This will usually be the case if the sequence diagram is owned by a use case. Boundary, control and entity elements from robustness diagrams can also own lifelines

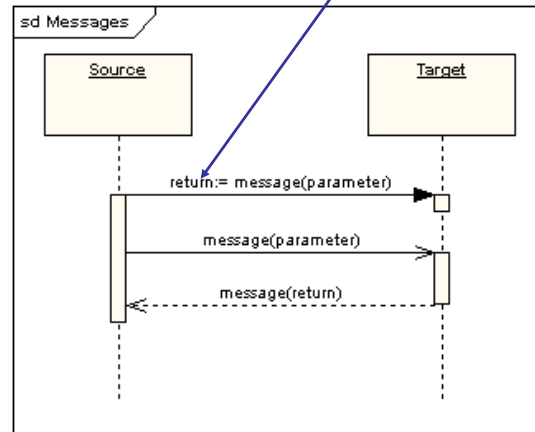


48



### Messages

Messages are displayed as arrows. Messages can be complete, lost or found; synchronous or asynchronous; call or signal. In the following diagram, the first message is a **synchronous** message (denoted by the **solid arrowhead**) complete with an **implicit return** message; the second message is **asynchronous** (denoted by **line arrowhead**), and the third is the asynchronous return message (denoted by the dashed line).

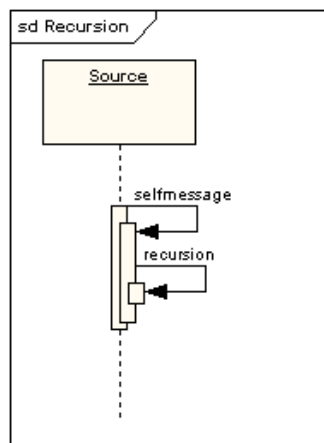


(synchronous message = sender waits for the result)

49

### Self Message

A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object. It is shown as creating a nested focus of control in the lifeline's execution occurrence



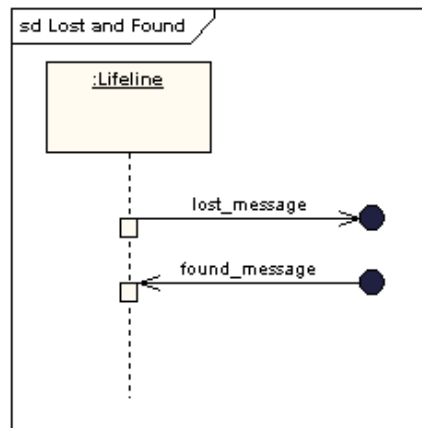
Poruka samom sebi :  
(različite procedure,  
funkcije metode u objektu).

Rekurzija: (ista procedura,  
funkcija, metoda).

50

### Lost and Found Messages

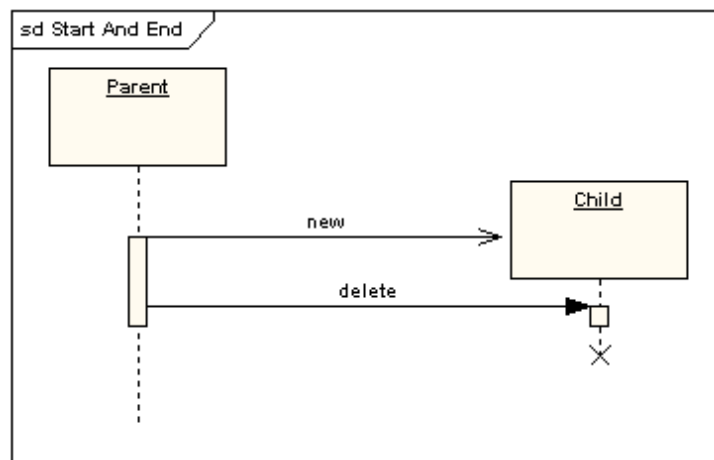
(izgubljene i nađene poruke). Lost messages are those that are either sent but do not arrive at the intended recipient, or which go to a recipient not shown on the current diagram. Found messages are those that arrive from an unknown sender, or from a sender not shown on the current diagram. They are denoted going to or coming from an endpoint element.



51

### Lifeline Start and End

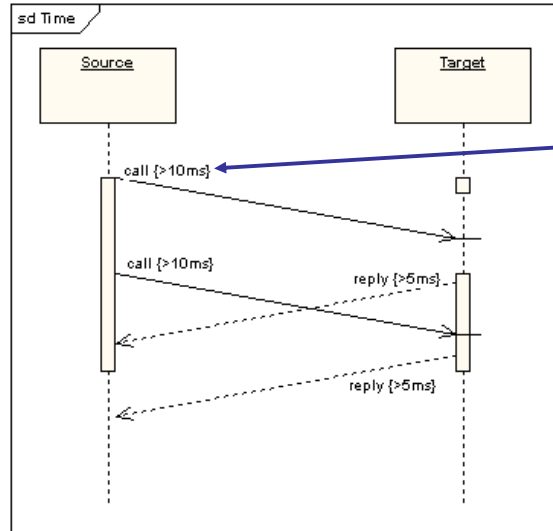
A lifeline may be created or destroyed during the timescale represented by a sequence diagram. The following diagram shows an object being created and destroyed.



52

### Duration and Time Constraints (ograničenja trajanja)

By default, a message is shown as a horizontal line. By setting a duration constraint for a message, the message will be shown as a sloping line.

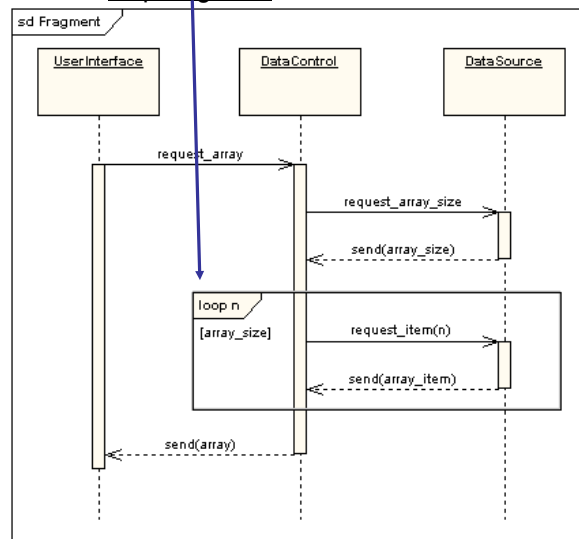


(Time from  
send to receive.  
e.g. slow  
communication  
path).

53

### Loop fragment

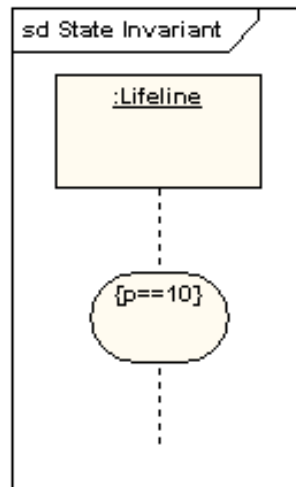
encloses a series of messages which are repeated. The following diagram shows a loop fragment.



54

### State Invariant / Continuations

A state invariant is a constraint placed on a lifeline that must be true at run-time. It is shown as a rectangle with semi-circular ends.



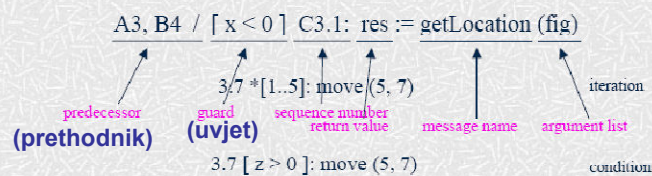
55

## Arrow Label

*predecessor guard-condition sequence-expression return-value := message-name argument-list*

move (5, 7)

3.7.4: move (5, 7)

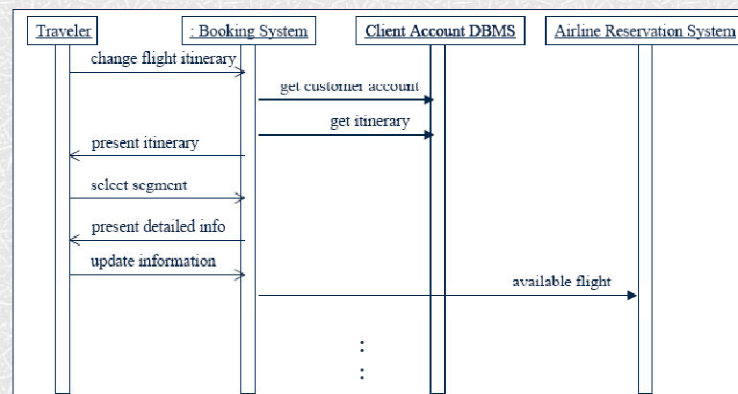


56

**Primier:****Use Case Description: Change Flt Itinerary**

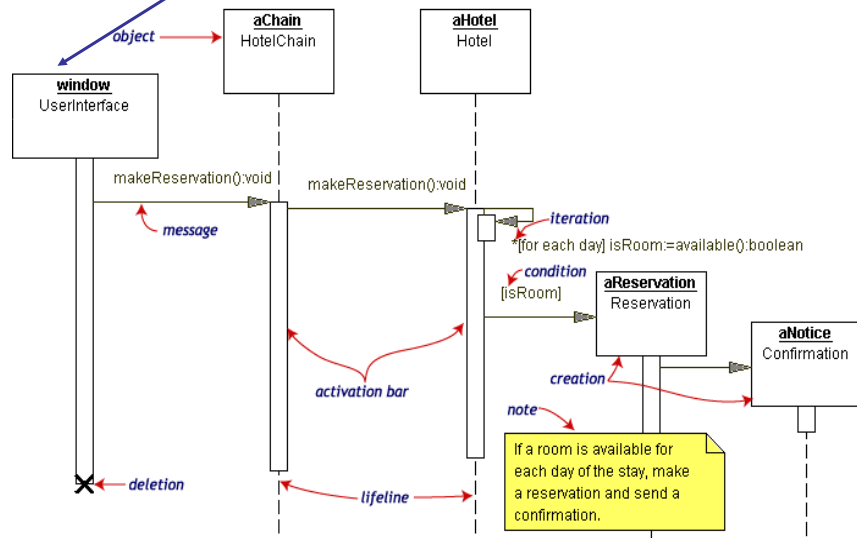
- **Actors:** traveler, client account db, airline reservation system
- **Preconditions:** Traveler has logged in
- **Basic course:**
  - Traveler selects 'change flight itinerary' option
  - System retrieves traveler's account and flight itinerary from client account database
  - System asks traveler to select itinerary segment she wants to change; traveler selects itinerary segment.
  - System asks traveler for new departure and destination information; traveler provides information.
  - If flights are available then ...
  - ...
  - System displays transaction summary.
- **Alternative course:**
  - If no flights are available then...

57

**Sequence Diagram: Change Flight Itinerary**

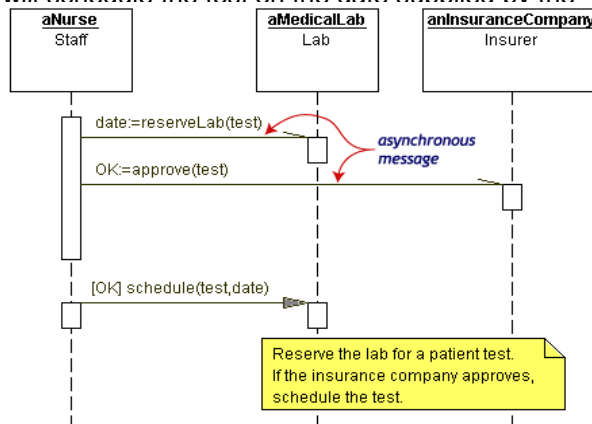
58

**Primjer:** Hotel reservation. The object initiating the sequence of messages is a **Reservation window**.



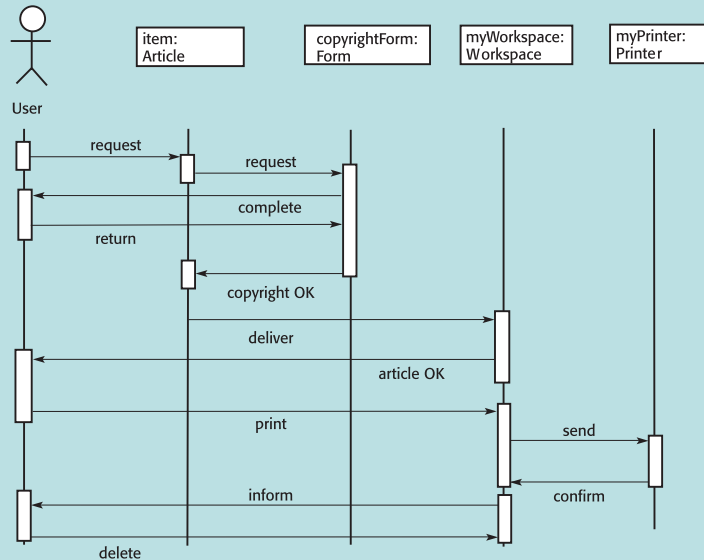
**Primjer:** Nurse requesting a diagnostic test

Two asynchronous messages from the **Nurse**: 1) ask the **MedicalLab** to reserve a date for the test and 2) ask the **InsuranceCompany** to approve the test. The order in which these messages are sent or completed is irrelevant. If the **InsuranceCompany** approves the test, then the **Nurse** will schedule the test on the date supplied by the **MedicalLab**.



## LIBSYS - dodatni dijagram sekvenci za ispis članka

(asinkrone povratne poruke nisu označene crtkano kao što bi trebale biti)



61

## UTJECAJ SOCIJALNIH I ORGANIZACIJSKIH ČIMBENIKA U INŽENJERSTVU ZAHTJEVA

Programski produkti se uvijek koriste u socijalnom i organizacijskom kontekstu. To može uvelike utjecati i čak dominirati na zahtjeve sustava.

Socijalni i organizacijski čimbenici nisu jedinstven pogled, već utjecaj na sve poglede.

Oblikovanje programske potpore mora to uvažiti, ali trenutno **ne postoji sistematski postupak** kako se to može uključiti u analizu zahtjeva.

Ne postoje jednostavni modeli za opis obavljanja nekog posla. Ako i postoje, to su modeli zasnovani na prošlim a ne obrascima ponašanja na novom poslu.

Djelomično se tome može doskočiti izradom prototipa, te praćenjem rada na njemu.

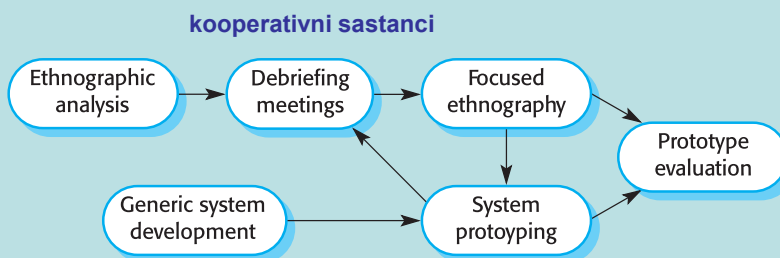
62

## ETNOGRAFIJA

je znanost koja istražuje kako ljudi stvarno rade, a ne kako bi definicija poslovnog procesa to propisivala.

Koristi se za izlučivanje zahtjeva uzimajući u obzir aktivnosti drugih sustavom involviranih ljudi.

### Fokusirana etnografija (etnografija + prototip):



63

## VALIDACIJA ZAHTJEVA

Cilj validacije je pokazati da zahtjevi odgovaraju sustavu koji naručitelj doista želi.

Naknadno ispravljanje pogreške u zahtjevima može biti 100 puta skuplje od ispravljanje pogreške u implementaciji.

### TEHNIKE VALIDACIJE:

- **Recenzija** zahtjeva (sistematska ručna analiza zahtjeva).
- **Izrada prototipa** (provjera zahtjeva na izvedenom sustavu).
- **Generiranje ispitnih slučajeva** (razvoj ispitnih sekvenci za provjeru zahtjeva).

64



## ŠTO SE U ZAHTEJIMA PROVJERAVA ?

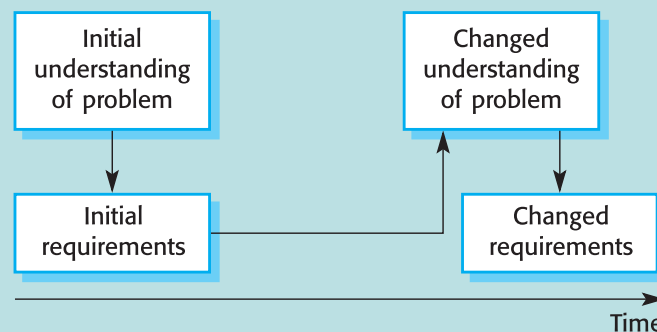
- **VALJANOST** (da li sustav osigurava funkcije koje podupiru potrebe korisnika ?).
- **KONZISTENCIJA** (da li postoji konflikt u zahtjevima ?).
- **KOMPLETNOST** (da li sustav uključuje sve funkcije koje je korisnik tražio ?).
- **REALNOST** (da li se sve funkcije mogu implementirati uz danu tehnologiju i proračun ?).
- **PROVJERLJIVOST** (da li se svi zahtjevi mogu provjeriti ?).
- **RAZUMLJIVOST** (da li je dokument jasno napisan ?).
- **SLIJEDIVOST** (da li je naveden izvor dokumenta ?).
- **ADAPTABILNOST** (mogu li se zahtjevi mijenjati bez velikog utjecaja na druge zahtjeve ?).

65

## UPRAVLJANJE PROMJENAMA U ZAHTEJIMA

Promjene nastupaju zbog promijenjenog modela poslovanja, boljeg razumijevanja procesa tijekom razvoja ili konfliktnim zahtjevima u različitim pogledima.

### EVOLUCIJA ZAHTEJEVA:



66

## KLASIFIKACIJA PROMJENA ZAHTJEVA

- Okolinom promijenjeni zahtjevi

Promjena zbog promjene okoline u kojoj organizacija posluje (npr. bolnica mijenja financijski model pokrivanja usluga).

- Novonastali zahtjevi (zahtjevi koji se pojavljuju kako kupac sve bolje razumije sustav koji se oblikuje).

- Posljedični zahtjevi (zahtjevi koji nastaju nakon uvođenja sustav u eksploataciju, a rezultat su promjena procesa rada u organizaciji nastalih upravo uvođenjem novoga sustava).

- Zahtjevi kompatibilnosti (zahtjevi koji ovise o procesima drugih sustava u organizaciji; ako se ti sustavi mijenjaju to traži promjenu zahtjeva i novo uvedeni sustav).

67

## ZAKLJUČCI

- Proces inženjerstva zahtjeva uključuje studiju izvedivosti, izlučivanje zahtjeva i analizu, specifikaciju zahtjeva i rukovanje (upravljanje promjenama) zahtjevima.
- Izlučivanje i analiza zahtjeva je iterativan proces koji uključuje razumijevanje domene primjene, prikupljanje, klasifikaciju, strukturiranje, sastavljanje prioriteta i validaciju zahtjeva.
- Sustavi imaju više dionika s različitim zahtjevima.
- Socijalni i organizacijski čimbenici utječu na zahtjeve sustava.
- Validacija zahtjeva bavi se valjanošću, konzistentnošću, kompletnošću, realizmom u izvedbi i provjerljivošću.
- Promjene načina poslovanja nužno mijenjaju zahtjeve.
- Rukovanje zahtjevima uključuje planiranje i upravljanje promjenama u zahtjevima.

68