TEHNIČKO VELEUČILIŠTE U ZAGREBU
POLYTECHNICUM ZAGRABIENSE

Objektno orijentirani razvoj programa

ISVU: 130938/19970

Dr. sc. Danko Ivošević, dipl. ing.
Predavač

Akademska godina 2021./2022.
Ljetni semestar

Objektno orijentirani razvoj programa

MODELIRANJE SUSTAVA

Referenca

- I. Sommerville, Software engineering, 10th ed., Pearson, 2016., https://iansommerville.com/software-engineering-book/

Modeliranje sustava

- proces razvoja apstraktnih modela sustava
- svaki model predstavlja različiti pogled ili perspektivu sustava.

- predstavljanje sustava uporabom grafičke notacije - skoro uvijek se temelji na UML-u

- pomoć za razumijevanje sustava analitičaru i za komunikaciju s korisnicima

Postojeći i/ili planirani sustavi

- Za postojeće sustave:
  - Pomažu u pojašnjavanju što sustav radi
  - Temelj rasprave o jakim i slabim stranama sustava što može dovesti do novih zahtjeva
- Za nove sustave:
  - Tijekom inženjerstva zahtjeva
  - Dionici - za razjašnjavanje zahtjeva dionicima
  - Inženjeri - za predstavljanje prijedloga oblikovanja sustava i dokumentiranje ostvarenja
- U procesu *modelno usmjerenog inženjerstva:*
  - Generiranje ostvarenja sustava na temelju modela (djelomično ili u potpunosti)

Gledište na sustav (perspektiva)

- Vanjsko
  - modeliranje konteksta ili okoline sustava
- Interakcijsko
  - modeliranje međudjelovanja između sustava i okoline ili između dijelova sustava
- Strukturno
  - modeliranje organizacije sustava ili strukture podataka koji se obrađuju u sustavu
- Ponašajno
  - modeliranje dinamike ponašanja sustava i načina odgovora na događaje (poticaje)

Vrste UML dijagrama

- Dijagrami aktivnosti
- Dijagrami obrazaca uporabe
- Dijagrami slijeda
- Dijagrami razreda
- Dijagrami stanja

Uporaba grafičkih modela

- Kao potpore raspravi o sustavu
  - Mogu biti nepotpuni i nedovoljno točni
- Kao način dokumentacije sustava
  - Trebaju biti precizni, ali ne moraju biti potpuni
- Kao detaljan opis sustava na temelju kojega se može generirati ostvarenje sustava
  - Moraju biti i točni i potpuni

MODELI KONTEKSTA
(ILI KONTEKSTUALNI MODELI)

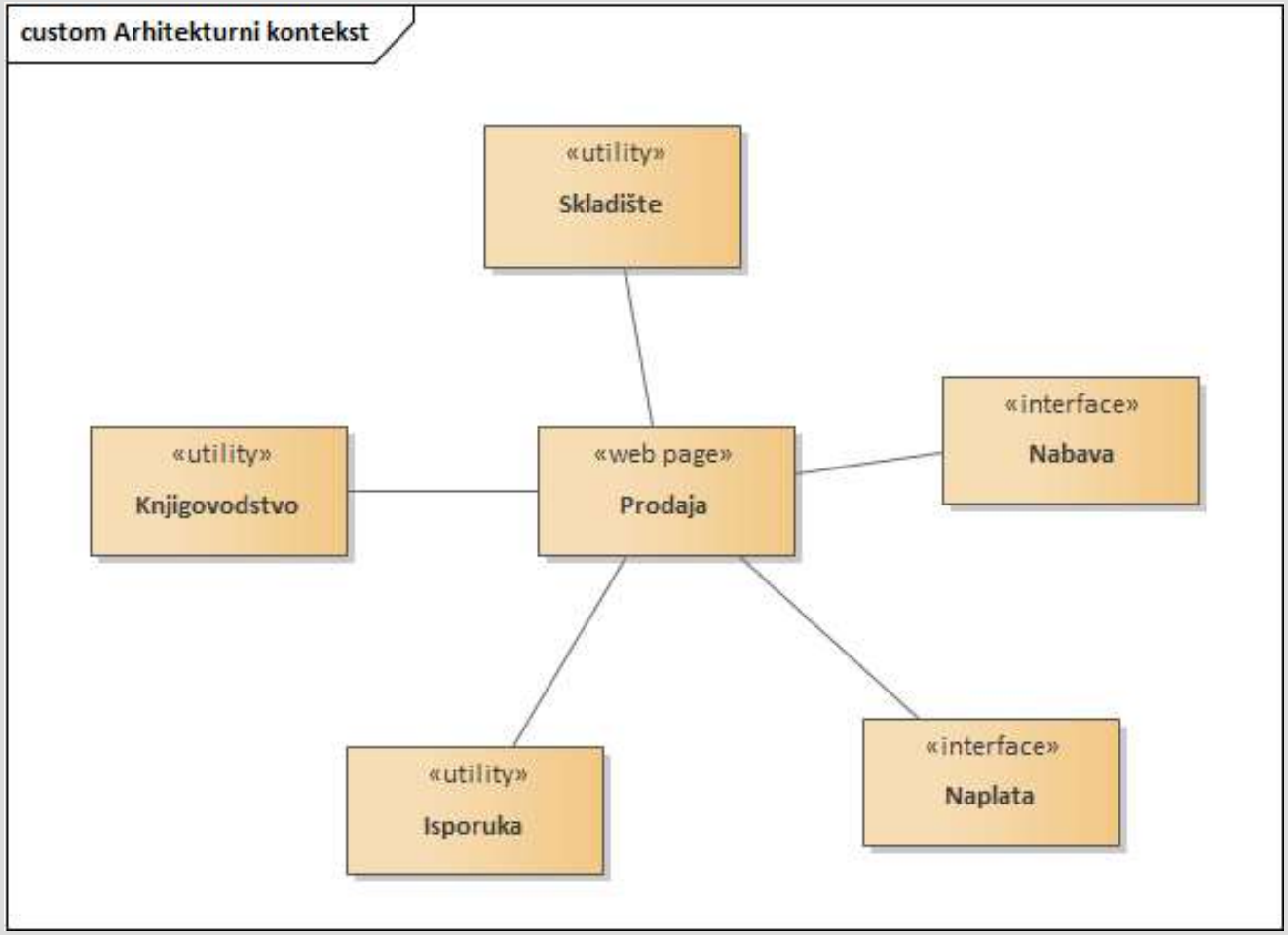
Kontekstualni modeli

- prikaz operativnog konteksta sustava
  - Što se nalazi unutar a što izvan granica sustava?
  - Koji su drugi sustavi u okruženju?
- gledište arhitekture sustava ➔ arhitekturni model

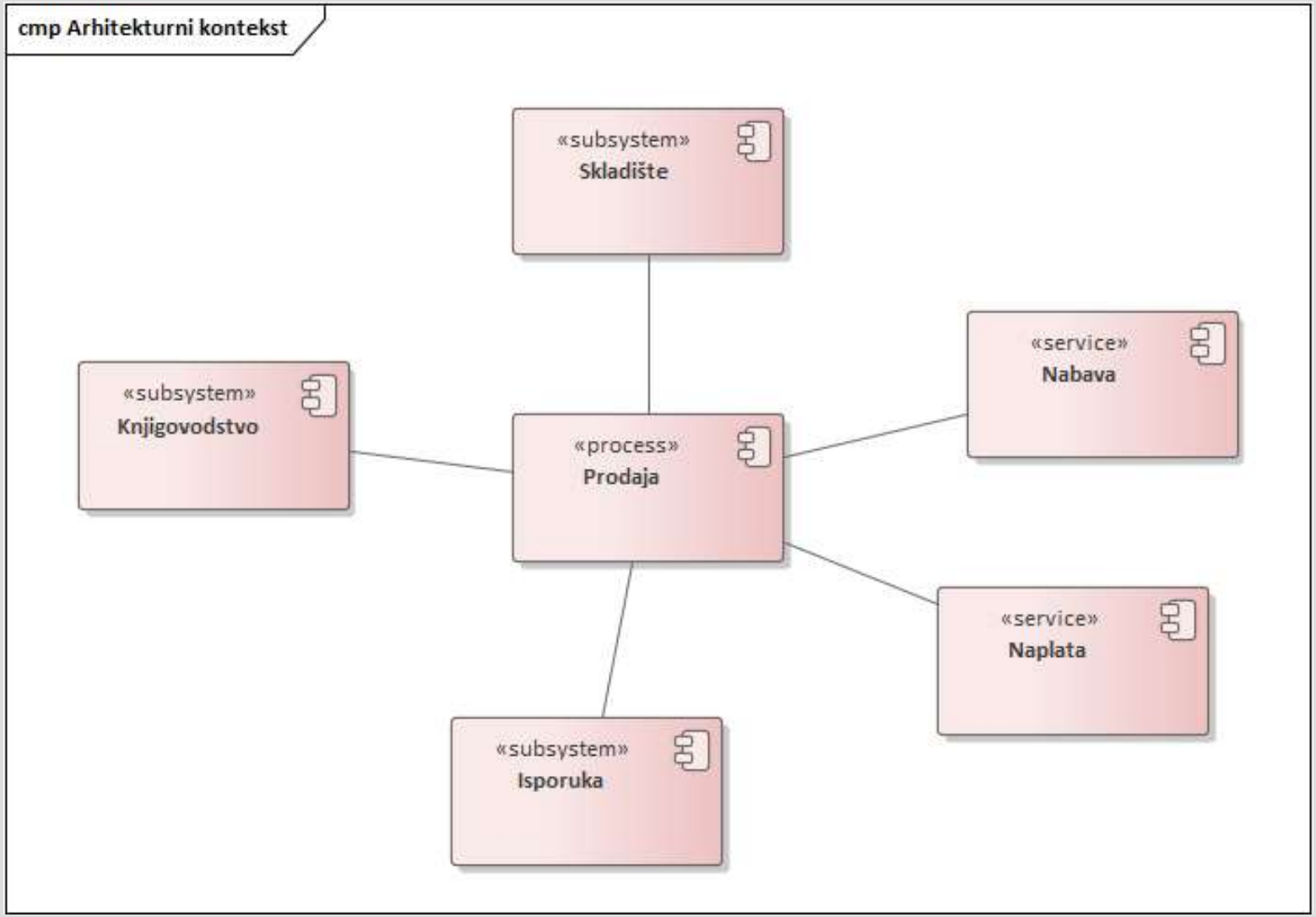
Granice sustava

- prikazuje i druge sustave koji se koriste ili ovise o našem sustavu
- pozicija granice
  - društveno, organizacijsko ili političko pitanje
  - određuje zahtjeve sustava
  - utječe na raspodjelu posla između različitih dijelova organizacije

Arhitekturni kontekst sustava prodaje auto-dijelova



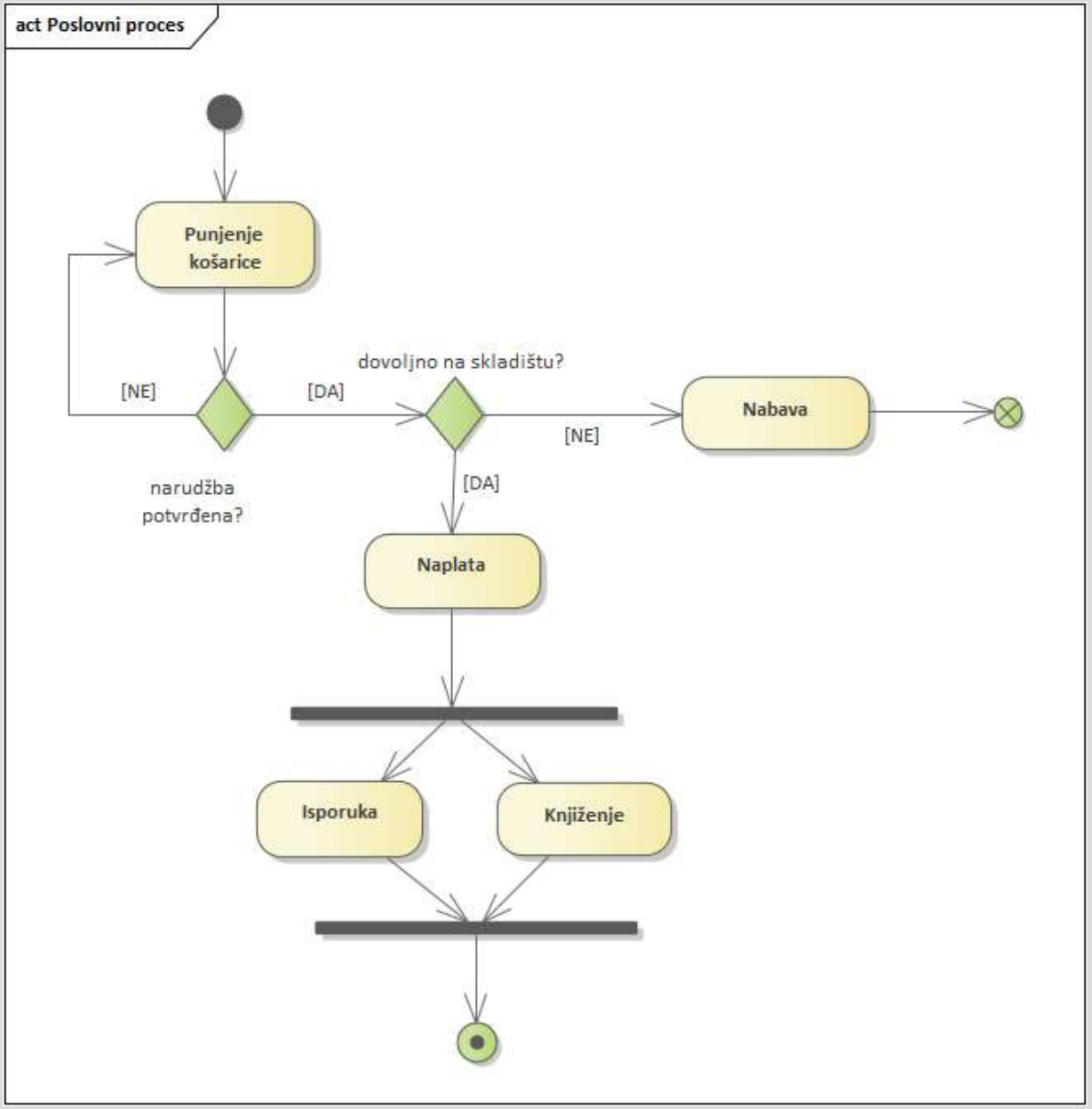
Arhitekturni kontekst sustava prodaje auto-dijelova



Gledište poslovnog procesa

- Kako se sustav koristi u širem poslovnom procesu ➔ procesni model
- UML dijagram aktivnosti se koristi za prikaz poslovnog procesa

Poslovni kontekst prodaje auto-dijelova



MODELI INTERAKCIJE

Modeli interakcije

- Modeliranje interakcije korisnika pomaže u definiranju korisničkih zahtjeva.
- Modeliranje interakcije između (dijelova) sustava naglašava probleme komunikacije koji se mogu pojaviti.
- Modeliranje interakcije između komponenti pomaže u otkrivanju hoće li predviđena struktura sustava postići željene performanse i pouzdanost.
- UML dijagrami obrazaca uporabe i dijagrami slijeda koriste se za modeliranje interakcija

Modeliranje obrazaca uporabe

- Obrasci uporabe
  - razvijeni kao potpora izlučivanju zahtjeva
- Svaki obrazac uporabe:
  - Predstavlja zasebni zadatak koji uključuje vanjsku interakciju sa sustavom
- Aktori:
  - ljudi
  - drugi sustavi
- Prikaz:
  - dijagramima
  - tekstualno

Dijagrami slijeda

- Modeliranje interakcije:
  - između aktora i objekata sustava
- Slijed interakcije koji se događa:
  - unutar obrazaca uporabe
- Objekti i aktori:
  - na vrhu dijagrama sa svojim životnim linijama
- Interakcije:
  - prikazane označenim strelicama

STRUKTURNI MODELI

Strukturni modeli

- Prikazuju organizaciju sustava u pogledu komponenti sustava i njihovih odnosa
- Mogu biti:
  - Statički – prikaz strukture oblikovanja sustava
  - Dinamički – prikaz organizacije sustavu tijekom rada
- Pri raspravi i oblikovanju arhitekture sustava

Dijagrami razreda

- Korišteni kod razvoja objektno-usmjerenog modela:
  - prikazuju razrede i njihove odnose
- Razred objekta:
  - predstavlja opću definiciju neke vrste objekta u sustavu
- Veze između razreda:
  - Pokazuju odnose između razreda
- U ranim stupnjevima razvoja:
  - Objekti predstavljaju neki entitet iz stvarnog svijeta

PONAŠAJNI MODELI

Ponašajni modeli

- Modeli dinamike ponašanja sustava pri radu:
  - prikazuju što se događa ili bi se trebalo događati prilikom odgovaranja sustava na poticaje iz okoline
- Dvije vrste poticaja:
  - podaci koji dolaze na obradu
  - događaji koji potiču ili okidaju procese u sustavu (mogu sadržavati i podatke)

Modeliranje upravljano podacima

- Mnogi poslovni procesi:
  - Upravljani podacima koji ulaze u sustav
  - S malo obrade vanjskih događaja
- Modeli upravljani podacima:
  - Pokazuju slijed akcija:
    - pri obradi ulaznih podataka
    - Pri generiranju izlaznih podataka
- Korisni pri analizi zahtjeva:
  - Prikaz cijelog ciklusa obrade unutar sustava

Modeliranje upravljano događajima

- Sustavi za rad u stvarnom vremenu:
  - Vođeni događajima
  - Malo obrade podataka
- Modeli upravljani događajima:
  - pokazuju kako sustav reagira na vanjske i unutarnje događaje ili poticaje
- Pretpostavka postojanja ograničenog broja stanja sustava
  - prijelazi između stanja poticani događajima

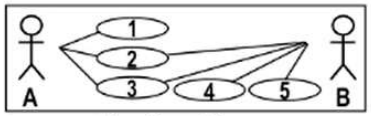
Modeli stroja stanja

- Prikazi stanja kao čvorova i događaja kao lukova između čvorova
- Dijagrami stanja: integralni dio UML standarda

„SEMANTIČKI MODEL”

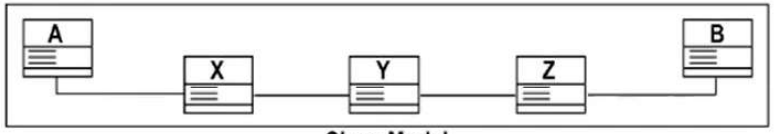
Semantički model

Semantic Model Content



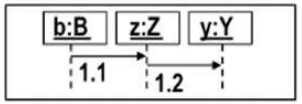
Use Case Diagram

Process: high level

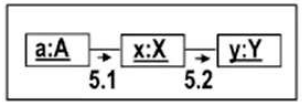


Class Model

Policy

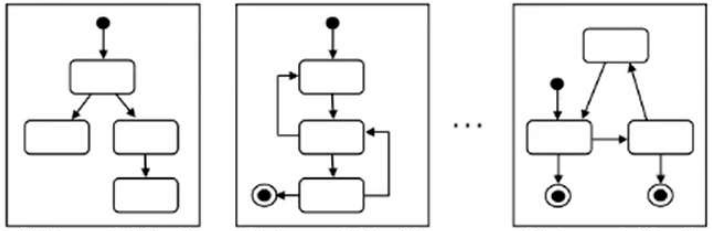


Sequence Diagram for 1



Communication Diagram for 5

Process: intermediate level



State model for X | State model for Y | State model for Z

Process: detailed level

Izvor: How to Engineer Software, Part 2: A Deeper Dive into Semantic Models | Steve Tockey – YouTube

MODELNO-USMJERENO INŽENJERSTVO

Modelno-usmjereno inženjerstvo

- engl. *model-driven engineering* (MDE)
- rezultat: model, a ne program
- izvršni programi se automatski generiraju iz modela
- podizanje razine apstrakcije programskog inženjerstva iznad razine:
  - detalja programskog jezika
  - detalja ciljne implementacijske platforme

Primjena modelno-usmjerenog inženjerstva

- u ranoj fazi razvoja
- nije potpuno jasno koliko će još imati značajnog utjecaja na prakse programskog inženjerstva

Primjena modelno-usmjerenog inženjerstva

- prednosti
  - Pogled s viših razina apstrakcije
  - Generiranja koda bi značilo bržu prilagodbu sustava novim platformama.
- mane
  - Model apstrahiranja problem nisu nužno pravi modeli za implementaciju
  - Dodatni trošak razvoja prevodilaca za različite platforme

Modelno-usmjerena arhitektura

- engl. *model-driven architecture* (MDA)
  - preteča šireg pojma modelno usmjerenog inženjerstva
- za opis sustava koristi podskup UML modela
- modeli se stvaraju na različitim razinama apstrakcije
- platformski neovisni model ~> (bez ručne intervencije) funkcionalan program ?

Vrste modela (I)

- Računalno neovisan model
  - engl. *computation independent model* (CIM)
  - modeliranje domenskih apstrakcija → *domenski model*
- Platformski neovisan model
  - engl. *platform independent model* (PIM)
  - ne dotiče se implementacije
    - statička struktura sustava
    - ponašajni opis, kako reagira na vanjske i unutarnje događaje/poticaje

Vrste modela (II)

- Platformsko specifični model
  - engl. *platform specific model* (PSM)
  - transformacija platformsko neovisnog modela u zasebne platformsko specifične modele za svaku platformu
  - Mogu postojati i slojevi koji postepeno uključuju platformsko specifične detalje

Agilne metode i Modelno-usmjerena arhitektura

- iscrpno unaprijedno modeliranja je u proturječju s temeljnim idejama agilnog manifesta
- kad bi se postigla potpuna (100%-tna) automatizacija transformacija i cijeli programski kôd generirao iz platformsko neovisnog modela MDA bi se mogao koristiti u agilnom procesu jer ne bi bilo potrebno posebno kodiranje

Usvojenost modelno-usmjerene arhitekture

- ograničenja:
  - potrebna je potpora specijaliziranih alata za transformacije modela
  - potrebna je prilagodba postojećih alata (kojih nema puno) različitim okolinama
  - za dugoročne sustave razvijene modelno-usmjerenim pristupom potrebno je razvijati svoje alate

Usvojenost modelno-usmjerene arhitekture

- koristi:
  - modeli olakšavaju rasprave o načinu oblikovanja programske potpore, ali te apstrakcije nisu one prave za implementacijske probleme
  - za većinu složenih sustava implementacija je puno manji problem od inženjerstva zahtjeva, sigurnosti, pouzdanosti, integracije s postojećim sustavima i procesa ispitivanja

Usvojenost modelno-usmjerene arhitekture

- stanje:
  - opravdanost pristupa neovisnosti o platformi vrijedi samo za velike dugoročne sustave
  - za manje proizvode i informacijske sustave ušteda uporabe modelno-usmjerene arhitekture se poništava troškovima njezinog uvođenja i podržavanja
  - široka prihvaćenost agilnih metoda je bacila modelno-usmjereni pristup u drugi plan

Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
    - interakcija
    - strukture
    - ponašanja

Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
      - prikazuju kako se sustav pozicionira unutar radne okoline i procesa
    - interakcija
    - strukture
    - ponašanja

Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
    - interakcija
      - koriste se dijagrami obrazaca uporabe i dijagrami slijeda
      - dijagrami obrazaca uporabe načelno prikazuju interakcije aktora sa sustavom
      - dijagrami slijeda prikazuju detalja interakcije
    - strukture
    - ponašanja

Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
    - interakcija
    - strukture
      - koriste se dijagrami razreda prikazuju organizaciju i strukturu sustava
    - ponašanja

Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
    - …
    - ponašanja
      - dinamika kod izvršavanja sustava može se prikazati s gledišta obrade podataka ili događaja/poticaja
      - dijagrami aktivnost → modeliranje obrade podataka
      - dijagrami stanja → modeliranje ponašanja temeljenog na unutarnjim ili vanjskim događajima

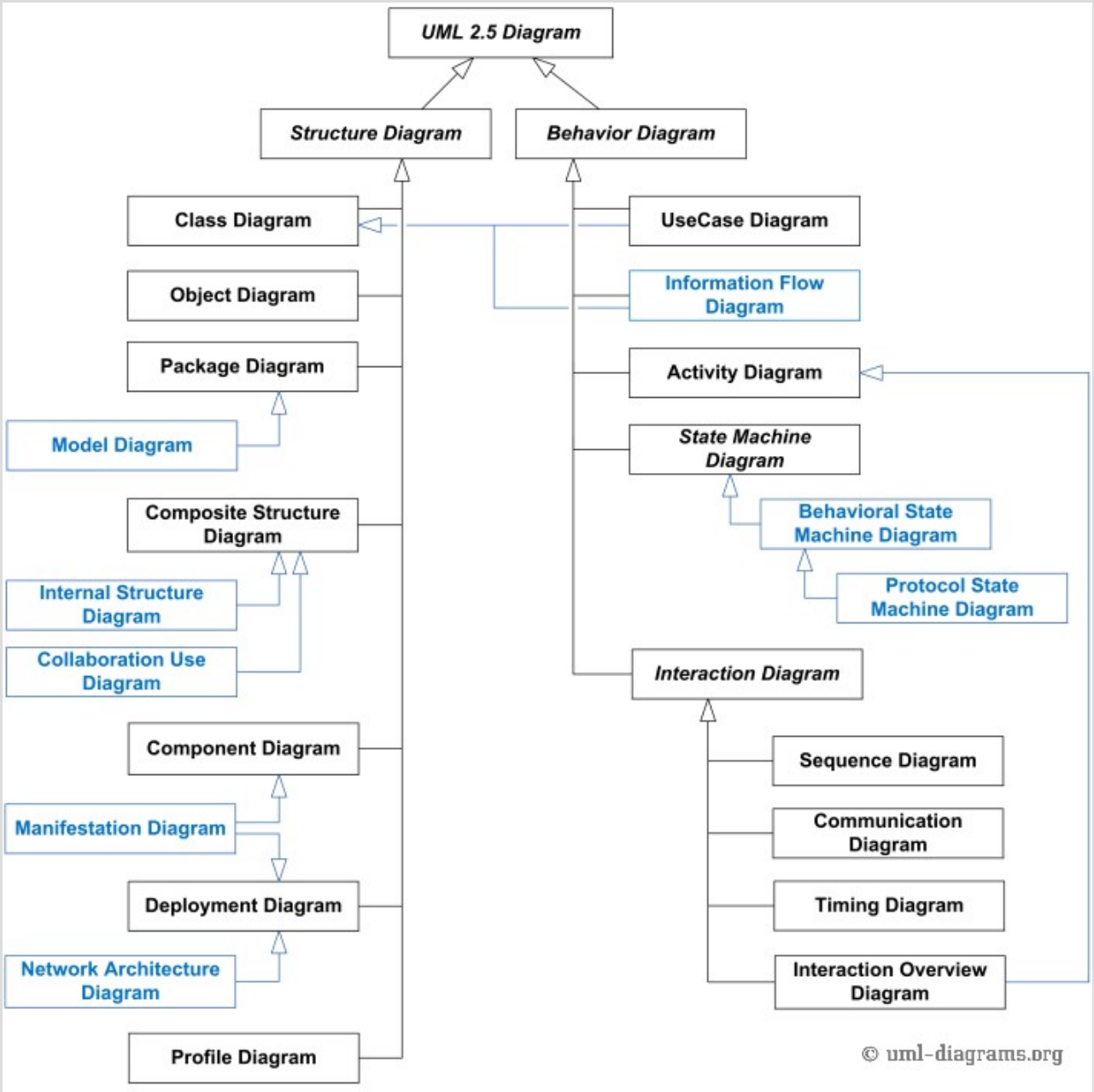
Zaključno

- model ≡ apstraktan pogled koje zanemaruje detalje
  - međusobno komplementarni modeli se razvijaju za prikaz:
    - konteksta
      - prikazuju kako se sustav pozicionira unutar radne okoline i procesa
    - interakcija
      - koriste se dijagrami obrazaca uporabe i dijagrami slijeda
      - dijagrami obrazaca uporabe načelno prikazuju interakcije aktora sa sustavom
      - dijagrami slijeda prikazuju detalja interakcije
    - strukture
      - koriste se dijagrami razreda prikazuju organizaciju i strukturu sustava
    - ponašanja
      - dinamika kod izvršavanja sustava može se prikazati s gledišta obrade podataka ili događaja/poticaja
      - dijagrami aktivnost se mogu koristiti za modeliranje obrade podataka
      - dijagrami stanja se koriste za modeliranje ponašanja temeljenog na unutarnjim ili vanjskim događajima

Zaključno

- Modelno-usmjereno inženjerstvo je pristup razvoju programske potpore gdje se sustav prikazuju u obliku skupa modela koji bi se automatski trebali pretvarati u izvršni kôd

Vrste dijagrama



Enterprise Architect

- https://sparxsystems.com/

Enterprise Architect

Firstly... What is UML?

The Object Management Group (OMG) specification states:

> *"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."*

The important point to note here is that UML is a 'language' for specifying and not a method or procedure. The UML is used to define a software system; to detail the artifacts in the system, to document and construct - it is the language that the blueprint is written in. The UML may be used in a variety of ways to support a software development methodology (such as the Rational Unified Process) - but in itself it does not specify that methodology or process.

UML defines the notation and semantics for the following domains:

- The User Interaction or Use Case Model - describes the boundary and interaction between the system and users. Corresponds in some respects to a requirements model.
- The Interaction or Communication Model - describes how objects in the system will interact with each other to get work done.
- The State or Dynamic Model - State charts describe the states or conditions that classes assume over time. Activity graphs describe the workflows the system will implement.
- The Logical or Class Model - describes the classes and objects that will make up the system.
- The Physical Component Model - describes the software (and sometimes hardware components) that make up the system.
- The Physical Deployment Model - describes the physical architecture and the deployment of components on that hardware architecture.

https://sparxsystems.com/resources/tutorials/uml/part1.html

The UML also defines extension mechanisms for extending the UML to meet specialized needs (for example Business Process Modeling extensions).

Part 2 of this tutorial expands on how you use the UML to define and build actual systems.

REFERENCE I LITERATURA

- A. Jović, N. Frid, D. Ivošević: Procesi programskog inženjerstva, 3. izdanje, Sveučilište u Zagrebu, FER ZEMRIS, 2019.
- I. Sommerville, Software engineering, 10th ed., Pearson, 2016., https://iansommerville.com/software-engineering-book/
- S. Tockey: How to Engineer Software, Wiley-IEEE Computer Society Press, 2019.

Hvala na pažnji!

Pitanja?