1. Razvoj programske potpore

* Svojstva programa:
  + Veličina
  + Funkcionalnost
  + Složenost
* Problemi razvoja programske potpore
  + Suštinski – nezaobilazni
  + Neočekivani – nenamjerni
  + Svojstva programa
    - Složenost (Complexity)
    - Usklađenost (Conformity)
    - Promjenljivost (Changeability)
    - Nevidljivost (Invisibility)
* Karakteristike programske potpore:
  + Neopipljiva
  + Jednostavna za reproducirati
  + Industrija izrazito radno intenzivna
  + Lagano je unijeti izmjene
  + Programska potpora se ne troši
* Uspješnost projekata – atributi:
  + OnTime
  + OnBudget
  + OnTarget
  + OnGoal
  + Value
  + Satisfaction
* Problem složenosti
  + Osnovna složenost
    - Ovisi o problemu
    - Ne može se eliminirati tehnologijom i tehnikama
  + Nenamjerna
    - Uvedena uporabom alata i tehnika
  + Problem razvoja programske potpore je prekomjerna nenamjerna složenost
* Uloga programskog inžinjera
  + Nije dovoljna samo vještina programiranja
    - Mora razumjeti zahtjeve i pisati specifikacije
      * Izgrađuje modele
    - Izvrsno poznaje programiranje
    - Upotrebljava znanstvene metode i interpretaciju pri analizi i rješavanju inžinjerskih problema
    - Član tima
      * Vještine komunikacije
      * Vještine upravljanja
* Tradicijsko oblikovanje programske potpore
  + Formulacija zahtjeva (Requirements)
  + Specifikacija i analiza
  + Oblikovanje
  + Kodiranje
  + Ispitivanje modula
  + Integracija i ispitivanje sustava
* Nedostaci postupka oblikovanja sustava
  + Analiza i oblikovanje
    - Zahtjevi nisu formalizirani
      * Nemoguće postići preciznost
    - Izgradnja ad hoc prototipa
    - Procjena performansi
    - Dijeljenje na sklopovski i programski dio
    - Dokumentacija se teško interpretira
* Analiza postojećeg stanja
  + Moderan postupak oblikovanja sustava
    - Propisani strukturirani postupci
    - Dokumentiranje
    - Modeliranje (apstrakcija)
    - Višestruka uporaba komponenata
    - Formalna verifikacija modela i ispitivanje
  + Oblikovanje zasnovano na modelima (model based design)
  + Oblikovanje zasnovano na komponentama (component based design)

Oblikovanje programske potpore

* Principi savladavanja problema složenosti
  + Dekompozicija (devide and conquer)
  + Inkrementalno poboljšanje
  + Ponovno korištenje dijelova
  + Odvojiti podprobleme
  + Apstrakcija (eliminacija nepotrebnih detalja)
    - Uvođenje modela
  + Uvođenje formalizma (matematički određena semantika)
* Metodologije oblikovanja zasnovane na modelima
  + Model
    - Inžinjerski model – sažeta reprezentacija sustava koja naglašava značajna svojstva iz nekog pogleda na sustav
    - Ne prikazuje se odjednom
    - Oblik prezentacije jednostavno razumljiv u području primjene
  + Uporaba modela
    - Razumijevanje složenih sustava
      * Specifikacije zahtjeva
      * Projektiranje
      * Rano otkrivanje pogrešaka
      * Olakšano komuniciranje
    - Osnova za implementaciju
  + Svojstva modela
    - Apstrakcija
    - Razumljivost
    - Točnost
    - Predvidljivost (odgovor na pitanja o promatranom sustavu)
    - Jednostavnost
  + Osobito svojstvo programa
    - Razvoj apstraktnih modela u potpunu implementaciju bez promjene inžinjerske okoline, metoda ili alata
    - Moguće generiranje apstraktnih modela izravno automatizirano iz implementacije
      * Model = sustav
  + Posljedice
    - Programska potpora manje ovisna o okolini
    - Individualne mogućnosti dolaze do izražaja
    - Brz razvojni ciklus
* Oblikovanje programske potpore zasnovano na modelima
  + Model u centru razvojnog procesa
  + Cilj
    - Povećanje razine apstrakcije
    - Poboljšanje komunikacije
    - Produktivnost
    - Održavanje
* Poboljšanja razvoja programske potpore
  + Uvesti propisane postupke (procedure)
    - Precizno definiranje faza oblikovanja
  + Dokumentiranje faza
  + Analiza i izbor stila arhitekture
  + Obikovanje programske potpore u komponentama
  + Uvesti formalne metode provjere (verifikacije)

Programsko inžinjerstvo

* Sistematski i organiziran pristup procesu izrade
* Upotrebljavanje prikladnih alata i tehnika ovisno o problemu i ograničenjima te postojećim resursima
* Izazovi programskog inžinjerstva
  + Heterogenost
    - Razvoj programske potpore za različite platforme
  + Vrijeme isporuke
  + Povjerenje
  + Promjena zahtjeva
  + Složenost
* Etika
  + Načela
    - Javni interes
    - Klijent i poslodavac
    - Proizvod
    - Prosudba
    - Upravljanje
    - Struka
    - Kolegijalnost
    - Odnos prema sebi

1. Životni cikljus programske potpore

* Cilj je poboljšanje procesa oblikovanaj proizvoda, upravljanja proizvodom i upravljanja projektima
* Model životnog ciklusa programske potpore
  + Cjelokupni proces planiranja, analize, izrade, ispitivanja, puštanja u rad i evolucije programske potpore
  + Opisuje kako nastaju faze razvoja pp
  + Sastoji se od faza
  + Opisuje odnose glavnih točaka PROCESA i rezultata
  + Uloga procesa
    - Osigurava potrebne informacije u trenutku kada su potrebne, u upotrebljivom obliku
    - Proces definira TKO radi ŠTO, KADA i kako postići željeni cilj
* Proces programskog inžinjerstva
  + Skup aktivnosti čiji je cilj razvoj ili evolucijka programskog proizvoda
  + Generičke aktivnosti
    - Specifikacija
      * Analiza zahtjeva – određivanje što sustav treba raditi i koja su ograničenja
    - Oblikovanje i implementacija
      * Izbor arhitekture i ostvarenje programskog sustava
    - Validacija i verifikacija
      * Provjera radi li sustav ono što se od njega zahtjeva
    - Evolucija
      * Promjene sukladno novim zahtjevima
  + Idealan razvoj
    - Zahtjev > analiza/specifikacija > oblikovanje > implementacija > održavanje > umirovljenje
* Upravljanje projektima
  + Efikasno PI fokusira se na
    - Ljude
    - Projekt
    - Proizvod
    - Proces
    - Alate
    - Diagram

      Description automatically generated
* ISO/IEC/IEEE
  + Dogovaranje
  + Organizacija projekta
  + Upravljanje
  + Tehnički pocesi

Dionici na projektu i timski razvoj

* Dionik – svatko tko sudjeluje na projektu ili na koga projekt utječe
  + Primarni dionici
    - Projekt na njih utječe izravno
      * Razvojni tim
  + Sekundarni dionici
    - Projekt na njih utječe sporedno
      * Klijenti, prodavači
  + Ključni dionici
    - Donose odluke na projektu ili odlučuju o tome je li projekt uspio ili nije
      * Mogu biti primarni ili sekundarni
* Dionici na projektu unutar tvrtke
  + Podjela ovisi o
    - Korištenom modelu životnog ciklusa
    - Vrsti i ciljevima projekta
    - Veličini i organizaciji tvrtke
  + Primjeri organizacije
    - Rukovoditelj projekta
    - Poslovni analitičar
    - Vlasnik proizvoda
    - Razvojni tim
    - Inžinjer osiguranja kvalitete
* Značajke razvojnog tima
  + Veličina tima (4-7)
  + Kompetencije članova tima
  + Vrsta organizacije tima
  + Podjela uloga unutar tima
* Stupnjevi razvoja programskog tima
  + Tipično 5 stupnjeva razvoja
    - Formiranje
    - Uspostava komunikacije
    - Definiranje uloga
    - Ostvarivanje ciljeva
    - Raspuštanje
* Programska potpora timskom razvoju
  + Sustavi za komunikaciju između članova
  + Sustavi za upravljanje verzijama programske potpore
* Inžinjerstvo zahtjeva
  + Postupak pronalaženja, analiziranja, strukturiranja, dokumentiranja i provjere korisnički zahtjevanih usluga sustava te ograničenja u uporabi
  + Zahtjevi
    - Opisuju što programski sustav treba raditi kao i ograničenje u njegovom radu
* Klasifikacija zahtjeva
  + Poslovni zahtjevi
    - Visoka razina apstrakcije
  + Korisnički zahtjevi
    - Specifikacija visoke razine apstrakcije
    - Prirodnim jezikom, tablicama, dijagramima
  + Zahtjevi sustava
    - Vrlo detaljna specifikacija o funkcijama sustava
    - Strukturirani prirodni jezik, specifični jezici oblikovanja, grafička notacija, matematička specifikacija
* Klasifikacija zahtjeva s obzirom na sadržaj
  + Funkcionalni zahtjevi
    - Opisuju ponašanje sustava
    - Definiraju usluge koje sustava osigurava
  + Nefunkcionalni zahtjevi
    - Opisuju ostale zahtjevane atribute sustava
    - Moraju biti mjerljivi
    - Postavljaju ograničenja na sustav ili proces oblikovanja
  + Zahtjevi domene primjene
    - Specifični funkcionalni ili nefunkcionalni zahtjevi za domenu koja se promatra
* Kriteriji zahtjeva
  + Potreba
  + Provjerljivost
  + Ostvarivost
* Svojstva zahtjeva
  + Zahtjev mora davati odgovor na pitanje ŠTO, a ne KAKO
  + Potrebna svojstva
    - Korektnost
    - Provjerljivost
    - Jednoznačnost
    - Kompletnost
    - Konzistentnost
    - Promjenjivost
* Klasifikacija nefunkcionalnih zahtjeva
  + Zahtjevi programskog proizvoda
    - Defninirani način ponašanja proizvoda
  + Organizacijski zahtjevi
    - Rezultat organizacijskih pravila i procedura
  + Vanjski zahtjevi
  + Nefunkcionalni zahtjevi moraju biti mjerljivi
* Zahtjevi domene primjene
  + Mogu biti novi funkcionalni zahtjevi ili ograničenja na postojeće zahtjeve
  + Problemi
    - Razumljivost
    - Implicitnost
* Zahtjevi sustava
  + Detaljnija specifikacija funkcije sustava, njegovih usluga i ograničenja nego zahtjevi korisnika
  + Mogu se prikazati nekim od modela sustava
  + Odnos zahtjeva i oblikovanja
    - Zahtjev = što sustav radi
    - Oblikovanje = kako sustav radi
* Izražavanje zahtjeva sustava
  + Strukturirani prirodni jezik
  + Jezik za opis oblikovanja (SDL)
  + Grafička notacija (UML)
  + Matematička specifikacija (FSM, teorija skupova)
* Procesi inžinjerstva zahtjeva
  + Proces predstavlja strukturiran skup aktivnosti koji vodi nekom cilju
  + Proces inžinjerstva zahtjeva je skup aktivnosti koje generiraju i dokumentiraju zahtjeve
* Generičke aktivnosti inžinjerstva zahtjeva
  + Procesi se razlikuju ovisno o domeni primjene, ljudskim resursima i organizaciji koja oblikuje zahtjeve
    - Nema jedinstvenog procesa IZ
  + 2 uobičajena modela procesa zahtjeva
    - Klasični
    - Spiralni
  + Generičke aktivnosti
    - Studija izvedivosti
    - Izlučivanje i analiza zahtjeva
    - Specifikacija zahtjeva
    - Validacija zahtjeva
    - Upravljanje zahtjevima
* Klasični model
  + Diagram

    Description automatically generated
* Spiralni model
  + Trostupanjska aktivnost
    - Specifikacija
    - Validacija
    - Izlučivanje zahtjeva
  + ITERACIJE
    - U svakoj iteraciji različit intenzitet aktivnosti
      * Rane iteracije – fokus na razumijevanje poslovnog modela
      * Kasnije – modeliranje detalja sustava
  + Zahtjevi se u pojedinim iteracijama specificiraju s različitom razinom detalja
  + Diagram, radar chart

    Description automatically generated
* Studija izvedivosti
  + Određuje isplati li se predloženi sustav
  + Kratka fokusirana studija koja provjerava
    - Doprinose sustava
    - Mogućnosti ostvarenja
    - Mogućnosti integracije predloženog sustava u postojeće
  + Temelji se na određivanju koje informacije su potrebne za studiju, prikupljanje informacija i pisanju izvješća
* Izlučivanje i analiza zahtjeva
  + Najznačajnija aktivnost
  + Otkrivanje zahtjeva
    - Razrješava domenu primjene
    - Definira uskule koje sustav treba pružiti
    - Određuje ograničenja u radu sustava
  + Može uključivati različite dionike
  + Koraci
    - Prikupljanje informacija
    - Organiziranje prikupljenih zahtjeva
    - Provjera zahtjeva
  + Rezultat = navedeni zahtjevi
  + Rezultati procesa izlučivanja
  + Diagram

    Description automatically generated
* Spiralni model izlučivanja i analize zahtjeva
  + 4 osnovne aktivnosti
    - Izlučivanje / otkrivanje zahtjeva
    - Klasifikacija i organizacija zahtjeva
    - Ustanovljavanje prioriteta i pregovaranje
    - Dokumentiranje zahtjeva
* Pogledi
  + Način strukturiranja zahtjeva
  + Višeprespektivna analiza – omogućava razrješavanje konflikata
  + Tipovi pogleda
    - Pogledi interakcije
      * Ljudi i drugi sustavi koji komuniciraju sa sustavom
    - Indirektni pogledi
      * Dionici koji ne koriste sustav izravno
    - Pogledi domene primjene
      * Karakteristike odmene i ograničenja koje utječu na zahtjeve
* Metode izlučivanja zahtjeva
  + Intervjuiranje
  + Scenarij
  + Obrasci uporabe
  + Specificiranje dinamičkih interakcija u sustavu (sekvencijski dijagrami)
  + Etnografija – promatranje rada
  + Izrada prototipa
* Intervjuiranje
  + Dionici se ispituju o sustavu koji trenutačno koriste te o novopredloženom sustavu
  + Tipovi
    - Zatvoreni intervju
      * Skup ranije definiranih pitanja
    - Otvoreni intervju
      * Nema definiranih pitanja
  + U praksi intervju često ne daju dobre rezultate za zahtjeve domene primjene
* Scenarij
  + Primjeri iz stvarnog života o načinu korištenja sustava
  + Sadržaj
    - Opis početne situacije
    - Opis standardnog tijeka događaja
    - Opis što se može dogoditi krivo
    - Informacije o paralelnim aktivnostima
    - Opis gdje scenarij završava
  + Dionici diskutiraju i kritiziraju scenarij
* Obrasci uporabe
  + Opisuju se aktori u interakciji s uslugama sustava
  + Opisuje sve moguće interakcije sustava
  + Temeljni elementi
    - Obrasci uporabe
    - Aktori
    - Odnosi
  + Modeliranje obrascima uporabe
    - Model obrazaca uporabe je pogled koji ističe ponašanje sustava kako ga vide vanjski korisnici
* Specificiranje dinamičkih interakcija
  + Metoda izlučivanja zahtjeva modeliranjem dinamičkih interakcija u sustavu
  + Modeliranje ponašanja sustava
  + Osnovni tipovi dijagrama interakcija
    - Sekvencijski
    - Komunikacijski
* Interakcije
  + Prikazuje komunikaciju elemenata sustava
  + Omogućava specificiranje međudjelovanja između elemenata sustava
  + Olakšava identifikaciju sučelja
  + Utvrđuje se potreba za raspodjelom sustava
* Društveni i organizacijski čiimbenici
  + Utječu na sve poglede
  + Ne postoje jedinstveni modeli za opis obavljanja nekog posla
  + Korisna znanja iz društvenih znanosti
* Etnografija
  + Tehnika opažanja koja podrazumijeva dolazak jednog ili više ljudi iz razvojnog tima u tvrtku gdje će se sustav primjenjivati i uključivanje tih inžinjera u svakodnevne aktivnosti kako bi ustanovili kako ljudi stvarno rade
  + Diagram

    Description automatically generated
* Analiza zahtjeva
  + Provjere zahtjeva po kriterijima
    - Neophodnost zahtjeva
    - Konzistentnost i kompletnost
    - Mogućnost ostvarenja
  + Cilj je utvrđivanje problema, nekompletnosti i nejednoznačnosti u izlučenim zahtjevima
* Profiliranje dionika
  + Potrebno odrediti važnost dionika u pojeidnim dijelovima sustava
    - Velika
    - Srednja
    - Mala
  + Odrediti vještine i sposobnosti
  + Dokumentirati ponašanje i stavove
  + Olakšava razumijevanje sadašnjeg i željenog stanja
  + Olakšava grupiranje i postavljanje prioriteta zahtjeva
* Konflikti i utvrđivanje prioriteta
  + Zahtjevi visokog prioriteta
    - Obvezno pri preuzimanju
  + Srednji prioritet
    - U drugim iteracijama
  + Nizak prioritet
    - Prikaz mogućnosti budućeg razvoja
* Validacija zahtjeva
  + Cilj pokazati da zahtjevi predstavljaju prihvatljiv opis sustava koji naručitelj doista želi
  + Tehnike validacije
    - Recenzija zahtjeva
    - Izrada prototipa
    - Generiranje ispitnih slučaja
  + Rezultati validacije
    - Lista problema
    - Lista utvrđenih akcija za razrješavanje problema
* Elementi provjere zahtjeva
  + Razumljivost
  + Kompletnost
  + Konzistencija
  + Valjanost
  + Ostvarivost
  + Provjerljivost
  + Sljedivost
  + Adaptabilsnost
* Upravljanje promjenama u zahtjevima
  + Upravljanje ili rukovanje zahtjevima
    - Planiranje i upravljanje promjenama u zahtjevima
  + Klasifikacija
    - Okolinom promjenjivi zahtjevi
    - Novonastali zahtjevi
    - Posljedični zahtjevi
    - Zahtjevi kompatibilnosti
* Upravljanje procesom promjena
  + Planiranje promjena
  + Proces koji slijedi kada se utvrdi potreba za promjenom
  + Sljedivost
  + Izbor CASE alata

1. Primjena UML-a u inžinjerstvu zahtjeva

* Motivacija
  + Izražavanje zahtjeva prirodnim jezikom
  + Izrada točne specifikacije
  + Potreban strukturiran i normirani način zapisa
* UML
  + Objedinjeni jezik modeliranja
    - Pravila usmjerena na konceptualnu i fizičku reprezentaciju sustava
    - Omogućava
      * Vizualizaciju
      * Specifikaciju
      * Oblikovanje
      * Dokumentiranje

artefakata programske potpore

* + Osnovni elementi
    - Stvari
      * Strukturne stvari
        + Razred, sučelje, obrasci uporabe
      * Stvari ponašanja
        + Interakcija, stanje
      * Stvari grupiranja
        + Organizacija elemenata u grupe, paketi
      * Stvari označavanja
        + Opisni elementi, fomalni/neformalni opis
    - Relacije
      * Asocijacija, generalizacija, realizacija, ovisnost, uključivanje, proširenje
    - Dijagrami
      * Dijagram razreda, objekata, obrazaca uporabe, toka, suradnji, stanja, aktivnosti, komponenata
  + Namjena UML-a
    - Vizualni prikaz pojedinog aspekta sustava
    - Pogled na model
      * Djelomična reprezentacija sustava
* Odnos sustava, modela i pogleda
  + Sustav
    - skupina međusobno povezanih elemenata koji djeluju prema skupu pravila tvoreći jedinstvenu cjelinu
      * struktura, ponašanje, međusobne veze
  + Model
    - Informativna reprezentacija sustava
  + Pogled
    - Prikazuje odabrane aspekte modela
    - Notacija
  + Pogledi i modeli jednog sustava mogu se preklapati
* UML – višedijagramski jezik
  + Svaki dijagram = pogled u model
  + Primjer pogleda u UML-u
    - Logički pogled
    - Razvojni pogled
    - Implementacijski pogled
    - Procesni pogled
  + Diagram

    Description automatically generated
* UML dijagrami
  + Obrasci uporabe
  + Sekvencijski
  + Komunikacijski
  + Dijagram stanja
  + Dijagram aktivnosti
  + Dijagram komponenti
  + Dijagram paketa
  + Dijagram pogleda interakcije
  + Vremenski dijagram
  + Dijagram profila
  + Dijagram klasa
  + Dijagram objekata
  + Dijagram složene strukture
* 3 kategorije dijagrama
  + Strukturni
    - koje stvari moraju biti modelirane
    - razreda, objektni, paketa, profila...
  + ponašajni
    - što se događa u sustavu
    - obrasci uporabe, stanja, aktivnosti
  + interakcije
    - upravljački i podatkovni tijek između stvari modeliranog sustava
    - sekvencijski, komunikacije, interakcije, vremenski
* Dijagram obrazaca uporabe
  + Obrasci uporabe
    - Modeliraju funkcionalne zahtjeve sustava
      * Opisuje tipi;nu uporabu sustava značajnu korisniku (aktoru)
      * Pogled na vanjsko ponašanje sustava prema korisniku (pogled interakcije)
    - Specificiraju sustave neovisno o načinu implementacije
  + Dijagram obrazaca uporabe
    - Opisuje obrasce uporabe, aktore i njihove odnose
    - Jednostavan grafički prikaz
      * Granica sustava
      * Tko koristi sustav
      * Kako koristi sustav
  + Oblikovanje dijagrama obrazaca uporabe
    - Odrediti aktore
    - Definirati obrasce uporabe
    - UML dijagramom prikazati povezanost aktora s obrascima uoprabe te međusobni odnos obrazaca uporabe
    - Prikazati granice sustava i imenovati sustav
  + Elementi dijagrama
    - Obrazac uporabe
      * Funkcionalnost
        + Slijed akcija koje sustav ili drugi entitet obavlja u interakciji s aktorima sustava
      * Predstavljaju funkcionalnosti sustava i odgovornosti
    - Aktor
      * Vanjski subjek koji komunicira sa sustavom
      * Skup uloga koje imaju korisnici u interakciji sa sustavom
      * Jedan korisnik – više uloga
    - Relacije
      * Između aktora i obrazaca uporabe
    - Granica sustava
      * Granica između fizikalnog sustava i aktora koji su u interakciji s njime
  + Elementi obrasca uporabe
    - Redni broj
    - Namjena
    - Naziv
    - Opis
    - Glavni aktor
    - Preduvjeti
    - Pokretač
    - Opis osnovnog tijeka
    - Opis mogućih odstupanja
  + Aktori
    - Uloga jednog ili više korisnika
    - Primarni
      * Generiraju podatke ili primaju informacije
        + Aktivni – inicijatori

Smjer strelice od aktora prema obrascu

Klijent, administrator

* + - * + Pasivni – sudionici

Smjer strelice od obrasca prema aktoru

Baza podataka, pisač

* + - Sekundarni
      * Pružaju podršku sustavu
    - Obrasci uporabe se pišu iz perspektive aktora
    - SUSTAV NIJE AKTOR
    - Diagram

      Description automatically generated
  + Numeracija
    - Redni broj ispred obrasca uporabe
    - Višerazinska brojčana oznaka za include i extend (2.1.1)
    - Jednoznačno
  + Relacije
    - Pridruživanje / asocijacija
      * Komunikacija aktora i obrasca
      * Shape, rectangle

        Description automatically generated
    - Obuhvaćanje / nužno sadrži
      * Include
      * Osnovni obrazac sadrži ponašanje definirano u obuhvaćenom obrascu
      * Dijeljeno ponašanje više obrazaca
      * A picture containing shape

        Description automatically generated
      * Od osnovnog obrasca prema include obrascu
    - Proširenje
      * Extend
      * Proširenje osnovnog obrasca
      * A picture containing text

        Description automatically generated
      * Od extend obrasca prema osnovnom
    - Poopćenje
      * Generalizacija
      * Odnos između općeg i specifičnog
      * Može i kod aktora i kod obrazca
      * A picture containing text, furniture, table, dining table

        Description automatically generated
  + Include
    - Funkcionalna dekompozicija
      * Da bi se nešto kupilo nužno je prijaviti se u sustav
      * Kupovina ---- include---- > prijava u sustav
    - Ponovna uporaba
      * Isplata i uplata novca nužno uključuju autentifikaciju korisnika
  + Extend
    - Dopunsko ponašanje koje se može i ne mora ostvariti
      * Prilikom pregleda košarice ona se može, ali ne mora uređivati
  + Generalizacija
    - Različita implementacija neke funkcionalnosti
      * Plaćanje se može obaviti karticom ili gotovinom
    - Između aktora (nasljeđivanje među klasama)
      * Student može biti preddiplomski ili diplomski
* Višestrukost u obrascima uporabe
  + Na svakom kraju spojnice
    - Diagram

      Description automatically generated
    - Klijent može imati najviše 1 isplatu odjednom
    - Banka može imati po volji broj transakcija
* Sekvencijski dijagram
  + Modeliranje ponašanja
    - Interakcija – komunikacija između objekata
    - Definiranje načina komunikacije
    - Identifikacija sučelja
    - Raspodjela zahtjeva
  + Detaljniji razvoj i prikaz scenarija
  + Pokazuje objekte kao životne crte prema dnu stranice
  + Vremensko uređenje
  + Prikazivanje i identificiranje komunikacije među objektima
  + Prikaz paralelnosti u sustavu
  + Diagram

    Description automatically generated



* + Entitet sustava
    - Sudionik, objekt u komunikaciji
    - Izmjenjuje poruke s ostalim objektima
    - Instanca objekta popraćena razredom
      * Objekt:Razred
  + Životna linija i područje aktivnosti
    - Životna linija
      * Period postojanja objekta
      * Može biti stvorena ili prekinuta unutar dijagrama
        + Create – stvaranje
        + X – uništavanje (destroy)
    - Područje aktivnosti
      * Pravokutnik na životnoj liniji
      * Objekt je instanciran i aktivan
      * Obavlja komunikaciju
  + Poruke
    - Označavaju komunikaciju između objekata
      * Strelica od izvorišta (pošiljatelj) prema odredištu (primatelj)
      * Obvezan message\_label
      * Može biti poslana s desna na lijevo, lijeva na desno ili samom sebi
    - Tipovi poruke
      * Sinkrona
      * Asinkrona
      * Povratna
      * Rekurzivna
      * Poruka stvaranja sudionika
      * Poruka uništavanja sudionika
      * Izgubljene i nađene poruke
    - Sinkrona poruka
      * Pošiljatelj čeka završetak obrade na strani primatelja i povratnu informaciju
      * Odgovor može biti prikazan (crtkana strelica) ili ne mora biti prikazan (poruka je kompletna s implicitnom povratnom porukom)
      * 
    - Asinkrona poruka
      * Pošiljatelj ne čeka završetak obrade poruke
      * Nastavlja odmah daljnji tijek izvođenja
      * 
    - Rekurzije
      * Poruka samom sebi
        + Rekurzivni poziv procedre
        + Poziv druge metode istog objekta
        + Diagram, schematic

          Description automatically generated
    - Poruka stvaranja
      * Create
      * Objekt spušten u odnosu na druge
      * Ne postoji prije trenutka pozivanja stvaranja
    - Poruka uništavanja
      * Destroy
      * Objekt prestaje postojati
      * Diagram

        Description automatically generated
  + Izgubljenje poruke
    - Poslane, ali nisu stigle do odredišta
    - Poruke poslane primateljima koji nisu vidljivi u trenutnom dijagramu
  + Nađene poruke
    - Dolaze od nepoznatog pošiljatelja
    - Dolaze od pošiljatelja koji nije vidljiv u trenutnom dijagramu
    - Diagram

      Description automatically generated
* Ograničenja trajanja
  + Za jednu poruku
    - Označava se nagibom poruke
    - Diagram

      Description automatically generated
  + Za dvije poruke
    - A picture containing diagram

      Description automatically generated
* Označavanje grananja i petlji
  + Okvir
    - Područje dijela slijeda
  + Oznake područja
    - If = opt [uvjet]
    - If / else = alt [uvjet]
    - Petlja = loop [(min[, max])], [uvjet]
  + Alt [uvjet]
    - Izbor između 2 ili više slijeda poruka
      * If then else
      * Crtkana linija razdvaja alternative (operande)
      * Uvjet se navodi za svaki operand
    - Diagram

      Description automatically generated
  + Opt [uvjet]
    - Slijed poruka koji se događa samo pri ispunjenom uvjetu
      * If then
      * Nema operanda
    - Diagram

      Description automatically generated
  + Loop [uvjet]
    - Slijed poruka koji se ponavlja
    - Diagram

      Description automatically generated
  + Paralelne aktivnosti – par
    - Diagram, schematic

      Description automatically generated
* Invarijante stanja
  + Ograničenja na živosnoj liniji na kojoj se nalazi
    - Poruka koja prethodi ograničenju smatra se nevaljanom ako ograničenje nije zadovoljno
* Povezivanje sekvencijskih dijagrama
  + Referencom na dijagram
  + Otvorenom strelicom i komentarom

1. Modeli procesa programskog inžinjerstva

* Proces programskog inžinjerstva
  + Strukturirani skup aktivnosti koji čini okvir neophodan za izvođenje sustavnog plana razvoja i oblikovanja programske potpore
  + Generičke aktivnosti
    - Specifikacija
    - Oblikovanje i implementacija
    - Validacija i verifikacija
    - Evolucija
  + Apstraktna reprezentacija procesa
* Specifikacija
  + Proces određivanja potrebnih usluga i ograničenja u radu i razvoju sustava
    - Određuje se procesom inžinjerstva zahtjeva
  + Generičke aktivnosti procesa inžinjerstva zahtjeva
    - Studija izvedivosti
    - Izlučivanje i analiza zahtjeva
    - Specifikacija zahtjeva
    - Validacija zahtjeva
    - Rukovanje promjenama u zahtjevima
* Oblikovanje i implementacija
  + Proces preslikavanja specifikacije u stvarni sustav
  + Oblikovanje programske potpore
    - Realizira specifikaciju
  + Implementacija
    - Izvršni program
* Izbor i oblikovanje strukture
  + Oblikovanje arhitekture
  + Arhitektura dokumentirana skupom modela
  + Apstraktna specifikacija
  + Oblikovanje sučelja
* Implementacija programskog proizvoda
  + Preslikavanje dokumentiranog oblikovanja u program i otklanjanje pogrešaka
  + Proces otklanjanja pogrešaka
    - Diagram

      Description automatically generated
* Validacija i verifikacija programskog proizvoda
  + Validacija
    - Zadovoljava li sustav funkcijske zahtjeve
  + Verifikacija
    - Zadovoljava li sustav zahtjeve na ispravan način
* Evolucija programskog proizvoda
  + Ovisi o promjeni zahtjeva na sustav
  + Lehmanovi zakoni evolucije
* Pregled modela procesa PI
  + Ad hoc model
  + Vodopadni model
  + Prototipni model
  + Evolucijski model
  + Inkrementalni model
  + Spiralni model
  + Unificirani model
  + Ubrzani (agilni) razvoj
* Ad hoc ili oportunistički model
  + Izrada programske potpore započinje se bez strukturiranja skupa aktivnosti na projektu
  + Nema razrade zahtjeva ili oblikovanja prije implementacije
  + Programska potpora se neprestano mijenja
  + Samo za male projekte
    - Brže dolazi do propadanja programske potpore
    - Nije moguće ocjeniti napredak
* Vodopadni model
  + Diagram

    Description automatically generated
  + Procesne faze
    - Analiza zahtjeva
    - Oblikovanje sustava
    - Implementacija i ispitivanje modula
    - Integracija i ispitivanje sustava
    - Rad sustava i održavanje
  + !!!! pojednina faza se mora završiti prije pokretanja nove faze !!!!
  + Pretpostavke vodopadnog modela
    - Ranije poznati zahtjevi
    - Zahtjevi se rijetko mijenjaju
    - Korisnik zna što treba
    - Nije pogodno za složene projekte
  + Problemi vodopadnog modela
    - Teško ugrađivanje promjena nakon pokretanja procesa
    - Nefleksibilna podjela projekta = teže unošenje promjena
    - Zahtjevi moraju biti dobro razumljivi i promjene svedene na minimum !!!
    - Samo za projekte u kojima su zahtjevi dobro definirani
* Evolucijski model
  + Uključuje iterativni i inkrementalni pristup razvoju
  + Ciklus aktivnosti ponavlja se za svaku novu verziju programske potpore
  + Oblici
    - Metoda odbacivanja prototipa
    - Istraživački razvoj i oblikovanje
  + Metoda odbacivanaj prototipa
    - Cilj = razumijevanje zahtjeva sustava pomoću prototipa
    - Započinje s grubo definiranim zahtjevima – kasnije se razrješavaju detaljnije
    - Prototip se ne koristi za završni projekt
  + Istraživački razvoj i oblikovanje
    - Cilj = kontinuiran rad s kupcem te na temelju inicijalne specifikacije izgraditi robustan prototip i stalno ga unaprjeđivati
  + Prototip
    - Radni model
    - \*Prototipni model
      * Naglasak na razvoj prototipova rano u razvojnom procesu
      * Omogućuje ranu povratnu informaciju i analizu
    - Prototip nije sustav koji radi
    - Shape

      Description automatically generated
  + Problemi evolucijskog modela
    - Proces razvoja i oblikovanja nije jasno vidljiv
    - Često loše strukturirani sustavi
    - Potreban brz razvoj prototipa
  + Primjena
    - U malim i srednje interaktivnim sustavima
    - U dijelovima velikih sustava
    - Za sustave s kratkim vijekom trajanja
* Iteracije u modelima
  + Zahtjevi za sustav uvijek evoluiraju i prate razvoj projekta
  + 2 pristupa iteracijama
    - Inkrementalni pristup
    - Spiralni razvoj i oblikovanje
* Inkrementalni pristup
  + Sustav se ne isporučuje u cjelini
  + Zahtjevi se svrstaju u prioritetne cjeline
    - Viši prioritet = ranija isporuka
  + Početkom razvoja inkrementa zahtjevi se fiksiraju
  + Inkrement prolazi nekoliko iteracije prije nego što bude spreman za isporuku
  + Prednosti inkrementalnog razvoja
    - Kupac dobiva vrijednost sa svakim inkrementom
    - Funkcionalnost u ranim fazama projekta
    - Iz ranih inkremenata se izlučuju zahtjevi za kasnije
    - Manji rizik od neuspjeha
    - Detaljnije ispitivanje
    - Smanjuje rizik neuspjeha u odnosu na vodopadni model
* Spiralni model
  + Proces prikazan spiralom umjesto slijedom
  + Svaka petlja predstavlja fazu
    - nema fiksnih faza
  + Rizici se eksplicitno određuju i razrješuju
    - velik naglasak na analizu rizika
  + kombinira se inkrementalni razvoj sa sustavnim
  + Diagram

    Description automatically generated
  + Sektori u spiralnom modelu
    - Slijede vodopadni model
    - Poszavljanje ciljeva
    - Procjena i smanjivanje rizika
    - Razvoj i validacija
    - Planiranje
  + Karakteristike spiralnog modela
    - Prednosti
      * Održava iterativnu prirodu razvoja
      * Obuhvaća šrednosti vodopadnog modela i brze izrade prototipa
      * Smanjuje rizik razvoja
      * Preglednost projekta
    - Nedostaci
      * Složen proces, puno administracije
      * Potrebno poznavanje tehničke analize rizika
* Unificirani model
  + Izveden na temelju jezika za modeliranje UML i pridruženih aktivnosti
  + Svojstva
    - Priznaje utjecaj korisnika
    - Sugerira evolucijski pristup
    - Podržava OO
    - Prilagodljiv
  + 3 perspektive
    - Dinamička
      * Slijed faza kroz vrijeme
    - Statička
      * Aktivnosti procesa
    - Praktična
      * Aktivnosti kroz iskustvo i dobru praksu
  + RUP (Rational Unified Process)
    - Suvremeni model procesa
      * Više iteracija na pojedinim inkrementima
      * Temeljene na obrascima uporabe
      * U fokusu je arhitektura sustava
    - Predstavlja okosnicu procesa za razvoj i prilagodbu programske potpore
    - Specifična implementracija UP-a
  + Struktura unificiranog procesa (UP)
    - Elementi
      * Proces
      * Disciplina
      * Uloga
        + Ponašanje i odgovornosti
      * Aktivnost
      * Artefakt
        + Rezultat procesa
      * Tijek rada
    - Naglašava kontinuiranu komunikaciju s klijentima u svim fazama
    - Korisnici predlažu izmjene na kraju svake iteracije
  + Pregled procesa razvoja
    - Faze
      * Početak
      * Razrada
      * Izgradnja
      * Prijenos
    - Organizacija procesa u vremenu
      * 2 dimenzije
        + Horizontalna

Dinamika – faze, iteracije, ključne točke

* + - * + Vertikalna

Statika – aktivnosti, discipline, uloge, artefakti

* + - * Ključne točke
        + Definiraju pridružene dokumente ili aktivnosti
    - Table

      Description automatically generated with medium confidence
  + Faze unificiranog procesa
    - Početna faza
      * Definira doseg projekta i razvoj modela poslovnog procesa
      * Visoka razina apstrakcije
    - Faza razrade
      * Razumijevanje domene problema i razvoj arhitekture sustava
      * Plan projekta, specifikacija značajki, temelj arhitekture sustava, utvrđivanje rizika
    - Faza izgradnje
      * Razrada preostalih zahtjeva
    - Faza prijenosa
      * Instalacija proizvoda korisnicima
  + UP se zasniva na obrascima uporabe
    - Koriste se kroz sve faze
    - Poketači iteracija
      * Kreiranje i validacija arhitekture
      * Definicija ispitnih slučajeva, scenarija i procedura
      * Planiranje iteracija
      * Izrada dokumenacije
      * Razmještaj sustava
    - Sinkroniziraju sadržaj različitih modela
  + UP arhitektura sustava
    - Struktura ili strukture sustava koja sadrži elemente, njihova vidljiva obilježja i odnose između njih
  + Praksa za UP
    - Iterativni razvoj
    - Upravljanje zahtjevima
    - Komponentno zasnovana arhitektura
    - Vizualno modeliranje UML
    - Verifikacija kvalitete
    - Upravljanje promjenama
  + Značajke UP
    - Iterativno + inkrementalno oblikovanje
    - U središtu obrasci uporabe
    - Definira i povezuje faze i aktivnosti procesa
    - Odgovorajući modeli za opis pojedinih aktivnosti
    - Modeli dokumentirani dijagramima
    - Dijagrami definirani UML normom
    - Arhitektura sadrži skup pogleda u modele
* Agilni razvoj
  + Rješenje problema tradicionalnih procesa
  + Ubrzani, žustri razvoj
  + Grupa metoda za razvoj pp kojima je zajednički iterativni razvoj uz male inkremente te koje podržavaju brzi odziv na korisničke zahtjeve
  + Za male i srednje projekte
  + Načela
    - Otvorenost prema promjenama
    - Često isporučivanje inačica
    - Kontinuirana suradnja
    - Održivi razvoj
    - Jednostavnost
  + Primjeri agilnog razvoja
    - Ekstremno programiranje
      * Vrsta inkrementalnog postupka razvoja i isporuke
      * Pristup baziran na razvoju, oblikovanju i isporuci vrlo malih funkcionalnih inkremenata
      * Kontinuirano poboljšanje koda
      * Programiranje u paru
      * 12 pravila dobrog ekstremnog programiranja
        + Planiranje rada
        + Mala izdanja
        + Jednostavni projekti
        + Automatsko testiranje
        + Kontinuirana integracija
        + Prepravljanje postojeće inačice
        + Programiranje u paru
        + Kolektivno vlasništvo
        + 40-satni tjedan
        + Kupac u timu
        + Standardi kodiranja
        + Terminologija
      * Prednosti
        + Za srednje velike projekte
        + Brzi odgovor na zahtjeve korisnika
        + Iterativnost
        + Kvaliteta se osigurava stalnim ispitivanjem
      * Nedostaci
        + Loše za velike projekte
        + Potrebno iskustvo
        + Veća cijena
    - SCRUM
      * Radni okvir za projekte strukturiran da podrži razvoj složenih proizvoda
      * Empirizam
      * Iterativni, inkrementalni pristup za optimizaciju i kontrolu rizika
      * Timovi Scruma, uloge, aktivnosti, događaji, artefakti, pravila
      * Agilni pristup
        + Početak = lista zadataka
        + Prvo najprioritetniji zadaci
        + Rad u kratkim vremenskim intervalima = SPRINT
        + Nedovršeni rad mijenja prioritete za nadolazeće iteracije
      * Elementi
        + Scrum tim

Vlasnik proizvoda

Razvojni tim

Vođa

* + - * + Događaji u Scrumu

Sprint

Sastanak planiranja sprinta

Dnevni scrum

Razvoj sprinta

Retrospektiva sprinta

* + - * + Artefakti scruma

Projektni dnevnik zaostataka

Dnevnik sprinta

Inkrement

* + - * Scrum tim
        + 5-9 članova
        + Izmjena članova samo između sprintova
        + 3 uloge

Vlasnik proizvoda

Odgovoran za dnevnik zadataka

Određuje prioritete zadataka

Razvojni tim

Razvija i isporučuje inkrement

Vođa scruma

Odgovoran za odvijanje procesa

Brine o timu i suradnji

* + - * + Dionici

Konstruktivne informacije timu

* + - * Sprint
        + Vremenski ograničeni ciklus rada

Rad mora dati mjerljivi rezultat

* + - * + Kontejner za događaje

2-4 tjedna

Elementi

Sastanak planiranja sprinta

Sprint kao posao razvoja s dnevnim scrumom

Revizija sprinta

Retrospektiva sprinta

* + - * Artefakti scruma
        + Predstavljaju zadatke ili vrijednosti neophodne za uspjeh tima Scruma u isporuci inkrementa
        + Projektni dnevnik zaostataka

Lista zahtjeva na proizvod

Vlasnik odgovoran za sadržaj

* + - * + Dnevnik sprinta

Skup stavaka odabranih za sprint

* + - * + Inkrement

Zbroj svih stavki završenih tijekom sprinta

Svaka stavka mora imai kriterij prihvatljivosti

Mora biti završen na kraju sprinta

* + - Disciplinirana agilna isporuka
      * Prilagodljiv za velike programske projekte
      * Značajke
        + Ljudi prije svega
        + Hibridni razvoj
        + Puni ciklus isporuke
        + Ovisnost o kontekstu
        + Iskoristivo programsko rješenje umjesto funkcionalne programske potpore
        + Samoorganizacija timova, uz odgovarajuće upravljanje
    - Kanban
      * Glavne prakse
        + Vizualizacija rada i procesa

Radni tok (WIP) – Kanban ploča

Radne stavke predstavljene karticama

Ekspicitno definirane procedure

* + - * + Ograničavanje količine rada u tijeku

Načelo povlačenja

* + - * + Upravljanje tokom procesa

Identifikacija uskih grla procesa

* + - * + Jasna pravila
        + Implementacija povratne veze
        + Zajednička poboljšanja
        + A picture containing diagram

          Description automatically generated
    - Kontinuirana integracija i isporuka
      * Kontinuirana integracija
        + Zahtjeva integraciju koda u zajedničko spremište više puta dnevno
      * Kontinuirana isporuka
        + Osigurava da se programsko rješenje može pouzdano objaviti u bilo kojem trenutku
      * CI/CD CICD
      * Kontinuirana implementacija
        + Promjene se autorizirano implementiraju
    - Virtualizacija
      * Virtualni stroj
        + Docker

1. Arhitektura programa

* Uloga arhitekture programa
  + Apstrakcija sustava na visokoj razini
  + Nositelj kvalitete sustava
  + Strukturira razvojni projekt
  + Osnovica za komunikaciju dionika
* Definicija arhitekture
  + Struktura(e) sustava koja sadrži elemente programa, njihova izvana vidljiva obilježja i odnose između njih
  + Proširuje se odlukama vezanim uz odabit arhitekture
  + Arhitektura = elementi, struktura, razlozi
* Komponente arhitekture
  + Elementi programa
    - Struktura
    - Ponašanje
    - Podaci
    - Procedure
  + Odnosi
    - Unutarnji
    - Vanjski
  + Principi
    - Pravila oblikovanja
    - Pravila arhitekture
* Arhitektura programa
  + Opis arhitekture je skup dokumentiranih pogleda raznih dionika
  + Pojedini pogled dokumentiran dijagramom
    - Dijelovi sustavam topologija, vanjska obilježja
* Oblikovanje arhitekture programske potpore
  + Proces identificiranja i strukturiranja podsustava koji čine cjelinu te okruženja za upravljanje i komunikaciju između podsustava
  + Prednosti
    - Smanjuje cijenu oblikovanja
    - Jasno objašnjenje strukture pp
    - Poboljšava kvalitetu proizvoda i dokumentacije
      * Poboljšava razumljivost
    - Mogućnosti za ponovnu uporabu
    - Pomaže u određivanju prioriteta ciljeva
* Uloga arhitekta
  + Razumije potrebe poslovnog modela
  + Svjestan različitih tehničkih pristupa
  + Vrednuje dobre i loše strane pristupa
  + Preslikava vrednovane zahtjeve u tehnički opis
* Konceptualni model arhitekture
  + Konceptualni okvir
  + Uspostavlja pojmove i koncepte
  + Osnova za evoluciju znanjau području s malo zajedničke terminologije
  + AD
    - Skup dokumenata arhitekture
    - Arhitekturni opis
  + Dionik
  + Pitanja
* Koraci oblikovanja arhitekture elemenata
  + 1. ustanoviti arhitekturu sustava najviše razine
  + 2. ocjeniti arhitekturu prema kriterijima
  + 3. pretvoriti zahtjeve u arhitekturu
  + 4. oblikovanje sučelja na najvišoj razini
  + 5. oblikovanje dokumenata najviše razine baze podataka
  + 6. dokumentirati preliminarnu korisničku dokumentaciju
  + 7. dokumentirati zahtjeve preliminarnog ispitivanja
  + 8. procijeniti arhitekturu prema kriterijima
  + 9. provesti zajednički pregled
* Pogledi na arhitekturu pp
  + Klasifikacija modela po dosegu
    - Koncepcijska
      * Pažnja na pogodnu dekompoziciju sustava
      * UML arhitekturni dijagrami, CRC kartice
    - Logička
      * Precizno dopunjena koncepcijska arh
      * Detaljan nacrt pogodan za razvoj komponenti
    - Izvršna
      * Namijenjena raspodijeljenim i paralelnim sustavima
      * Pridruživanje aktivnosti fizičkom sustavu
* Arhitektonski pogledi
  + Pogled
    - Prikaz cijelog sustava iz perspektive jednog skupa povezanih problema
  + Prikaz jednog ili više strukturnih aspekata arh koji ilustriraju kako arh rješava jedan ili više problema koje imaju jedan ili više njezinih dionika
  + Svaki pogled = nove informacije
  + Pogledi su modularni
    - Mogu sadržavati jedan ili više arh modela
  + Dosljednost između pogleda
* Klasifikacija pogleda
  + Statičan strukturni pogled
    - Pogled modula
    - Pokazuju kompoziciju/dekompoziciju sustava
  + Dinamički procesni pogled
    - Ponašajni
    - Komponente i konektori
      * Najstariji opis
      * Opisuje ponašanje sustava tijekom izvođenja
        + Komponente

Osnovna jedinica izračunavanja i pohrane podataka

* + - * + Konektori

Interakcija između komponenata

Socket, middleware

* + - * Svodi se na oblikovanje kompozicije komponenata i konektora
      * Danas nije dostatan – potrebni višestruki pogledi
    - Runtime
  + Pogled alokacije
    - Odnos programske potpore i razvojne okoline
* Višestruki pogledi na arhitekturu
  + Logički pogled
    - Funkcije sustava
  + Pogled dekompozicije modula
    - Hijerarhijska dekompozicija funkcionalnosti sustava
  + Pogled komunikacijskih procesa
    - Dretve, sinkronizacija, tokovi podataka
  + Pogled implementacije
    - Kako se program pridružuje sklopovlju
* 4+1 pogled arh
  + Logički pogled
    - Funkcionalnosti prema pogledu korisnika
  + Procesni pogled
    - Opis sustava kroz aktivnosti
  + Fizički pogled
    - Topologija sustava na fizičkoj razini
  + Razvojni pogled
    - implementacija
  + Scenarij
    - Podskup obrazaca uporabe korišten u izgradnji arhitekture
  + Diagram

    Description automatically generated
* Arhitekturni stilovi
  + Arhitekturni okvir
    - Skup interesa, dionika, gledišta, pogleda i pravila
    - Npr 4+1 pogled
  + Stilovi arh pp
    - Familija sustava definirana sličnim oblicima strukturne organizacije
    - Stil – ima jasno definiran rječnik komponenti i konektora te pripadajućih topoloških ograničenja s kojima se stil opisuje
      * Najviša razina apstrakcije
      * Zasnovan na arhitekturnim obrascima (koncept, temeljna struktura za neki stil)
    - Razlikovanje stilova = temeljem kolekcije komponenata i načina njihove interakcije
    - Semantički model – specificira kako odrediti globalna svojstva
    - Populani primjeri stilova
      * Apstrakcija podataka i objektno usmjereni stil
      * Protok podataka
      * Rapozitorij podataka
      * Arhitektura zasnovana na događajima
* Apstrakcija podataka i objektno usmjereni stil
  + Objektno usmjerena paradigma
  + Temelji se na zatvaranju reprezentacije podataka i operacija u ADT
  + ADT – abstract data types
  + Instance ADT-a su objekti = komponente novog stila
  + Sustav = skup objekata u interakciji
  + Izračunavanja se odvijaju u kontekstu objekta
* Arhitektura zasnovana na događajima
  + Komponente nisu eksplicitno povezane
  + Neke komponente generiraju signale = događaji
  + Prijavljivanje komponenti na strukturu za povezivanje komponenata
  + Događaji se javno objavljuju te se pozivaju registrirane procedure
  + MVC
    - Diagram

      Description automatically generated
* Arhitektura MVC
  + Odvaja sučelje od ostatka sustava
  + Razinska kohezija
  + Smanjuje međuovisnost
  + Model sadrži razrede čiji objekti se obrađuju
  + Pogled sadrži razrede čiji objekti služe za prikaz podataka
  + Controller sadrži razrede koji upravljaju korisničkom interakcijom
* Proces izbora i vrednovanja arhitekture
  + Diagram

    Description automatically generated
  + Proces donošenja odluka
    - Odluke oblikovanja
  + Prostor oblikovanja
    - Više inačica rješenja problema
    - Skup opcija
      * Debeli klijent
      * Tanki klijent
      * Ostali podproblemi
    - Skup svih mogućih rješenja oblikovanja zasnovanih na različitom izboru
  + Donošenje odluka u oblikovanju
    - Zadaće
      * Postavljanje prioriteta sustava
      * Dekompozicija sustava
      * Definiranje svojstava
      * Postavljanje sustava u kontekst
      * Cjelovitost sustava
  + Klasifikacija odluka u oblikovanju
    - Strukturne odluke
      * Statički model
      * Stvaranje podsustava, razina, komponenata
    - Ponašajne odluke
      * Dinamički model
      * Formiranje interakcija u sustavu
    - Odluke o svojstvima
      * Vodilje
      * Ograničenja
    - Izvršne odluke
  + Text

    Description automatically generated
  + Arhitekturno značajni zahtjevi
    - Daleko dublji utjecaj na arhitekturu
    - Ocijenjeni od dionika s obzirom na prioritete i doseg
      * Uobičajena 3 stupnja
  + Prioriteti i ciljvi u donošenju odluka
    - Tipovi vrednovanja arhitekture
      * Tehnički
        + Procjena prema atributima kvalitete sustava (izvedba, sigurnost...)

Metoda analize arhitekturnih kompromisa (tradeoff analysis)

* + - * Ekonomski
        + Metoda analize troškova i dobiti (cost-benefit analysis)
* Jezici za analizu i oblikovanje arhitekture
  + AADL
    - Modeliranje programskih i sklopovsnih arhitektura
    - Komponente, veze, implementacije
    - Provjera elemenata i svojstva
  + Analiza arhitekture zasnovana na modelu
    - Modele opisuju jezici ADL
      * Jezik za opis i analizu programske potpore u pojmovima njihovih arhitekturnih elemenata i odnosa između njih
      * Fokus samo na specifične sustave
  + EA
    - Arhitektura informacijskih sustava velikih organizacija
      * Proširenje koncepata UML-a
      * Poslovna, primjenska i tehnološka razina
  + Oblikovanje arhitekture
    - Sinteza
    - Ciljevi
      * Smanjenje cijene i povećanje profita
      * Ubrzanje razvoja i implementacije
      * Poboljšanje kvalitete
    - Graphical user interface, text, application, chat or text message

      Description automatically generated
    - Oblikovanje od vrha prema dolje
    - Oblikovanje od dna prema vrhu
    - Hibridno oblikovanje
* Principi oblikovanja arhitekture programske potpore
  + Podijeli pa vladaj
  + Povećaj koheziju
  + Smanji međuovisnost
  + Zadrži razinu apstrakcije
  + Povećaj ponovnu uporabivost
  + Povećaj uporabu postojećeg
  + Oblikuj za fleksibilnost
  + Planiraj zastaru
  + Oblikuj za prenostivost
  + Oblikuj za ispitivanje
  + Oblikuj konzervativno
  + Oblikuj po ugovoru
* SOLID
  + Fokus na objektno usmjerene sustave
  + Cilj
    - Razumljivije oblikovanje, olakšano održavanje i nadogradnja
  + Princip jedne odgovornosti
  + Princip otvoren-zatvoren
  + Princip Liskovine supstitucije
  + Princpi segregacije sučelja
  + Princip inverzne ovisnosti
* S – princip jedne odgovornosti
  + Svaki razred treba implementirati samo jednu odgovornost
    - Ima samo jedno izvorištew i razlog za promjenu specifikacije
* O – princip otvoren-zatvoren
  + Svaki programski entitet mora biti otvoren za proširenja, ali zatvoren za izmjene
  + Proširenje u podrazredima ne smije zahtjevati promjene u nadrazredima
* L – princip Leskovine supstitucije
  + Svaki objekt u programu treba moći biti zamijenjiv s primjercima njegovih podtipova bez potrebe za izmjenama u programu
  + Podrazredi proširuju ponašanje nadrazreda, ali ga ne smiju mijenjati
* I – princip segregacije sučelja
  + Klijent koji implementira sučelje ne smije ovisiti o metodama koje ne koristi
  + Bolje više specifičnih sučelja, nego jedno veliko
* D – princip inverzije ovisnosti
  + Rad komponenti više razine ne smije ovisiti o implementacijskim detaljima komponenti niže razine
  + Apstrakcije ne smiju ovisiti o konkretnim detaljima niže razine
* Dokumentiranje arhitekture
  + Potrebno zbog rane analize sustava i mogućnosti ponovne uporabe
  + Nositelj obilježja kvalitete
  + Dokumentacija ne zastarijeva
  + Omogućuje vrednovanje oblikovanja i poboljšanje
  + Sredstvo komunikacije
  + Pisati s gledišta čitatelja
  + Minimalna dokumentacija
    - Referentna specifikacija
      * potpuni skup dokumentiranih pokretača arhitekture
      * matrica odluka
    - Pregled za upravu
      * pregled visokog nivoa
    - dokumentacija komponenti
      * pogled na razini sustava za komponente
      * specifikacija komponente, sučelja te dijagrami suradnje
  + Text

    Description automatically generated
  + Svojstva dokumentacije
    - Dobra
      * Doseže potrebe i ciljeve ključnih dionika
    - Ispravna
      * Tehnički ispravna i jasna
    - Uspješna
      * Upotrebljava se u stvarnom razvoju sustava

1. Modeliranje OO arhitekture UML dijagramom razreda

* UML dijagram razreda
  + Strukturni dijagram koji pokazuje razrede u OO sustavu, njihove atribute i metode te veze između razreda koji međusobno komuniciraju ili se nasljeđuju
  + Osnovni elementi
    - Razred
      * Atributi
      * Operacije
    - Relacije
      * Pridruživanje
      * Generalizacija
      * Ovisnost
      * Realizacija
  + Razred
    - Opisuje skup objekata sa zajedničkim obilježjima, ograničenjima i ponašanjem
    - UML reprezentacija
      * Naziv
      * Atributi
      * Operacije
      * vrijednosti atributa određuju stanje objekta, metode mijenjaju stanje objekta mijenjajući vrijednosti atributa
      * A picture containing diagram

        Description automatically generated
    - Detalji
      * Različita razina u različitim fazama
        + Konceptualni
        + Specifikacijski
        + Implementacijski
      * Diagram

        Description automatically generated
  + Atributi
    - Vidljivost
      * Public +
      * Private –
      * Protected #
      * Package ~
    - Naziv
    - Brojnost
    - Vrsta
    - Početna vrijednost
    - Modifikacije atributa
  + Odgovornost, operacija, metoda
    - Odgovornost
      * Ono što sustav mora izvršiti
      * Svaki funkcionalni zahtjev mora se pridijeliti nekom razredu
      * Jedan razred = jedna odgovornost (u načelu)
    - Operacije
      * Ostvaruju odgovornosti pojedinog razreda i implementiraju se metodama
        + Vidljivost

Private, public, package, protected

* + - * + Modifikatori

Static, abstract

* + - * + Parametri
        + Povratna vrijednost
      * Jedna odgovornost kroz više operacija
    - Hijerarhija – metoda -> operacija -> odgovornost
  + Odnosi među razredima
    - Pridruživanje
      * Asocijacija
      * Agregacija
        + Kompozicija
      * Zadaje se smjer i brojnost
      * Diagram

        Description automatically generated
      * 



* + - Generalizacija
      * Nasljeđivanje
      * Diagram

        Description automatically generated with medium confidence
    - Realizacija
      * Implementacija sučelja
      * A picture containing text

        Description automatically generated
    - Ovisnost
      * Shape

        Description automatically generated with low confidence
  + Pridruživanje
    - Veza između instanciranih objekata
    - Prikazuje se na krajevima relacije pridruživanja
    - Svako pridruživanje može se imenovati
    - Naziv uloge = opisuje prirodu veze
      * Text

        Description automatically generated
    - smjer pridruživanja
      * jednosmjerna veza
        + 
      * Dvosmjerna veza
        + 
      * Nenaznečena veza je dvosmjerna ili nespecificirana
    - Višestrukost pridruživanja
      * 1 = točno jedan objekt
      * n = točno određen broj objekata (2,4,10)
      * n1...n2 = između n1 i n2 objekata
      * n1...\* = između n1 i više
      * 0...\* ili \* = neograničeno
  + Refleksivno pridruživanje
    - Pridruživanje se spaja na isti razred
      * Predmet ima druge predmete kao preduvjet, jedan nalog može sadržavati više dodatnih naloga...
      * Diagram

        Description automatically generated
  + Agregacija
    - Pridruživanje koje pokazuje da jedan razred sadrži druge razrede
      * Odnos cjelina – dio
      * Cijeli dio = agregat
    - Agregacija se koristi kada vrijedi
      * Dijelovi su dio agregata / agregat je sastavljan od dijelova
      * Kada je netko ili nesto vlasnik agregata, vlasnik je i njegovih dijelova
      * Text

        Description automatically generated with medium confidence
    - Brisanjem agregata njegovi dijelovi mogu ostati samostalni
  + Kompozicija
    - Jaka agregacija
    - Kod uništavanja objekta uništavaju se i pripadnici agregata
    - Dijelovi ne mogu biti dijeljeni s drugim kompozicijama
    - Graphical user interface, text, application

      Description automatically generated
      * Objekt se inicijalizira unutar razreda
  + Ovisnost
    - Najopćenitiji odnos između razreda
    - Označava utjecaj jednog objekta na drugi
    - 
    - Uvijek usmjereno
    - Bez brojnosti
    - Stereotipovi
      * Use
      * Call
      * Create
      * Modifies
    - Slabije od relacije pridruživanja
    - Diagram

      Description automatically generated
  + Generalizacija
    - Usmjerena veza od izvedenog razreda (podrazred) prema općenitijem razredu (nadrazred)
      * Izveredni razred proširuje nadrazred
      * Instanciranjem podrazreda instancira se i nadrazred
        + Veza „IS A“ (podskup)

Jabuka je Voće

* + Liskovin princip zamjene
    - LSP
    - Ako postoji varijabla čiji tip je nadrazred, program se mora korektno izvoditi ako se u varijabli pohrani instanca tog nadrazreda ili instanca bilo kojeg nadrazreda
      * Izvedeni razredi proširuju ponašanje nadrazreda, ne smiju ih mijenjati
* Sučelje
  + Opisuje dio vidljivog ponašanja skupa objekata
    - Specificira usluge nekog razreda
    - Podvrsta apstraktnog razreda
    - Plug and play
  + Razred koji realizira sučelje = implements
    - Mora implementrirati sve operacije yadane sučeljem
    - Paziti na princip segregacije
      * Ne stavljati metode vezane uz nesrodna ponašanja u sučelje
* Nasljeđivanje vs realizacija
  + Diagram

    Description automatically generatedDiagram

    Description automatically generated
* Nabrajanje
  + Enumeracija
    - Tip podatak koji sadržava uređene parove imenovanih identifikatora i njima pridruženih vrijednosti
    - Diagram

      Description automatically generated
* Parametrizirani razred
  + Predstavlja predložak za generiranje novih elemenata modela iz predloška
  + Mora se povezati sa stvarnim vrijednostima
    - Vezujući odnos
      * Dodjeljuje vrijednosti parametrima predloška
      * Generira novi element modela iz predloška
* Objektno orijentirana analiza i oblikovanje
  + OOAD
    - Iterativni i inkrementalni pristup analizi i oblikovanju sustava primjenom objektno usmjerenih koncepata
  + Aktivnosti u životnom ciklusu
    - Analiza
      * Usmjerena na razumijevanje problema
    - Oblikovanje
      * Usmjereno na razumijevanje implementacije
  + Aktivnosti oblikovanja
    - Diagram

      Description automatically generated
* Razlika analize i oblikovanja
  + Osnovna razlika je u naglasku na detaljima pogodnim za implementaciju

Graphical user interface, text, application

Description automatically generated

* Objektno usmjerena analiza
  + Cilj
    - Obuhvatiti cjelovito i nedvosmisleno zahtjeve sustava te ustanoviti objekte i načine njihove suradnje
  + Tehnike
    - Ad-hoc
    - Lista imenica
    - CRC kartice
    - Analiza obrascima uporabe
  + 3 osnovna koraka
    - Oblikovanje obrazaca uporabe
    - Oblikovanje razreda
    - Modeliranje dinamičkog ponašanja
* Objektno usmjereno oblikovanje
  + Usmjereno na razumijevanje implementacije
  + Analitički model => model oblikovanja (podloga za implementaciju)
  + Primjeri principa oblikovanja
    - SOLD, GRASP, RDD
* Proces razvoja dijagrama razreda
  + Modeli
    - Istraživački model domene primjene
      * Opis razreda na konceptualnoj razini
      * Cilj razumijevanja domene
    - Model domene sustava
      * Opis razreda na razini specifikacije, modelira se apsekt domene koji će biti predstavljen programskim sustavom
    - Model sustava
      * Implementacijska razina
  + Oblikovanje razreda
    - Iterativni postupak
    - Cilj
      * Utvrditi razrede, njihove atribute, operacije i odnose
    - Koraci
      * Identifikacija razreda
      * Određivanje temeljnih odgovornosti razreda
      * Dodavanje pridruživanja i atributa
      * Specificiranje ovisnosti s drugim razredima
      * Specifikacija sučelja
      * Ponavljanje koraka dok se ne dobije zadovoljavajući model
* Pristupi identifikaciji OO elemenata
  + Klasični pristup
  + Analiza obrazaca uporabe
  + Zajednički obrasci razreda
  + Analiza domene
  + Analiza ponašanja
  + Razred, odgovornosti i suradnje – CRC kartice
* Utvrđivanje razreda
  + Obuhvaća i modelira stvarnost
  + Otkrivanje razreda tijekom razvoja modela domene
  + Osmišljavanje razreda tijekom razvoja sučelja ili arhitekture
  + Mogućnost ponovne uporabe razreda
* Zajednički obrasci razreda
  + Temelji se na bazi znanja zajedničkih razreda koji su predloženi u praksi
    - Događaji
    - Organizacija
    - Ljudi
    - Mjesta
    - Opipljive stvari
    - Koncepti
* Utvrđivanje odgovornosti
  + Analiza obrazaca uporabe
    - Svaki funkcionalni zahtjev mora se pridijeliti nekom razredu
  + Razred nema odgovornosti = beskoristan
  + Voditi računa o SRP-u
* Identificiranje pridruživanja i atributa
  + Zapičinje s razredima u središtu / najvažnijim
  + Proširivati dijagram iznutra prema van
* Pridruživanja između obrazaca
  + Postoji ako razreda
    - Posjeduje
    - Upravlja
    - Spojen je s
    - Odnosi se prema
    - Dio je
    - Ima dijelove
    - Član je
    - Ima članove
      * Nekog drugog razreda
* Utvrđivanje operacija
  + Klasifikacija
    - Osnovne
      * Konstruktori i destruktori
    - Pomoćne
      * Selectors, getters
    - Modificirajuće
      * Modifiers, setters
    - Operacije pretvorbe
      * Conversion
    - Ponavljajuće operacije
      * Iterators
* Identificiranje generalizacija i sučelja
  + Dva načina
    - Odozdo prema gore
    - Odozgo prema dolje
  + Stvaranje sučelja umjesto nadrazreda ako postoji potreba za varijablom koja bi morala držati instance nekoliko razreda

Diagram

Description automatically generated