Oblikovanje programske potpore

2012./2013. grupa P01

Procesi programskog inženjerstva

Prof.dr.sc. Vlado Sruk

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroel., računalne i inteligentne sustave

Literatura

- Sommerville, I.: ***Software Engineering***, Addison-Wesley, 2007.
- Lethbridge, T. C., Laganière: ***Object-Oriented Software Engineering: Practical Software Development Using UML and Java***, McGraw-Hill, 2005.
- Pollice, G., Augustine L.; et al: ***Software Development for Small Teams: A RUP-Centric Approach***, Addison-Wesley, 2003

Tema

- Upoznavanje i opis modela procesa programskog inženjerstva.
- Uloga iteracija u modelima procesa programskog inženjerstva.
- Generičke aktivnosti u procesima programskog inženjerstva:
  - osnove modela za analizu zahtjeva, oblikovanje programske potpore, validaciju i evoluciju
- Prikaz modela unificiranog procesa (engl. *Unified Process)*
- Svojstva CASE tehnologija u procesima programskog inženjerstva

Procesi programskog inženjerstva

- *Podsjetnik:*
- Strukturirani skup aktivnosti koji čini okvir neophodan za izvođenje sustavnog plana razvoja i oblikovanja programske potpore
- Generičke aktivnosti :
  - specifikacija;
  - razvoj i oblikovanje;
  - validacija;
  - evolucija.
- Model procesa programskog inženjerstva je apstraktna reprezentacija procesa.
  - predstavlja opis procesa iz određene perspektive.

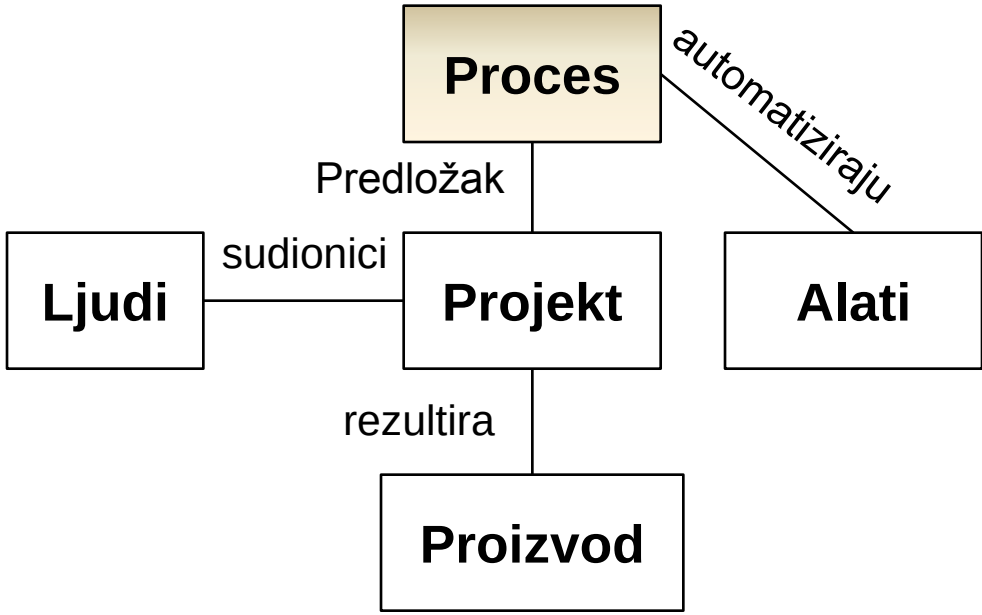
Uloga procesa

- Proces razvoja programske potpore osigurava:
  - potrebne informacije
  - u trenutku kada su potrebne
  - u upotrebljivom obliku
    - ni previše ni premalo
    - jednostavno pronalaženje

- Proces definira ***TKO*** radi ***ŠTO***, ***KADA*** i kako postići željeni cilj.

Pogledi na razvoj PP

- Upravljanje projektima PP (*engl. Software Project Management*) je krovna aktivnost u području programskog inženjerstva
- Efikasno programsko inženjerstvo fokusira se na:
  - ljudi – *engl. People*
    - najvažniji
  - projekt – *engl. Project*
    - rezultira produktom
  - proizvod – *engl. Product*
    - nije samo kod
  - proces – *engl. Process*
    - upravlja projektom
  - alati – *engl. Tools*



Model životnog ciklusa

- *engl. Lifecycle model*
- U svrhu ostvarenja cilja provodi se niz povezanih aktivnosti koje nazivamo fazom životnog ciklusa.
- Model životnog ciklusa opisuje faze od začetka projekta do kraja životnog vijeka proizvoda
  - opisuje odnose glavnih točaka procesa i rezultata
  - određuje redoslijed odvijanja aktivnosti
- **Životni ciklus razvoja programske potpore**
  - *engl. software development life-cycle, software development process, software life cycle, software process*
  - predstavlja standardizirani format procesa programske potpore.
- Primjer standarda:
  - ISO/IEC 12207 *Systems and software engineering — Software life cycle processes*

Česti modeli

- Ad hoc model *engl. Code and fix life-cycle model*
- Vodopadni model *engl. Waterfall life-cycle model*
- Prototipni model *engl. Rapid prototyping life-cycle model*
- Inkrementalni model *engl. Incremental/evolutionary*
- Spiralni model *engl. Spiral life-cycle model*
- Unificirani proces *engl. Unified process (UP)*
- Agilni proces *engl. Agile processes*

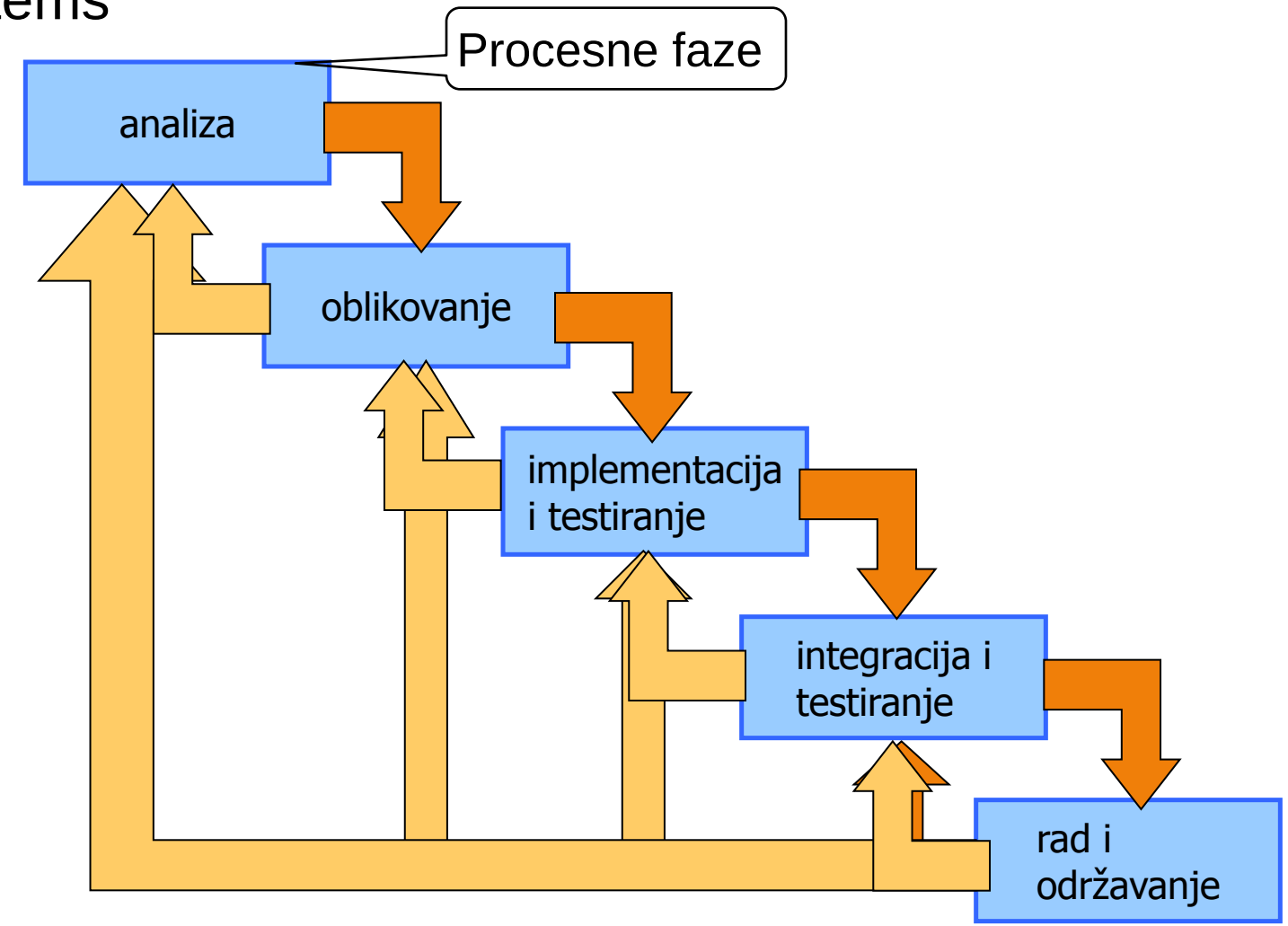
- Ovo su samo primjeri češće upotrebljavanjih modela od stotine postojećih te njihovih podvarijanti

Generički modeli programskog inženjerstva

- ***Vodopadni***
  - odvojene i specifične faze specifikacije i razvoja
- ***Evolucijski***
  - specifikacija, razvoj i validacija su isprepleteni
- ***Komponentno usmjeren***
  - sustav se gradi od postojećih komponenata
- ***Unificirani proces –*** *engl. Unified Process*
  - zasnovan na oblikovanju uporabom modela
  - *engl. Model Based Design*

Vodopadni model programskog inženjerstva

- 1970 Winston W. Royce "Managing the development of large software systems"



Procesne faze u vodopadnom modelu

- Analiza zahtjeva i definicije
- Razvoj i oblikovanje sustava i programske potpore
- Implementacija i ispitivanje (testiranje) modula
- Integracija i testiranje sustava
- Uporaba sustava i održavanje

- Temeljni značajka vodopadnog modela:
  - pojedina faza ***se mora dovršiti*** prije pokretanja nove faze!!!

Pretpostavke vodopadnog modela

- Zahtjevi poznati prije oblikovanja
- Zahtjevi se rijetko mijenjaju
- Korisnik zna što treba i nisu mu potrebna pojašnjenja
- Oblikovanje se može provoditi odvojeno i rijetko dovodi do pogrešaka
- Tehnologija sama po sebi rješava neke zahtjeve
- Sustavi nisu složeni

Problemi vodopadnog modela

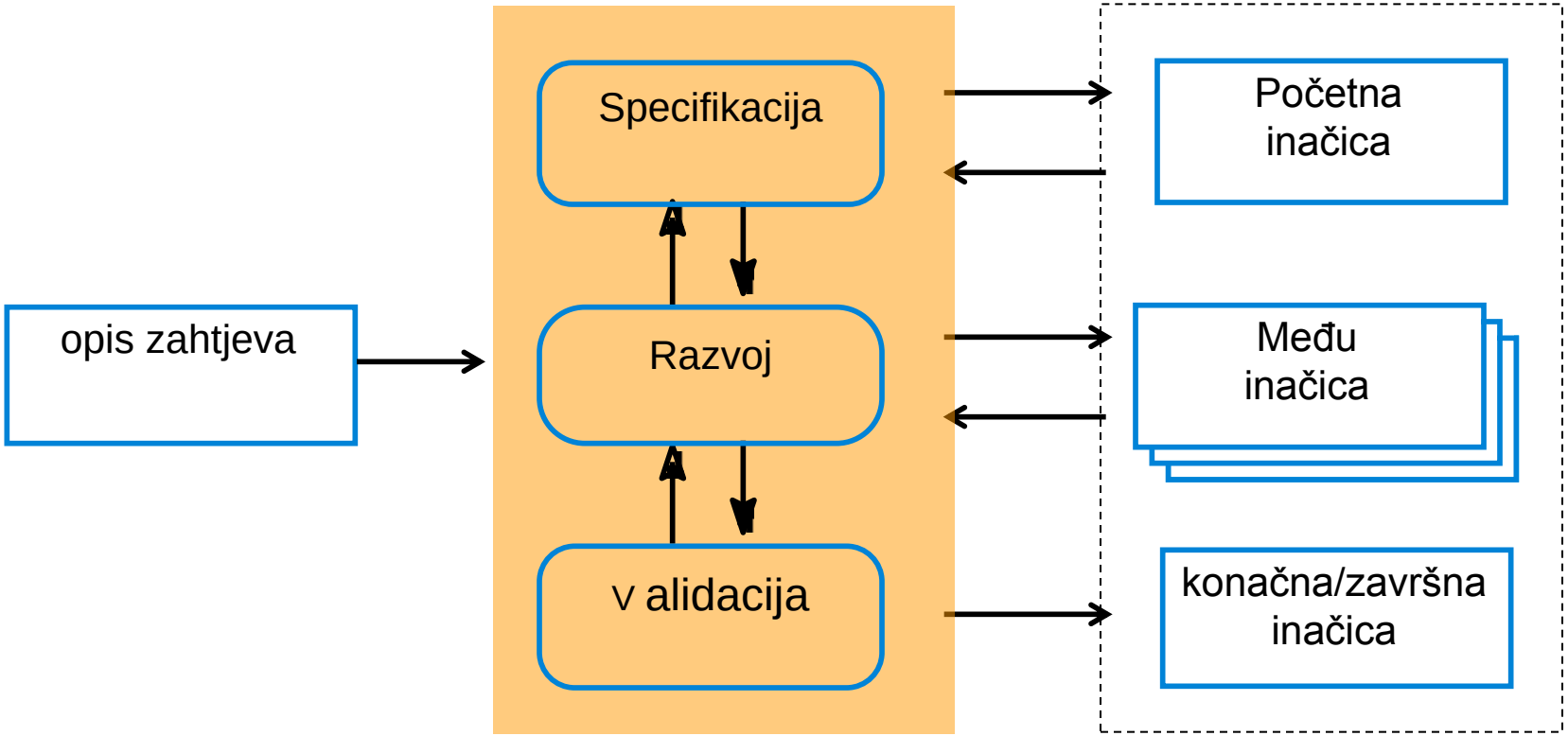
- Temeljni nedostatak vodopadnog modela
  - poteškoće ugradnje promjena nakon što je proces pokrenut
  - nefleksibilna podjela projekta u odvojene dijelove čini implementaciju promjena koje zahtijeva kupac vrlo teškim.
- Model je prikladan samo ako su zahtjevi dobro razumljivi i eventualne promjene svedene na minimum.
- Vrlo malo poslovnih sustava ima stabilne zahtjeve.
- Vodopadni model se uglavnom koristi za velike inženjerske projekte gdje se sustav razvija na više odvojenih mjesta.

Evolucijski model razvoja i oblikovanja

- Dva uobičajena postupka:
- ***Metoda odbacivanja prototipa***
  - cilj je razumijevanje zahtjeva sustava.
  - započinje se s ***grubo definiranim zahtjevima*** koji se tijekom postupka razjašnjavaju u smislu što je doista potrebno.
  - prototip NIJE sustav koji radi
    - sučelja i kosturi stvarne funkcionalnosti
    - zašto korisnici misle da ga mogu odmah upotrijebiti?
- ***Istraživački razvoj i oblikovanje***
  - cilj ovakvog pristupa je kontinuiran rad s kupcem na temelju inicijalne specifikacije.
  - započinje se s ***dobro definiranim zahtjevima*** a nove funkcionalnosti se dodaju temeljem prijedloga kupca.

Evolucijski model razvoja i oblikovanja

Paralelne aktivnosti
- ponavljaju se tijekom razvoja



Evolucijski model razvoja i oblikovanja

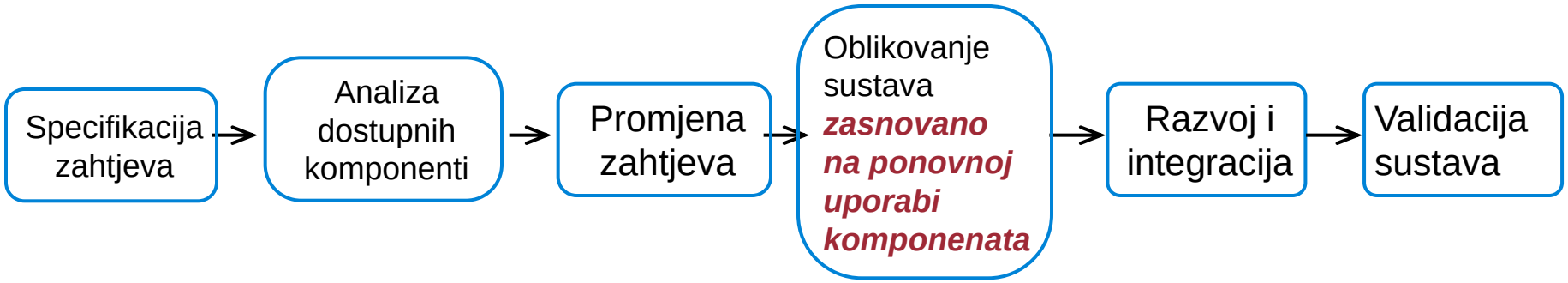
- ***Problemi***:
  - proces razvoja i oblikovanja nije jasno vidljiv.
  - sustavi su često vrlo loše strukturirani.
  - često su potrebne posebne vještine
    - npr. brzi razvoj prototipa – *engl. Rapid System Prototyping*
- ***Primjenljivost***:
  - za male i srednje interaktivne sustave.
  - za dijelove velikih sustava
    - npr. korisničko sučelje
  - za sustave s kratkim vijekom trajanja.

Programsko inženjerstvo zasnovano na komponentama

- *engl. Component-based software engineering - CBSE*
- Sustav se integrira višestrukom uporabom postojećih komponenata ili uporabom komercijalnih, gotovih komponenata (*engl. commercial-of-the-shelf COTS*).
- Stupnjevi procesa:
  - specifikacija i analiza zahtjeva
  - analiza komponenata
  - modifikacija zahtjeva
  - oblikovanje sustava s višestrukom uporabom komponenata (*engl. reuse*)
  - razvoj i integracija
- S povećanom standardizacijom komponenata CBSE
  - zrelo tržište
  - 12 International Symposium on Component Based Software Engineering (CBSE)

Programsko inženjerstvo zasnovano na komponentama

- Višestruka uporaba komponenata
- *engl. Reuse-oriented development*

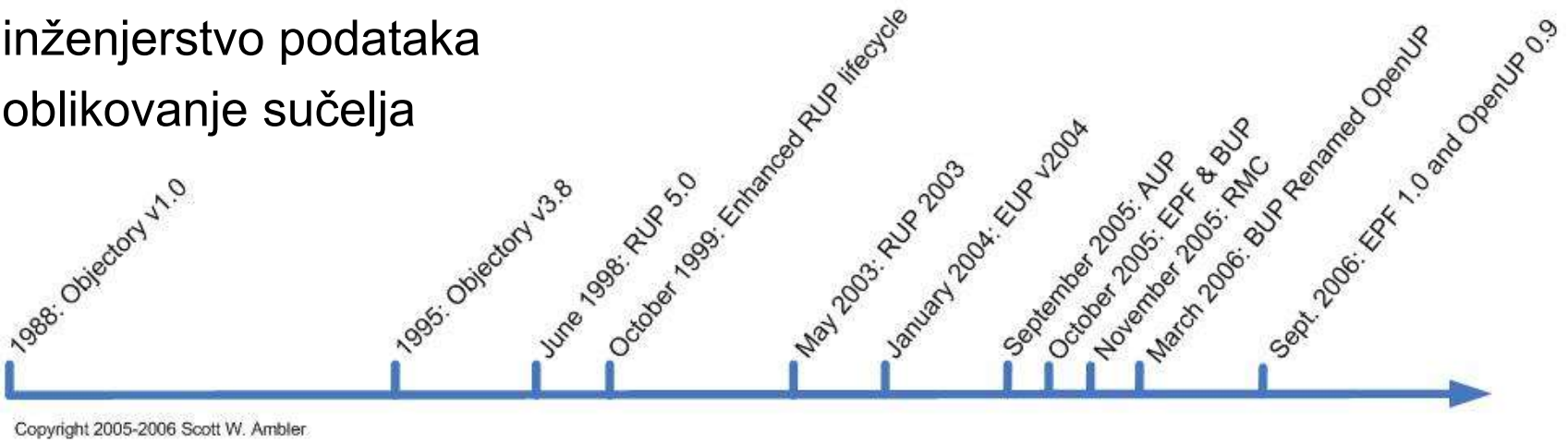


Unificirani proces

- *engl. Unified Process*
- **Model** procesa programskog inženjerstva izveden na temelju jezika za modeliranje UML-a (*engl. Unified Modeling Language*) i pridruženih aktivnosti.
- Svojstva
  - priznaje utjecaj korisnika
  - sugerira evolucijski pristup
  - podržava OO
  - prilagodljiv
- Najčešće opisan kroz tri perspektive:
  - dinamička perspektiva koja pokazuje slijed faza kroz vrijeme.
  - statička perspektiva koja pokazuje aktivnosti procesa.
  - praktična perspektiva koja sugerira aktivnosti kroz iskustvo i dobru praksu.

Povijesni razvoj

- 1999. Booch, Jacobson, Rumbaugh: “The Unified Software Development Process”
  - objedinjivanje tri različite metodologije ⇒ **Unified Process**
- Objedinjuje postupke:
  - inženjerstvo poslovnog procesa
  - rukovanje zahtjevima
  - rukovanje oblikovanjem i promjenama
  - ispitivanje
  - evaluaciju performansi
  - inženjerstvo podataka
  - oblikovanje sučelja



Prog. potpora za unificirani proces

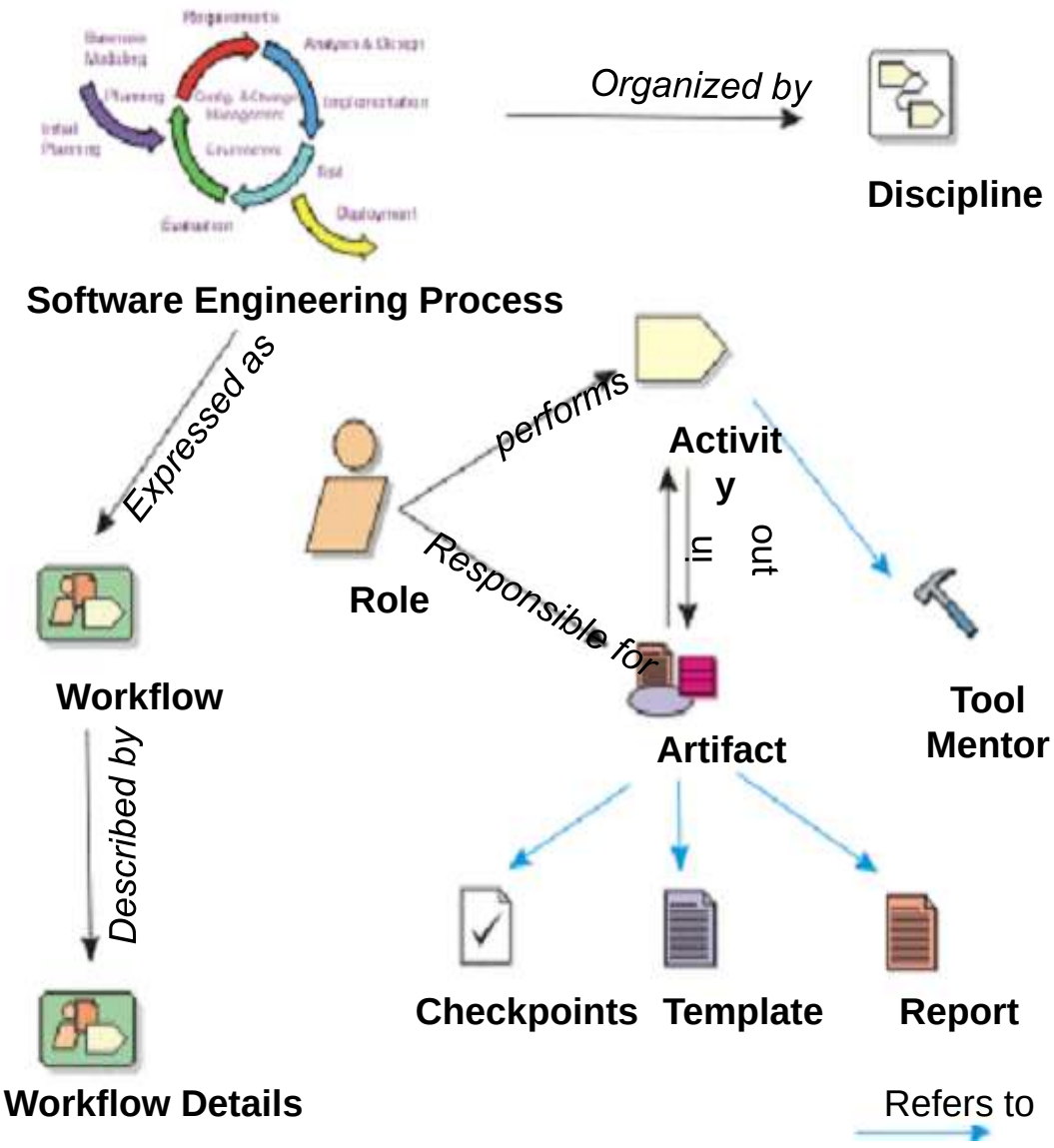
- Modeliranje razvoja OO prog. podrške
- Rational Unified Process (RUP)
  - poznat kao Unified Process
  - IBM Rational Unified Process
- Object Oriented Software Process - OOSP
- OPEN Process
  - www.open.org.au
- ICONIX Unified Object Modeling
  - www.iconixsw.com

The Rational Unified Process

- Obilježja modernog modela procesa
  - promiče više iteracija na pojedinim inkrementima (dijelovima) programske potpore
  - temelji se na obrascima uporabe
    - predlošci scenarija
  - u fokusu je arhitektura sustava
- RUP predstavlja okosnicu procesa (*engl. process framework*) za razvoj i/ili prilagodbu programske potpore.
- RUP je proizvod koji razvija i održava IBM (zaštićena robna marka)
  - metodologija *engl. process product*
  - integrirani skup programskih alata
  - http://www-01.ibm.com/software/awdtools/rmc/

Struktura RUP-a

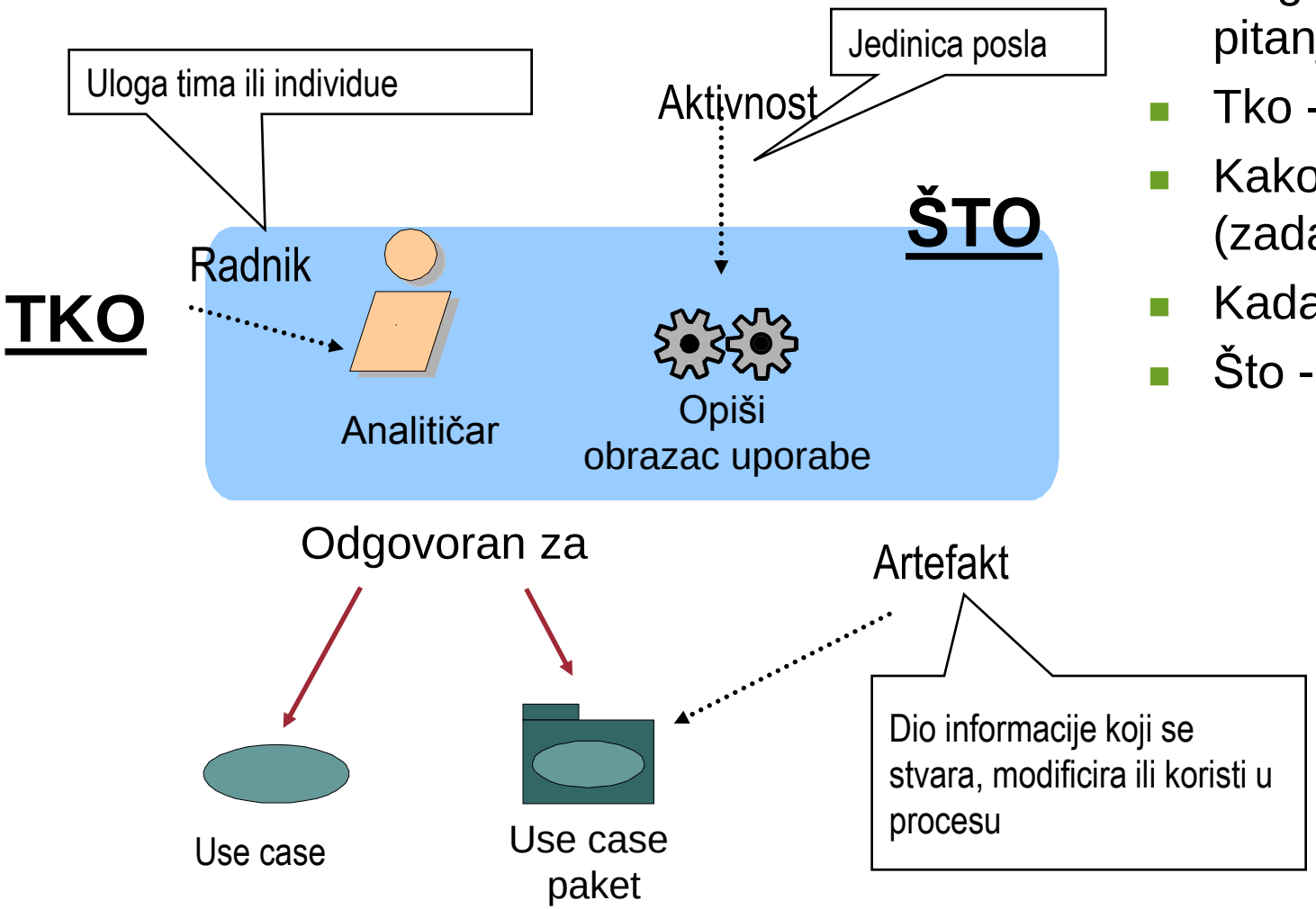
- Elementi RUP-a:
- Proces- *engl. process*
  - skup koraka
- Disciplina – *engl. discipline*
  - skup aktivnosti
- Uloga – *engl. role*
  - ponašanje i odgovornosti
- Aktivnosti – *engl. activity*
  - radnje daju neki rezultat
- Artefakt – *engl. artifact*
  - rezultat procesa (aktivnosti)
- Radni tijek – *engl. workflow*
  - sekvenca aktivnosti koja producira vidljiv rezultat



Izvor: Charbonneau S.:Software Project Management - A Mapping between RUP and the PMBOK

RUP kao proces

- Predstavlja inženjerski pristup

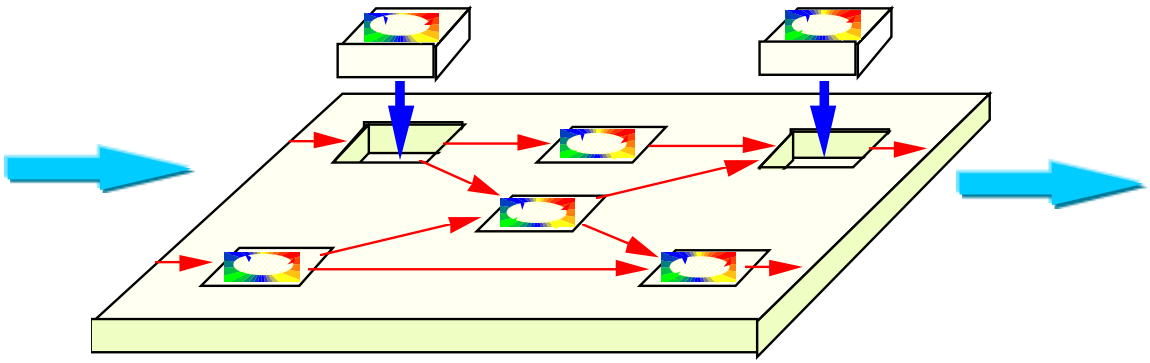


- Odgovori na pitanja:
- Tko -> ljudi (uloge)
- Kako ->aktivnosti (zadaci)
- Kada ->- radni tijek
- Što -> Artefakt

RUP kao okosnica

- RUP inherentno podržava fleksibilnost i proširenje
- Omogućuje razne strategije životnog ciklusa
- Selektira koje artefakte treba proizvesti
- Definira aktivnosti i radnike
- Modelira koncepte

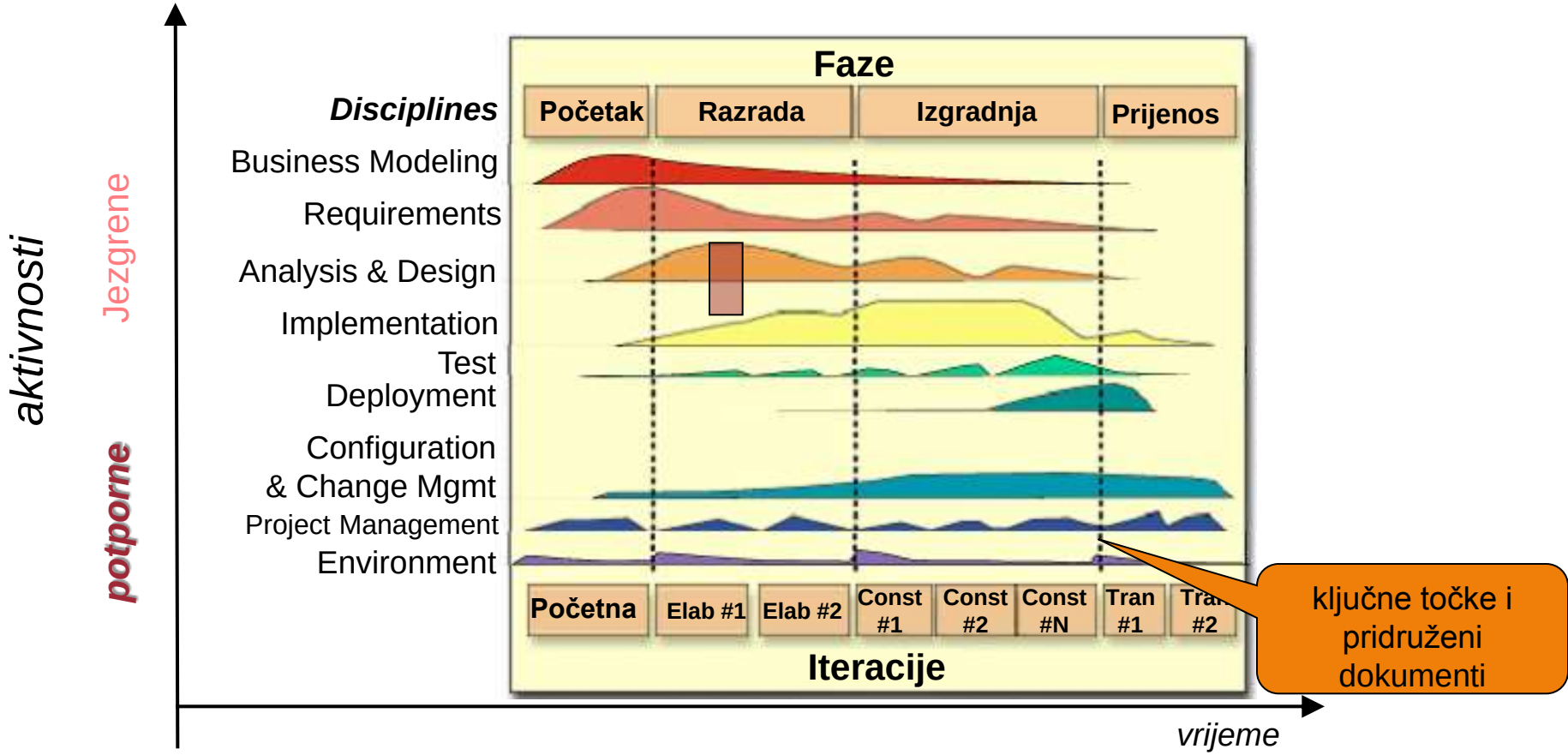
•*engl. Framework*



Ne postoji univerzalni proces oblikovanja programske potpore!!

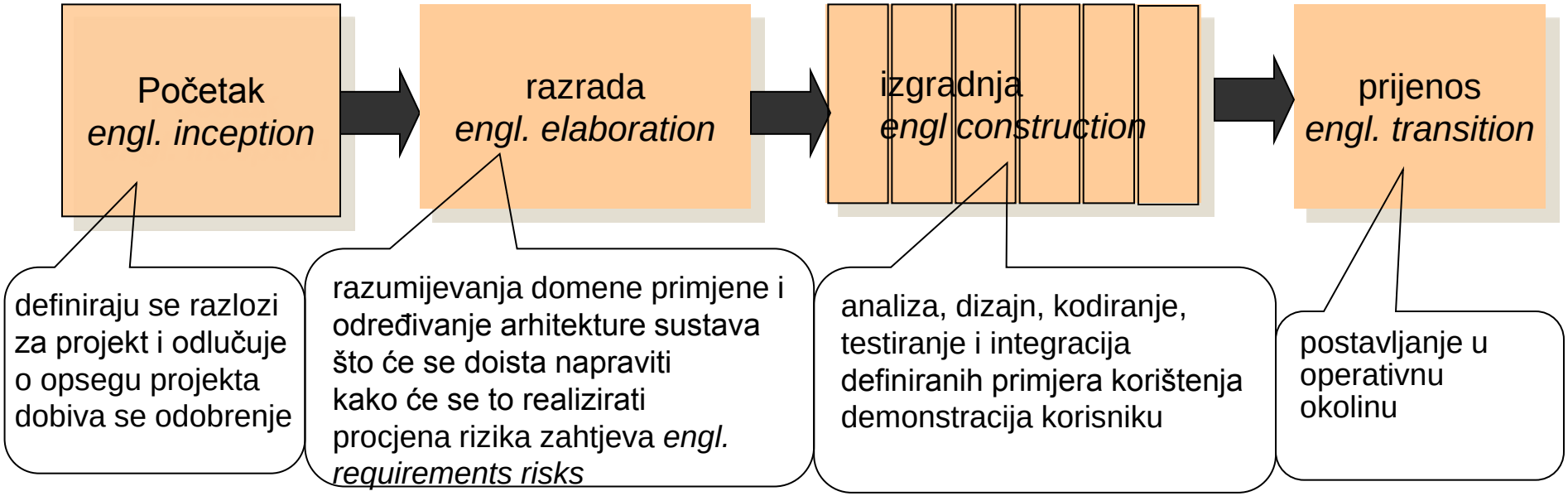
RUP proces

- Dvije dimenzije:
  - horizontalna - dinamika: ciklusi, faze, iteracije i ključne točke
  - vertikalna – statika: opis procesa: aktivnosti, discipline, uloge, artefakti
- Organizacija procesa u vremenu (fazama) i kontekstu (aktivnostima)



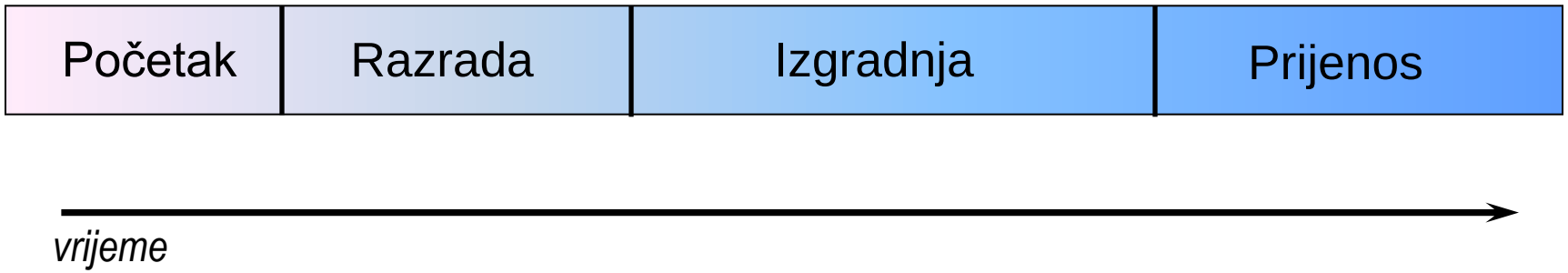
Pregled procesa razvoja

- ***Faze:*** Početak, razrada, izgradnja, prijenos
- ***Iteracija*** je sekvenca aktivnosti u okviru prihvaćenog plana i kriterija evaluacije.
  - rezultira u dokumentu, a kasnije i jednoj izvršnoj inačici programa (izdanju, *engl. Release*).



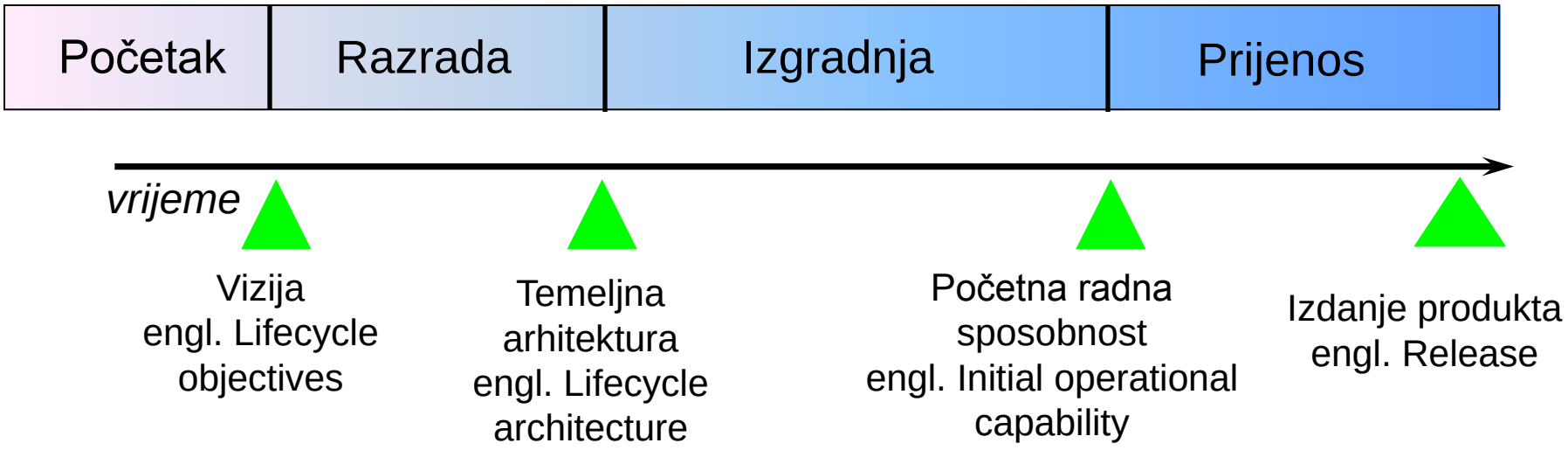
Faze u životnom ciklusu RUP procesa

- Početna faza (*engl. Inception*)
  - definira doseg projekta, razvoj modela poslovnog procesa
- Faza razrade (*engl. Elaboration*)
  - obuhvaća plan projekta, specifikaciju značajki i temelje arhitekture sustava
- Faza izgradnje (*engl. Construction*)
  - izgradnja produkta (oblikovanje, programiranje, ispitivanje)
- Faza prijenosa (*engl. Transition*)
  - prijenos produkta korisnicima (postavljanje u radnu okolinu)

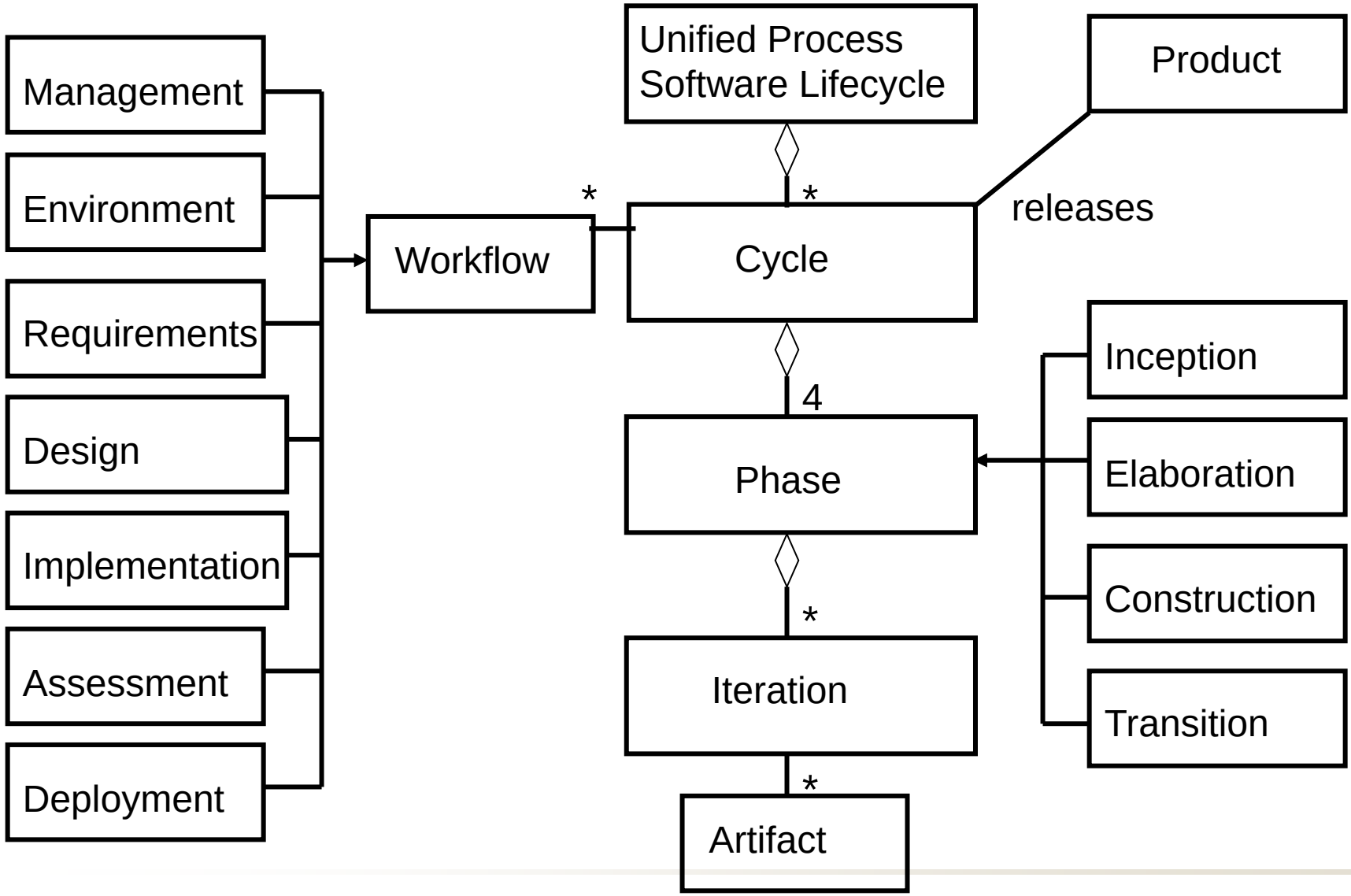


Ključne točke

- *engl. Milestones*
- Ključne točke definiraju pridružene dokumente ili aktivnosti



Formalna struktura



Implementacija Rational Unified Procesa

1: Procjena trenutnog stanja

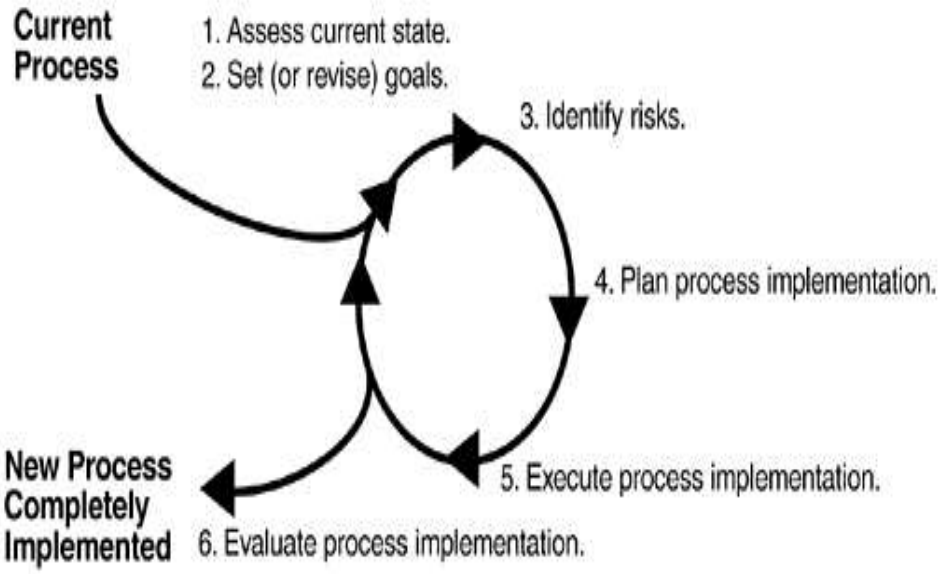
2: Postavljanje/revizija ciljeva

3: Identifikacija rizika

4: Planiranje procesa implementacije

5: Implementacija

6: Evaluacija implementacije



Prednosti uporabe UML-a

- Otvoren standard.
- Defacto industrijski standard
- Podupire cijeli životni ciklus oblikovanja programske potpore.
- Podupire različite domene primjene.
- Temeljen je na iskustvu i potrebama zajednice oblikovatelja i korisnika programske potpore.
- Razvijena dobra programska potpora

Mnogo dionika, mnogo pogleda

- Arhitekturu programske potpore različito vidi:
  - korisnik-kupac
  - vođa projekta
  - inženjer sustava
  - osoba koje razvija sustav
  - arhitekt
  - osoba koja održava sustav (*engl. Maintainer*)
  - drugi dionici
- Višedimenzionalna realnost
- Mnogo dionika
  - višestruki pogledi, višestruki nacrti

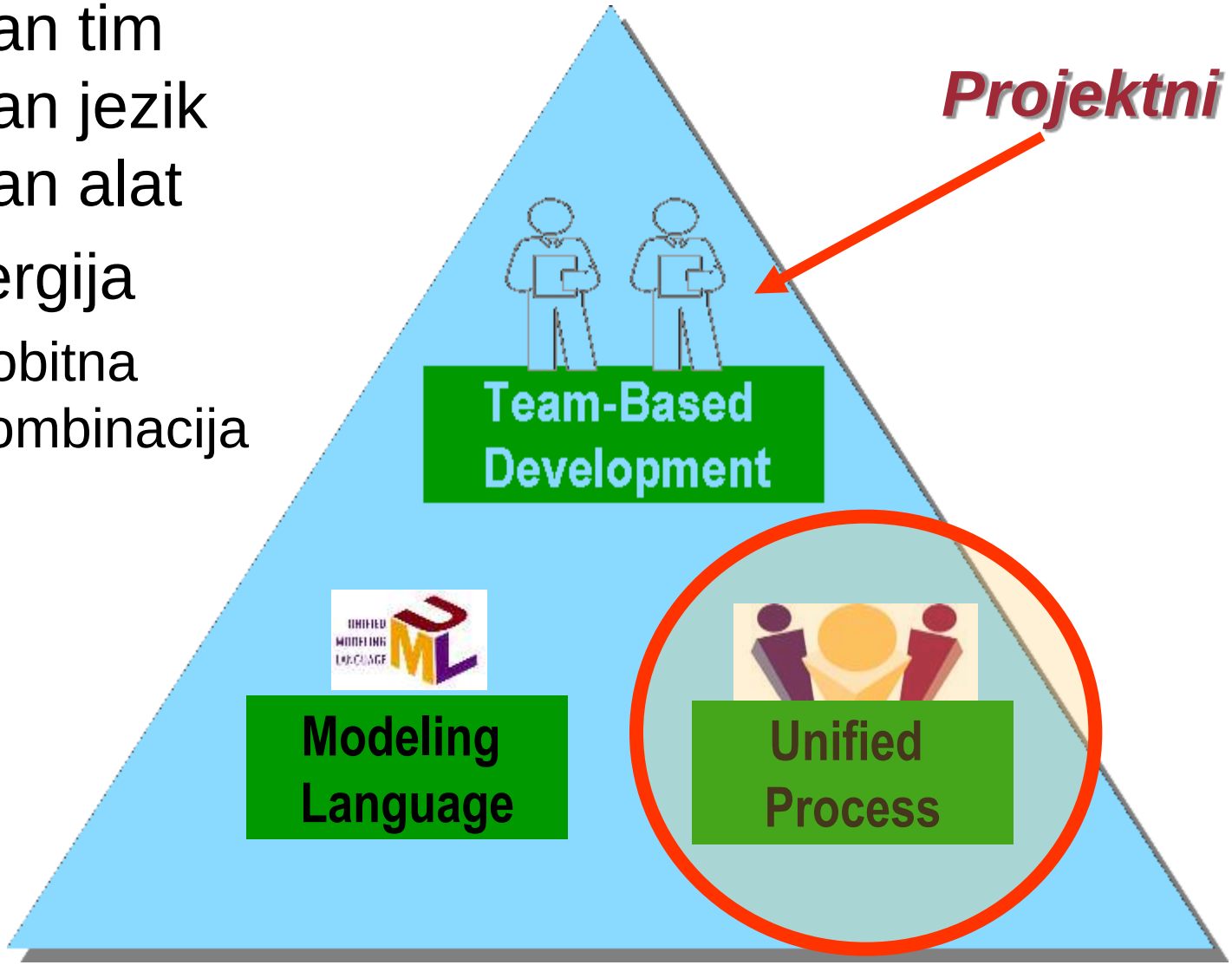
Koliko pogleda?

- Pogledi odgovaraju nekom kontekstu.
  - svi sustavi ne trebaju sve poglede:
    - Jedan procesor, nije potreban nacrt razmještaja procesora.
    - Jedan proces, nije potreba nacrt razmještaja procesa.
    - Vrlo mali programi, nije potreban nacrt implementacije.
  - neki dodatni pogledi:
    - Pogled podataka, pogled sigurnosti u sustavu

- Dijagram
  - svaki pogled dokumentira se nacrtom
- MODEL
  - skup više pogleda (dijagrama) u okviru nekog konteksta
  - model je potpuni apstraktan opis sustava iz određene perspektive.
    - reduciran, pojednostavljen, smanjen, ograničen

Tim + UML + RUP

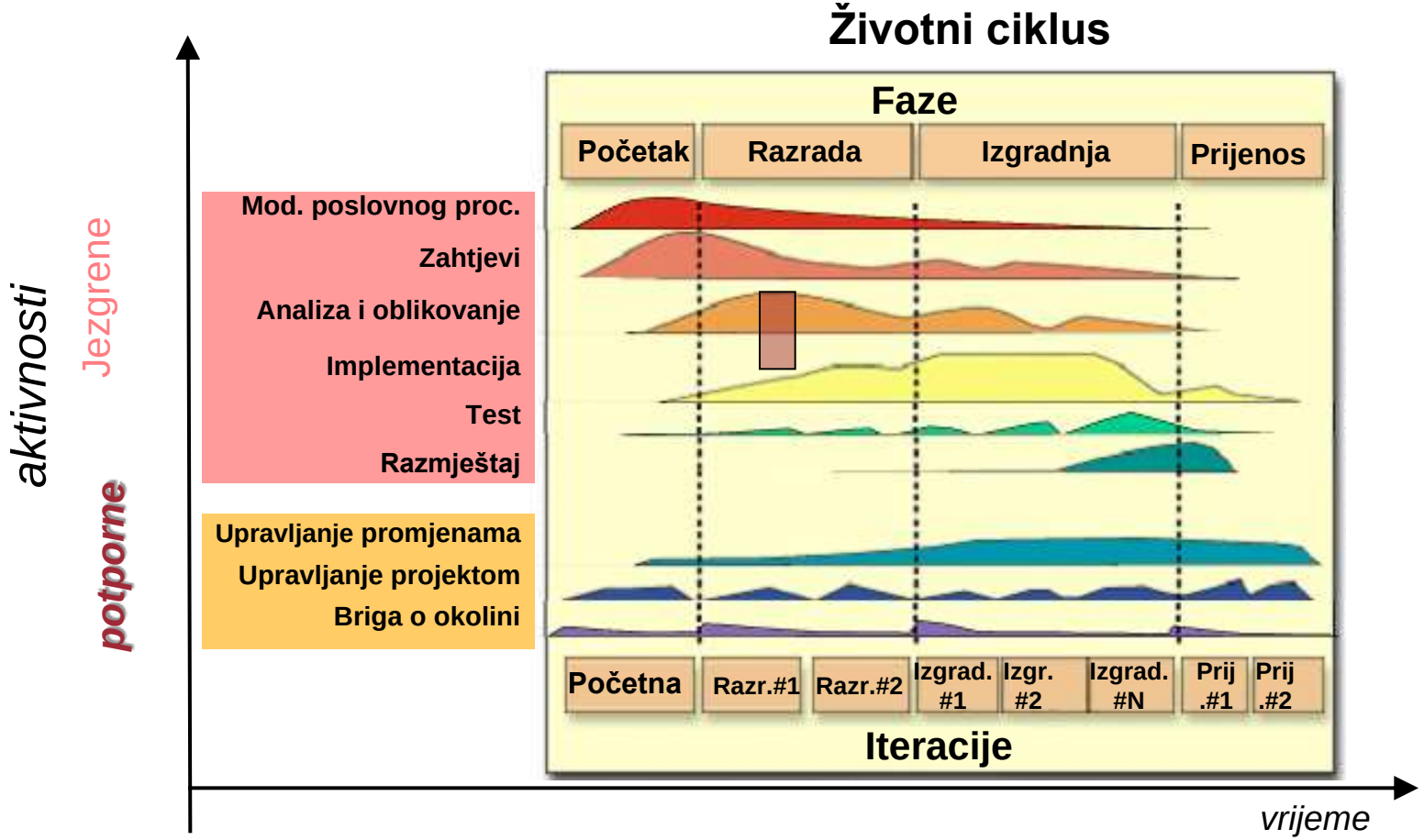
- Jedan tim
  Jedan jezik
  Jedan alat
- Sinergija
  - dobitna kombinacija



Izvor: I. Jacobson: UML i RUP

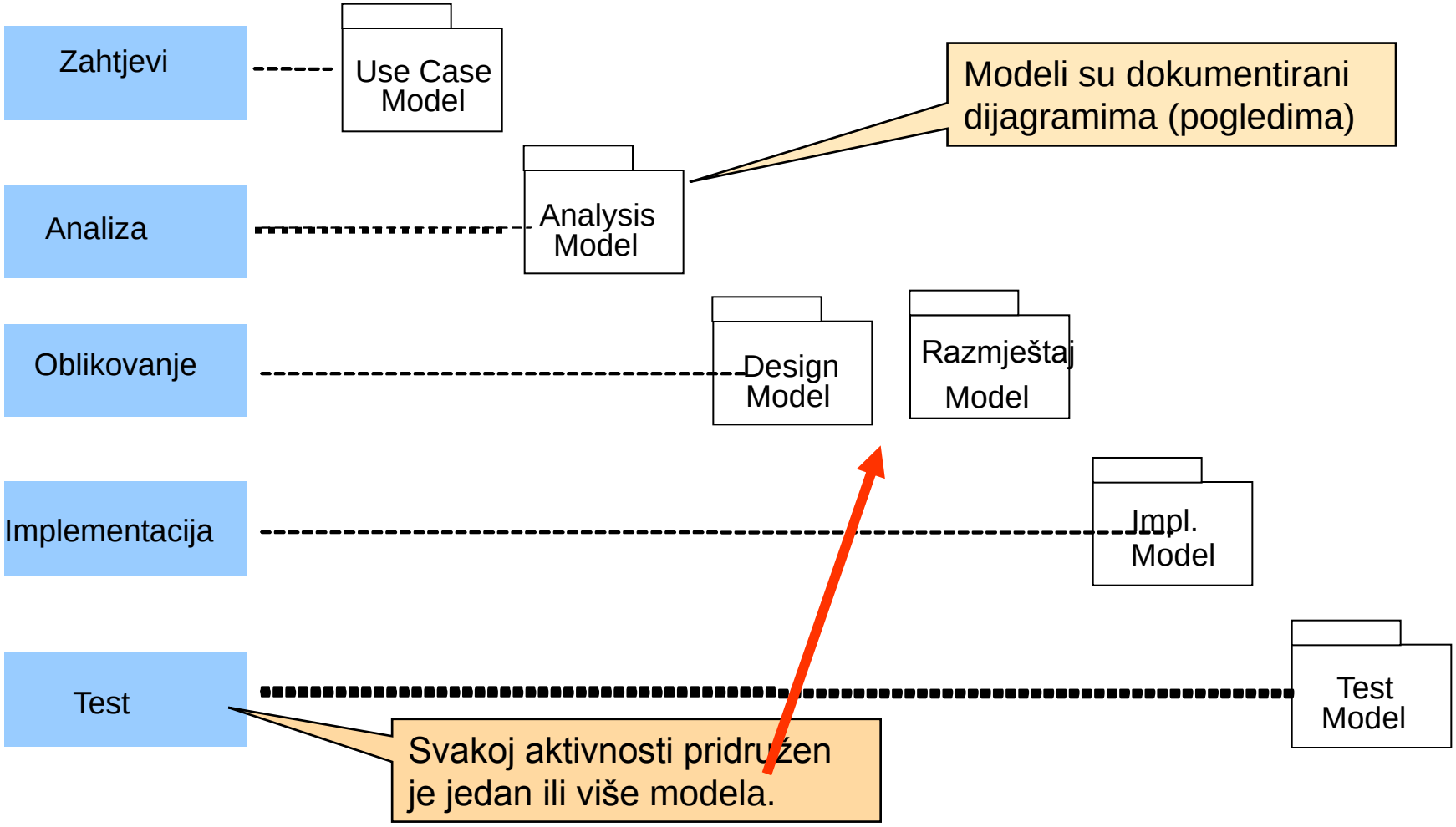
Iteracije i aktivnosti

- Tijek rada, *engl. workflow*

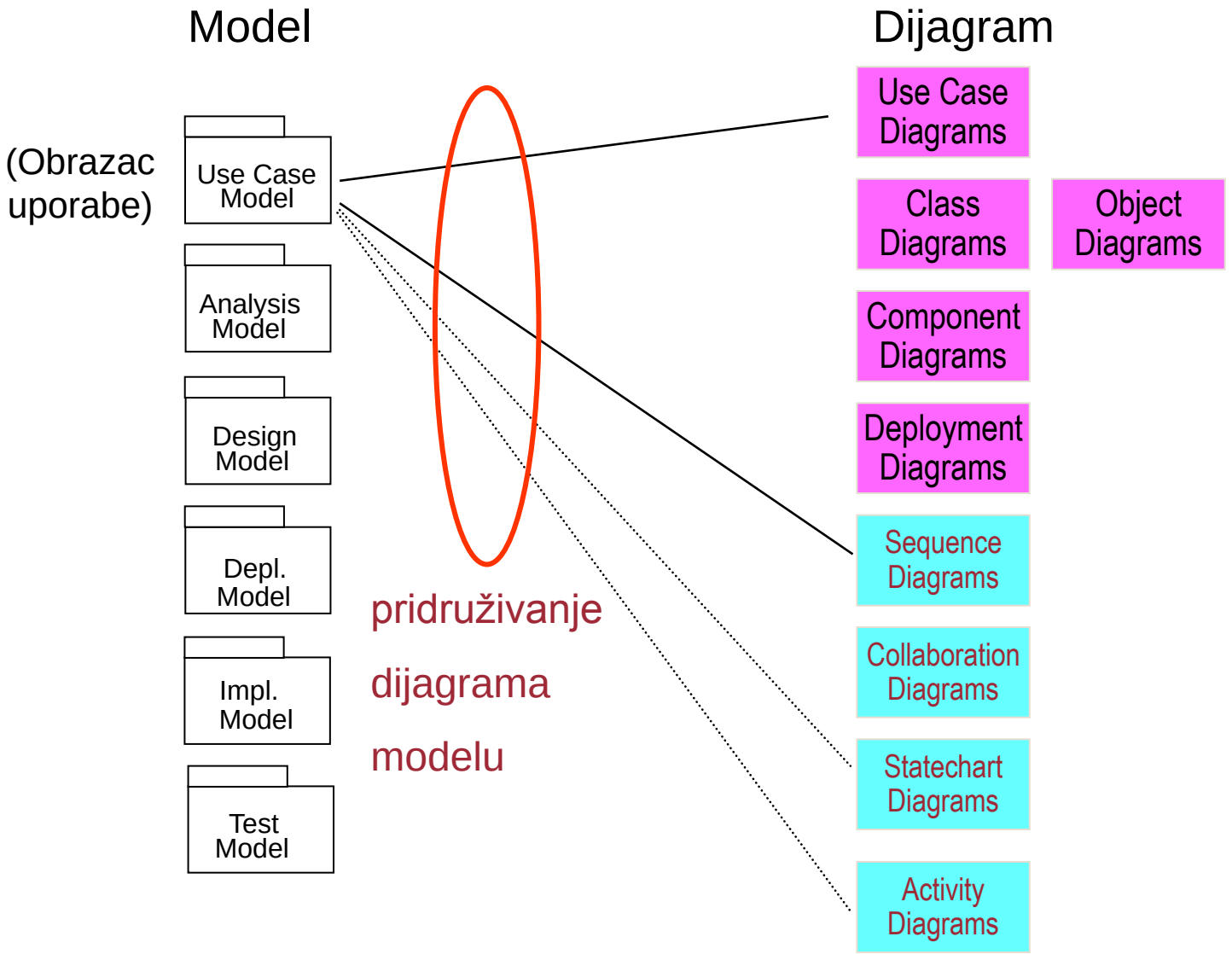


Modeliranje aktivnosti

Aktivnosti:

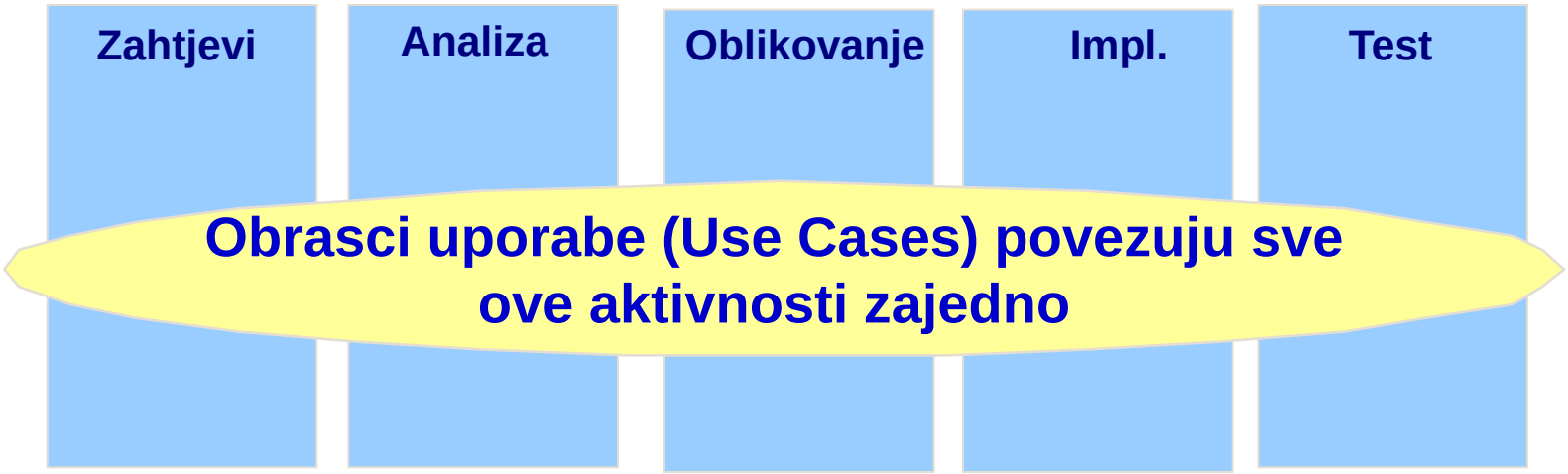


Primjer: Use Case Model



RUP i obrasci uporabe

- RUP se zasniva se na obrascima uporabe (tj. predlošku scenarija)
- Obrasci uporabe koriste se kroz sve faze

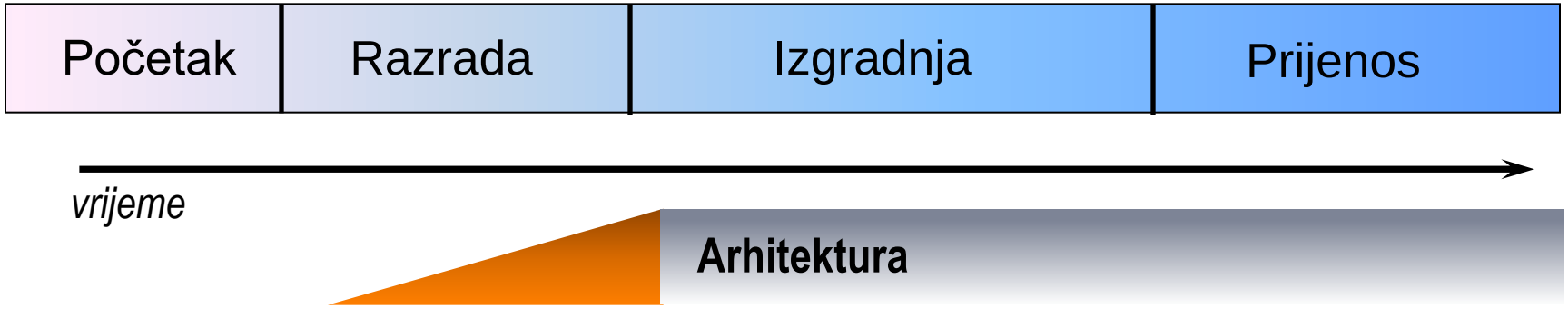


Obrasci uporabe - pokretači iteracija

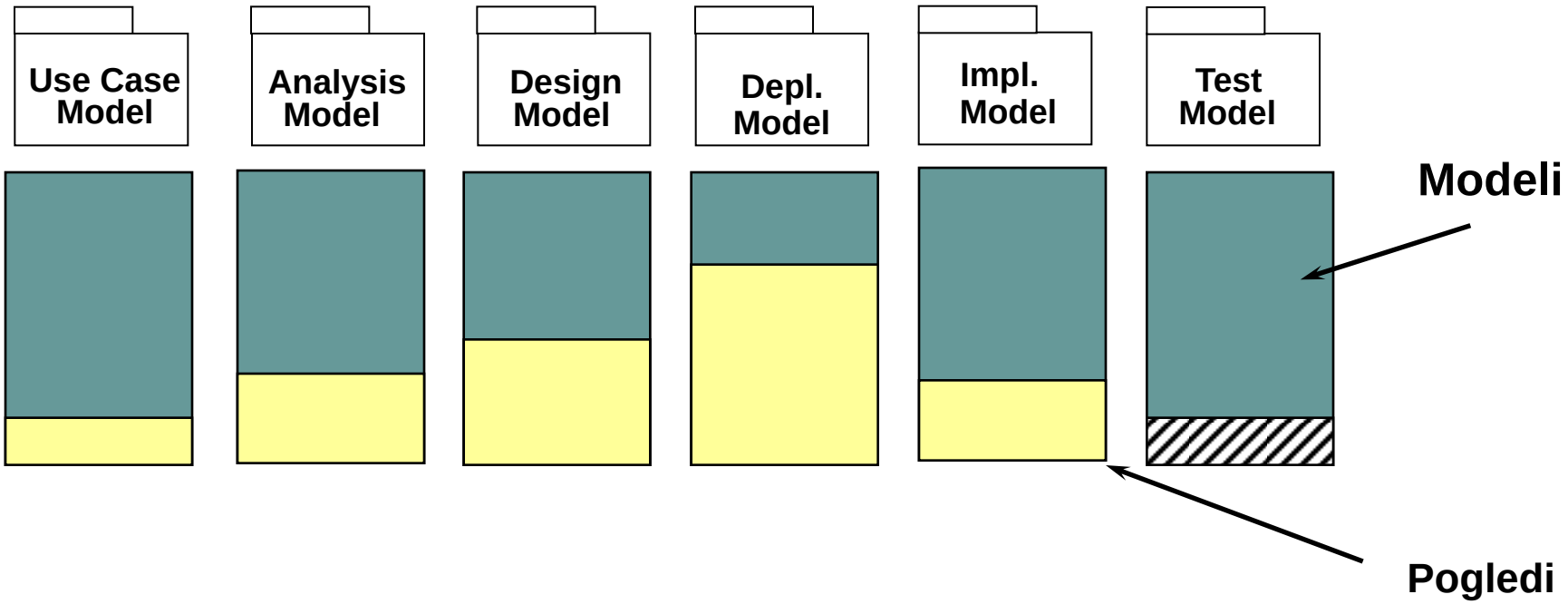
- Obrasci uporabe pokreću brojne aktivnosti u životnom ciklusu oblikovanja programske potpore, kao npr.:
  - kreiranje i validacija arhitekture sustava
  - definicija ispitnih slučajeva, scenarija i procedura
  - planiranje iteracija
  - kreiranje korisničke dokumentacije
  - razmještaj (*engl. Deployment*) sustava
- Sinkroniziraju sadržaj različitih modela

RUP: arhitektura sustava

- ***Arhitektura programske potpore*** je struktura ili strukture sustava koji sadrži elemente, njihova izvana vidljiva obilježja i odnose između njih.
- Modeli su prijenosnici za vizualizaciju, specifikaciju, konstruiranje (oblikovanje, implementacija) i dokumentiranje arhitekture
- RUP propisuje sukcesivno rafiniranje od grubog modela do konačne izvršne arhitekture
- RUP promiče oblikovanje programske potpore zasnovano na modelima (*engl. Model Based Design - MBD*)



Arhitektura i modeli

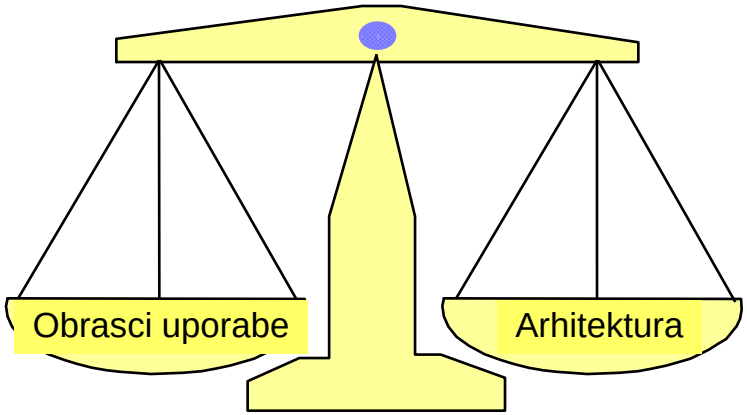


Arhitektura sadrži kolekciju pogleda u modele

(tj. kolekciju dijagrama)

Obrasci uporabe i arhitektura

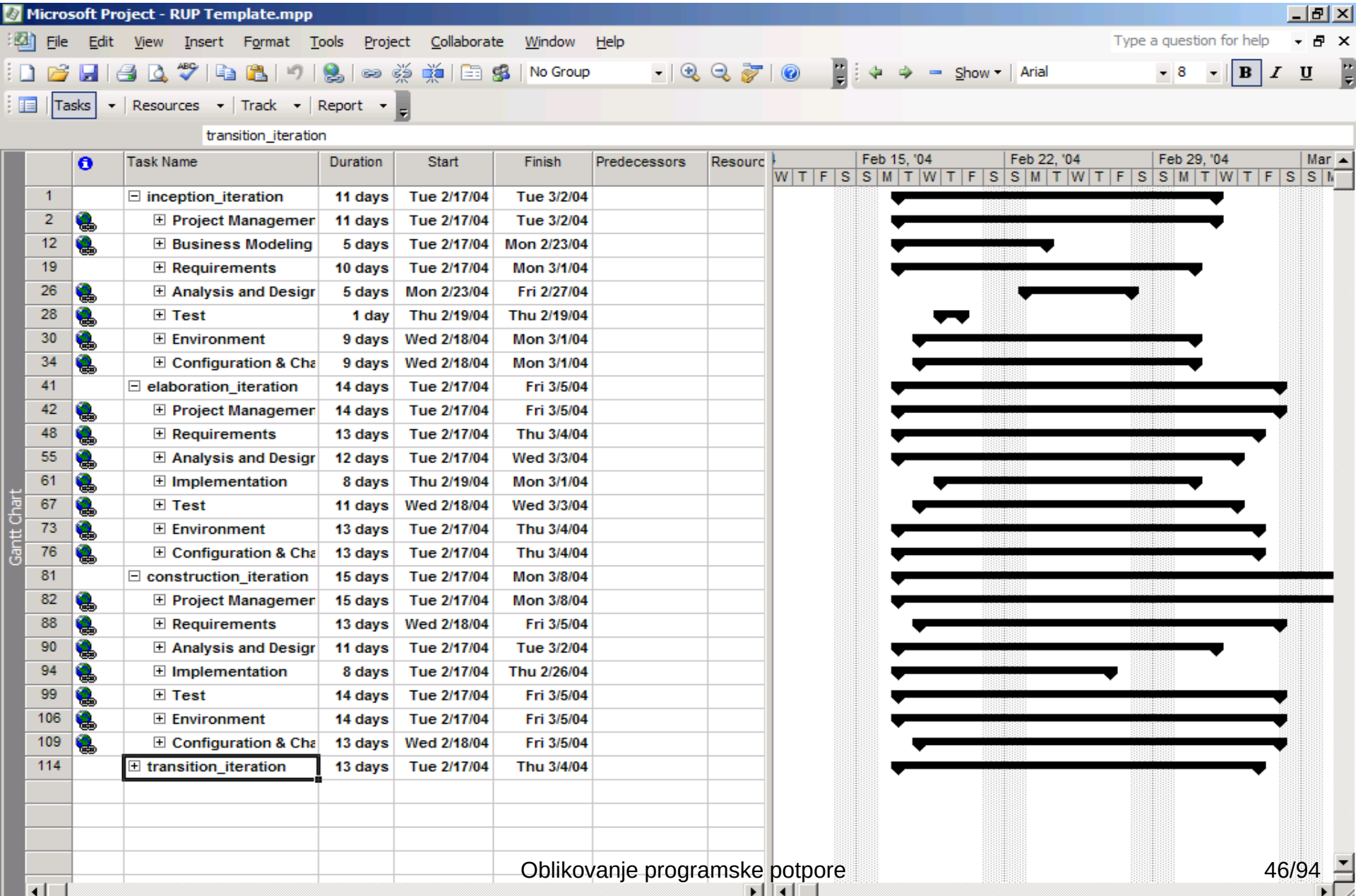
- Obrasci uporabe specificiraju funkcije
- Arhitektura specificira formu
- Obrasci uporabe i arhitektura moraju biti izbalansirani



Nabolja praksa svojstvena RUP-u

1. Iterativan razvoj
2. Upravljanje zahtjevima
3. Uporaba komponentno zasnovane arhitekture
4. Vizualno modeliranje (UML)
5. Kontinuirana verifikacija kvalitete
6. Upravljanje promjenama programske potpore

Primjer projektnog plana



Značajke RUP-a

- RUP posjeduje značajke ***iterativnog i inkrementalnog*** oblikovanja programske potpore.
- U središtu RUP procesa su ***obrasci uporabe*** sustava koji semantički povezuju sve aktivnosti.
- RUP definira i međusobno povezuje ***faze i aktivnosti*** procesa
- Za opis pojedinih ***aktivnosti koriste se*** odgovarajući modeli.
- ***Modeli*** su dokumentirani jednim ili više dijagrama
- ***Dijagrami*** su definirani UML standardom.
- ***Arhitektura sustava*** sadrži skup pogleda u modele (tj.. skup dijagrama)

Procesi programskog inženjerstva.

ITERACIJE U MODELIMA

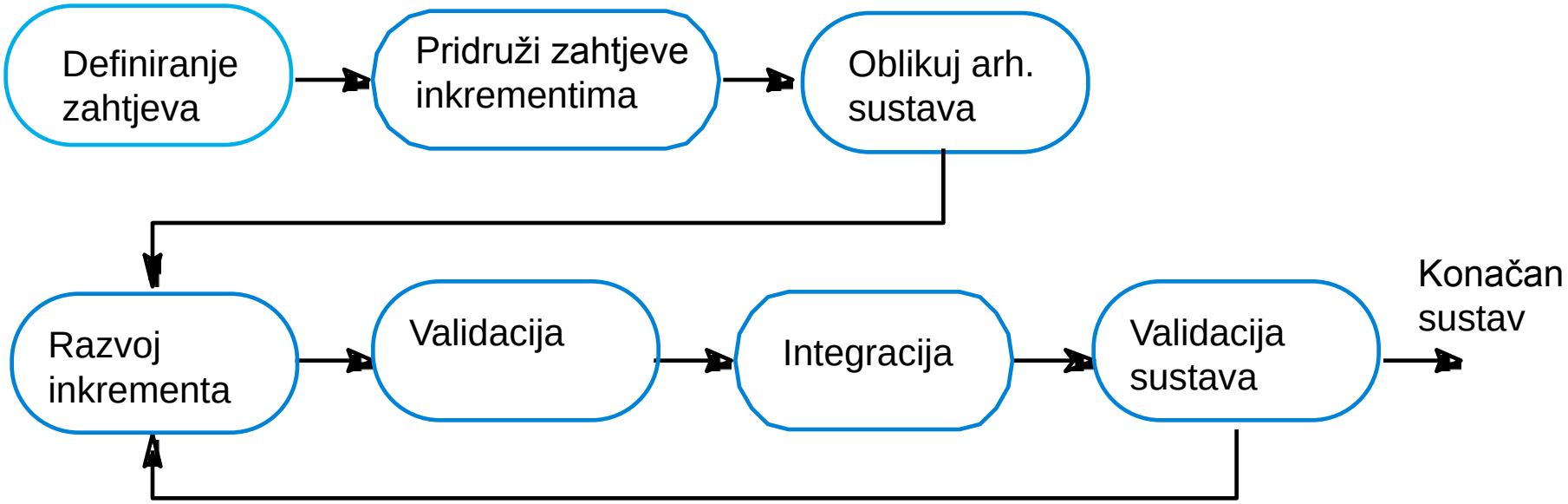
Iteracije

- *Iteracije u modelima procesa programskog inženjerstva.*
- Zahtjevi na sustav uvijek evoluiraju i prate razvoj projekta:
  - iteracije procesa sastavni su dio velikih projekata jer se pojedini stupnjevi moraju ponovno oblikovati.
  - iteracije se mogu primijeniti na bilo koji generički model procesa programskog inženjerstva.
- Postoje dva međuovisna pristupa iteracijama:
  - ***inkrementalni pristup***
  - ***spiralni razvoj i oblikovanje***
- Primjena u raznim modelima od modificiranog, vodopadnog, unificiranog, agilnih modela ....

Inkrementalni pristup

- Sustav se ne isporučuje korisniku u cjelini
  - razvoj, oblikovanje i isporuka razbiju se u inkrementalne dijelove koji predstavljaju djelomične funkcionalnosti.
- Zahtjevi korisnika se svrstaju u prioritetne cjeline.
  - dijelovi višega prioriteta isporučuju se u ranim inkrementima.
- S početkom razvoja pojedinog inkrementa njegovi zahtjevi se fiksiraju (zamrzavaju).
  - zahtjevi na kasnije inkremente nastavljaju evolvirati.

Inkrementalni razvoj, oblikovanje i isporuka



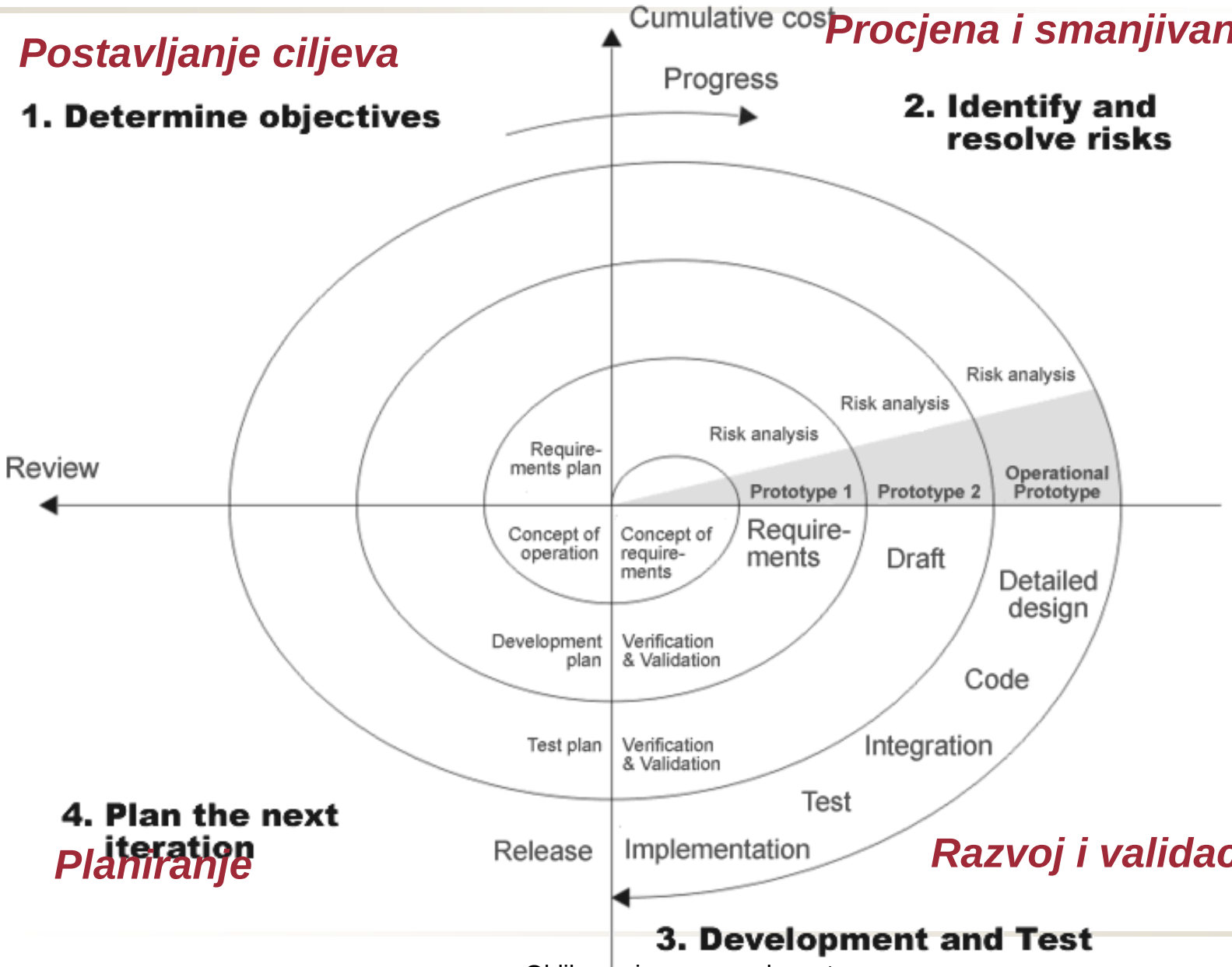
Prednosti inkrementalnog razvoja i isporuke

- Kupac dobiva svoju vrijednost sa svakim inkrementom. Funkcionalnost sustava se ostvaruje u ranim fazama projekta.
- Rani inkrementi služe kao prototipovi na temelju kojih se izlučuju zahtjevi za kasnije inkremente.
- Manji rizik za neuspjeh projekta.
- Prioritetne funkcionalne usluge sustava imaju mogućnost detaljnijeg ispitivanja (testiranja).

Spiralni razvoj i oblikovanje

- 1988 Barry Boehm "A Spiral Model of Software Development and Enhancement"
- Proces se predstavlja spiralom umjesto sekvencom aktivnosti s povratima.
- Svaka petlja u spirali predstavlja fazu procesa.
- Nema fiksnih faza
  - petlje u spirali izabiru se prema potrebnim zahtjevima.
- Rizici razvoja programskog produkta eksplicitno se određuju i razrješuju.

Spiralni pristup



Postavljanje ciljeva

1. Determine objectives

Procjena i smanjivanje rizika

2. Identify and resolve risks

3. Development and Test

Razvoj i validacija

4. Plan the next iteration

Planiranje

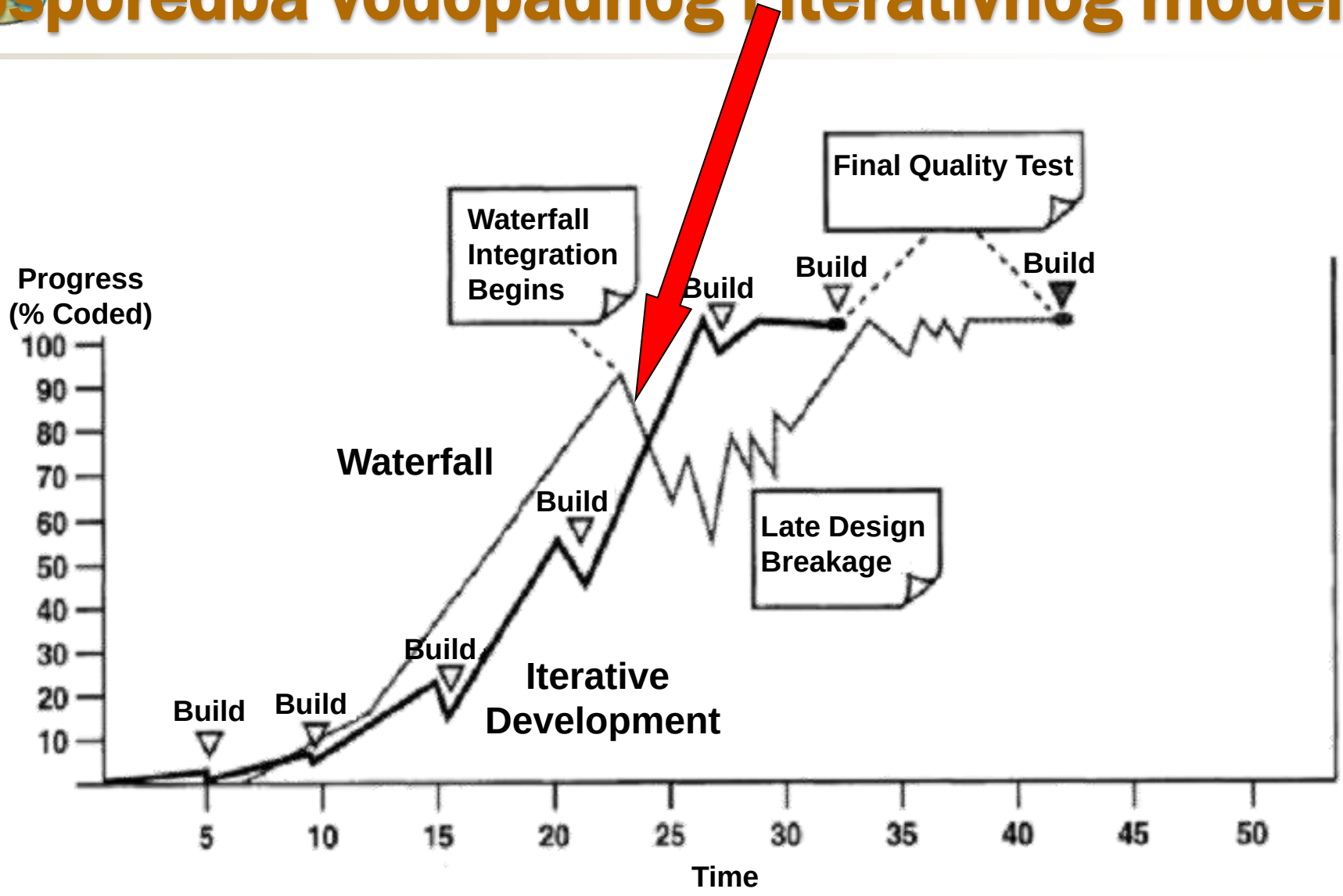
Sektori u spiralnom modelu

- Slijede vodopadni model
- Postavljanje ciljeva
  - identifikacija specifičnih ciljeva sektora.
- Procjena i smanjivanje rizika
  - procjenjuju se rizici i preslikavaju u aktivnosti koje ih reduciraju. (npr. SWOT analize)
- Razvoj i validacija
  - odabire se model razvoja i oblikovanja. To može biti bilo koji generički model.
  - u ranim spiralama koncept, kasnije spirale nose sve detaljnije aktivnosti (specifikacija, oblikovanje, razvoj, testiranje, uporaba).
- Planiranje
  - projekt se kritički ispituje (revidira) i planira se slijedeća spiralna faza.

Karakteristike spiralnog modela

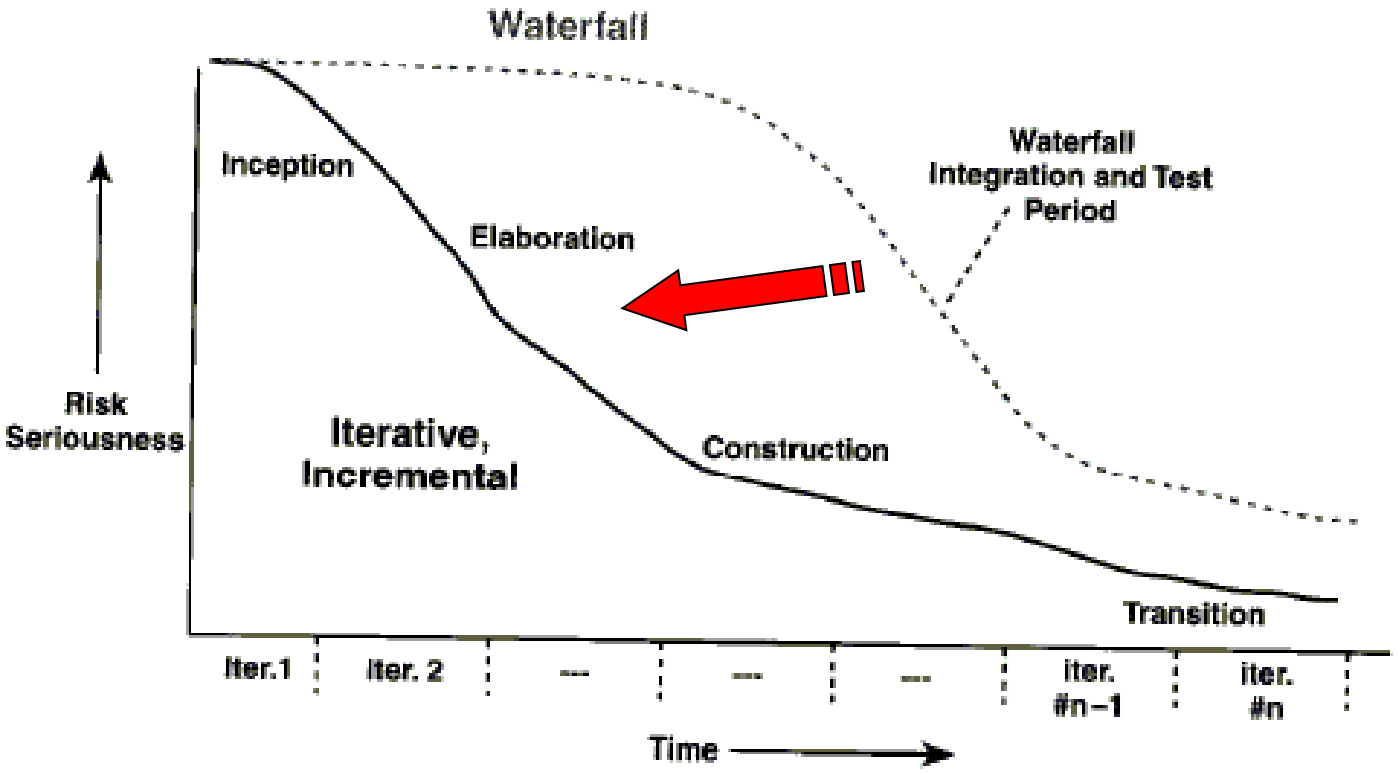
- Prednosti
  - odražava iterativnu prirodu razvoja programske podrške uzimajući u obzir nejasnoće zahtjeva
  - prilagodljivo obuhvaća prednosti vodopadnog modela i brze izrade prototipa
  - smanjuje rizik razvoja
  - preglednost projekta
- Nedostaci
  - složen, veliko administrativno opterećenje
  - zahtjeva poznavanje tehničke analize rizika
  - nerazumljiv netehničkom rukovodstvu

Usporedba vodopadnog i iterativnog modela



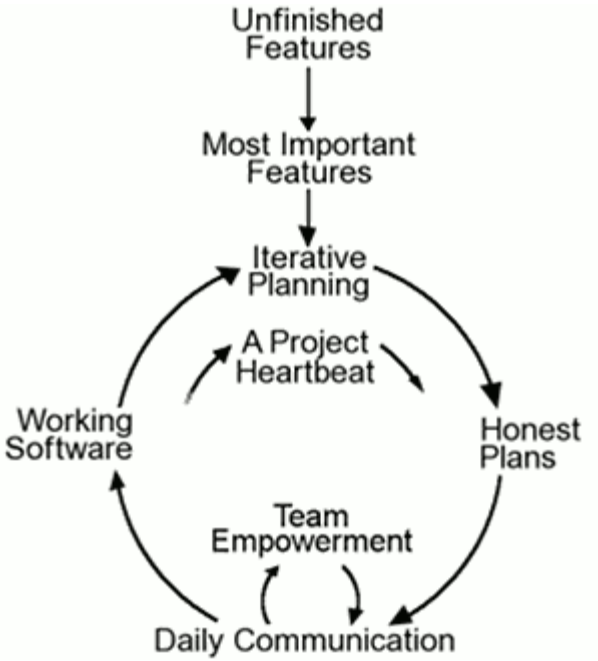
Izvor: I. Jacobson, G. Booch, J. Rumbaugh: The Unified Software Development Process, 1999

Upravljanje rizikom



Primjer: Ekstremno programiranje

- *engl. extreme programming - XP*
- Kao vrsta inkrementalnog postupka razvoja i isporuke
- Pristup se bazira na razvoju, oblikovanju i isporuci vrlo malih inkremenata funkcionalnosti.
- Kontinuirano poboljšanje koda
- Sudjelovanje korisnika u razvojnom timu
- Programiranju u paru
  - *engl. pairwise programming*
  - jedno radno mjesto, međusobno provjeravanje
- Http://www.extremeprogramming.org/
- Spada u ubrzane/agilne modele



Ubrzani modeli

- Još se naziva i žustri, *engl. Agile methods*
- Javlja se sredinom 1990-tih kao odgovor na probleme strukturiranih procesa programskog inženjerstva.
- Smatra se da vodopadni model unosi previše birokracije u proces oblikovanja.
- Jedina mjera napretka je funkcionalni programski produkt.
- Pristup se bazira na razvoju, oblikovanju i isporuci funkcionalnih dijelova.
- Postupak uključuje kontinuirano poboljšanje koda.
- Sudjelovanju korisnika u razvojnom timu.
- U razvoju sustava najčešće se programira u paru (jedno radno mjesto, međusobno provjeravanje (*engl. pairwise programming*).
- Nedostatak: nerazumljivost, ne podupire ponovnu uporabu rješenja.
- Http://agilemanifesto.org/, http://agilemanifesto.org/iso/hr/

GENERIČKE AKTIVNOSTI U PROCESU PROGRAMSKOG INŽENJERSTVA

SPECIFIKACIJA PROGRAMSKOG PRODUKTA

Specifikacija programskog produkta

- *"The hardest single part of building software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, machines, and to other SW systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."*
  - frederick Brooks, The Mythical Man-month essays on software engineering, Addison-Wesley,1987
- Proces određivanja potrebnih usluga i ograničenja u radu i razvoju sustava.
  - specifikacija programskog produkta određuje se procesom inženjerstva zahtjeva (*engl. Requirements engineering*).
  - proces rezultira dokumentom u kojem se navode potrebne usluge i ograničenja u radu i razvoju sustava.
- Proces inženjerstva zahtjeva (*engl. Requirements engineering*)
  - Studija izvedivosti
  - Izlučivanje i analiza zahtjeva
  - Specifikacija zahtjeva
  - Validacija zahtjeva

GENERIČKE AKTIVNOSTI U PROCESU PROGRAMSKOG INŽENJERSTVA

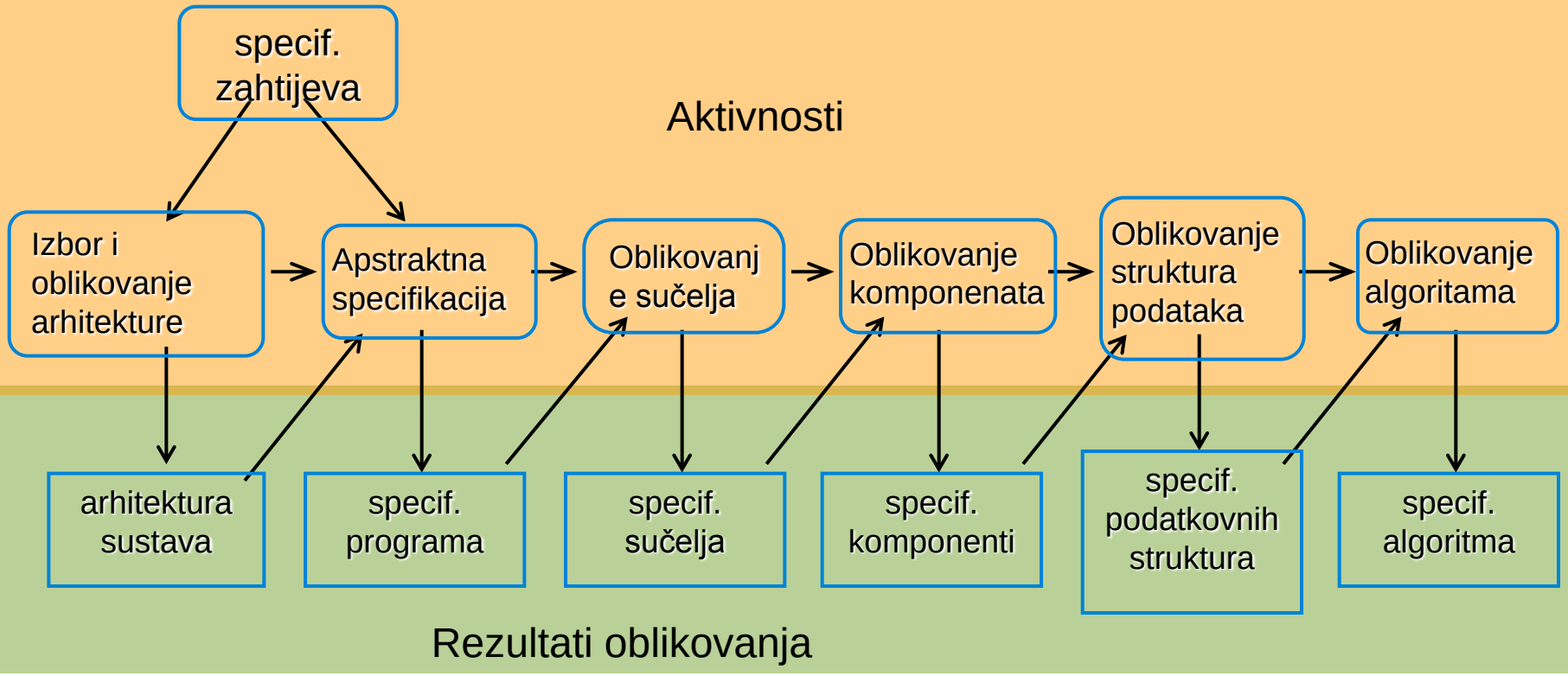
OBLIKOVANJE I IMPLEMENTACIJA PROGRAMSKOG PRODUKTA

Oblikovanje i implementacija programskog produkta

- Proces preslikavanja specifikacije u stvarni, realni sustav.
- ***Oblikovanje programske potpore***
  - oblikovanje strukture sustava koja realizira specifikaciju (izbor i modeliranje arhitekture).
- ***Implementacija***
  - preslikavanje strukture u izvršni program.
- Aktivnosti oblikovanja i implementacije su povezane i mogu biti isprepletene.

Aktivnosti procesa oblikovanja

- Izbor i oblikovanje arhitekture
- Apstraktna specifikacija
- Oblikovanje sučelja
- Oblikovanje komponenata
- Oblikovanje struktura podataka
- Oblikovanje algoritama



Izbor arhitekture

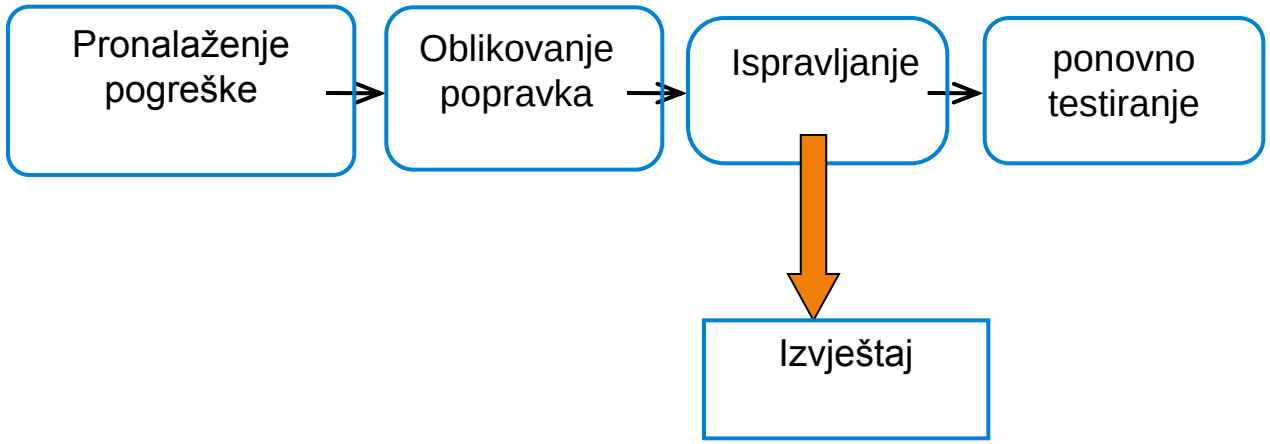
- Prva aktivnost
  - temelji se na neformalnoj analizi i primjeni dobre inžinjerske prakse
  - nema formalnog postupka
- Sistematski pristupi oblikovanja programskog produkta provodi se na temelju odabrane arhitekture.
- Arhitektura je dokumentirana skupom modela
  - najčešće grafički dijagrami.
- Tipovi arhitekture programske potpore:
  - protok podataka (*engl. data-flow*)
  - objektno usmjerena arhitektura
  - repozitorij podataka
  - …

Implementacija programskog produkta

- Implementacija
  - programiranje i otklanjanje pogrešaka
- Preslikavanje dokumentiranog oblikovanja u program i otklanjanje pogrešaka u programu.
- Programiranje je osobna aktivnost
  - nema generičkog proces programiranja.
- Programeri izvode neke aktivnosti ispitivanja (testiranja) s ciljem otkrivanja pogrešaka u programu i njihovog otklanjanja (*engl. debugging*), oblikovanje i programiranje ispitnih programa.

Proces otklanjanja pogrešaka

- U okviru implementacije – programiranja



- Ispitivanje integrirane i cjelovite programske potpore je posebna aktivnost koja ne spada u generičke aktivnosti implementacije!!!

GENERIČKE AKTIVNOSTI U PROCESU PROGRAMSKOG INŽENJERSTVA

VALIDACIJA I VERIFIKACIJA PROGRAMSKOG PRODUKTA

Validacija i verifikacija programskog produkta

- Potrebno pokazati da sustav odgovara specifikaciji i zadovoljava zahtjevima kupca i korisnika.
- Validation – “Are we building the right system?”
  - da li sustav zadovoljava funkcijske zahtjeve
  - provodi se ispitivanjem sustava
- Verification – “Are we building the system right?”
  - da li sustav zadovoljava zahtjeve na ispravan način
  - uključuje provjeru poželjno zasnovanu na formalnim (matematičkim i logičkim) metodama.

Ispitivanje programske potpore

- Ispitivanje je nezaobilazna aktivnost u procesima programskog inženjerstva.
  - ispitivanje se dopunjuje formalnom verifikacijom.
- Ispitivanje sustava temelji se na radu sustava s ispitnim ulaznim parametrima (podacima) koji se izvode iz specifikacije realnih podataka koje sustav treba prihvatiti.
  - skupo (resursi, vrijeme)
- Pretpostavka za ispitivanje:
  - bez specifikacije nema ispitivanja!
- Testiranje znači usporedbu stvarnih rezultata s postavljenim standardima.
- IEEE definicija:
  - test - jedan ili više testnih slučaja (*engl. Test case*)
  - testiranje - proces analize programskog koda sa svrhom pronalaska razlike između postojećeg i zahtijevanog stanja te vrednovanja svojstava programa.

Terminologija

- Testni podaci (I)
  - ulazi odabrani za provođenje određenog testa
- Očekivani izlaz (O)
  - zabilježen prije provođenje testa
- Testni scenarij (*engl. test case*)
  - uređeni par (I, O)
  - kako testirati modul, funkciju, ...
  - sadrži opis stanja prije testiranja, funkcije koja se testira
- Stvarni izlaz
  - rezultat dobiven provođenjem testa
- Kriterij prolaza testa
  - kriterij usporedbe očekivanog i stvarnog izlaza određen prije provođenja testa

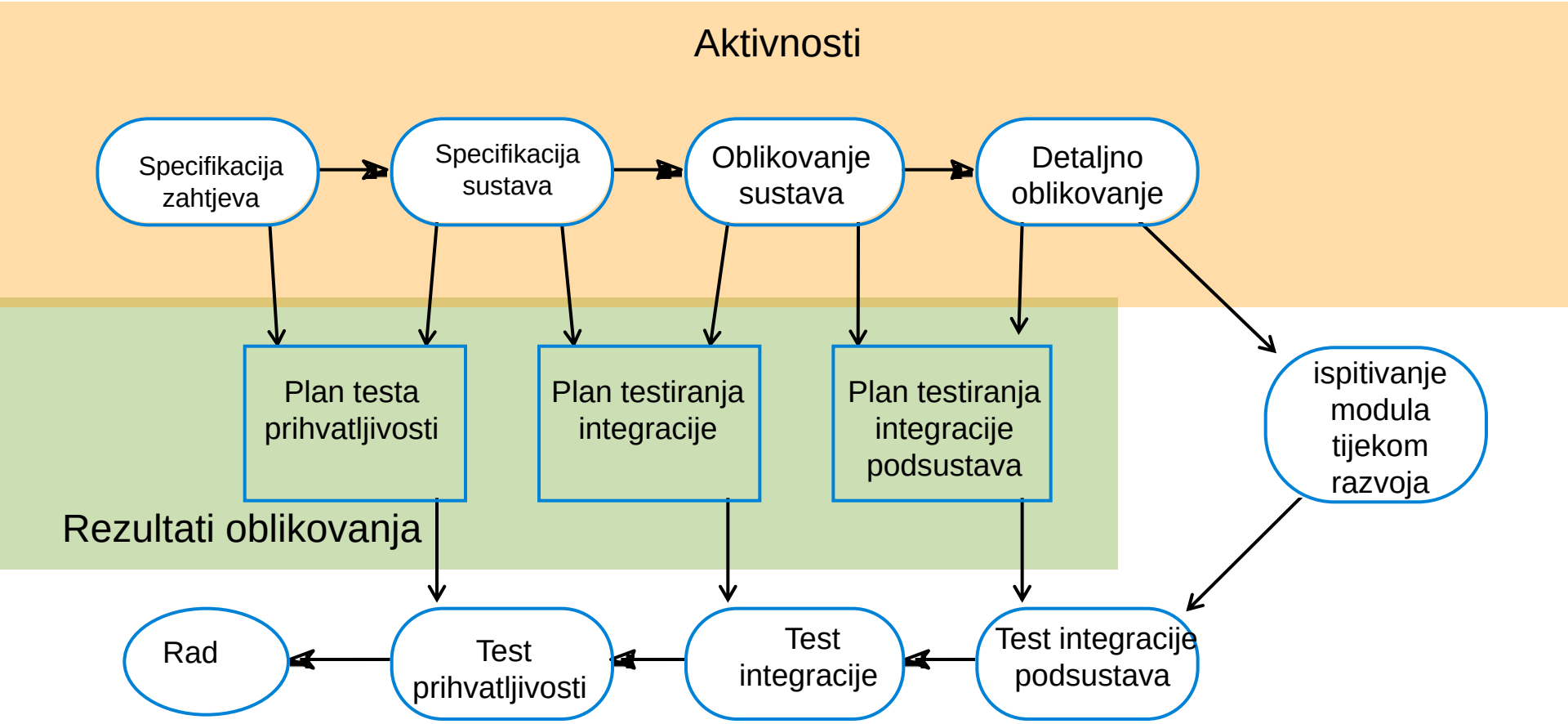
Proces ispitivanja sustava

- Ispitivanje ***komponenti i modula***
  - individualne komponente se ispituju nezavisno.
  - komponente mogu biti funkcije, objekti ili koherentne skupine tih entiteta.
- Ispitivanje ***sustava***
  - ispitivanje cjelovitog sustava. Posebice je važno ispitivanje novih i nenadanih svojstava.
- Ispitivanje ***prihvatljivosti***
  - značajki na temelju kojih kupac prihvaća i preuzima sustav (*engl. acceptance*).

Vrste ispitivanja programske potpore

- Ispitivanje komponenti, *engl. Unit (Component) testing*.
- Integracijsko ispitivanje, *engl. Integration testing.*
- Ispitivanje sustava, *engl. System/Release testing.*
- Test prihvatljivosti, *engl. Acceptance Testing.*
- Test instalacije, *engl. Installation testing.*
- Alpha test (interno).
- Beta test (vanjsko).

Aktivnosti i produkti procesa ispitivanja

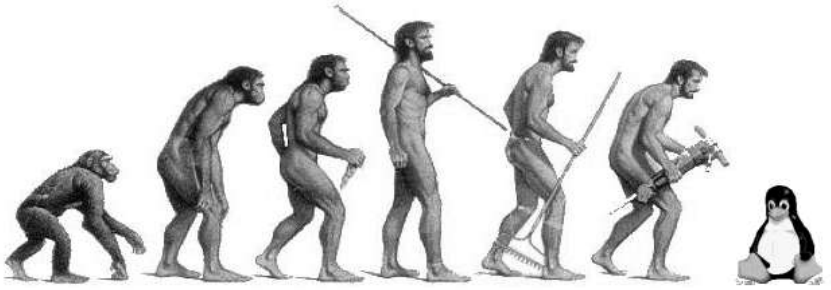


Status ispitivanja programske potpore

- Moguće pokazati postojanje ali ne i nepostojanje greške
- 1972 Dijkstra: “Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence”

GENERIČKE AKTIVNOSTI U PROCESU PROGRAMSKOG INŽENJERSTVA

EVOLUCIJA PROGRAMSKOG PRODUKTA



Evolucija programskog produkta

- Programski produkt je inherentno fleksibilan i može se mijenjati.
- Kako se mijenjaju zahtjevi na sustav (zbog promjene poslovnog procesa) programski produkt koji podupire poslovni proces također se mora mijenjati.
- Iako postoji granica između razvoja i oblikovanja s jedne strane i evolucije i održavanja s druge strane, to ona postaje sve više irelevantna, jer je sve manje sustava potpuno novo.

RAČUNALOM PODRŽANO PROGRAMSKO INŽENJERSTVO

Računalom podržano programsko inženjerstvo

- *engl. Computer-aided software engineering - CASE*
- Programski produkti namijenjeni podršci aktivnostima u procesu programskog inženjerstva (specifikacija, oblikovanje i evolucija)
  - izrada računalnog programa uz pomoć računala i prikladnih programskih alata
- Poboljšanje i kontrola procesa
- Alati za automatizaciju oblikovanja:
  - grafički editori za razvoj modela sustava.
  - rječnici i zbirke za upravljanje entitetima u oblikovanju.
  - grafičko okruženje za oblikovanje i konstrukciju korisničkih sučelja.
  - alati za pronalaženje pogrešaka u programu.
  - automatizirani prevoditelji koji generiraju nove inačice programa.

CASE tehnologija

- CASE tehnologija je dovela do značajnog unapređenja procesa oblikovanja programske potpore.
- Unapređenja i poboljšanja nisu sukladna očekivanjima (poboljšanje za red veličine?)
- Programsko inženjerstvo zahtijeva ***kreativnost*** koja nije automatizirana.
- Programsko inženjerstvo je ***timska aktivnost***. U većim projektima mnogo vremena se utroši na interakcije unutar tima.
  - CASE tehnologija slabo podupire timski rad.

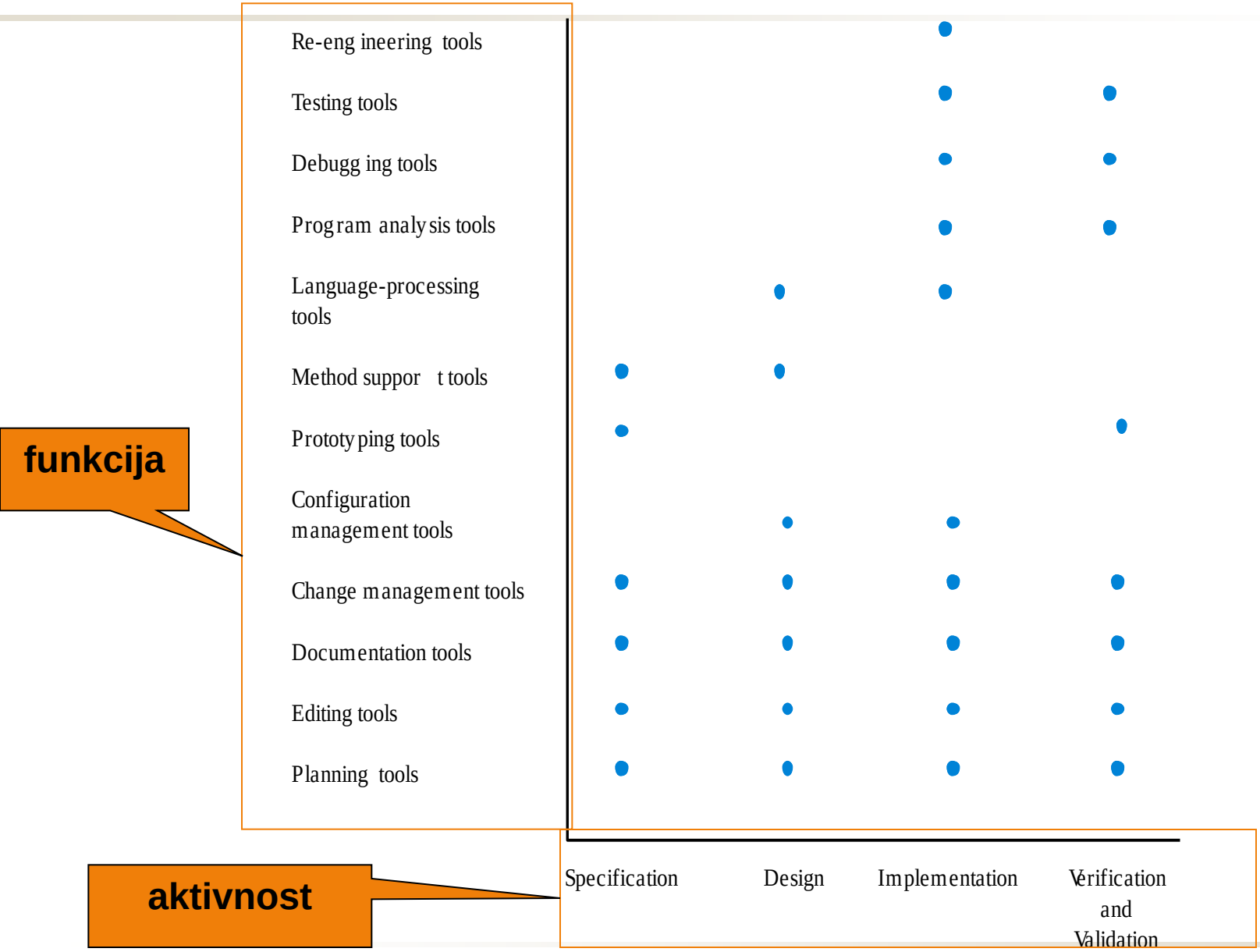
Klasifikacija CASE alata

- Klasifikacija omogućuje razumijevanje različitih tipova CASE alata i njihovu potporu aktivnostima u procesu programskog inženjerstva.
- ***Funkcionalna perspektiva***
  - alati se klasificiraju prema specifičnoj funkciji.
- ***Procesna perspektiva***
  - alati se klasificiraju prema aktivnostima koje podupiru u procesu.
- ***Integracijska perspektiva***
  - alati se klasificiraju prema njihovoj organizaciji u integrirane cjeline.

Funkcionalna klasifikacija CASE alata

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Klasifikacija CASE alata prema aktivnostima

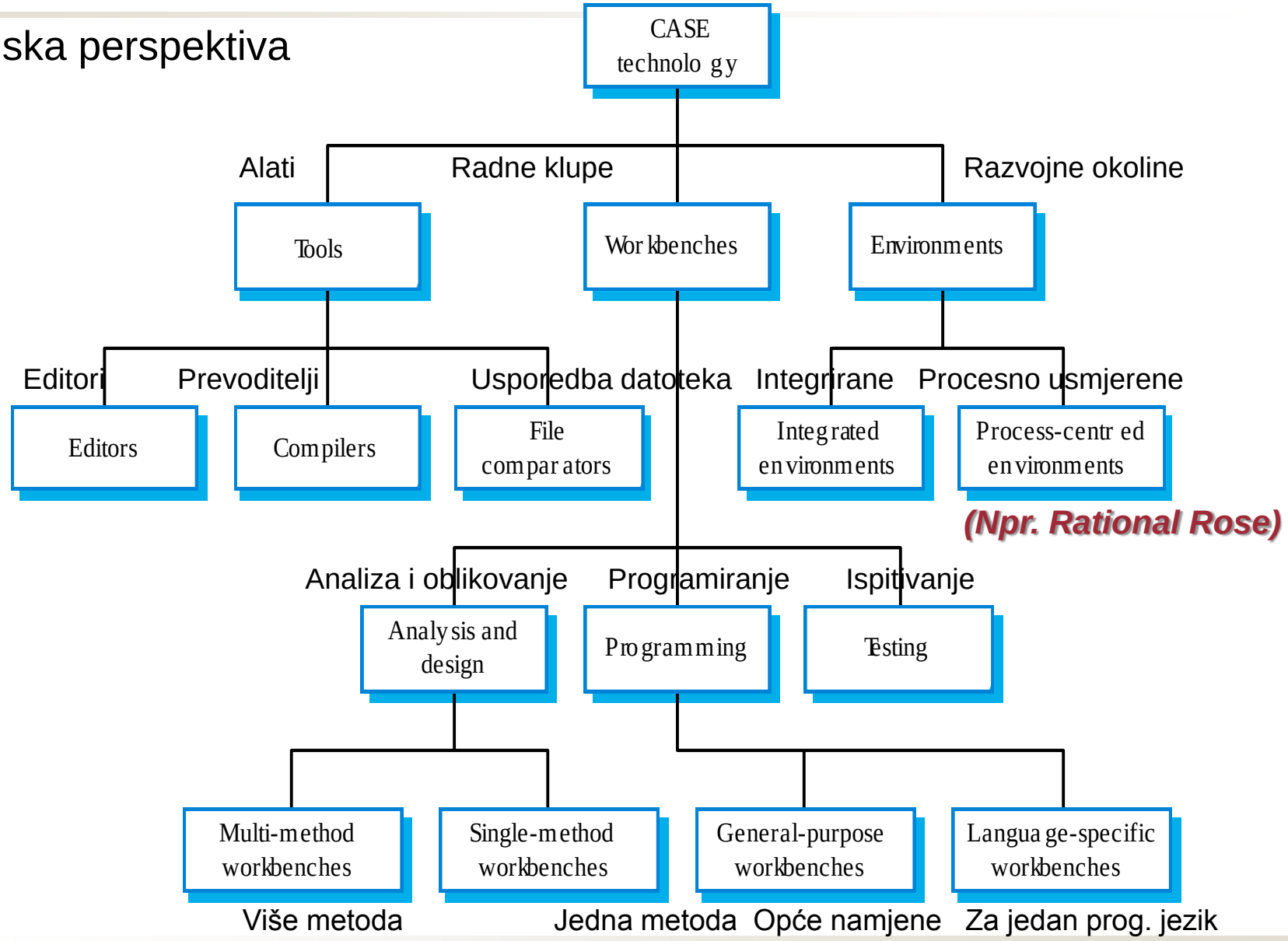


CASE integracija

- Alati
  - podupiru individualne zadatke u procesu (npr. oblikovanje, provjera konzist., editiranje teksta, …)
- Radne klupe (*engl. workbenches*)
  - podupiru pojedine faze procesa (npr. specifikaciju)
  - integriraju vise alata u jedinstvenu okolinu
- Razvojne okoline (*engl. environments*)
  - skupina alata i radih klupi
  - podupiru cijeli ili značajan dio procesa programskog inženjerstva. Uključuju nekoliko integriranih radnih klupa.

Alati, radne klupe, razvojne okoline

- Integracijska perspektiva



Prednosti CASE alata

- Veća produktivnost
- Bolja dokumentacija
- Veća točnost
- Poboljšana kvaliteta
- Smanjeni troškovi održavanja
- Utjecaj na organizaciju rada
- Nedostaci
  - velika cijena
  - vrijeme učenja
  - potreba za različitim alatima

CASE alati u okviru predmeta:

- Subversion – upravljanje konfiguracijama programske potpore.
- ArgoUML - oblikovanje objektno usmjerene arhitekture i generiranje koda.

- Radne klupe i razvojne okoline (temeljno usmjerene na dio procesa programskog inženjerstva tj. implementaciju i ispitivanje):
  - Visual Studio
  - Eclipse
  - NetBeans
  - ....

Zaključak

- ***Procesi u programskom inženjerstvu***
  - aktivnosti usmjerene na produkciju i evoluciju programskih proizvoda.
- ***Generičke aktivnosti***
  - specifikacija, oblikovanje i implementacija, validacija i evolucija.
- ***Modeli procesa***
  - apstraktna reprezentacija
- ***Generički modeli procesa***
  - opisuju njihovu organizaciju
  - npr.: vodopadni, evolucijski, komponentni, RUP.
- ***Iteracije***
  - opisuju proces kao ciklus različitih aktivnosti unutar generičkih modela procesa (različiti intenzitet)

- Nema univerzalnog i standardnog procesa za sve slučajeve!!

Zaključak

- ***Inženjerstvo zahtjeva*** je proces izrade specifikacije programskog produkta.
- ***Procesi oblikovanja i implementacije*** preslikavaju specifikaciju u arhitekturu i radni/izvršni program.
- ***Validacija i verifikacija*** provjerava da li sustav zadovoljava specifikaciju i potrebe korisnika.
- ***Evolucija*** se bavi modifikacijama sustava tijekom njegove uporabe.
- ***Unificirani proces*** (RUP)
  - generički model procesa koji odvaja aktivnosti od faza izvođenja.
- ***CASE*** tehnologija podupire aktivnosti tijekom procesa programskog inženjerstva.

Diskusija

Tipična distribucija pitanja za PI

- Business Practices & Eng. Economics (3-4%)
- Software Requirements (13-15%)
- Software Design (22-24%)
- Software Construction (10-12%)
- Software Testing (15-17%)
- Software Maintenance (3-5%)
- Software Configuration Management (3-4%)
- Software Engineering Management (10-12%)
- Software Engineering Process (2-4%)
- Software Tools and Methods (2-4%)
- Software Quality (6-8%)