PROCESI PROGRAMSKOG INŽENJERSTVA

Sadržaj ove prezentacije:

- Procesi i modeli procesa programskog inženjerstva.
- Iteracije u modelima procesa programskog inženjerstva.
- Generičke aktivnosti u procesima programskog inženjerstva.
- Rational Unified Process (RUP) Model.
- CASE tehnologije za potporu aktivnostima u procesima programskog inženjerstva.
- Zaključci

1

Procesi i modeli procesa programskog inženjerstva.

2

Proces programskog inženjerstva je strukturirani skup aktivnosti potreban da se razvije i oblikuje programski produkt. Generički skupovi **aktivnosti** su:

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

Model procesa programskog inženjerstva je apstraktna reprezentacija procesa. Predstavlja opis procesa iz određene perspektive.

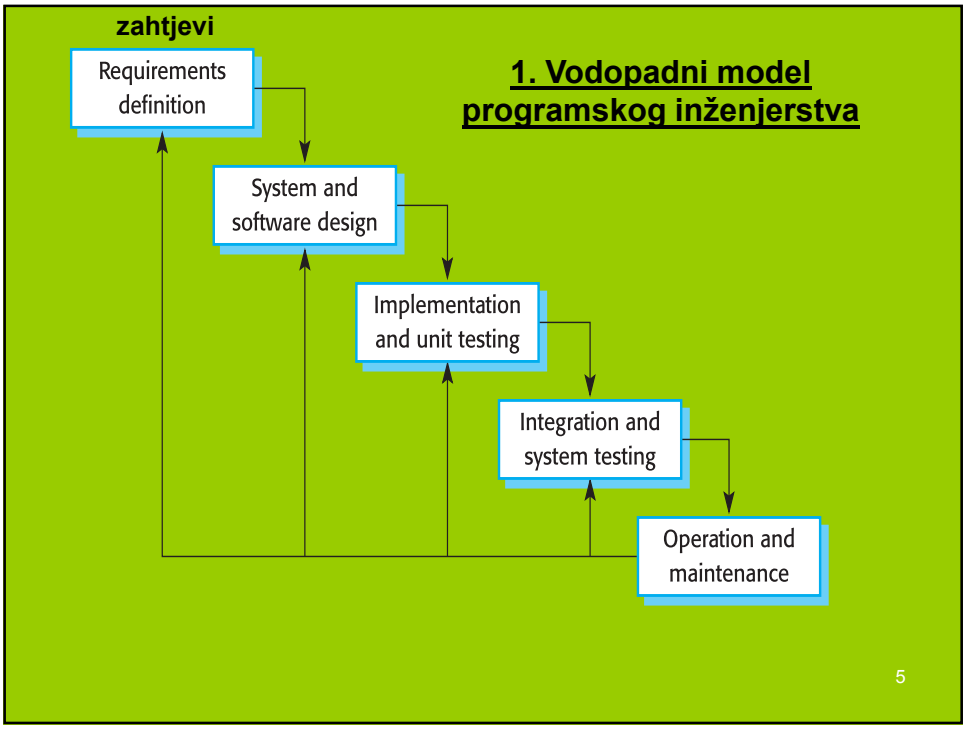
3

GENERIČKI MODELI PROCESA PROGRAMSKOG INŽENJERSTVA

1. **Vodopadni**
   - Odvojene i specifične faze specifikacije i razvoja
2. **Evolucijski**
   - Specifikacija, razvoj i validacija su isprepleteni
3. **Komponentno usmjeren**
   - Sustav se gradi od postojećih komponenata

Postoje mnoge varijacije ovih generičkih modela (npr. model u kojem je vodopadni slijed aktivnosti ali se preslikava formalna specifikacija do implementacije), t.j. aktivnosti su matematička preslikavanja.

4



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

Temeljna značajka vodopadnog modela:

Pojedina faza se mora dovršiti prije pokretanja nove faze.

6

PROBLEMI VODOPADNOG MODELA

- Nefleksibilna particija projekta u odvojene dijelove čini implementaciju promjena koje zahtijeva kupac vrlo teškim (temeljni nedostatak vodopadnog modela).
- 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 nekoliko odvojenih mjesta.

7

2. EVOLUCIJSKI MODEL RAZVOJA I OBLIKOVANJA

Dva uobičajena postupka:

- **Istraživački razvoj i oblikovanje**

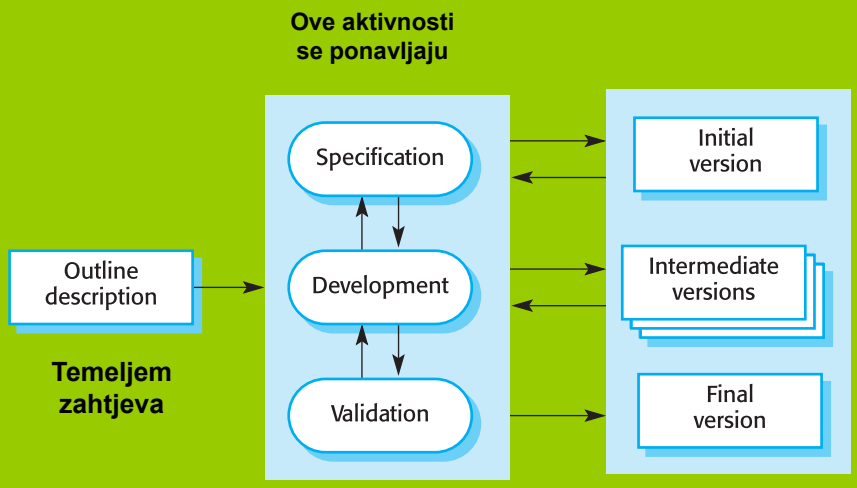
  Cilj ovakvog pristupa je kontinuiran rad s kupcem na temelju inicijalne specifikacije. Započinje se s dobro definiranim početnim zahtjevima a nove funkcionalnosti se dodaju temeljem prijedloga kupca.

- **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.

8

EVOLUCIJSKI MODEL RAZVOJA I OBLIKOVANJA



9

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.**

10

3. MODEL PROGRAMSKOG INŽENJERSTVA ZASNOVAN NA KOMPONENTAMA (CBSE)

Sustav se integrira višestrukom uporabom postojećih komponenata ili uporabom komercijalnih, gotovih komponenata (COTS: *engl. commercial-of-the-shelf*).

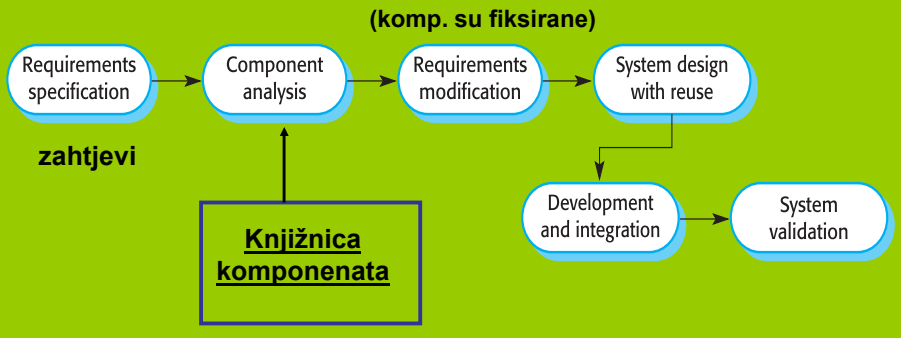
Stupnjevi procesa:

- Specifikacija i analiza zahtjeva
- Analiza komponenata
- Modifikacija zahtjeva (jer su komponente fiksirane)
- Oblikovanje sustava s višestrukom uporabom komponenata (*engl. reuse*)
- Razvoj i integracija

S povećanom standardizacijom komponenata CBSE se sve više prihvaća u razvoju sustava.

11

MODEL PROGRAMSKOG INŽENJERSTVA TEMELJEN NA VIŠESTRUKOJ UPORABI KOMPONENATA



12

Iteracije u modelima procesa programskog inženjerstva.

13

ITERACIJE U MODELIMA PROCESA PROGRAMSKOG INŽENJERSTVA

Zahtjevi na sustav evoluiraju i uvijek idu u smjeru s projektom. 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:

- Inkrementalna isporuka sustava
- Spiralni razvoj i oblikovanje

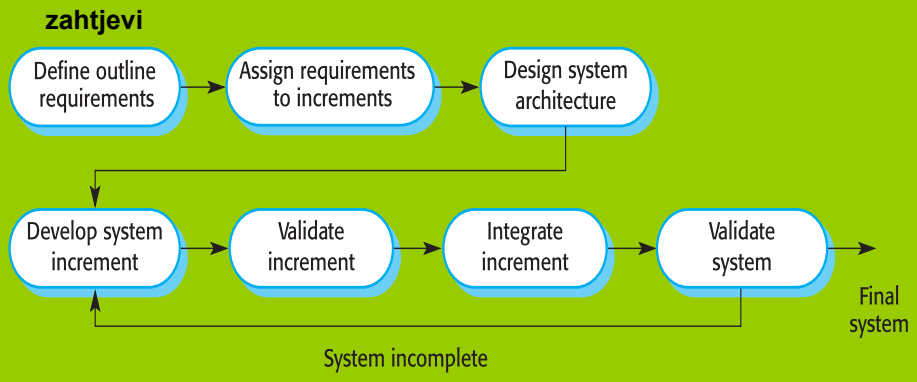
14

INKREMENTALNA ISPORUKA SUSTAVA

- 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 ostale kasnije inkremente nastavljaju evoluirati.

15

INKREMENTALNI ITERACIJSKI PRISTUP



16

PREDNOSTI INKREMENTALNOG PRISTUPA

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

17

EKSTREMNO PROGRAMIRANJE (XP) – kao poseban oblik inkrementalnog pristupa

- Pristup se bazira na razvoju, oblikovanju i isporuci vrlo malih funkcionalnih inkremenata.
- Postupak uključuje kontinuirano poboljšanje koda.
- Sudjelovanju korisnika u razvojnom timu.
- Programiranju u paru (jedno radno mjesto, međusobno provjeravanje (*engl. pairwise programming*).

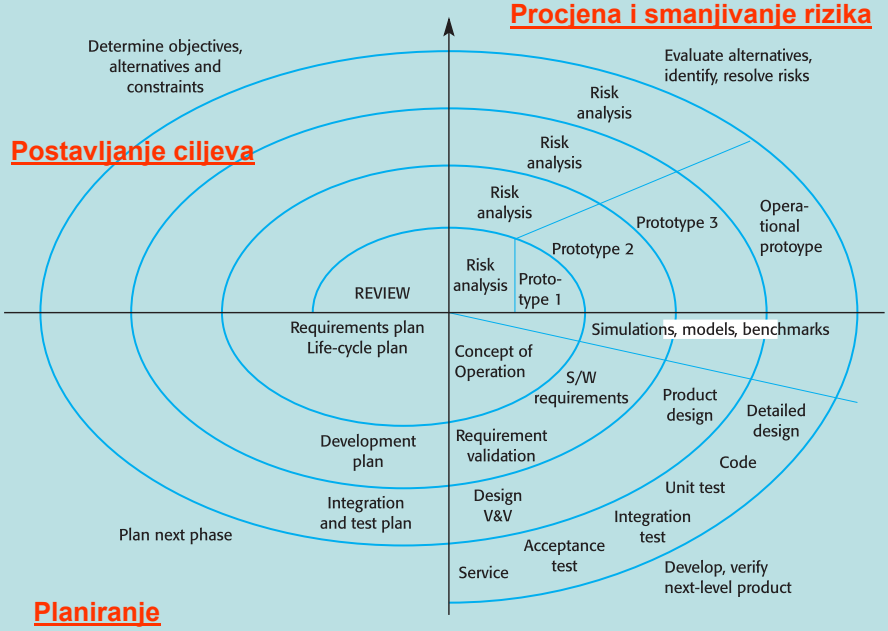
18

SPIRALNI RAZVOJ I OBLIKOVANJE

kao iterativni pristup

- **Proces oblikovanja se predstavlja spiralom umjesto sekvencom aktivnosti s povratima.**
- **Svaka petlja u spirali predstavlja jednu fazu procesa.**
- **Nema ranih završenih faza (kao npr. specifikacija, razvoj, …); petlje u spirali izabiru se prema potrebnim zahtjevima.**
- **Rizici razvoja programskog produkta spiralnim pristupom eksplicitno se određuju i razrješavaju.**

19



20

SEKTORI U SPIRALNOM MODELU

- **Postavljanje ciljeva**

  Identifikacija specifičnih ciljeva (opcije i ograničenja) temeljem zahtjeva.

- **Procjena i smanjivanje rizika**

  Procjenjuju se opcije i rizici te preslikavaju u aktivnosti koje ih reduciraju (poput SWOT analize – snage, slabosti, prilike, prijetnje).

- **Razvoj i validacija**

  Odabire se model razvoja i oblikovanja. To može biti bilo koji generički model.

- **Planiranje**

  Projekt se kritički ispituje (revidira) i planira se slijedeća spiralna faza.

21

Generičke aktivnosti u procesima programskog inženjerstva

(detaljnije razmatranje)

22

GENERIČKE AKTIVNOSTI U PROCESU PROGRAMSKOG INŽENJERSTVA

- Specifikacija programskog produkta
- Oblikovanje i implementacija programskog produkta
- Validacija i verifikacija programskog produkta
- Evolucija programskog produkta

23

Aktivnost: SPECIFIKACIJA PROGRAMSKOG PRODUKTA

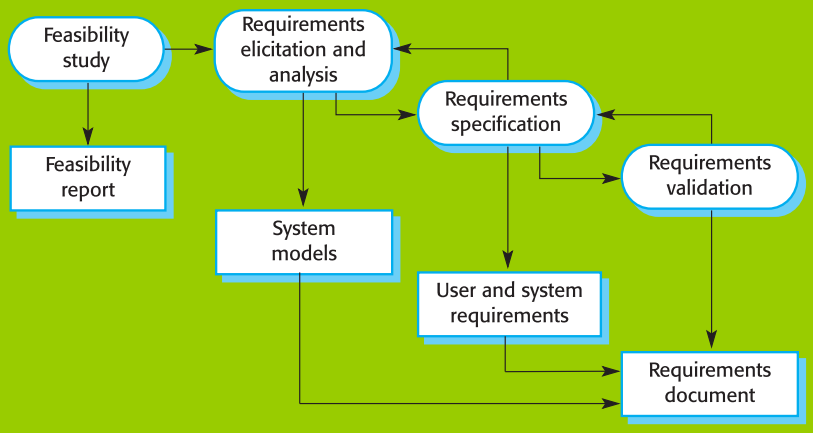
- Proces određivanja potrebnih usluga i ograničenja u radu i razvoju sustava.

Temeljni dio specifikacije čini:

- Proces inženjerstva zahtjeva *(engl. Requirements engineering)*, koji uključuje:
  - Studiju izvedivosti *(engl. Feasibility study)*
  - Izlučivanje i analizu zahtjeva *(engl. Requirements elicitation and analysis)*
  - Specifikaciju zahtjeva
  - Validaciju zahtjeva

24

PROCES INŽENJERSTAV ZAHTJEVA



25

Aktivnost: OBLIKOVANJE I IMPLEMENTACIJA PROGRAMSKOG PRODUKTA

To je proces preslikavanja specifikacije u stvarni, realni sustav. Uključuje:

- **Oblikovanje programskog produkta**

  Oblikovanje strukture sustava koja realizira specifikaciju (izbor arhitekture).

- **Implementaciju**

  Preslikavanje strukture u izvršni program.

Aktivnosti oblikovanja i implementacije su povezane i mogu biti isprepletene.

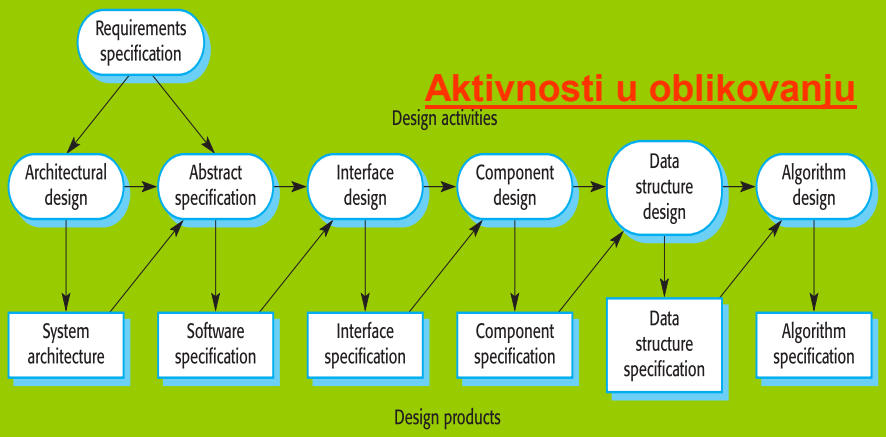
26

PODAKTIVNOSTI UNUTAR OBLIKOVANJA PROGRAMSKOH PRODUKTA

- Izbor i oblikovanje arhitekture
- Apstraktna specifikacija (koncepcijska, ne tehnička, iz nje slijedi konkretna specifikacija)
- Oblikovanje sučelja
- Oblikovanje komponenata
- Oblikovanje struktura podataka
- Oblikovanje algoritama

27

OBLIKOVANJE PROGRAMSKOG PRODUKTA
(podaktivnosti i rezultati – dokumenti)



28

OBLIKOVANJE PROGRAMSKOG PRODUKTA - (izbor arhitekture)

- Prva aktivnost unutar oblikovanja (vidi prethodnu sliku).
- Na temelju odabrane arhitekture slijedi sistematski pristupi oblikovanja programskog produkta.
- Arhitektura je dokumentirana skupom modela (najčešće grafičkih dijagrama).
- Neke moguće arhitekture programske potpore:

  Protok podataka (*engl.* data-flow)

  Objektno usmjerena arhitektura

  Repozitorij podataka

  …

29

OBLIKOVANJE PROGRAMSKOG PRODUKTA – Implementacija

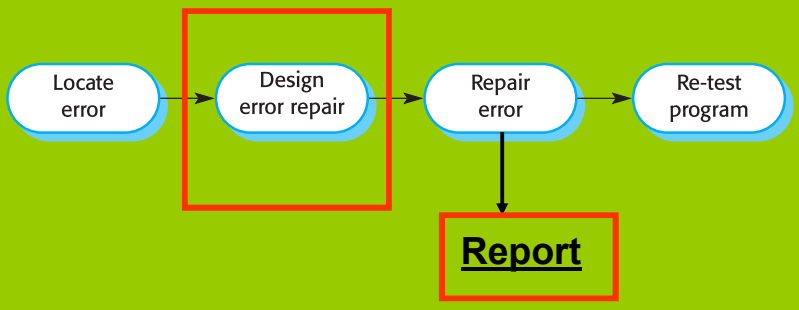
(programiranje i otklanjanje pogrešaka)

- Preslikavanje dokumentiranog oblikovanja (odabrane arhitekture) u program i otklanjanje pogrešaka u dijelu programa za koji su zaduženi.
- Programiranje je osobna aktivnost – ne postoji generički proces programiranja.
- Programeri izvode i neke dodatne aktivnosti: ispitivanje (testiranje) cjelokupnog programskog produkta s ciljem otkrivanja i otklanjanja pogrešaka (*engl. debugging*), oblikovanje i programiranje ispitnih programa.

30

PROCES OTKLANJANJA POGREŠAKA

(u okviru implementacije - programiranja)



31

Aktivnost: VALIDACIJA I VERIFIKACIJA PROGRAMSKOG PRODUKTA

- **Validacija i verifikacija** treba pokazati da sustav odgovara specifikaciji i zadovoljava zahtjevima kupca i korisnika.

  Validation – "Are we building the right system ?"

  Verification – "Are we building the system right ?"

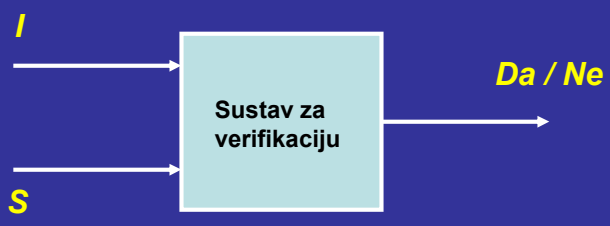
- Uključuje provjeru (poželjno formalnu – matematički zasnovanu) i reviziju procesa te ispitivanje (testiranje) sustava.
- **Ispitivanje sustava** temelji se na radu sustava s ispitnim ulaznim parametrima (podacima) koji se izvode iz specifikacije realnih podataka koje sustav treba prihvatiti.

32

Formalna verifikacija

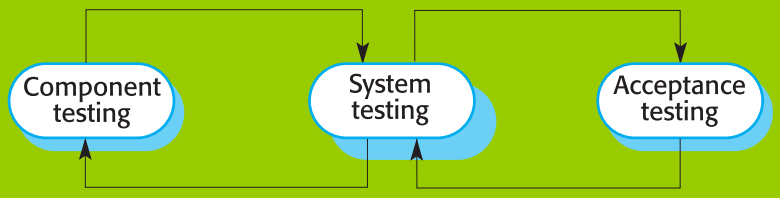
postupak provjere da formalni model izvedenog sustava (*I*), odgovara formalnoj specifikaciji (*S*)

sa matematičkom izvjesnošću



33

PROCES ISPITIVANJA SUSTAVA



34

STUPNJEVI U ISPITIVANJU (TESTIRANJU)

- Ispitivanje **komponenti i modula**

  Individualne komponente se ispituju nezavisno.

  Komponente mogu biti funkcije ili objekti ili koherentne skupine tih entiteta.

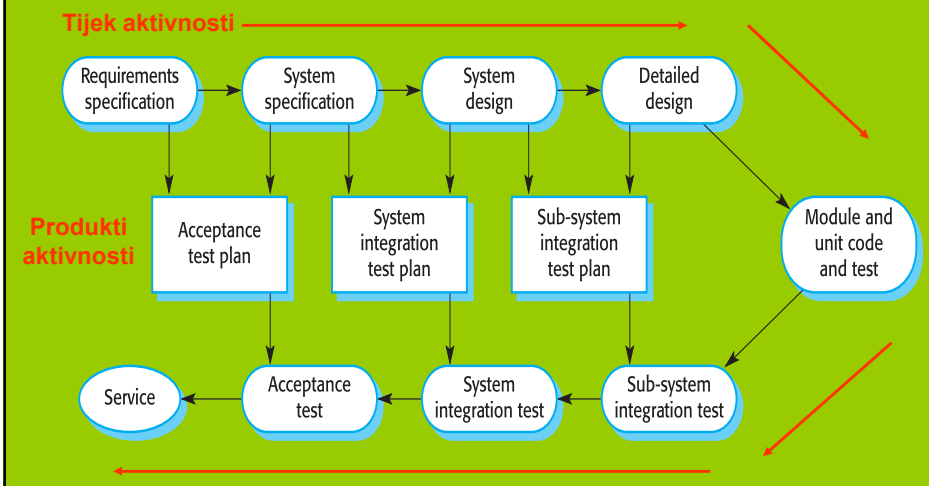
- Ispitivanje **sustava**

  Ispitivanje cjelovitog sustava. Posebice je važno ispitivanje novih i nenadanih svojstava.

- Ispitivanje **značajki na temelju kojih kupac prihvaća i preuzima sustav** (*engl. acceptance*).

35

FAZE U ISPITIVANJU (TESTIRANJU)



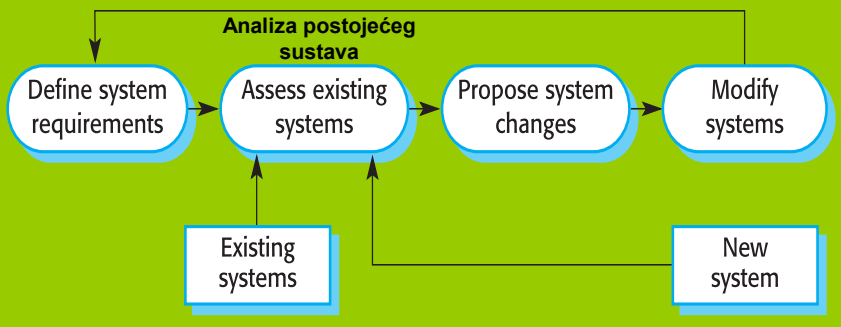
36

Aktivnost: 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.

37

EVOLUCIJA SUSTAVA



38

Rational Unified Process (RUP)

Modeli procesa programskog inženjerstva:

1. Vodopadni
2. Evolucijski
3. Komponentni
4. RUP

U svakom modelu se pojavljuju generičke aktivnosti i iteracije.

39

Posebno istaknuti model procesa programskog inženjerstva: RATIONAL UNIFIED PROCESS (RUP)

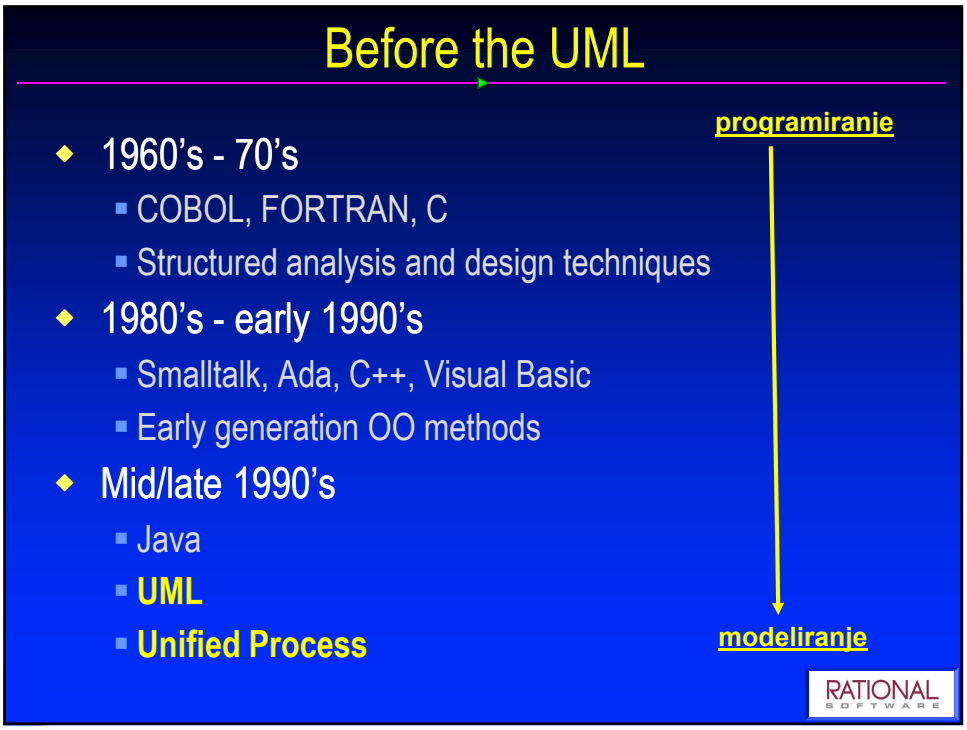
Moderan model procesa programskog inženjerstva izveden na temelju jezika za modeliranje UML-a (*engl. Unified Modeling Language*) i pridruženih aktivnosti.

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.

40


Preuzeto od:
Applying UML in
The Unified Process
Ivar Jacobson
Rational Software
email: ivar @rational.com


Before the UML
programiranje
1960's - 70's
COBOL, FORTRAN, C
Structured analysis and design techniques
1980's - early 1990's
Smalltalk, Ada, C++, Visual Basic
Early generation OO methods
Mid/late 1990's
Java
UML
Unified Process
modeliranje
RATIONAL
SOFTWARE

Overview of the UML

- The UML is a language for
  - visualizing
  - specifying
  - constructing
  - documenting

the artifacts of a software-intensive system

- **UML je posebice prikladan za specificiranje objektno usmjerene arhitekture programske potpore.**
- **Dijelovi UML-a pogodni su u specificiranju i drugih arhitektura.**
- **Arhitektura programske potpore je struktura ili strukture sustava koji sadrži elemente, njihova izvana vidljiva obilježja i odnose između njih.**

RATIONAL SOFTWARE

The Value of the UML

- Is an open standard
- Supports the entire software development lifecycle
- Supports diverse applications areas
- Is based on experience and needs of the user community
- Supported by many tools

RATIONAL SOFTWARE

Many stakeholders, many views

- Architecture is many things to many different interested parties
  - end-user
  - customer
  - project manager
  - system engineer
  - developer
  - architect
  - maintainer
  - other developers
- Multidimensional reality
- Multiple stakeholders

multiple views, multiple blueprints

RATIONAL SOFTWARE

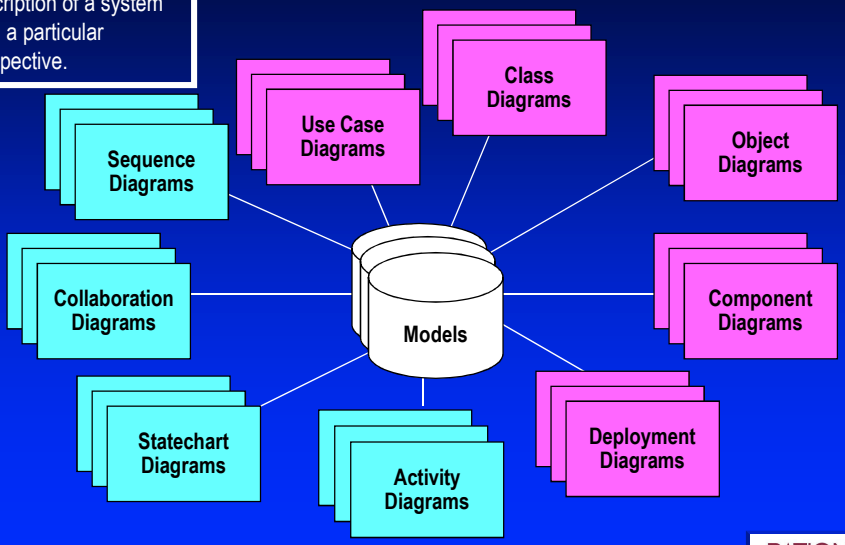
How many views?

- Simplified models to fit the context
- Not all systems require all views:
  - Single processor: drop deployment view
  - Single process: drop process view
  - Very Small program: drop implementation view
- Adding views:
  - Data view, security view

RATIONAL SOFTWARE

UML Models and Diagrams

A ***model*** is a complete description of a system from a particular perspective.

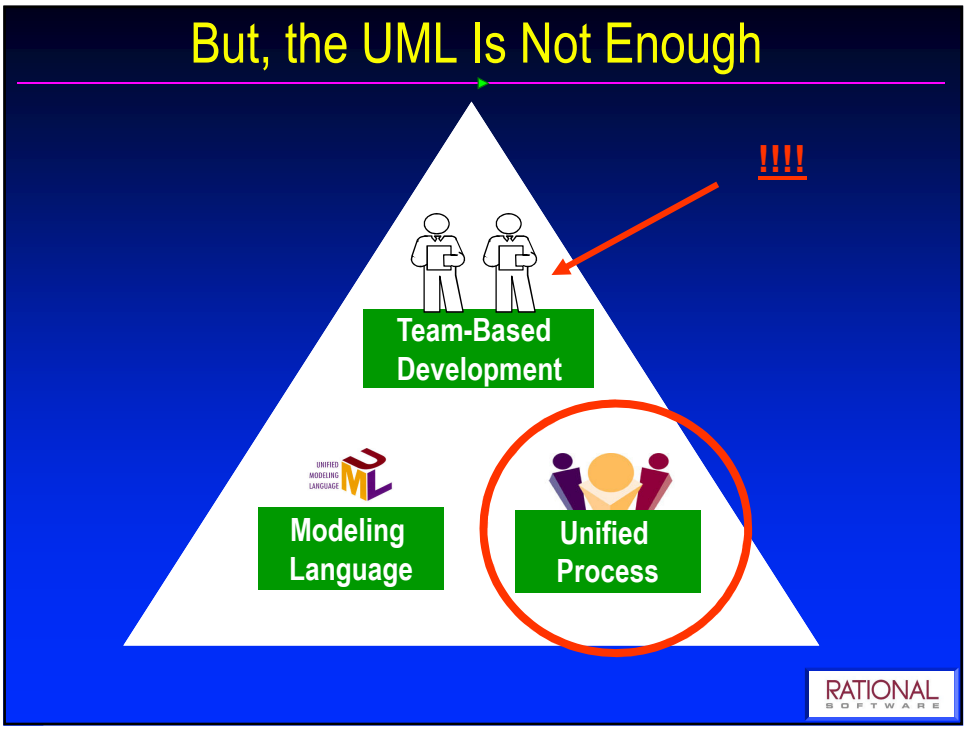


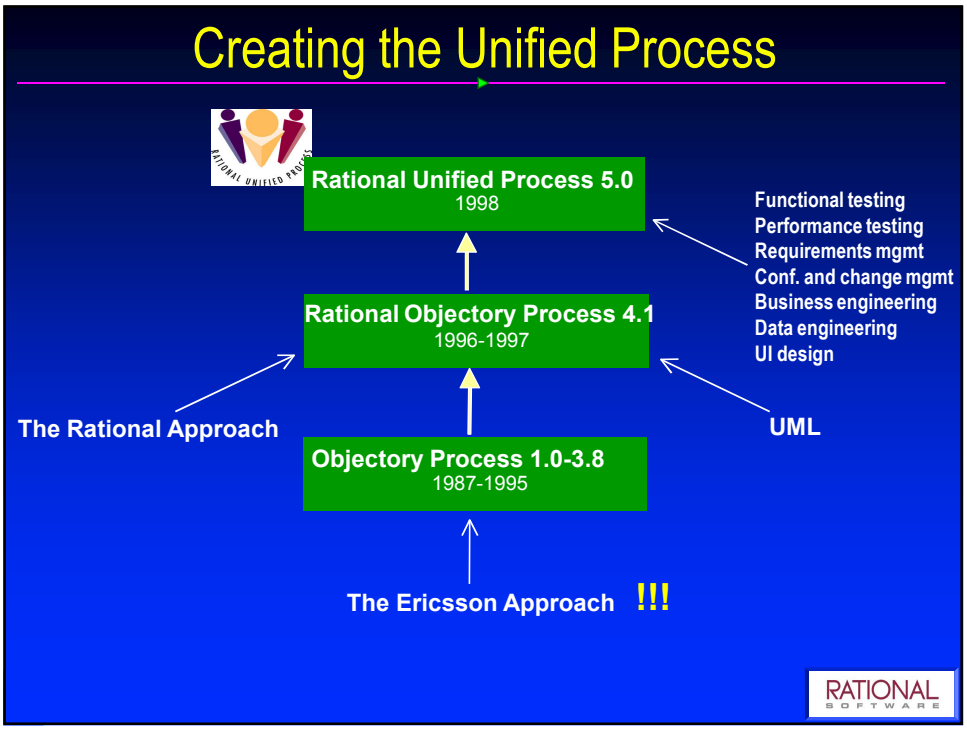
RATIONAL SOFTWARE

Diagrams

- A diagram is a view into a model
  - Presented from the aspect of a particular stakeholder
  - Provides a partial representation of the system
  - Is semantically consistent with other views
- In the UML, there are nine standard diagrams
  - **Static views:** use case, class, object, component, deployment
  - **Dynamic views:** sequence, collaboration, statechart, activity
  - **Svakoj aktivnosti u procesu programskog inženjerstva pridružen je jedan ili više modela za opis, koji su dokumentirani s jednim ili više dijagrama (pogleda,** *engl. views*).

RATIONAL SOFTWARE


But, the UML Is Not Enough
!!!!
Team-Based Development
Modeling Language
Unified Process
RATIONAL SOFTWARE


Creating the Unified Process
Rational Unified Process 5.0
1998
Rational Objectory Process 4.1
1996-1997
Objectory Process 1.0-3.8
1987-1995
The Rational Approach
The Ericsson Approach !!!
Functional testing
Performance testing
Requirements mgmt
Conf. and change mgmt
Business engineering
Data engineering
UI design
UML
RATIONAL SOFTWARE

What Is a Process?

- Defines Who is doing What, When to do it, and How to reach a certain goal.

RUP

RATIONAL SOFTWARE

Overview of the Unified Process

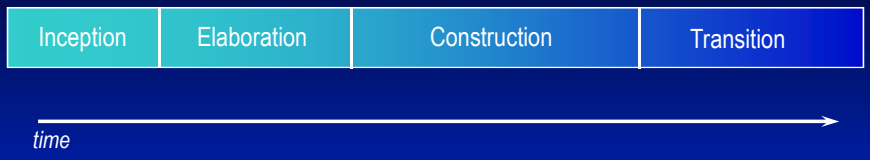
- The Unified Process is
  - Iterative and incremental
  - Use case driven → **Obrazac uporabe**
  - Architecture-centric

Obrasci uporabe temeljeni su na ideji scenarija. Skup obrazaca uporabe opisuje sve moguće interakcije sustava s okolinom (korisnicima) ili drugim sustavima, t.j. što sustav treba raditi (a ne kako to ostvariti).

RATIONAL SOFTWARE

RUP Lifecycle **Phases**

(početak)



- Inception — Define the scope of the project and develop business case
- Elaboration — Plan project, specify features, and baseline the architecture
- Construction — Build the product
- Transition — Transition the product to its users

RATIONAL SOFTWARE

Rekapitulacija: RUP FAZE

- **Početak** (*engl. inception*)

  Odredi poslovni model sustava.

- **Elaboracija**

  Oblikovanje razumijevanja domene primjene i određivanje temeljne arhitekture sustava.

- **Konstrukcija**

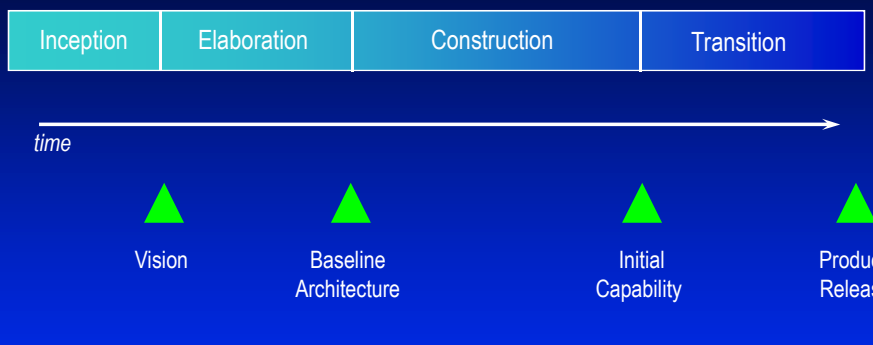
  Oblikovanje sustava, programiranje i ispitivanje.

- **Tranzicija**

  Postavljanje sustava u operativnu okolinu.

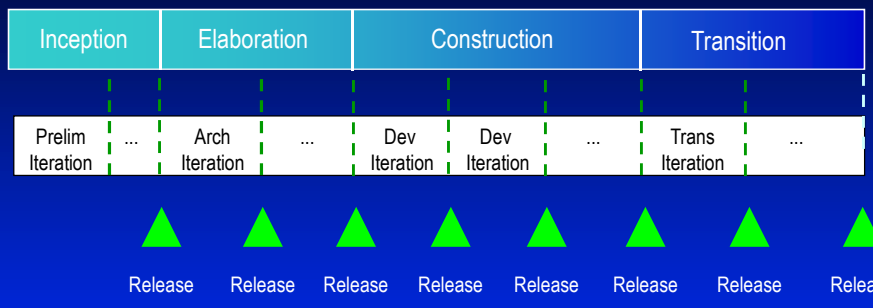
RATIONAL SOFTWARE

Major Milestones (ključne točke - dokumenti)



RATIONAL SOFTWARE

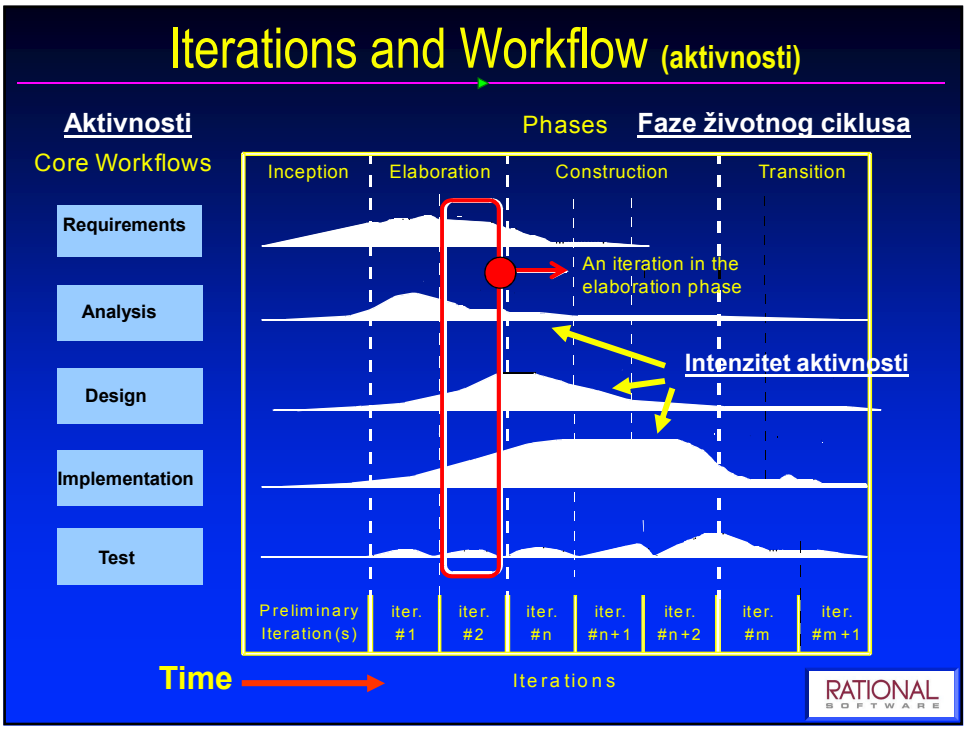
Phases and Iterations

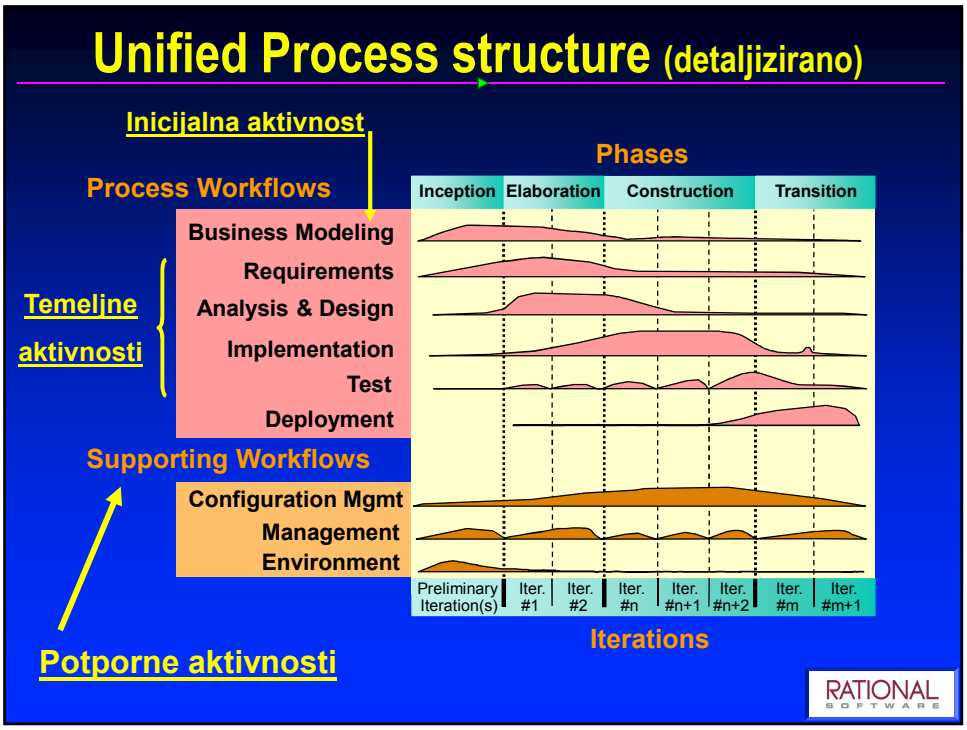


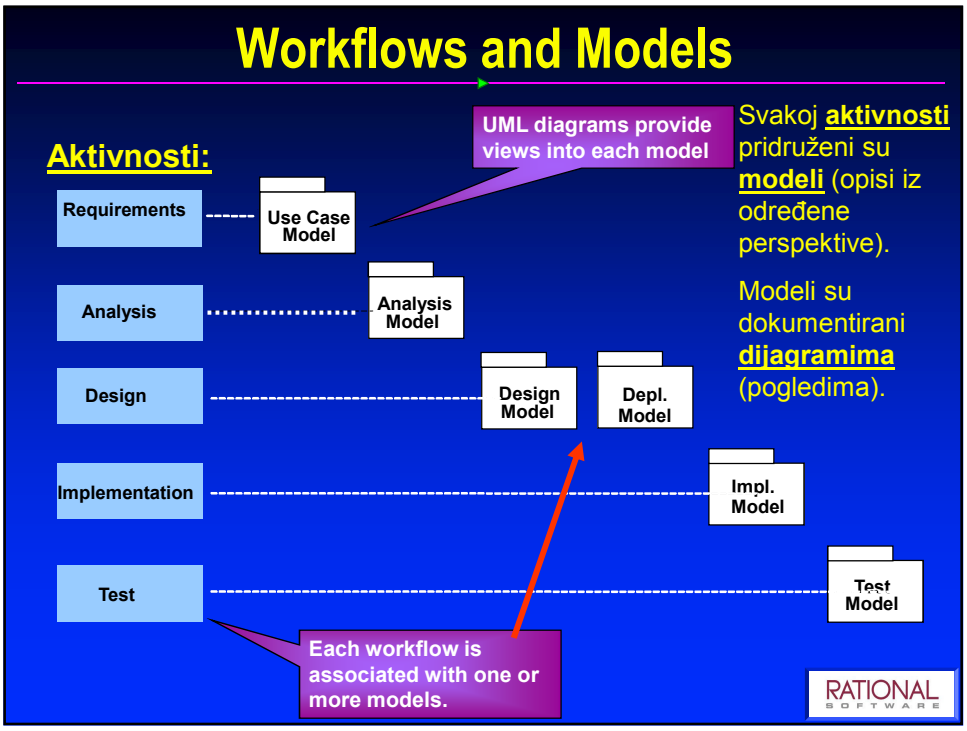
An iteration is a sequence of activities with an established plan and evaluation criteria, resulting in an executable release

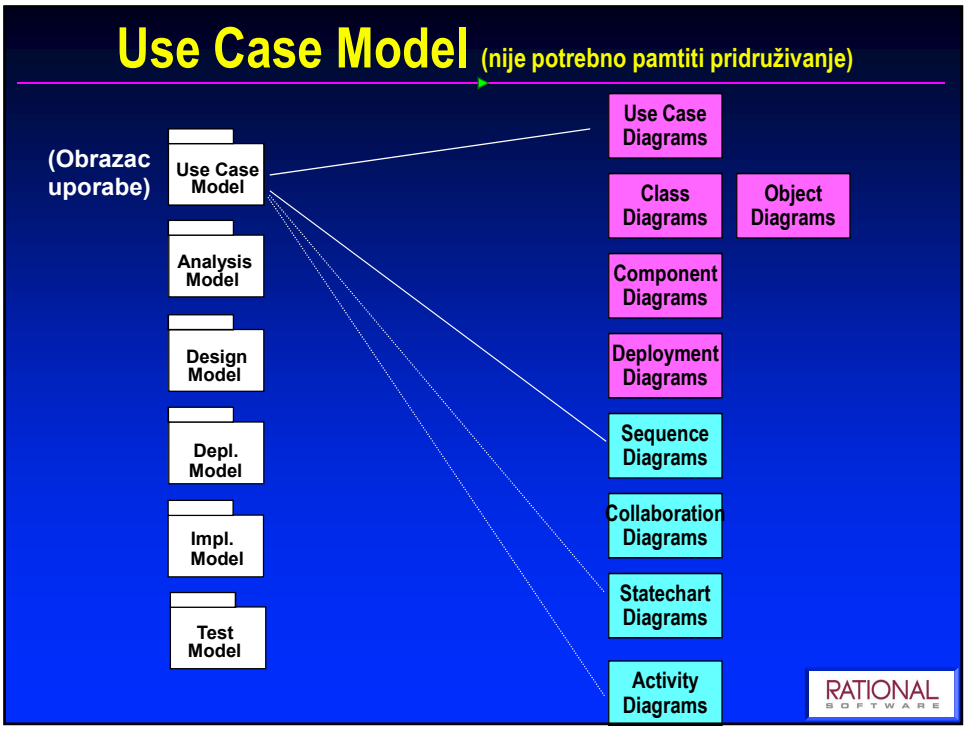
(izdanje)

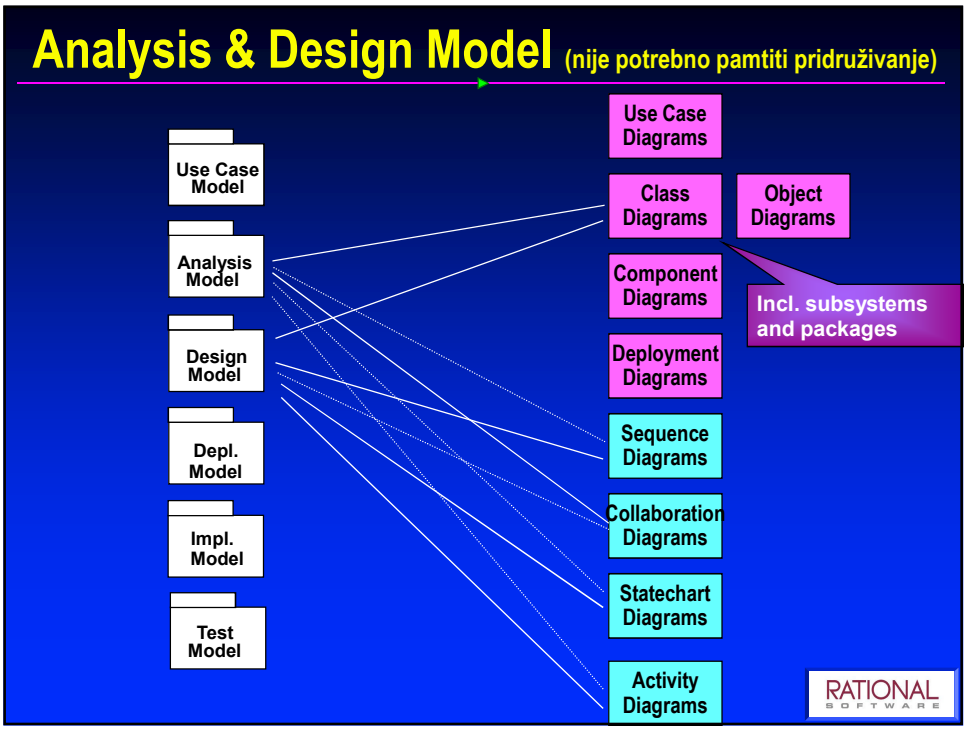
RATIONAL SOFTWARE

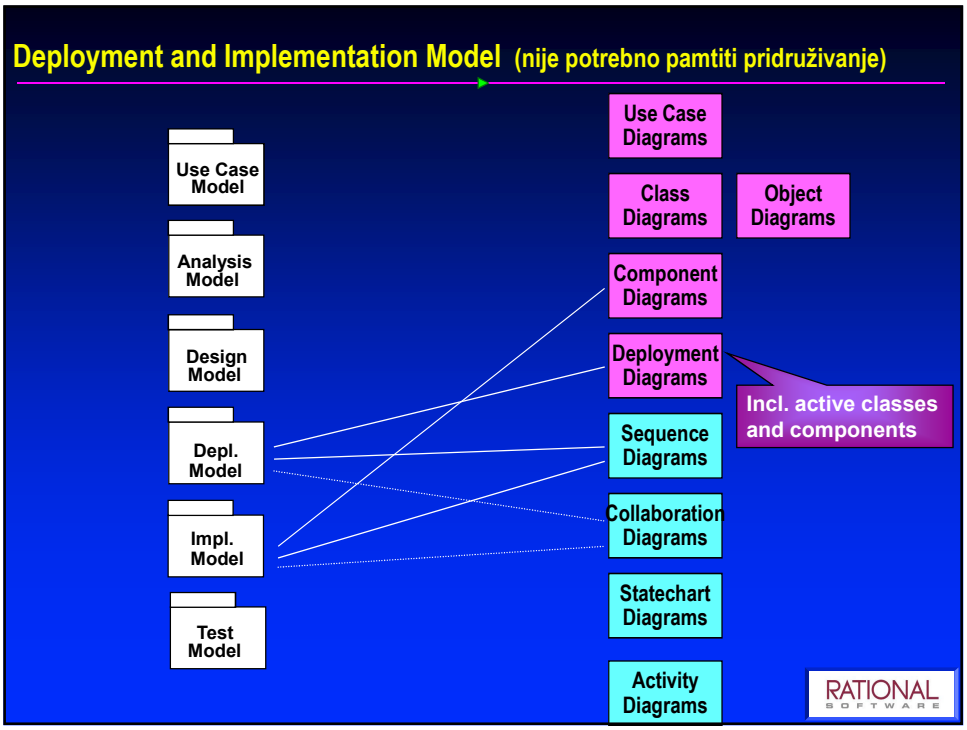

Iterations and Workflow (aktivnosti)
Aktivnosti
Core Workflows
Requirements
Analysis
Design
Implementation
Test
Phases
Faze životnog ciklusa
Inception
Elaboration
Construction
Transition
An iteration in the elaboration phase
Intenzitet aktivnosti
Preliminary Iteration(s)
iter. #1
iter. #2
iter. #n
iter. #n+1
iter. #n+2
iter. #m
iter. #m+1
Time
Iterations
RATIONAL SOFTWARE

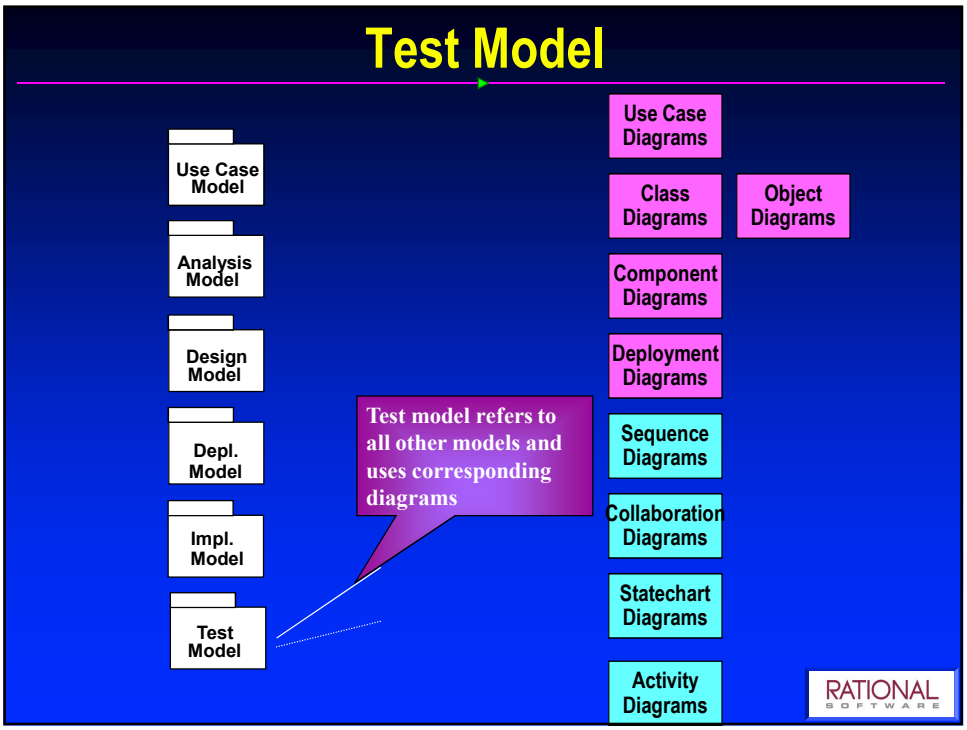

Unified Process structure (detaljizirano)
Inicijalna aktivnost
Process Workflows
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Temeljne aktivnosti
Supporting Workflows
Configuration Mgmt
Management
Environment
Potporne aktivnosti
Phases
Inception
Elaboration
Construction
Transition
Preliminary Iteration(s)
Iter. #1
Iter. #2
Iter. #n
Iter. #n+1
Iter. #n+2
Iter. #m
Iter. #m+1
Iterations
RATIONAL SOFTWARE

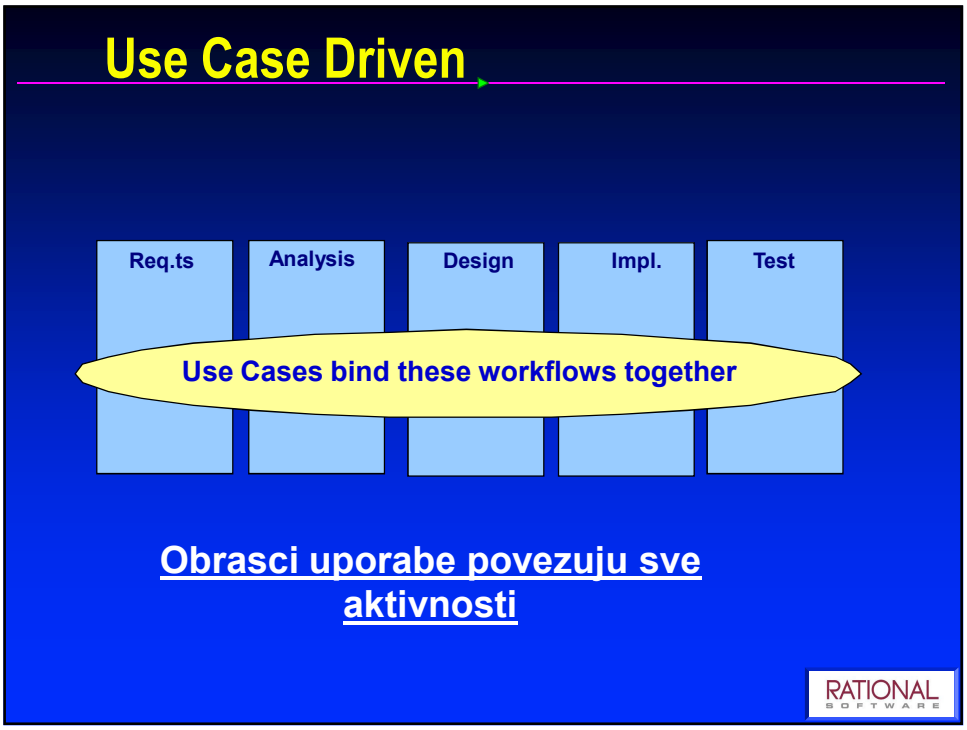

Workflows and Models
Aktivnosti:
Requirements
Use Case Model
Analysis
Analysis Model
Design
Design Model
Depl. Model
Implementation
Impl. Model
Test
Test Model
UML diagrams provide views into each model
Each workflow is associated with one or more models.
Svakoj aktivnosti pridruženi su modeli (opisi iz određene perspektive).
Modeli su dokumentirani dijagramima (pogledima).
RATIONAL SOFTWARE


Use Case Model (nije potrebno pamtiti pridruživanje)
(Obrazac uporabe)
Use Case Model
Analysis Model
Design Model
Depl. Model
Impl. Model
Test Model
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Deployment Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams
RATIONAL SOFTWARE


Analysis & Design Model (nije potrebno pamtiti pridruživanje)
Use Case Model
Analysis Model
Design Model
Depl. Model
Impl. Model
Test Model
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Incl. subsystems and packages
Deployment Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams
RATIONAL SOFTWARE


Deployment and Implementation Model (nije potrebno pamtiti pridruživanje)
Use Case Model
Analysis Model
Design Model
Depl. Model
Impl. Model
Test Model
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Deployment Diagrams
Incl. active classes and components
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams
RATIONAL SOFTWARE


Test Model
Use Case Model
Analysis Model
Design Model
Depl. Model
Impl. Model
Test Model
Test model refers to all other models and uses corresponding diagrams
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Deployment Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams
RATIONAL SOFTWARE


Use Case Driven
Req.ts
Analysis
Design
Impl.
Test
Use Cases bind these workflows together
Obrasci uporabe povezuju sve aktivnosti
RATIONAL SOFTWARE

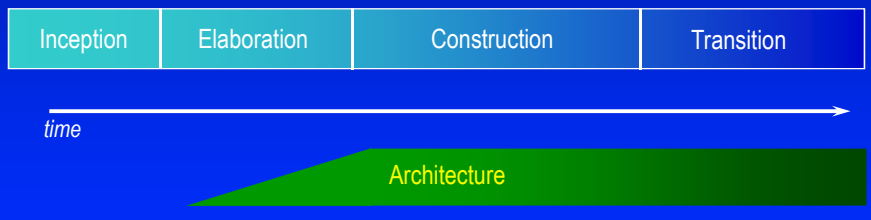
Use Cases Drive Iterations

- Drive a number of development activities
  - Creation and validation of the system's architecture
  - Definition of test cases and procedures
  - Planning of iterations
  - Creation of user documentation
  - Deployment of system
- Synchronize the content of different models

RATIONAL SOFTWARE

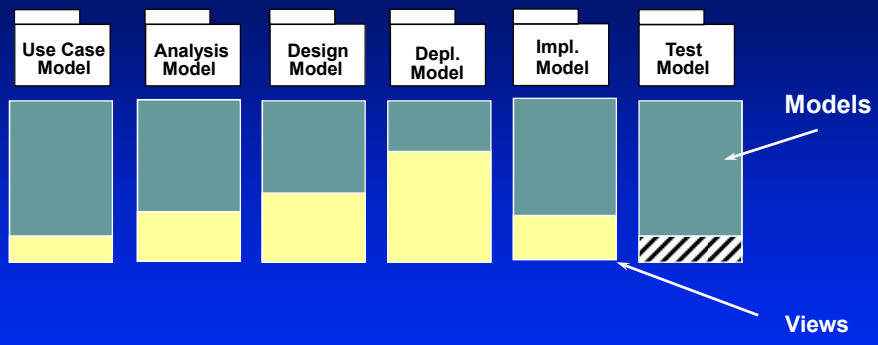
Architecture-Centric

- Models are vehicles for visualizing, specifying, constructing, and documenting architecture
- The Unified Process prescribes the successive refinement of an executable architecture



RATIONAL SOFTWARE

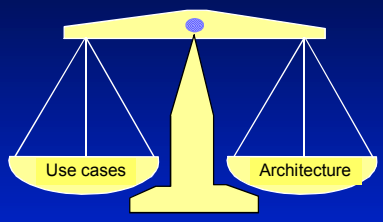
Architecture and Models



Architecture embodies a collection of views of the models
(i.e. a collection of diagrams)

RATIONAL SOFTWARE

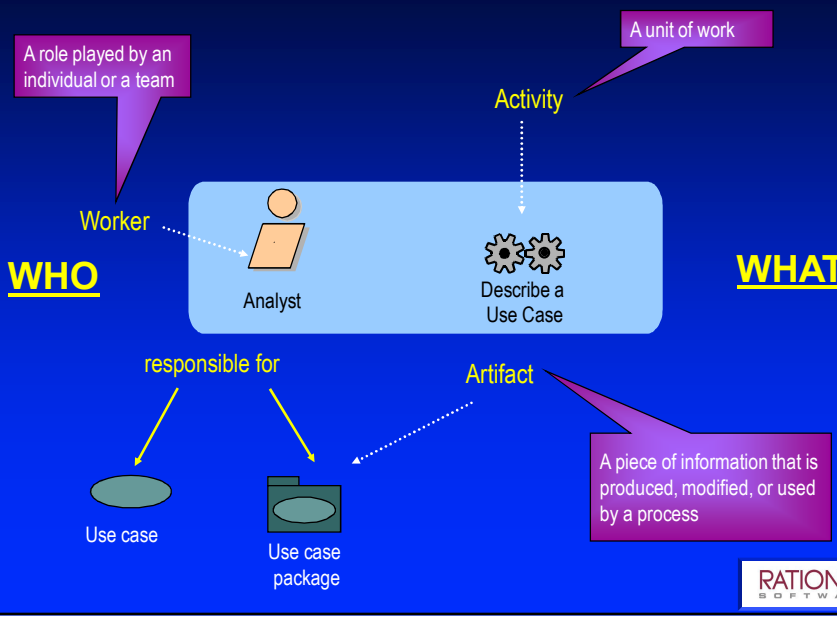
Function versus Form



- Use case specify function; architecture specifies form
- Use cases and architecture must be balanced

RATIONAL SOFTWARE

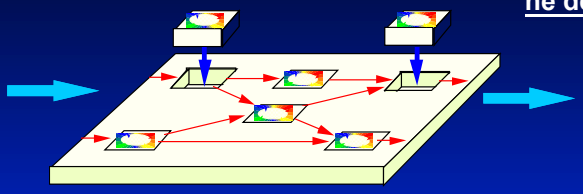
The Unified Process is Engineered



RATIONAL SOFTWARE

The Unified Process is a Process Framework

(samo okosnica, a ne detaljni prikaz)



There is NO Universal Process!

- The Unified Process is designed for flexibility and extensibility
  - » allows a variety of lifecycle strategies
  - » selects what artifacts to produce
  - » defines activities and workers
  - » models concepts

RATIONAL SOFTWARE

Two Parts of a Unified Whole

The Unified Modeling Language

The Unified Process

- OMG standard

- Convergence in the future
- Convergence through process frameworks

RATIONAL SOFTWARE

RUP - DOBRA PRAKSA

- Razvijaj programsku potporu iterativno.
- Upravljaj zahtjevima (t.j. promjenama zahtjeva).
- Uporabi arhitekturu zasnovanu na komponentama.
- Vizualno modeliraj programsku potporu.
- Verificiraj kakvoću programske potpore.
- Upravljaj promjenama programske potpore.

72

OBJAŠNJENJE STATIČKIH AKTIVNOSTI

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 29). **(Sommerville)**
Project management	This supporting workflow manages the system development (see Chapter 5). **(Sommerville)**
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

73

RUP zaključci:

- 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 dinamičke faze i statičke aktivnosti procesa (vidi 2D prikaz).
- Za opis pojedinih aktivnosti koriste se odgovarajući modeli.
- Modeli su dokumentirani jednim ili više dijagrama (pogleda).
- Dijagrami su definirani UML standardom.
- Arhitektura sustava sadrži skup pogleda u modele (t.j. skup dijagrama).

74

CASE tehnologija

75

CASE - Computer-aided software engineering

To su programski produkti koji podupiru proces programskog inženjerstva, a posebice aktivnosti razvoja, oblikovanja i evolucije.

Iako je CASE tehnologija dovela do značajnog unapređenja procesa oblikovanja programske potpore, ipak poboljšanja nisu sukladna očekivanjima (poboljšanje efikasnosti za red veličine !?).

Zašto :

Programsko inženjerstvo zahtijeva kreativnost, koju nije lako, a često ni moguće automatizirati.

76

CASE TEHNOLOGIJA

- Programsko inženjerstvo je timska aktivnost. U većim projektima mnogo vremena se utroši na interakcije unutar tima. CASE tehnologija slabo podupire timski rad (vidi Dom. zad. 1).

CASE podupire automatizaciju oblikovanja raznim alatima kao npr.:

- Grafički editori za razvoj modela sustava.
- Rječnici i zbirke za upravljanje entitetima u oblikovanju.
- Grafička okruženja za oblikovanje i konstrukciju korisničkih sučelja.
- Alati za pronalaženje pogrešaka u programu. Automatizirani prevoditelji koji generiraju nove inačice programa.

Kako izabrati pogodan alat ?

77

CASE klasifikacija

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.

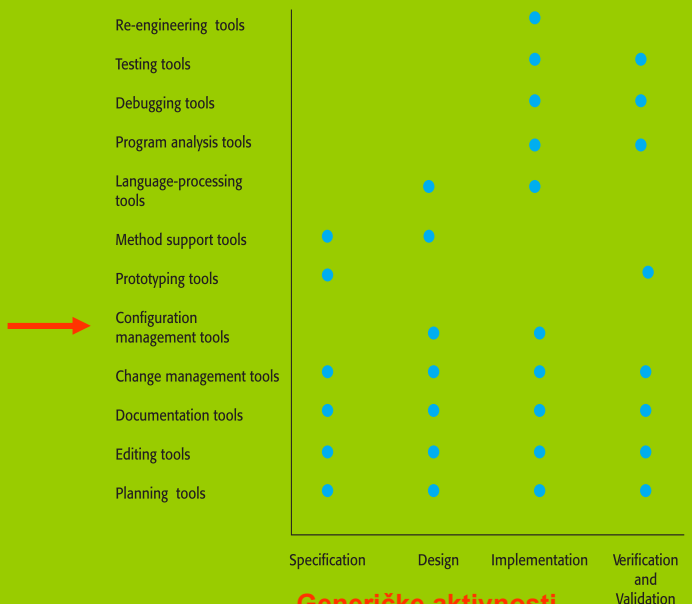
78

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

79

Klasifikacija CASE alata prema aktivnostima



Generičke aktivnosti

80

CASE INTEGRACIJSKA PERSPEKTIVA

Sveobuhvatnost u procesu

- **Alati**

  Podupiru individualne zadatke u procesu (npr. oblikovanje, provjera konzistencija, editiranje teksta, …)

- **Radne klupe** (*engl.* workbenches)

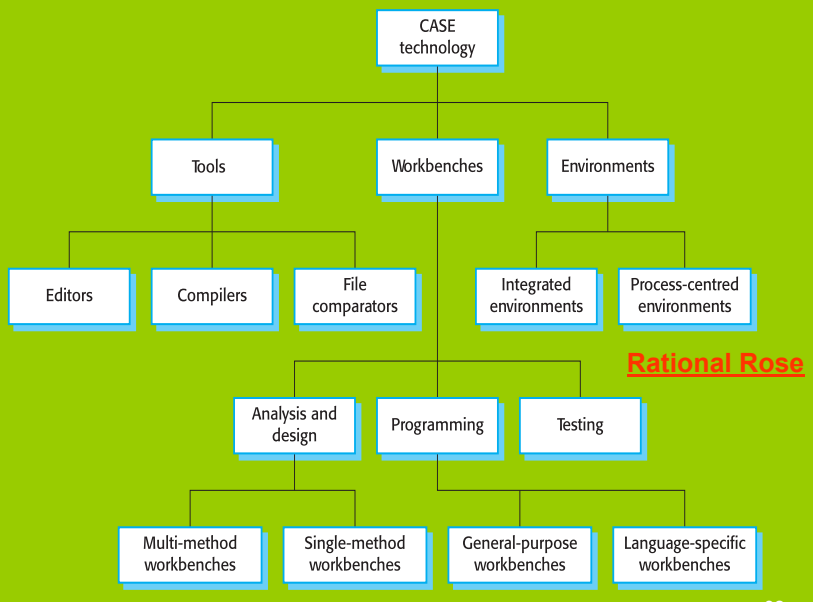
  Podupiru pojedine faze procesa (npr. specifikaciju)

- **Razvojne okoline** (*engl.* environments)

  Podupiru cijeli ili značajan dio procesa programskog inženjerstva. Uključuju nekoliko integriranih radnih klupa.

81

Alati, radne klupe, razvojne okoline



82

CASE alati u okviru predmeta:

"Oblikovanje programske potpore"

Subversion – upravljanje konfiguracijama programske potpore.

ArgoUML - oblikovanje objektno usmjerene arhitekture i generiranje koda.

SMV - verifikacija dijelova programske potpore simboličkom provjerom modela.

83

ZAKLJUČCI (1 od 2)

- Procesi u programskom inženjerstvu (SE) su strukturirane aktivnosti usmjerene na produkciju i evoluciju programskih proizvoda.
- Modeli procesa SE su njihova apstraktna reprezentacija.
- Nema univerzalnog i standardnog SE procesa.
- Generičke aktivnosti u procesu su specifikacija, oblikovanje i implementacija, validacija i verifikacija te evolucija.
- Generički modeli SE procesa opisuju njihovu organizaciju (npr.: vodopadni, evolucijski, komponentni, RUP).
- Iteracije opisuju proces SE kao ciklus različitih aktivnosti unutar generičkih modela procesa.
- Inženjerstvo zahtjeva je proces izrade specifikacije programskog produkta.

84

ZAKLJUČCI (2 od 2)

- 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.
- Rational Unified Process je model procesa koji povezuje aktivnosti i faze izvođenja procesa, definira arhitekturu sustava kroz modele i dijagrame koristeći UML standard.
- CASE tehnologija podupire automatizaciju aktivnosti tijekom procesa programskog inženjerstva.

85