Metodologije razvoja softvera

- Softverski inženjering
- Razvoj metodologija
- Modeli i modelovanje
- Softverski procesi
- *Microsoft Solution Framework*
- *Rational Unified Process* metodologija
- *Scrum*
- *MSF for Agile*
- *Open Unified Process*

Softverski inženjering

- Formalne **definicije** softverskog inženjeringa:
  - Inženjerska disciplina koja se bavi **svim aspektima proizvodnje softvera.**
  - Disciplina koja obuhvata **znanje, alate i metode** za definisanje zahteva, razvoja softvera, rukovanje i održavanje softvera.
  - Sistemski pristup u:
    - Analizi
    - Projektovanju
    - Razvoju
    - Testiranju
    - Implementaciji
    - Održavanju
    - Reinženjeringu softvera

- Podrazumeva **primenu inženjeringa na softver** - integriše matematiku, računarske nauke i praktične veštine čije poreklo leži u inženjeringu.
- Termin se prvi put upotrebio **1968**. g. na NATO *Software Engineering* konferenciji.

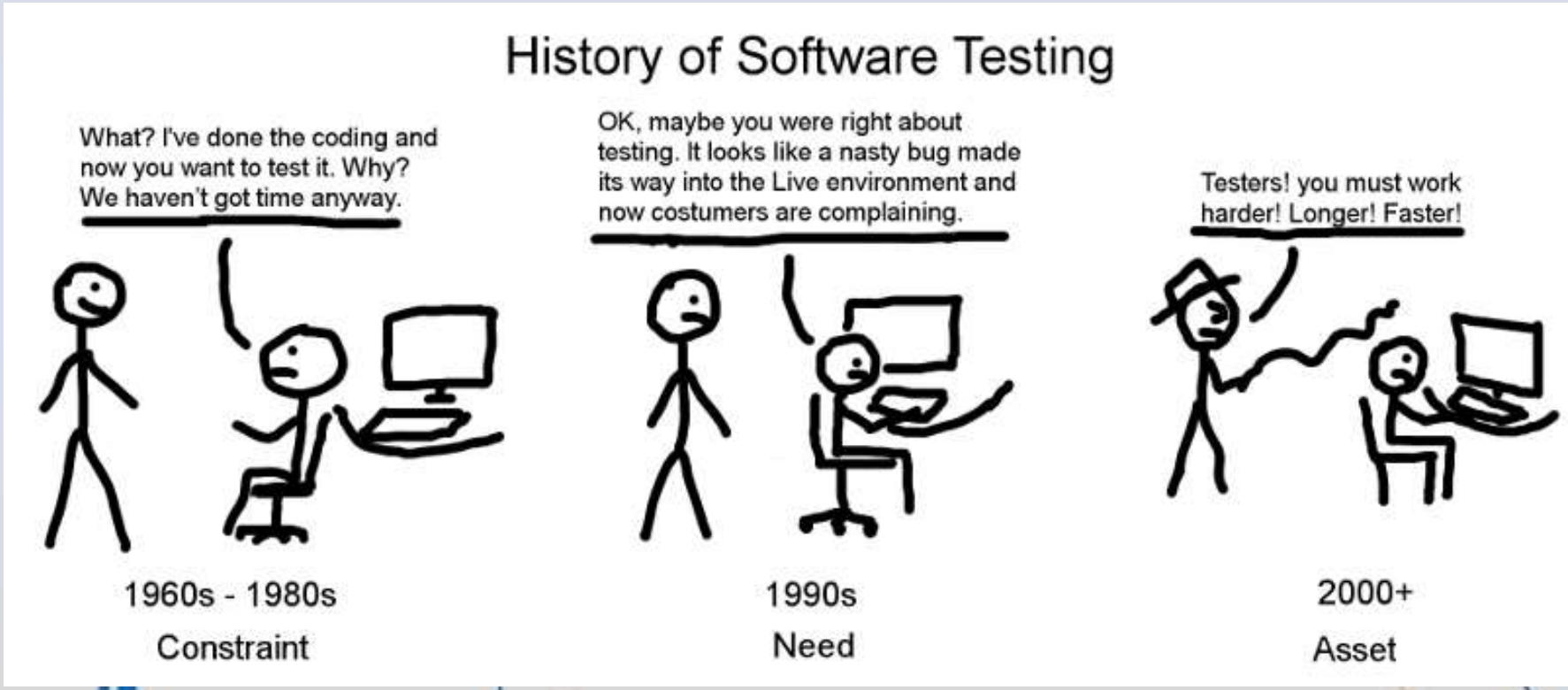
Ko su softverski inženjeri?

- Softverski inženjeri (*Software Engineers*) su ljudi koji koriste svoje znanje iz oblasti računarskih nauka, inženjerstva i matematike kako bi analizirali, projektovali, razvijali, konfigurisali i instalirali softvere u poslovna okruženja koja imaju potrebe za računarskim sistemima.
- Rade u timovima koji često uključuju eksperte iz različitih domena (npr., proizvodnja, marketing, dizajn) kako bi isporučili kompletan i kvalitetan softver.

Istorija razvoja softverskog inženjeringa

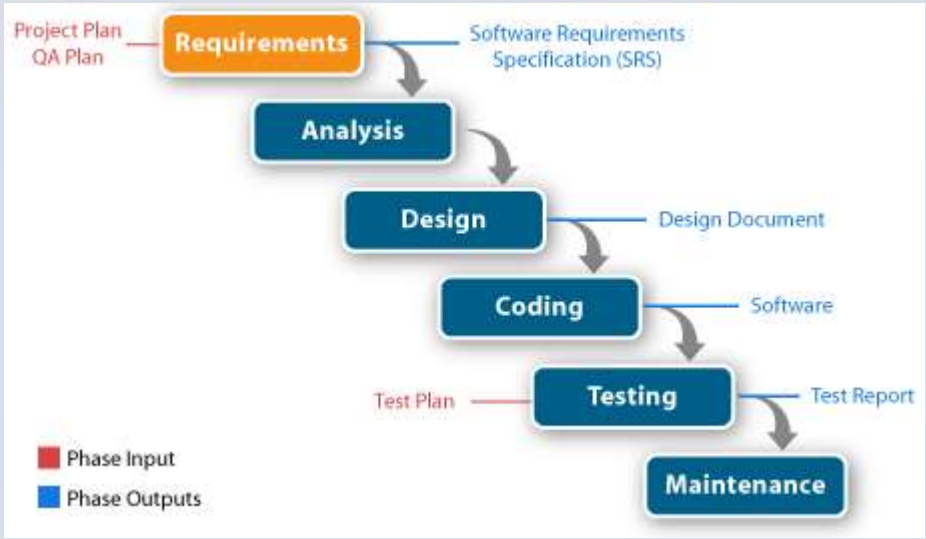
Godina	Kratak opis razvoja
1940-te	▪ ručno pisanje mašinskog koda (0 i 1)
1950-te	▪ makro asembleri, interpreteri i prva generacija kompajlera
1960-te	▪ funkcionalno programiranje (Basic, Fortran, Cobol …) ▪ mainframe računari i softveri za velike korporacije
1970-te	▪ kolaborativni alati ▪ rast manjih poslovnih softvera
1980-te	▪ personalni računari (PC) i radne stanice ▪ rast potrošačkih softvera (MRP I, MRP II …)
1990-te	▪ objektno-orjentisano programiranje (C++, C#, Java …) ▪ agilni procesi ▪ integrisana poslovna rešenja (ERP, CRM …)
2000-te do danas	▪ web servisi i servisno-orijentisano programiranje ▪ inteligentna poslovna rešenja (BI) ▪ servisi u Cloud Computing okruženju

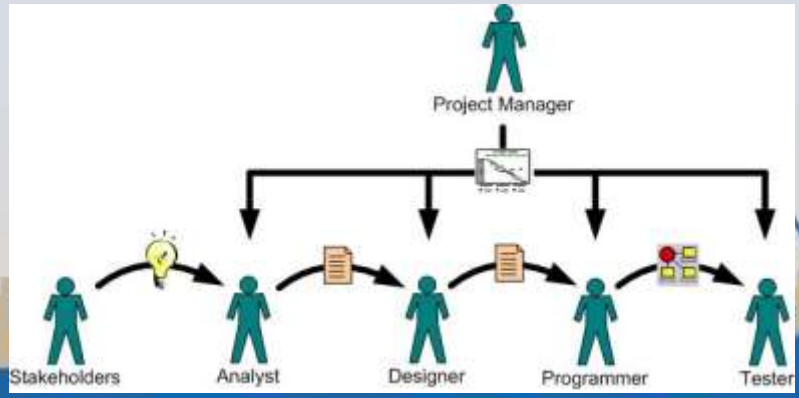
Istorija softverskog testiranja :-)



Pojava metodologija

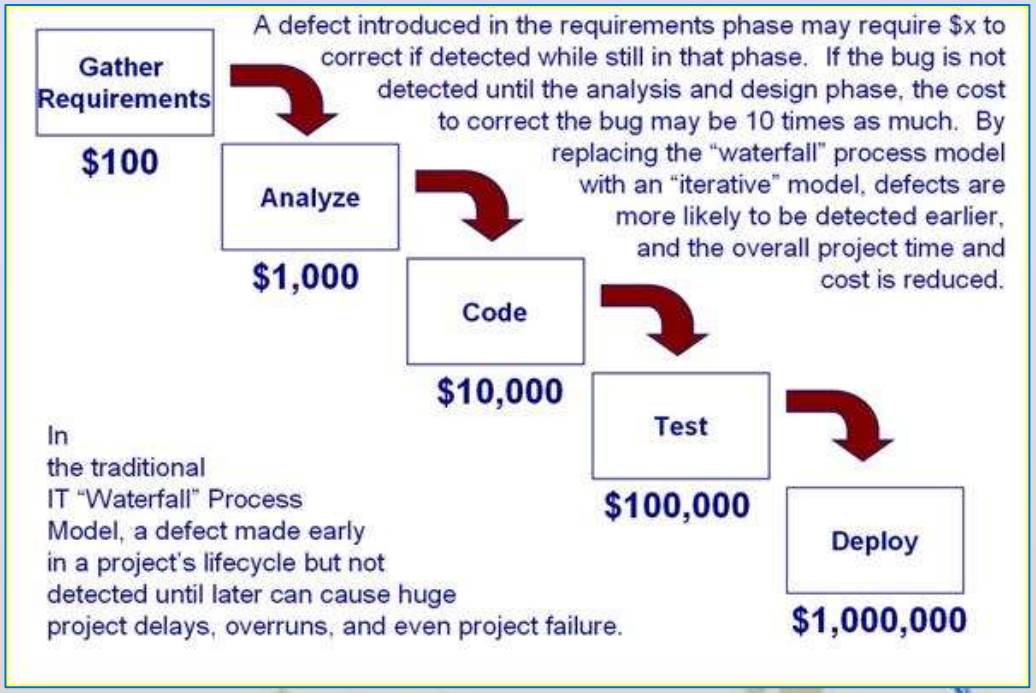
- Razvojem softvera pojavila se potreba za:
  - **organizovanjem** rada **tima**
  - **grupisanjem** određenih **procesa** u sisteme
  - pravljenje **obrazaca** (šablona) koji će olakšati razvoj sistema
  - …
- Pored naziva dodavana je reč proces, metodologija ili okvir (*framework*)





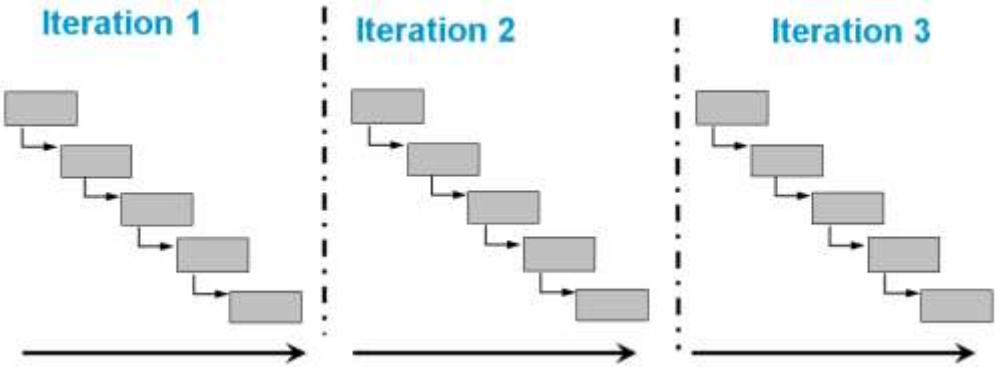
Razvoj metodologija
Tradicionalne metodologije

- **Metodologija**
  - Herbert D. Benington, 1956. godine
  - Prva upotreba termina metodologija - prezentacija o razvoju automatizovanog sistema kontorole za praćenje i presretanje neprijateljskih aviona
- **Vodopad (*Waterfall*) model**
  - Vremenom, na osnovu iskustva, metodologije su proširivane i menjane
  - Vinston W. Rojs, 1970. godine – prvi formalni opis vodopad modela



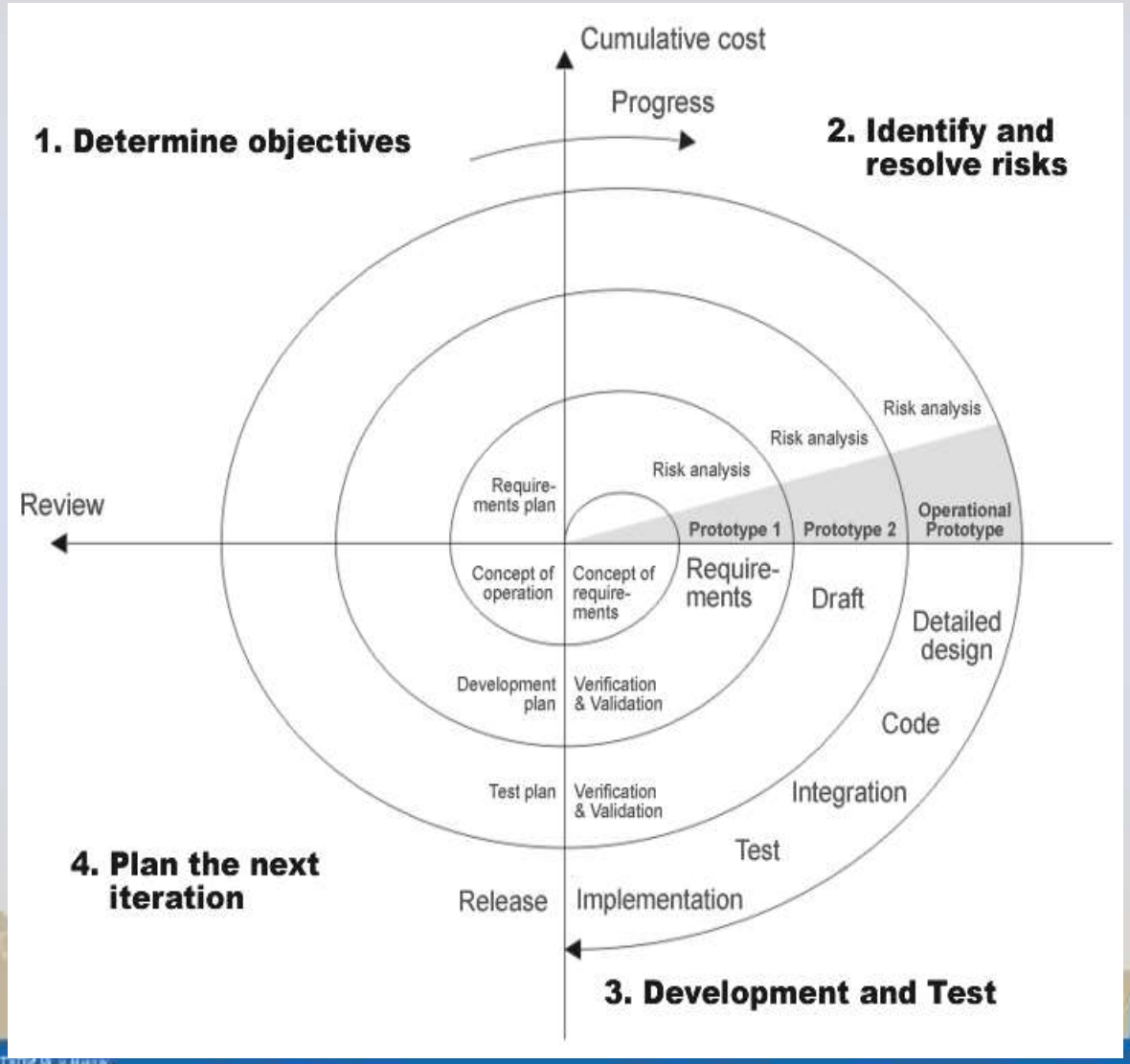
Razvoj metodologija
Tradicionalne metodologije

- **Iterativne (*Iterative*) metodologije**
  - Nastaju kao odgovor na nedostatke vodopad modela
  - Iterativno – inkrementalni razvoj
  - Jedan od predstavnika iterativnih metodlogija je spiralni model
  - **Spiralni model** je definisao Barry Boehm, 1986. godine



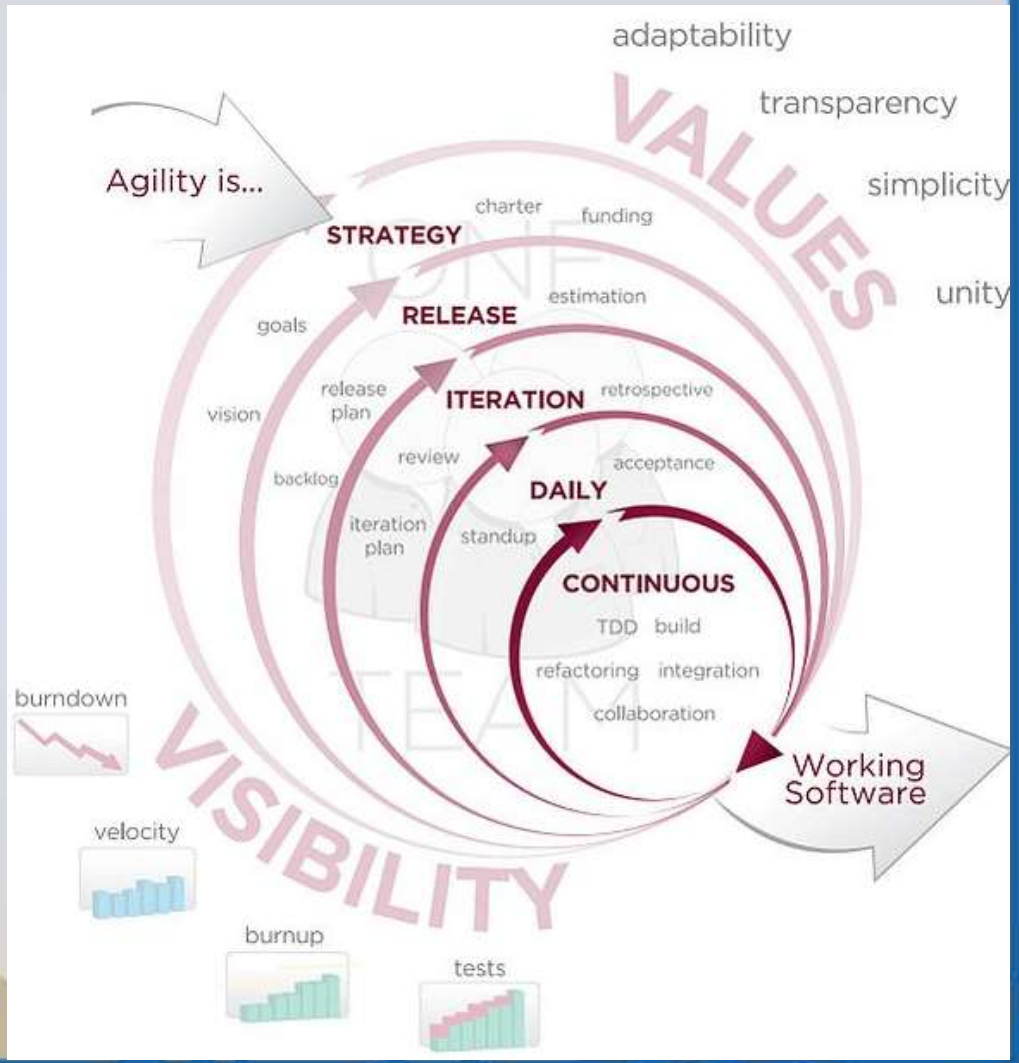
Iterativno inkrementalni proces razvoja softvera
proces koji obezbeđuje da sistem koji se razvija inkrementalno raste u vremenu iz iteracije u iteraciju

Spiralni model (Boehm, 1988)



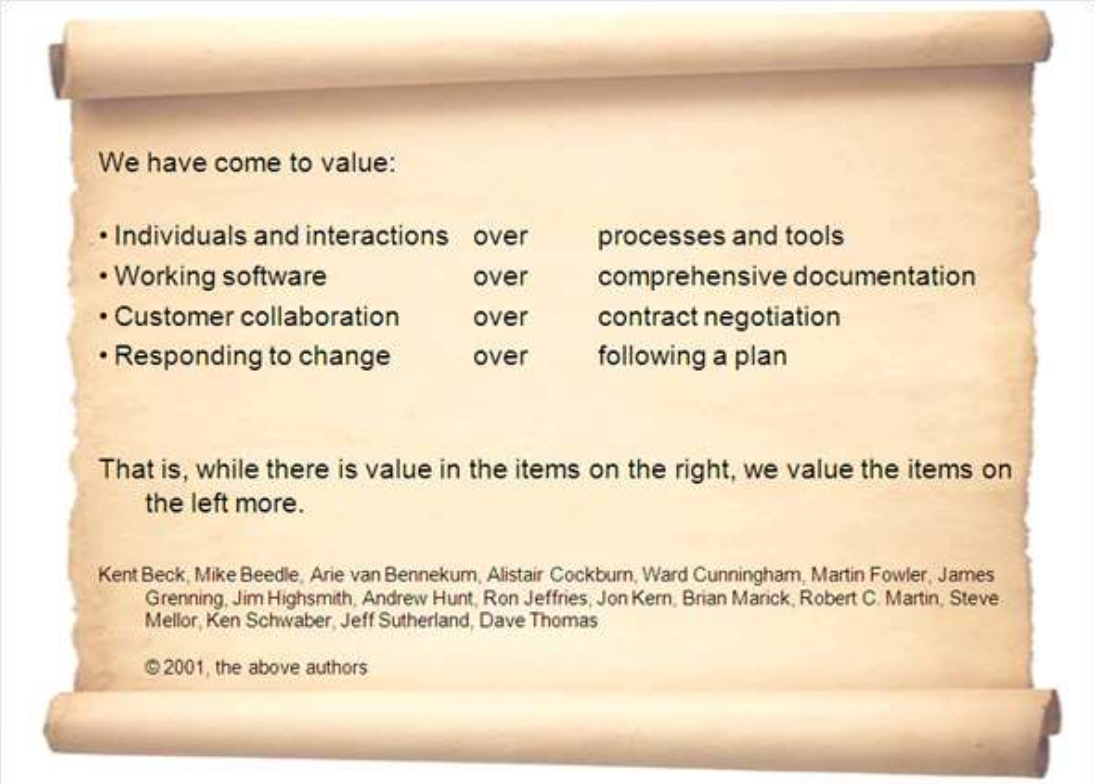
Agilne metodologije

- U Utahi, SAD, okupili su se eksperti softverske industrije, 2001. godine
  - Rezultat skupa je bio formiranje **Agilne Alijanse** - neprofitna organizacija koja promoviše koncepte agilnog razvoja i pomaže organizacijama u njihovom usvajanju
  - Kreira se ***Agile Manifesto*** koji uključuje četiri postulata i seriju pridruženih principa



Manifest za agilni razvoj softvera

- **Pojedinci i interakcije** između njih su važnije od procesa i alata
- **Softver koji radi** je važniji od dokumentacije koja raste
- **Saradnja sa korisnicima** je važnija od procesa ugovaranja
- **Odziv na promene** je važniji od striktnog pridržavanja i praćenja plana



We have come to value:

• Individuals and interactions	over	processes and tools
• Working software	over	comprehensive documentation
• Customer collaboration	over	contract negotiation
• Responding to change	over	following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Razvoj metodologija

Vodopad procesi

Prediktivni

Iterativni procesi

Iterativno-inkrementalni

(Spiralni, RAD,

RUP ...)

Agilni procesi

Adaptivni

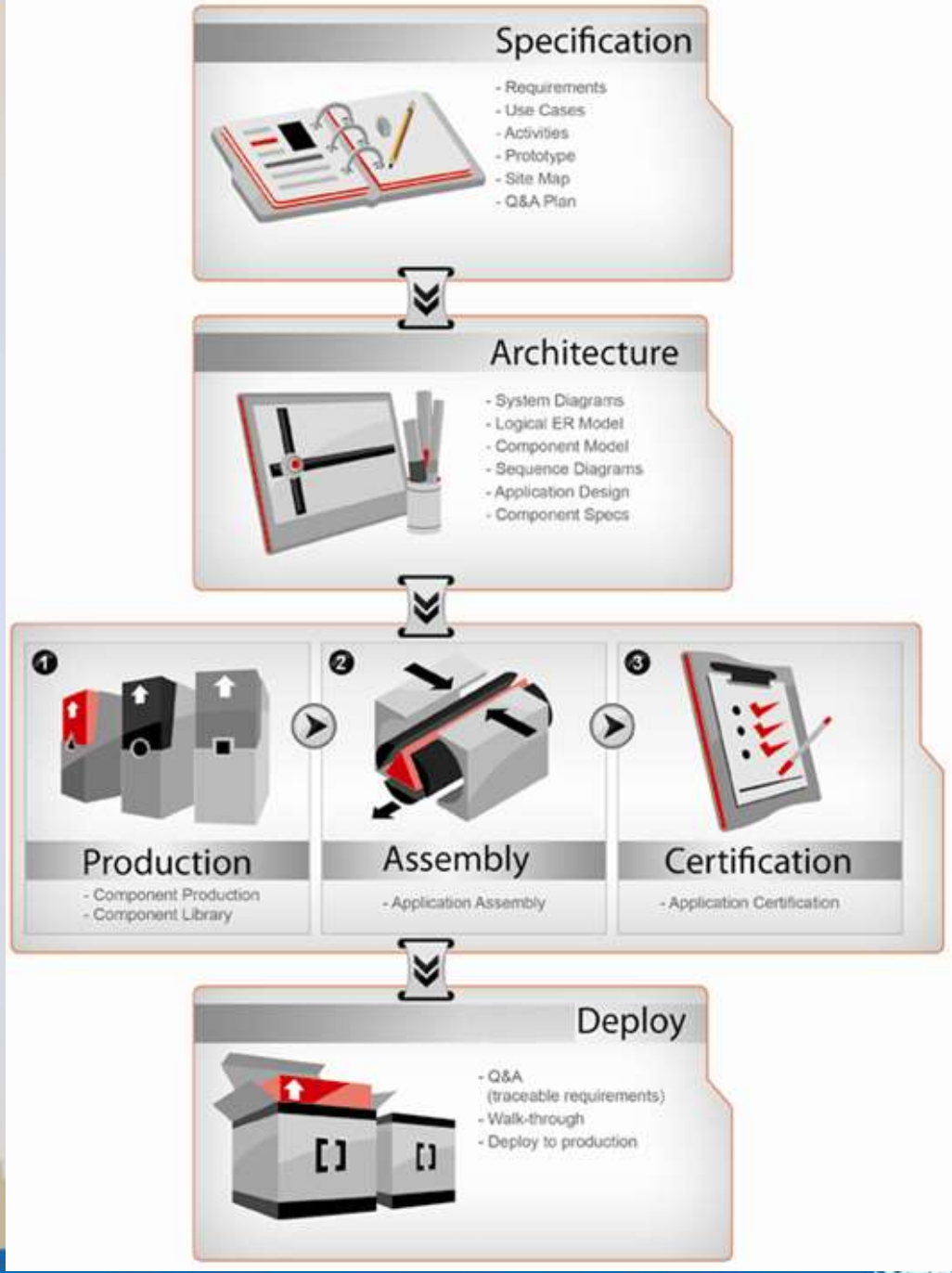
(Scrum, XP, Open UP,

Crystal, Lean, DSDM,

Agile MSF ...)

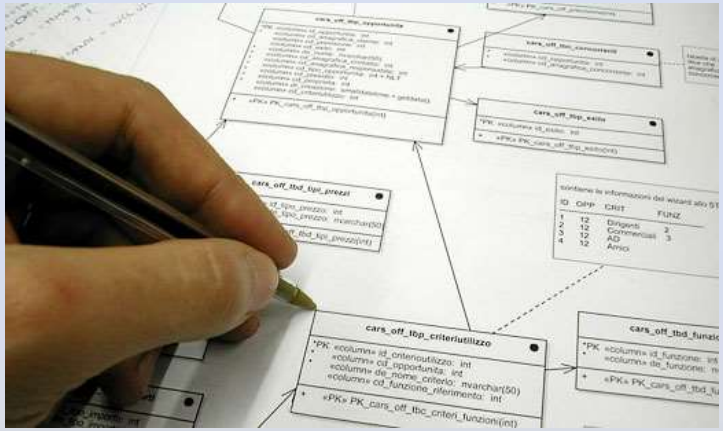
Metodologija

- Fizička implementacija logičkog životnog ciklusa sistema koji uključuje:
  - **aktivnosti** za svaku fazu
  - individualna i grupna **pravila** za svaku aktivnost
  - **standarde kvaliteta** i **isporučljivosti** za svaku aktivnost
  - **alate i tehnike** koje treba da budu upotrebljene za svaku aktivnost



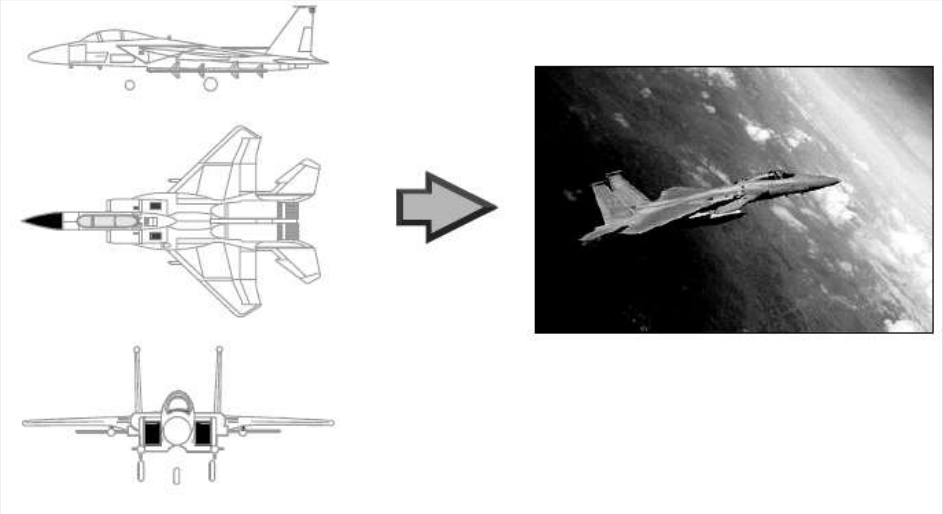
Zašto kompanije koriste metodologije?

- Obezbeđuje **konzistentan** prilaz
- **Smanjuje rizik** od grešaka
- Izdaje **kompletnu** i konzistentnu **dokumentaciju** za trenutne i buduće projekte
- Isporučuje se **kvalitetan softver** koji se lako može menjati
- Koriste se **najbolje prakse** razvoja sistema
- Usled činjenice da se **projektni timovi menjaju**, omogućava da oni timovi koji nastavljaju rad, brzo i lako shvate rezultate rada svojih prethodnika
- Lakše je **izmene** vršiti **nad modelima** nego u programskom kodu



Šta su modeli?

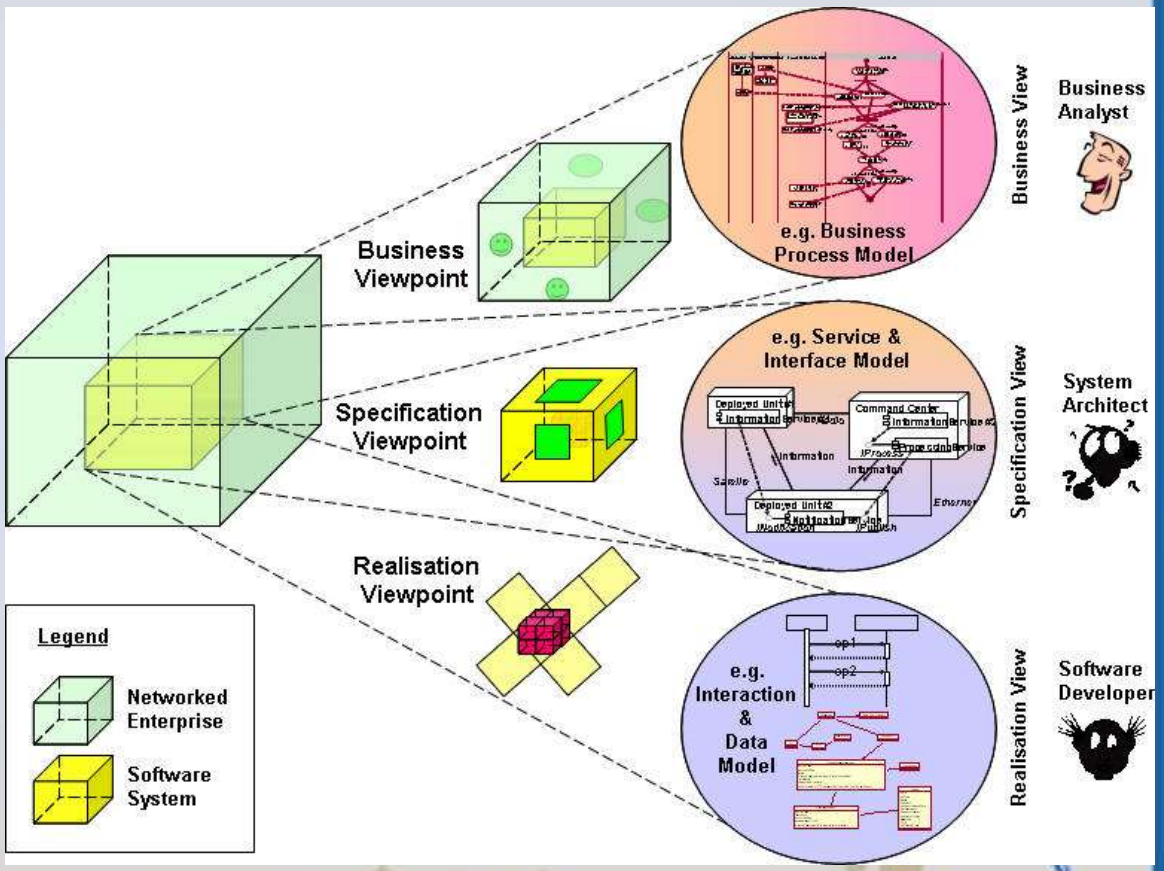
- Model je **pojednostavljenje realnosti**
  - Maketa aviona, plan zgrade …
  - Šema baze podataka
- Modeli se **iskazuju u nekom jeziku** (jezik modelovanja):
  - **Tekstualna** notacija (modeli u tekstualnom obliku)
  - **Grafička** notacija (modeli u vidu dijagrama)
- **Neformalni** i **formalni** modeli
  - Ako su sintaksa i semantika jezika formalno definisani, onda su modeli iskazani na tom jeziku formalni
- **Logički** i **fizički** modeli
  - **Logički** modeli pokazuju šta je sistem i šta on radi - opisuju sistem nezavisno od bilo koje tehničke implementacije
  - **Fizički** modeli pokazuju ne samo šta je sistem i šta on radi, već i kako je sistem fizički i tehnički implementiran



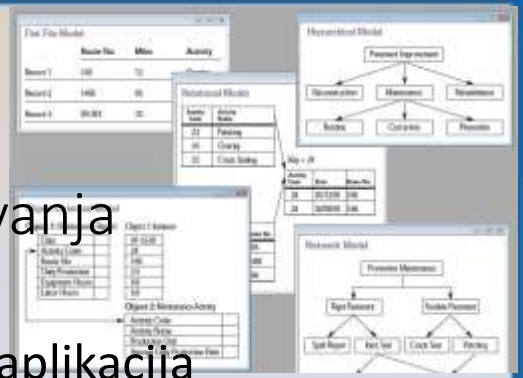
- Svaki sistem se može opisati sa **različitih aspekata** korišćenjem različitih modela
  - **Struktuirani** modeli - ističu strukturu sistema tj. sastavne delove sistema i njihov odnos
  - Modeli **ponašanja** - prikazuju dinamiku sistema

Čemu služe modeli?

- Modeli se grade kako bi se **bolje razumeo sistem** koji se razvija
- **Modelovanjem** se dostižu četiri cilja:
  - **Vizuelizacija sistema** kakvog želimo da bude – model pomaže timu da lako sagleda sistem sa svih perspektiva
  - **Specifikacija strukture ili ponašanja sistema** – modeli dokumentuju ponašanje i strukturu sistema pre njegovog kodiranja
  - Model daje **smernice za razvoj sistema** - služi kao vodič za developere
  - Modeli **dokumentuju** diskusije tj. odluke donešene tokom projektovanja sistema

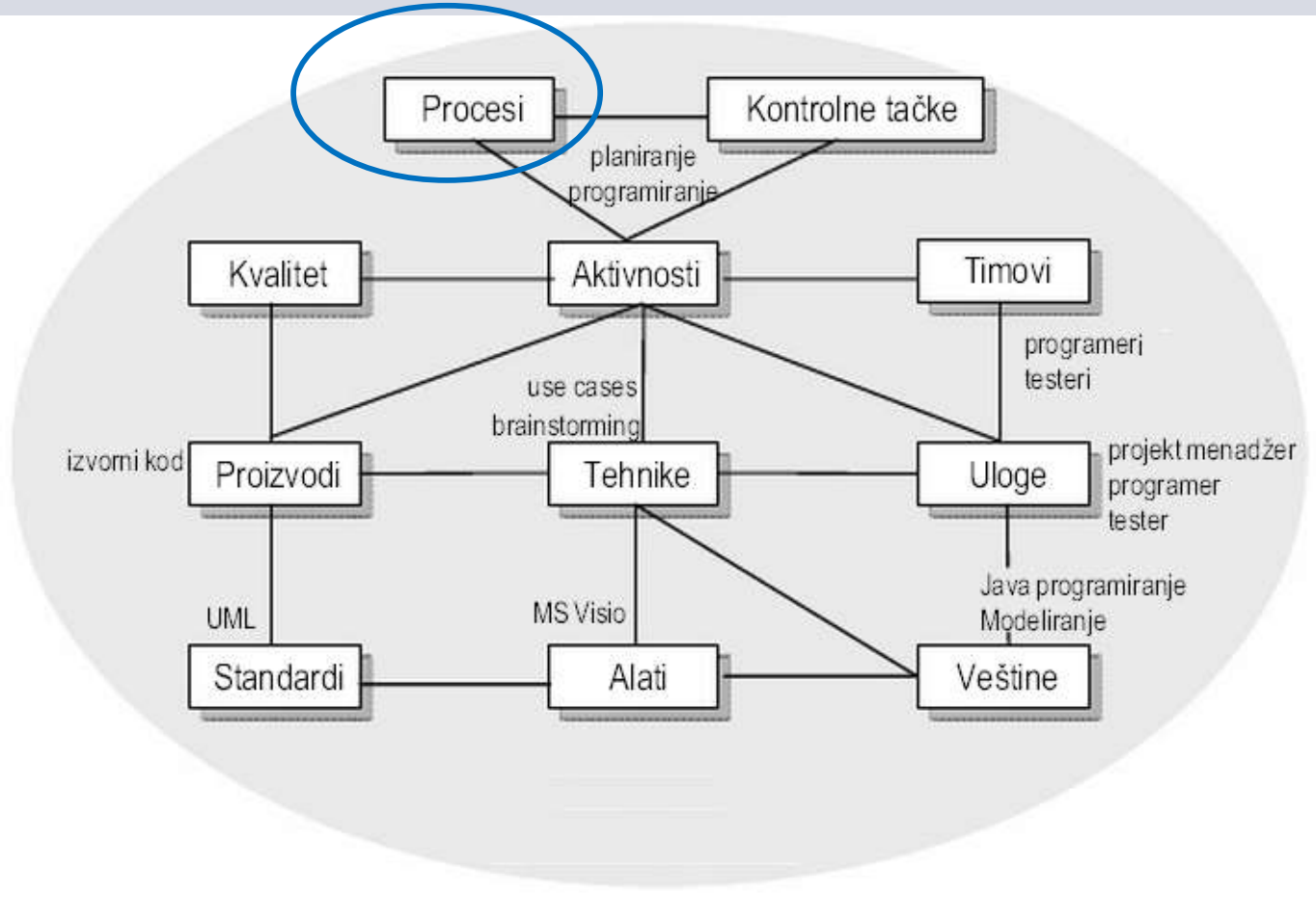


Istorija modelovanja
od funkcionalnog do servisnog modelovanja



- **Modelovanje ili modeliranje** je projektovanje softverskih aplikacija pre njihovog kodiranja
- Trenutno i buduće stanje sistema se modeluje kako bi se **identifikovali zahtevi, problemi, rizici i informacije koje nedostaju**
- **Modeliranje podataka** - najpopularnija tehnika za izražavanje poslovnih zahteva za podacima koji će biti uskladišteni u bazu podataka sistema
- **Modeliranje procesa** - tehnika koja se dosta praktikuje za izražavanje zahteva poslovnih procesa, tokova procesa, ulaza i izlaza
- **Objektno-orijentisano modeliranje** - tehnika koja eliminiše veštački razdvojene podatake od procesa. OO je naročito postao popularan pojavom UML-a (*Unified Modeling Language*) koji obezbeđuje i grafičku sintaksu za objektne modele
- **Servisno-orijentisano modeliranje** – tehnika koja povezuje gotove veb servise u servisno-orijentisanu aplikaciju, a zatim u servisno-orijentisane sisteme i omogućava njihovo objavljivanje i korišćenje u *cloud* okruženju

Elementi modela razvoja softvera



Softverski procesi

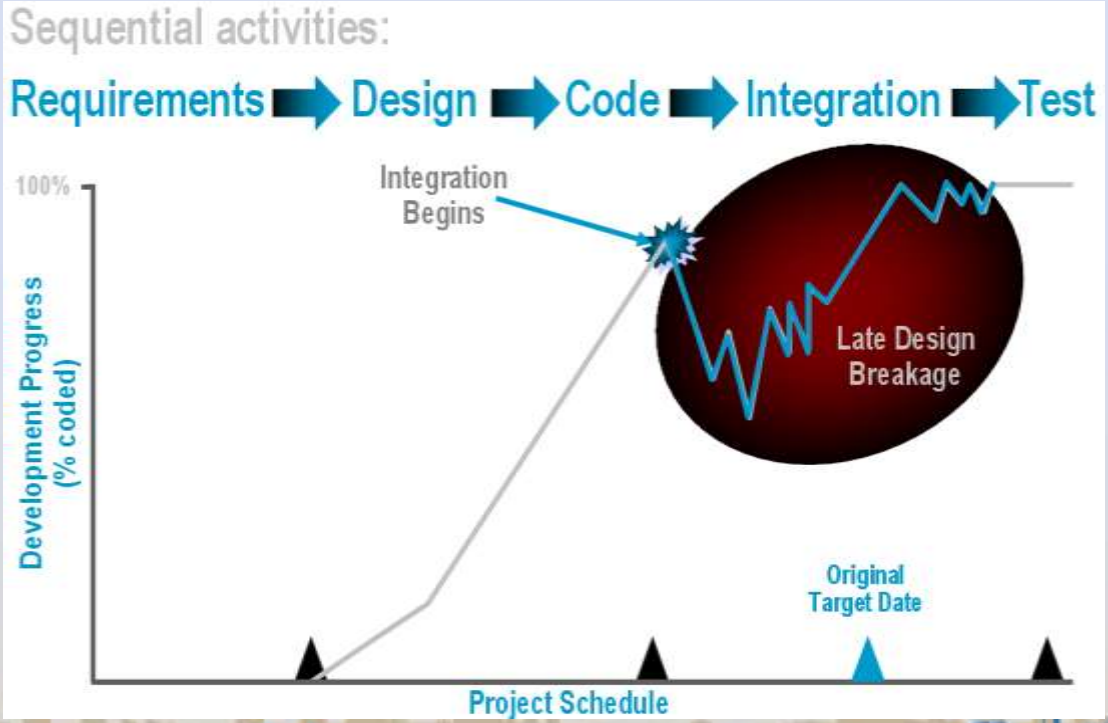
- **Teški procesi**
  - niz strogo definisanih koraka - naredna aktivnost ne može da počne ukoliko se nije završila prethodna
  - striktna podela uloga
  - dokumentacija
  - i druge formalne procedure u realizaciji softvera
- **Agilni procesi**
  - Termin “Agilno” – brzo, lako, okretno, sazrevanje u vremenu
  - Agilni softverski procesi – realizacija softvera koji se adaptira promenama okoline i zahteva korisnika
  - Oslanja se na povratnu spregu radi analize rezultata svake iteracije
  - Veoma brzi odziv na zahteve klijenata kroz interakciju sa klijentima
  - Timski rad, zajednička podela znanja i odgovornosti

Vodopad životni ciklus razvoja

Waterfall Development Lifecycle

- Vodopad pristup ne može adekvatno da se bori sa sve većom kompleksnošću koja nastaje:
  - Produženim trajanjem projekta
  - Sve većom **veličinom aplikacije**
  - Velikim ili **distribuiranim timovima**
  - Povećanom tehničkom kompleksnošću
  - Novinama u tehnologijama
- Glavni problem ovog pristupa je što ne omogućava identifikovanje i umanjenje rizika u ranim fazama projekta

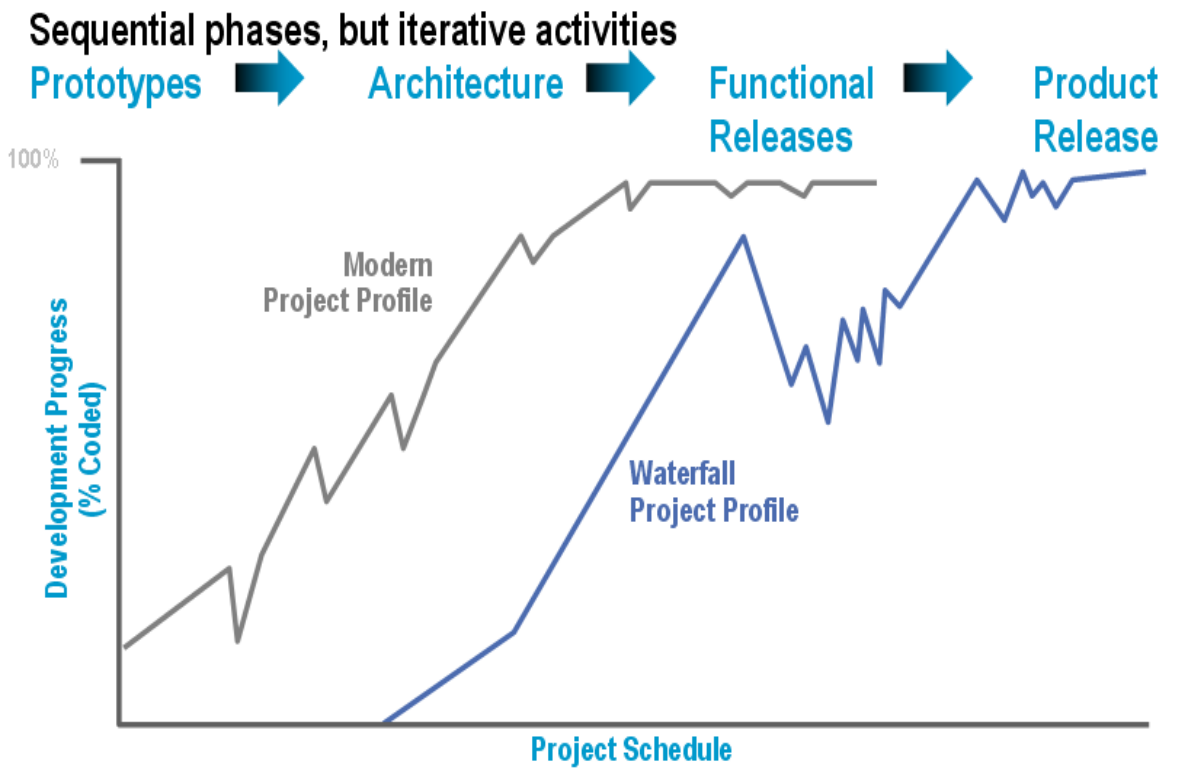
Šta se dešava u praksi



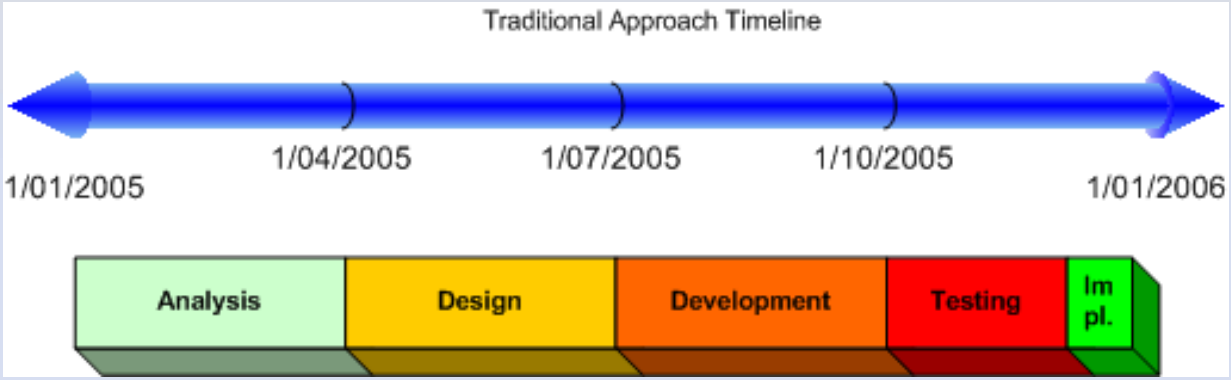
Iterativni razvoj

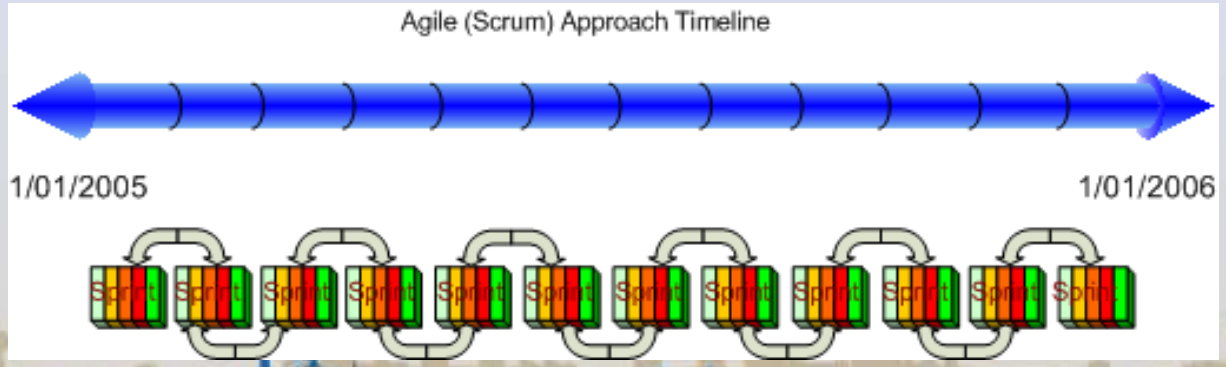
Iterative Development

- Životni ciklus softvera se sastoji od nekoliko sekvencijalnih iteracija
  - Svaka iteracija je **poseban mini-projekat** koji se sastoji od aktivnosti kao što su analiza zahteva, dizajn, programiranje i testiranje
  - Iteracije u prvim fazama ukazuju na najveće rizike
  - Svaka iteracija **proizvodi izvršne verzije**
  - Svaka iteracija **uključuje integraciju i testiranje**
  - Cilj završetka iteracije je stabilan, integrisan, testiran deo celokupnog softverskog sistema koji se gradi
- Razlika između dve sukcesivne iteracije je **inkrement**

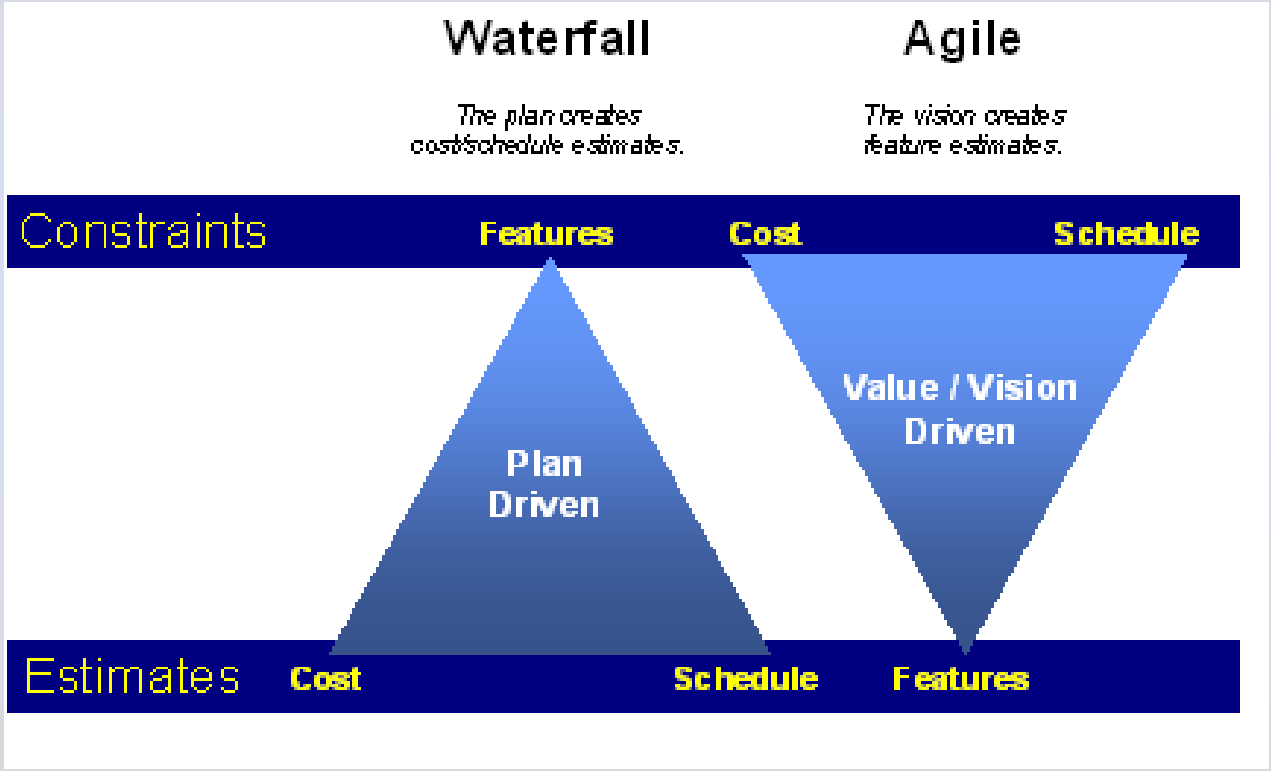


Proceduralni *vs* agilni pristup





Proceduralni *vs* agilni pristup

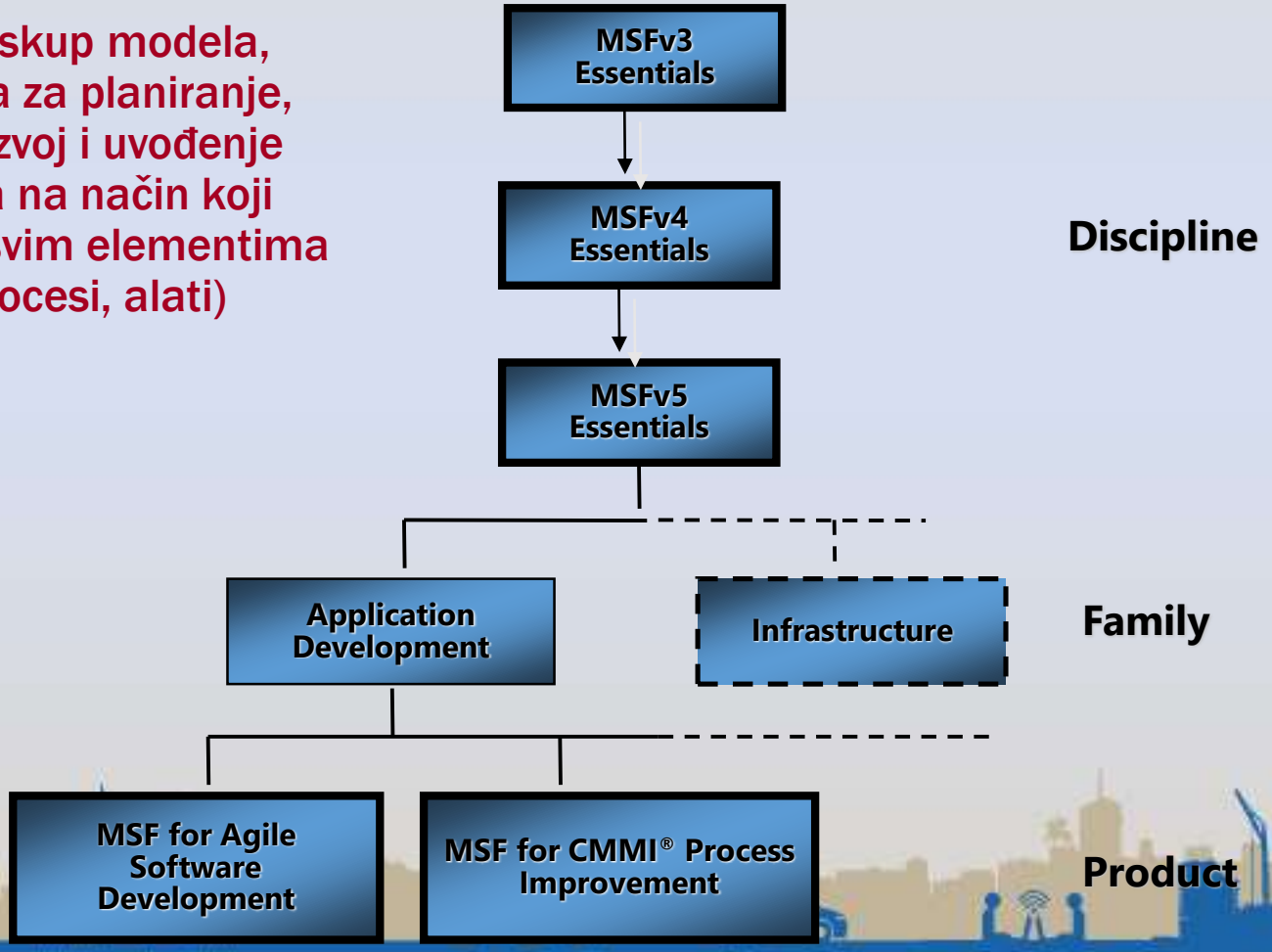


MSF procesni model
Microsoft Solution Framework

- Tradicionalni MSF
- *Framework vs* metodologija
- MSF discipline razvoja sistema
- Upravljanje rizicima
- MSF timski model
- Upravljanje projektima
- MSF faze procesa razvoja sistema

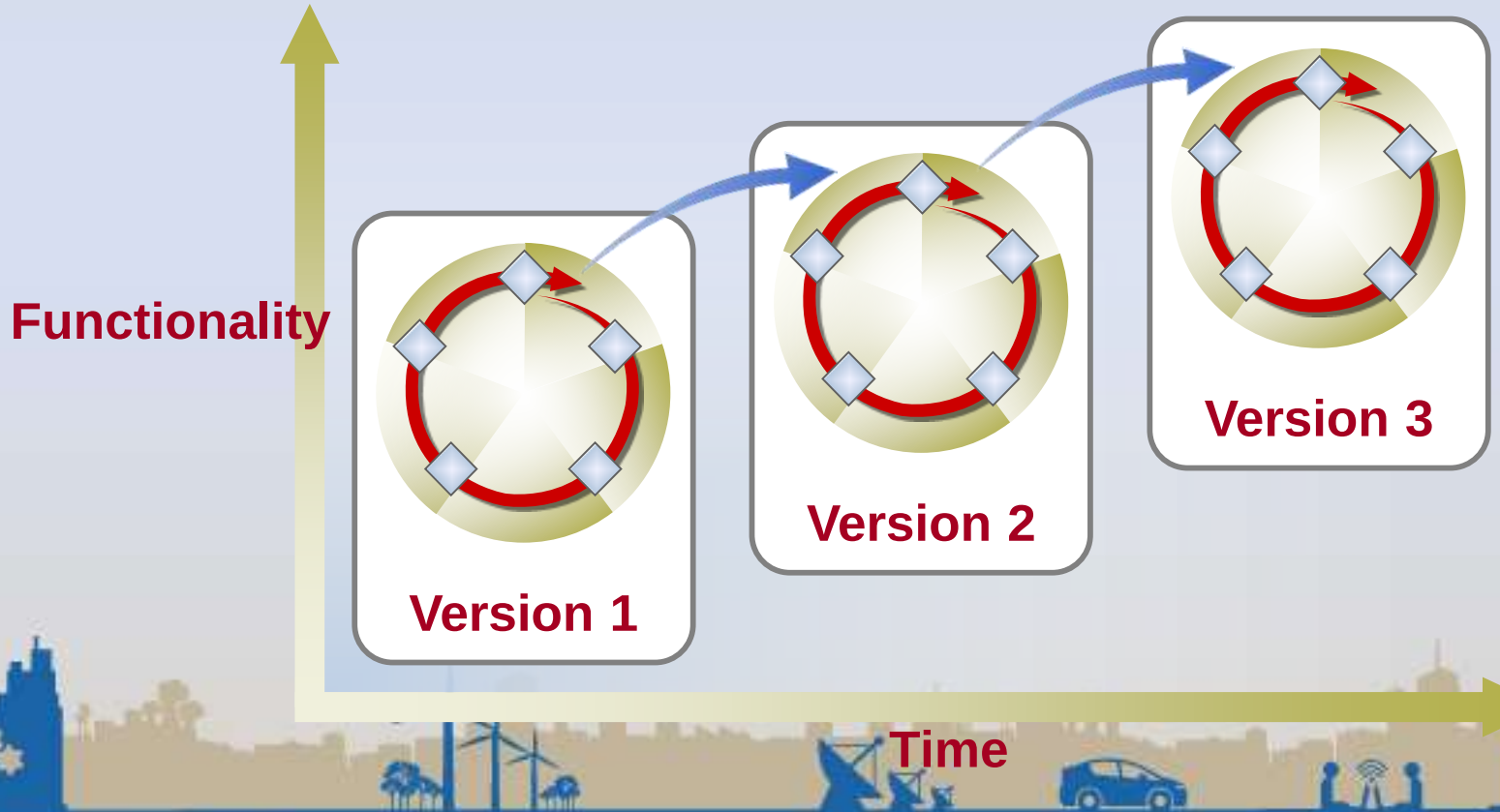
Microsoft Solutions Framework

- **MSF obezbeđuje skup modela, pravila i smernica za planiranje, projektovanje, razvoj i uvođenje poslovnih rešenja na način koji osigurava da se svim elementima projekta (ljudi, procesi, alati) uspešno upravlja**



Kako radi tradicionalni MSF procesni model?

- Verzionisanje istog proizvoda



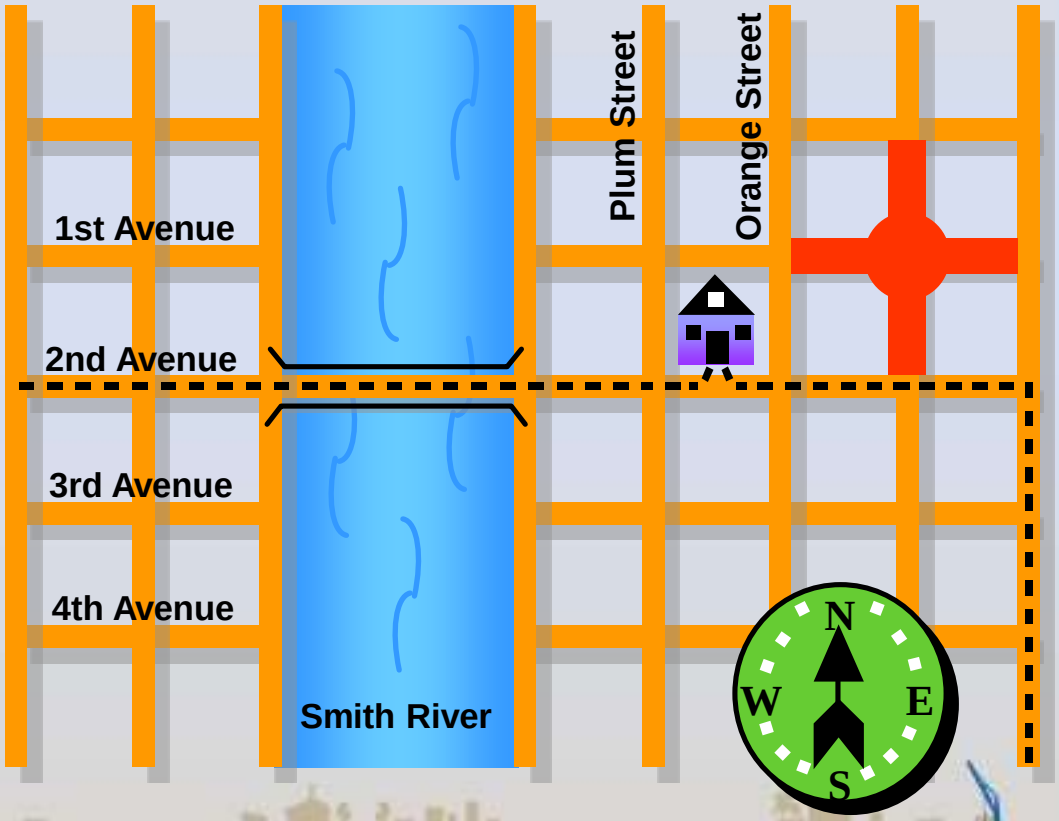
Framework podrška metodologijama

Metodologija

daje tačne smernice za poznato odredište

Framework

kao kompas, potvrđuje napredovanje i daje upravljačke smernice

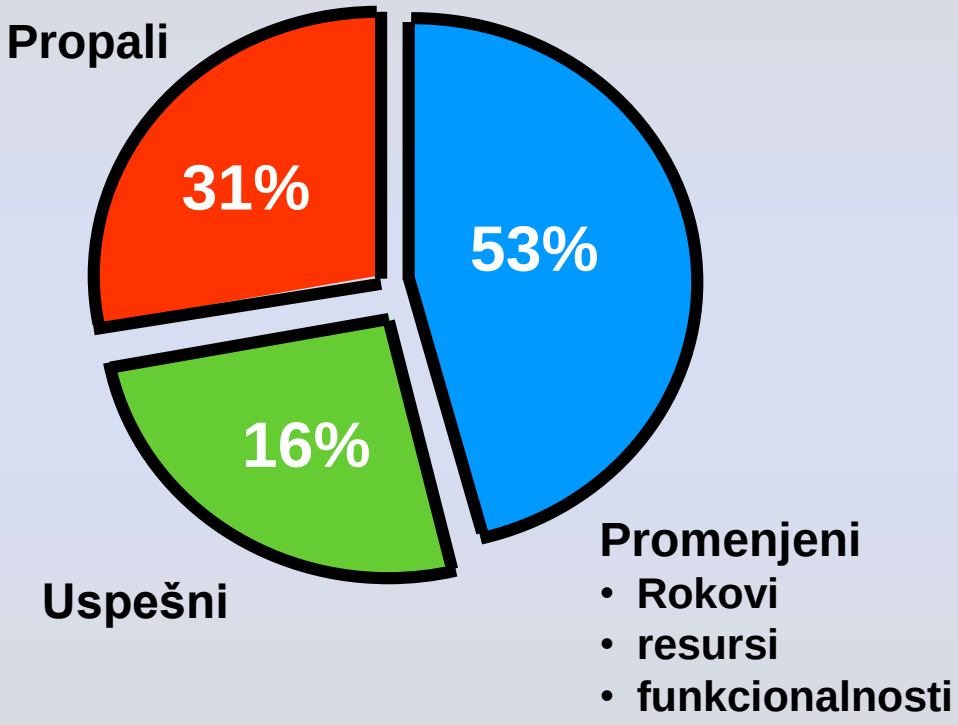


MSF

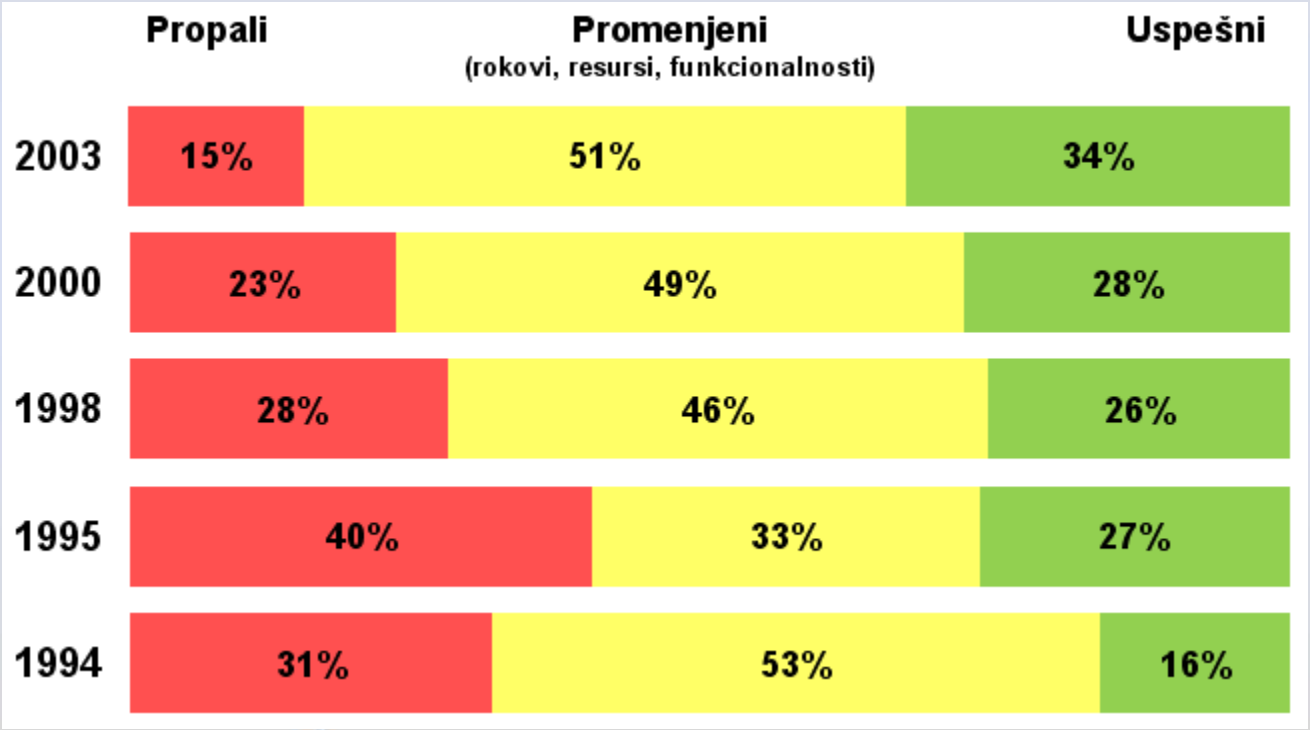
Zašto MSF?

- 16 % uspešnih projekata
- 31% neuspelih projekata
- 53% projekata koji su uspeli, ali su probili rokove ili su isporučeni sa lošijom funkcionalnošću i dr.

- Sa MSF-om zabeležen je rast od 20% uspešnih projekata sa svakodnevnom tendencijom porasta.



Procentualni prikaz uspešnosti projekata za period od 1994 do 2003



MSF discipline razvoja poslovnih rešenja

Upravljanje timom

MSF

Upravljanje rizicima

Upravljanje procesima

Upravljanje rizicima
Risk management

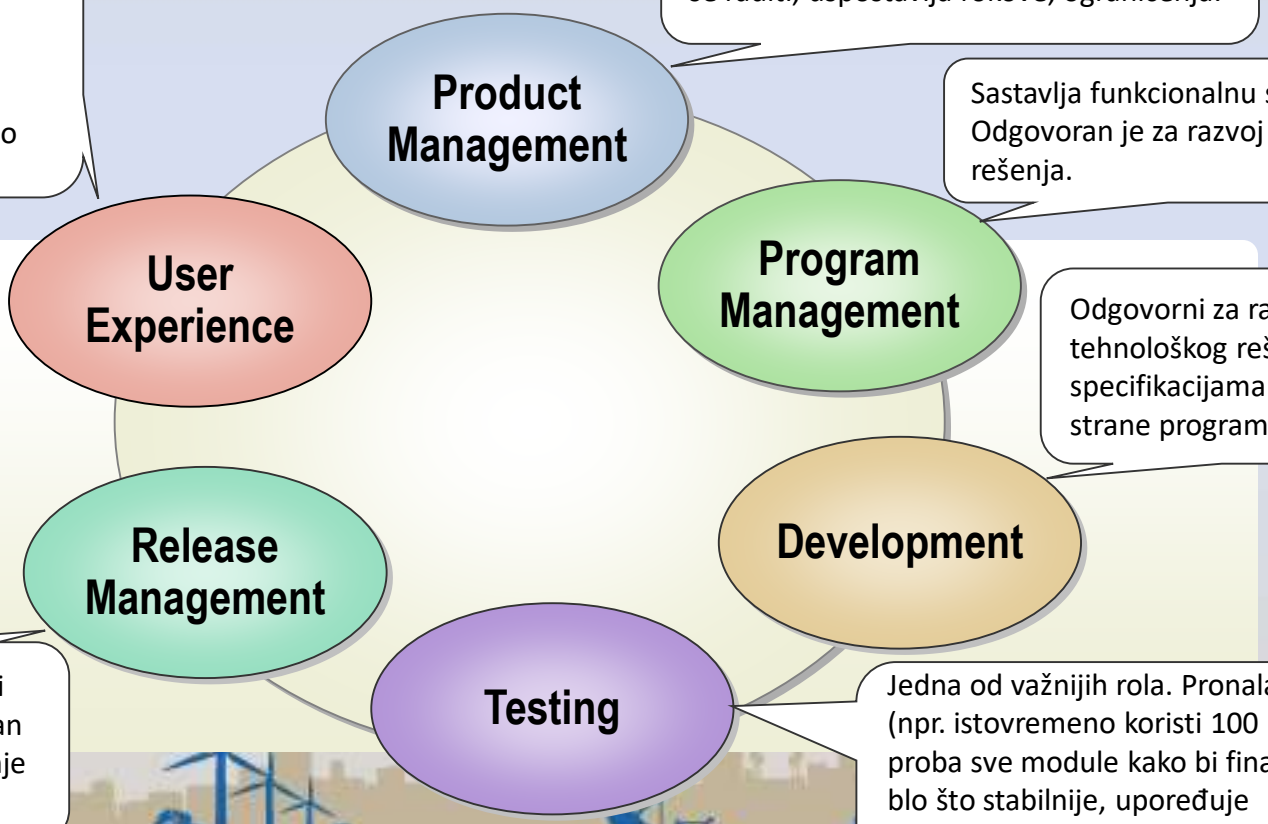
- podržava pro-aktivno upravljanje rizicima, kontinualnu ocenu rizika i odlučivanje tokom životnog ciklusa projekta.
- Projektni tim upravlja rizicima tako što kreira dokument procene rizika, identifikuje i dokumentuje sve moguće rizike i ocenjuje rizike prema verovatnoći pojave i uticaja na projekat.

Sadržaj	Opis
Opis rizika	Priroda svakog rizika
Verovatnoća rizika	Verovatnoća da se rizik dogodi (u skali od 1 do 10)
Mera rizika	Uticaj rizika na sistem
Izloženost riziku	Sveukupna pretnja rizika (verovatnoća * uticaj = izloženost riziku)
Planovi umanjenja	Planovi zaštite ili minimizovanja rizika
Planovi akcija	Koraci koji se preduzimaju kada se rizik dogodi
Odgovorni za rizik	Naziv osobe koja je zadužena i odgovorna za upravljanje rizikom

Primer rangiranja rizika prema prioritetu

	A	B	C	D	E	F	G	H	I	J
1	**Lista ocene rizika**									
2										
3		**Opis rizika**		(Skala 1-100%)	(Skala 1-10)					
4	**Rbr**	**Stanje**	**Posledice**	**Verovatnoća**	**Uticaj**	**Izloženost**	**Umanjenje**	**Kontigencija**	**Trigeri**	**Odgovorni**
5	1	Korisnici neće biti dostupni 50% od zahtevanog vremena.	Može doći do odlaganja projekta. Mogu se doneti nekorektne odluke.	80%	9	7,2	Nabaviti pismeni zahtev za angažovanje ugovorenog korisničkog vremena.	Sortirati pitanja prema prioritetu, koja će biti postavljena korisnicima. Odrediti druge izvore informacija.	Početak projekta	Program menadžer
6	2	Mogući napad radi blokiranja usluga (*denial of service attack* - DoS).	Sistem je nedostupan klijentima i krajnjim korisnicima.	70%	10	7	Ispravljanje sigurnosnih propusta, korišćenje *firewall* tehnologije i procedura za ažuriranje (*update*) tehnologija.	Upotreba softvera za detekciju radi zaštite od DoS-a. Razvijati, testirati i redovno pregledati procedure oporavka.	Početak projekta	IT menadžer, menadžer mreže
7	3	Ukoliko prodaja opadne, budžet projekta će biti umanjen za 10,000e ili će biti otkazan.	Otkazivanje projekta, odlaganje ili redukcija granica projekta.	50%	10	5	Nisu moguće akcije unutar granica projekta.	Razmotriti odlaganje nekih funkcionalnosti za narednu verziju rešenja.	Kada su poznate izmene budžeta projekta.	Proizvodni menadžer
8	4	Nepotpuni podaci o klijentima dovode do otežane analize.	Nemogućnost identifikovanja najboljih klijenata.	80%	6	4,8	Utvrditi opseg problema i razmotriti načine prikupljanja podataka koji nedostaju.	Iščistiti podatke izvlačenjem iz živog sistema.	Kada je donešena odluka o procentu nekompletnih podataka i njihovog uticaja na sistem.	Proizvodni menadžer
9	5	Može se dogoditi da administrator baze podataka ne bude na raspolaganju 50% od ugovorenog vremena.	Projekat bi mogao da se odloži. Mogu biti donešene nekorektne odluke.	30%	6	1,8	Obavezati administratora na ugovoreno vreme.	Sortirati pitanja prema prioritetu i ukoliko je moguće odrediti druge izvore informacija.	Početak projekta	Program menadžer

MSF timski model



Vodi početne razgovore sa naručiocima posla, prikuplja zahteve, dogovara šta će se raditi, uspostavlja rokove, ograničenja.

Sastavlja funkcionalnu specifikaciju. Odgovoran je za razvoj i isporuku rešenja.

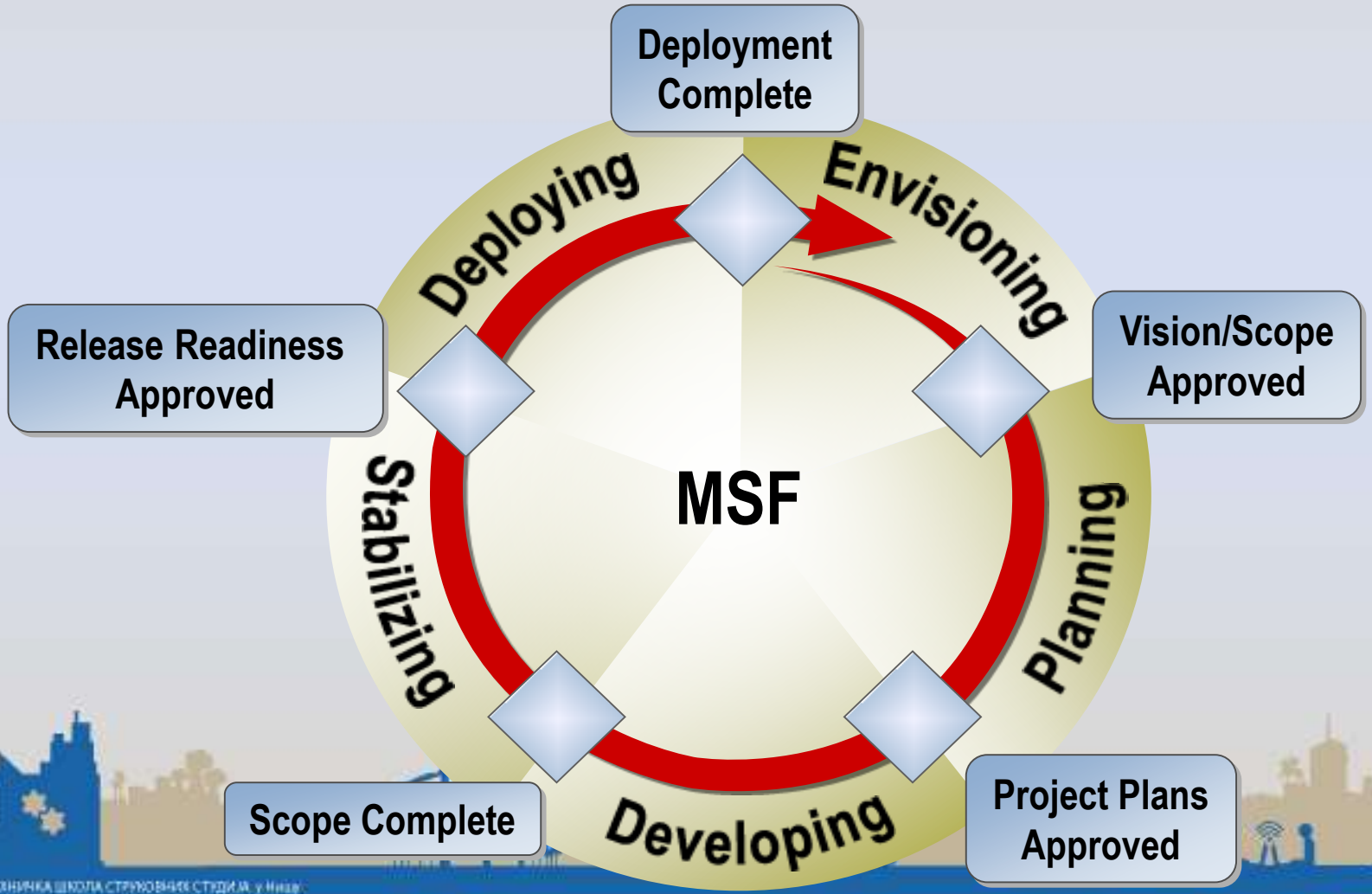
Odgovorni za razvoj tehnološkog rešenja prema specifikacijama dobijenih od strane program menadžera.

Jedna od važnijih rola. Pronalazi bug-ove (npr. istovremeno koristi 100 korisnika), proba sve module kako bi finalno rešenje blo što stabilnije, upoređuje funkcionalnost programa sa ciljevima i vizijom projekta.

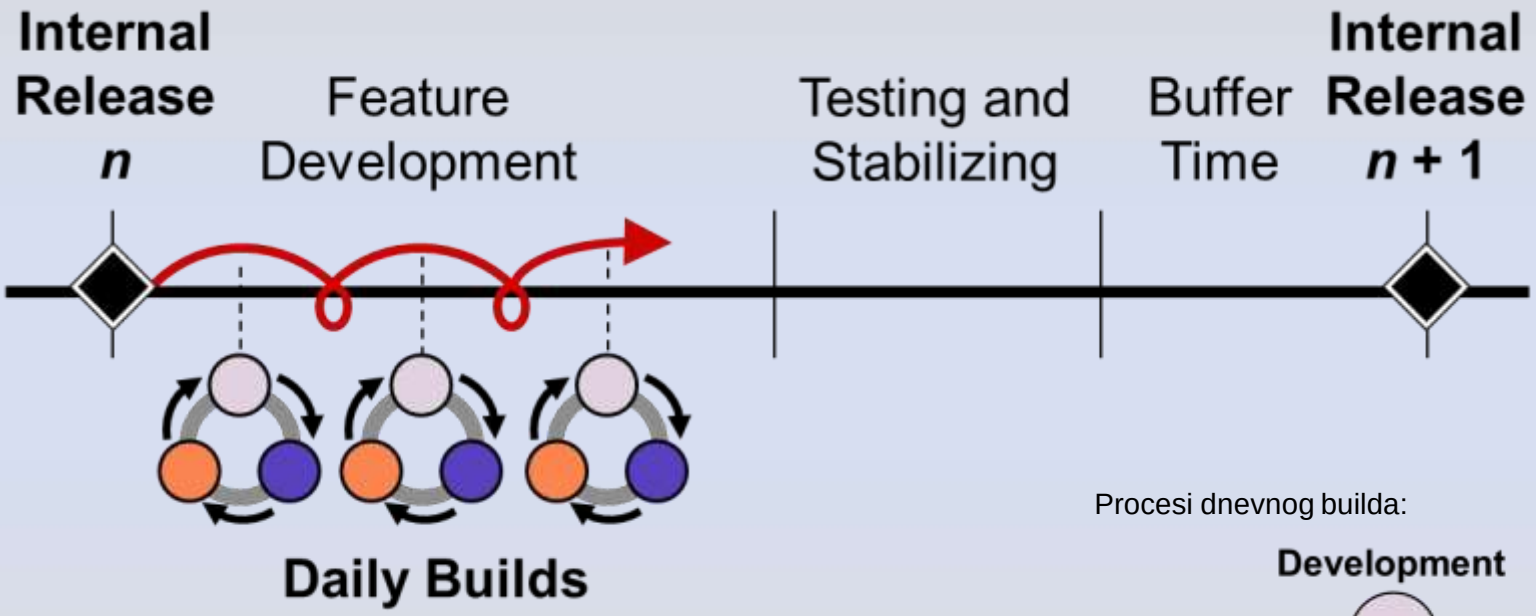
Pravi *Help*, implementira i instalira sistem. Odgovoran je za rukovanje i održavanje sistema.

Poznaje obe strane procesa, proizvodnje i upotrebe. Unosi podatke, analizira performanse, koriguje ekrane kako bi rešenje bilo što upotrebljivije...

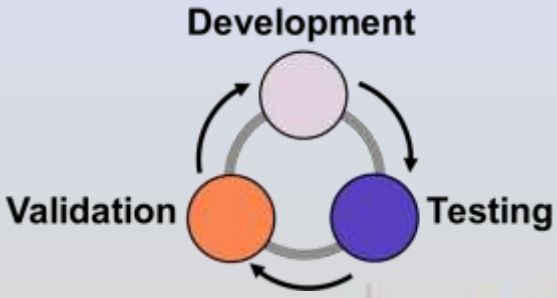
Faze procesa razvoja poslovnog rešenja



Razvoj u više internih verzija



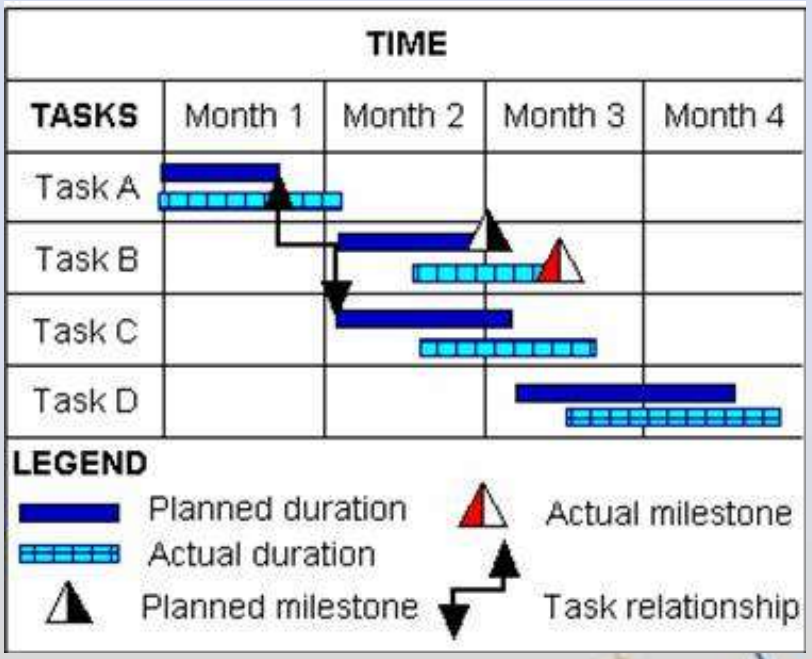
Procesi dnevnog builda:



MSF *Milestones*

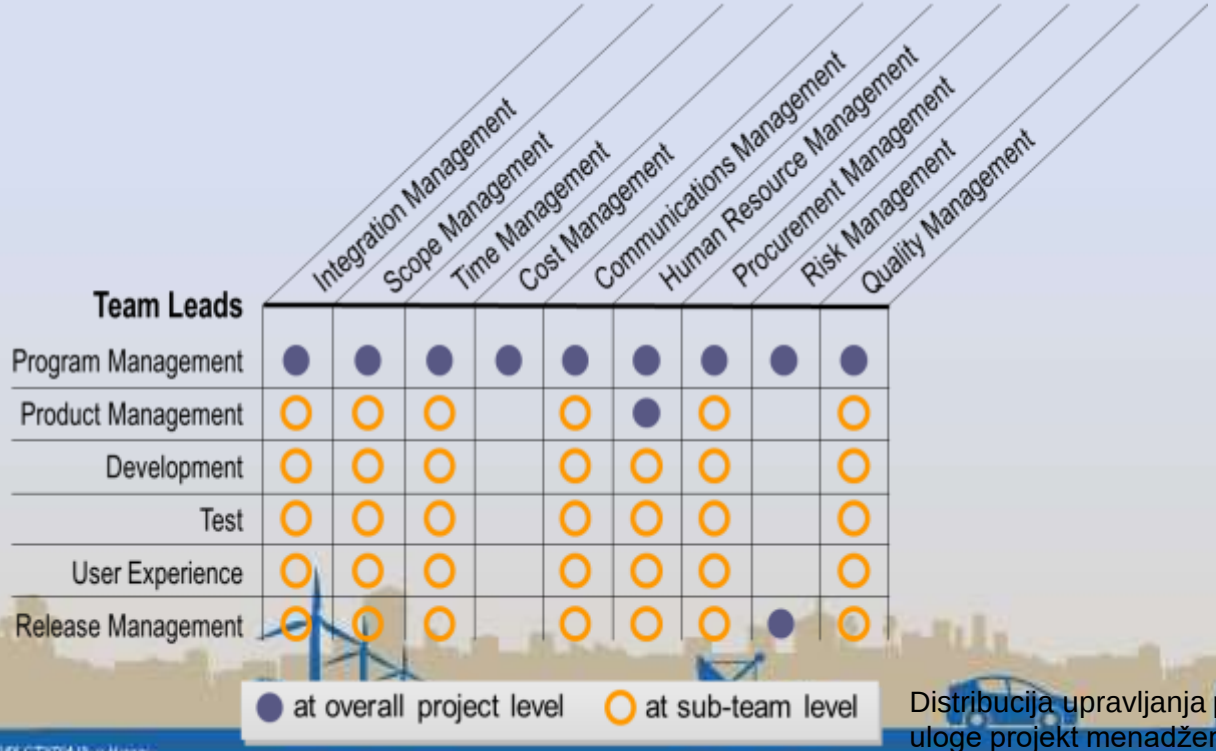
Upravljanje projektima
Project Management

- Zadaci:
  - Integracija planiranja i upravljanje promenama
  - Defininisanje i upravljanje ciljevima i oblašću projekta
  - Pripremanje budžeta i upravljanje troškovima
  - Pripremanje i praćenje rasporeda odvijanja projekta
  - Osigurati da su odgovarajući resursi alocirani na projektu
  - Olakšati timsku i spoljnu komunikaciju
  - Olakšati proces upravljanja rizicima
  - Dokumentovanje i nadgledanje procesa upravljanja kvalitetom tima…



Projekt menadžer

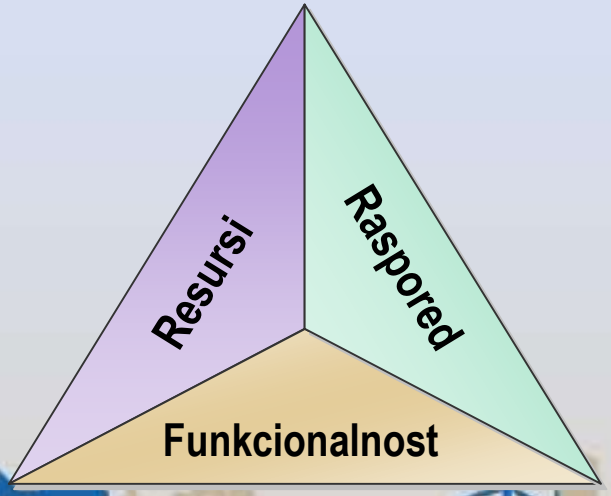
- Ne postoji striktna hijerarhija uloga, već svi zajedno učestvuju u razvoju svake faze i svi su odgovorni za određene delove projekta
- Ukoliko pak demokratski ne može da se reši problem, finalnu reč ima menadžer proizvoda



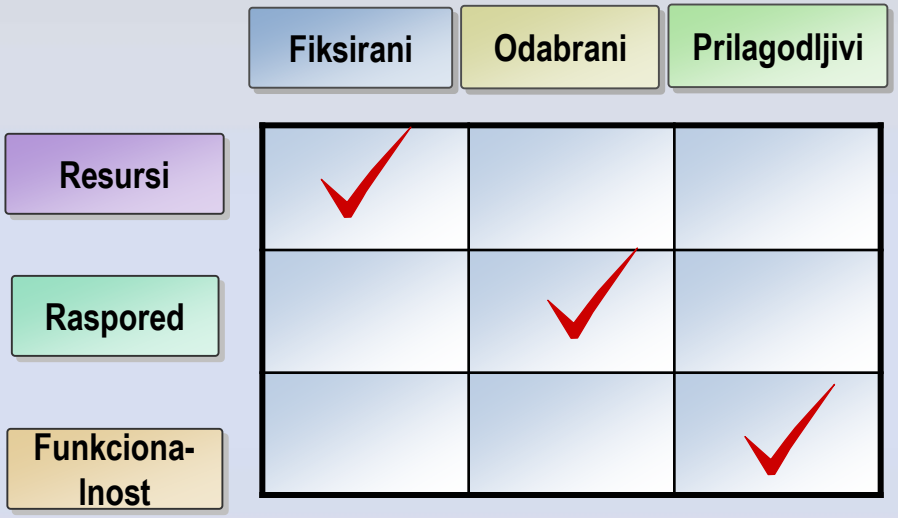
Distribucija upravljanja projektom i uloge projekt menadžera

Kako upravljati kompromisima?

- Da bi uspešno definisali cilj i upravljali projektom, neophodno je:
  - identifikovati ograničenja projekta
  - upravljati kompromisima
  - uspostaviti kontrolu promena
  - pratiti napredak projekta



a) Trougao kompromisa



b) Matrica kompromisa

- U projektima postoji jasan odnos između resursa, rasporeda i funkcionalnosti projekta
- S obzirom da je skoro nemoguće ostvariti istovremeno sve ciljeve, neophodno je upravljati kompromisima

RUP metodologija
Rational Unified Process

- Objedinjeni proces
- RUP faze
- Najbolje prakse

Objedinjeni proces
Unified Process

- nastaje ujedinjavanjem pristupa projektovanja softverskih sistema koji su formulisani od strane stručnjaka: **Jacobson, Booch, Rumbaugh**
- Glavni sponzor razvoja ove metodologije je bila firma *Rational* (vodeći prodavac CASE alata) otuda naziv **Rational Unified Process** ili kraće RUP
- RUP se oslanja na **UML (*Unified Modelling Language*)**
- UML služi za:
  - **Specifikaciju**
  - **Vizuelizaciju**
  - **Konstrukciju**
  - **Dokumentaciju** razvoja softvera

Dr Ivar Jacobson je otac komponenata i komponentne arhitekture, slučajeva korišćenja, UML-a i RUP-a

Grady Booch je lider u objektno-orijentisanoj analizi i razvoju UMLa

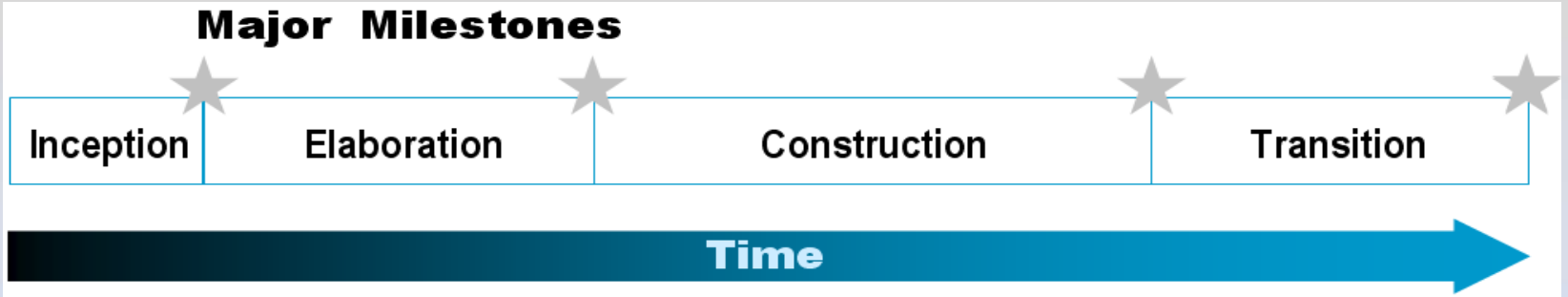
James Rumbaugh je kreator tehnike objektnog modelovanja (OMT) i UML-a

Rational Unified Process

RUP je pristup razvoja softvera koji je iterativan, centriran oko arhitekture i vođen slučajevima korišćenja

- **Osnovni koncepti:**
  - **Iterativno inkrementalni** proces
  - ***Milestone*** (kontrolna/ključna/kritična tačka)
    - Svaka faza u razvoju projekta se završava sa nekim *milestone*-om koji sumiraju rezultate svih prethodnih iteracija i u njoj se donose značajne odluke za ceo projekat u celini
  - **Role** - definišu ponašanja i odgovornosti pojedinca ili tima
  - **Artifakti**:
    - **Dokumenta**: beleže zahteve sistema, kao i upotrebljivost, pouzdanost, performanse i podršku zahtevima
    - **Modeli**: pojednostavljeni pogled sistema koji prikazuje osnovni sistem bez nepotrebnih detalja
    - **Elementi modela**: pomaže timu da vizuelizuju, konstruišu i dokumentuju strukturu i ponašanja sistema

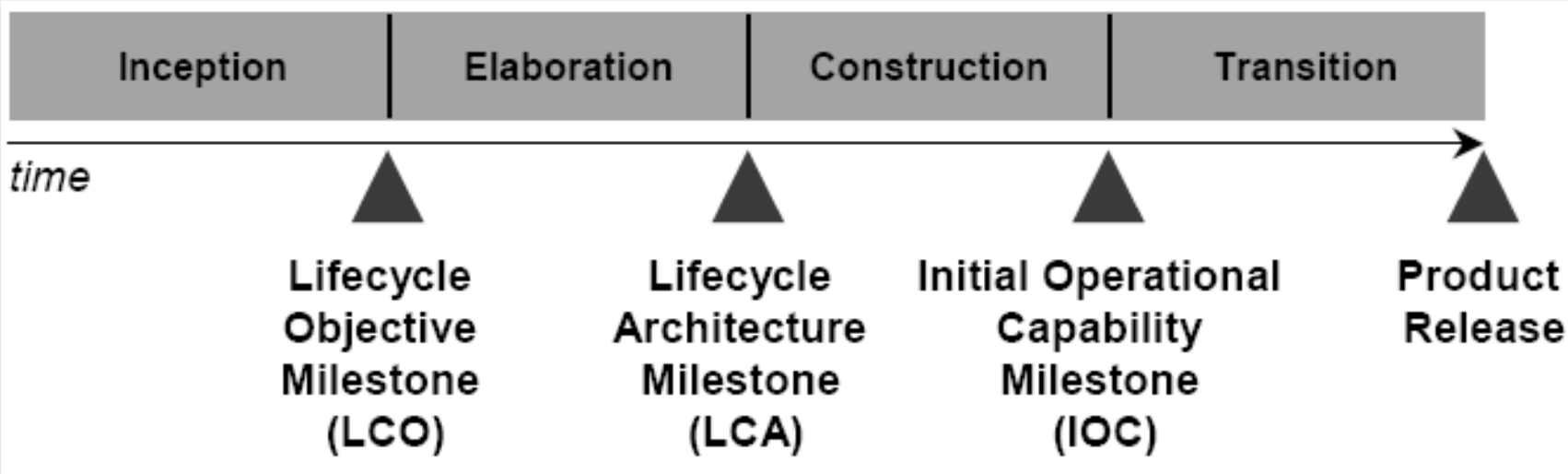
Faze iterativnog razvoja



- **Početna faza (*Inception*)**
  Razumeti **šta** treba graditi:
  - Vizija, zahtevi visokog nivoa, poslovni **slučajevi korišćenja**
  - Identifikovanje **rizika**
  - Procena **troškova**, **vremena**, plana i kvaliteta proizvoda
  - Inicira se kreiranje poslovne studije **opravdanosti** ulaženja u projekat

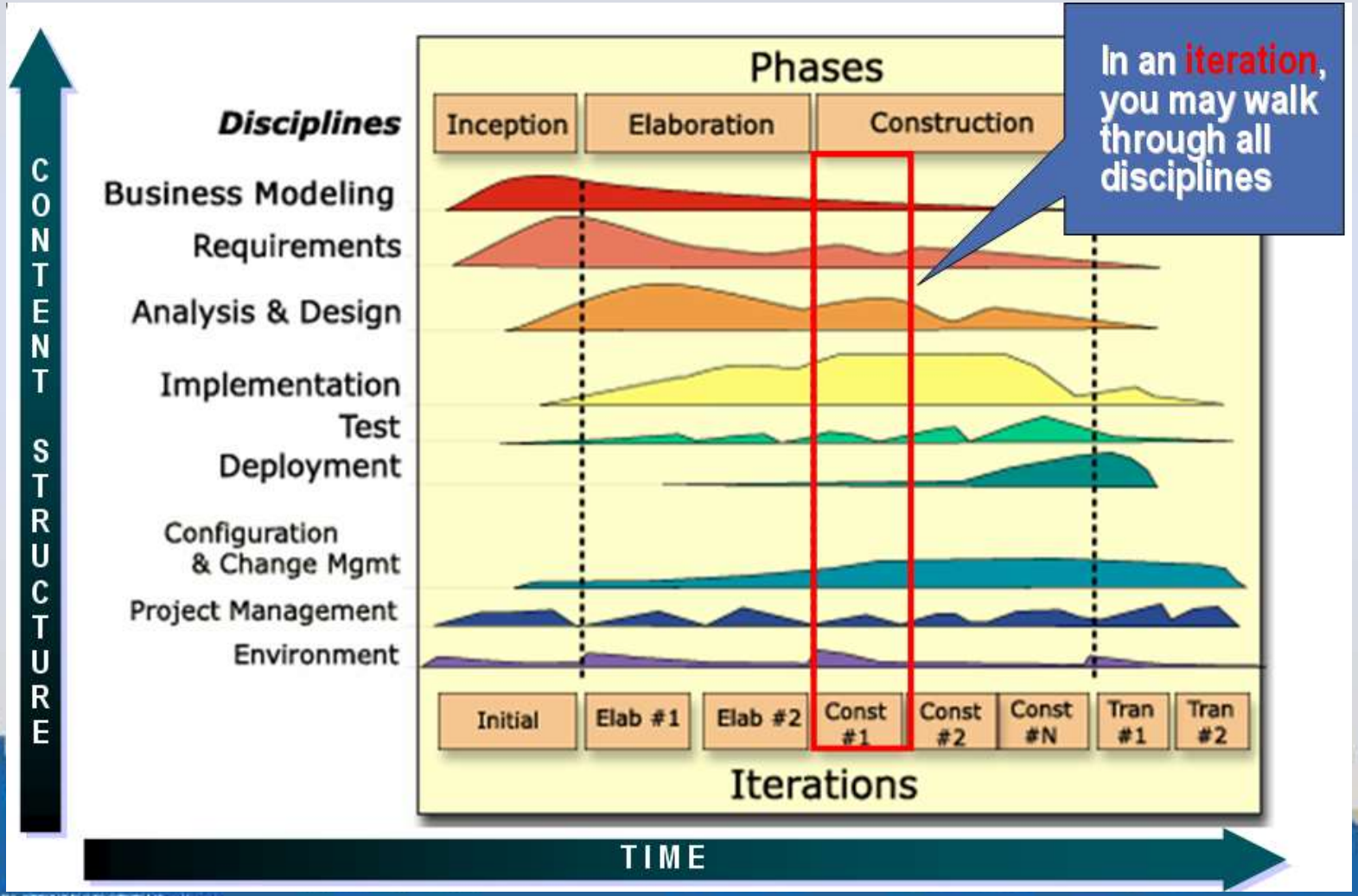
- **Faza elaboracije (*Elaboration*)**
  Razumeti **kako** treba graditi:
  - Osnovna **arhitektura**
  - **Detaljno opisani zahtevi**
  - Dizajn nije detaljan
  - Precizna procena resursa i vremena
- **Konstrukcija (*Construction*)**
  Izgradnja proizvoda - završeno testiranje sistema
- **Tranzicija (*Transition*)**
  Validacija rešenja - Prihvatljivost od strane stejkoldera

Glavne kontrolne tačke

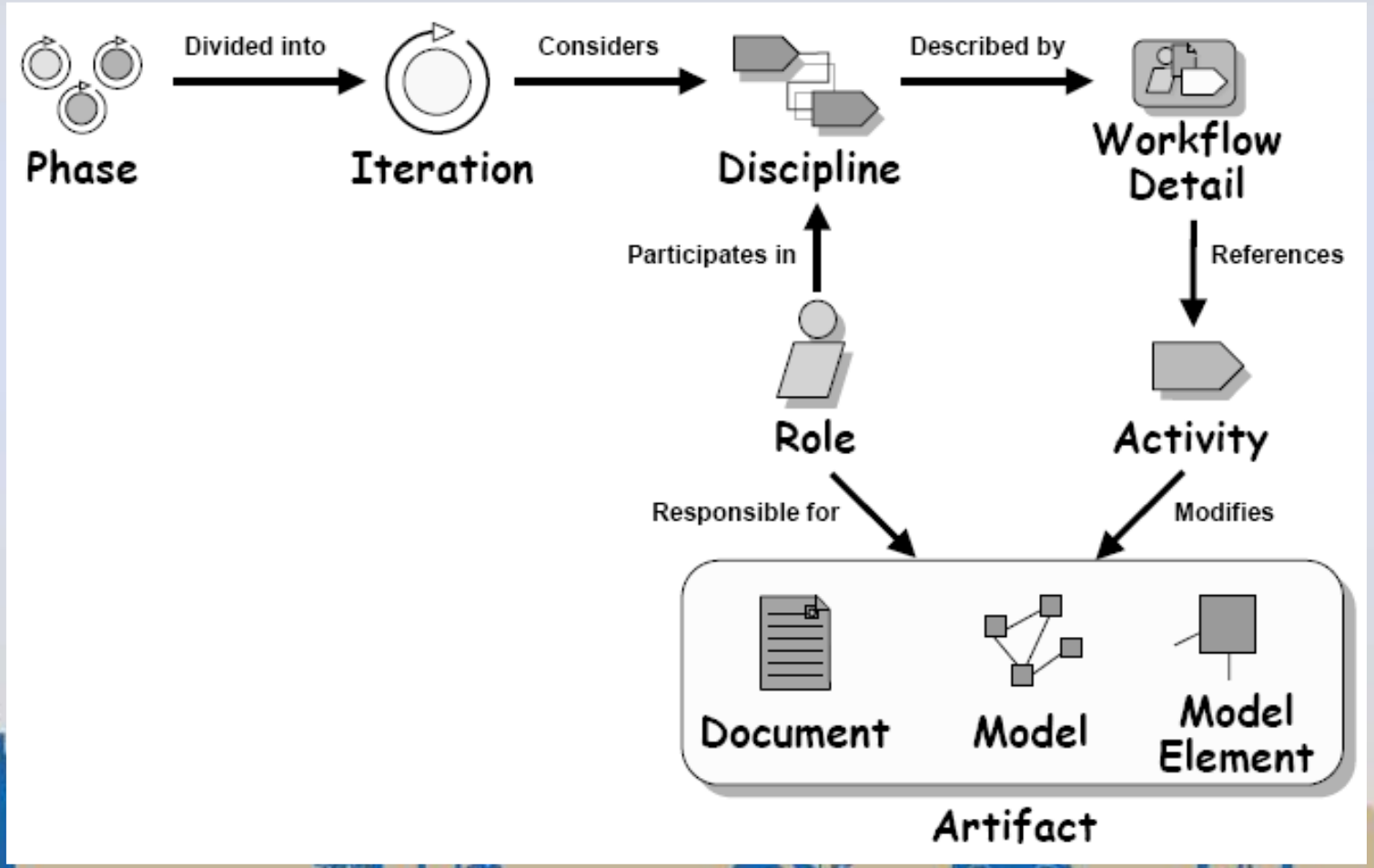


- LCO: dogovoreno je oko domena i rizici su razumljivi i prihvatljivi
- LCA: visoki rizici su rešeni i arhitektura je stabilna
- IOC: proizvod je završen i kvalitet je prihvatljiv

Graf iterativnog životnog ciklusa

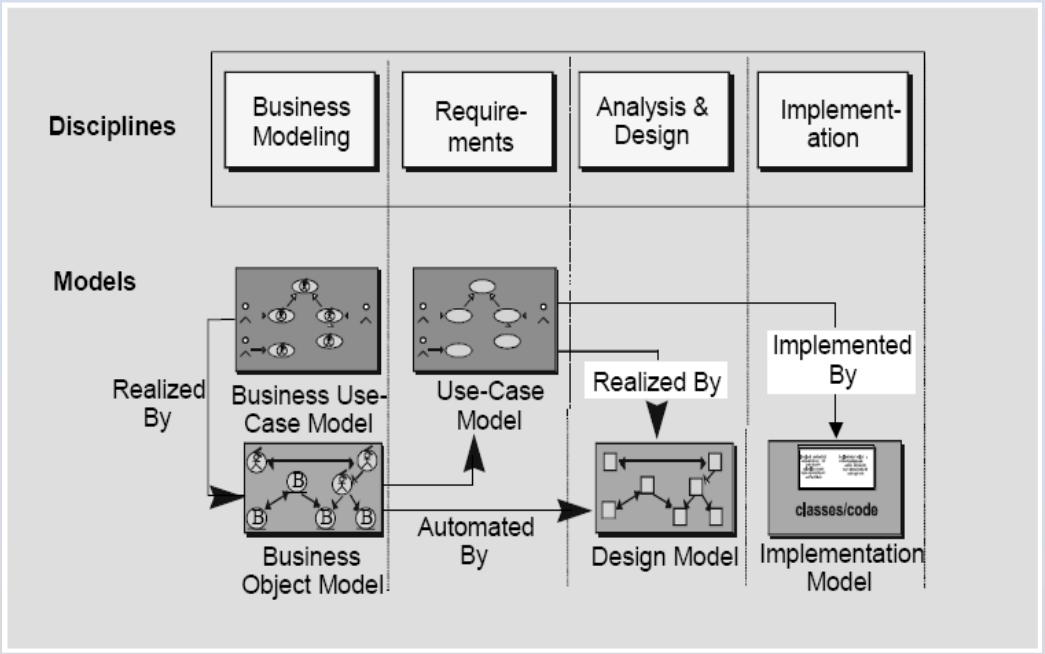


Kako radi RUP?



Šta su RUP modeli?

- **Da bi se sistem kompletno opisao, potrebno je nekoliko modela**
- Modeli se razvijaju inkrementalno kroz iteracije:
  - **Biznis model** – opisuje poslovne procese i poslovno okruženje
  - **Model slučajeva korišćenja** – opisuje šta sistem treba da radi i sistemsko okruženje
  - **Model projektovanja (dizajn model) -** opisuje realizaciju slučajeva korišćenja i služi kao apstrakcija izvornog koda
  - **Model implementacije** - kolekcija komponenata i podsistema

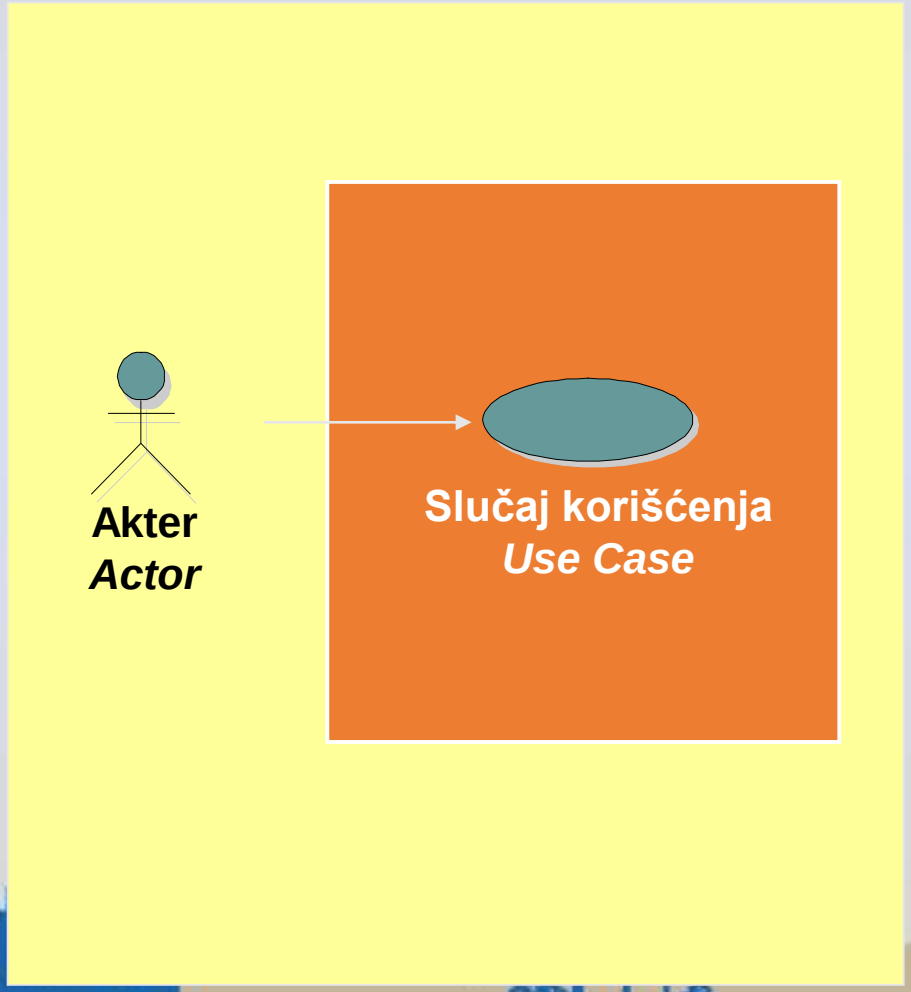


Početna faza (*Inception*)
Šta razvijate?

Inception	Elaboration	Construction	Transition

- **Cilj 1: Razumeti šta se razvija**
  - Navesti ključne aktere i slučajeve korišćenja
    - Identifikovati sve aktere
    - Pridružiti svakog aktera slučajevima korišćenja
    - Ukratko objasniti svakog aktera i slučajeve korišćenja sa 1 do 2 rečenice
    - Napraviti rečnik - definiše opštu terminologiju za sve modele i sadrži tekstualni opis zahtevanog sistema
    - Identifikovati najosnovnije i kritične slučajeve korišćenja (<20% slučajeva korišćenja)
  - Odrediti domen sistema
  - Navesti detaljne ključne nefunkcionalne zahteve

Osnovni koncepti u modelu slučajeva korišćenja

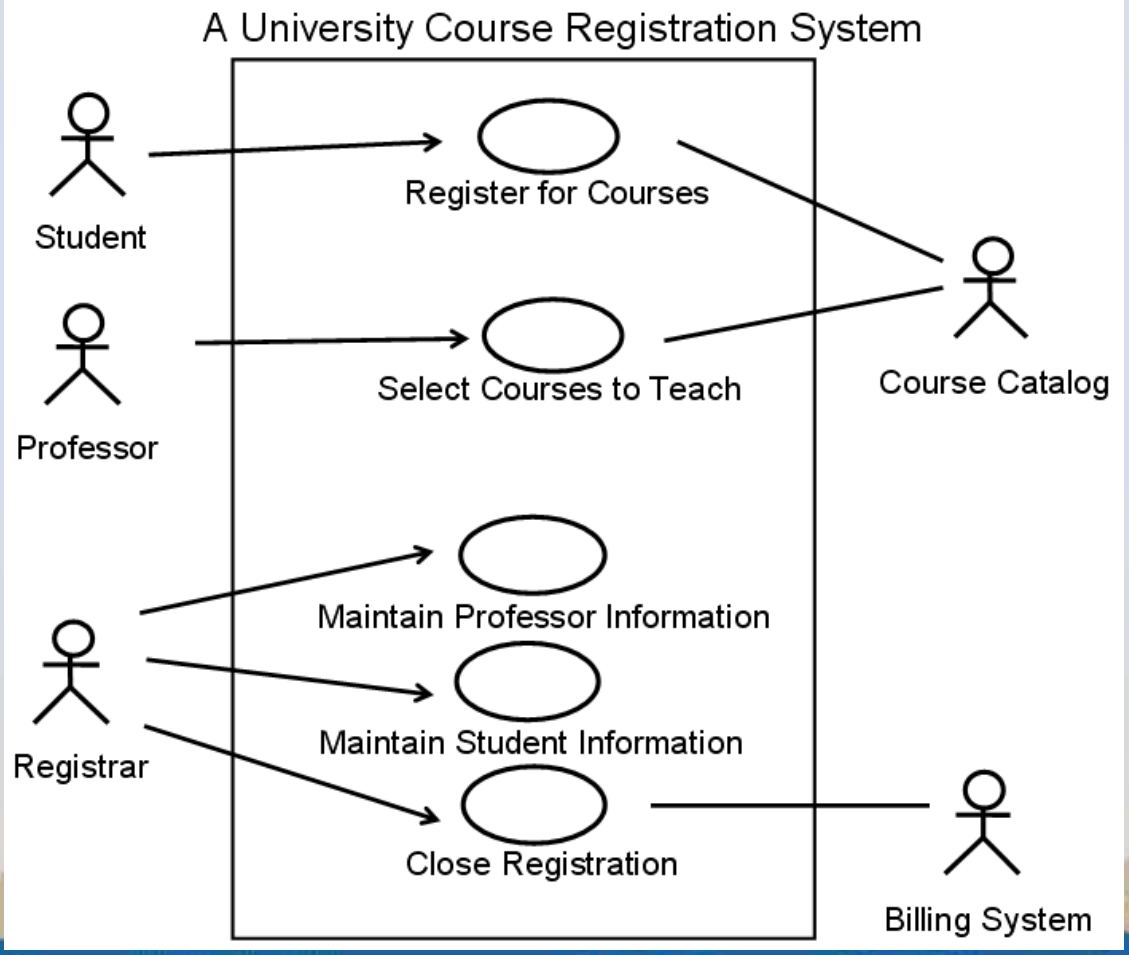


- **Akter** (učesnik) predstavlja osobu ili drugi sistem koji je u interakciji sa sistemom
- **Slučaj korišćenja** (*use case*) definiše sekvencu akcija koje sistem izvršava, a predstavlja **rezultat od određene vrednosti za aktera**.
- predstavlja glavni deo neke **komplentne funkcionalnosti** od početka do kraja

  Na primer:
  - Student treba da izabere kurseve za semestar, da se upiše i uplati izabrane kurseve (use case: registrovanje na kurseve)
  - Sekretar treba da dodaje, briše i modifikuje kurseve (use case: održavanje nastavnog plana, jer započinje isti akter i radi sa istim entitetima u sistemu)

Dijagram slučajeva korišćenja
Use Case Model

- Opisuje funkcionalne zahteve sistema u terminima slučajeva korišćenja



Ciljevi početne faze (nastavak)

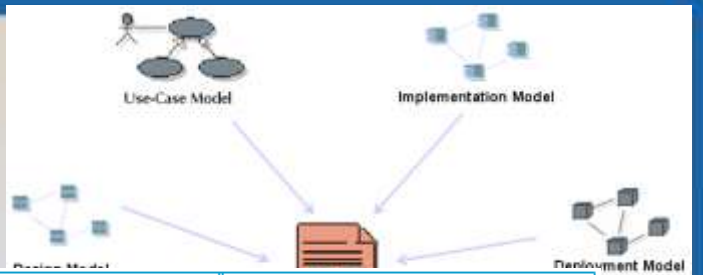
- **Cilj 2: Identifikovati ključne zahteve**
  - Iscrtati **ključne interfejse**
  - Navesti **rizike** koji se odnose na performanse, redundnantnost, bezbednost podataka ...
- **Cilj 3: Odrediti najmanje jedno potencijalno rešenje**
  - Da li ste razvijali slične sisteme?
  - Sa kojom arhitekturom i sa kojim troškovima?
  - Da li će trenutna arhitektura raditi ili će biti potrebno redizajnirati?
  - Da li ste u mogućnosti da angažujete osoblje sa pravim kompetencijama?
  - Da li se zahtevaju softverske komponente?
  - Da li se mogu nabaviti pri prihvatljivim troškovima?

Ciljevi početne faze (nastavak)

Inception	Elaboration	Construction	Transition

- **Cilj 4: Razumevanje troškova, rasporeda i rizika**
  - Da li treba da finansiramo projekat?
  - Troškovi
  - Povratak investicija (ROI)
  - Plan projekta
  - Potreba za resursima
- **Cilj 5: Razumevanje koje procese pratiti i koje alate koristiti**

Faza elaboracije
Kako razvijate?



Inception	Elaboration	Construction	Transition

- **Cilj 1: Detaljnije razumevanje zahteva**
  - Sa 20% slučajeva korišćenja obuhvatiti 80% zahteva (proći sa stejkholderima)
- **Cilj 2: Projektovati, implementirati, vrednovati i postaviti arhitekturu**
  - Dokument arhitekture softvera
- **Cilj 3: Umanjiti osnovne rizike, napraviti precizniji raspored i proceniti troškove**
  - Tehnički rizici - umanjiti implementiranjem i testiranjem arhitekture
  - Poslovni rizici - implementiranjem i testiranjem ključnih funkcionalnosti sistema
  - Timski rizici – tim treba da prođe kroz ceo životni ciklus softvera implementirajem realnog koda
- **Cilj 4: Uvođenje razvojnog okruženja**
  - Obučiti osoblje ukoliko je potrebno

Faza konstrukcije
Razvite proizvod!

- **Cilj 1: Minimizirati troškove razvoja i dostići neki stepen paralelnog rada razvojnog tima**
  - Organizovati tim oko arhitekture softvera
  - Osigurati kontinuirani progres
  - Projektovati, implementirati i testirati svaku komponentu …
- **Cilj 2: Iterativno razviti kompletan proizvod koji je spreman za tranziciju u korisničko okruženje**
  - Razviti prvu operativnu verziju sistema (beta verzija)
  - Uključiti materijale za obuku, korisničku dokumentaciju i dokumentaciju uvođenja rešenja u sistem …

Višestruke iteracije

	Elaboration (End)	Construction 1	Construction 2	Construction 3
Require-ments	▪ 15 UCs identified ▪ 8 detailed ▪ 4 some depth ▪ 3 briefly	▪ 12 UCs detailed ▪ 3 some depth	▪ 14 UCs detailed ▪ 1 removed (scope reduction)	▪ 14 UCs detailed
Comp-onents	▪ 18 main components ▪ 4 -> 50% done ▪ 10 –> interfaces ▪ Lower arch. layers almost done ▪ All code unit tested	▪ 19 main components ▪ 3 almost done ▪ 8 -> 50% ▪ 6 -> interfaces ▪ Lower arch. layers almost done ▪ All code unit tested	▪ 18 main components ▪ 10 almost done ▪ 8 -> 50% ▪ All code unit tested	▪ 18 main components ▪ All almost done, minor tuning left ▪ Beta release, all functionality implemented
Tests	▪ Initial load & performance test of arch. ▪ 4 UCs functionally tested	▪ Continued load & perf. testing ▪ Functional UC testing as completed	▪ Continued load & perf. testing ▪ Functional UC testing as completed	▪ Continued load & perf. testing ▪ Functional UC testing as completed

Faza tranzicije
Uvedite (*Deploy*) sistem kod korisnika!

- **Cilj 1: Beta test za vrednovanje očekivanja korisnika -** Ispraviti bagove
- **Cilj 2: Obučavanje korisnika**
- **Cilj 3: Pripremiti uvođenje sistema –** Nabaviti hardver, migrirati podatke ...
- **Cilj 4: Poboljšati buduće performanse projekta kroz naučene lekcije**

Simptomi problema u razvoju softvera

- ✓Nepoznavanje potreba korisnika ili poslovanja
- ✓Neukazivanje na zahteve
- ✓Moduli nisu integrisani
- ✓Poteškoće u održavanju
- ✓Kasno otkrivanje grešaka
- ✓Slab kvalitet sa aspekta krajnjih korisnika
- ✓Loše performanse
- ✓Nekoordinisan rad tima

Simptomi – uzroci – najbolje prakse

Simptomi

- Nepoznavanje potreba
- Neukazivanje na zahteve
- Neodgovarajući moduli
- Teško održavanje
- Kasno otkrivanje grešaka
- Loš kvalitet
- Loše performanse
- Sukobljavanje developera

Uzroci

- Nedovoljno zahteva
- Nejasna komunikacija
- Nestabilna arhitektura
- Kompleksnost
- Neotkrivena nekonzistentnost
- Slabo testiranje
- Subjektivne procene
- Vodopad razvoj
- Nekontrolisane promene
- Nedovoljno automatizacije

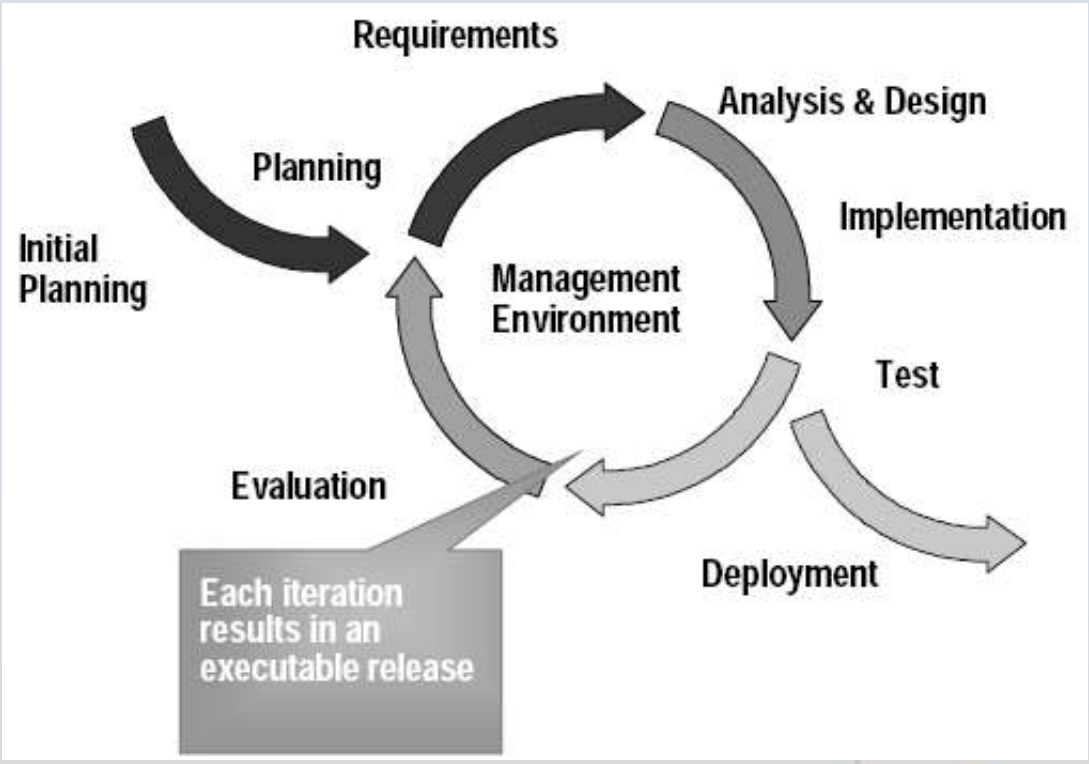
Najbolje prakse

- Iterativan razvoj
- Upravljanje zahtevima
- Korišćenje komponentne arhitekture
- Vizuelno modelovanje (UML)
- Kontinualna verifikacija kvaliteta
- Upravljanje promenama

Praksa 1: Iterativan razvoj

Najbolje prakse

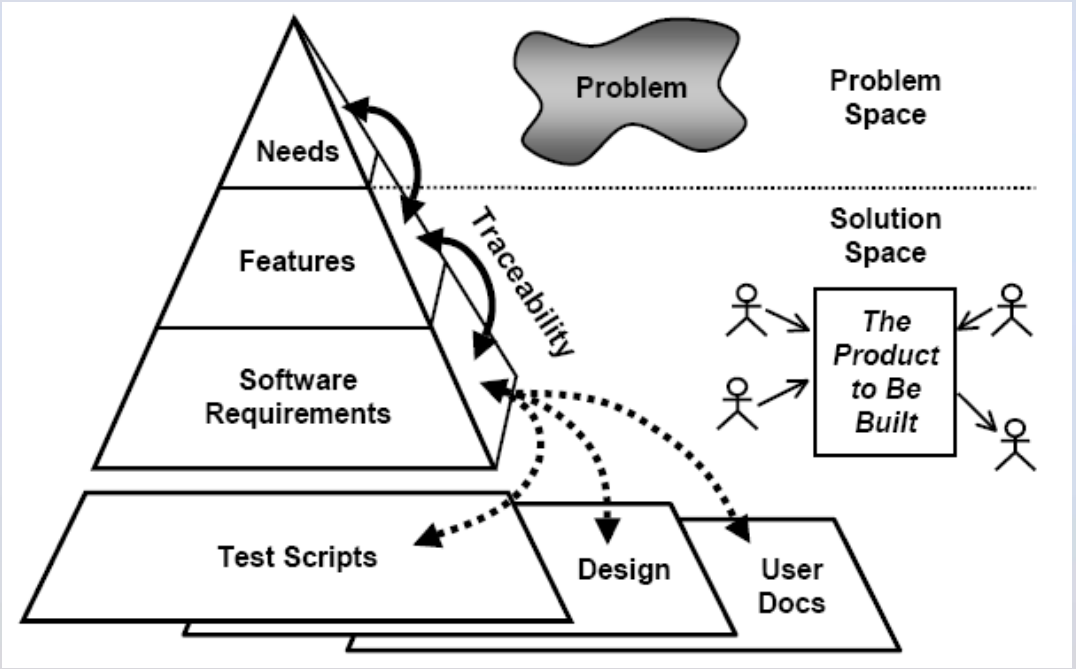
- **Iterativan razvoj**
- Upravljanje zahtevima
- Upotreba komponentne arhitekture
- Vizuelno modelovanje (UML)
- Kontinualna provera kvaliteta
- Upravljanje promenama



Praksa 2: Upravljanje zahtevima

Najbolje prakse

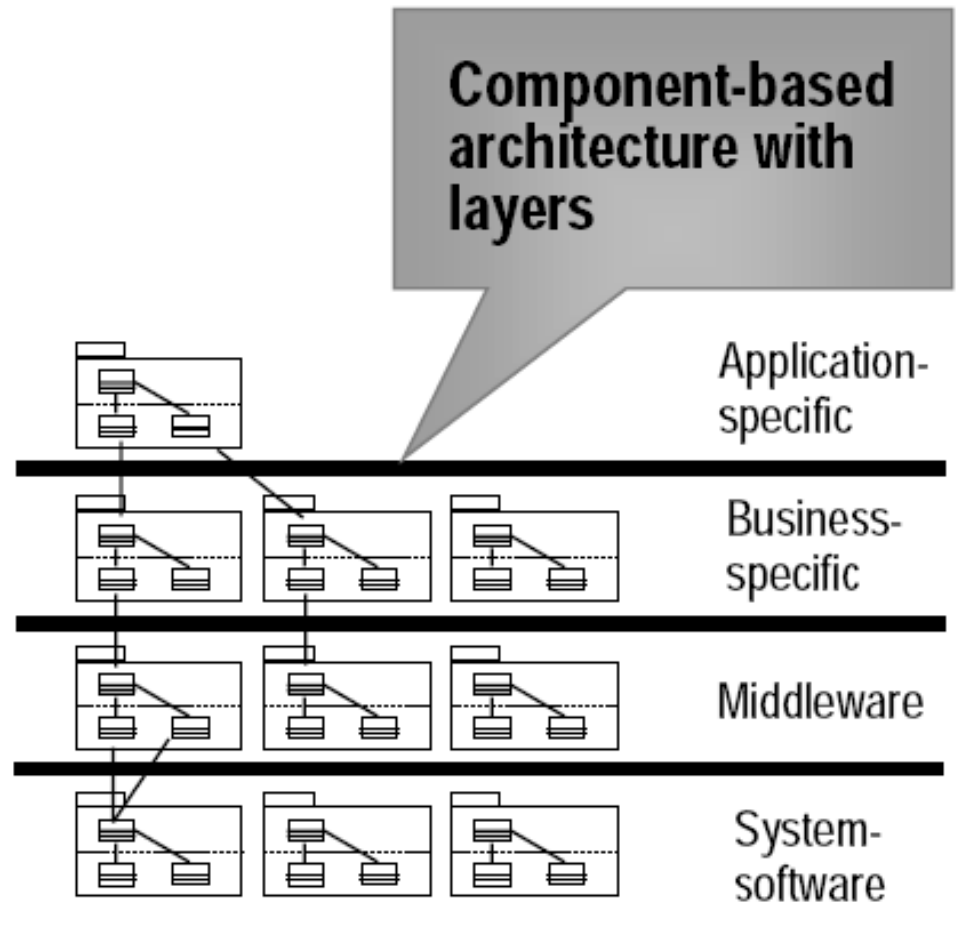
- Iterativan razvoj
- **Upravljanje zahtevima**
- Upotreba komponentne arhitekture
- Vizuelno modelovanje (UML)
- Kontinualna provera kvaliteta
- Upravljanje promenama



Praksa 3: Upotreba komponentne arhitekture

Najbolje prakse

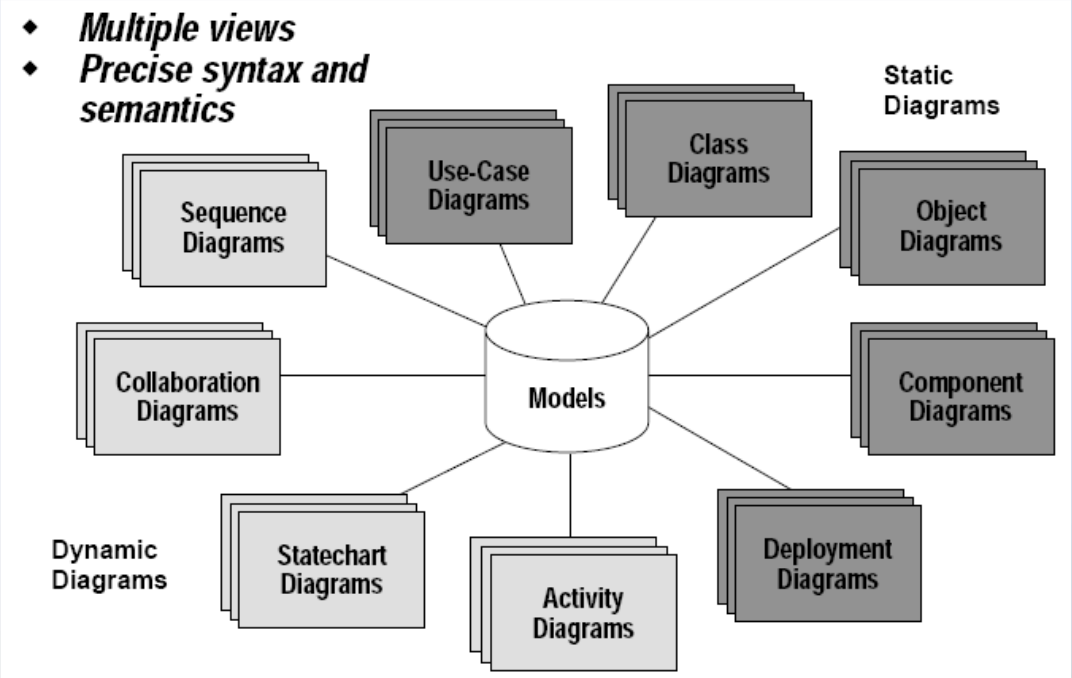
- Iterativan razvoj
- Upravljanje zahtevima
- **Upotreba komponentne arhitekture**
- Vizuelno modelovanje (UML)
- Kontinualna provera kvaliteta
- Upravljanje promenama



Praksa 4: Vizuelno modelovanje (UML)

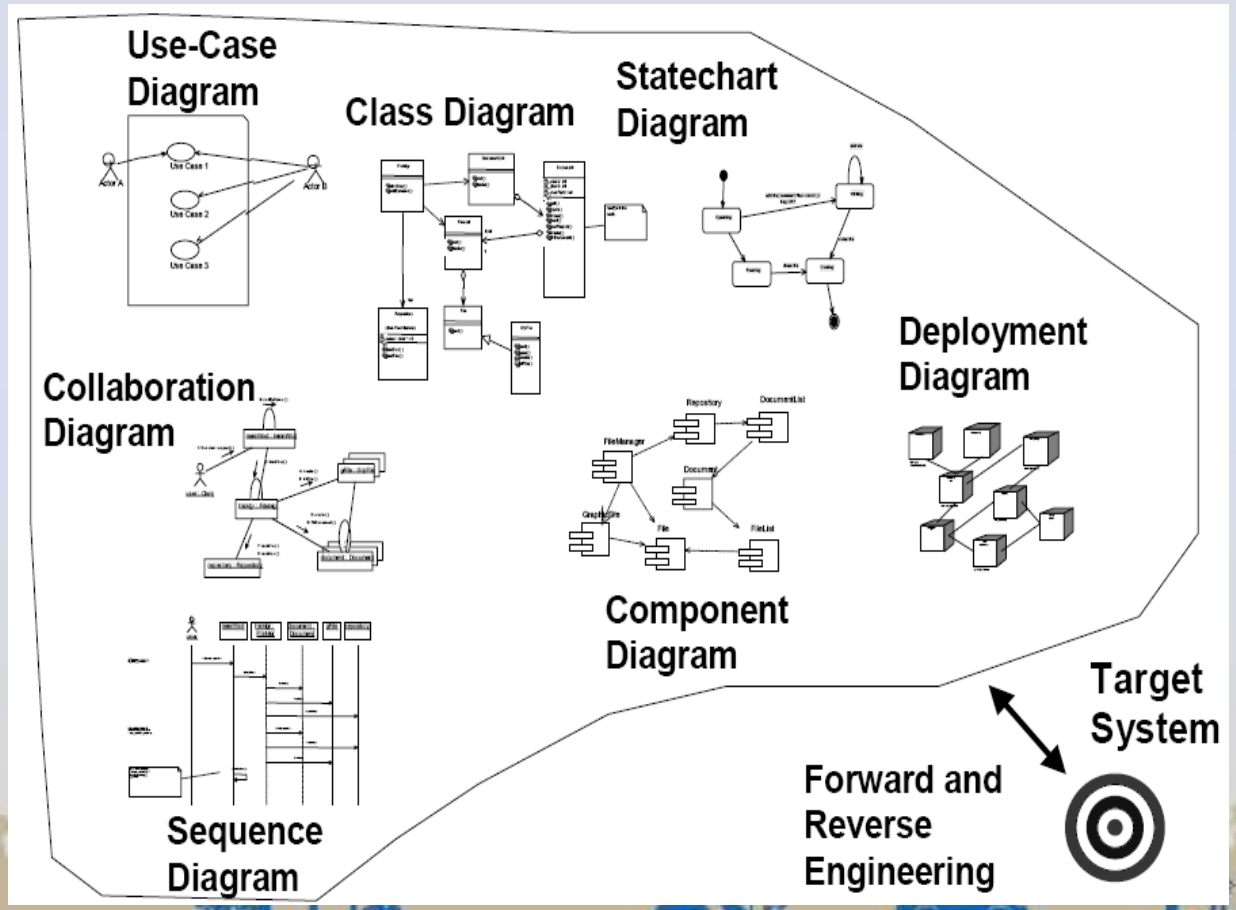
Najbolje prakse

- Iterativan razvoj
- Upravljanje zahtevima
- Upotreba komponentne arhitekture
- **Vizuelno modelovanje (UML)**
- Kontinualna provera kvaliteta
- Upravljanje promenama



UML dijagrami

- Široko **usvojeni standard** tako da je razumevanje modela i komunikacija dosta olakšana

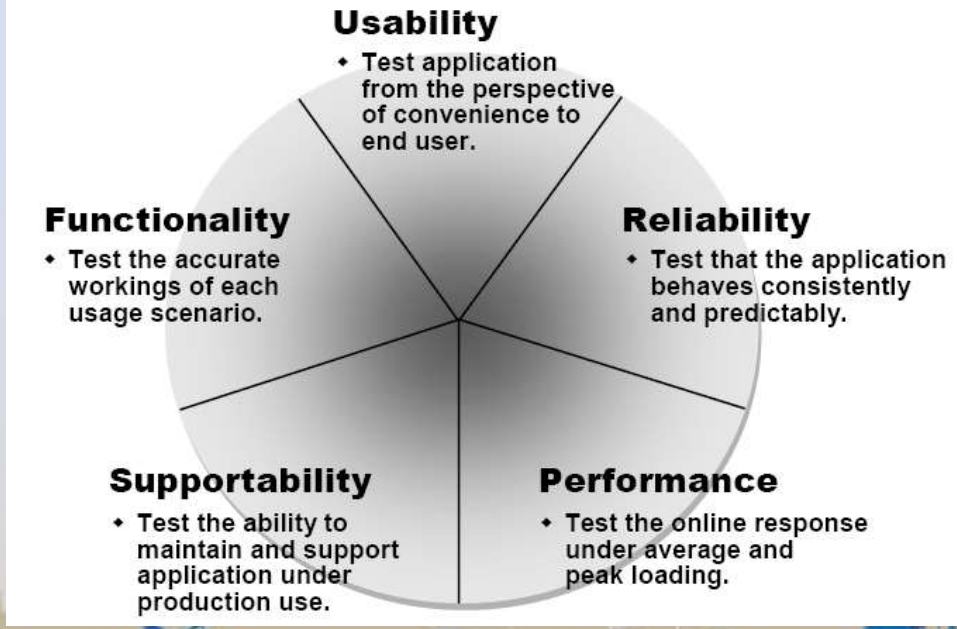


Praksa 5: Kontinualna provera kvaliteta

Najbolje prakse

- Iterativan razvoj
- Upravljanje zahtevima
- Upotreba komponentne arhitekture
- Vizuelno modelovanje (UML)
- **Kontinualna provera kvaliteta**
- Upravljanje promenama

- Testiranje softvera iznosi od 30-50% troškova razvoja softvera
- Testiranje se uglavnom radi bez jasne metodologije i bez podržanih alata

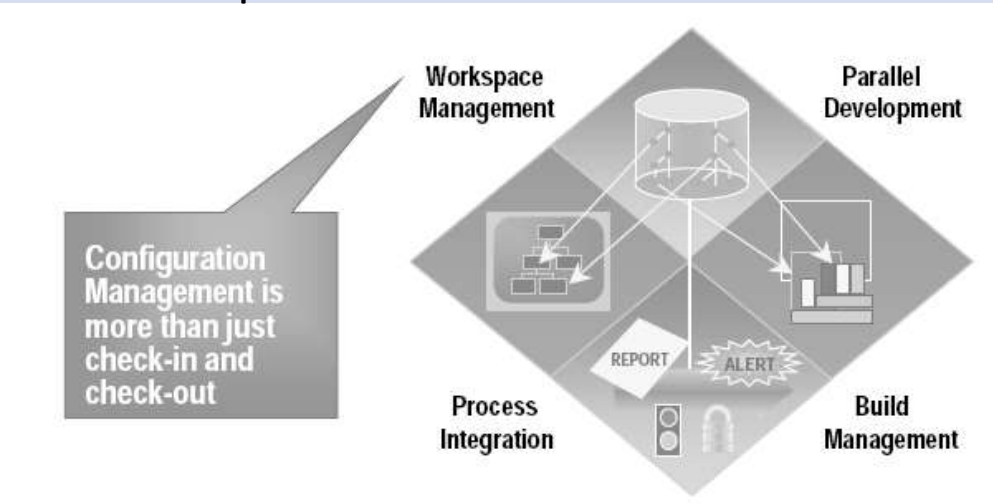


Praksa 6: Upravljanje promenama

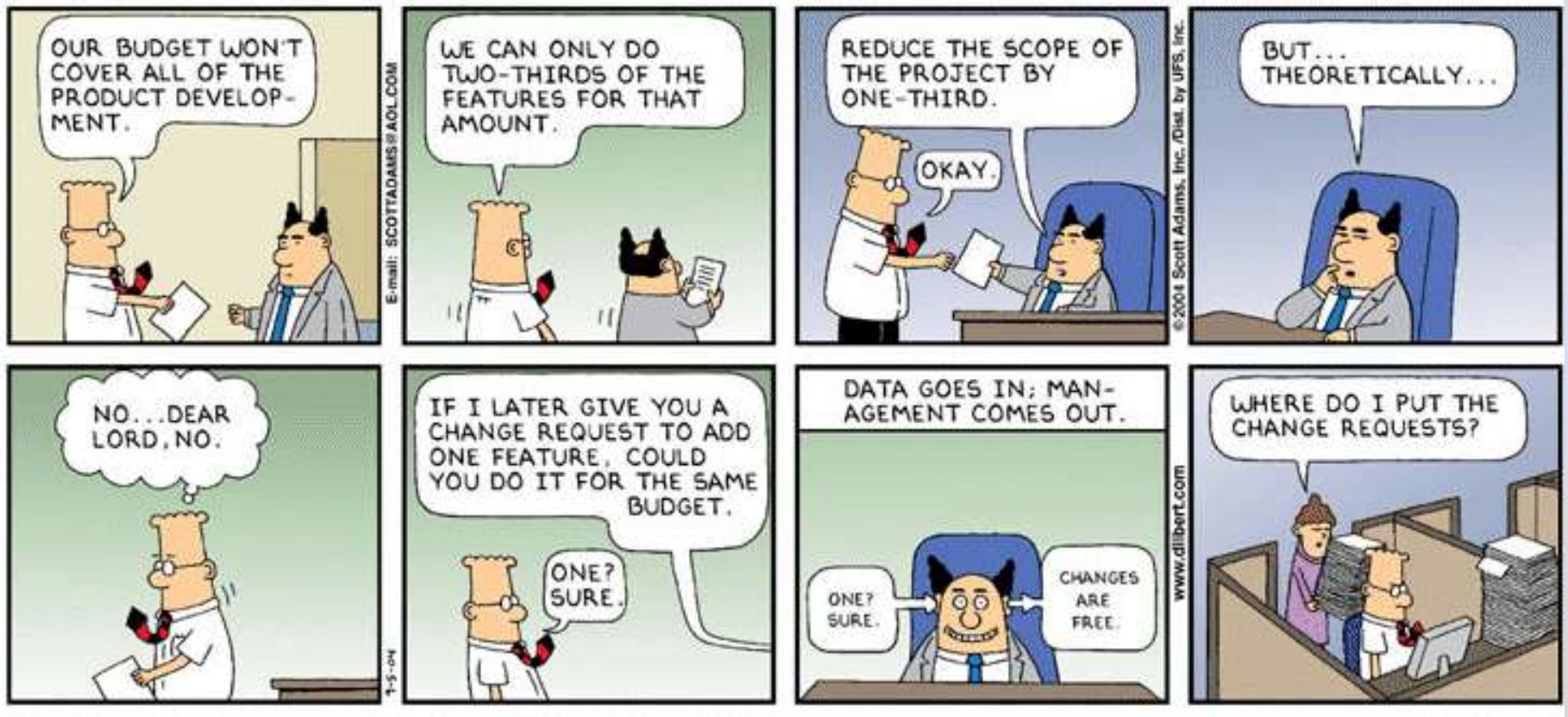
Najbolje prakse

- Iterativan razvoj
- Upravljanje zahtevima
- Upotreba komponentne arhitekture
- Vizuelno modelovanje (UML)
- Kontinualna provera kvaliteta
- **Upravljanje promenama**

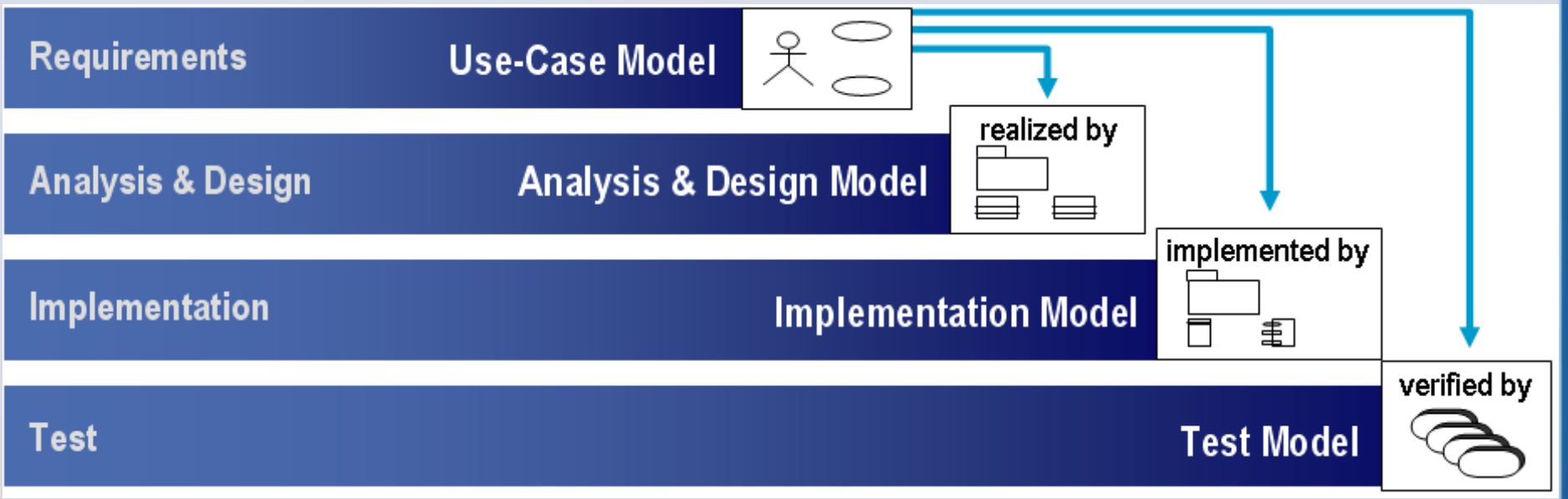
- *Unified Change Management* (UCM) je *Rational Software* pristup za upravljanje promenama u razvoju softvera od zahteva do isporuke



Upravljanje promenama korisničkih zahteva u praksi :)



Razvoj vođen slučajevima korišćenja



Scrum

- Kratka istorija
- Šta je Scrum?
- Scrum okvir
- Scrum role
- Scrum model procesa
- Scrum dokumenta

“We’ve focused enough on process . . .

how about we just write some software”

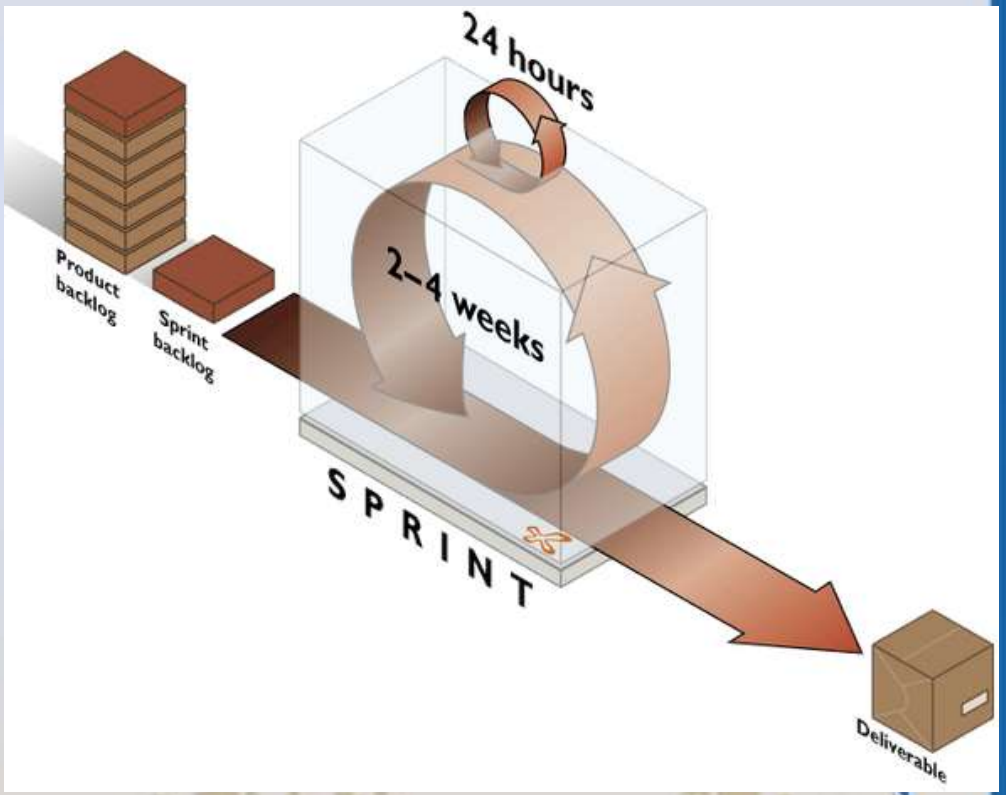
David Anderson, SEPG 2008

Scrum
Kratka istorija

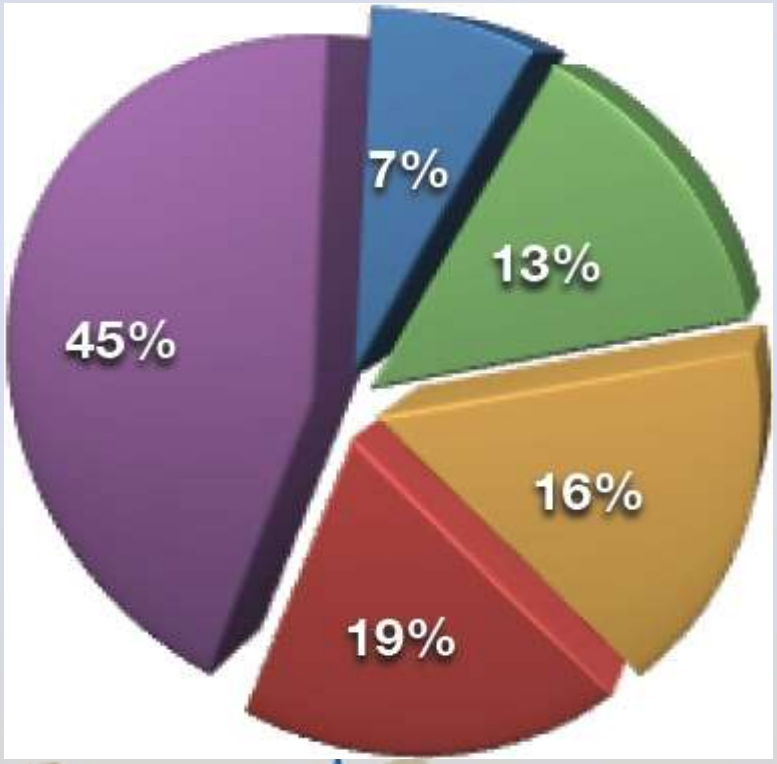
- **Scrum nije akronim**, već termin koji se koristi u ragbiju i označava ponovno pokretanje igre nakon manjeg prekršaja
- 1995. god. - prva javna prezentacija od strane **Ken Schwaber** i **Jeff Sutherland** (slika)
- 2001. godine – Schwaber zajedno sa Mike Beedle opisuje Scrum u knjizi ***Agile Software Development with Scrum***

Šta je Scrum?

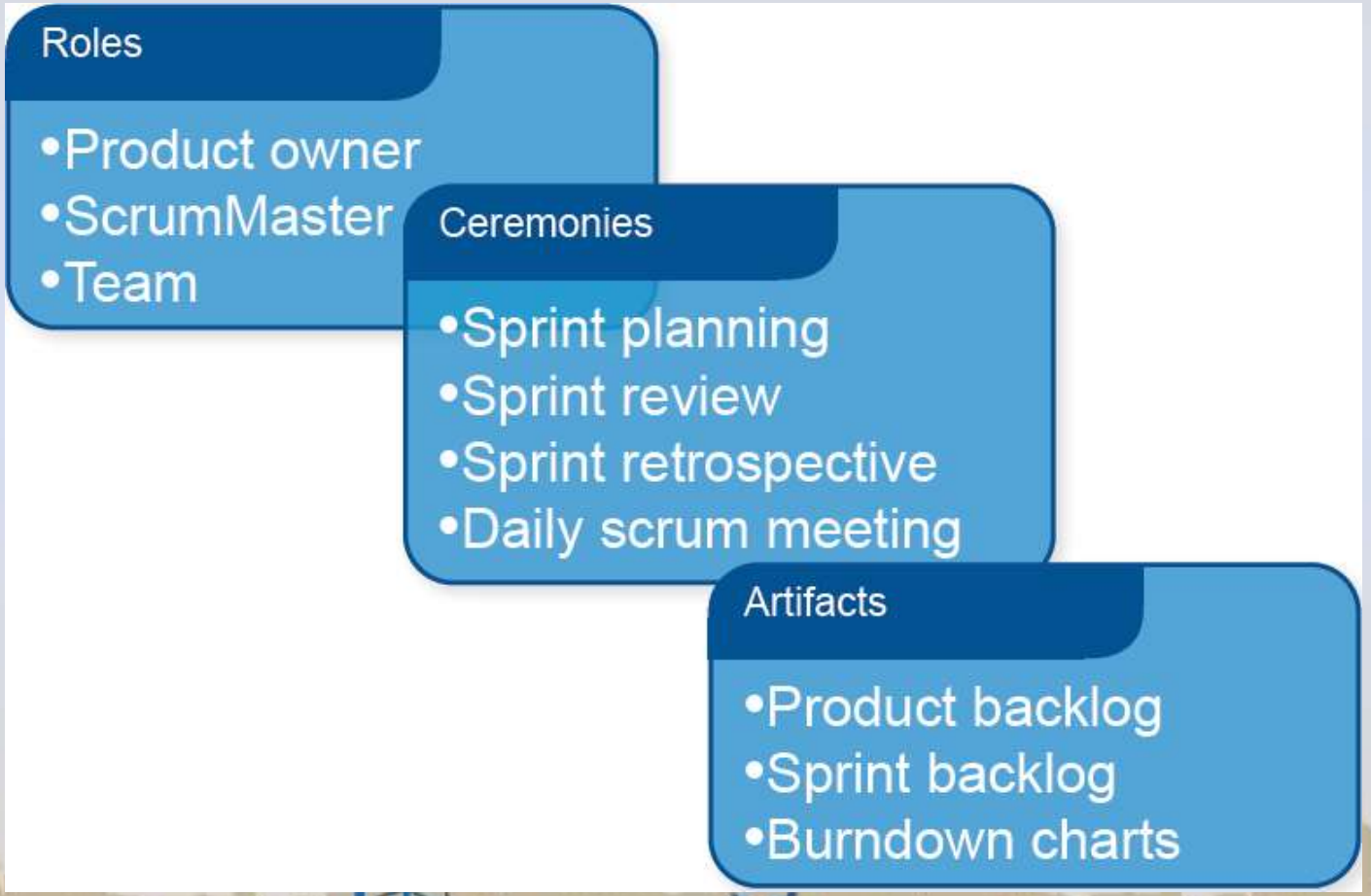
- Okvir (*framework)* za brzi agilni proces **razvoja i održavanja** složenih softverskih proizvoda
- Okvir koji ima **iterativno-inkrementalan pristup** razvoju proizvoda kako bi se kontrolisali rizici i optimizovala predvidljivost
- Okvir u kome ljudi ukazuju na složene probleme, dok produktivno i kreativno **isporučuju visoko kvalitetne proizvode**
- Okvir u kom se mogu **primeniti razni procesi i tehnike**
- Okvir koji se sastoji od Scrum timova i njima pridruženih **uloga**, **događaja**, **artifakta** i **pravila**



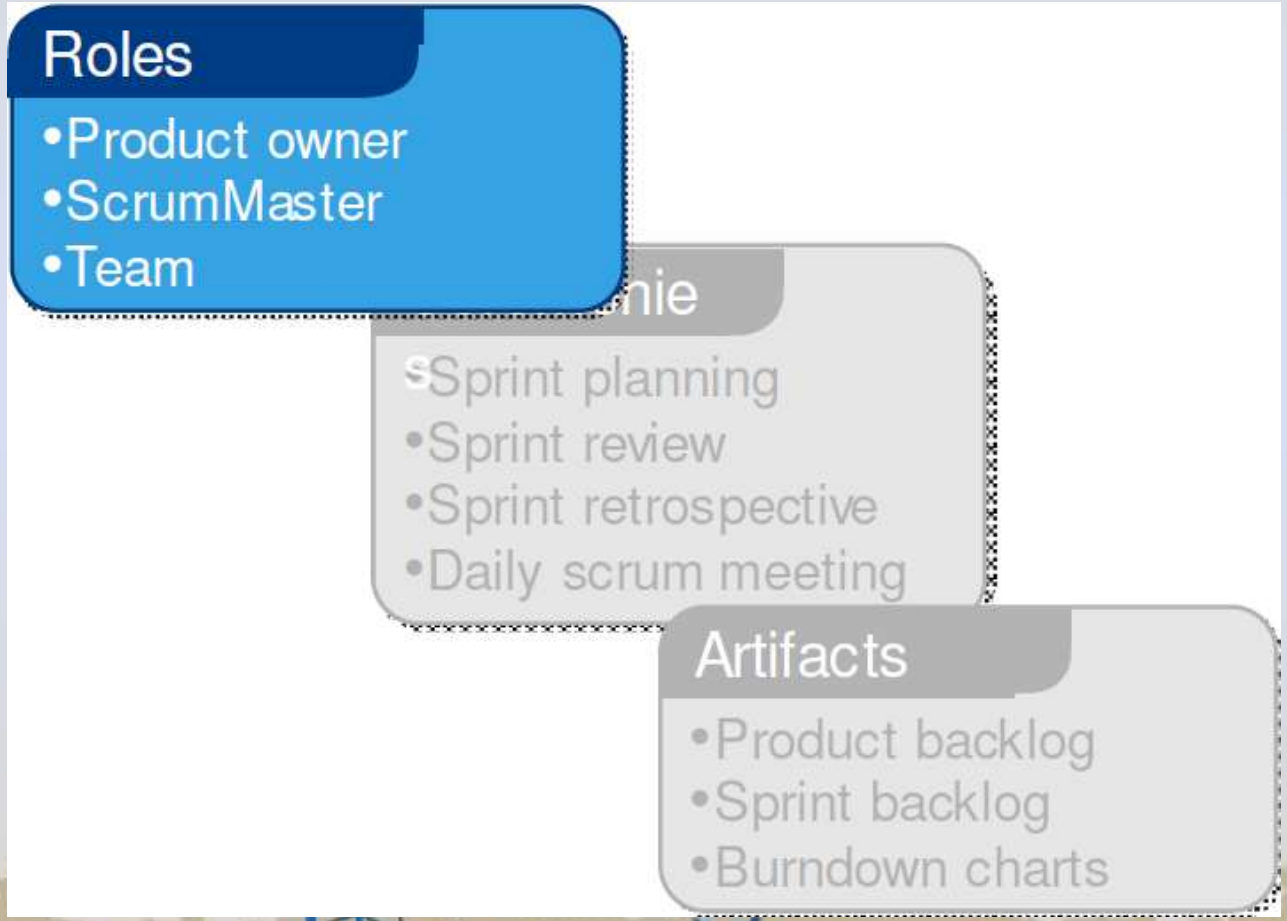
Procenat korišćenja razvijenih zahteva u sistemima



Scrum framework



Scrum framework - Role

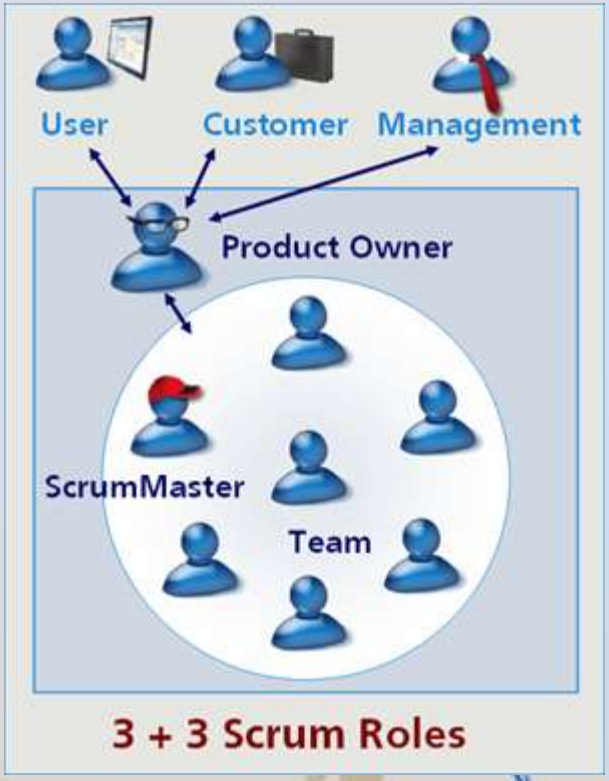


Osnovne Scrum role

- **Vlasnik proizvoda (*Product Owner*)**
  - Zastupa klijenta
  - Definiše i postavlja prioritetne funkcionalnosti
  - Odlučuje o datumu i sadržaju verzije
  - Odgovoran za profitabilnost (ROI)
  - Prihvata ili odbacuje rezultate sprinta …
- **Scrum Master**
  - Menadžer projekta
  - Osigurava da je tim potpuno funkcionalan i produktivan
  - Osigurava blisku saradnju između svih rola
  - Štiti tim od spoljnjih ometanja
  - Rešava sve probleme tokom sprinta
- **Razvojni tim (*Development Team)***
  - Od 5 do 9 osoba - unakrsan tim: programeri, testeri, dizajneri …
  - Puno radno vreme
  - Samostalno se organizuje - prijateljstvo, osećanje zajedništva, bez titula …
  - Mogu se menjati između sprintova

Formiranje tima

- Na početku projekta formalni menadžment bira vlasnika proizvoda i Scrum mastera
- Vlasnik i Scrum master biraju ostale članove tima
- Tim treba da čine **kreativne**, fleksibilne i produktivne osobe
- Tim treba da ima zahtevane kompetencije i da budu **profesionalci** u svom poslu, a ne da zavise od drugih
- Tim se **samostalno** organizuje i odlučuje o načinu obavljanja posla (čak ni Scrum Master ne govori timu kako treba da radi)
- **Ne postoje podtimovi**
- Tim svakodnevno radi sa klijentima
- Tim isporučuje proizvod iterativno i inkrementalno
- Maksimalno prati povratne reakcije i mogućnosti



TYPES OF AGILE TEAMS TO AVOID

"THE SINKING SHIP TEAM"

"THE MERRY-GO-ROUND TEAM"

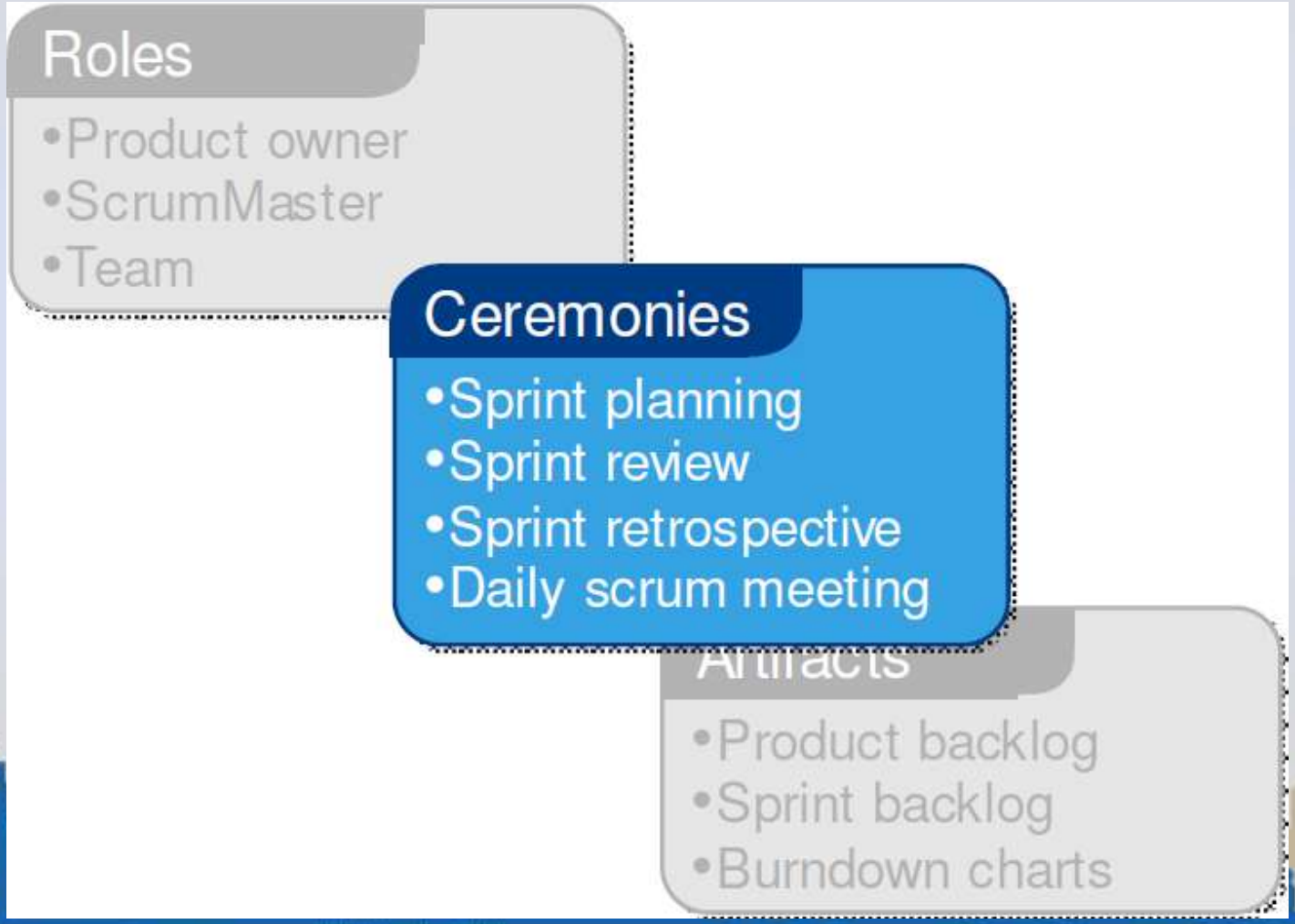
THE ROLLER COASTER TEAM

"THE WELL-OILED MACHINE TEAM"

SMARTBEAR

ORIGINALLY PUBLISHED ON THE SMARTBEAR BLOG AT BLOG.SMARTBEAR.COM

Scrum framework - Događaji



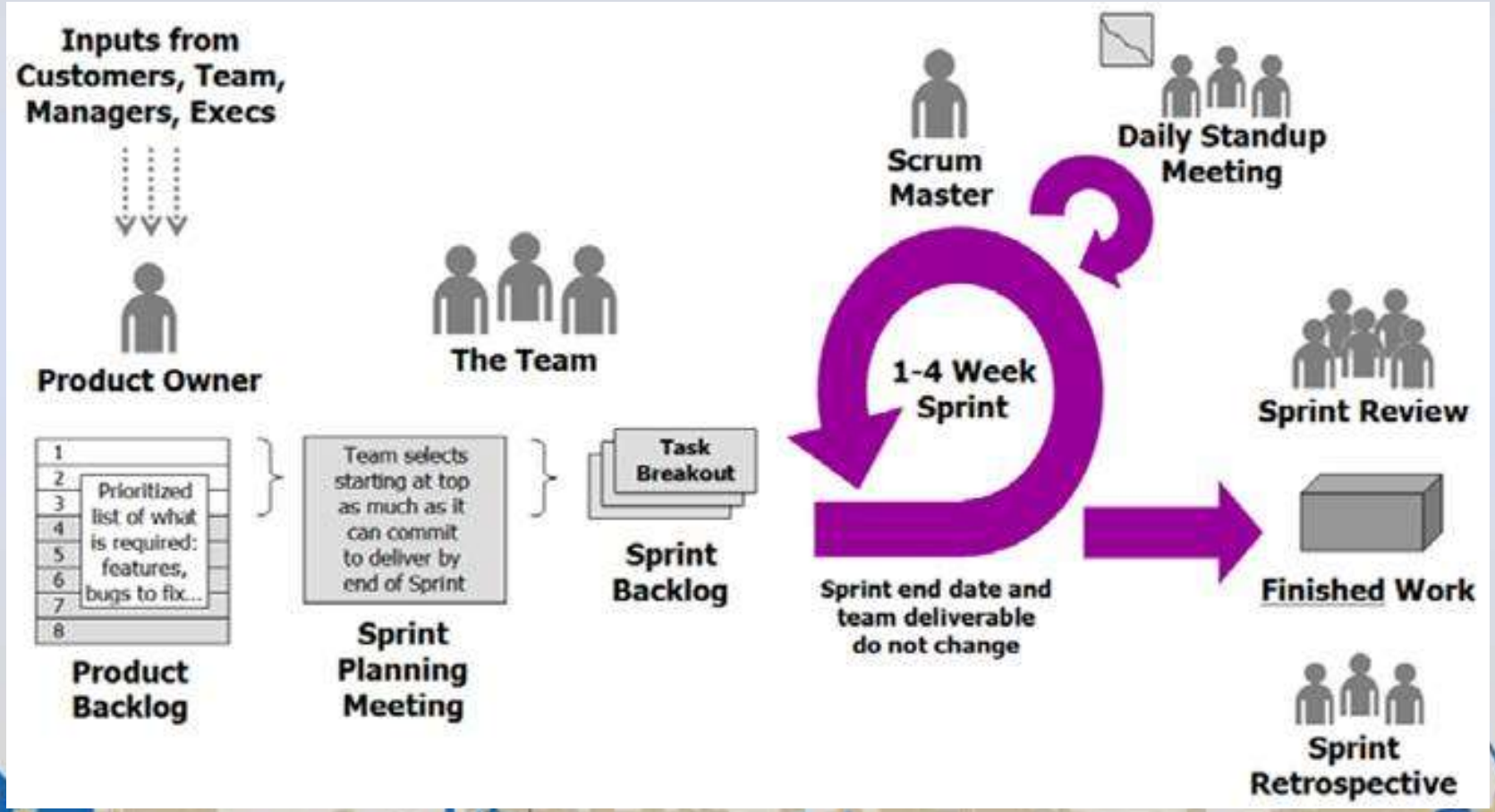
Stubovi Scrum-a

- Scrum se oslanja se na tri stuba:
  - **Transparentnost**
  - **Inspekcija/Kontrola**
  - **Adaptacija/Prilagođavanje**
- Kontrola i prilagođavanje su opisani Scrum događajima (*Scrum events*):
  - Sastanak Planiranja Sprinta (*Sprint Planning Meeting*)
  - Dnevni Scrum (*Daily Scrum*)
  - Pregled sprinta (*Sprint Review*)
  - Retrospektiva sprinta (*Sprint Retrospective*)

Scrum model procesa – u kratkim crtama

1. Na početku projekta: **spisak funkcionalnosti na celom projektu** se definiše u **Product backlo**g-u
2. Iz tog spiska tim **bira funkcionalnosti** koje može da realizuje u toku **jednog sprinta**
3. Tokom jednog sprinta nastaje **kompletna softverska celina** koja može da se pokaže vlasniku proizvoda
4. Sprint se **ne prekida**
5. Sprint je **uspeo ako su spaljeni svi poeni** koje je tim predvideo za konkretni sprint, a za predviđeno vreme
6. Sprint **može prekinuti** samo **Scrum Master**, ako je tim u kašnjenju, te zakazuje sastanak na kome treba da sagleda razloge neuspeha i izvede zaključke
7. **Svakog dana** tim održava **kratke sastanke** od 15-tak min gde se informišu šta je neko radio prethodnog dana i šta namerava da radi i da li ima nekih problema
8. Posle svakog sprinta klijentu se na ***Sprint Review*** sastanku prikazuje ono što je **realizovano** – nema prezentacije već se prikazuje rad aplikacije
9. Ako **klijent prihvati** rezultate, onda se funkcionalnosti mogu **postaviti u produkciono okruženje**

Scrum model procesa



Sastanak planiranja sprinta
Sprint Planning Meeting

- Tim bira zadatke za koje se obavezuje da će završiti
- Odgovara se na pitanja:
  - Šta će biti urađeno u sprintu?
  - Kako će se izabrani posao uraditi?
- Kreira se *Sprint backlog* gde se zajednički identifikuju zadaci i procenjuju vremena (1-16 sati)
- Dizajn je visokog nivoa

Teme koje nas vode tokom projekta

- **Cilj**:
  - Isporučiti poslovnu vrednost što je pre moguće
  - Isporučiti najkvalitetnije moguće
- **Vizija** nam pomaže u svakodnevnim odlukama:
  - Da li nam je stvarno potrebna ta funkcionalnost?
  - Da li je TextBox dovoljno dobar ili nam je potrebno automatsko popunjavanje na osnovu unosa pretrage?
  - Vizija omogućava da se dostigne stabilnost u projektu
- **Alat** developera
- **Product backlog** (sadrži korisničke zahteve (*user stories*) koji su potrebni za narednu verziju)
  - **Kako doći do korisničkih zahteva?** – Projekat jedino postoji ukoliko postoji poslovna potreba. Ova potreba definiše skup funkcionalnosti. Funkcionalnost je opis funkcija koje korisnik može da izvršava.

Dnevni Scrum
Daily Scrum

- Karakteristike: dnevni, 15 min., stojeći
- Pomaže da se izbegnu drugi nepotrebni sastanci
- Članovi tima odgovaraju na pitanja:
  - **Šta sam uradio juče?**
  - **Da li je nešto ostalo nezavršeno?**
  - **Šta radim danas?**
  - **Da li me nešto usporava i da li mi nešto stoji na putu?**
- Ne rešavaju se problemi
- Nije merilo kontrole, već poveravanje i obavezivanje drugim članovima tima

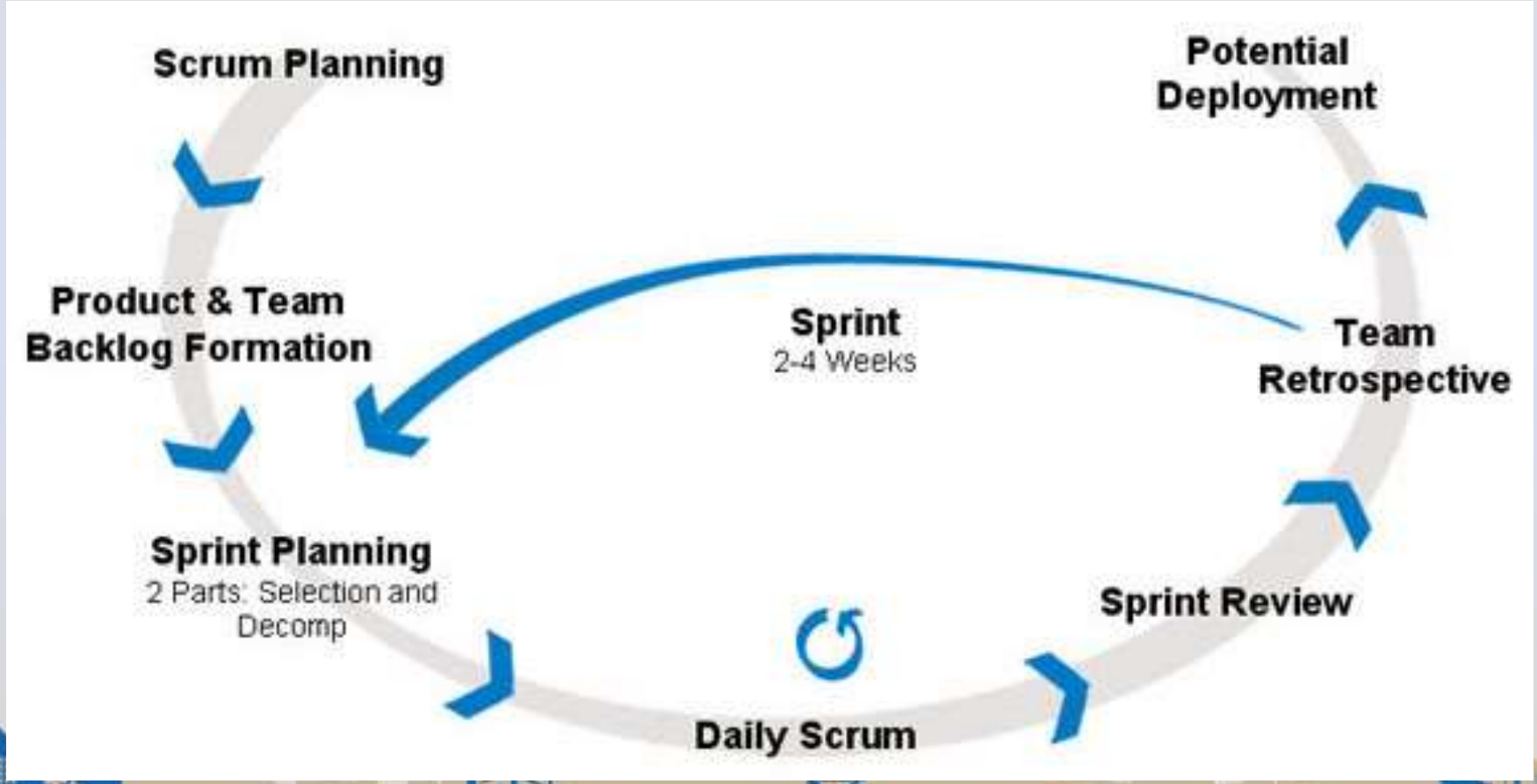
Pregled sprinta
Sprint Review

- Tim prezentuje šta je urađeno tokom sprinta
- Obično je u formi demo novih funkcionalnosti
- Neformalan
  - Nema slajdova
  - 2 sata pripreme
- Učestvuju svi članovi tima
- Svi su pozvani

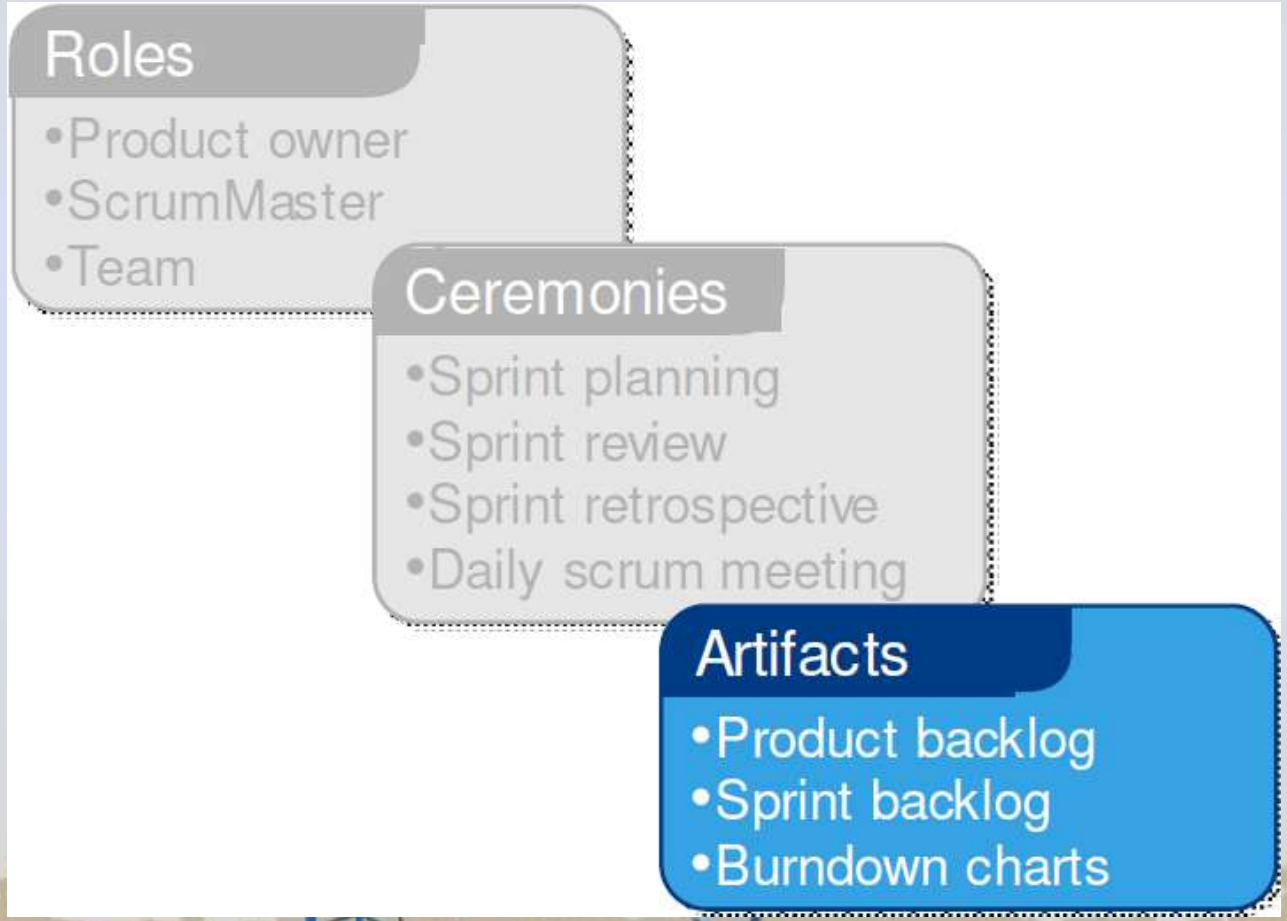
Retrospektiva sprinta
Sprint retrospective

- Periodično treba videti šta radi, a šta ne radi
- Identifikuje se oblast poboljšanja – osmisliti način poboljšanja
- Traje 15-30 min
- Nakon svakog sprinta
- Svi timovi učestvuju, a mogu prisustvovati i klijenti
- Ceo tim odlučuje o tome da li:
  - Započeti
  - Prekinuti
  - Nastaviti

Redosled Scrum događaja



Scrum framework - Dokumenta



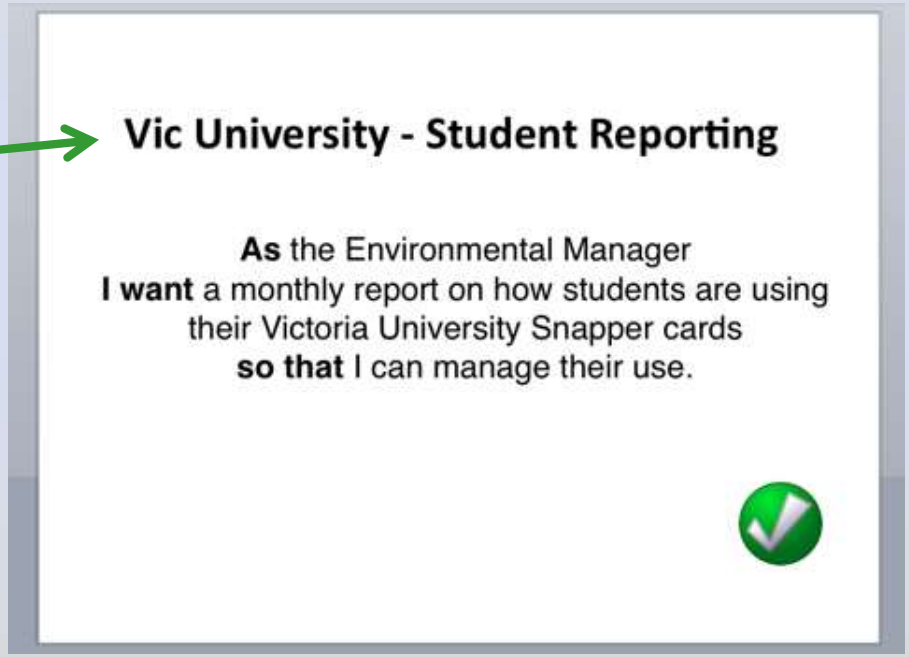
Product backlog

Product Backlog

- Zahtevi koji imaju konkretnu vrednost za klijente
- Lista svih željenih poslova na projektu napisanih u vidu korisničkih priča (*user stories*)
- Product owner postavlja prioritete
- Pre početka svakog sprinta prioriteti se ponovo razmatraju
- Mogu se dodavati nove funkcionalnosti ili ukloniti postojeće

Backlog item	Estimate
Allow a guest to make a reservation	3
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
...	30
...	50

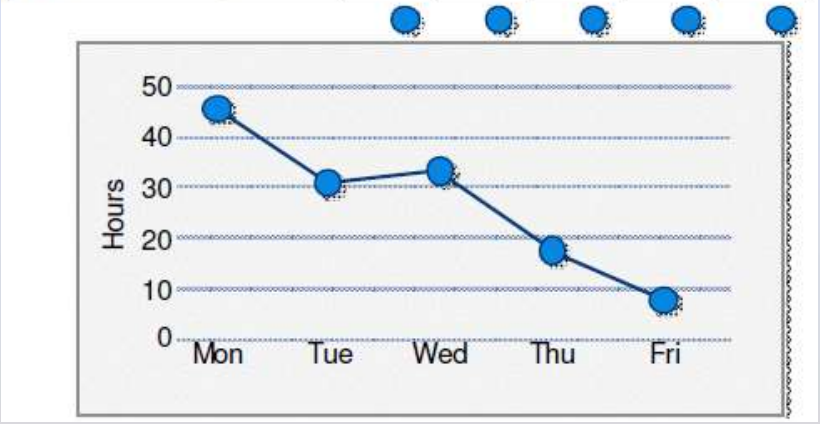
Primeri korisničkih priča (*User stories*)



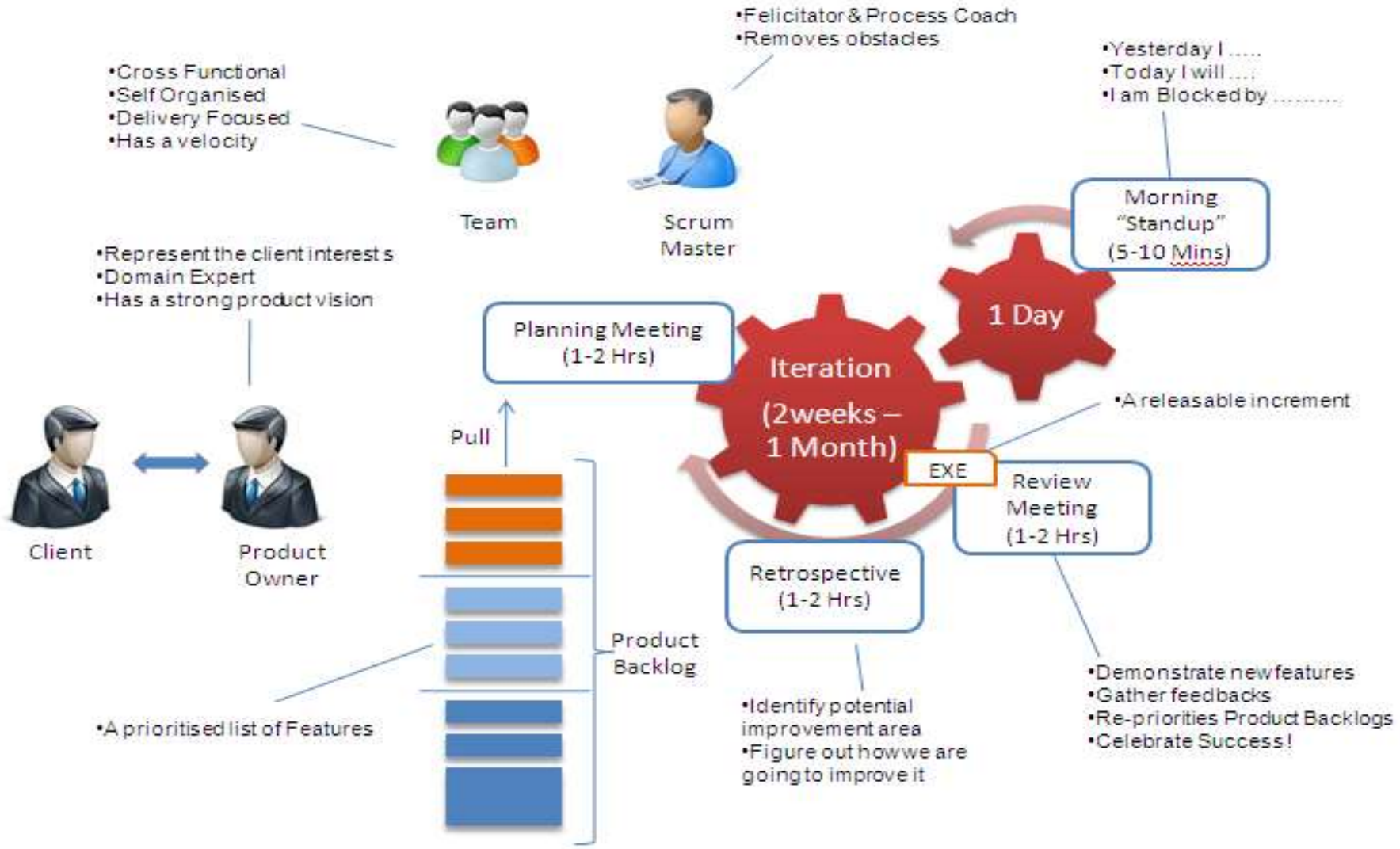
Sprint backlog

- Svaki sprint se posebno planira
- Identifikovati **cilj sprinta** - kratak iskaz o tome šta je fokus u narednom sprintu
- **Tim bira zahteve iz product backloga** i **kreira sprint backlog**
- **Jedna stavka iz product backloga** se obično opisuje sa **više stavki u sprint backlogu**
- Tim licitacijom daje **težinske poene** (*Story points*) zadacima (**težina izvođenja zadataka**)
- Tokom sprinta, realizacijom zahteva iz Sprint backloga, smanjuje se broj poena koji treba da se realizuje, tzv. **spaljivanje poena** (*burndown*)
- Za praćenje napredovanja sprinta može da se kreira **sprint burndown grafikon** u koji se po danima unosi koliko je poena ostalo timu za spaljivanje na kraju svakog radnog dana

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				



Mikro inkrement u sprintu?



Scrum Methodology

Prednosti Scrum-a

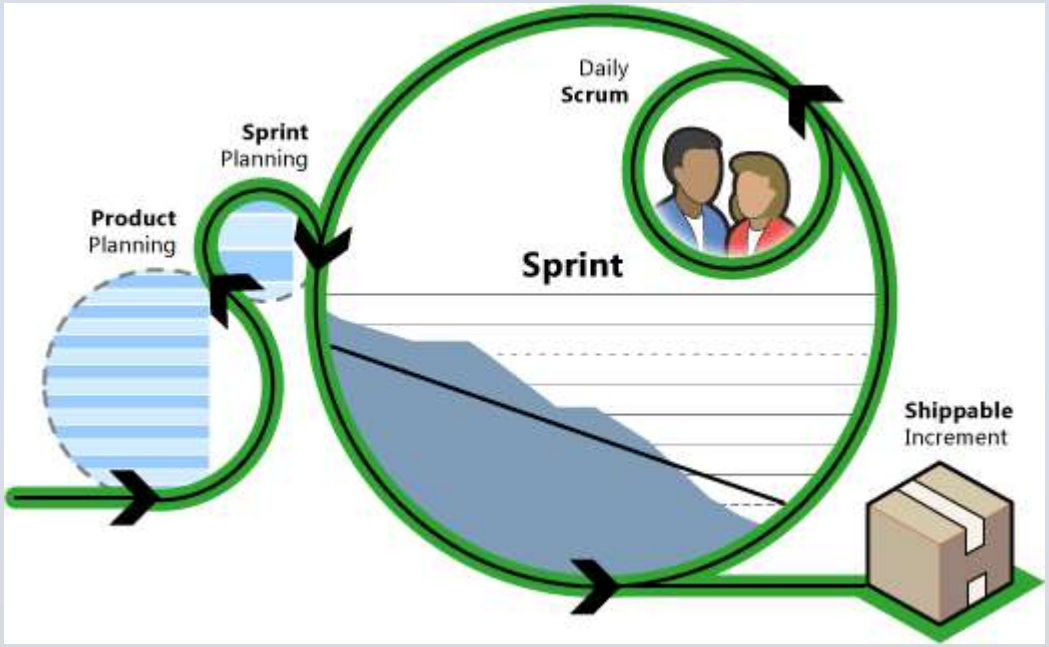
- Brži povraćaj uloženog (ROI)
- Samoorganizujući, visoko stručni timovi
- Brže implementiranje izmena zahteva klijenta
- Isporučen kvalitetan proizvod
- Definisana stabilna arhitektura
- Lagana dokumentacija

Modifikovane tradicionalne metodologije

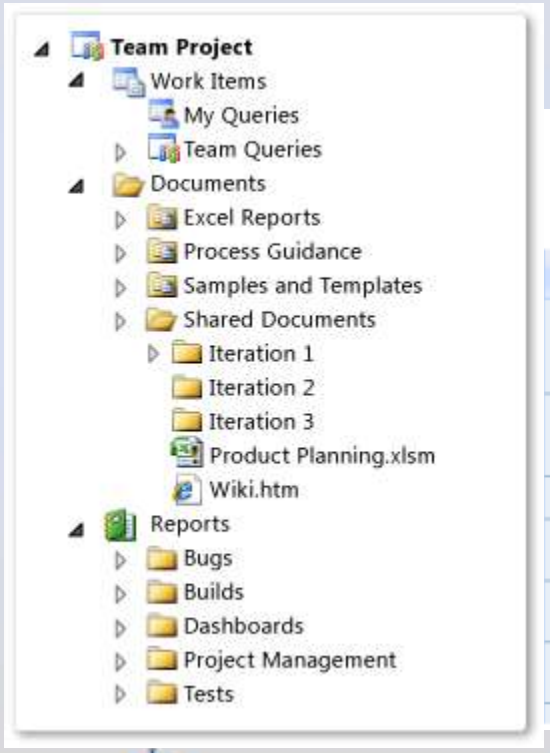
- MSF for Agile
- Open Unified Process

MSF za agilni razvoj softvera v5.0
MSF for Agile Software Development v5.0

- Baziran na Scrumu
- Integrisan u Visual Studio 2010 pod nazivom **Visual Studio Application Lifecycle Management** (ALM)
- Daje smernice za primenu Scruma i agilnih inženjerskih praksi
- Kako? - > MSDN: http://msdn.microsoft.com/en-us/library/dd380647.aspx



Izgled ekrana i primer prodact backloga



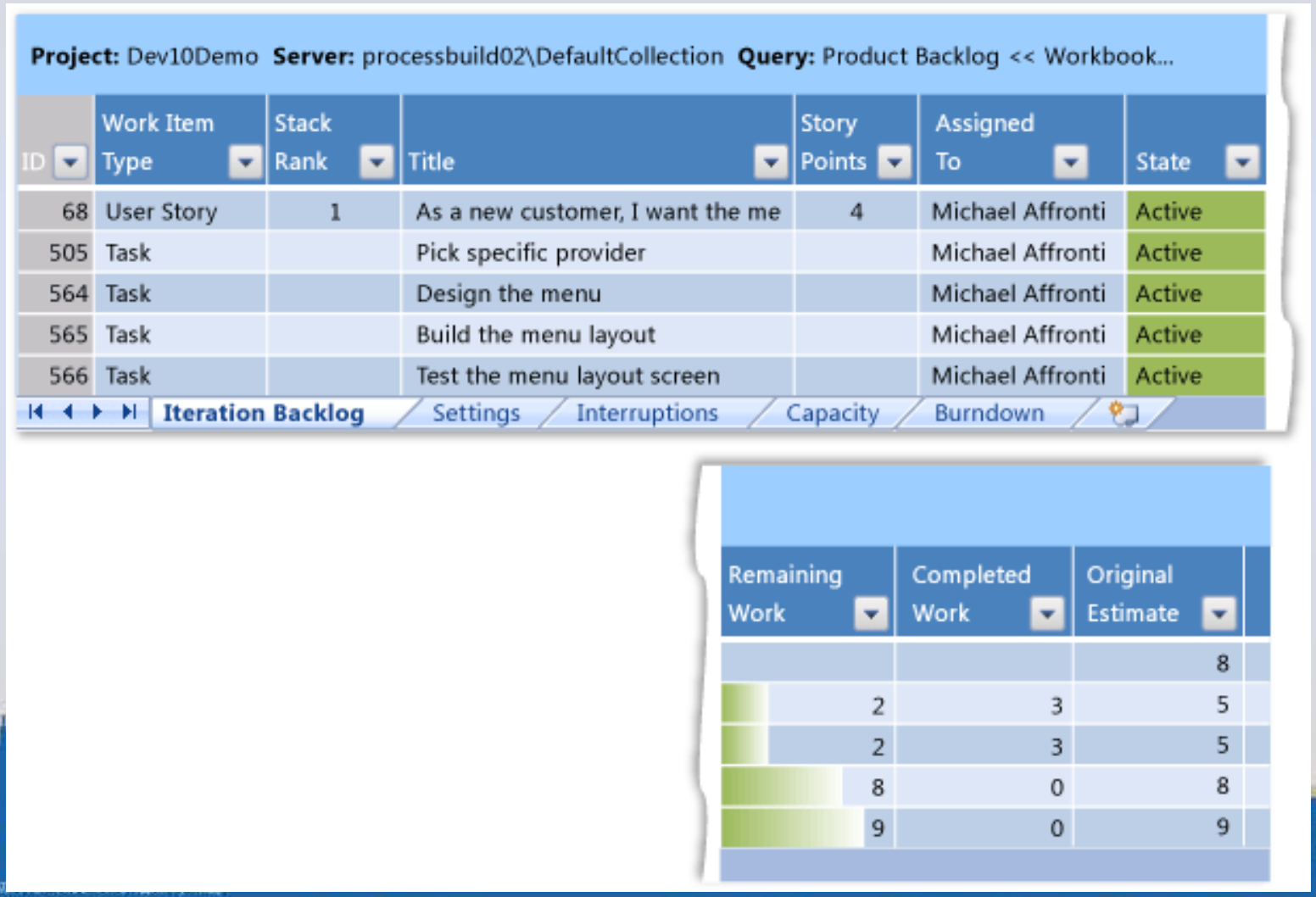
1 Create items

2 Rank the items

3 Estimate the items

	A	B	C	D	E	F
1	**Project:** Dev10Demo **Server:** processbuild02\DefaultCollection **Query:** Product Backlog **List type:** Flat					
2	ID	Title	Rank	Points	State	Iteration Path
3	58	As a new customer, I want to order a meal.	1	4	Resolved	\Iteration 0
4	68	As a new customer, I want the menu to be limited to those available for delivery to my location.	2	4	Active	\Iteration 0
5	59	As a new customer, I want to get an idea of what DinnerNow offers with a brief glance at the web site.	3	5	Active	\Iteration 0
6	60	As a customer who has completed an order, I want Dinner-Now to keep track of my meal preferences.	4	9	Active	\Iteration 0

Primer Sprint backloga



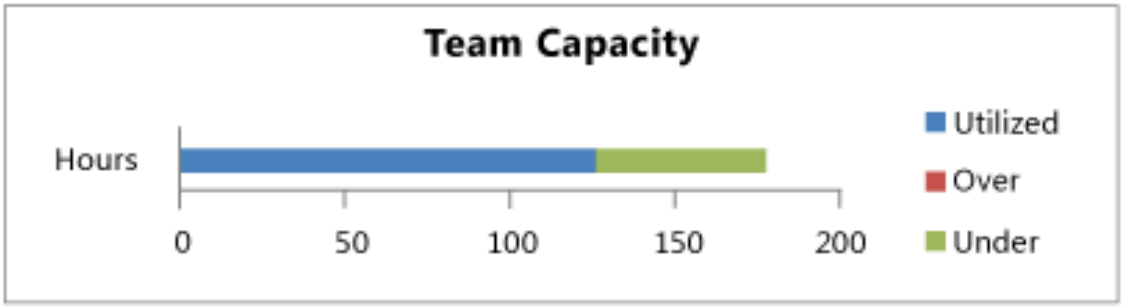
Project: Dev10Demo **Server:** processbuild02\DefaultCollection **Query:** Product Backlog << Workbook...

ID	Work Item Type	Stack Rank	Title	Story Points	Assigned To	State	Remaining Work	Completed Work	Original Estimate
68	User Story	1	As a new customer, I want the me	4	Michael Affronti	Active			8
505	Task		Pick specific provider		Michael Affronti	Active	2	3	5
564	Task		Design the menu		Michael Affronti	Active	2	3	5
565	Task		Build the menu layout		Michael Affronti	Active	8	0	8
566	Task		Test the menu layout screen		Michael Affronti	Active	9	0	9

Iteration Backlog | Settings | Interruptions | Capacity | Burndown

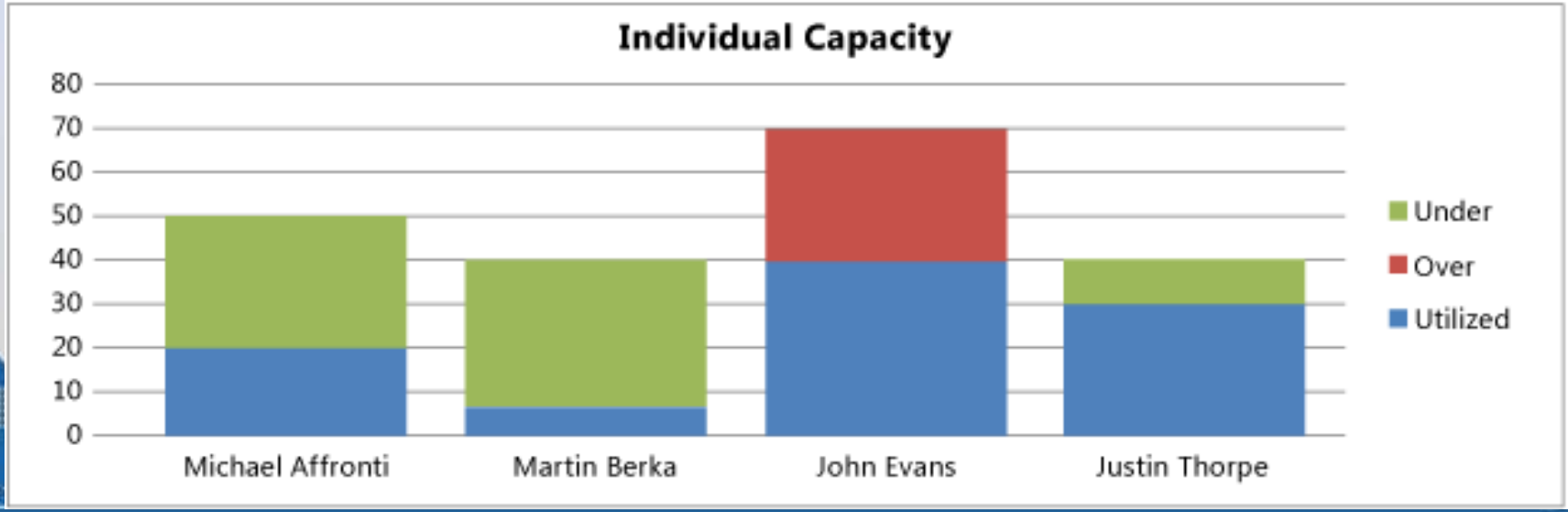
Team Capacity

Totals	Hours
Remaining Work	129
Remaining Capacity	177
Utilized	129
Over	0
Under	48



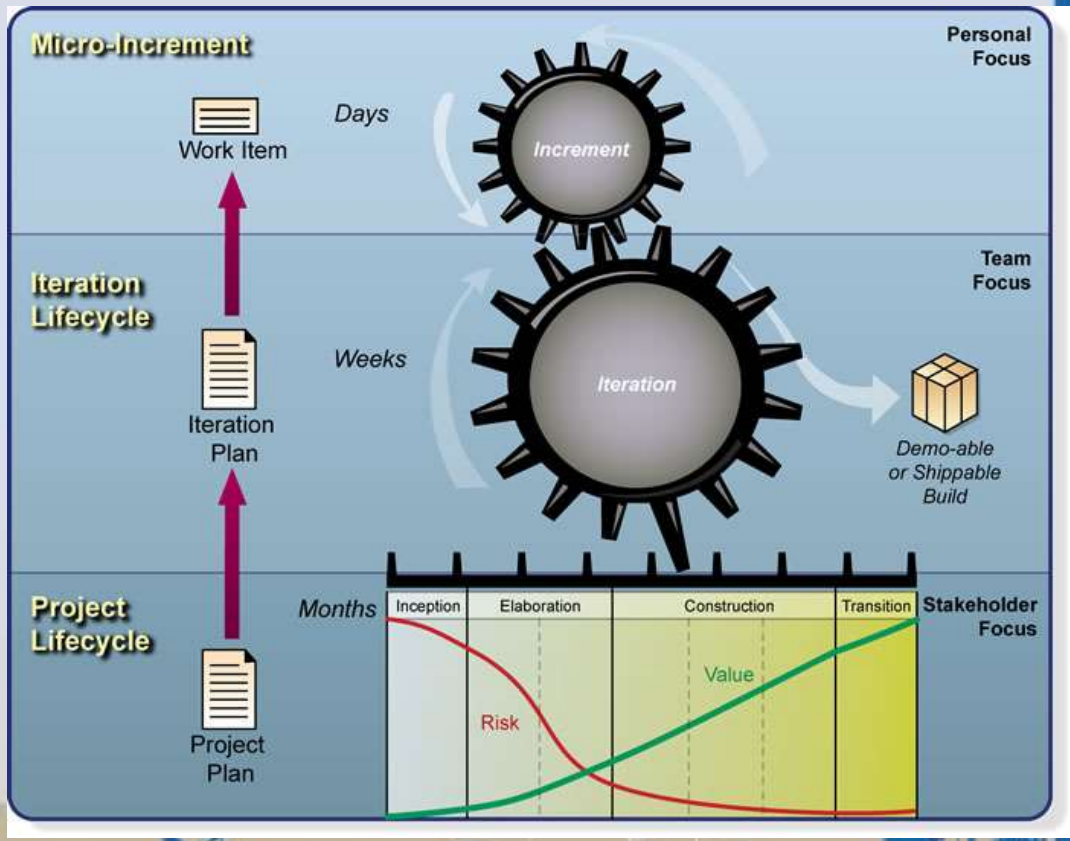
Individual Capacity

Team Member	Hours/Day	Days	Capacity	Assigned	Utilized	Over	Under
Michael Affronti	3	17	51	21	21	0	30
Martin Berka	2	21	42	6	6	0	36
John Evans	3	14	42	72	42	30	0
Justin Thorp	2	21	42	30	30	0	12



Open Unified Process
Otvoreni ujedinjeni proces

- Deo *Eclipse Process Framework*-a (*open source process framework)* razvijenog od strane *Eclipse Foundation* (IBM)
- Cilj: usvajanje jezgra RUPa i agilne filozofije
- Zasniva se na tri ključna koncepta:
  - Mikro-inkrement
  - Životni ciklus iteracije (više mikro inkrementa)
  - Životni ciklus projekta (Inception, Elaboration, Construction, Transition)



Sa čime se današnji developer timovi suočavaju?

Project plan templates

Article on serialized java beans

Website with Configuration mgmt guidelines

Book on J2EE

Lessons learnt from previous project and iteration

JUnit user guide

Wiki on agile development

Latest research on effectiveness of pair programming

Knowledge base on managing iterative development

Corporate guidelines on compliance

- Ne postoji zajednički jezik ili terminologija između procesa – redundantnost i nekonzistentnost
- Znanje se ne može lako prilagoditi novim najboljim praksama ili različitim projektima
- Ne postoji centralna zajednica ili okvir za komunikaciju koji će olakšati približavanje najboljih praksi kroz različite domene

Open Source

"Open Source is not about Free. It's about Freedom. The freedom to collaborate. The freedom to innovate."

"Open source gives more people access to the building blocks of innovation, enabling diverse perspectives and influences to be integrated into the creative process.

Nick Donofrio
Sr. VP, Technology&Manufacturing, IBM
At LinuxWorld August, 2004

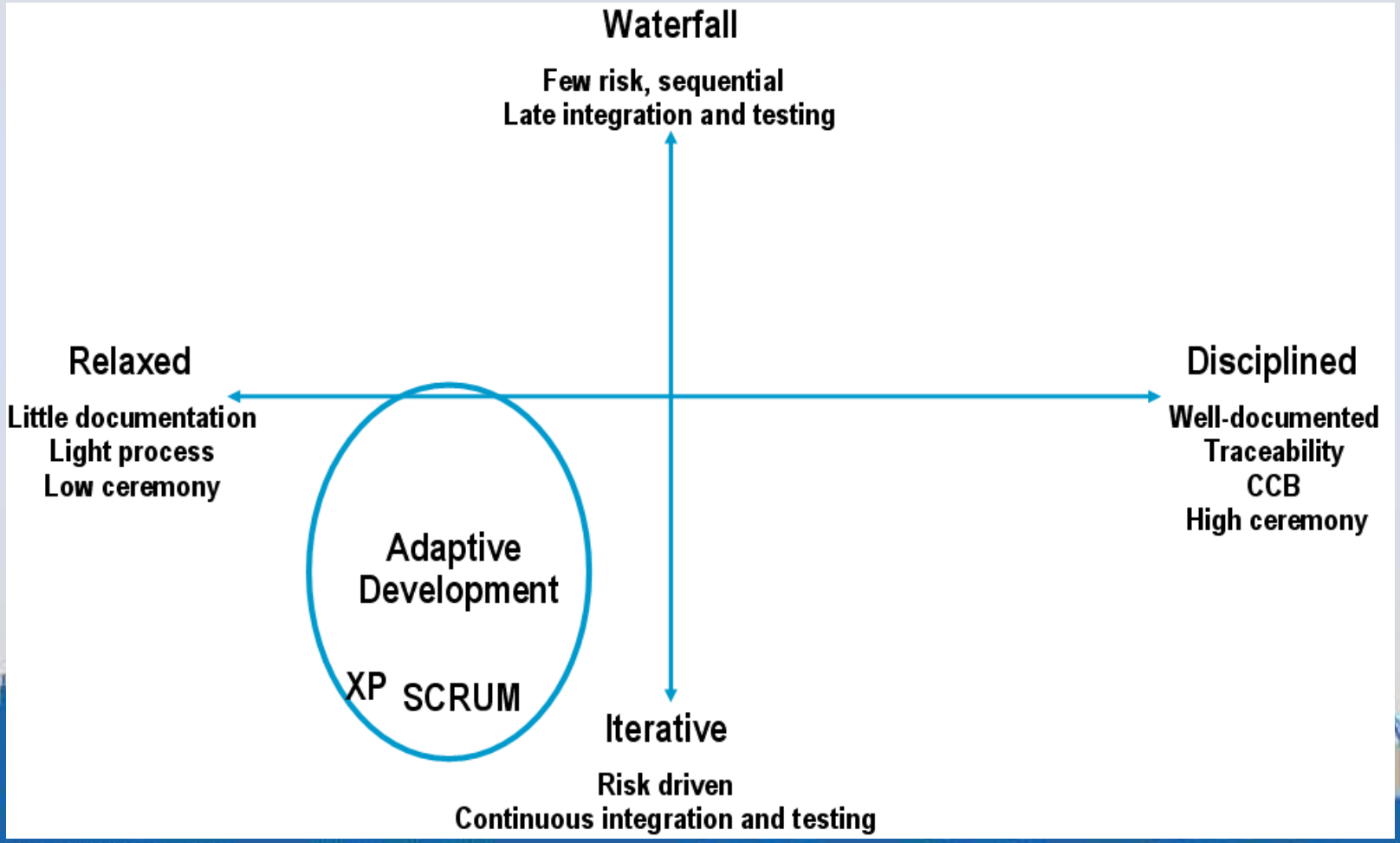
Odakle početi?

- OpenUP: http://epf.eclipse.org/wikis/openup/

Za više informacija

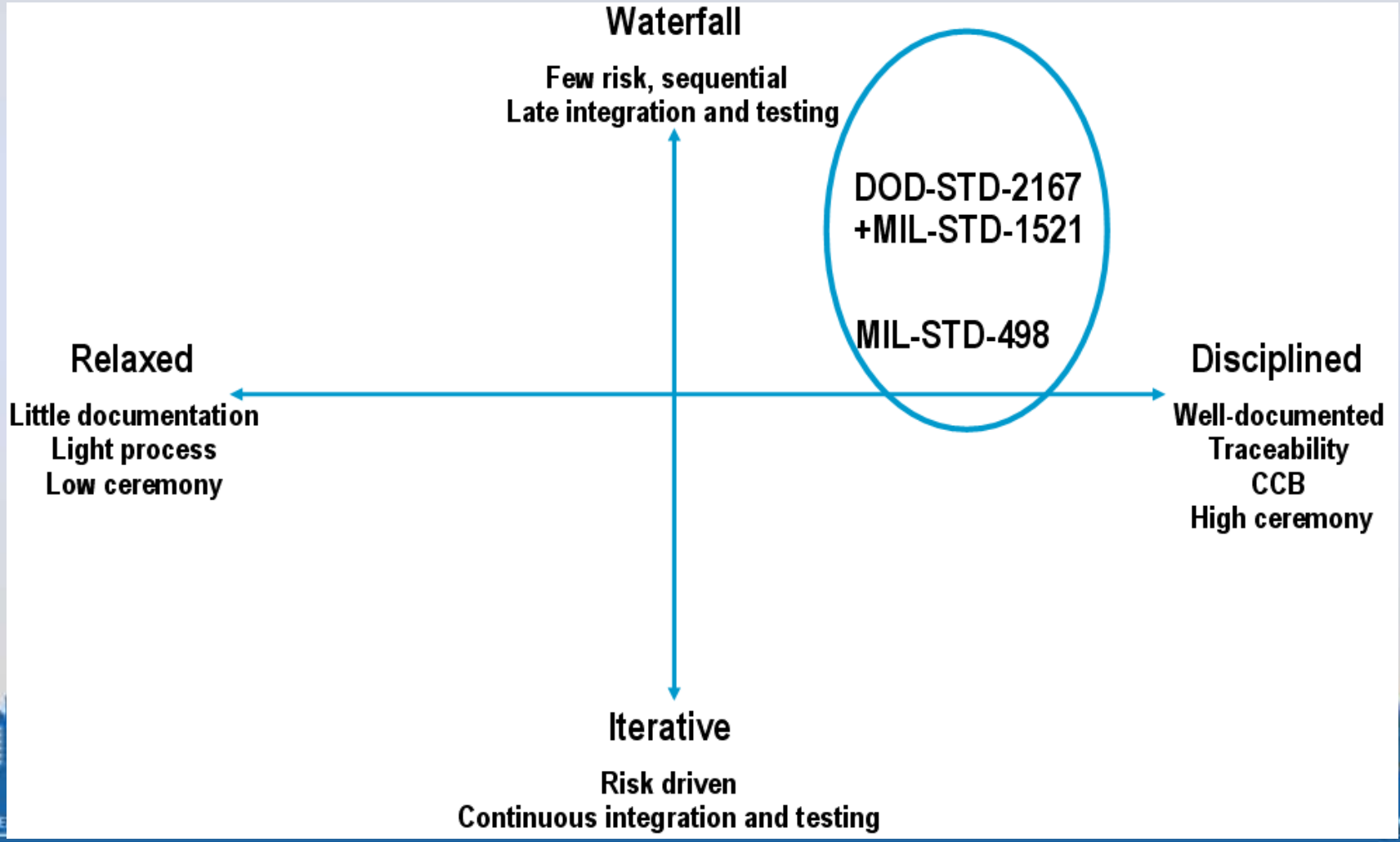
- Infomracije: Eclipse Process Framework (EPF)
  - http://www.eclipse.org/epf/ (See right column for Getting Started, Newsgroup, Developers Mail List, etc)
- Članci: www.therationaledge.com
  - DeveloperWorks: The Eclipse Process Framework Project, Kroll, http://www.ibm.com/developerworks/rational/library/05/1011_kroll/
  - Eclipse Review: A Development Library at your Fingertips, Kroll and Sand, http://www.eclipsereview.com/retrieve/er_200609.htm
  - Rational Edge: Eclipse Process Framework Composer - Part 1: Key Concepts, Haumer, http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf
  - Rational Edge: Eclipse Process Framework Composer - Part 2: Authoring Method Content and Processes, Haumer, http://www.eclipse.org/epf/general/EPFComposerOverviewPart2.pdf
- Knjiga
  - Per Kroll and Bruce MacIsaac, Agility and Discipline Made Easy—Practices from OpenUP and RUP, Addision-Wesley (2006)

Poređenje metodologija
Položaj agilnih procesa

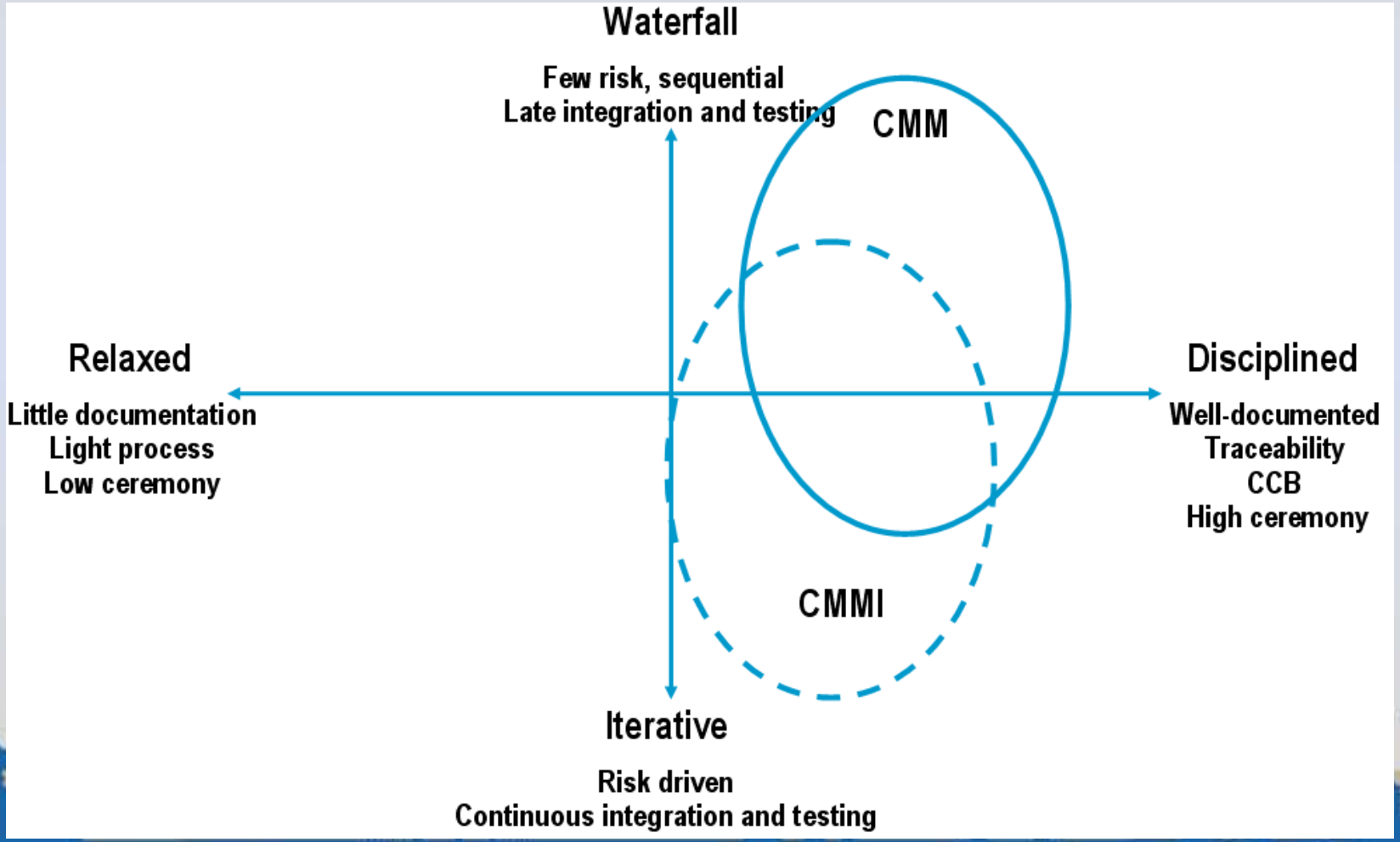


DOD standardi

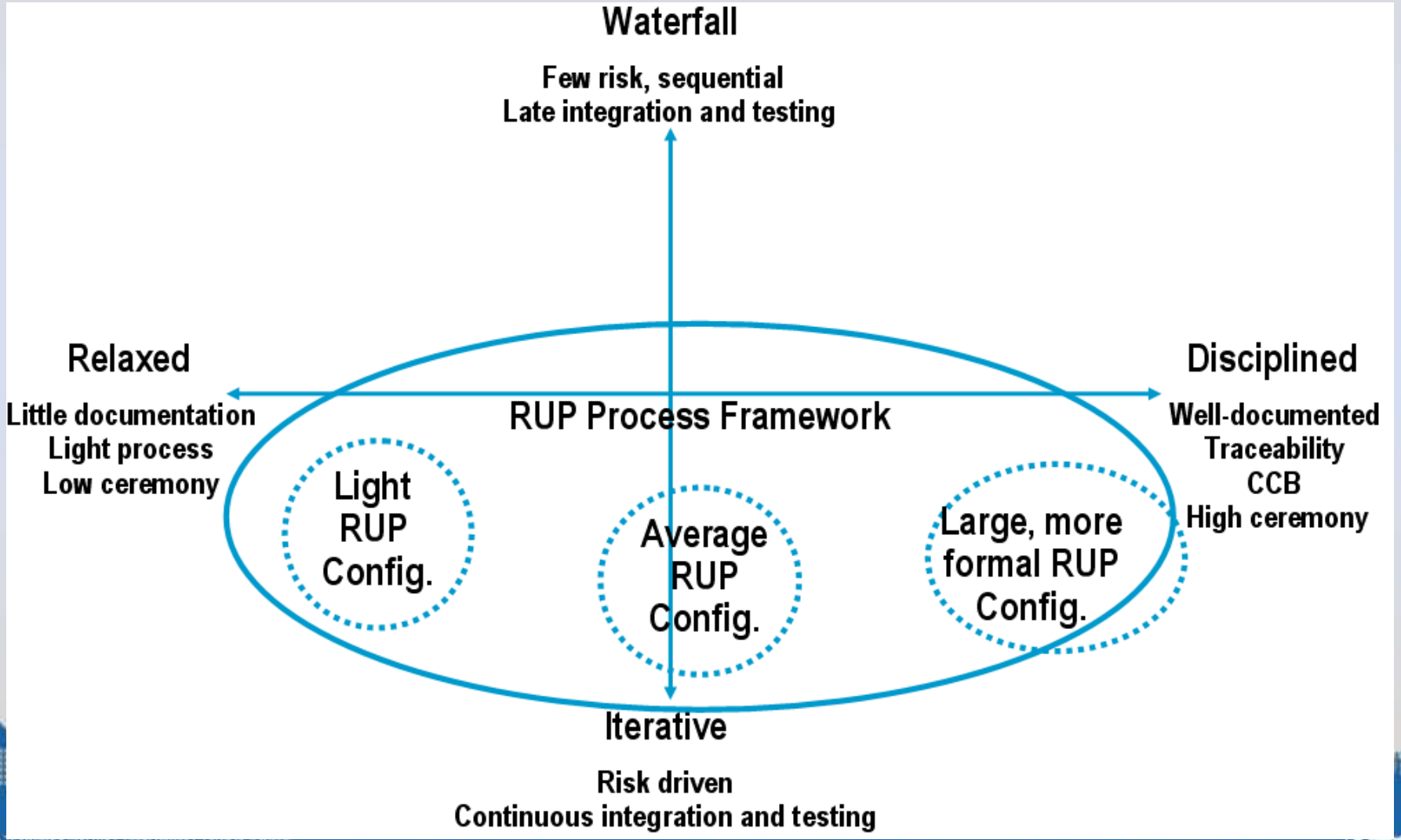
(*Department of Defense standards*)



SEI CMM i SEI CMMI



RUP


Waterfall
Few risk, sequential
Late integration and testing
Relaxed
Little documentation
Light process
Low ceremony
RUP Process Framework
Disciplined
Well-documented
Traceability
CCB
High ceremony
Light RUP Config.
Average RUP Config.
Large, more formal RUP Config.
Iterative
Risk driven
Continuous integration and testing