Projektiranje informacijskih sustava

Uvod

Ak. god. 2011/2012

Literatura

- System Analysis and Design, Fourth Edition; Dennis, Wixom and Roth; Wiley, 2009
- System Analysis and Design, Third Edition; Dennis, Wixom and Roth; Wiley, 2006

Informacijski sustav - definicija

- Informacijski sustav (*Information System – IS system*) je sustav za prikupljanje, pohranu, obradu i pružanje informacija.
- Premda takav sustav može biti implementiran i bez računala danas se pod pojmom informacijskog sustava podrazumijevaju računalni sustavi za prikupljanje, pohranu, obradu i pružanje informacija.

Informacijski sustav

- U ovom kolegiju se bavimo složenim informacijskim sustavima koje se obično primjenjuju u složenim poslovnim sustavima (ili kompanijama, firmama, institucijama, organizacijama,..).
- Poslovni sustav je pravno identificiran entitet koji pruža usluge ili robe korisnicima (Business is legally recognized organizational entity designed to provide goods or services to the customers).

Složeni informacijski sustavi

- Svojstva:

1. za razvoj je potrebno više ljudi organiziranih u timove
2. zahtjevaju pažljivu analiza, dizajn, implementacija i testiranje
3. > 100 KLOC (LOC – *line of code*) (1KLOC = 1000 linija koda)
4. cijena softvera nadmašuje cijenu hardvera
5. sustav je u uporabi više godina i mijenja se s vremenom

Informacijski sustav

- Primjer:
- FESB je akademska institucija koja se bavi nastavnom i znastvenom djelatošću (poslom).
- Informacijski sustavi na FESB-u:
  - e-learning portal
  - informacijski sustav referade
  - računovodstveni informacijski sustav
  - ....

IT pojmovi

- IT (*Information Technology*) tehnologija ili (*Information and Communication Technology* - ICT) se bavi korištenjem tehnologije u obradi i upravljanju informacijama (najčešće u velikim organizacijama) osobito korištenjem računala i računalnog softvera za pretvorbu, pohranu, zaštitu, obradu, prijenos i dohvaćanje informacija.
- Vrlo često se organizacija koja se bavi računalnim tehnologijama i poslovima naziva IT organizacija.
- ITAA (*Information Technology Association of America*) (www.itaa.org) organizacija okuplja američke IT kompanije poput Hawlett-Packarda, Microsofta, IBM-a, Oracle,….

IT pojmovi

- Koja je razlika između informacijske tehnologije (*Information Technology-IT*), računalne znanosti (*Computer Science-CS*), programskog inženjerstva (*Software Engineering-SE*) i računalnog inženjerstva (*Computer Engineering-CEN*)?
- IT stručnjaci identificiraju potrebu za računalnom tehnologijom dok CEN ili SE stručnjaci razvijaju tu tehnologiju.
- CS stručnjaci bave se više teoretskim dijelom računarstva (algoritmi, strukture podataka, kriptografija …) dok CEN ili SE (ta se dva područja često izjednačavaju) stručnjaci imaju znanja kako teoretske postavke pretvoriti u stvarni softver (programski jezici, mreže, OS, …).

System Analysis & Design - SAD

- Projektiranje informacijskih sustava ili SAD (System analysis and design) bavi se razvojem što efikasnijih načina izrade informacijskog sustava.
- Svaki sustav u toku izrade i uvođenja prolazi kroz četiri faze razvoja koje čine proces koji se naziva “system development life cycle - SDLC”.

Metodologije razvoja informacijskih sustava

- Metodologija je formalno definiran pristup razvoju informacijskog sustava sa definiranim koracima razvoja u kojima se koriste definirani postupci ili tehnike ili metode (npr. use case tehnika za identificiranje zahtjeva), te rezultatima svakog koraka razvoja (npr. prikupljanje zahtjeva rezultira dokumentom sa formalno definiranim zahtjevima) tzv. *deliverables*.

Metodologije razvoja informacijskih sustava

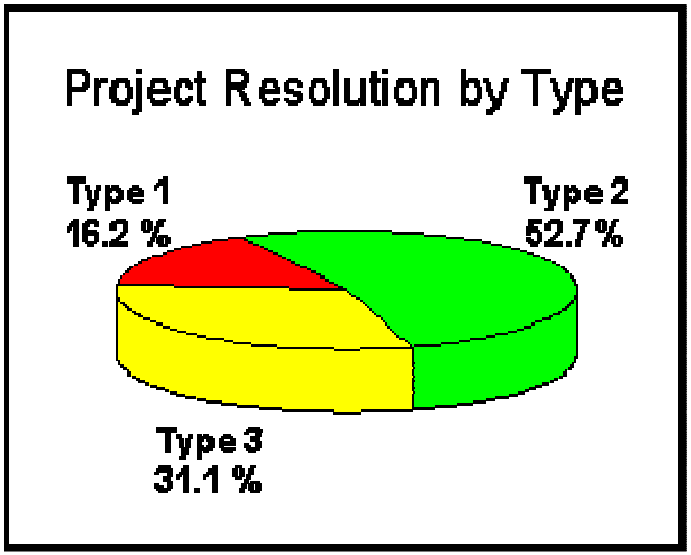
- Koju metodologiju koristiti?
- Ne postoji tzv. “silver bullet” (Frederick Brooks, No Silver Bullet – Essence and Accident in Software Engineering, Information Processing IFIP Proceedings, 1986) ili jedinstveni, 100% pouzdani pristup.
- Metodologija je skupina metoda, postupaka, tehnika i sl. koje se koriste za razvoj informacijskog sustava. Jedna metoda (npr. DFD) se može koristiti u različitim metodologijama.

Razvoj informacijskih sustava

- Metodologije se bave razvojem, implementacijom i održavanjem složenih informacijskih sustava uz ispunjenje osnovnih zahtjeva:
  - na vrijeme,
  - u okviru sredstava,
  - željenih funkcionalnosti,
  - prihvatljivih performansi,
  - ispravan, pouzdan, robustan.

Uspješnost razvoja informacijskih sustava

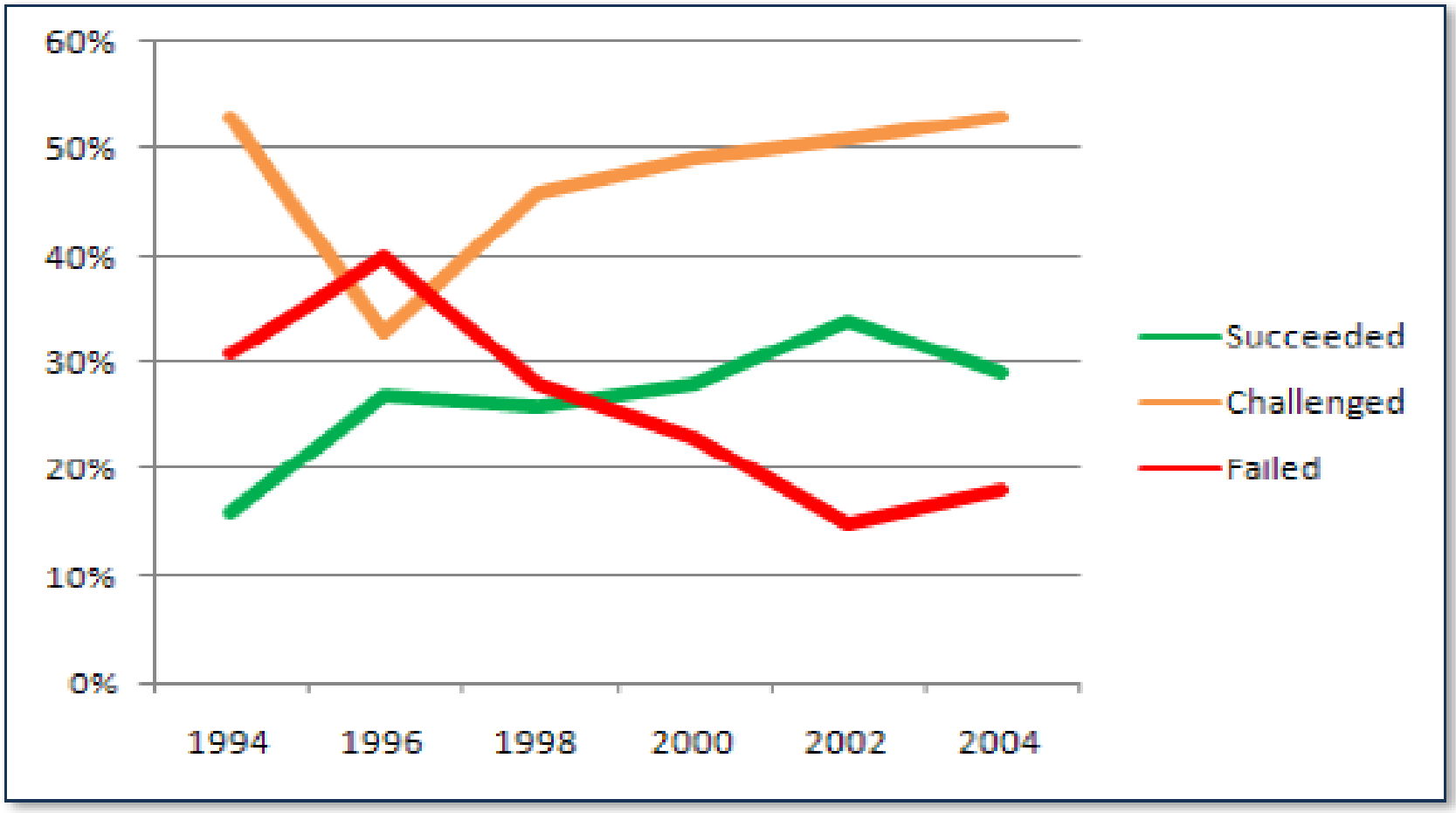
- 1994. godine je Standish Group (http://standishgroup.com/index.php) izvršila analizu 8.380 projekata u državnom i privatnom sektoru u SAD-u i ustanovila:
  - Oko 31% projekata razvoja programske podrške je obustavljeno prije završetka.
  - Oko 53% je kompletirano s prosječnom cijenom koja je za oko 189% premašila predviđenu cijenu, od ovih 53% samo 42% projekata je zadržalo njihove izvorne (originalne) značajke i funkcije.
  - Oko 16% projekata je kompletirano na vrijeme, s predviđenim budžetom i definiranim funkcionalnostima.



Uspješnost razvoja informacijskih sustava

- Podaci Standish Group za 2004. su nešto bolji, ali još uvijek zabrinjavajuće loši.
- Svega 28% projekata može se smatrati uspješnima dok se još uvijek oko 18% započetih projekta napušta.

Uspješnost razvoja informacijskih sustava



Zašto projektiranje informacijskih sustava?

- Cijena pogreške može biti velika:
  - bankrot proizvođača softvera
  - bankrot klijenata koji ovise o softveru
  - financijski gubici zbog grešaka u softveru (*bugova*), nefleksibilnosti sustava ili vremena "nerada" sustava (*downtime*)
  - ljudski gubici u sigurnosno kritičnim sustavima
- Profitabilnost softvera ovisi znatno o troškovima održavanja i *upgrade* softvera
  - zahtjevi za promjenjivim dizajnom
  - modularnost, evolucija, portabilnost, odvajanje podataka od funkcionalnosti

- http://www.pcauthority.com.au/Feature/264645,ten-of-the-worlds-most-disastrous-it-mistakes.aspx
- http://en.wikipedia.org/wiki/Northeast_Blackout_of_2003
- http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html
- http://www.youtube.com/watch?v=kYUrqdUyEpI

System Development Life Cycle (SDLC)

- Četiri faze izrade:
  - Planiranje
  - Analiza
  - Dizajn
  - Implementacija
- Svaka faza sastoji se od niza postupaka koji se oslanjaju na različite tehnike, pristupe, alate, a rezultat svake faze su između ostalog i različiti dokumenti koji opisuju projekt.
- Za razvoj informacijskih sustava definirane su različite formalne metodologije, ali se u praksi koriste i različiti neformalni pristupi.

Planiranje

- Osnovni proces shvaćanja zašto se izrađuje sustav i kako će projektni tim razvijati sustav.
- Sastoji se od dva koraka:
  1. Pokretanje projekta (*project initiation*) je faza u kojoj se procjenjuje poslovna vrijednost sustava (koliko će smanjiti troškove ili povećati prihode kompanije). Dobiveni dokumenti su studija izvedivosti ili provedivosti (*feasibility analysis*) i zahtjev sustava (*system request*).
  2. Upravljanje projektom (*project management*) je faza u kojoj se izrađuje radni plan (*workplan*), određuje raspodjela posla te definiraju tehnike izrade, kako bi se pomoglo projektnom timu u kontroliranju i usmjeravanju projekta kroz cijeli ciklus SDLC-a. Dobiveni dokument je projektni plan (*project plan*).

Analiza

- Analiza sustava treba dati odgovore na pitanja tko će biti korisnik sustava, što će sustav raditi, i gdje i kada će se koristiti (dobro mjesto za korištenje dijagrama slučajeva korištenja).
- U ovoj fazi trebalo bi proučiti postojeće sustave (vrlo rijetko razvoj nekog informacijskog sustava kreće od nule), odrediti što se može unaprijediti te postaviti osnovni koncept sustava.

Analiza

- Sastoji se od tri koraka:

1. Analiza strategije uključuje analizu već postojećeg sustava (*as-is system*), a služi usmjeravanju projektnog tima u izradi novog sustava (*to-be system*).
2. Prikupljanje zahtjeva (*requirements gathering*) služi u svrhu izrade koncepta za novi sustav, koji će služiti kao predložak za izradu modela za analizu.
3. Prijedlog sustava (*system proposal*) se predstavlja sponzoru projekta i drugim ključnim osobama koje odlučuju o tome da li se izrada sustava nastavlja ili ne. Prijedlog sustava opisuje koje poslovne zahtjeve sustav treba ispunjavati.

Dizajn

- Rezultat dizajna sustava treba biti odluka o načinu funkcioniranja sustava u smislu određivanja:
  - potrebnog hardvera, softvera, mrežne infrastrukture;
  - korisničkog sučelja, formi i izvještaja koji će se koristiti u sustavu;
  - specifičnih programa, baza i datoteka koje će biti potrebne.

Dizajn

- Sastoji se od četiri koraka:
  1. Strategija dizajna (*Design Strategy*): Strategija dizajna sustava određuje hoće li se željeni sustavu razvijati unutar organizacije, hoće li se unajmiti neka druga kompanija za razvoj sustava ili će se kupiti već gotovi programski paket.
  2. Dizajn arhitekture (*Architecture Design*): Određuje hardver, softver i mrežnu infrastrukturu koja će se koristiti. Često se radi o proširenju i pojačanju postojeće arhitekture.
  3. Specifikacija baza i datoteka (*Database and File Specifications*): Definiraju su točno podaci koji će se pohranjivati i gdje će se pohranjivati.
  4. Dizajn programa (*Program Design*): Definira koje programe je potrebno napraviti i što će ti programi raditi.

Implementacija

- U fazi implementacije sustav se ili razvija ili se kupuje (ukoliko je strategijom dizajna definirano da je bolje kupiti postojeći softverski sustav).
- Ova faza je obično najduža i zahtjeva najviše financijskih sredstava.

Implementacija

- Sastoji se od tri koraka:

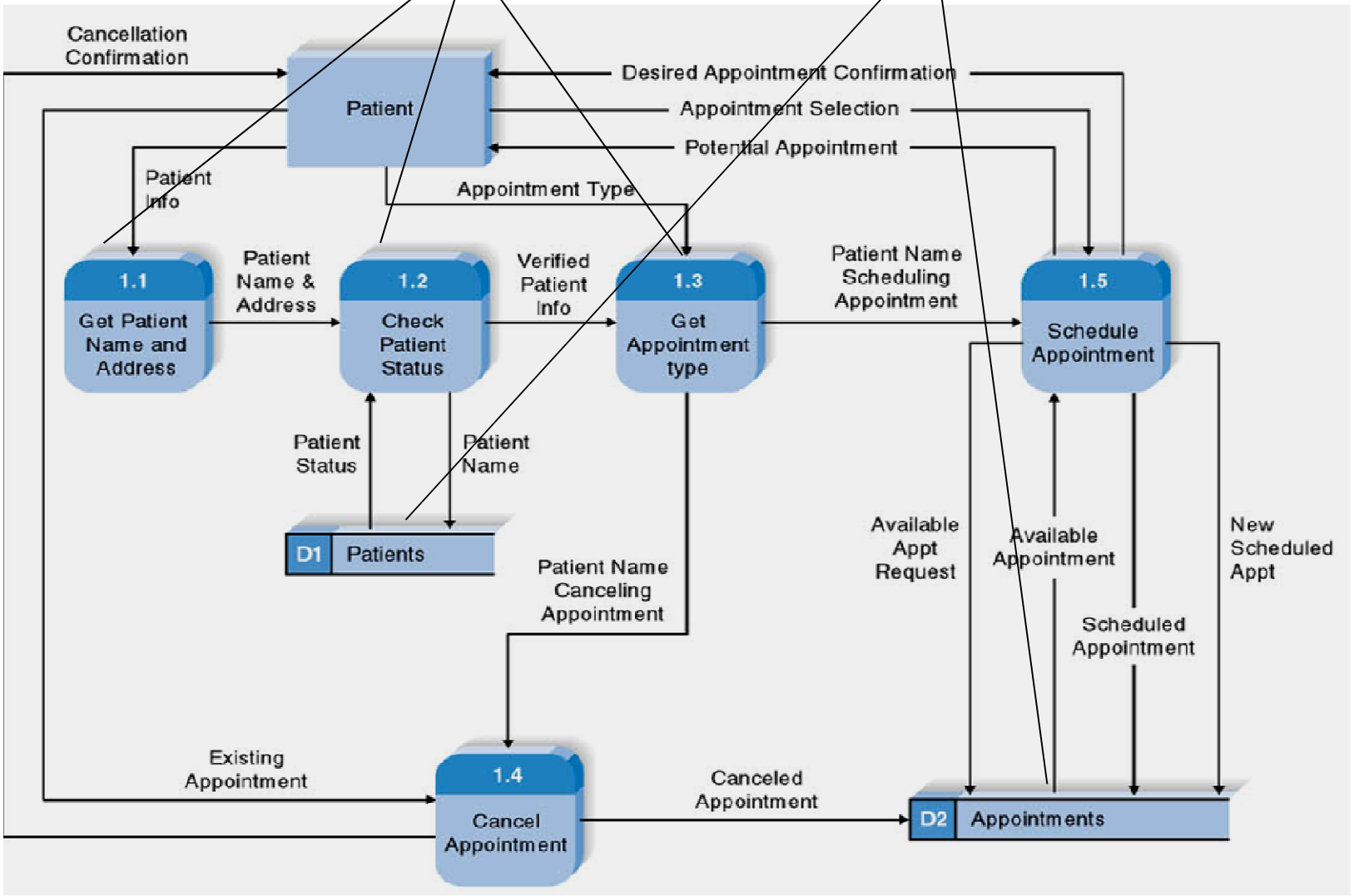
1. Izrada sustava (*System Construction*): U ovom koraku sustav se izrađuje i testira kako bi se osiguralo da se ponaša u skladu sa zahtjevima.
2. Instalacija (*Installation*): U postupku instalacije sustav se stavlja u rad. Ukoliko je postojala starija verzija sustava (čest slučaj) stari sustav se može ugasiti prije no što se novi stavi u funkciju, ili novi sustav može neko vrijeme raditi paralelno sa starim zbog dodatnog testiranja ili se novi sustav uvodi dio po dio, a ne sve odjednom. Ova faza uključuje i obuku korisnika.
3. Plan podrške (*Support Plan*): Definira post-implementacijsku reviziju sustava (*upgrade* i sl.).

METODOLOGIJE RAZVOJA INFORMACIJSKIH SUSTAVA

Metodologije razvoja sustava

- Metodologije razvoja mogu se podijeliti na:
  1. usmjerene na proces koji se informacijskim sustavom obuhvaća (sustav za obračunavanje plaća: proračun radnih sati za mjesec, obračun prethodnih plaća, izračun plaće za ovaj mjesec,….)
  2. usmjerene na podatke koji su obuhvaćeni sustavom (sustav za obračunavanje plaća: podaci o radnim satima, podaci o poreznim olakšicama, ….)
  3. objektno-orijentirane metodologije obično balansiraju između procesno orijentiranog i podatkovno orijentiranog pristupa informacijskom sustavu

Procesi ↔ podaci



Metodologije razvoja sustava

- Metodologije s naglaskom na proces:
  - Bitno je fokusirati se na definiranje aktivnosti povezanih sa sustavom
  - Cilj je prikazati sustav kao skup procesa s informacijom koja ulazi i izlazi iz njih (tok informacija = flow)

Metodologije razvoja sustava

- Metodologije s naglaskom na podatke:
  - Bitno je definirati sadržaj spremnika podataka i način na koji su oni organizirani.
  - Koriste se modeli podataka kao osnova formalnog opisa sustava.

Metodologije razvoja sustava

- Objektno orijentirane metodologije:
  - Balans između prethodne dvije metodologije (proces ↔ podaci)
  - Za opis sustava najčešće se koriste UML dijagrami
  - Sustav se opisuje kao skup objekta koji uključuju i procese i podatke.

Metodologije razvoja sustava

- Podjela metodologija obzirom na formalne zahtjeve, brzinu i složenost razvoja sustava:

1. Metodologije strukturiranog dizajna (*Structured design*) (1980-tih) uvode formalne modele sustava, najčešće u obliku dijagrama. Prednosti su u pouzdanosti zbog strogih formalnih zahtjeva, a nedostatci u sporosti i složenosti.
2. RAD (*Rapid Application Development*) metodologije (1990-te) ubrzavaju razvoj sustava. Mogu kao i metodologije strukturiranog dizajna biti ili procesno orijentirane ili podatkovno orijentirane ili objektno-orijentirane.
3. Metodologije agilnog razvoja (*Agile Development*) još više ubrzavaju razvoj sustava, ali na štetu formalnih zahtjeva. Ovo je najnovija kategorija metodologija i još se razvija.

STRUKTURIRANI DIZAJN

- Primjenjuje se formalni “korak po korak” pristup SDLC-u, faze se strogo odvojene, a sustav se razvija korak po korak tj. faza po faza
- Uvodi se korištenje formalnog modeliranja ili tehnika koje koriste dijagrame da bi se opisali glavni poslovni procesi sustava

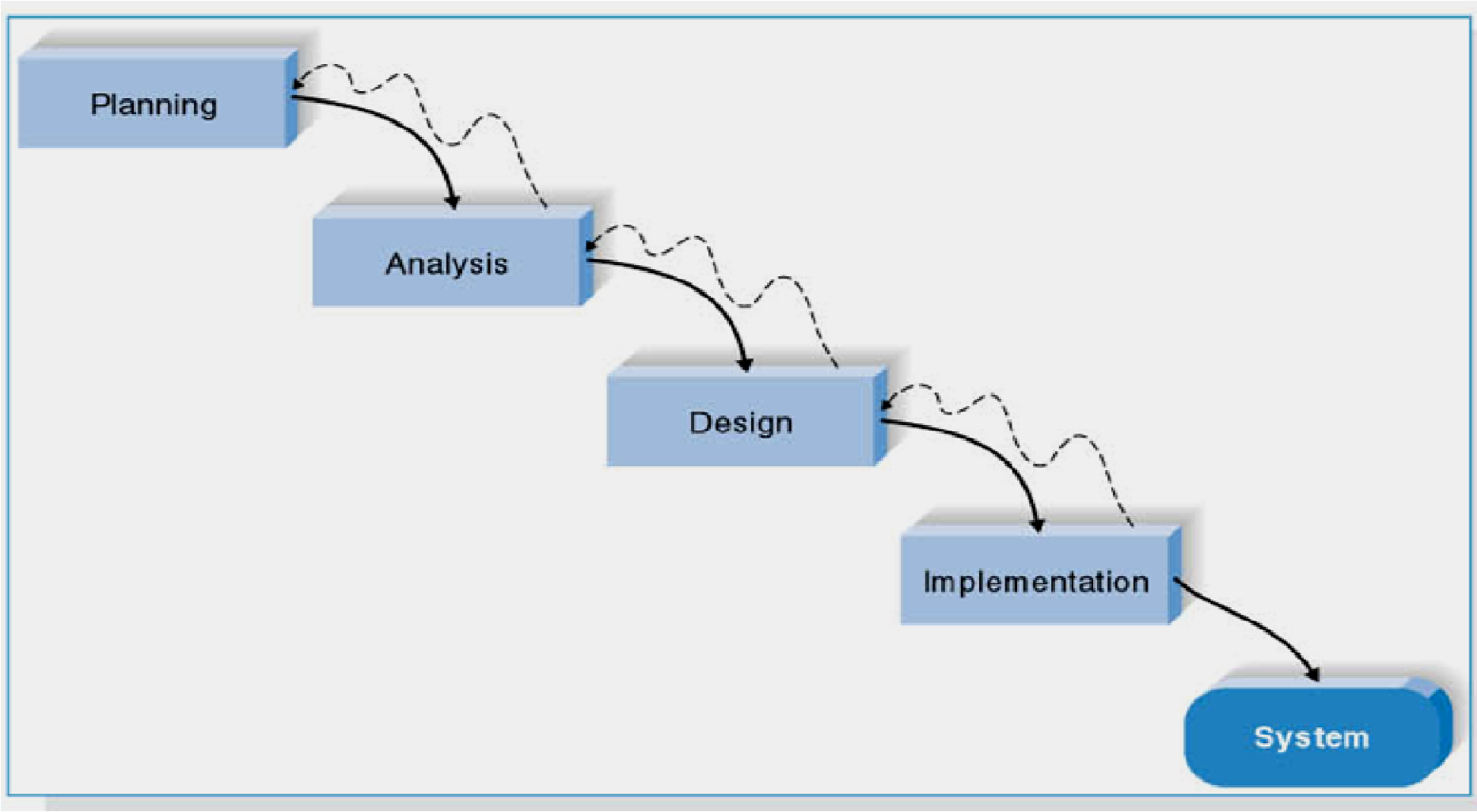
Metodologija Vodopada (*waterfall development*)

- Analitičari (razvojni tim) i korisnici sekvencijalno razvijaju fazu po fazu projekta
- PREDNOSTI
  - Zahtjevi sustava su definirani puno prije nego počne samo programiranje
  - Promjene zahtjeva su svedene na minimum kako se projekt razvija

Metodologija Vodopada

- NEDOSTACI
  - Dizajn mora kompletno biti specificiran prije nego počne programiranje
  - Predugo vremena protekne između završetka prijedloga sustava u fazi analize i same isporuke sustava

Metodologija Vodopada



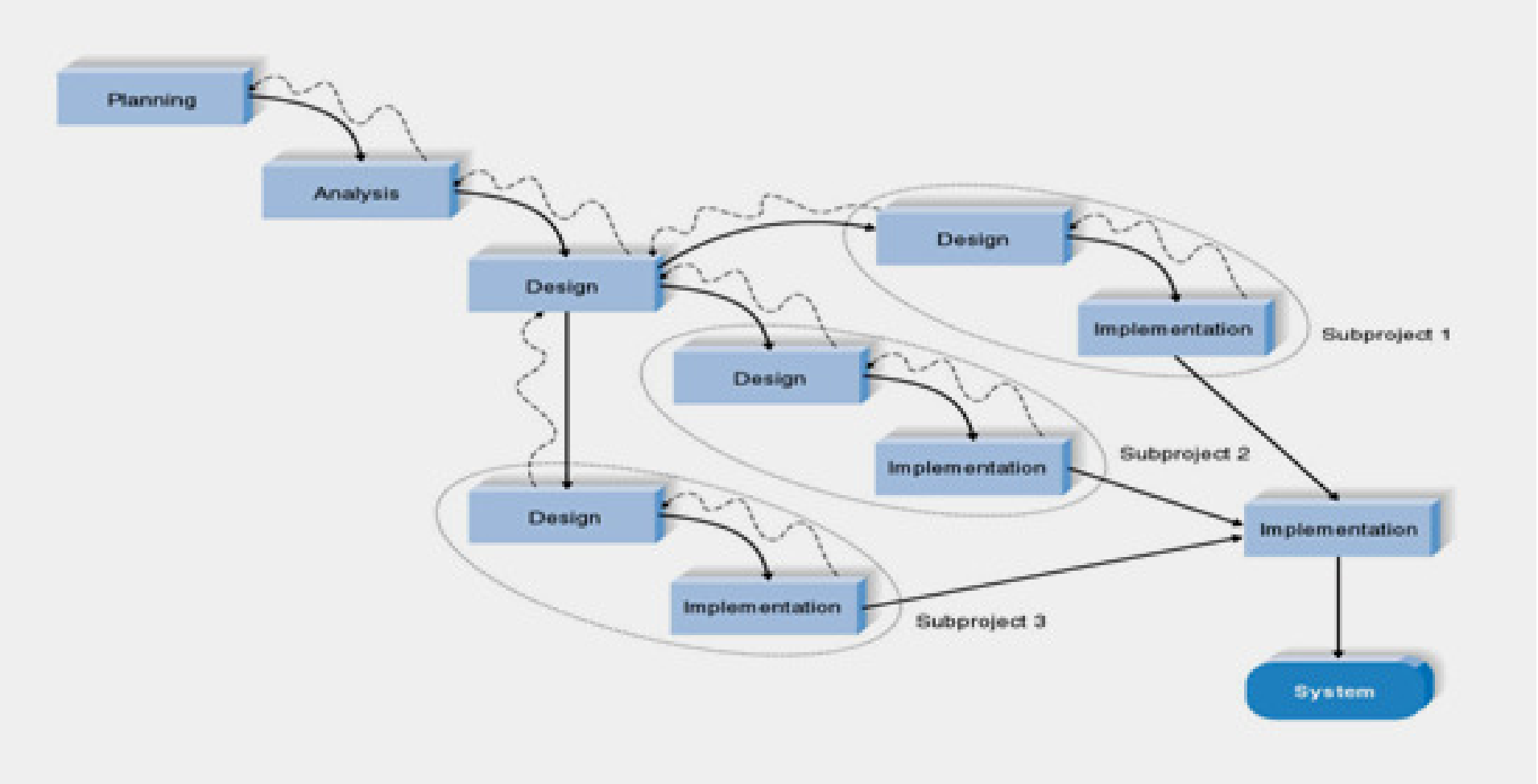
Metodologija Paralelnog razvoja (*parallel development*)

- Cilj je smanjiti vremenski interval koji protekne između analize i isporuke sustava kod vodopadnog pristupa.
- Umjesto da se cijeli sustav izrađuje sekvencijalno (dio po dio), sustav se u fazi dizajna dijeli na više podsustava koji se paralelno izrađuju i integriraju.
- Na kraju se vrši finalna integracija svih podsustava te se vrši isporuka.

Metodologija Paralelnog razvoja

- PREDNOSTI
  - Vrijeme koje protekne do isporuke se smanjuje, što dovodi do toga da je vjerojatnost promjene početnih zahtjeva zbog promjene u stvarnom poslovanju kompanije smanjena
- NEDOSTACI
  - Ako podsustavi nisu samostalni, dolazi do problema u integraciji ako planiranje nije dobro izvršeno

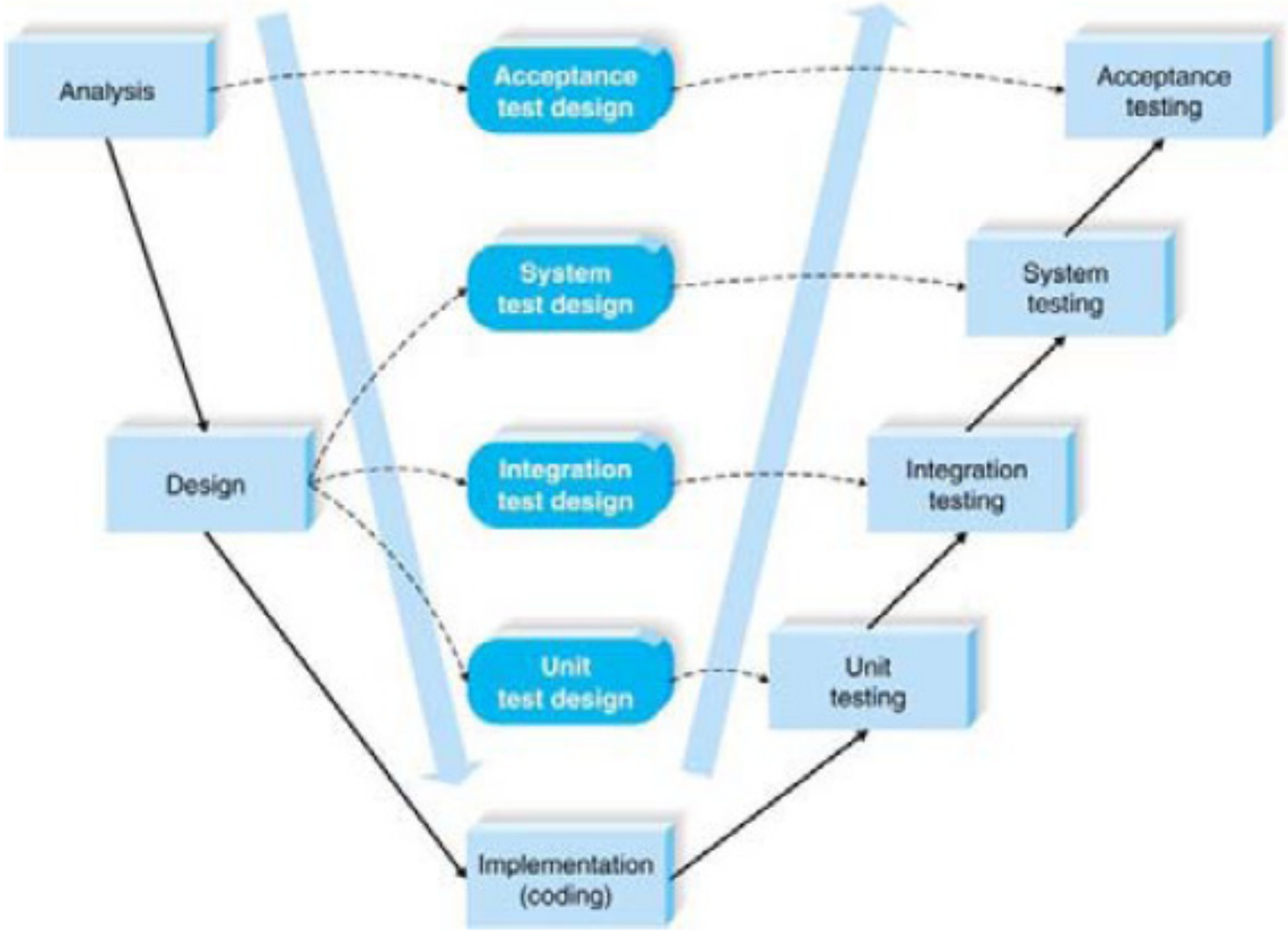
Metodologija Paralelnog razvoja



V-model

- V-model je također oblik vodopanog razvoja.
- Osnovno svojstvo je da se tijekom specifikacije zahtjeva i dizajna komponenti odmah definiraju i testiranja.

V-model


Analysis
Acceptance test design
Acceptance testing
System test design
System testing
Design
Integration test design
Integration testing
Unit test design
Unit testing
Implementation (coding)

BRZI RAZVOJ APLIKACIJE

(Rapid Application Development - RAD)

- Metodologija se razvija ’90-ih kao rješenje nedostataka strukturiranog razvoja. SDLC faze se prilagođavaju kako bi se što prije dijelovi sustava razvili i dostavili kupcu.
- Oslanjaju se na korištenje CASE (Computer-Aided Software Engineering) alata koji ubrzavaju proces izrade, korištenje viših programskih jezika sa mogućnošću vizualnog razvoja, korištenje alata za automatsko generiranje kôda iz specifikacija iz faze dizajna, itd.

CASE

- *Computer-Aided Software Engineering* (CASE) obuhvaća različite programske alate za podršku aktivnosti u softverskom procesu kao što su:
  - analiza zahtjeva (*requirements analysis*),
  - modeliranje sustava (*system modeling*),
  - testiranje (*debugging and testing*)…
- Sve metodologije se danas koriste CASE alatima kao što su različiti uređivači koda, moduli za analizu i provjeravanje modela sustava prema postavljenim pravilima, kao i generatori izvještaja koji pomažu u izgradnji sustava dokumentacije.

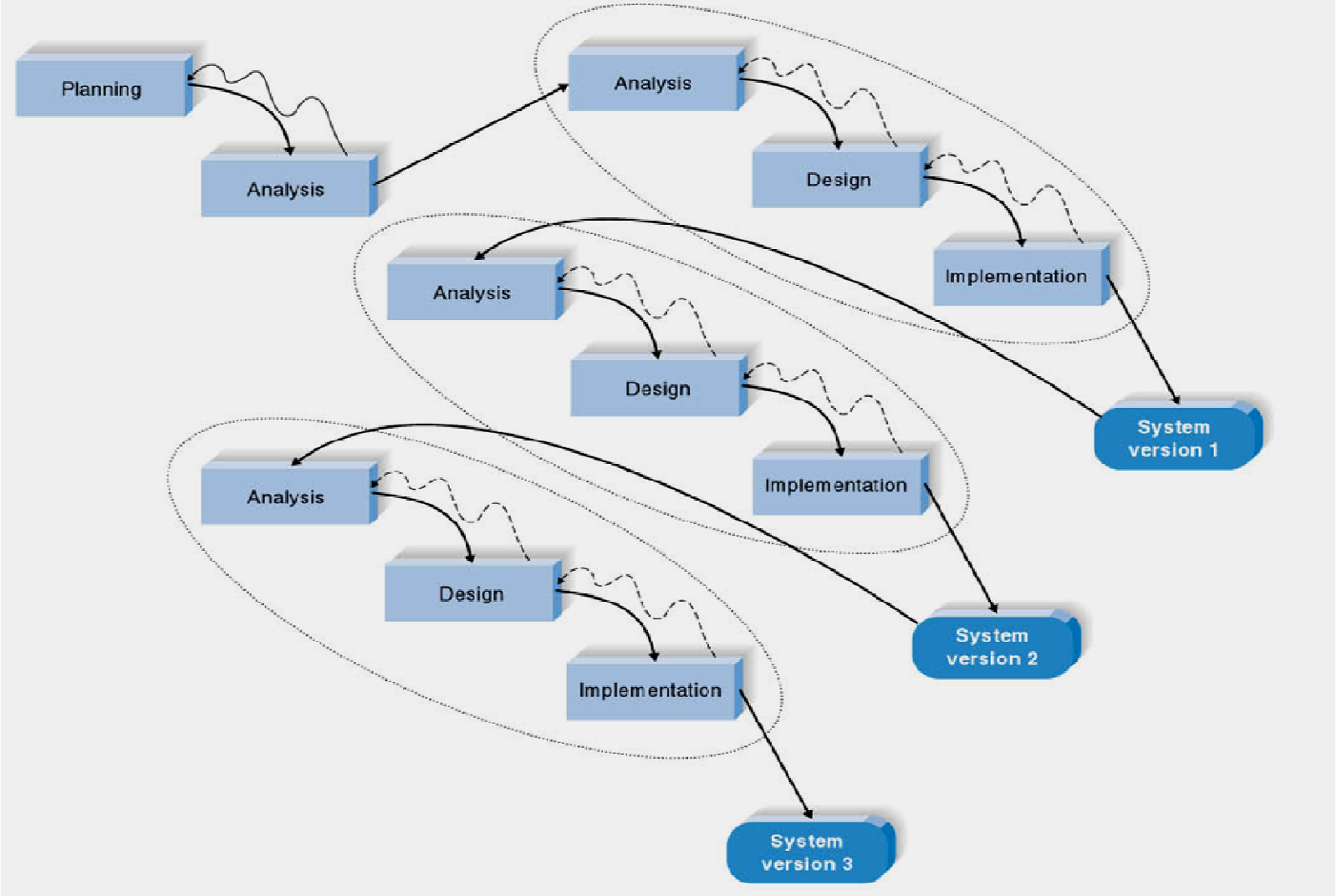
CASE

- CASE alate za podržavanje implementacije i testiranja koda (*debuggers*, *program analysis systems*, *test case generators*, *program editors*) nazivamo CASE alatima niske razine
- CASE alati koji podržavaju analizu i oblikovanje koda se nazivaju CASE alati visoke razine. CASE visoke razine danas uključuju čak i generatore izvornog programskog kôda iz specifikacija i sl.

Metodologija razbijanja u faze ili iterativni razvoj (*phased development or iterative development*)

- Cijeli sustav se “razbija” u seriju verzija koje se razvijaju sekvencijalno sa mini-vodopadnim pristupom.
- Zahtjevi se kategoriziraju kroz verzije, a najvažniji i neophodni su uključeni u prvu verziju sustava.
- Nakon toga faza analize vodi u dizajniranje i implementaciju, ali samo prve verzije.
- Nakon što je svaka verzija sustava završena, odmah počinje rad na slijedećoj verziji.
- Nedostatak je što je izuzetno bitno točno odrediti najvažnije zahtjeve koji moraju biti uključeni u prvu verziju sustava. Ako se pogriješi u prvoj verziji cijeli sustav će vjerojatno biti loš.

Metodologija razbijanja u faze



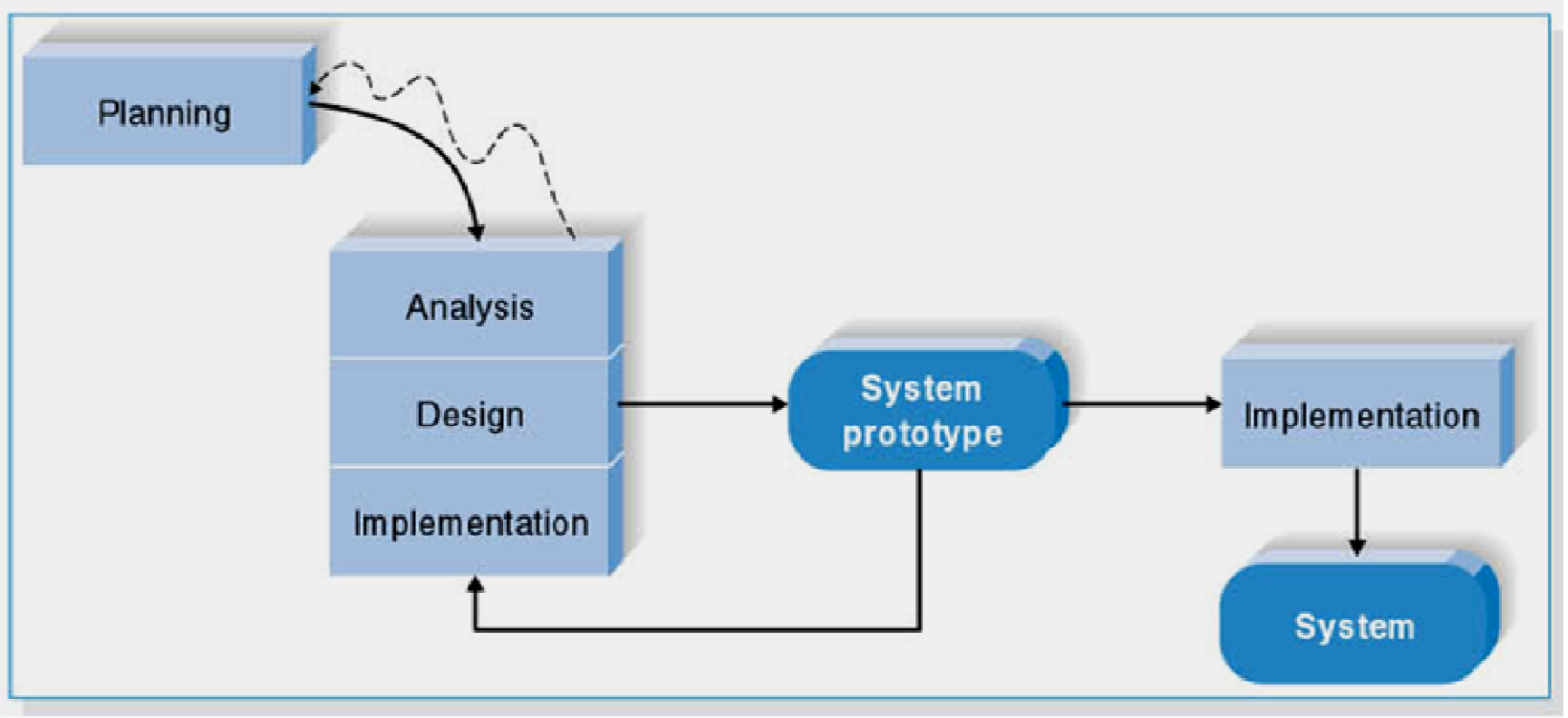
Metodologija izrade prototipa (*prototyping*)

- Istodobno se izvode faze analize, dizajna i implementacije što znači da analiza i dizajn nisu tako dobri kao kod faznog pristupa, ali je sustav prije isporučiv korisniku (u vrlo ograničenoj, neizbrušenoj verziji - prototipu).
- Prototip je mala verzija sustava s minimalnim setom funkcionalnosti koja se odmah počne razvijati da bi se sustav što prije isporučio korisniku.
- Korisnik dobiva na uvid prototipove i nakon što se zaključi da je prototip dovoljno funkcionalan postaje sustav.

Metodologija izrade prototipa

- Prednost je što korisnici kontinuirano imaju uvid u stanje izrade sustava (nema dugih perioda izrade i čekanja).
- Problem ove metode je u tome što se nakon određenog vremena mogu uočiti propusti nastali u prvim prototipovima jer analiza i dizajn nisu detaljni kako kod faznog razvoja.
- Prvi prototipovi su temelj svih budućih verzija sustava te njihova promjena rezultira velikim utroškom vremena.

Metodologija izrade prototipa



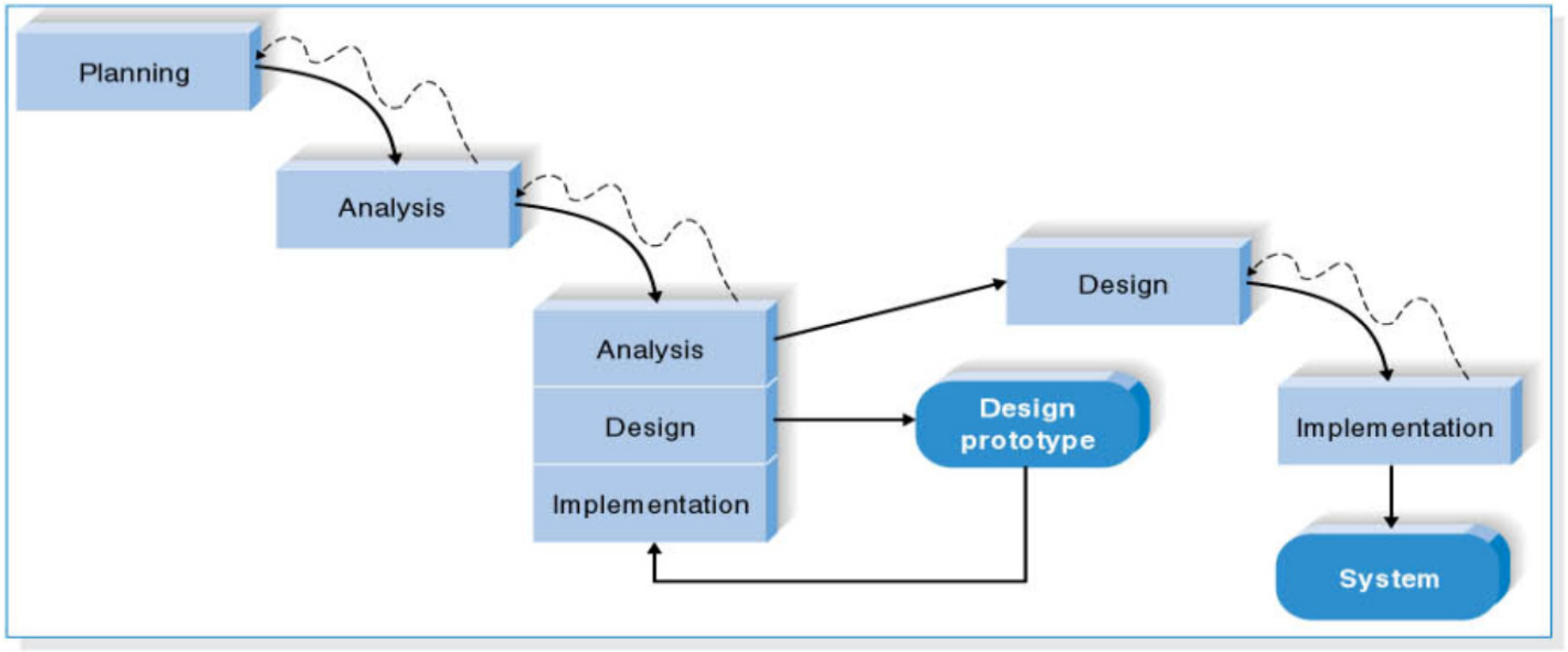
Metodologija odbačenih prototipova (*throwaway prototyping*)

- Metodologija odbačenih prototipova (*throwaway prototyping*) je slična prethodnoj metodologiji, samo što se sada prototip koristi samo kao sredstvo za osiguravanje ispunjavanja zahtjeva korisnika (nakon toga se “baca”) i to obično samo za dijelove sustava za koje korisnički zahtjevi nisu do kraja identificirani pa napravljeni prototip olakšava korisniku definiranje zahtjeva.

Metodologija odbačenih prototipova

- Korisnik dobiva na uvid prototip dizajna sustav, te nakon što se on odobri kreće se u izradu samog sustava, na osnovu tog prototipa, ali sam prototip se odbacuje.
- Ova metodologija produžuje vrijeme izrade u odnosu na prethodnu, ali zato u pravilu rezultira stabilnijim i pouzdanijim sustavom.

Metodologija odbačenih prototipova



AGILNI RAZVOJ

- Najnovije metodologije (2000-tih) su metodologije agilnog (žurnog) razvoja. Cilj je postići što brži i efikasniji razvoj aplikacije. (http://agilemanifesto.org/)
- Odbacuje se dio modeliranja i dokumentacije, a uvodi se jednostavni iterativni razvoj.
- Neke od glavnih metoda su:
  - Ekstremno programiranje (*Extreme programming*-XP)
  - Scrum
  - Dynamic Systems Development Method (DSDM)

Ekstremno programiranje

- Bazira se na:
  - Komunikaciji (između razvojnog tima i korisnika, te između članova razvojnog tipa)
  - Jednostavnosti (koristi se KISS (Kip It Simple, Stupid) princip u razvoju aplikacije, koristi se postupak refaktoriranja kôda i slične metode.
  - Povratnoj informaciji (*feedback*) (korisnici su kontinuirano uključeni u razvoj sustava, te u svakom koraku daju povratnu informaciju timu o dodatnim zahtjevima i sl.)

Refaktoriranje kôda

- Refaktoriranja kôda (code refactoring) je postupak kojim se mijenja izvorni kôd u svrhu poboljšanja koda smanjenjem složenosti (complexity) kôda, povećanjem preglednosti (readability) kôda, složenost održavanja (maintainability) kôda i sl. i to bez promjene funkcionalnosti kôda nego samo promjene nefunkcionalnih karakteristika kôda.
- Mijenjamo unutarnju strukturu kôda bez mijenjanja vanjskog ponašanja kôda.

Refaktoriranje kôda

- Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.
- Izvor: http://martinfowler.com/refactoring/
- Refaktoriranja kôda su mali zahvati u kôdu pa u pravilu ne uzrokuju probleme. http://martinfowler.com/refactoring/catalog/index.html

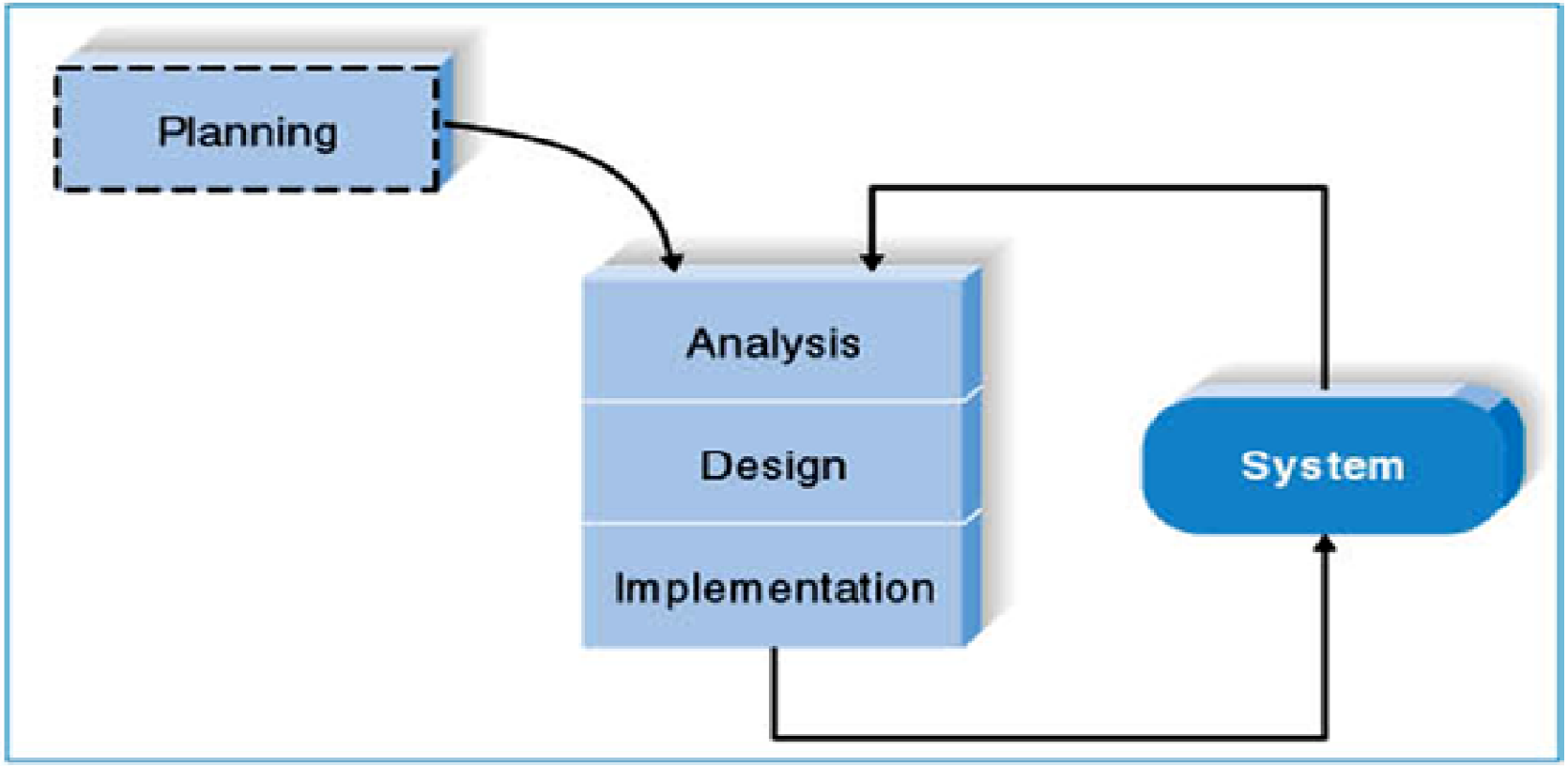
Ekstremno programiranje

- Glavni principi u izradi uspješnog sustava
  - Kontinuirano testiranje
  - Što jednostavnije programiranje, u parovima
  - Što bolja interakcija s korisnicima (ponekad su dio razvojnog tima)
- Nakon površnog procesa planiranja, projektni tim iterativno provodi faze analize, dizajna te na kraju implementacije sustava.

Ekstremno programiranje

- Namijenjeno za timove do 10 članova uz uvjete:
  - Visoka motivacija
  - Osjećaj zajedništva
  - Stabilnost
  - Iskustvo
- Najbolje koristiti za izradu manjih i srednje velikih aplikacija, zbog nedostatka dokumentacije.

Ekstremno programiranje



Izbor odgovarajuće metodologije razvoja

- S obzirom na broj i raznolikost pristupa, izbor nije lak. Nijedna metodologija nije najbolja u svakom slučaju i svaka ima određene prednosti i nedostatke.
- U praksi kompanije obično imaju:
  - Jednu uhodanu metodologiju
  - Više metodologija koje se koriste po potrebi
  - Ništa od navedenog
- Postoje kriteriji za odabir metodologije.

Izbor odgovarajuće metodologije razvoja

	Vodop. razvoj	Paral. razvoj	V-model	Fazni razvoj	Protot.	Odbačeni prototip.	XP
Nejasni korisnički zahtjevi	loše	loše	loše	dobro	izvrsno	izvrsno	izvrsno
Nepoznata tehnologija	loše	loše	loše	dobro	loše	izvrsno	loše
Složeni sustav	dobro	dobro	dobro	dobro	loše	izvrsno	loše
Pouzdan sustav	dobro	dobro	izvrsno	dobro	loše	izvrsno	dobro
Kratki rok isporuke	loše	dobro	loše	izvrsno	izvrsno	dobro	izvrsno
Transp. toka izrade	loše	loše	loše	izvrsno	izvrsno	dobro	dobro

Definiranost korisničkih zahtjeva

- Ako zahtjevi nisu jasno definirani, ne može ih se lako protumačiti niti objasniti programerima.
- Korisnici moraju znati kako tehnologija funkcionira da znaju što mogu od sustava očekivati.
- RAD metode s prototipovima, te agilne metode (uz prisutnost korisnika u projektnom timu) su najbolje rješenje ovog problema.

Poznavanje tehnologije

- Ako se koristi nova tehnologija pri razvoju moguće je da se zbog nepoznavanja ili krivog korištenja naiđe na neplanirane probleme (npr. prije ste radili u ASP.NET web aplikacije, a sada se trebate prebaciti na JSP).
- Moguće je da odabrana tehnologija nema mogućnosti izvršiti postavljene korisničke zahtjeve, a očekivalo se da može.

Poznavanje tehnologije

- Pogodna metoda je metoda odbačenih prototipova jer omogućava i upoznavanje sa tehnologijom i provjeru mogućnosti tehnologije. Također i metoda faznog razvoja (tj. razbijanja na faze) jer se tehnologija upozna prije no što se dođe do konačne verzije sustava.
- Zašto prototipovi baš i nisu najbolja metoda u ovom slučaju?
- Jer kod prototipova postavljene verzije sustava ne odbacujete već na njih nadograđujete. Ako ste nešto pogriješili ili previdili na početku to će vam kasnije biti teško nadoknaditi.

Kompleksnost sustava

- Kompleksni sustavi zahtijevaju opreznu i detaljnu analizu i dizajn, pogotovo ako njihovo loše funkcioniranje može rezultirati velikom štetom bilo kakvog oblika.
- Projektni timovi koji slijede fazni razvoj manje pozornosti pridodaju analizi cijelog problema.
- Iz tog razloga se za ovakve sustave predlaže model odbačenih prototipova.

Pouzdanost sustava

- Sustavi za npr. medicinsku opremu ili nuklearne elektrane na prvom mjestu moraju biti pouzdani, dok nekim sustavima poput igara to ne mora biti važna karakteristika.
- Za dizajniranje pouzdanih sustava predlaže se korištenje throwaway prototipa (ne prototipa sustava) zbog toga što se na taj način vrši dvostruka kontrola, prvo zahtjeva, a onda se programiranje vrši odvojeno, s već gotovim predloškom u obliku prototipa.

Kratki vremenski rokovi

- Za sustave s kratkim rokom isporuke koriste se metode agilnog razvoja, jer je njihova osnovna namjena ubrzavanje razvoja.
- Također se koriste i RAD metode prototipa sustava i faznog razvoja, jer se sustav može isporučiti brzo s djelomičnom funkcionalnošću.

Transparentnost toka izrade

- Vrlo je teško predvidjeti tok izrade sustava, pogotovo neiskusnim timovima.
- Metode RAD-a pomažu u prepoznavanju i uklanjanju rizičnih faktora, kako bi se razvoj odvijao u pretpostavljenim rokovima.

SPOSOBNOSTI I ULOGE ČLANOVA PROJEKTNOG TIMA

- Kako bi projektni tim bio uspješan, mora se sastojati od skupa iskusnih stručnjaka iz različitih područja.
- Sposobnosti članova se mogu podijeliti na:
  - Tehničke,
  - Poslovne,
  - Analitičke,
  - Međuljudske,
  - Organizacijske….

SPOSOBNOSTI I ULOGE ČLANOVA PROJEKTNOG TIMA

- Unutar tima, podjela na uloge je slijedeća:
  - Poslovni analitičar
  - Sistem analitičar
  - Infrastrukturni analitičar
  - Analitičar promjena (Change Management)
  - Voditelj ili menadžer projekta

SPOSOBNOSTI I ULOGE ČLANOVA PROJEKTNOG TIMA

Role	Responsibilities
Business analyst	Analyzing the key business aspects of the system
	Identifying how the system will provide business value
	Designing the new business processes and policies
Systems analyst	Identifying how technology can improve business processes
	Designing the new business processes
	Designing the information system
	Ensuring that the system conforms to information systems standards
Infrastructure analyst	Ensuring the system conforms to infrastructure standards
	Identifying infrastructure changes needed to support the system
Change management analyst	Developing and executing a change management plan
	Developing and executing a user training plan
Project manager	Managing the team of analysts, programmers, technical writers, and other specialists
	Developing and monitoring the project plan
	Assigning resources
	Serving as the primary point of contact for the project