Projektiranje informacijskih sustava

SDLC faza planiranja - Upravljanje projektom

Ak. god. 2011/2012

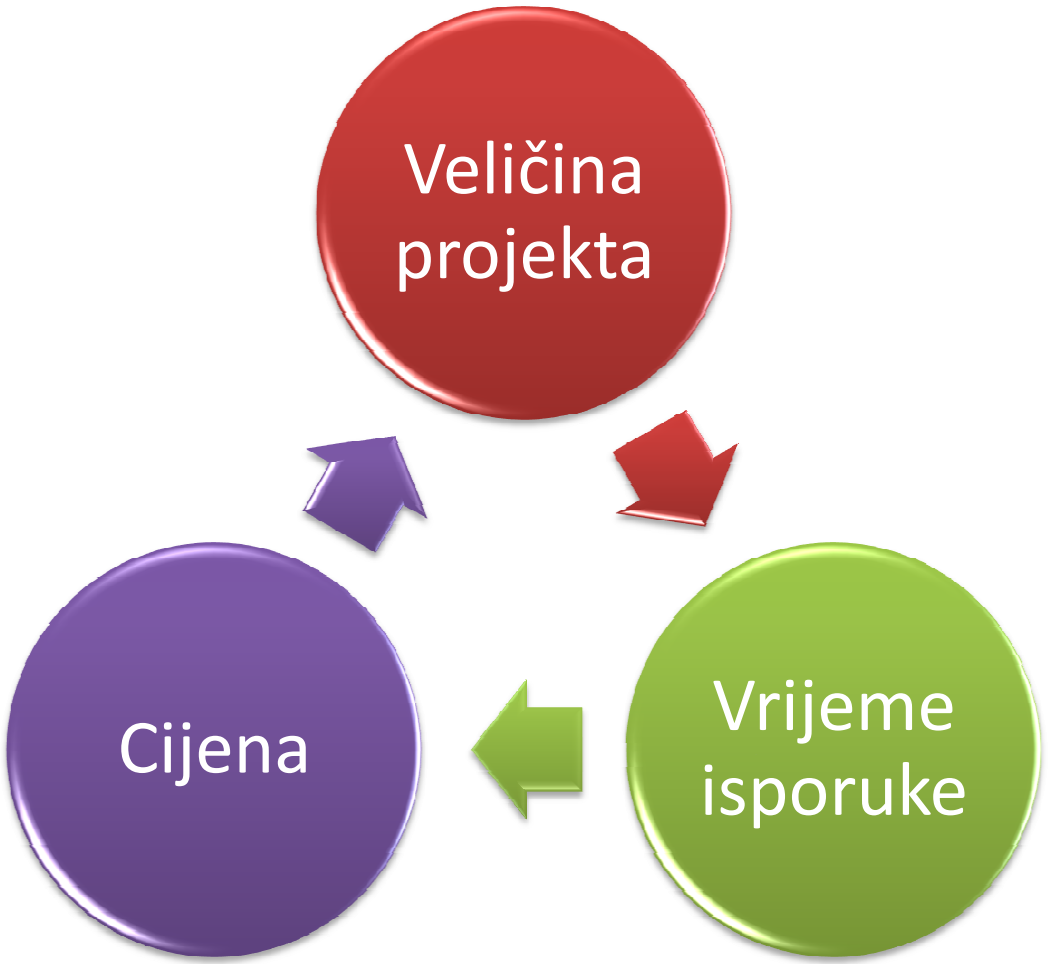
Upravljanje projektom

- Upravljanje projektom (*project management*) je proces planiranja i kontroliranja razvoja sustava unutar zadanog vremenskog okvira uz minimalnu cijenu koštanja i ispravne funkcionalnosti sustava.
- Zahtjevi koje upravljanje projektom treba ispuniti su:
  1. Sustav se mora razviti unutar točno određenog vremenskog perioda i uz minimalnu cijenu.
  2. Gotov sustav treba ispunjavati svoje funkcionalne i nefunkcionalne zahtjeve.
- Upravljanje projektom započinje u fazi planiranja, ali se nastavlja tijekom cijelog SDLC ciklusa.
- Upravljanjem projektom se bavi voditelj projekta (*project manager*).

Upravljanje projektom

- Upravljanje projektom se ustvari svodi na upravljanje (*trad-off*) trima osnovnim parametrima projekta:
  1. Veličina projekta (tj. njegova funkcionalnost)
  2. Vremenski rok unutar kojeg sustav treba biti isporučen
  3. Cijena sustava
- Ova tri parametra su međusobno povezana i ako se promjeni jedan mijenjaju se i ostali parametri. Npr. treba dodati još jedan izvještaj – mijenja se funkcionalnost sustava (tj. njegova veličina) → isporuka sustava se odgađa za mjesec dan ili → povećava se cijena jer programeri trebaju raditi prekovremeno.

Upravljanje projektom



Upravljanje projektom

- Upravljanje projektom se radi kroz 4 ključna koraka:
  1. Procjena veličine projekta (*project estimation*)
  2. Stvaranje i održavanje plana rada (*workplan*) ili radnog plana
  3. Odabir osoba koje će raditi na projektu (*staffing project*)
  4. Koordiniranje aktivnosti vezanih uz projekt

PROCJENA VELIČINE PROJEKTA

Procjena veličine projekta

- U fazi planiranja se još ne znaju svi detalji projekta, zato i govorimo o procjeni veličine projekta.
- Upravljanje projektom traje tijekom cijelog SDLC-a pa se tako i početne procjene usklađuju sa novim informacijama kako se projekt razvija.
- Česta pogreška inicijalne procjene veličine projekta je preoptimistična procjena (1984 Microsoft je planirao razvoj Word-a u 1 godini, razvoj je na kraju trajao 5 godina).

Procjena veličine projekta

- Procjena veličine projekta odnosi se na procjenu vremena i truda uloženih u projekt tzv. *person-months* ili *effort* (10 mjeseci, 5 ljudi ili 6 mjeseci, 8 ljudi,…).
- Procjena veličine projekta je u početku samo okvirna, no s vremenom postaje sve specifičnija.
- Da bismo procijenili koliko je vremena potrebno za razvoj projekta koristimo najviše prijašnje iskustvo (konzultantske kompanije koje su radile slične projekte, slične prijašnje projekte u vlastitoj kompaniji koji su razvijani sa istom metodologijom, sličnim funkcionalnostima,…)

Procjena veličine projekta

- Dvije najčešće korištene formalne metode za procjenu veličine projekta su:
  1. Pristup orijentiran na planiranje (*Planning Phase Approach*)
  2. Funkcijski pristup (*Function Point Approach*)

Pristup orijentiran na planiranje

- Procjena vremena potrebnog za izradu projekta može se napraviti na osnovu vremena potrošenog za planiranje.
- Ukupno vrijeme može se izračunati korištenjem industrijskog standarda po kojem se uobičajeno 15% vremena koristi za planiranje sustava, 20% za analizu, 35% za dizajn i 30% za implementaciju.
- Glavni nedostatak ovog pristupa je da previše pojednostavljuje procjenu projekta te da izjednačava projekte (uzima se da je za sve projekte planiranje 15% projekta što dosta odstupa ovisno o složenosti projekta).

Pristup orijentiran na planiranje

	Planning	Analysis	Design	Implementation
Typical industry standards for business applications	15%	20%	35%	30%
Estimates based on actual figures for first stages of SDLC	Actual: 4 person-months	Estimated: 5.33 person-months	Estimated: 9.33 person-months	Estimated: 8 person-months

SDLC = systems development life cycle.

Pristup orijentiran na planiranje

- Vrijeme potrebno za fazu planiranja dijeli se sa određenim industrijskim postotkom.
- Na ovaj način možemo izračunati vrijeme potrebno za razvoj ostalih faza (analiza, dizajn, implementacija).
- Ako nam za planiranje projekta trebaju 4 *person-months*, za razvoj čitavog projekta trebati će nam 22.66 *person-months*.

$$\frac{4}{15\%} = \frac{4}{\frac{15}{100}} = \frac{4}{0.15} = 22.66$$

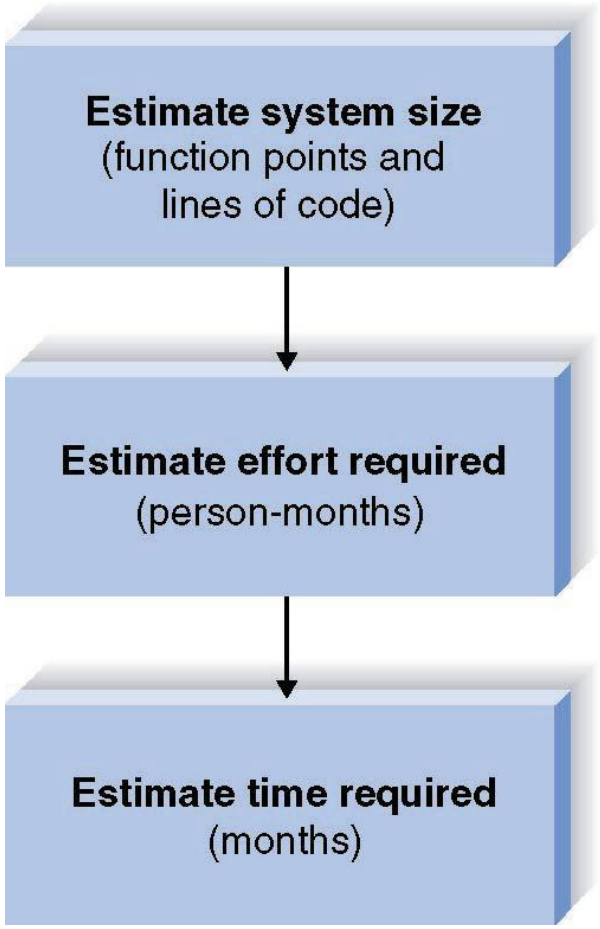
Funkcijski pristup

- Drugi način za procjenu potrebnog vremena za projekt je pristup funkcijske točke (*function point approach*) ili funkcijski pristup.
- Pristup je razvio 1979 Allen Albrecht iz IBM-a.
- Menadžer projekta procjenjuje veličinu projekta kroz broj linija kôda (LOC) koje bi projekt trebao sadržavati.
- Iz tog podatka se dobiva potrebni trud kao broj zaposlenih po broju mjeseci (*person-months*).
- Odatle se dobiva vremenski plan (*time schedule*) kao broj mjeseci od početka do kraja projekta.

Funkcijski pristup

- http://en.wikipedia.org/wiki/Function_Point_Analysis
- http://conferences.embarcadero.com/article/32094
- http://www.isbsg.org/

Funkcijski pristup



- Funkcijski pristup se sastoji od tri koraka:

1. Procjena veličine sustava
2. Procjena potrebne količine truda (*effort estimation)*
3. Procjena potrebnog vremena

Funkcijski pristup

- Procjena veličine projekta kod funkcijskog pristupa se temelji na tzv. funkcijskoj točki.
- Funkcijska točka je mjera veličine programa koja se proračunava prema broju i kompleksnosti komponenti sustava.
- Komponente sustava kod funkcijskog pristupa su:
  - ulazi (*input*) (npr. forma za unos podataka)
  - izlazi (*output*) (npr. izvještaji koji se printaju)
  - upiti (*query*) (prema bazama podataka)
  - programska logika
  - programska sučelja (npr. komponenta koja iz baze druge aplikacije importira podatke u promatranu aplikaciju)
- Voditelj projekta treba procijeniti broj komponenti i njihovu složenost.

Radni list

Komponente se prikazuju na radnom listu (*worksheet*).

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi					
Izlazi					
Upiti					
Programska logika					
Programska sučelja					
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					

Radni list

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi		3	2	1	
Izlazi		4	10	5	
Upiti		7	0	3	
Programska logika		0	15	0	
Programska sučelja		1	0	2	
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					

Radni list

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi	6	3	2	1	
Izlazi	19	4	10	5	
Upiti	10	7	0	3	
Programska logika	15	0	15	0	
Programska sučelja	3	1	0	2	
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					

Radni list

Težinski faktor složenosti također procjenjuje voditelj projekta

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi	**6**	3 x 3	2 x 4	1 x 6	
Izlazi	**19**	4 x 4	10 x 5	5 x 7	
Upiti	**10**	7 x 3	0 x 4	3 x 6	
Programska logika	**15**	0 x 7	15 x 10	0 x 15	
Programska sučelja	**3**	1 x 5	0 x 7	2 x 10	
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					

Radni list

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi	6	3 x 3	2 x 4	1 x 6	23
Izlazi	19	4 x 4	10 x 5	5 x 7	101
Upiti	10	7 x 3	0 x 4	3 x 6	39
Programska logika	15	0 x 7	15 x 10	0 x 15	150
Programska sučelja	3	1 x 5	0 x 7	2 x 10	25
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					

Radni list

		Kompleksnost			
Opis	Ukupan broj	Niska	Srednja	Visoka	Ukupno
Ulazi	6	3 x 3	2 x 4	1 x 6	23
Izlazi	19	4 x 4	10 x 5	5 x 7	101
Upiti	10	7 x 3	0 x 4	3 x 6	39
Programska logika	15	0 x 7	15 x 10	0 x 15	150
Programska sučelja	3	1 x 5	0 x 7	2 x 10	25
Ukupan broj neprilagođenih funkcijskih točaka Total Unadjusted Function Points (TUFP):					338

Funkcijski pristup

- Na primjeru s prethodnog slajda sustav ima 6 ulaznih komponenti, 19 izlaznih, 10 upita, 15 datoteka programske logike i 3 programska sučelja.
- Menadžer projekta također treba procijeniti složenost komponenti tj. koje su komponente jednostavne (*low complexity*), složenije (*medium complexity*) i najsloženije (*high complexity*) kao i težinski faktor složenosti.
- Iz svega toga se računa ukupan broj neprilagođenih funkcijskih točaka - TUFP (*total unadjusted function points*).

Funkcijski pristup

- Na kompleksnost sustava utječu i dodatni faktori poput zahtjeva za performanse sustava, distribuiranost sustava i sl.
- Da bi dobili *realnu procjenu* veličine sustava, i ovi se faktori moraju uzeti u obzir.
- Zato se u proračun uvodi i faktor kompleksnosti procesiranja - PC (Total Processing Complexity).
- Faktor uključuje dodatne procjene voditelja projekta o svojstvima projekta koja utiču na njegovu složenost jer je složenost cjelokupnog projekta veća od samog zbroja složenosti njegovih dijelova.

Funkcijski pristup

Data Communications	3
Heavy use configuration	0
Transaction rate	0
End-user efficiency	0
Complex processing	0
Installation ease	0
Multiple sites	0
Performance	0
Distributed functions	2
Online data entry	2
Reusability	0
Operational ease	0
Extensibility	0
Total Processing Complexity (PC):	**7**

Funkcijski pristup

- Iz ukupnog broja neprilagođenih funkcijskih točaka (TUFP) i faktora kompleksnosti procesiranja (PC) proračunavaju se faktor prilagođene kompleksnosti programa - APC (Adjusted Program Complexity), a zatim ukupan broj prilagođenih funkcijskih točaka – TAFP (Total Adjusted Function Points) prema sljedećim formulama:

$$APC = 0.65 + \frac{PC}{100}$$

$$TAFP = APC \cdot TUFP$$

Funkcijski pristup

- Primjer:

Za ukupan broj neprilagođenih funkcijskih točaka TUFP = 338 i kompleksnost procesiranja PC = 7

Faktor prilagođene kompleksnosti programa APC

$$APC = 0.65 + \frac{7}{100} = 0.72$$

Ukupan broj prilagođenih funkcijskih točaka TAFP

$$TAFP = 0.72 \cdot 338 = 243$$

Funkcijski pristup

- Faktor prilagođene kompleksnosti programa (APC) uvijek ima temeljnu vrijednosti 0.65 na koju se dodaje faktor kompleksnosti procesiranja (PC) podijeljen sa 100.
- Kako je dosta složeno odrediti APC često se koristi i njegova procjena pa se za jednostavne projekte uzima da ja APC 0.65, za srednje složene projekte APC 1, za složene projekt uzima se da je APC 1.35.

Funkcijski pristup

- Sljedeći korak u funkcijskom pristupu je konvertiranje ukupnog broja prilagođenih funkcijskih točaka u broj linija kôda (LOC).
- Konvertiranje ovisi o programskom jeziku u kojem se aplikacija programira. To znači i da veličina projekta ovisi o izabranom programskom jeziku.
- Podaci potrebni za konvertiranje ukupnog broja prilagođenih funkcijskih točaka u broj linija kôda su komercijalni (http://www.spr-global.com/catalog/)

Funkcijski pristup

Language	Approximate Number of Lines of Code per Function Point
C	130
COBOL	110
Java	55
C++	50
Turbo Pascal	50
Visual Basic	30
PowerBuilder	15
HTML	15
Packages (e.g., Access, Excel)	10–40

Source: Capers Jones, Software Productivity Research, http://www.spr.com

Funkcijski pristup

U našem primjeru broj ukupnih funkcijskih točaka je 243.

Programski jezik	TAFP x broj linija koda po 1 funkcijskoj točci	Broj linija koda
C	243 x 130	**31590**
JAVA	243 x 55	**13365**
C++	243 x 50	**12150**
VISUAL BASIC	243 x 30	**7290**
HTML	243 x 15	**3645**

Funkcijski pristup

- Ovakav pristup računanja veličine projekta ima svoje nedostatke.
- Ne daje uvijek pouzdane rezultate, jer na samom početku planiranja sustava nije moguće odmah znati koliko će sustav imati ulaza, izlaza,...
- Ali je bolje nego ništa.

Procjena truda

- Najčešće se računa po COCOMO (*COnstructive COst MOdel*) modelu koji je razvio Barry Boehm 1981, a zadnja verzija modela COCOMO II je definirana 2000 (http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html).
- Različite verzije COCOMO modela se primjenjuju za različite veličine projekata, tipove projekata (operacijski sustav ili bankarski IS). Navedena verzija se primjenjuje za jednostavne ili srednje složene projekte, do 100 000 linija koda i do 10 programera.

$$TRUD = 1.4 \cdot \frac{broj\_linija\_koda}{1000}$$ [broj osoba / broj mjeseci]

Procjena truda

- Za primjer za koji smo računali ukupan broj funkcijskih točaka, izračunati ćemo i TRUD.
- Pretpostaviti ćemo da programiramo u C++.

$$TRUD = 1.4 \cdot \frac{12150}{1000} = 1.4 \cdot 12.15 = 17.01$$ [broj osoba / broj mjeseci]

Procjena truda

- Ukoliko je projekt potrebno završiti za npr. 5 mjeseci, znači da u timu moramo imati 4 osobe.

$$TRUD = 1.4 \cdot \frac{12150}{1000} = 1.4 \cdot 12.15 = 17.01$$

[broj osoba / broj mjeseci]

$$17.01 \div 5 = 3.402$$

Treba se zaokružiti na veći cijeli broj, inače će projekt kasniti

$$4 \cdot 5 = 20 > 17.01$$

Broj članova tima

Broj mjeseci

Rezultat mora biti veći od TRUDA

Procjena truda

- Preporučeno trajanje projekta računa se po sljedećoj formuli:

$$VRIJEME = 3.0 \cdot TRUD^{\frac{1}{3}}$$

- Umjesto faktora 3.0 može se koristiti 2.5 ili 3.5.
- Npr. ako je TRUD = 17.01, onda:

$$VRIJEME = 3 \cdot 17.01^{\frac{1}{3}} = 3 \cdot 2.5717 = 7.7153 \quad \text{mjeseci}$$

- Ova procjena se odnosi na fazu analize, dizajna i implementacije. Faza planiranja ne ulazi u ovih 8 mjeseci.

STVARANJE I ODRŽAVANJE PLANA RADA

Radni plan

- Radni plan je dinamički raspored svih zadataka koji se trebaju izvršiti tijekom trajanja projekta.
- Sadrži informacije o svakom zadatku: duljinu trajanja, vrijeme početka i završetka, tko izvodi zadatak, što je rezultat pojedinog zadatka i sl. informacije.
- Radni plan se može mijenjati tijekom SDLC jer se obično napredovanjem projekta povećava broj detalja identificiranih zadataka tj. sve se više zna o svakom zadatku.

Radni plan

- Za izradu radnog plana potrebno je identificirati zadatke i odrediti duljinu trajanja pojedinog zadatka.
- Zadaci se grafički prikazuju korištenjem Gantt i PERT dijagrama.
- Postoji cijeli niz CASE alata za upravljanje projektom koji obično sadrže funkcionalnost generiranja i održavanja radnog plana.

Dijagrami

- Gantt (Henry Laurence Gantt (1861-1919) - inženjer) dijagram je koristan za nadgledanje statusa projekta u bilo kojem trenutku trajanja projekta.
- PERT (*Program Evaluation and Review Technique*) dijagram bolje prikazuje ovisnosti zadataka (npr. zadatak 4 ne može započeti dok se ne dovrši zadatak 2) i kritične putove (*critical paths*).

Identificiranje zadataka

- Identificiranje zadataka radi se obično preko:

1. postojećih metodologija
   - Formalne metodologije imaju definirane korake i rezultate koji se generiraju u svakom koraku koji se dodaju u radni plan. Najčešće korištena metoda.
2. strukturiranog top-down pristup
   - Kod top-down pristupa ne postoje već definirani koraci ili zadaci, kreće se od početka. Naziv top-down je zbog toga što se kreće od vrha projekta tj. identificiraju se zadaci visoke razine koji se onda dijele na jednostavnije podzadatke (*subtasks*). Zadaci su popisani hijerarhijski. Takva struktura naziva se razložena struktura posla (*work breakdown structure).*

Strukturirani top-down pristup

- Razložena struktura posla (*work breakdown*) može biti organizirana na dva načina:
  1. Po SDLC fazama
  2. Po produktu

Organizacija po SDLC fazama

- Razložena struktura posla po SDLC fazama na najvišoj razini bi se dijelila na planiranje, analizu, dizajn, implementaciju.
- Zadatak u fazi planiranja bi mogao biti analiza izvedivosti (*feasibility analysis*) sustava.
- Analiza izvedivosti bi se podijelila na:
  - Tehničku analizu
  - Ekonomsku analizu
  - Organizacijsku analizu
- Svaki od ovih zadataka bi se podijelio na još jednostavnije itd.

Organizacija po produktu

- Razložena struktura posla se organizira po tipovima različitih produkata koji se trebaju razviti za sustav.
- Primjer za razvoj web stranice:
  - Apleti
  - Predlošci korisničkog sučelja
  - Modul za rad sa bazom podataka
  - itd.

Radni plan

- Radni plan omogućava voditelju projekta upravljanje projektom. Na osnovu radnog plana voditelj vidi da li projekt kasni, treba li nešto mijenjati da bi se projekt realizirao na vrijeme (npr. na nekom pojedinom zadatku zaposliti još ljudi i sl.)...
- Radni plan je ustvari tablica sa svim informacijama koje radni plan sadrži (duljina trajanja svakog zadatka, status zadatka (*npr. završen, otvoren*), o čemu zadatak ovisi (*npr. o nekom drugom zadatku*), važne datume (*npr. rokove*), itd.).

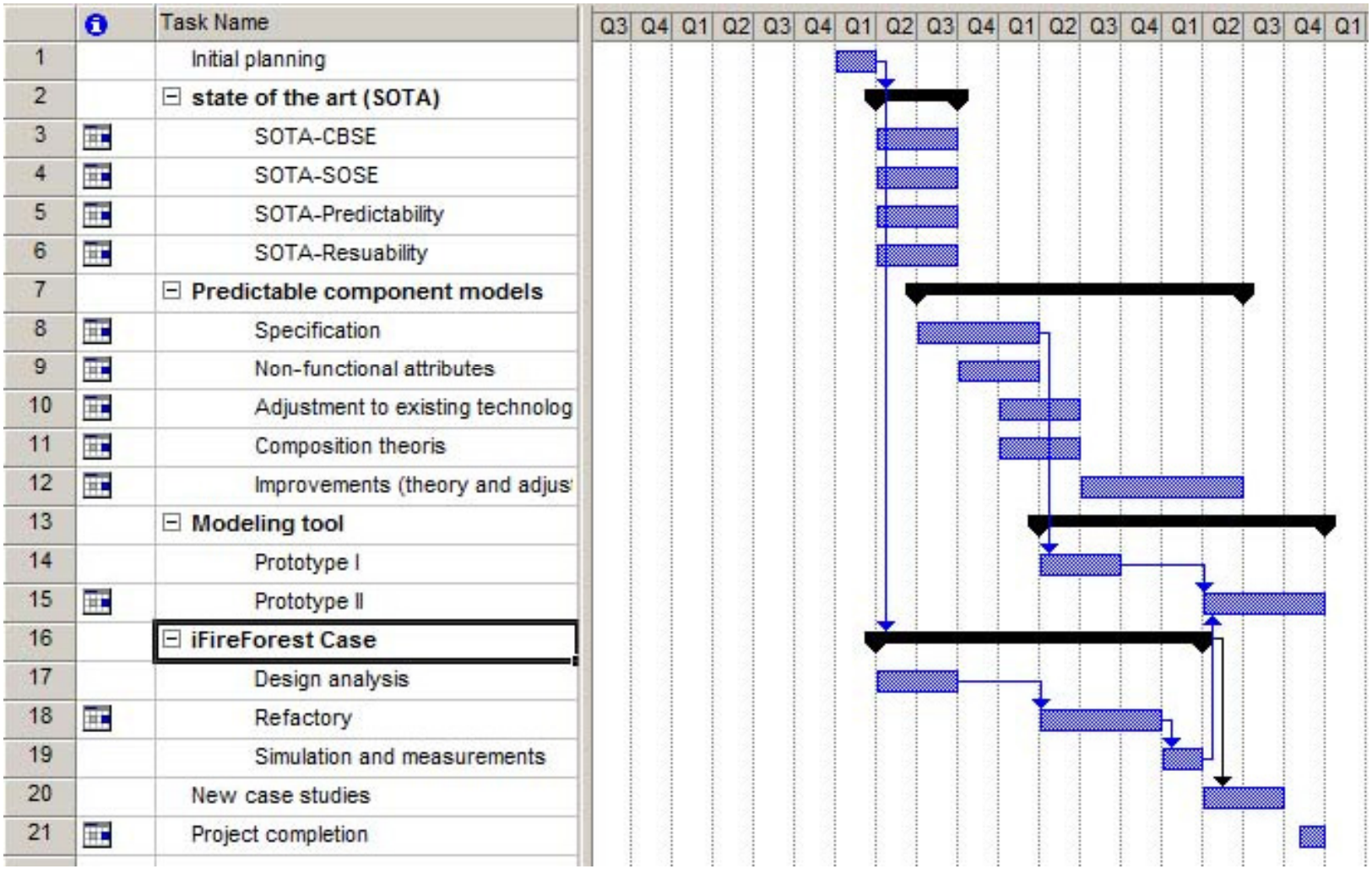
Gantt dijagrami

- Gantt dijagram (ili gantogram) je grafički horizontalno stupčasti prikaz radnog plana.
- Zadaci se navode kao redci u dijagramu, dok je vremenska linija smještena na vrhu dijagrama.
- Vremenski kratki projekt može se podijeliti na dane ili čak sate, dok se dulji projekt dijeli na tjedne ili mjesece.
- Svaki redak predstavlja jedan zadatak. Početak i kraj iscrtavanja retka su određeni vremenskom linijom.
- Obično je broj redaka ograničen na 20-30 zbog preglednosti. Ako ima više od 30 zadataka onda se i gantogram može razložiti na više dijelova.

Gantt dijagrami

- Kako se zadatak izvršava redak se ispunjava tako da se u svakom trenutku vidi gdje je došlo izvršavanje pojedinog zadatka.
- Voditelj projekta može iz gantograma vidjeti koliko su dugi pojedini zadaci, dokle se došlo sa realizacijom, ali i koji su zadaci slijedni (izvršavaju se jedan nakon drugog), koji se zadaci izvršavaju istovremeno, te koji su zadaci međusobno ovisni.
- Projektni rokovi (*milestone*) se označavaju trokutom ili rombom.
- Zavisnost između zadataka se označava sa strelicom koja povezuje međusobno ovisne zadatke.

Gantt dijagrami



PERT dijagrami

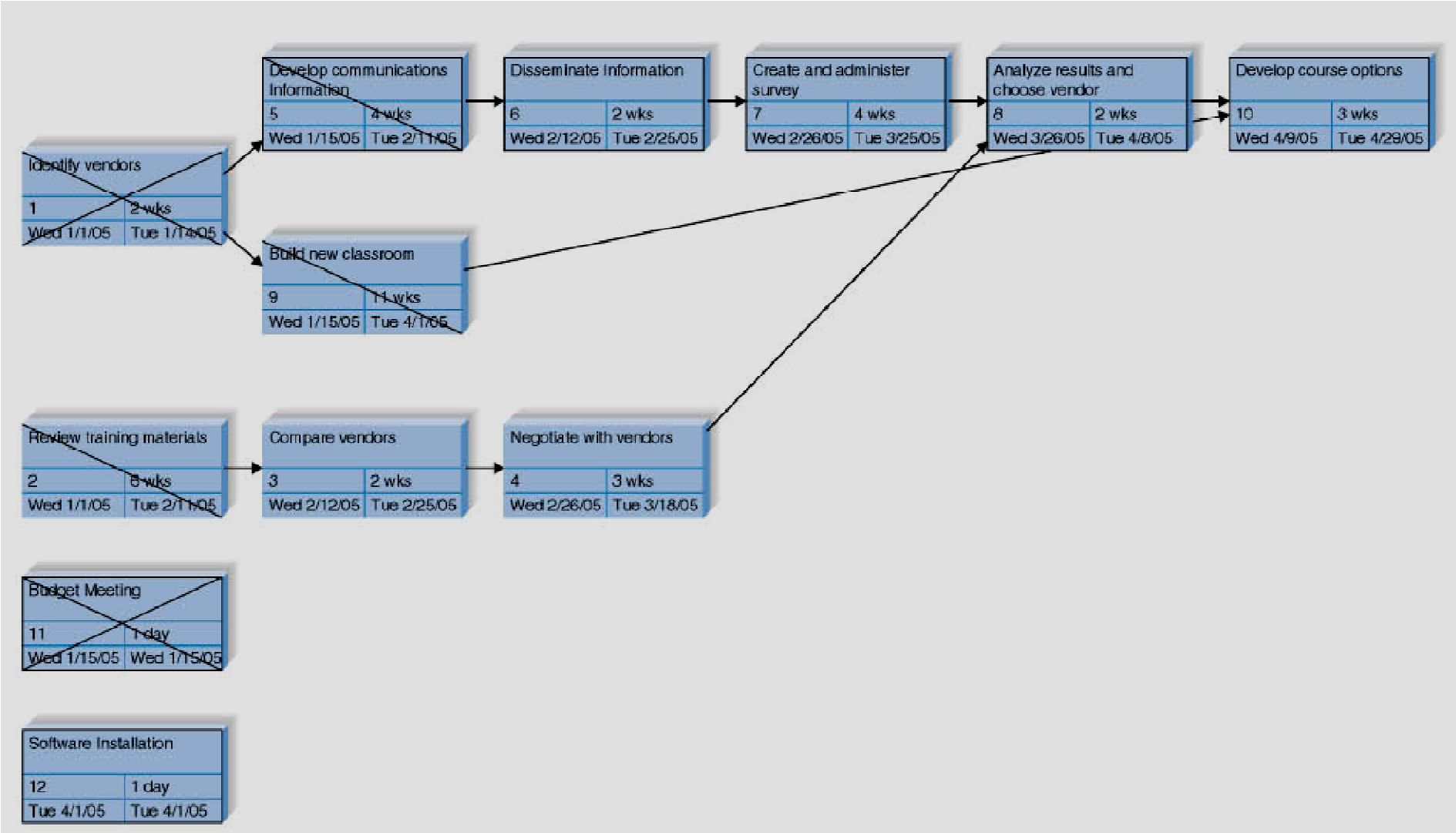
- PERT (*Program Evaluation and Review Technique*) grafički prikaz radnog plana prikazuje projektne zadatke kroz dijagram toka.
- Zadaci se prikazuju kao čvorovi grafa, dok linije grafa predstavljaju ovisnost pojedinih zadataka.
- Može se koristiti kada su procjene vremena trajanja pojedinog zadatka nesigurne.

PERT dijagrami

- Procjena trajanja zadatka u PERT dijagramima se radi kroz procjenu vrijemena trajanja razvoja kroz:
  - Najbolje vrijeme trajanja
  - Najvjerojatnije vrijeme trajanja
  - Najgore vrijeme trajanja
- Ta vremena se zatim kombiniraju korištenjem sljedeće formule:

$$PERT\_prosjek = \frac{najbolje + 4 \cdot najvjerojatnije + najgore}{6}$$

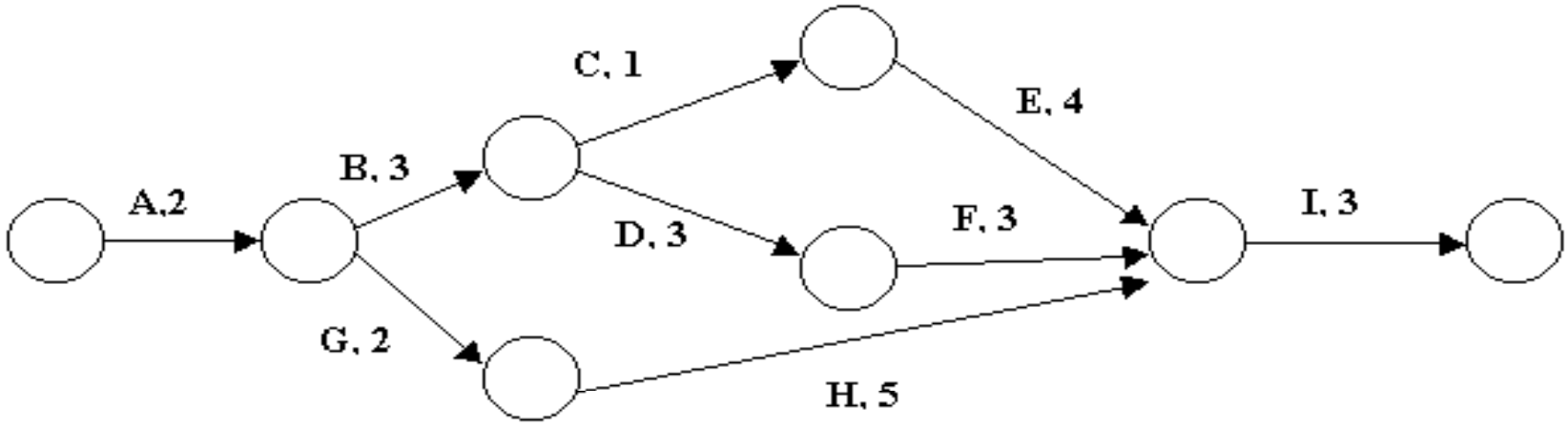
PERT dijagrami



PERT dijagrami

- Metoda kritičnog puta (*Critical Path Method*) identificira kritični put u grafu.
- To je najdulji put u PERT dijagramu od početka do kraja projekta.
- Pokazuje koji se zadaci moraju izvršiti na vrijeme da bi čitav sustav bio gotov na vrijeme.

PERT dijagrami



- Putovi izvršavanja projekta su:
- A,B,C,E,I = 2+3+1+4+3 = 13 dana
- A,B,D,F,I = 2+3+3+3+3 = 14 dana
- A,G,H,I = 2+2+5+3 = 12 dana

Usporedba dijagrama

- Gantogram je bolji za prikaz utroška vremena po zadatku.
- PERT je bolji za prikaz ovisnosti između zadataka.
- Obično se u vođenju projekta koriste oba dijagrama.

ODABIR OSOBA KOJE ĆE RADITI NA PROJEKTU

Odabir članova tima

$$TRUD = 1.4 \cdot \frac{broj\_linija\_koda}{1000}$$

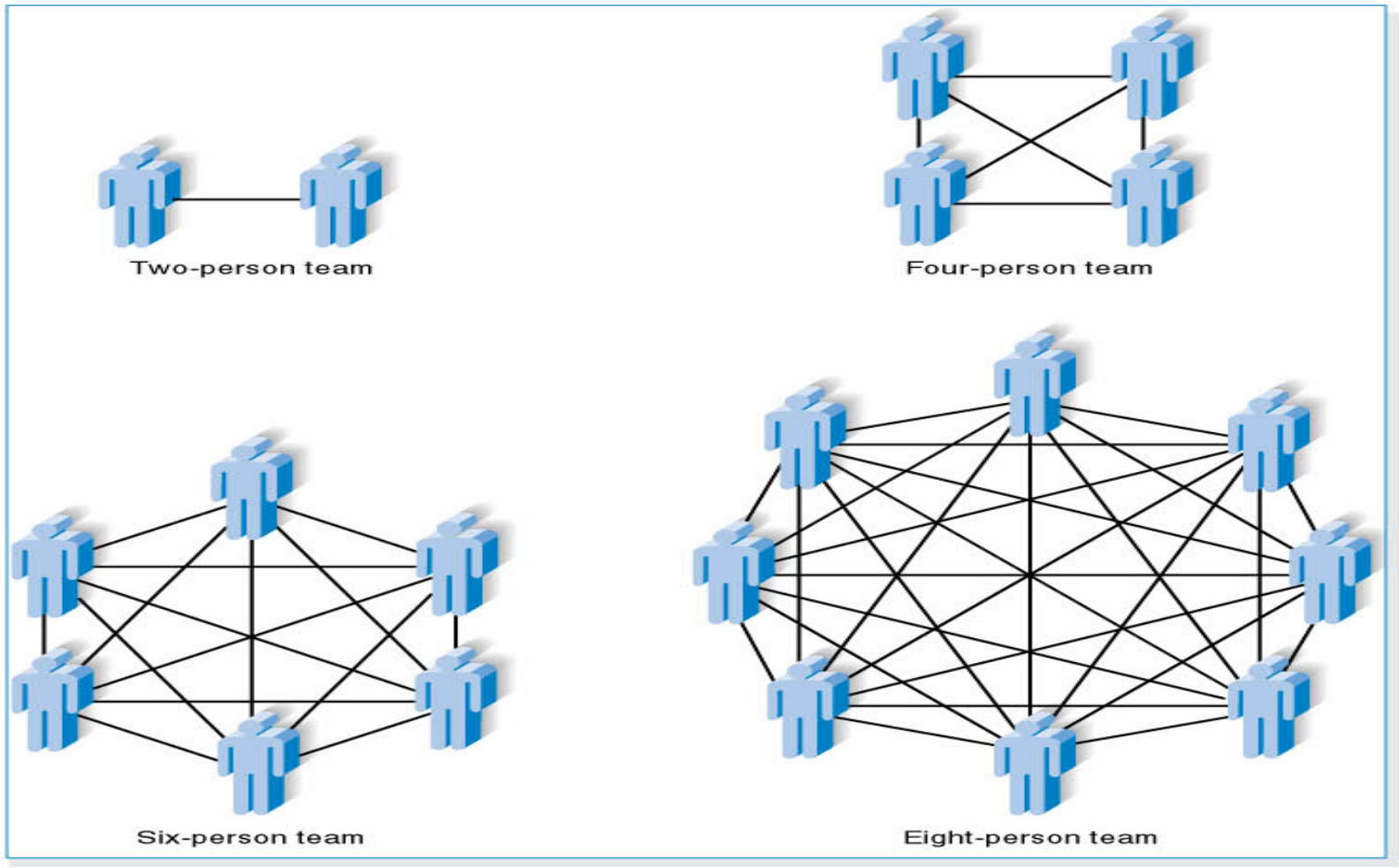
[broj osoba / broj mjeseci]

- Pretpostavimo da je TRUD = 40 i da projekt mora biti gotov za 10 mjeseci.
- Kako odrediti koliko nam treba osoba u timu? Na sljedeći način:
- TRUD / broj_mjeseci = 40 / 10 = 4

Odabir članova tima

- Više ljudi u timu ne znači da će projekt biti brže gotov. Naime više članova tima ne garantira bržu isporuku jer je teže upravljati većim timom.
- Preporučena veličina tima je 8-10 članova tima.
- Ako je potrebno više od 10 ljudi, tim razbijamo u podtimove, od kojih svaki podtim ima svog voditelja.
- Projekt menadžer uglavnom komunicira samo sa tim voditeljima.

Problemi kod velikih timova – otežana komunikacija i koordinacija



KOORDINIRANJE AKTIVNOSTI VEZANIH UZ PROJEKT

Koordiniranje projektnih aktivnosti

- Koordiniranje projektnih aktivnosti treba osigurati uspješno dovršenje projekta, a uključuje cijeli niz aktivnosti.
- Menadžer projekta treba tijekom čitavog SDLC pratiti što se dešava s projektom, da li se projektni zadaci izvršavaju na vrijeme, da li je potrebno ponovo procijeniti određene parametre sustava (npr. složenost i veličina sustava mogu se povećati s novim korisničkim zahtjevima koji se mogu javiti prilikom dizajna sustava).

Koordiniranje projektnih aktivnosti

- Koordiniranje projektnih aktivnosti uključuje:
  1. upravljanje promjenama u specifikaciji zahtjeva
  2. postavljanje različitih standarda (npr. formalna pravila za imenovanje datoteka, podnošenje izvještaja u zadanoj formi, ….)
  3. izradu i vođenje dokumentacije tijekom čitavog projekta, a ne na kraju!!!
  4. upravljanje rizicima (*risk managment*)
  5. upravljanje ljudskim resursima

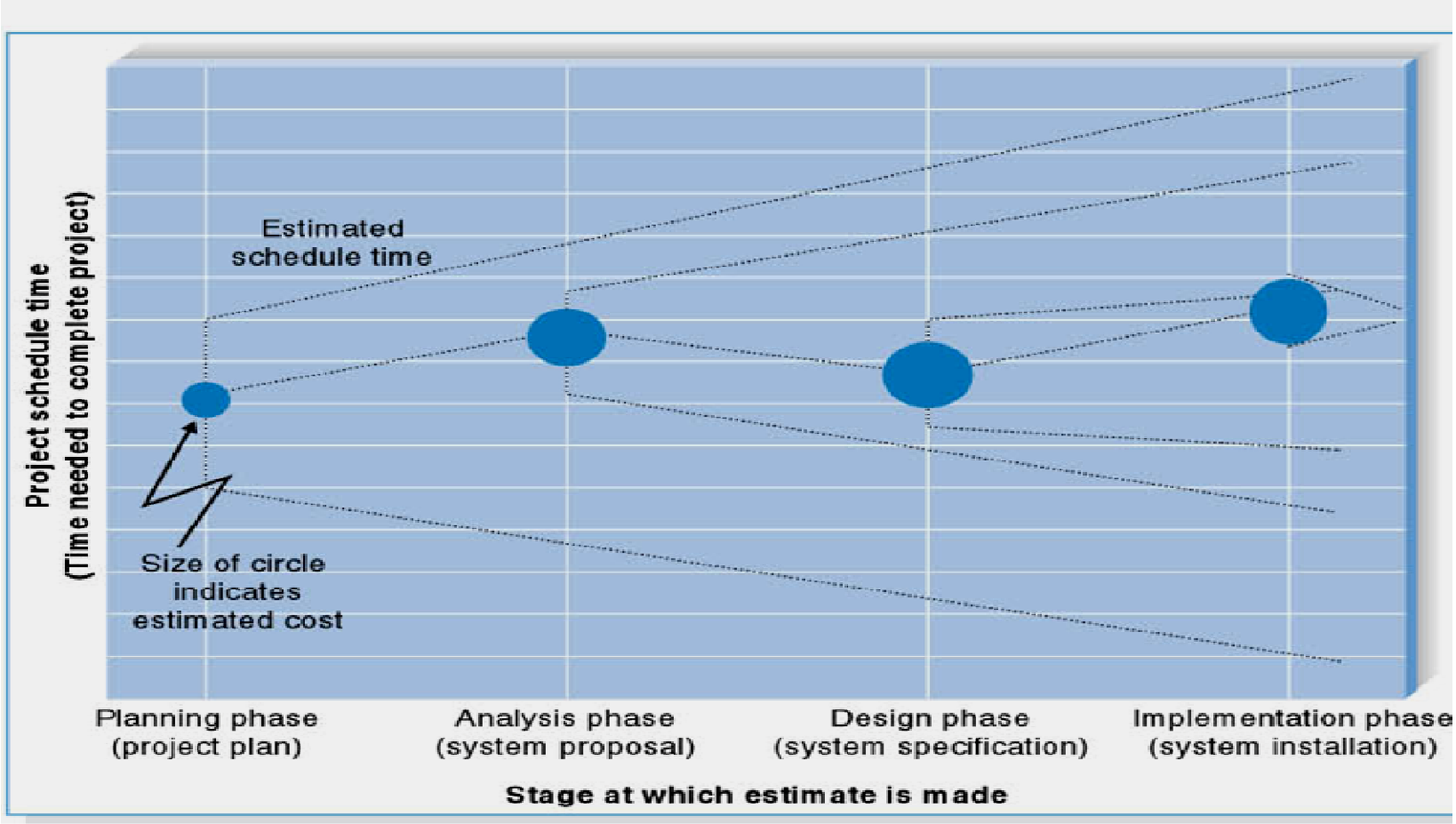
Koordiniranje projektnih aktivnosti

- Promjene u projektu su posljedica:
  1. Grešaka prilikom planiranja projekta
  2. Novih zahtjeva koji se postavljaju nakon što je projekt počeo (*scope creep*)

Model uragana (Hurricane Model)

- Procjene koje su obavljene u fazi planiranja sustava (cijena projekta, vrijeme...) moraju se ponovo obavljati u tijekom razvoja projekta.
- Procjenjivanje cijene i vremena slijedi *model uragana*, tj. procjene postaju sve točnije kako rad na projektu napreduje.

Model uragana



Model uragana

- Cijena projekta predviđena projektnim planom u stvarnosti može biti i 100% skuplja i projekt može kasniti za otprilike 25% vremena predviđenog za njegov razvoj prema Boehmu (Costs Models for Future Software Life Cycle Processes: COCOMO 2.0, Annals of Software Engineering).
- To su margine greški (*margin of error*) procjene projekta.
- Ako smo procijenili da će projekt koštati 100 000 dolara i da će trajati 20 tjedana, on u stvarnosti može koštati i 200 000 dolara i trajati i 25 tjedana.
- Ove se margine grešaka odnose samo na dobro obavljene projektne planove. Za one slabije obavljene ove su margine znatno veće.

Upravljanje promjenama u specifikaciji zahtjeva

- Najčešći razlog kašnjenja projekta ili povećanja cijene su mijenjanje zahtjeva nakon što se sustav već počne razvijati - *scope creep*.
- Do mijenjanja zahtjeva može doći zbog toga što korisnici uviđaju mogućnosti novog sustava, što direktori žele uvesti neke nove funkcionalnosti, ….
- Projekt menadžer u pravilu mora odobriti sve promjene u specifikaciji zahtjeva.
- Ako neku promjenu nije moguće trenutno prihvatiti, implementira se u idućoj verziji sustava (ako se radi sa nekom inkrementalnom metodologijom razvoja sustava).

Timeboxing

- Timeboxing je tehnika koja se koristi kod upravljanja dosegom projekta (*scope management*).
- Najčešće se upotrebljava kod ubrzanog razvoja aplikacija (RAD – Rapid Application Development).
- Postavlja se fiksni datum isporuke sustava i sustav se isporučuje do tog datuma i ako nije potpuno funkcionalan.
- Najprije se implementiraju funkcije sa najvišim prioritetom i one se isporuče korisniku.

Timeboxing

- Koraci u timeboxingu:
  1. Postavite rok do kojeg sustav mora biti gotov.
     - Rok ne bi trebao biti nemoguć.
     - Trebao bi ga postaviti razvojni tim.
  2. Poredajte funkcije sustava po važnosti.
  3. Isprogramirajte “jezgru sustava”, odnosno najvažnije funkcije.
  4. Funkcije koje se ne mogu isprogramirati unutar zadanog roka odgodite za sljedeću verziju sustava.
  5. Dostavite sustav korisniku.
  6. Ponavljajte korake od 3 do 5, da bi implementirali nove zahtjeve i poboljšanja.

Upravljanje ljudskim resursima

- Ekipiranje tima nije kraj upravljanja ljudskim resursima (*human resources management*). Motivacija je glavni način utjecaja na radni učinak.
- Oprezno koristite novčane nagrade.
- Efikasnije su sljedeće nagrade:
  - Priznanje
  - Osjećaj da su nešto postigli
  - Sam rad
  - Odgovornost
  - Napredovanje
  - Stjecanje novih znanja

Upravljanje ljudskim resursima

- Kohezija tima više doprinosi produktivnosti tima no pojedinačne sposobnosti.
- Jasno definiranje položaja, uloge i zadataka članova tima smanjuje mogućnost konflikta.
- Postaviti neke norme i pravila ponašanja (npr. sastanci su svaki petak u 14:00).
- ...

"I'M LOOKING FOR SOMEONE WHO CAN TAKE ON A GREAT DEAL OF RESPONSIBILITY AND MAKE EXECUTIVE DECISIONS AND BE WILLING TO START AT THE MINIMUM WAGE."

Projektni standardi

- Standardi osiguravaju da svi članovi tima rade na jednak način i koriste iste procedure. (npr. nova verzija korisničkog dokumenta se treba prebaciti na server u direktorij xxx, treba je pogledati voditelj projekta i odobriti).
- Standardi definiraju konvencije imenovanja varijabli u kodu, forme testiranja aplikacije, …

Projektni standardi

- Postavljeni standardi olakšavaju komunikaciju među članovima tima i vođenje projekta, a obično se postavljaju:
  1. Standardi za dokumentaciju (npr. svaki dokument treba imati zaglavlje sa verzijom dokumenta, imenom kreatora dokumenta, datum, …)
  2. Standardi za kodiranje (npr. na svakih 10 linija koda treba doći jedna linija komentara, …)
  3. Proceduralni standardi (npr. svaki ponedjeljak u 12 prijaviti napredovanje zadatka koji se izvršava, …)
  4. Standardi programa (npr. ime programa, opis namjene programa, pseudokod, ...)
  5. Standardi za dizajn korisničkog sučelja (npr. naslovi na korisničkom sučelju trebaju biti *boldani*, …)

Dokumentacija

- Tijekom planiranja projekta već se započinje sa dokumentacijom projekta.
- Dokumentacija sadrži detaljne informacije o svim fazama razvoja sustava.
- Ne smije se pisati u zadnji čas!
- Loša dokumentacija = problemi sa održavanjem i update-om sustava!

Upravljanje rizicima

- Upravljanje rizikom (*risk management*) odnosi se na prepoznavanje i rješavanje problema do kojih može doći tijekom razvoja sustava.
- Rizik se javlja zbog promjene zahtjeva sustava, previše optimistično postavljenog projekta, lošeg dizajna itd.
- Ovaj korak bi trebao rezultirati dokumentom(ima) za procjenu rizika (*risk assessment*) koji identificira potencijalne rizike projekta, njihovu vjerojatnost pojave i utjecaj na projekt.

Upravljanje rizicima

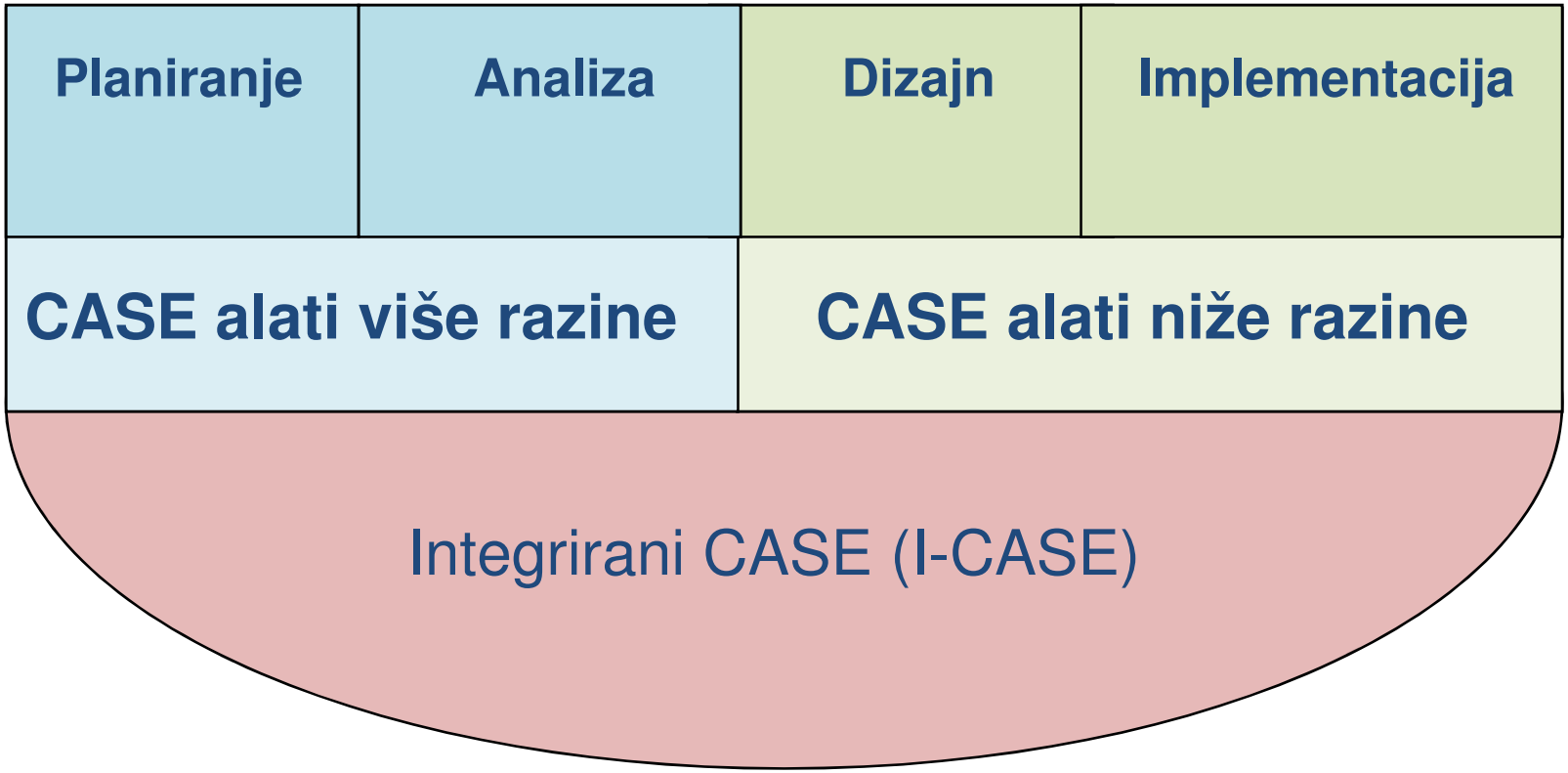
Primjer

Rizik 1	Razvoj sustava će vjerojatno biti usporen jer programeri nemaju iskustva s Java programskim jezikom.
Vjerojatnost rizika	Velika
Potencijalni utjecaj na projekt	Rizik može produžiti vrijeme završetka zadataka vezanih uz programiranje za 50%
Načini rješavanja	Zapošljavanje vanjskog suradnika sa iskustvom programiranja u Javi. Slanje programera na tečajeve…..

CASE

- Za koordiniranje projektnih aktivnosti mogu se koristiti različiti CASE alati (Oracle Designer/2000, Rational Rose, Visible Analyst Workbench, …).
- *Computer-Aided Software Engineering* (CASE) su različiti programske alati za podršku različitih aktivnosti u SDLC-u.
- CASE alati se najčešće dijele na: →

CASE


Planiranje
Analiza
Dizajn
Implementacija
CASE alati više razine
CASE alati niže razine
Integrirani CASE (I-CASE)

CASE

- Alati za upravljanje projektom uključuju različite funkcionalnosti (procjena veličine projekta, upravljanje radnim planovima, komunikacija među članovima projektnog tima, vođenje dokumentacije,...).
- Možemo ih podijeliti prema namjeni na:
  - Kolaborativni softver (collaborative)
  - Softver za praćenje pojedinih pitanja (issue tracking software)
  - Softver za vremensko upravljanje (scheduling)
  - Softver za upravljanje portfoliom (project portfolio management)
  - Upravljanje resursima (resource management)
  - Upravljanje dokumentacijom (document management)

CASE

- Kolaborativni softver (collaborative software ili groupware, workgroup support systems ili group support systems) je softver koji pomaže ljudima koji su uključeni u zajednički zadatak da postignu svoj cilj (npr. chat za dogovoriti tulum ili forum na e-learning portalu preko kojeg djelimo informacije o kolegiju).
- Naravno kolaborativni softver ima i znatno složenije primjene. Npr. Microsoft Share Point omogućava dijeljenje Visual Studio projekata i zajednički rad na njima.

CASE

- Tipični kolaborativni softver koji se koristi kod implementacije aplikacije (ali i u drugim aktivnostima) je tzv. softver za reviziju verzija ili version control systems (VCS) (npr. Microsoft Visual SourceSafe, CVS (Concurrent Versions System),...).
- To je softver koji omogućava praćenje verzija dokumenata, dodavanje novih verzija, vraćanje na stare verzije,...

CASE

- Softver za praćenje pojedinih pitanja (issue tracking software) ili (trouble ticket system, support ticket , incident ticket system) je softver preko kojeg se može kreirati i održavati lista pitanja (ili ticketa ili tema).
- Često se koristi kao podrška korisnicima u CRM (customer relationship management) sustavima, za dojavljivanje problema, bugova i sl., ali se može koristiti i za razvoj i upravljanje projektom.
- *Tiketing* sustava obično sadrži i cijeli niz informacija vezanih uz sustav.