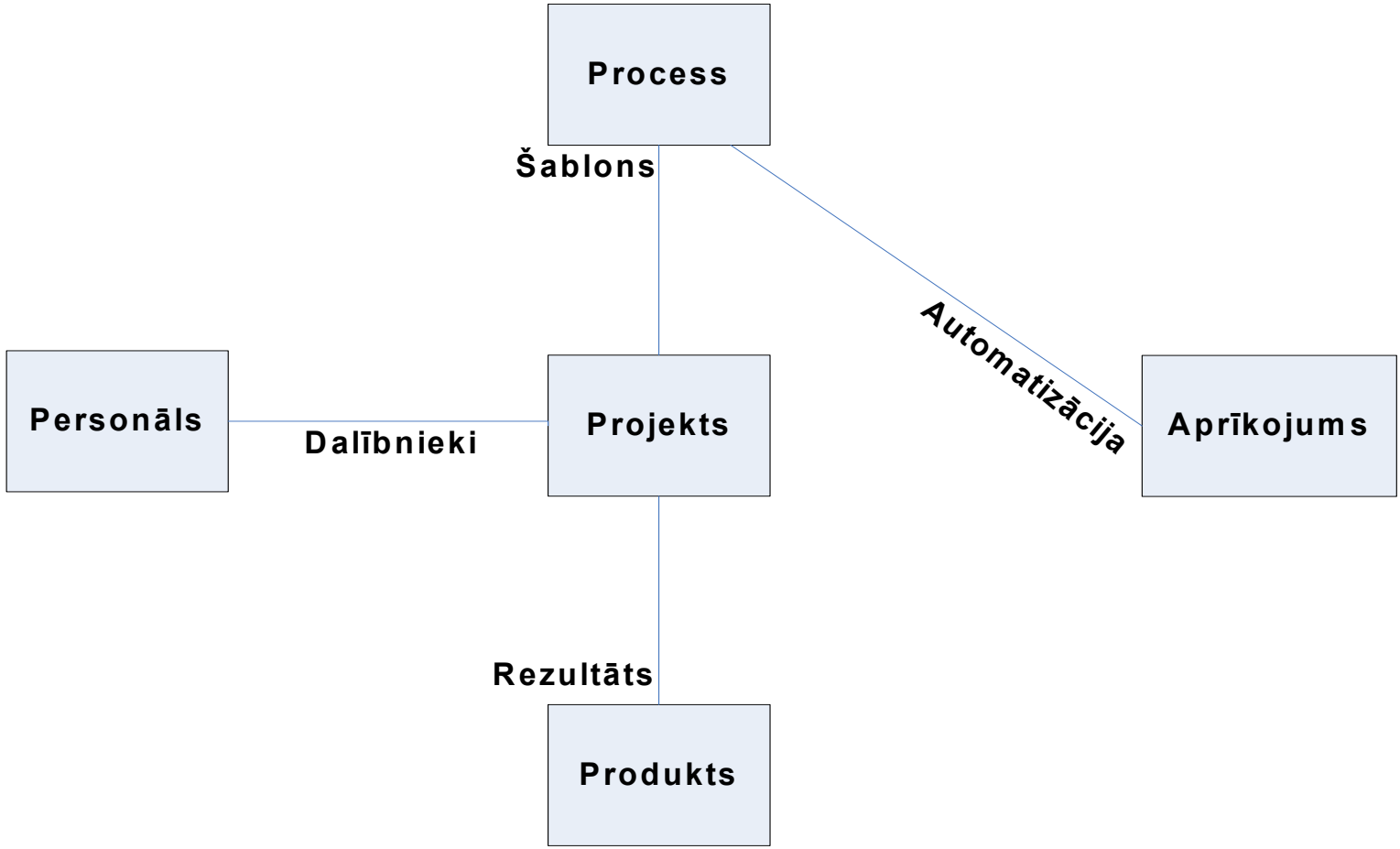
Programmatūras attīstības tehnoloģijas

Cilvēka faktors
Profesijas
Komandas organizācija

Dr.sc.ing., asoc. prof. Oksana Ņikiforova
DITF LDI
Lietišķo datorzinātņu katedra
Rīga - LV1048, Meža 1/3, 510.kab., tel.67 08 95 98
oksana.nikiforova@rtu.lv

Četri "P" programmatūras izstrādē



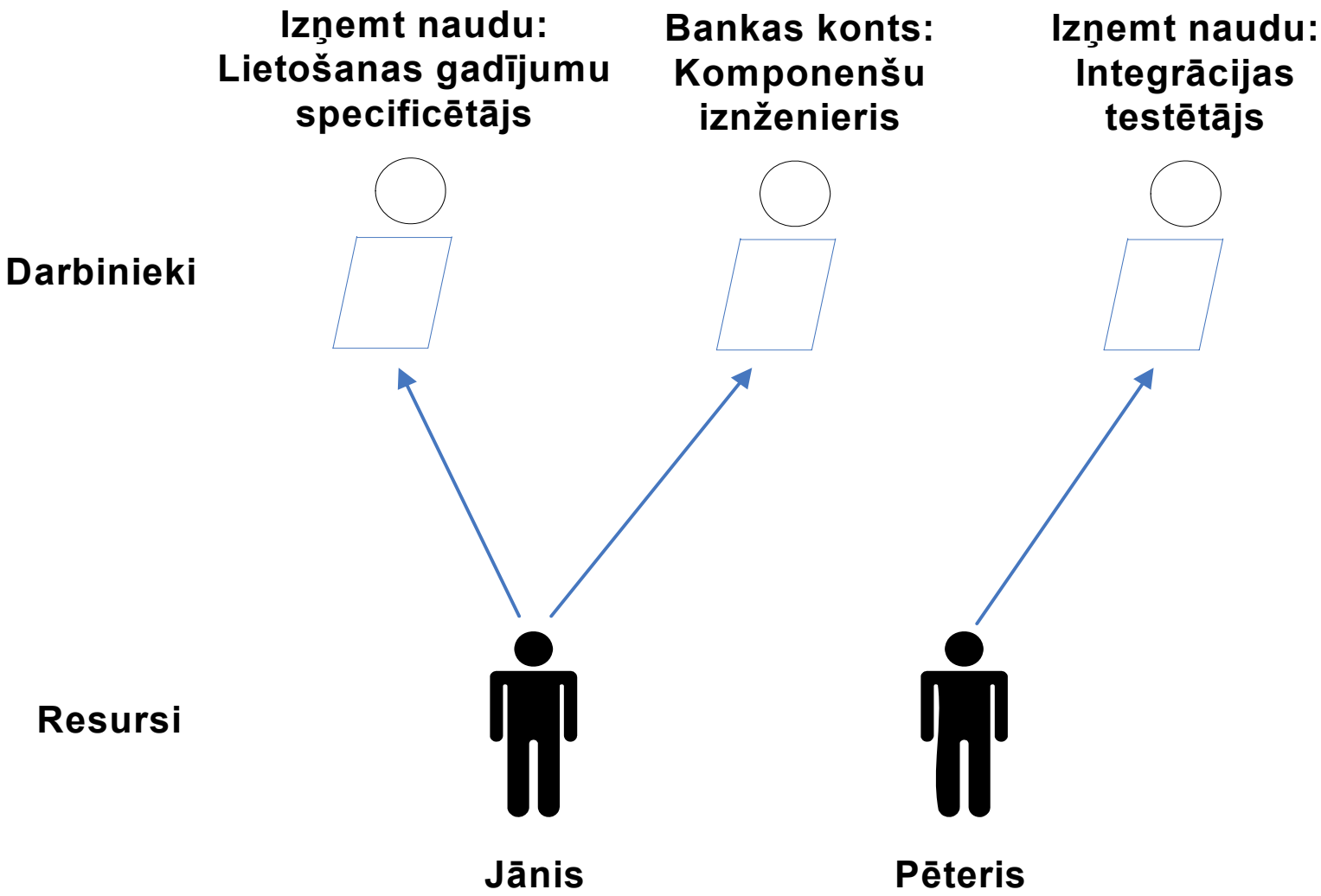
Lomu saraksts (1.daļa)

- Vadība:
  - projektu direktors
- Kvalitātes nodrošināšana:
  - kvalitātes vadītājs
- Tehniskā atbalsta vadība:
  - tehniskā atbalsta vadītājs
- Projektu vadība:
  - projekta vadītājs, vadošais projekta vadītājs, projekta vadītājs, grupas vadītājs
- Sistēmu arhitektūra:
  - sistēmas arhitekts
- Sistēmanalīze:
  - vadošais sistēmu analītiķis, vecākais sistēmu analītiķis, sistēmu analītiķis
- Programmēšana:
  - vadošais programmētājs, vecākais programmētājs, programmētājs
- Testēšana:
  - vadošais testēšanas speciālists, vecākais testēšanas speciālists, testēšanas speciālists

Lomu saraksts (2.daļa)

- Dokumentēšana:
  - vadošais dokumentētājs, vecākais dokumentētājs, dokumentētājs
- Grafiskais dizains:
  - vadošais grafiskais dizaineris, vecākais grafiskais dizaineris, grafiskais dizaineris
- Datu bāzes administrēšana/ pārvaldība:
  - vadošais datu bāzes administrators/ pārvaldnieks, vecākais datu bāzes administrators/ pārvaldnieks, datu bāzes administrators
- Sistēmu/ tīklu administrēšana:
  - vadošais sistēmu/ tīklu administrators, vecākais sistēmu/ tīklu administrators, sistēmu/ tīklu administrators
- Lietotāju atbalsts:
  - vadošais lietotāju atbalsta speciālists, vecākais lietotāju atbalsta speciālists, lietotāju atbalsta speciālists
- Risinājumu konsultācijas:
  - vadošais IT risinājumu konsultants, vecākais IT risinājumu konsultants, IT risinājumu konsultants
- Uzņēmējdarbības konsultācijas:
  - vadošais uzņēmējdarbības konsultants, vecākais uzņēmējdarbības konsultants, IT risinājumu konsultants
- Mācību koordinācija:
  - mācību vadītājs, mācību koordinators

Resursu izvietošana darbiniekos



Programmētāja nodarbinātības apraksts

- Programmētājs strādā organizācijās, kurās veic programmatūras izstrādi, ieviešanu vai uzturēšanu.
- Programmētājs spēj izstrādāt programmatūru atbilstoši funkcionalitātes, kvalitātes un resursietilpības nosacījumiem, spēj organizēt un vadīt programmētāju darba grupu, kā arī sistemātiski pilnveido zināšanas un prasmes.

Programmētāja pienākumi un uzdevumi

Pienākumi	Uzdevumi
1. Kodēšana	1.1. Analizēt ieejas un izejas datus
	1.2. Konfigurēt izstrādes vidi
	1.3. Rakstīt programmas kodu saskaņā ar projektējumu un kodēšanas vadlīnijām
	1.4. Konstruēt algoritmus
	1.5. Lasīt un analizēt svešus programmu tekstus
	1.6. Veidot lietotāja saskarni
	1.7. Skaņot programmas un veikt vienībtestēšanu
	1.8. Analizēt programmas izpildes laiku un to optimizēt
	1.9. Dokumentēt koda izmaiņas
	1.10. Veidot programmatūras instalāciju
	1.11. Veidot iebūvēto palīdzības sistēmu
	1.12. Apstrādāt un realizēt izmaiņu pieprasījumus un problēmu ziņojumus

Programmētāja pienākumi un uzdevumi

Pienākumi	Uzdevumi
2. Projektēšana	2.1. Iepazīties ar programmatūras projektējuma apraksta standartiem
	2.2. Veidot un aprakstīt programmatūras arhitektūru 2.3. Analizēt dažādus tehniskos risinājumus un izvēlēties piemērotāko 2.4. Veidot datu konceptuālo modeli un fizisko modeli 2.5. Veidot realizācijas modeli (klašu un/vai funkciju hierarhiju) 2.6. Konstruēt un aprakstīt algoritmus 2.7. Projektēt lietotāja saskarnes aprakstu 2.8. Sagatavot programmatūras projektējuma apraksta dokumentu
3. Programmatūras uzturēšana	3.1. Apstrādāt un realizēt problēmu ziņojumus un izmaiņu pieprasījumus
	3.2. Konsultēt programmatūras lietotājus 3.3. Veikt izmaiņu ietekmes analīzi 3.4. Veikt uzturamās programmatūras konfigurācijas pārvaldību 3.5. Sistematizēt uzturēšanas gaitā uzkrāto atbalsta informāciju

Programmētāja pienākumi un uzdevumi

Pienākumi	Uzdevumi
4. Programmatūras ieviešana	4.1. Veikt vides sagatavošanu programmatūras uzstādīšanai
	4.2. Izpildīt programmatūras uzstādīšanu un parametrizēšanu
	4.3. Iepazīties ar lietotāja dokumentāciju
	4.4. Veikt datu pārnešanu
	4.5. Sniegt konsultācijas programmatūras ieviešanas laikā
5. Programmatūras testēšana	5.1. Sagatavot testēšanas plānu
	5.2. Sagatavot testēšanas specifikāciju
	5.3. Analizēt programmas kodu
	5.4. Sagatavot testa piemēra datus
	5.5. Sagatavot testēšanas vidi
	5.6. Izpildīt testa piemērus
	5.7. Pierakstīt testēšanas gaitu un rakstīt problēmu ziņojumus
	5.8. Analizēt kļūdu avotus (prasības specifikācijā, projektējuma aprakstā, u.c.)
	5.9. Reproducēt lietotāja konstatētās kļūda
	5.10. Sagatavot testēšanas pārskata dokumentu

Programmētāja pienākumi un uzdevumi

Pienākumi	Uzdevumi
6. Prasību specificēšana	6.1. Iepazīties ar programmatūras prasību specifikācijas standartiem
	6.2. Noskaidrot lietotāja funkcionālās prasības
	6.3. Noskaidrot prasības lietotāja saskarnei
	6.4. Novērtēt datu apjomus un noskaidrot veiktspējas prasības
	6.5. Noskaidrot drošības, drošuma un vides prasības, prasības mijiedarbībai ar citām sistēmām un citas tehniskās prasības
	6.6. Analizēt prasību realizācijas iespējas
	6.7. Sagatavot programmatūras prasību specifikācijas dokumentu
7. Esošās sistēmas analīze	7.1. Intervēt projekta pasūtītāju un apkopot interviju rezultātus
	7.2. Iepazīties ar pasūtītāja darbību reglamentējošo dokumentāciju
	7.3. Iepazīties ar esošo programmu nodrošinājumu
	7.4. Apkopot sistēmas analīzes rezultātus vienotā dokumentā

Programmētāja pienākumi un uzdevumi

Pienākumi	Uzdevumi
8. Lietotāja dokumentācijas sagatavošana	8.1. Iepazīties ar lietotāja dokumentācijas standartiem
	8.2. Iepazīties ar lietotāja biznesa terminoloģiju 8.3. Rakstīt un noformēt lietotāja dokumentācijas tekstu 8.4. Saskaņot lietotāja dokumentāciju ar iebūvēto palīdzības sistēmu (Help)
9. Programmatūras projekta plānošana	9.1. Novērtēt darba uzdevuma darbietilpību un izpildes laiku un sastādīt kalendāro plānu
	9.2. Veikt individuālā darba plānošanu un kontroli
	9.3. Piedalīties projekta gaitas izpildes apspriešanā

Programmētājam nepieciešamas prasmes

Speciālās prasmes profesijā	Kopīgās prasmes nozarē	Vispārējās prasmes/ spējas
• Veidot un atkļūdot programmas • Pielietot projektējuma shēmas un diagrammas • Projektēt algoritmus un datu struktūras • Izvēlēties problēmas risināšanai adekvātus programmproduktus un līdzekļus • Veikt datu aizsardzības un drošības pasākumus • Konfigurēt darba vietu un darba rīkus • Lietot programmatūras izstrādes rīkus • Realizēt algoritmus, lietojot programmēšanas valodu • Analizēt programmas kodu • Realizēt lietotāja interfeisu • Programmēt, izmantojot interneta tehnoloģijas • Lietot datu pieprasījumu valodas • Lietot programmas koda kvalitātes pār-baudes rīkus • Mērīt programmatūras veiktspēju • Lietot labu programmēšanas stilu • Lietot programmatūras testēšanas paņēmienus • Veikt sistēmu analīzi un projektēšanu	• Lietot IT nozares standartus • Lietot IT terminoloģiju angļu un latviešu valodā • Lietot operētājsistēmas • Lietot teksta un grafikas redaktorus u.c. biroja lietojumprogrammas • Lietot ātrrakstīšanas paņēmienus • Piedalīties projektu vadīšanā	• Komunikatīvā prasme • Strādāt komandā (grupā) • Veikt darbu patstāvīgi • Plānot izpildāmos darbus un noteikt to prioritātes • Lietot informācijas meklēšanas un atlases līdzekļus • Sagatavot prezen-tā-cijas mate-riālus un pasā-kumus un vadīt tos • Pārliecināt citus un argumentēt savu viedokli • Noformēt lietišķos dokumentus • Ievērot profesionālās ētikas principus • Ievērot darba higiēnas un drošības prasības • Spēt sazināties angļu valodā

Programmētājam nepieciešamas zināšanas

Zināšanas	Zināšanu līmenis		
	Priekšstats	Izpratne	Pielietošana
Angļu valoda			
Matemātika			
Ekonomika un uzņēmējdarbība			
Saskarsme un profesionālā ētika			
Darba aizsardzība un ergonomika			
Lietojumprogrammatūras klasifikācija un izmantošana			
Programmēšanas valodas			
Operētājsistēmu klasifikācija un izmantošana			
Datu bāzu tehnoloģijas			
Datorsistēmu uzbūve un funkcionēšana			
Datortīklu tehnoloģijas			
IT nozares tiesību pamati un standarti			
Programmatūras inženierija			
Programmatūras izstrādes tehnoloģijas			
Objektorientētā programmēšana			
Datu struktūrās un algoritmos			
Interneta tehnoloģijās			
Programmatūras izstrādes projektu vadīšana			

Sistēmanalītiķa nodarbinātības apraksts

- Sistēmanalītiķis strādā organizācijās, kurās veic programmatūras izstrādi.
- Sistēmanalītiķis biznesprocesu analīzes rezultātā izstrādā prasības IT risinājumam un piedalās šo risinājumu attīstīšanā.
- Sistēmanalītiķis spēj izstrādāt sistēmas prasības atbilstoši funkcionalitātes, kvalitātes un resursietilpības nosacījumiem, spēj organizēt un vadīt sistēmanalītiķu darba grupu, kā arī sistemātiski pilnveido zināšanas un prasmes.

Sistēmanalītiķa pienākumi un uzdevumi

Pienākumi	Uzdevumi
1. Esošo biznessistēmu analīze	1.1. Iepazīties ar esošajiem IT risinājumiem
	1.2. Aprakstīt sistēmas darbību 1.3. Aprakstīt sistēmas struktūru 1.4. Aprakstīt informācijas plūsmas 1.5. Iepazīties ar biznesprocesu reglamentējošiem dokumentiem 1.6. Sagatavot priekšlikumus biznesprocesa uzlabošanai 1.7. Apkopot analīzes rezultātus vienotā dokumentā
2. Sistēmu prasību specifikācijas izstrāde	2.1. Iepazīties ar analīzes rezultātiem
	2.2. Iepazīties ar pasūtītāju darbību reglamentējošiem dokumentiem 2.3. Noskaidrot sistēmas ierobežojumus 2.4. Noskaidrot izstrādes un lietošanas vides prasības 2.5. Izstrādāt sistēmas funkcionālās prasības 2.6. Izstrādāt sistēmas nefunkcionālās prasības 2.7. Analizēt prasību realizēšanas iespējas 2.8. Sagatavot prasību specifikāciju 2.9. Saskaņot prasības un to prioritātes

Sistēmanalītiķa pienākumi un uzdevumi

Pienākumi	Uzdevumi
3. Piedalīšanās sistēmas izstrādē	3.1. Konsultēt sistēmas izstrādātājus
	3.2. Dot slēdzienu par IT risinājumu 3.3. Novērtēt IT risinājumu resursietilpību 3.4. Izstrādāt programmatūras arhitektūru 3.5. Konstruēt datu modeļus 3.6. Konstruēt sistēmas funkcionālos modeļus 3.7. Izvēlēties piemērotāko tehnisko risinājumu 3.8. Sagatavot projektējuma aprakstu 3.9. Apstrādāt problēmziņojumus
4. Darbu organizēšana	4.1. Novērtēt darbietilpību
	4.2. Novērtēt izstrādes riskus 4.3. Plānot savu un komandas darbu 4.4. Uzturēt darba vidi
	4.5. Kontrolēt darba izpildi 4.6. Sniegt pārskatu par darbu izpildi 4.7. Organizēt prezentācijas 4.8. Ievērot uzņēmuma un projekta standartus un procedūras 4.9. Pilnveidot standartus un procedūras

Sistēmanalītiķa pienākumi un uzdevumi

Pienākumi	Uzdevumi
5. Dokumentācijas veidošana	5.1. Iepazīties ar standartiem un vadlīnijām 5.2. Sagatavot dokumentus, ievērojot standartus un vadlīnijas 5.3. Veikt dokumentu apskates 5.4. Kontrolēt dokumentu izmaiņas 5.5. Sagatavot prezentācijas materiālus 5.6. Apkopot priekšlikumus dokumentu pilnveidošanai 5.7. Veikt lietišķo saraksti (ar pasūtītāju)
6. Pasūtītāju un lietotāju intervēšana	6.1. Plānot intervijas
	6.2. Sagatavot intervijas 6.3. Vadīt intervijas 6.4. Pierakstīt intervijas 6.5. Apkopot interviju rezultātus
7. Zināšanu un prasmju pilnveidošana	7.1. Sekot jaunumiem IT nozarē
	7.2. Sekot jaunumiem biznessfērās 7.3. Apmeklēt seminārus un kursus
	7.4. Lasīt tehnisko literatūru

Sistēmanalītiķim nepieciešamas prasmes

Speciālās prasmes profesijā	Kopīgās prasmes nozarē	■Vispārējās prasmes/ spējas
•Strādāt ar tehnisko dokumentāciju; standartiem •Veidot un lietot shēmas un diagrammas •Lietot sistēmanalīzes metodiku •Izvēlēties sistēmanalīzes metodiku •Lietot sistēmanalīzes atbalsta rīkus •Lietot resursu novērtēšanas metodes •Sagatavot tehniskos aprakstus •Lietot teksta un grafiskos redaktorus, prezentācijas programmas un izklājlapas	•Lietot IT nozares standartus •Lietot IT terminoloģiju angļu un latviešu valodā •Lietot operētājsistēmas •Lietot teksta un grafikas redaktorus u.c. biroja lietojumprogrammas •Piedalīties projektu vadīšanā	•Prasme komunicēt •Strādāt komandā (grupā) •Veikt darbu patstāvīgi •Plānot izpildāmos darbus un noteikt to prioritātes •Lietot informācijas meklēšanas un atlases līdzekļus • Sagatavot prezentācijas materiālus un pasākumus un vadīt tos •Pārliecināt citus un argumentēt savu viedokli •Noformēt lietišķos dokumentus •Intervēt pasūtītājus un lietotājus, izmantojot intervēšanas metodiku. •Ievērot profesionālās ētikas principus •Ievērot darba higiēnas un drošības prasības •Spēt sazināties angļu valodā

Sistēmanalītiķim nepieciešamas zināšanas

Zināšanas	Zināšanu līmenis		
	Priekšstats	Izpratne	Pielietošana
Angļu valoda			
Matemātika			
Ekonomikas un uzņēmējdarbības pamati			
Saimniecisko procesu tehnoloģijas pamati			
Saskarsme un profesionālā ētika			
IS ekonomikas pamati			
Programmatūras projektu vadīšana			
Tehniskā rakstīšana			
Intervēšanas metodika			
Prasību inženierija			
Objektorientētā analīze un projektēšana			
Sistēmu arhitektūra un projektēšana			
Programmatūras inženierija			
Lietotāju saskarne			
Informācijas aizsardzība, drošība			
Programmēšanas pamati			
Datu struktūras un algoritmi			
E -bizness			
IT nozares tiesību pamati un standarti			

Komandas organizācija

- Komandu nav iespējams izveidot vienā dienā.
- Tipiski komandas veidošanā izšķir četras fāzes:
  - Forming (formēšanās): komandas locekļi definē mērķus, lomas, un virzienu
  - Storming (ārdīšanās) : komandas locekļu lomu un atbildību sadalīšana
  - Norming (normēšanās): procedūru, standartu un kritēriju saskaņošana
  - Performing (darbošanās): komanda sāk funkcionēt kā sistēma

Veiksmīga projekta komandas veidošanas principi

- 1. pareizi izvēlēties projekta darba grupu.
- 2. mācību vides izveide
- 3. vide, kurā valda uzticība.
- 4. ļaut izrādīt neapmierinātību.
- 5. palūgt nevis prasīt.
- 6. sasniegumu atzīšana.
- 7. izpildīt tikai tās darbības un veidot tikai tos artefaktus, kas ir nepieciešami mērķa sasniegšanai
- 8. izstrādes procesu ir nepieciešams orientēt uz riskiem.
- 9. jāsāk ar drošu pamatu.
- 10. projekta procesa pielāgošana.
- 11. biežāk ir jālieto veselais saprāts
- 12. instrumentālie līdzekļiem ir jāatpelna viņu apguves izmaksas.
- 13. laika plānošana apmācībai
- 14. savus instrumentālos līdzekļus ir nepieciešams radīt, ja ir nepieciešamība.

Veiksmīga projekta komandas veidošanas principi

- Sākumā komanda, pēc tam projekts
- Pareiza komandas dalībnieku izvēle – galvenais projekta veiksmes faktors
- Jāizvēlas komandas darbā orientēti dalībnieki, nevis individuālās "super-zvaigznes"
- Komandā jābūt vienam vienīgam vadītājam
- Jātur sasaiste
- Jāsadala atmaksa
- Jāpieraksta viss par ko tiek sarunāts
- Jāizdzen atpalikušais komandas dalībnieks

Nobeiguma secinājumi

- Neeksistē vienota risinājuma komandas organizācijas problēmas risināšanā
- "Pareizais" ceļš ir atkarīgs no
  - Produkta
  - Organizācijas vadītāja uzskatiem
  - Iepriekšējas pieredzes dažādās komandas struktūrās
- Ir maz pētījumu par programmatūras izstrādes komandas organizāciju
  - Komandas organizācija ir balstīta uz pētījumiem par grupas dinamiku kopumā
- Bez attiecīgajiem eksperimentāliem rezultātiem ir sarežģīti definēt optimālo komandas organizāciju specifiskam produktam

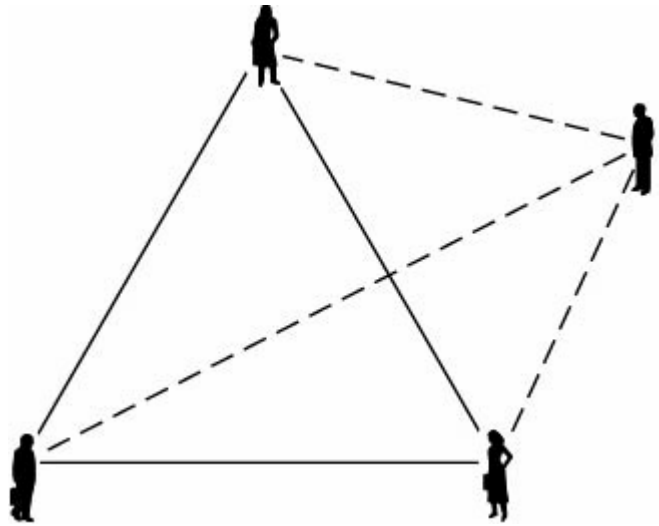
- Stephen R. Schach
- "Object-Oriented and Classical Software Engineering"
- CHAPTER 4 - "TEAMS"
- Fifth Edition, WCB/McGraw-Hill, 2002
- Sixth Edition, WCB/McGraw-Hill, 2006

Programmēšanas komandas organizācija

- Produktam jābūt gatavam pēc 3 mēnešiem, ir nepieciešams 1 cilvēks-gadā
- Risinājums
  - Ja viens programmētājs var izveidot produktu 1 gada laikā, četri programmētāji to nevar pabeigt pēc 3 mēnešiem
- Nonsense
  - Četri programmētāji pie šī produkta strādās ap gadu
  - Kvalitāte būs daudz zemākā
- Uzdevuma sadalīšana
  - Ja viens zemnieks var apstrādāt lauku 10 dienās, 10 zemnieki varēs to pašu lauku apstrādāt 1 dienā
  - Viena sieviete var iznesāt bērnu 9 mēnešos, bet 9 sievietes nepaveiks šo uzdevumu 1 mēnesī
  - Atšķirībā no bērna "ražošanas" programmēšanas uzdevumu ir iespējams sadalīt starp komandas locķļiem
  - Atšķirībā no zemniekiem, programmēšanas komandas locekļiem ir jāsazinās savā starpā un jāapmainās ar rezultātiem

Komunikācijas problēmas

- Piemērs
  - Ir 3 sazināšanas kanāli starp 3 programmētājiem projektā. Termiņš tuvojas, bet produkts nav gatavs
- "Acīmredzams" risinājums:
  - Papildināt komandu ar ceturto programmētāju
- Bet pārējiem trīs ir jāpaskaidro
  - Kas jau ir izdarīts
  - Kas vēl nav izdarīs
- Brūka līkums (Brooks's Law)
  - Papildinot komandu ar vēl vienu cilvēku laikā, kad projekts kavējas, aizkavē produkta nodošanu pat vairāk

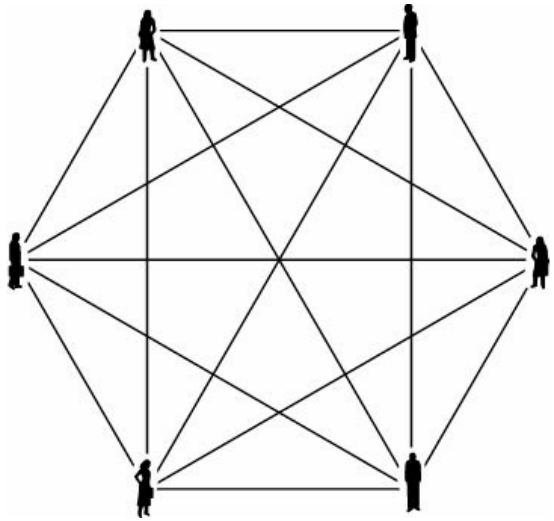


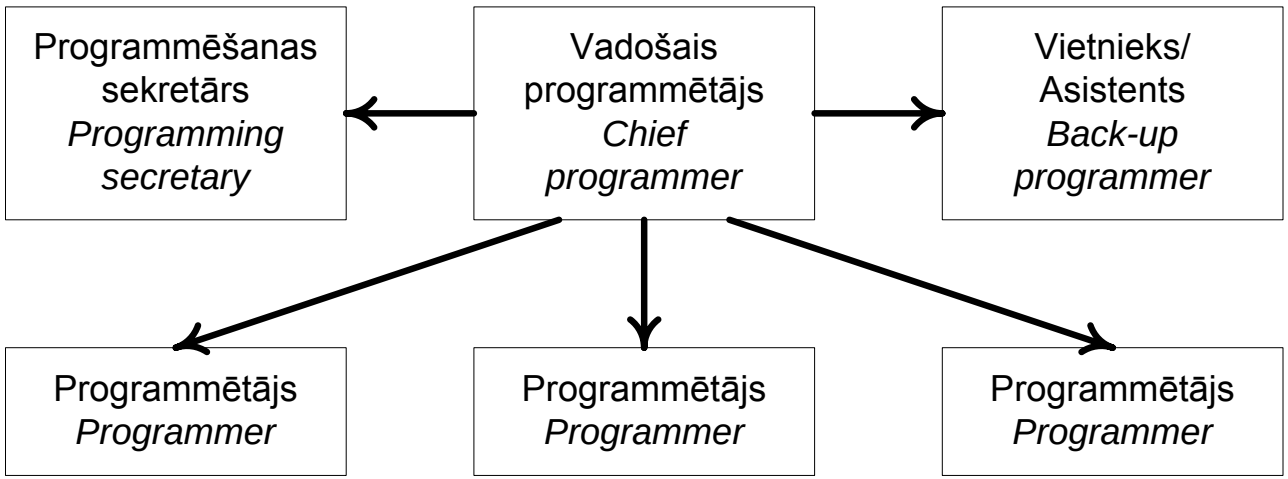
Demokrātiskā komandas organizācija

- Grupa pieņem lēmumus kopā
  - Nominmāls komandas līderis
- Veicina neegoistisku programmēšanu
  - Veicina meklēt kļūdas "kopējā" kodā
- Priekšrocības
  - Pozitīva attieksme kļūdu meklēšana → tiek atklātas agrāk
  - Labi strādā sarežģītu problēmu risināšanā (piem. pētījumos)
- Trūkumi
  - Sarežģīti uzspiest kaut ko darī
  - Produkts top spontāni
  - Sarežģīti noteikt apjomu
- Vai tas neatgādina kaut ko?

Vadošā programmētāja komanda (Autoritāra komandas organizācija) - Brooks, 1971; Baker, 1972

- 6 cilvēku komandā
  - piecpadsmit 2-cilvēku komunikācijas kanāli
  - Kopējais skaits 2-, 3-, 4-, 5-, un 6-cilvēku apakšgrupām ir 57
  - nevar izdarīt 6 cilvēk-mēnešu darbu vienā mēnesī
- Seši programmētāji, bet jau tikai 5 komunikācijas kanāli





Vadošā programmētāja komandas locekļi

- Vadošais programmētājs (Chief programmer)
  - Successful manager and highly skilled programmer
  - Does the architectural design
  - Allocates coding among the team members
  - Writes the critical (or complex) sections of code
  - Handles all the interfacing issues
  - Reviews the work of the other team members
  - Is personally responsible for every line of code
- Vietnieks/Asistents (Back-up programmer)
  - Necessary only because the chief programmer is human
  - The back-up programmer must be in every way as competent as the chief programmer
  - Must know as much about the project as the chief programmer
  - Does black-box test case planning and other tasks that are independent of the design process
- Programmēšanas sekretārs (Programming secretary)
  - A highly skilled, well paid, central member of the chief programmer team
  - Responsible for maintaining the program production library (documentation of project), including:
  - Programmers hand their source code to the secretary who is responsible for
- Programmētāji (Programmers)
  - Do nothing but program
  - All other aspects are handled by the programming secretary

The New York Times Project

- Chief programmer team concept
  - first used in 1971, by IBM, to automate the clippings data bank of The New York Times
- Chief programmer—F. Terry Baker
- 83,000 source lines of code (LOC) were written in 22 calendar months, representing 11 person-years
- After the first year, only the file maintenance system had been written (12,000 LOC)
- Most code was written in the last 6 months
- 21 faults were detected in the first 5 weeks of acceptance testing
- 25 further faults were detected in the first year of operation
- Almost half the subprograms (usually 200 to 400 lines of PL/I) were correct at first compilation
- But, after this fantastic success, no comparable claims for chief programmer team concept have been made

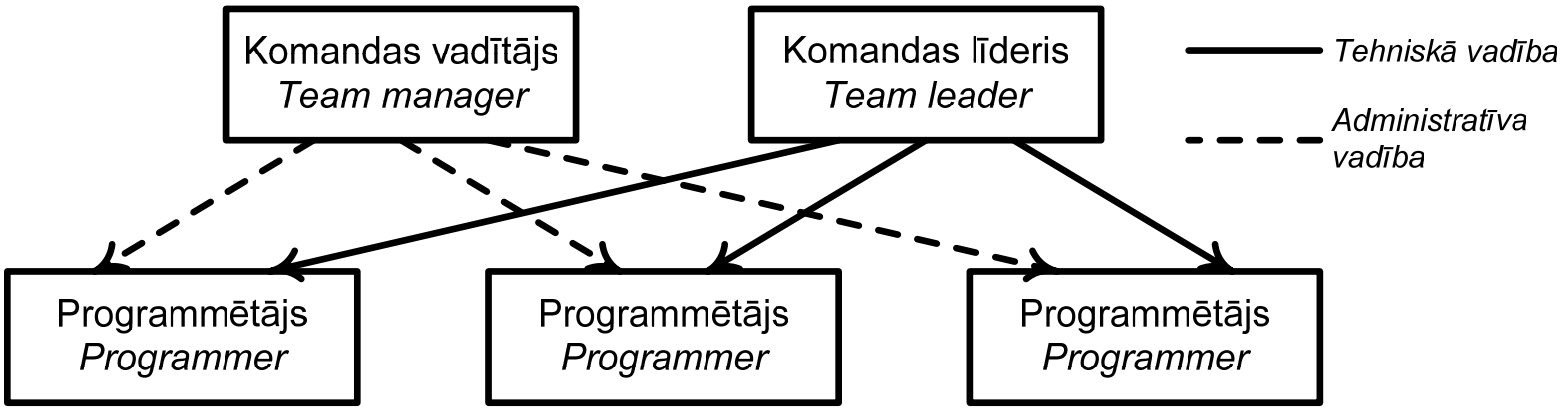
Impracticality of Classical CPT

- Chief programmer must be a highly skilled programmer and a successful manager
  - Shortage of highly skilled programmers
  - Shortage of successful managers
  - Programmers and managers "are not made that way"
- Back-up programmer must be as good as the chief programmer
  - But he/she must take a back seat (and a lower salary) waiting for something to happen to the chief programmer
  - Top programmers, top managers will not do that
- Programming secretary does nothing but paperwork all day
  - Software professionals hate paperwork
- Classical CPT is impractical

Beyond CP and Democratic Teams

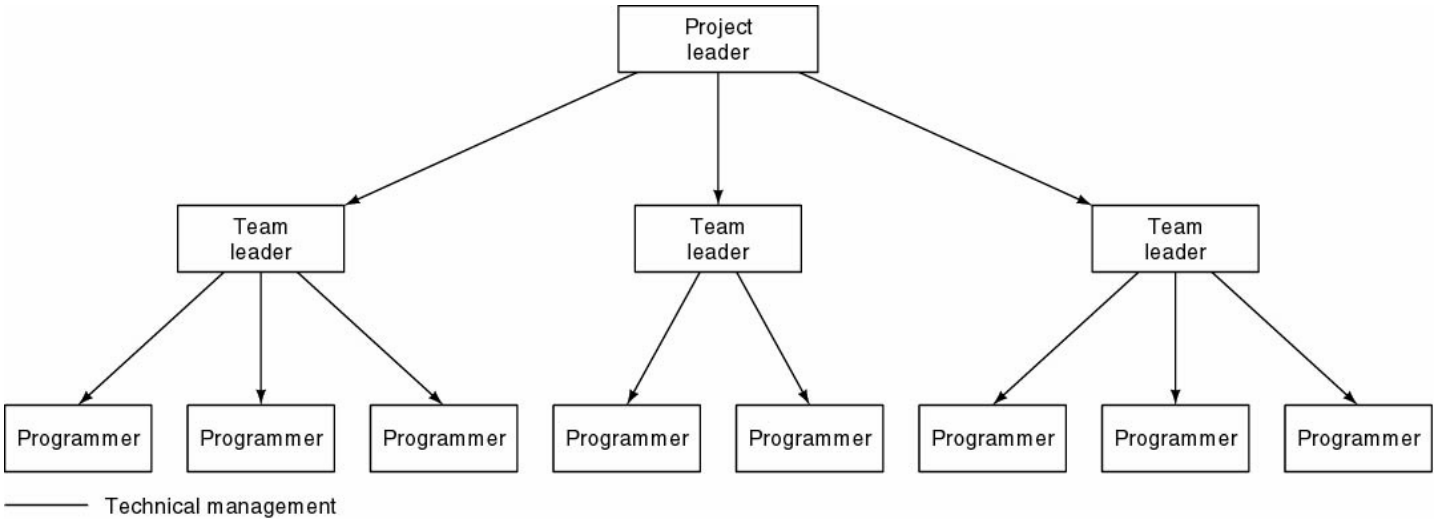
- We need ways to organize teams that
  - Make use of the strengths of democratic teams and chief programmer teams, and
  - Can handle teams of 20 (or 120) programmers
- Democratic teams
  - Positive attitude to finding faults
- Use CPT in conjunction with code walkthroughs or inspections
- Potential Pitfall
- Chief programmer is personally responsible for every line of code.
  - He/she must therefore be present at reviews
- Chief programmer is also the team manager
  - He/she must therefore not be present at reviews!
- Solution
  - Reduce the managerial role of the chief programmer

Decentralizēta vadība



- Sadalīta komandas vadība starp diviem cilvēkiem:
  - Komandas līderis ir atbildīgs par tehniskajiem lēmumiem
  - Komandas vadītājs ir atbildīgs par budžeta & izpildes ekspertīzi (administratīvie lēmumi)
  - Daži kopējie lēmumi (piem. cilvēku pieņēmšana komanda, vakances utml.)

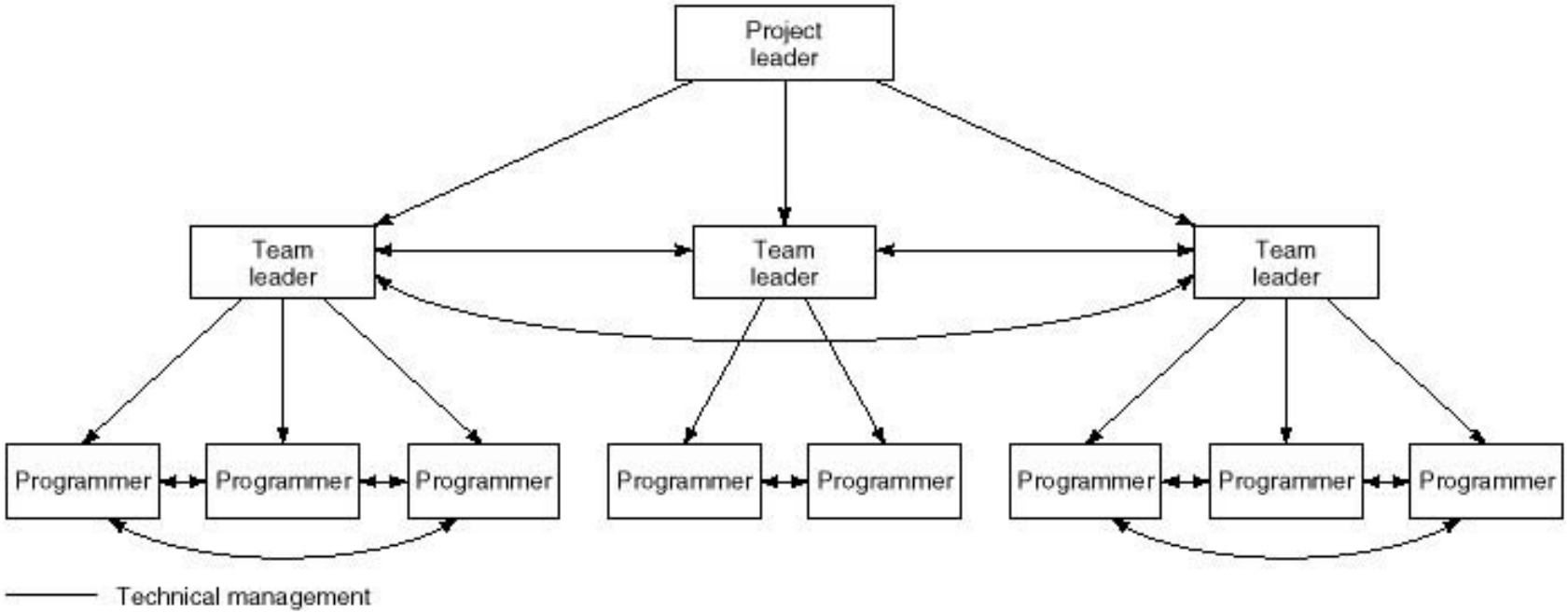
Larger Projects



- Nontechnical side is similar
- For even larger products, add additional layers
- Decentralize the decision-making process where appropriate
- Useful where the democratic team is good

Technical Managerial Approach

- Split team management across two people
  - team leader responsible for technical decisions
  - manger responsible for budget & performance appraisals
  - some overlap that must be clarified (e.g., vacation for key people)
  - decentralize decision-making to get advantages of democratic teams



Synchronize-and-Stabilize Teams

- Used by Microsoft
- Products consist of 3 or 4 sequential builds
- Small parallel teams
  - 3 to 8 developers
  - 3 to 8 testers (work one-to-one with developers)
  - Team is given the overall task specification
  - They may design the task as they wish
- Why this does not degenerate into hacker-induced chaos
  - Daily synchronization step
  - Individual components always work together
- Rules
  - Must adhere to the time to enter the code into the database for that day's synchronization
- Analogy
  - Letting children do what they like all day…
  - … but with a 9 P.M. bedtime

Extreme Programming Teams

- Feature of XP
  - All code is written by two programmers sharing a computer
    - "Pair programming"
- Advantages of Pair Programming
  - Test cases drawn up by one member of team
  - Knowledge not all lost if one programmer leaves
  - Inexperienced programmers can learn
  - Centralized computers promote egoless programming

Common SDLC Roles	Heavyweight methodologies		Lightweight (agile) methodologies
	MSF Role Clusters	RUP Role Sets	XP Roles
Project Manager	Program Management	Manager set	Manager
Development Manager	Product Management		Tracker / Programmer
Subject Matter Expert	none / Additional Team Members	none / Additional role set	Tracker
Functional Analyst	Product Management	Analyst role set	Customer
Solutions Architect	Program Management	Developer role set	Coach
Development Lead			Programmer
Developer	Development		
Quality Assurance	Testing	Tester Role Set	Customer / Tester
Deployment	Release management	Manager set	Programmer
Training	User experience	Additional role set	

Common Roles Mapping

Methodologies | Roles in Software Development Process

Software Development Life Cycle

MSF	RUP	Project Manager	Development Manager	Functional Analyst	Solutions Architect	Development Lead	Developer	Quality Assurance	Deployment	Training
Envisioning	Inception	Overall goals	Solution concept and design goals	Business-needs analysis	Feasibility study		Technology research & analysis	Testing strategies and acceptance criteria	Deployment implications, operations management and acceptance criteria	User expectations
Planning	Elaboration	Business-needs Analysis, Communications Plan	Budgets, Master Project Plan & Schedule	Solution Conceptual and Logical Design	Solution Logical and Physical Design	Functional Specification	Technology Validation, Development Plan & Schedule	Requirements, Test Plan & Schedule	Operations Issues and Requirements, Pilot/Deployment Plan & Schedule	User Needs, Usage Scenarios, Accessibility and Localization Issues
Developing	Construction	Customer Communication, Manage Expectations	Schedule, Functional Specification and Plan Tracking		Customer Communication	Schedule, Functional Specification	Code/ Infrastructure	Test Plans, Bug Triaging, Documentation Testing	Rollout Plans, Pilot Plans, Operations Documentation, Site Logistics	Training Development, User Interface, Usability Testing
Stabilizing		Product Launch Plans, Customer Relations	Schedule, Bug Tracking	Customer Relations	Solution Optimization, Bug Resolution			Testing, Bug Reports	Deployment Plan, Operations and Support Training, Pilot Testing	Refinement of Training Materials and Acceptance Testing
Deploying	Transition	Customer Sign-Off, Feedback and Project Assessment	Stabilization		Project Assessment	Problem Resolution, Escalation		Performance/ Training Testing	Deployment Management, Change Approval	User Training

Grupas --- Komandas

- Grupas
  - Stingrs vadītājs
  - Individuālā pakļautība
  - Organizācijas mērķi
  - Individuāla darba produkti
  - Produktīvas tikšanās (ar rezultātu)
  - Veiktspējas izmērīšana no ietekmes uz citiem
  - Sadalīta strādāšana
- Komandas
  - Sadalīta vadība
  - individuālā & abpusēja pkļautība
  - Specifiski komandas mērķi
  - Kolektīva darba produkti
  - "open-ended" tikšanās
  - Veiktspējas mērīšana no tā, kā produkts darbojas
  - Kopīga strādāšana

- Komandas & produkta veiktspēja nav atdalāmas
- Komanda ir vairāk nekā tās daļu summa

Keys to Team Success

- Common commitment
  - requires a purpose in which team members can believe
    - "prove that all children can learn", "revolutionizing X…"
- Specific performance goals
  - comes directly from the common purpose
    - "increasing the scores of graduates form 40% to 95%"
  - helps maintain focus – start w/ something achievable
- A right mix of skills
  - technical/functional expertise
  - problem-solving & decision-making skills
  - interpersonal skills
- Agreement
  - action item assignment, when to meet & work, schedules

Nobeiguma secinājumi

- Neeksistē vienota risinājuma komandas organizācijas problēmas risināšanā
- "Pareizais" ceļš ir atkarīgs no
  - Produkta
  - Organizācijas vadītāja uzskatiem
  - Iepriekšējas pieredzes dažādās komandas struktūrās
- Ir maz pētījumu par programmatūras izstrādes komandas organizāciju
  - Komandas organizācija ir balstīta uz pētījumiem par grupas dinamiku kopumā
- Bez attiecīgajiem eksperimentāliem rezultātiem ir sarežģīti definēt optimālo komandas organizāciju specifiskam produktam

- Stephen R. Schach
- "Object-Oriented and Classical Software Engineering"
- CHAPTER 4 - "TEAMS"
- Fifth Edition, WCB/McGraw-Hill, 2002
- Sixth Edition, WCB/McGraw-Hill, 2006