Metrike za poboljšavanje procesa razvoja softvera

Uvod

- Upotreba mjerenja za unapređivanje procesa razvoja softvera
- U prethodnim lekcijama su metrike korišćenje za kontrolu performansi tokom razvoja i zavisile su od načina razvoja, proces modela i delivery modela
- U ovoj lekciji koristimo iste metrike ali sagledavamo infromacije na drugi način i sa drugim ciljem
  - metrics can’t tell you what the underlying problems are—they can only raise a flag to get your attention

Metrike nezavisne od procesa razvoja

- Potrebno je eliminsati zavisnost metrike od metodologije razvoja softvera

- Tada metrike postaju korisne i sa stanovišta unapređivanja metologije razvoja, u suprotnom promjena metodologije znači i nemogućnost primjene određene metrike

Tehničke metrike

- Često se generiši automatski iz podataka dostupnih tokom razvoja (npr. alati za praćenje taskova)
- Obično imaju upotrebu unutar tima koji razvija proizvod, nijesu od značaja za top menadžment
- Mogu da ukažu, na primjer, na softverske module na kojima se pojavljuje najviše tehničkih problema ili module koji u značajnoj mjeri ne prolaze automatske testove

Human metrike

- Mjere zadovoljstvo kod članova tima, eventualno i klasifikuju članove tima na određene tipove ličnosti ili načine razmišljanja

- Posebno dobijaju na značajaju sa prodorom dominantno kolaborativnih metodologija razvoja

Anti primjene

- Tretiranje ljudi kao resursa ili *interchangeable machine parts*

- Resurs je sredstvo čije performanse uglavnom mogu da budu procijenjene sa visokim stepenom sigurnosti
  - U ovom smislu resurs može u svakom trenutku biti zamijenjem sa istim resursom

- Član tima ne može da bude posmatran kao resurs iz prethodnog paragrafa

Anti primjene (2)

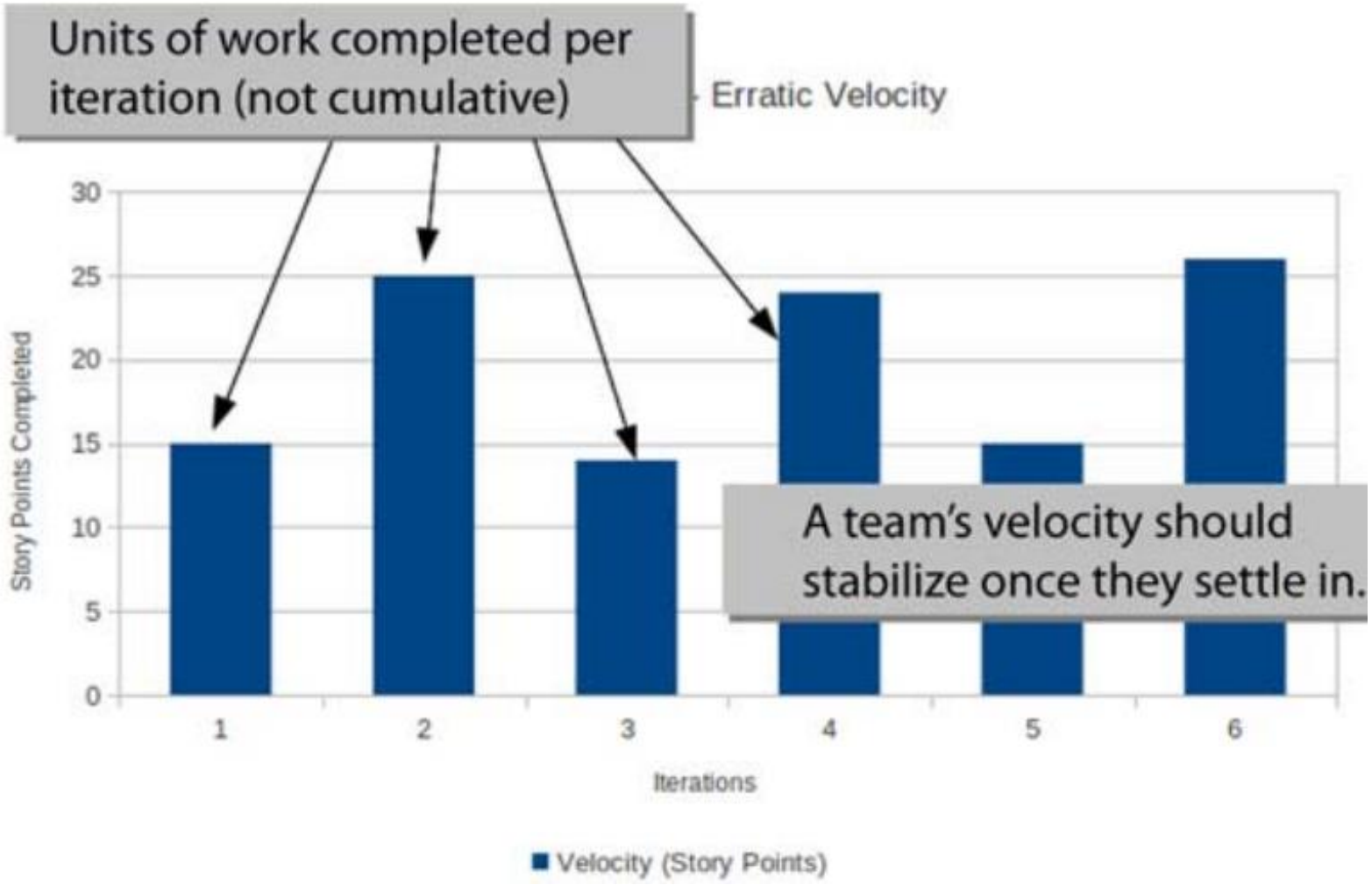
- Upravljanje progresom uglavnom podrazumijeva mjerenje rezultata a ne aktivnosti
  - Status projekta projenjujemo na osnovu delivery rate-a, brzine trošenja budžeta itd.
  - Mjerenje aktivnosti kao što je broj radnih sati ili dana članova tima ne može doprinijeti procjeni statusa

- Tell me how you measure me, and I will tell you how I will behave

Velocity

- Velocity se može koristiti za procjenu performansi tima na osnovu empirijskih procjena dobijenih u posljednjim iteracijama
  - Neophodno je definisati konzistentni način procjene taskova ili work item-a, kao što su npr. story poeni
  - Neophnodno je primjenljivati time-box iterativni postupak kada u svakoj iteraciji mora biti isporučen inkrement u smislu production-ready funkcionalnosti
  - Velocity se pod gornjim uslovima može dobiti „brojanjem“ story poena za tekuću iteraciju
- Rolling window
- Consistent velocity vs. variations in velocity

Primjer za adaptivni pristup



Primjer za adaptivni pristup (2)

- Varijacije u mjerenju velocity mogu da ukažu na:
  - Taskovi su preobimni za jednu iteraciju pa tim mora da nauči kako da efikasnije definiše dekompoziciju taskova
  - Taskovi su blokirajući
  - Tim nije cross-functional i ne postiže efekat kolaboracije već funkcioniše po waterfall principima
  - Ne primjenjuje se strogo time-boxed iterativni postupak već se dozvoljava proširivanje opsega u toku same iteracije

Velocity – anti pattern-i

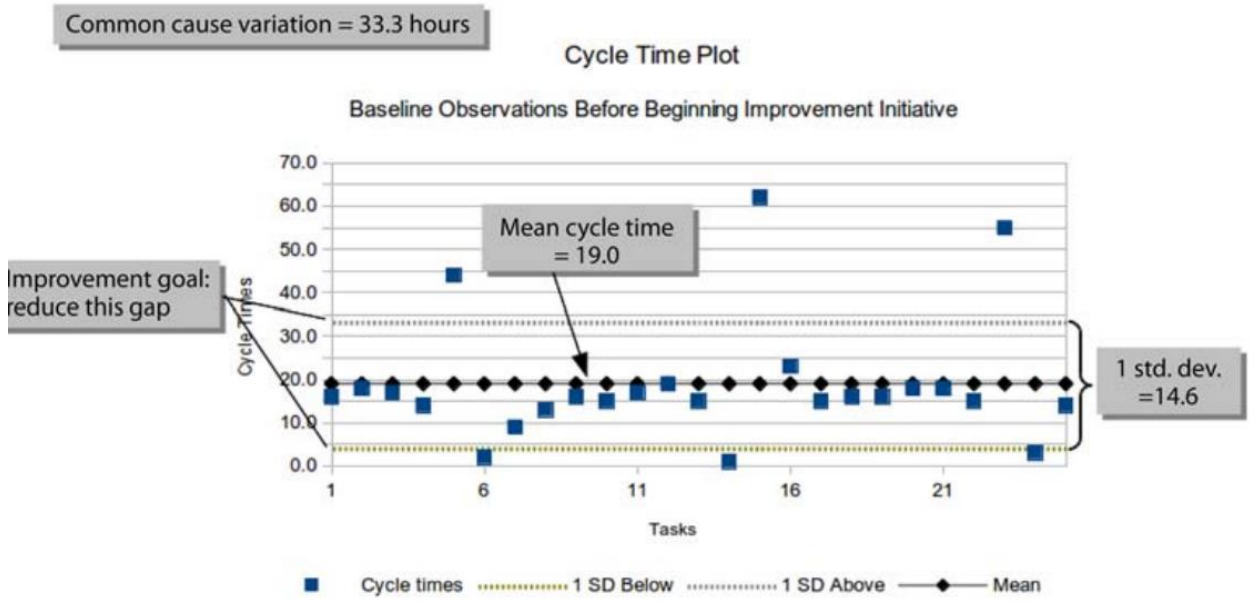
- Postavljanje ciljanih vrijednosti za velocity
- Oslabiti definiciju taskova koji su DONE
- Mijenjanje proces modela, odustajanje od time-boxed iteracija

Cycle time

- Ukazuje na prosječno vrijeme koje je potrebno timu da kompletira jedan work item
- Ukazuje na razliku u implementaciji malih i velikih work item-a
- Sa stanovišta poboljšanja procesa razvoja želimo
  - Smanjiti prosječno vrijeme po work item-u
  - Smanjiti razliku u implementaciju najmanjih i najvećih work item-a

Primjer 1

- Adaptivni način
- Potrebno je obezbijediti procjenu koliko tim može da isporuči funkcionalnosti u jednoj distribuciji
- Problem su velika odstupanja za cycle time



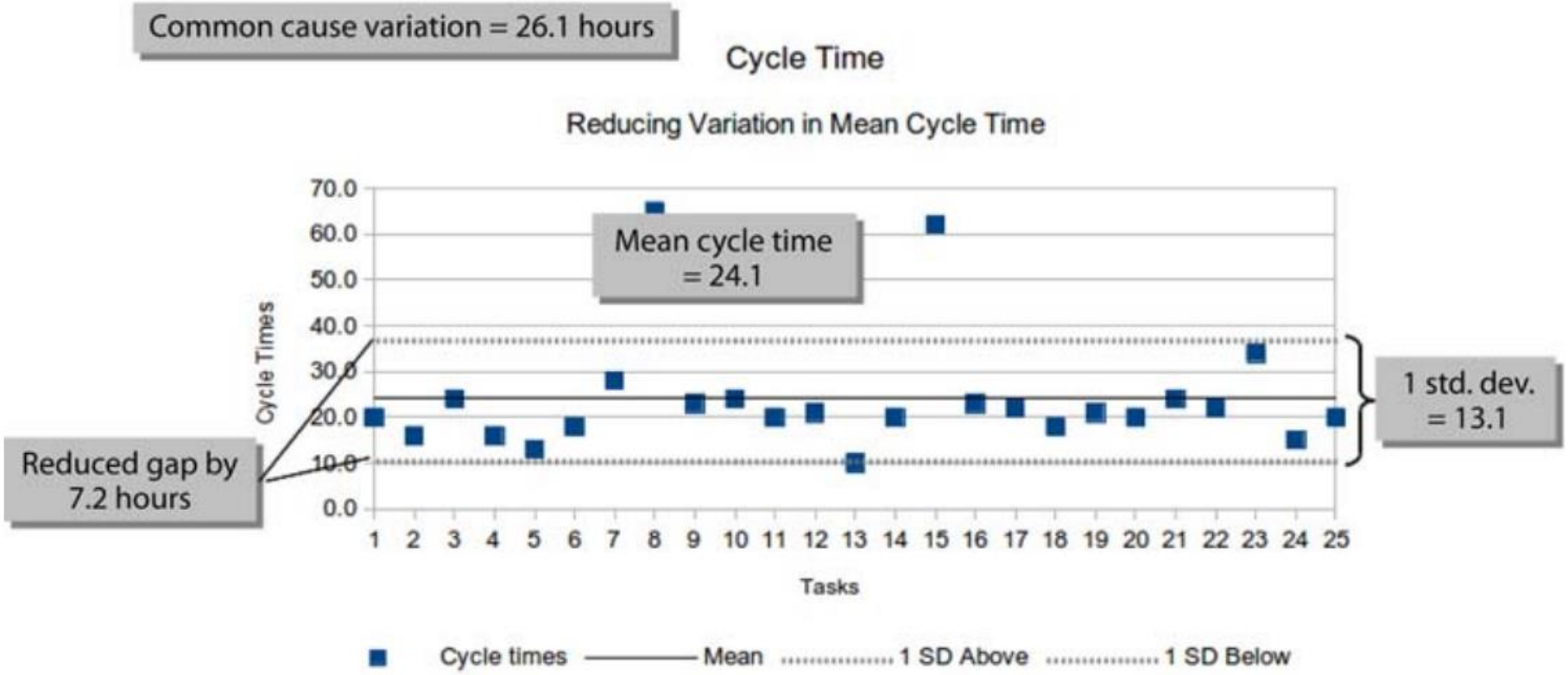
Primjer 1 - Nastavak

- Ako isključimo izuzetke, vrijeme za kompletiranje jednog work item-a je između 4.4h i 33.7h
  - it’s difficult to assure stakeholders of any particular delivery rate
- Potrebno je postići redukciju ovog intervala kako bi procjena u smislu planiranja bila određenija

- Mogući uzroci su:
  - Work itemi nijesu podijeljeni kao similarly sized chunks
  - Pojavljuju se tehnički izazovi koji su vidljiju tek kada počne razvoj
  - Tim zavisi od spoljnih resursa koji nepredvidljivo kasne
  - Sistem koji se razvija zavisi od drugih sistema

Primjer 1 - Nastavak (2)

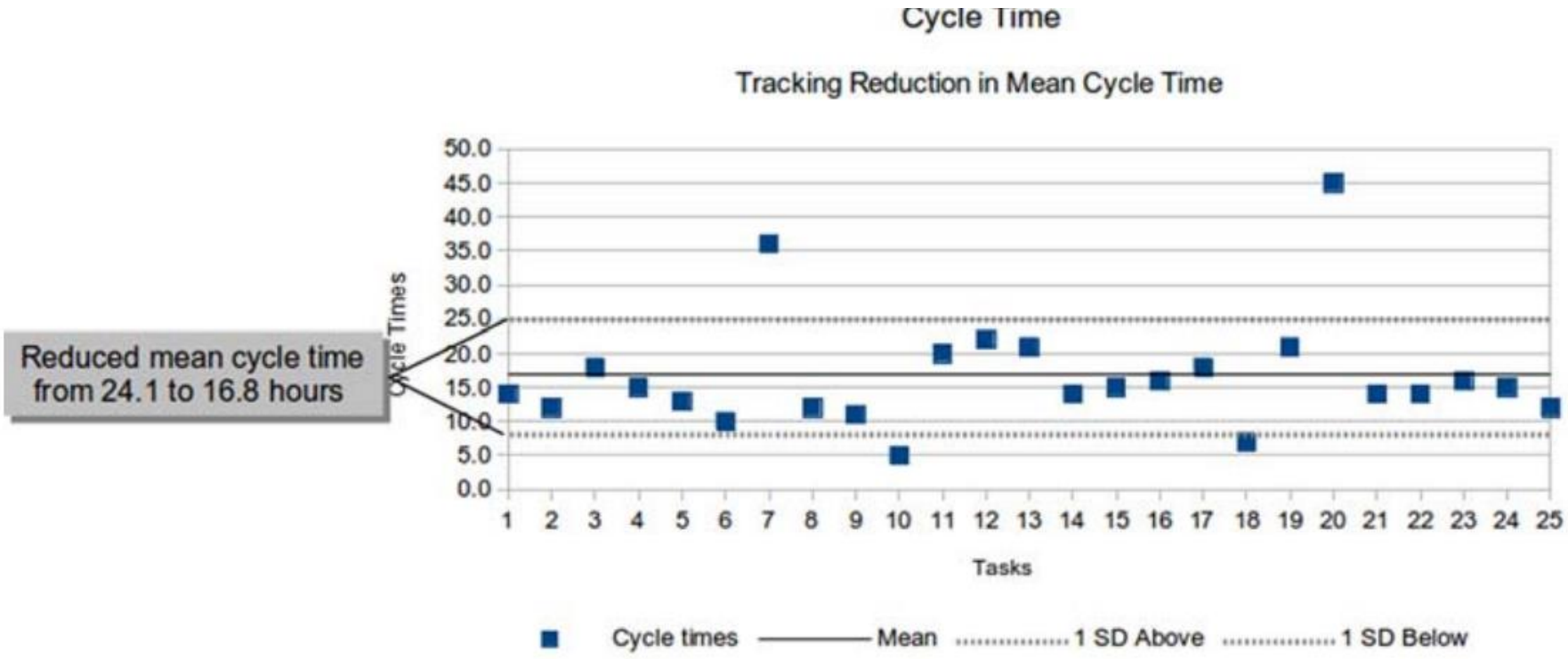
- Poslije promjene procesa rada grafikon je sljedeći. Šta je zaključak?



Primjer 2

- Prethodni grafikon ukazuje na povećanje prosječnog vremena po work item-u. Kako smanjiti ovo vrijeme?
- Mogući uzroci:
  - Preveliki broj WIP – work in process
  - Blokiranje work item-a
  - Ne postojanje automatizovanih postupaka za testiranje, konfigurisanje, deploy
  - Low bus number – nedovoljan broj članova tima posjeduje potrebna znanja

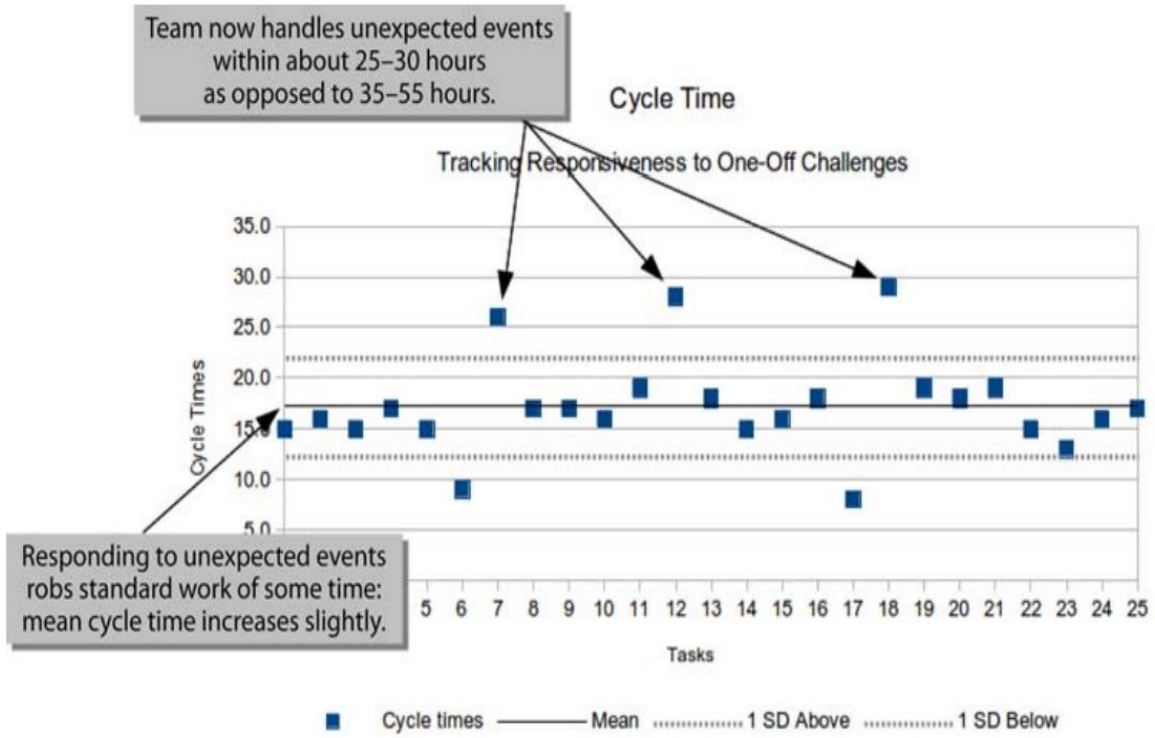
Primjer 2 - Nastavak



Primjer 3

- Common-cause vs. Special cause variations

- Specijalne akcije
  - Expediting work item
  - Swarming
  - Privremeno povećanje WIP limita



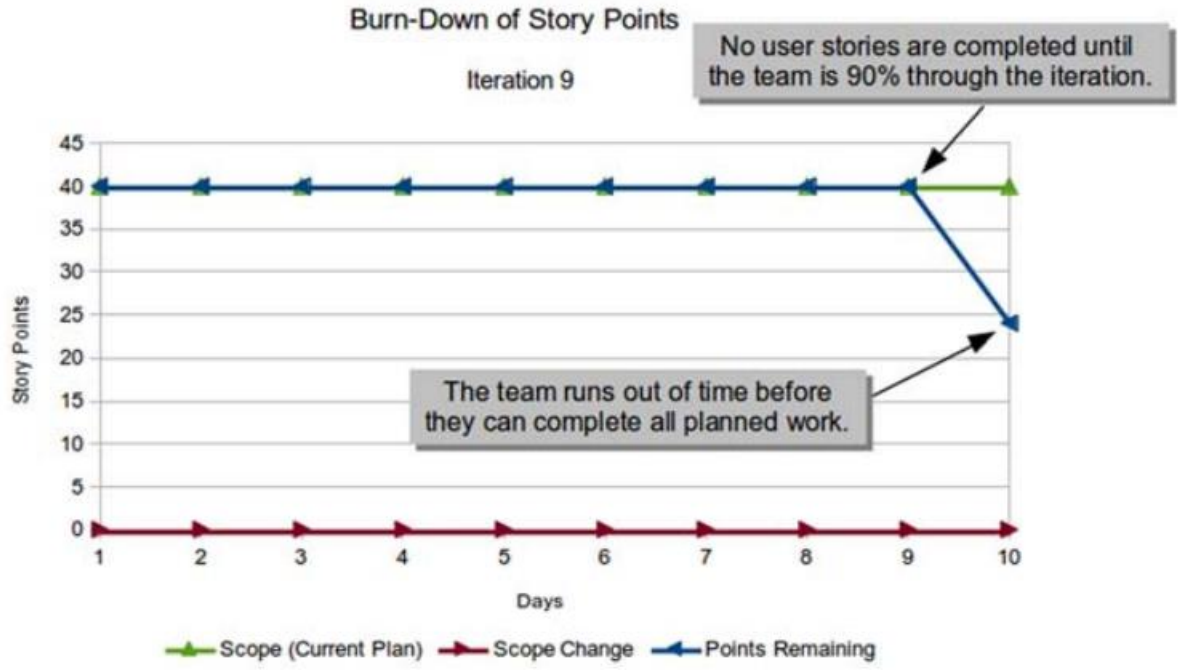
Burn chart

- Burn-up chart i burn-down chart

- Pokazuje koliko tim bez teškoća i zadrški napreduje ka cilju (burn-up) ili se približava nuli (burn-down)

Primjer

- Koristi se adaptivni pristup sa time-box proces modelom, dužina iteracije je dvije sedmice
- Primijećeno je da veliki broj work item-a ostaje nedovršeno što stvara pritisak na tim na kraju svake iteracije



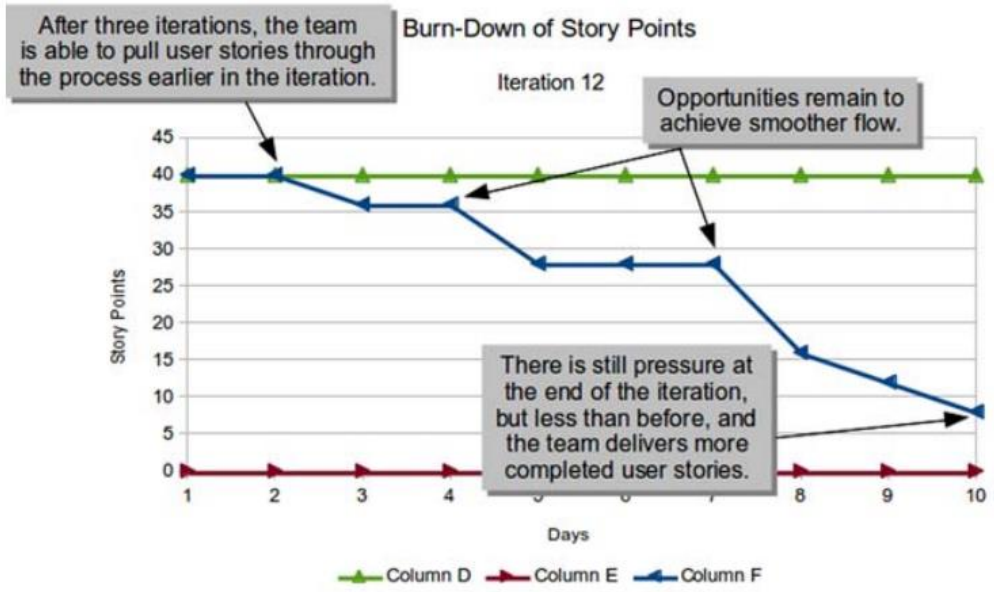
Primjer (2)

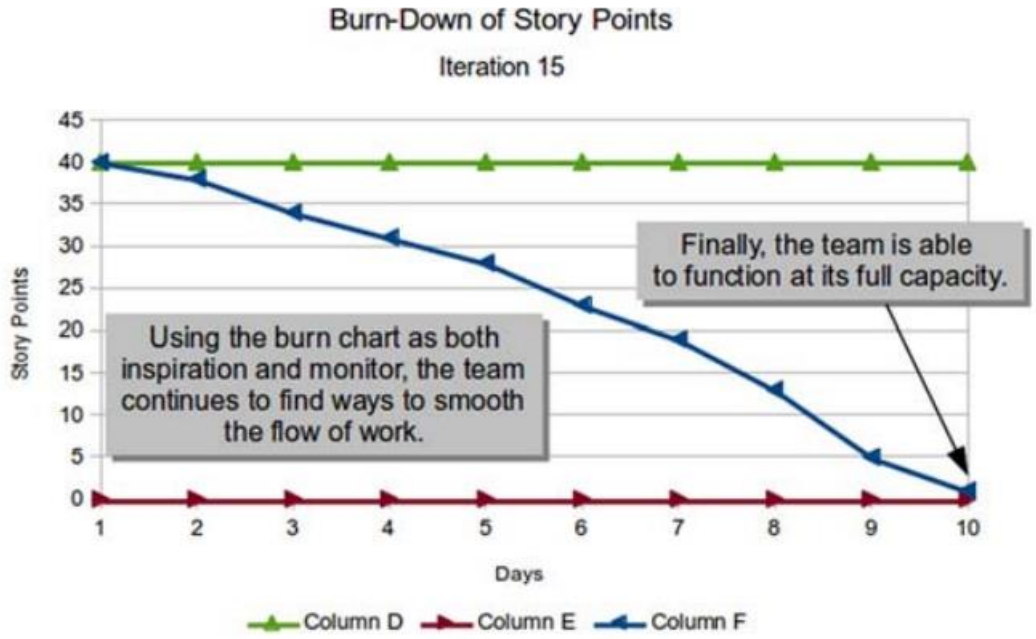
- Prethodni grafikon može da ukaže na sljedeće:
  - *The recurring pattern is causing stress, software defects, and incomplete work items. The stress leads to lower morale, lack of focus, and careless work; the defects create failure demand for non-value-add work for correction; the incomplete work items carry over into subsequent iterations, reducing the team's delivery capacity by pushing out planned work and by increasing the amount of context-switching overhead between planned work and unplanned defect correction; and the apparent inability of the team to deliver on a predictable schedule reduces stakeholder trust, which leads to increased oversight and administrative overhead, in turn feeding the cycle of slower delivery.*

Primjer (3)

- Mogući uzroci
  - Međusobna zavisnost story-ija
  - Preveliki broj WIP-a
  - Horizontalna umjesto vertikalna dekompozicija
  - Testiranje se dešava prekasno, moguće jer zavisi od spoljnjih faktora koji obezbjeđuju testne podatke ili okolinu
  - Low bus number
  - Single batch of work umjesto continuous flow

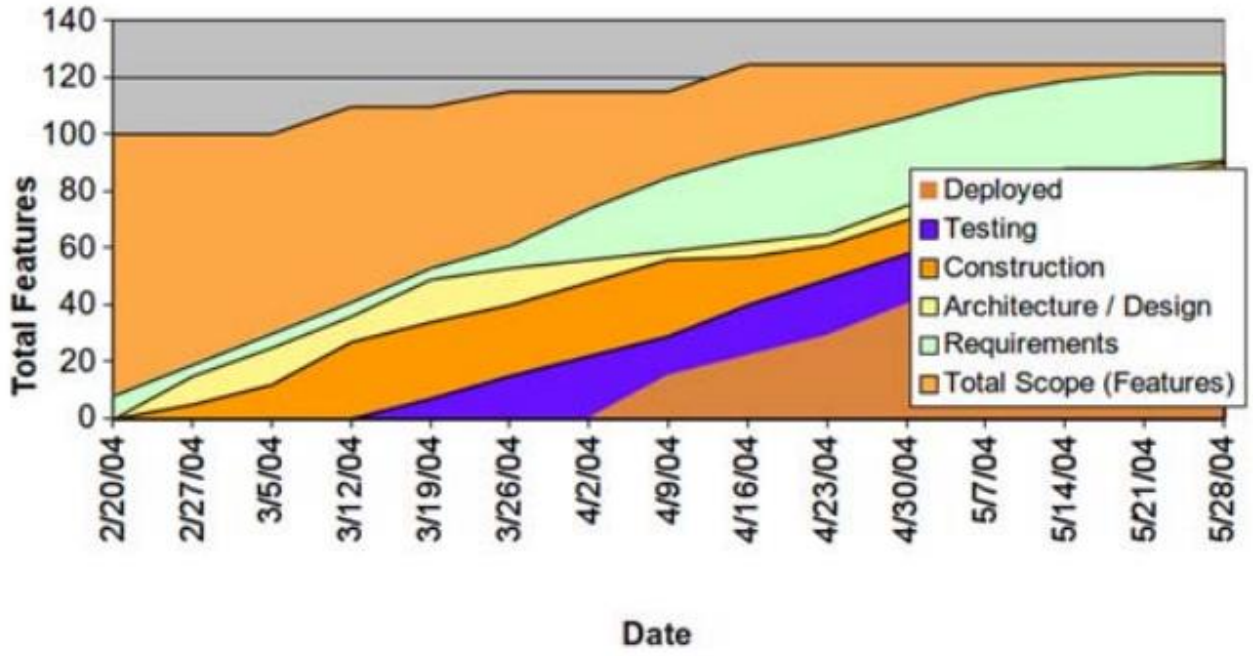
Primjer (4)


After three iterations, the team is able to pull user stories through the process earlier in the iteration.
Burn-Down of Story Points
Iteration 12
Opportunities remain to achieve smoother flow.
There is still pressure at the end of the iteration, but less than before, and the team delivers more completed user stories.
Story Points
Days
Column D
Column E
Column F


Burn-Down of Story Points
Iteration 15
Finally, the team is able to function at its full capacity.
Using the burn chart as both inspiration and monitor, the team continues to find ways to smooth the flow of work.
Story Points
Days
Column D
Column E
Column F

Cumulative flow

- Ukazuje na bottleneck mjesta
  - Work item-i prolaze kroz nekoliko faza
  - Regioni koji rastu po širini, Architecture / Design je bottleneck!



Cumulative flow (2)

- Moguće akcije su:
  - Ograničiti WIP u stanju Requirements
  - Ukloniti eventualne barijere koji usporavanju fazu Architecture / Design
  - Angažovati još ljudi