Agilne metodologije

XP, *Scrum*, *Kanban*

Principi softverskog inženjerstva, *Elektrotehnički fakultet Univerziteta u Beogradu*

Agilne metodologije

- 17 istraživača okupljenih kao „The Agile Alliance“ 2001. godine potpisuju **Agilni manifesto**.
- Agilni mentalitet ima 4 vrednosti koje ga razlikuju od tradicionalnih procesa softverskog razvoja:
- **Pojedinci i interakcija** imaju veću vrednost nego procesi i alati.
  - Projektom timu treba obezbediti neophodne resurse i imati poverenja da će oni poslove uraditi dobro.
  - Timovi se samoorganizuju i komuniciraju direktno (licem u lice), umesto posredstvom dokumentacije.
- Bolje uložiti vreme u izradu softvera, **softvera koji radi**, nego u izradu sveobuhvatne dokumentacije. Primarno merilo uspeha je stepen do koga softver radi ispravno.
- **Održavati kolaboraciju sa naručiocima**, uključiti ih u posao i ključne faze razvoja, umesto procesa ugovaranja i pregovaranja.
- **Odgovoriti na promene** koje se javljaju, umesto planirati i pratiti plan, jer je nemoguće sve zahteve predvideti na početku razvojnog procesa.

Agilni manifesto (1)

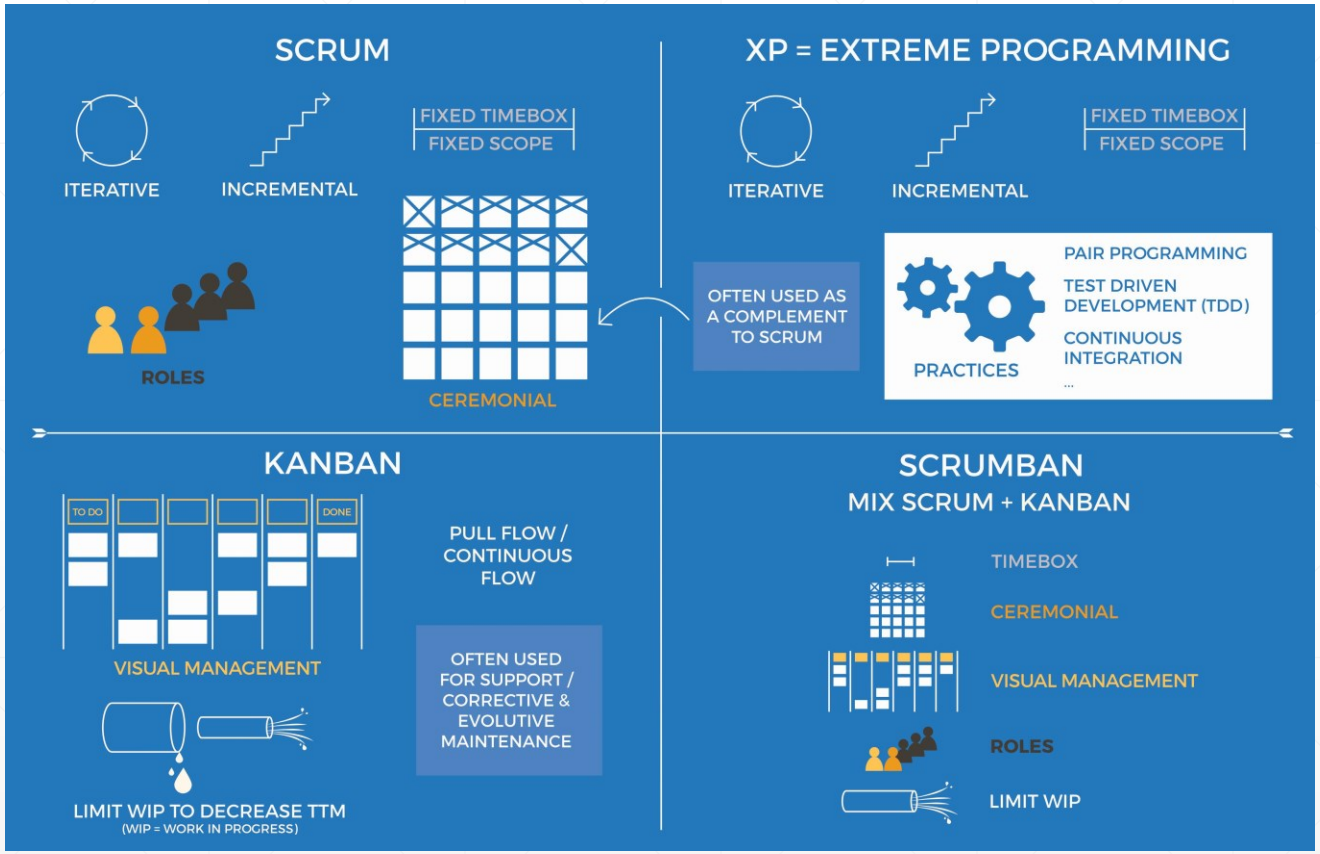
- 1) Naš najveći prioritet je da zadovoljimo kupca ranom ili kontinuiranom isporukom vrednog softvera.
- 2) Zahtevi koji se menjaju dobrodošli su čak i kasno u razvoju softvera.
- 3) Isporučujte softver relativno često, iz manjih delova,
  u periodima od nekoliko nedelja do nekoliko meseci.
- 4) Programeri, kao i svi učesnici projekta, treba da rade svakodnevno na projektu zajedno, tokom čitavog trajanja.
- 5) Izgraditi projekat oko motivisanih pojedinaca. Dajte im okruženje i podršku koja im je potrebna, i verujte im.
- 6) Najefikasnija i najefektnija razmena informacija ka i u okviru razvojnog tima je „licem u lice“.

Agilni manifesto (2)

- 7) Ispravan softver, u produkciji, glavna je mera napredovanja.
- 8) Agilni procesi promovišu održivi razvoj.
- 9) Stalna pažnja usmerena na tehničkoj izvrsnosti i dobrom dizajnu softvera povećava agilnost.
- 10) Jednostavnost, kao veština maksimizovanja količine nezavršenog posla, je od suštinske važnosti.
- 11) Najbolje softverske arhitekture, dizajn i zahtevi dolaze od timova koji su samoorganizovani.
- 12) U redovnim terminima, tim treba da razmišlja o tome kako da bude efikasniji, i da podešava i prilagođava svoje ponašanje u skladu sa tom analizom.

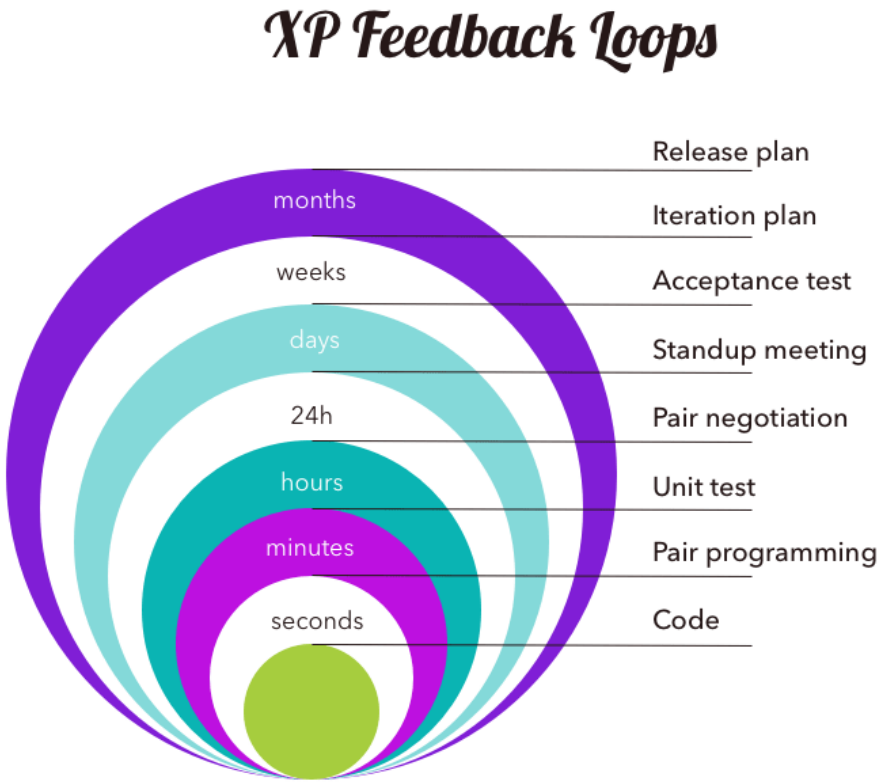
Vrste agilnih metodologija

- Predstavnici agilnih metodologija razvoja:
  - Ekstremno programiranje (XP)
  - Scrum
  - Kanban



Ekstremno programiranje - XP

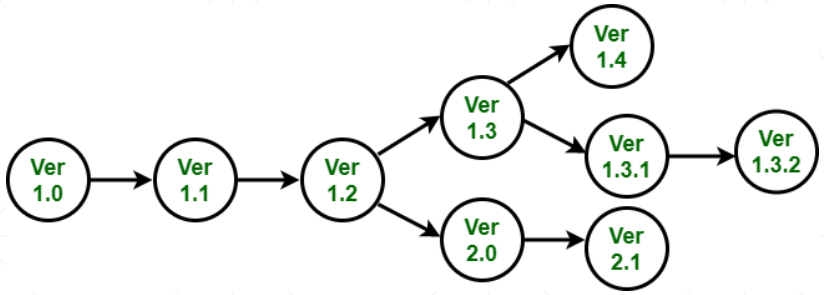
- Niz nekoliko važnih elemenata koji vode do uspešnog procesa i projekta.
- Izvući maksimum iz potencijala programera.
- Kruta pravila unutar tima krenula da se izostavljaju (npr. dnevni testovi integracije).
- Kritike menadžera: stalno prisutan klijent = stres
- Bez detaljnih zahteva i sa dizajniranjem softvera „u letu“ možemo imati veoma neefikasan projekat.
- Tvorac Ekstremnog programiranja (XP) *Kent Beck*, jedan od 17 potpisnika Agilnog manifesta, pionir u dizajnu softvera (*design patterns*) i tvorac pristupa Test Driven Development (TDD). Razvio je xUnit seriju radnih okvira za testove, iz kojih je proizašao i JUnit (2004).



Izvor: http://www.extremeprogramming.org

XP - osnovni koncepti (1)

- Igra planiranja:
  - Naručilac definiše svaki zahtev prema vrednosti koju njegova implementacija donosi sistemu.
  - Korisnici pišu scenarije o tome kako bi sistem treba da radi, a projektni tim procenjuje neophodne resurse za realizaciju.
  - Nakon pisanja scenarija, **potencijalni korisnici dodeljuju prioritete zahtevima, razdvajaju ih i spajaju**, sve dok se ne postigne konsenzus o tome šta je stvarno potrebno, šta može da se testira, i šta može da se uradi sa raspoloživim resursima.
  - **Planeri generišu mapu svake verzije**, dokumentujući šta ta verzija sadrži i kada će biti isporučena.



Kod softvera: v.1.3.2.4
1 – Major (velike izmene)
3 – Minor (manje izmene)
2 – Build / Patch (fiksiranje bagova)
4 – Revizija (opciona)

Nekad pre revizije postoji i naziv: Beta, ili Prerelease, npr.
v. 2.3.4-beta

XP - osnovni koncepti (2)

- Male verzije:
  - Sistem se projektuje tako da funkcionalnost može da se isporuči što je pre moguće.
  - Funkcije su dekomponovane na male delove,
    sa ciljem da se neka funkcionalnost isporuči u ranoj fazi,
    a zatim poboljšava ili proširuje u kasnijim verzijama.
  - **Male verzije zahtevaju fazni pristup razvoju**,
    sa inkrementalnim i iterativnim ciklusima.

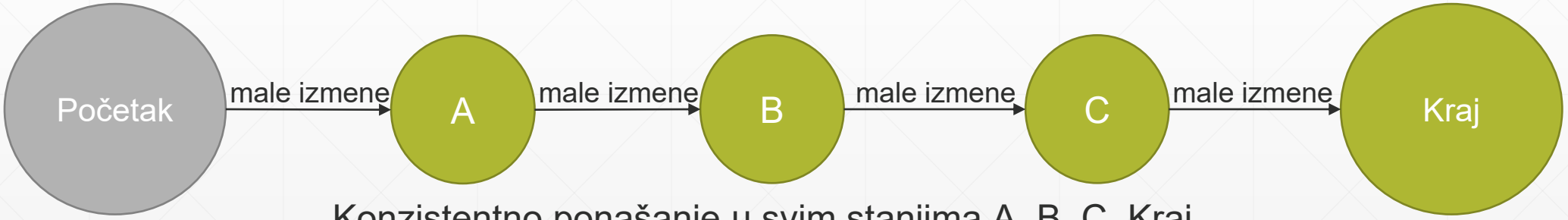
- Metafora:
  - Projektni tim se usaglašava oko zajedničke vizije načina na koji će sistem raditi.
    Kao podršku zajedničkoj viziji tim bira **zajedničku nomenklaturu** i sporazumeva se
    oko zajedničkog načina tretiranja ključnih pitanja.

XP - osnovni koncepti (3)

- Jednostavan dizajn:
  - **Samo aktuelne potrebe realizovati**.
    Predviđanje potencijalnih budućih potreba dovodi do nepotrebnih funkcija.
- Pisanje testova pre kodiranja (Razvoj vođen testovima, eng. TDD):
  - Kako bi potrebe naručioca bile glavni pokretač razvoja, **prvo se pišu scenariji testova**, koji prikazuju zahteve naručioca, i po izradi softvera se odmah može testirati i verifikovati.
  - Koriste se 2 vrste testova:
    - Funkcionalni testovi: određuje ih naručilac, a sprovode projektni tim i korisnici. Oni se smatraju delom funkcionalne specifikacije sistema.
    - Testovi delova koje piše i izvršava razvojni tim. Testovi delova se pišu i pre i posle kodiranja, da bi se verifikovalo da svaki modul implementacije radi prema očekivanjima.
  - Funkcionalni testovi su automatizovani, i izvršavaju se svakog dana (u idealnom slučaju).

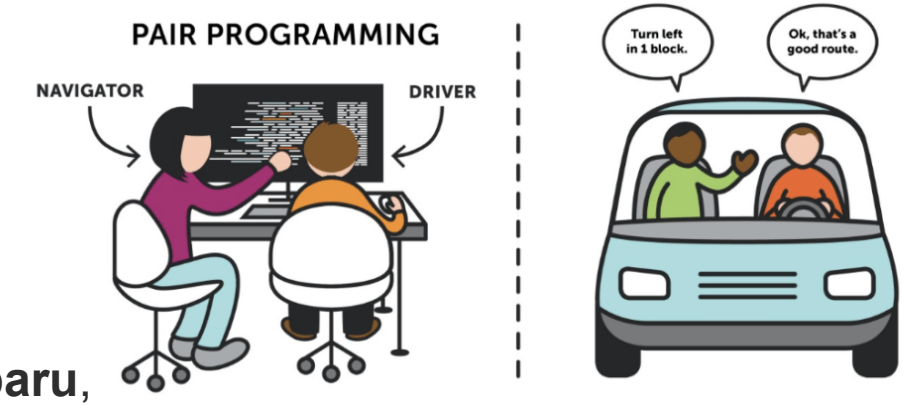
XP - osnovni koncepti (4)

- Refaktorisanje (preuređivanje koda):
  - Tokom izgradnje sistema, postoji verovatnoća da dođe do promene zahteva.
  - Ovaj koncept se odnosi na preispitivanje zahteva i dizajna, i njihovo preformulisanje u skladu sa novim i postojećim potrebama.
  - Nekada se bavi prestrukturisanjem programskog koda, bez izmena spoljašnjeg ponašanja sistema.
  - Refaktorisanje se radi **u malim koracima, uz testiranje delova**, i programiranje u paru, sa jednostavnošću kao idejom vodijom.



Konzistentno ponašanje u svim stanjima A, B, C, Kraj

XP - osnovni koncepti (5)



- Programiranje u paru (eng. *Pair Programming*):
  - Koristeći jednu tastaturu, **dva programera**, koji **rade u paru**, razvijaju sistem na osnovu specifikacije i dizajna. Jedna osoba je odgovorna za kodiranje.
  - Fleksibilnost kod rada u paru je da jedan programer može imati više partnera u istom danu.
- Kolektivna svojina (eng. *Collective Code Ownership*):
  - U fazi razvoja, svaki učesnik razvojnog tima može da **izmeni bilo koji deo sistema**, naravno uz obaveštenje drugima šta je uradio. Najčešće se softverski sistemi razvijaju na razvojnim (ili testnim) serverima, pa se tek na kraju stavljaju na produkcioni server.
- Neprekidna integracija (eng. *Continuous Integration*):
  - Brza isporuka funkcionalnosti - može da se obeća naručiocu **brza promena u sistemu**, u roku od jednog dana ili čak jednog sata.
  - Među revizije: mali inkrementi i poboljšanja, nema velikih skokova između revizija.

XP - osnovni koncepti (6)

- Održiv tempo (i bez prekovremenog posla):
  - Umor uvek proizvede greške! Zato je XP uveo sugestiju da je **cilj 40 radnih časova nedeljno**.
  - Ako projektni tim troši ogromno vreme da bi sustigao rokove, to je siguran znak da su **rokovi nerealni**, ili da postoji nedostatak resursa neophodnih za realizaciju svih zahteva.
- Naručilac raspoloživ na terenu:
  - **Naručilac** treba da **radi** sa razvojnim timom **na definisanju zahteva** i obezbeđivanju povratnih informacija o tome kako treba da se vrši testiranje.
- Standardi kodiranja (eng. *Coding Standard*):
  - **Jasno definisanje standarda kodiranja**, sa ciljem da se članovi osposobe da razumeju neophodne izmene u produktima rada drugih članova tima. Rezultat treba da bude kod koji izgleda **kao da ga je pisala jedna osoba**, konzistenta sa aspekta pristupa i izražavanja.

Neke glavne odlike standarda kodiranja

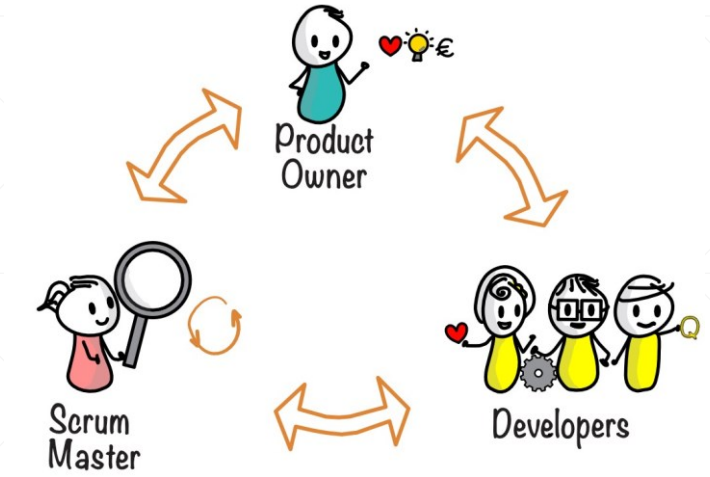
- Ograničena upotreba globalnih podataka
- Standardna zaglavlja za različite module (naziv, datum, autor, istorija izmena,...)
- Konvencije imenovanja za lokalne varijable, globalne varijable, konstante i funkcije (npr. localVar, GlobalVar, KONSTANTA)
  - *Snake case* (*underscore*) – primer: user_id, user_name, moja_funkcija (tipično za Python/PHP/Ruby)
  - *Camel case* – primer: userId, username, mojaFunkcija (tipično za Java / C# / JavaScript / Go)
- Pravilno uvlačenje koda zbog povećana čitljivosti.
- Vraćanje vrednosti greške i konvencija o rukovanju izuzecima.
- Izbegavati stilove kodiranja koji su previše teški za razumevanje. Pokušajte izbeći i GOTO.
- Programski kod lepo dokumentovati pisanjem komentara koji olakšavaju razumevanje.

SCRUM

Radni okvir za upravljanje projektima

Šta je SCRUM?

- ***Scrum*** je okvir za razvoj i održavanje složenih (softverskih) proizvoda.
- Bazira se na ideji empirijske kontrole procesa ili empirizma. Empirizam tvrdi da znanje dolazi iz iskustva i donošenja odluka na osnovu onoga što je poznato.
- Koristi iterativno-inkrementalni pristup sa ciljem povećanja izvesnosti i upravljanja rizicima.
- Okvir čine *Scrum* timovi i odgovarajuće uloge, događaji, artefakti i pravila.



ULOGE:

- **Scrum master (moderator timskog rada)**
  - obezbeđuje sve što je potrebno timu, priprema sastanke i nadgleda proces.
- **Product owner (vlasnik proizvoda)**
  - reprezentuje klijenta i odgovara za sadržaj i prioritete u product backlog-u.
- **Dev team (razvojni tim)**
  - grupa ljudi sa različitim specijalnostima (programeri, testeri) koji rade konkretne aktivnosti analize, dizajna, implementacije, testiranje itd.

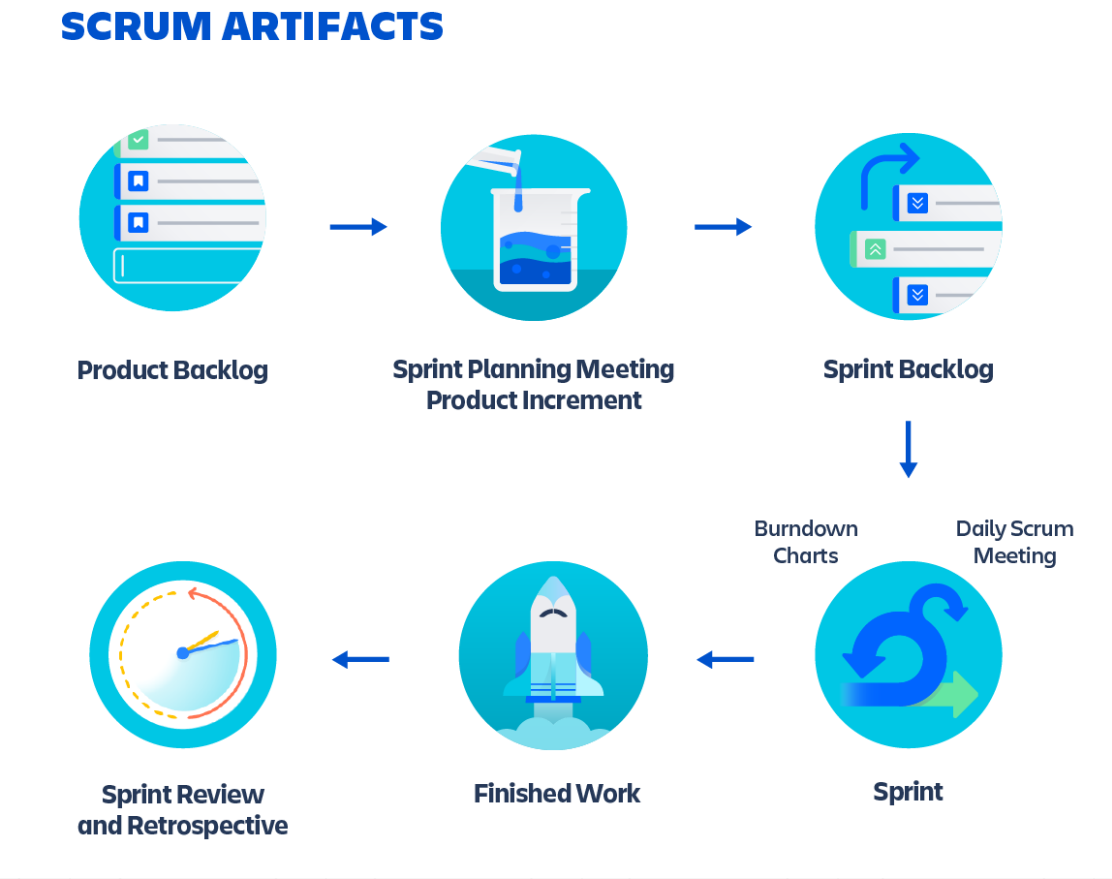
Glavni proizvodi (artefakti) u *SCRUM* okviru

Osnovni:

- Korisnička priča (*User story*) i zadatak (*Task*)
- *Product backlog* (preostali poslovi proizvoda)
- *Sprint backlog* (preostali poslovi u sprintu)
- *Product increment* (inkrement proizvoda)

Dodaci (*Extensions*):

- *Burndown* (grafikon preostalog posla)



Šta je Sprint?

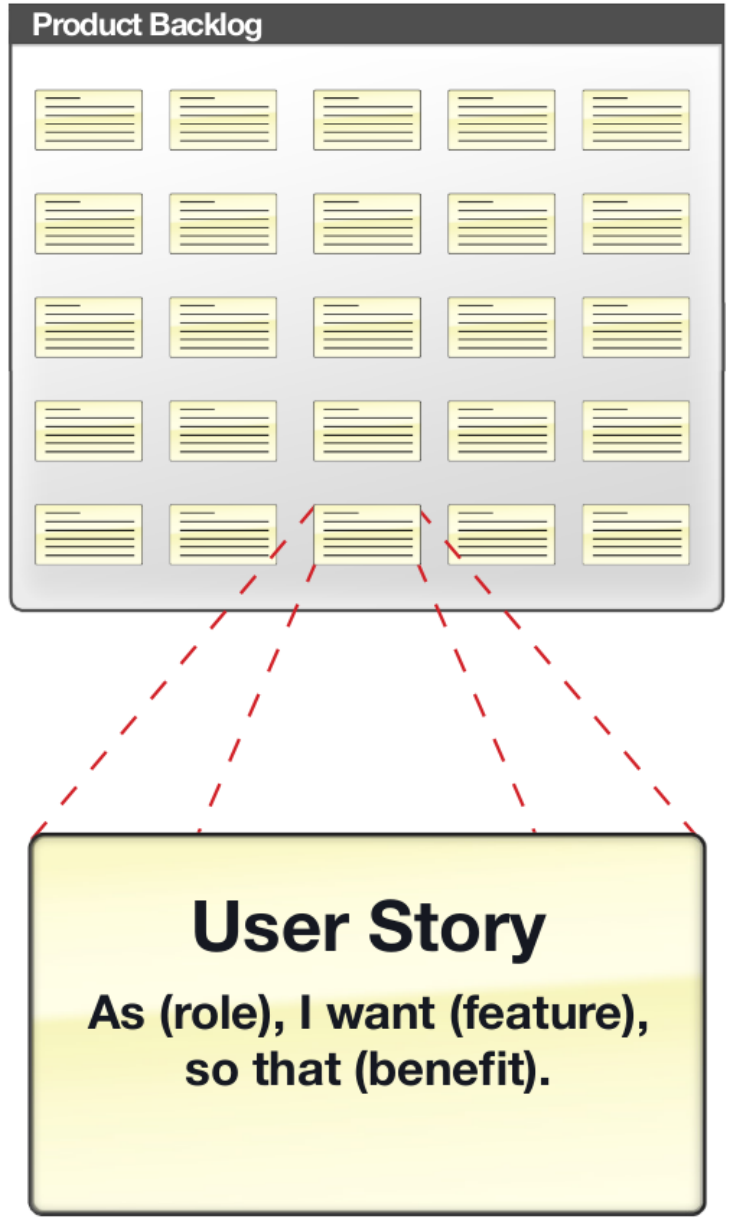
- Centralni koncept Scrum je Sprint, vremenski ograničen na mesec dana ili kraći vremenski period, tokom kojeg se stvara upotrebljiv, potencijalno isporučiv inkrement proizvoda.
- Svaki Sprint uobičajno ima konstantno trajanje tokom procesa razvoja (*time-boxed*).
- Novi Sprint počinje odmah po zaključivanju prethodnog.
- Sprint se sastoji od sledećih događaja i aktivnosti:
  - *Sprint Planning* (Sastanak planiranja Sprinta),
  - *Daily Scrum* (Dnevni Scrum ili brifing),
  - Razvoj proizvoda,
  - *Sprint Review* (Pregled Sprinta) i
  - *Sprint Retrospective* (Osvrt na Sprint).

Scrum proces

Izvor: Adapted from Agile Software Development with Scrum, by K.Schwaber, M.Beedle

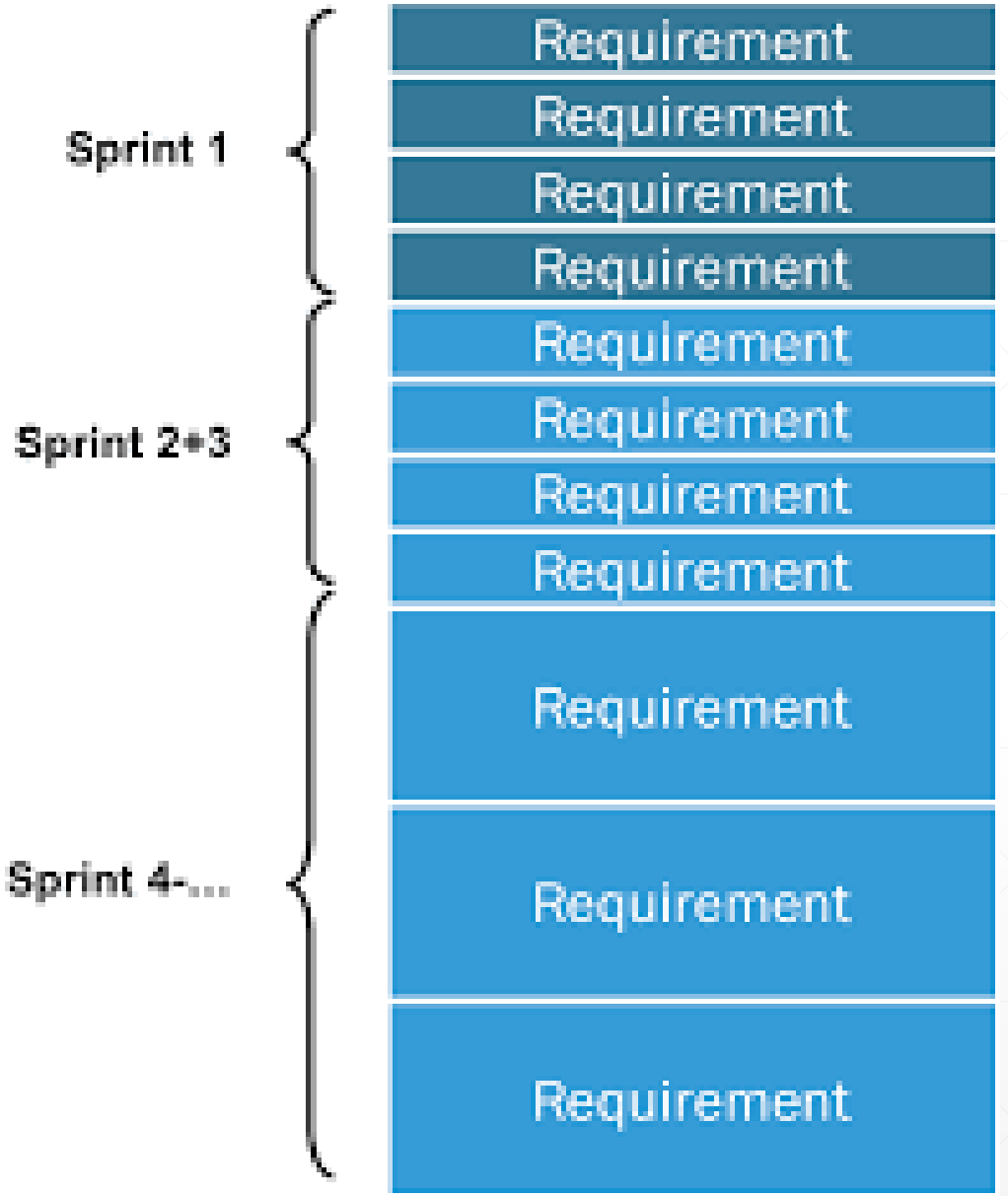
Proizvodni backlog (PB)

- PB je lista svih zahteva, funkcionalnosti, poboljšanja i popravki koje zajedno čine promene koje treba da se implementiraju u budućim inkrementima proizvoda.
- PB stavke imaju naziv i opis, redosled, procenu uloženog napora i (poslovnu) vrednost.
- Stavke su u obliku korisničkih priča (*user stories*).
- Za PB odgovoran je vlasnik proizvoda (*Product Owner*).
- Poželjno konstruisati RBS (*Requirements Breakdown Structure*).



Stavke kod PB

- Karakteristike softvera (*Features*), prikazuju se:
  - Korisničke priče (*User stories*)
  - Dijagrami slučajeva korišćenja (*Use cases*)
  - Scenario slučaja upotrebe (slobodnog teksta)
- Defekti u softveru, koje treba da ispravimo
- Tehnički rad (*Technical work*)
  - npr. instalirati neki alat, podesiti okruženja, poboljšati neke performanse unutar softvera, sigurnost, i sl.
- Sticanje znanja (*Knowledge acquisition*)
  - npr. učiti nove tehnologije i alate, učiti neku metodologiju

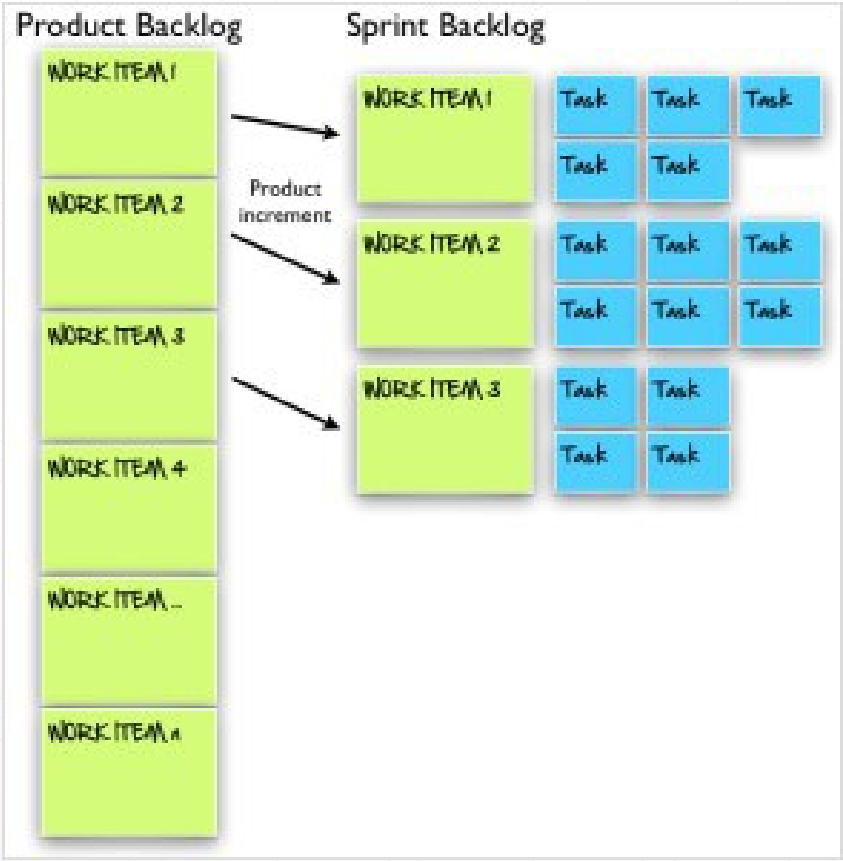


Važne karakteristike kod PB

- DEEP pravilo: Detailed appropriately + Emergent + Estimated + Prioritized.
- Detaljnost, koliko je potrebna
  - Stavke u PB se razlikuju u nivou detalja (one koje radimo ranije, i koje su prioritetnije su i detaljnije, one na kojima ne radimo biće sa manje detalja)
- Hitnost
  - PB je živ dokument, koji se neprestano menja.
  - Zdrav PB treba da ima dodate nove stvari, neke odbačene ili promenjene.
- Procenjenost
  - U određenom trenutku imati procenu veličine PB koja odgovara uloženom naporu potrebnom za razvoj tog proizvoda
  - Velike stavke visokog prioriteta razbijati u manje priče, pre nego što ih proglasimo spremnim za Sprint
- Prioritetnost - davati prioritete prema izdanju (verziji)

Sprint backlog (SB)

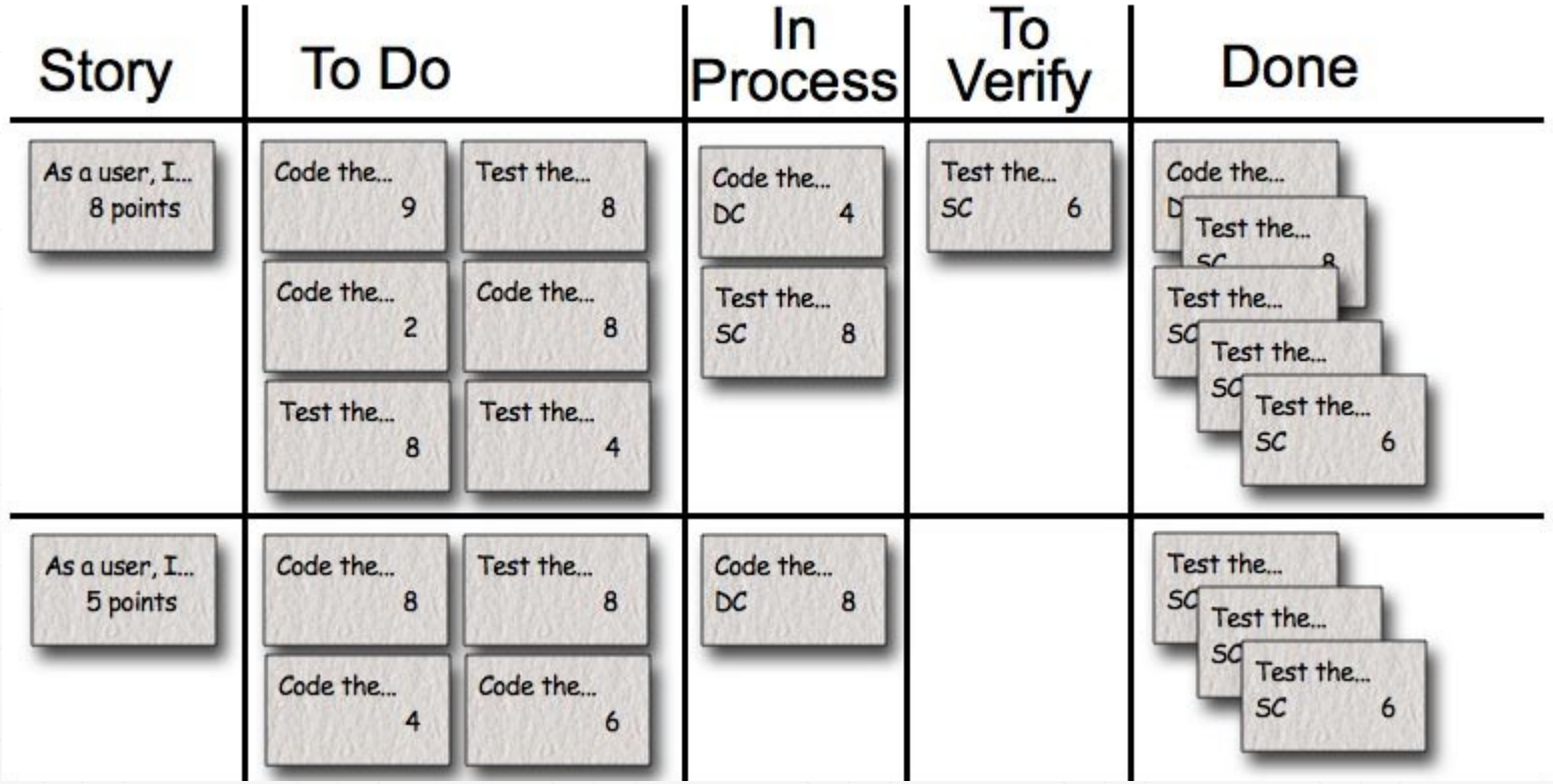
- SB je lista poslova (stavki) koje razvojni tim mora uraditi tokom sledećeg *Sprint*-a. Lista je izvedena izborom onoliko stavki sa vrha PB, koliko razvojni tim smatra da ima dovoljno posla da popuni *Sprint*.
- SB stavke razvojni tim deli na zadatke.
- Zadaci se nikada ne dodeljuju, već ih preuzimaju pojedini članovi tima u skladu sa postavljenim prioritetima i veštinama članova razvojnog tima. Ovo promoviše samoorganizovanje razvojnog tima.



Izvor: https://www.nutcache.com/blog/a-nerds-guide-on-sprint-backlog-and-product-backlog/

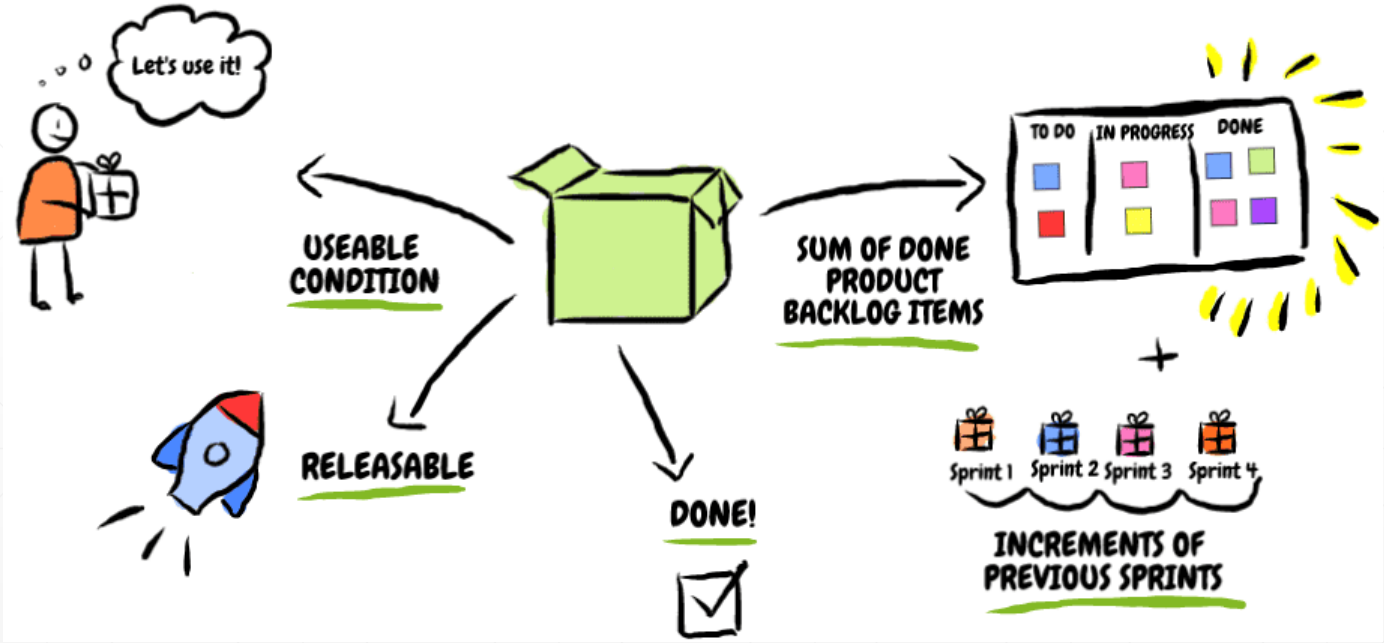
Sprint backlog - prikaz

- Često se za predstavljanje zadataka i njihovog statusa (*To do*, *In progress*, *To Verify*, *Done*,…) koristi **tabla sa zadacima** (eng. *Task board*).



Inkrement

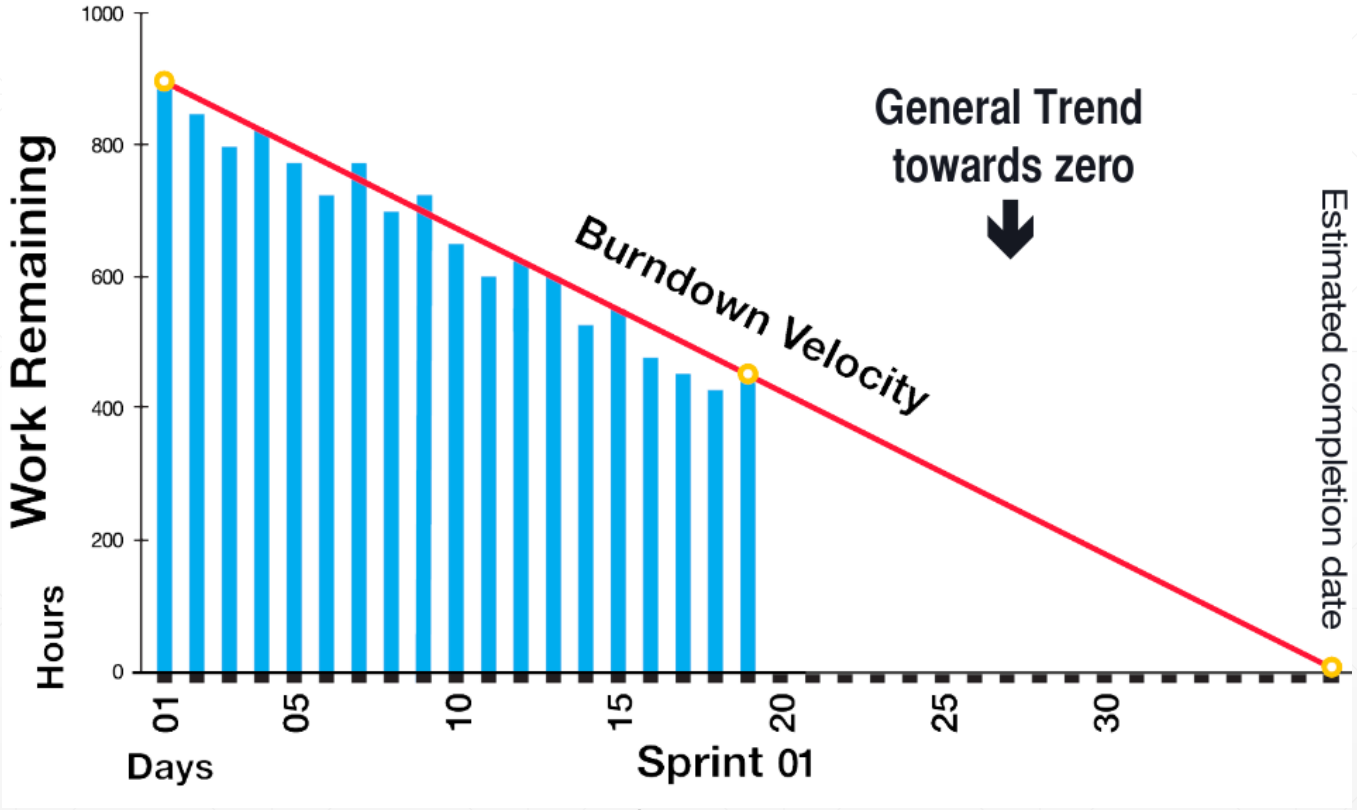
- Inkrement je skup svih stavki iz PB, koje su završene tokom tekućeg i svih prethodnih *Sprint*-ova.
- Na kraju *Sprint*-a, novi inkrement mora biti završen što znači da mora biti u upotrebljivom stanju i da ispunjava definiciju završenoga (eng. *Definition of Done*) tog Scrum tima.
- Inkrement mora biti upotrebljiv bez obzira na to da li će ga vlasnik proizvoda pustiti u upotrebu ili ne.



Izvor: https://letsscrumit.com/scrum-artifacts-3-the-increment

Burndown - grafikon preostalog posla

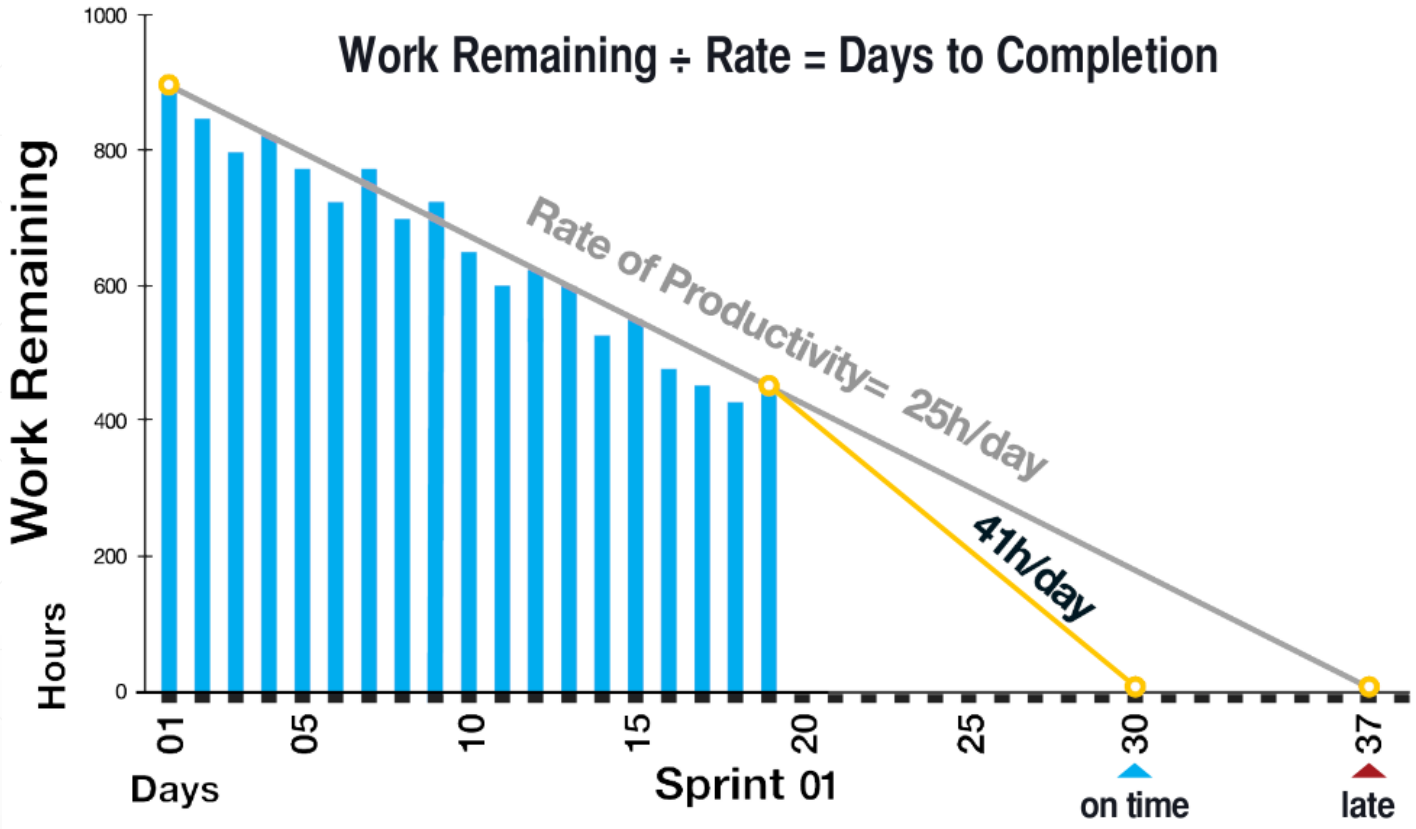
- Napredak tima se prati korišćenjem grafikona preostalog posla u vremenu.
- Na X-osi je vreme,
  na Y-osi je procena preostalog posla
  u tekućem *Sprint*-u.



Izvor: https://www.scrumhub.com/scrum-guide/burndowns/

Brzina

Izvor: https://www.scrumhub.com/scrum-guide/burndowns/

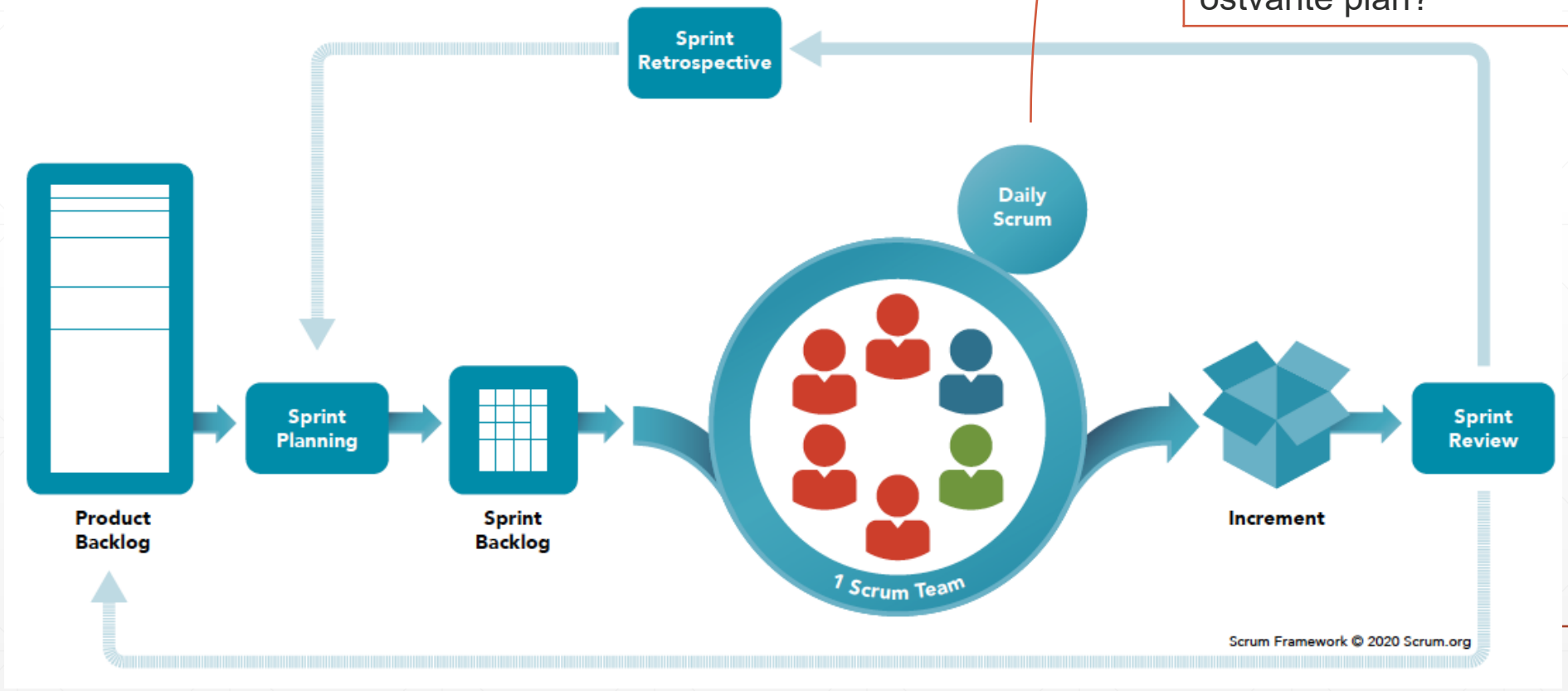


- Nagib grafikona preostalog posla, naziva se brzina (eng. *Velocity*) i predstavlja prosečnu produktivnost po danu.
- Informacija da li projekat ide po planu rano u toku Sprinta, može pomoći da se naprave doterivanja koja će omogućiti uklapanje u raspored.

Šta je još u Scrum procesu?

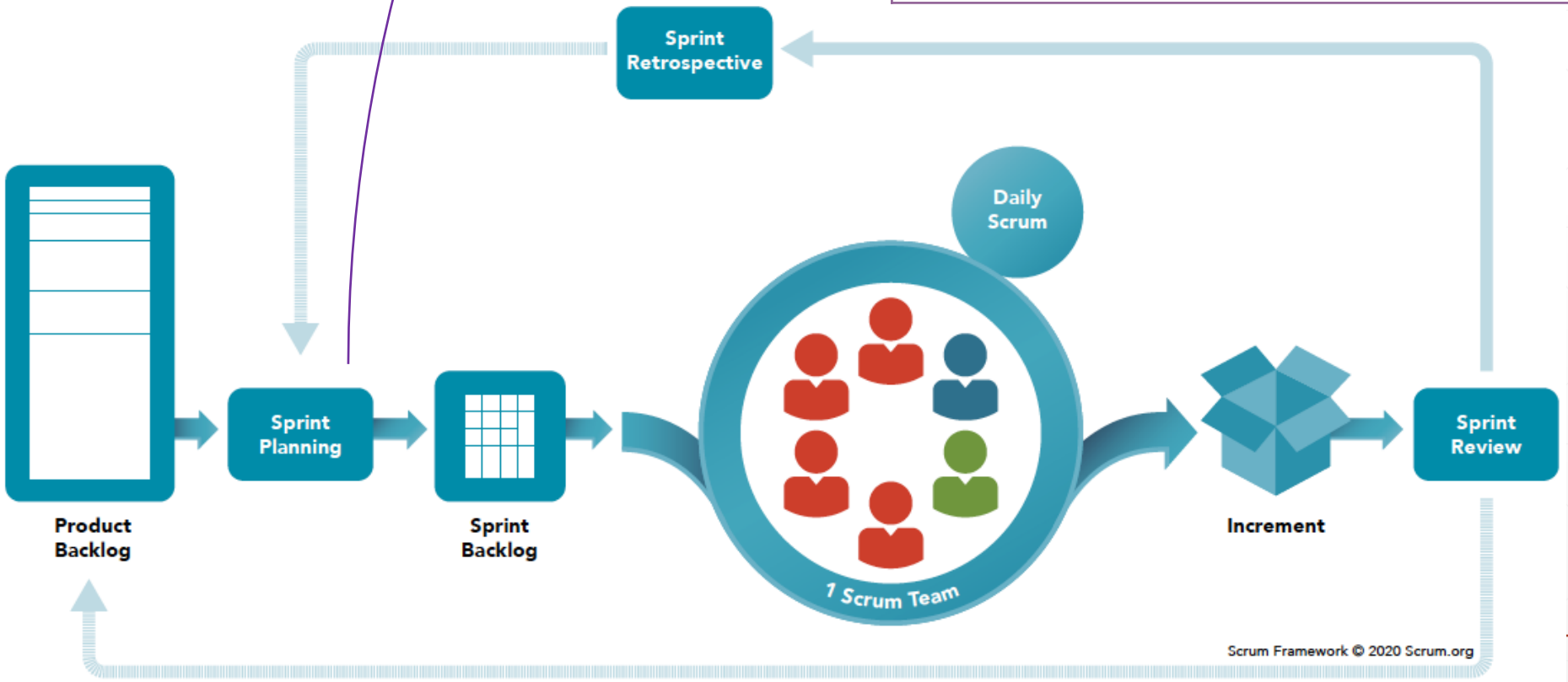
Da pojasnimo vezu između artefakata i događaja u Scrum-u.

Daily Scrum
Dnevni 15-minutni sastanci. Uglavnom svi stoje.
1. Šta ste uradili od juče?
2. Šta planirate za danas da uradite?
3. Ima li problema koji vas sprečavaju da ostvarite plan?



Sprint Planning (Sastanak planiranja)

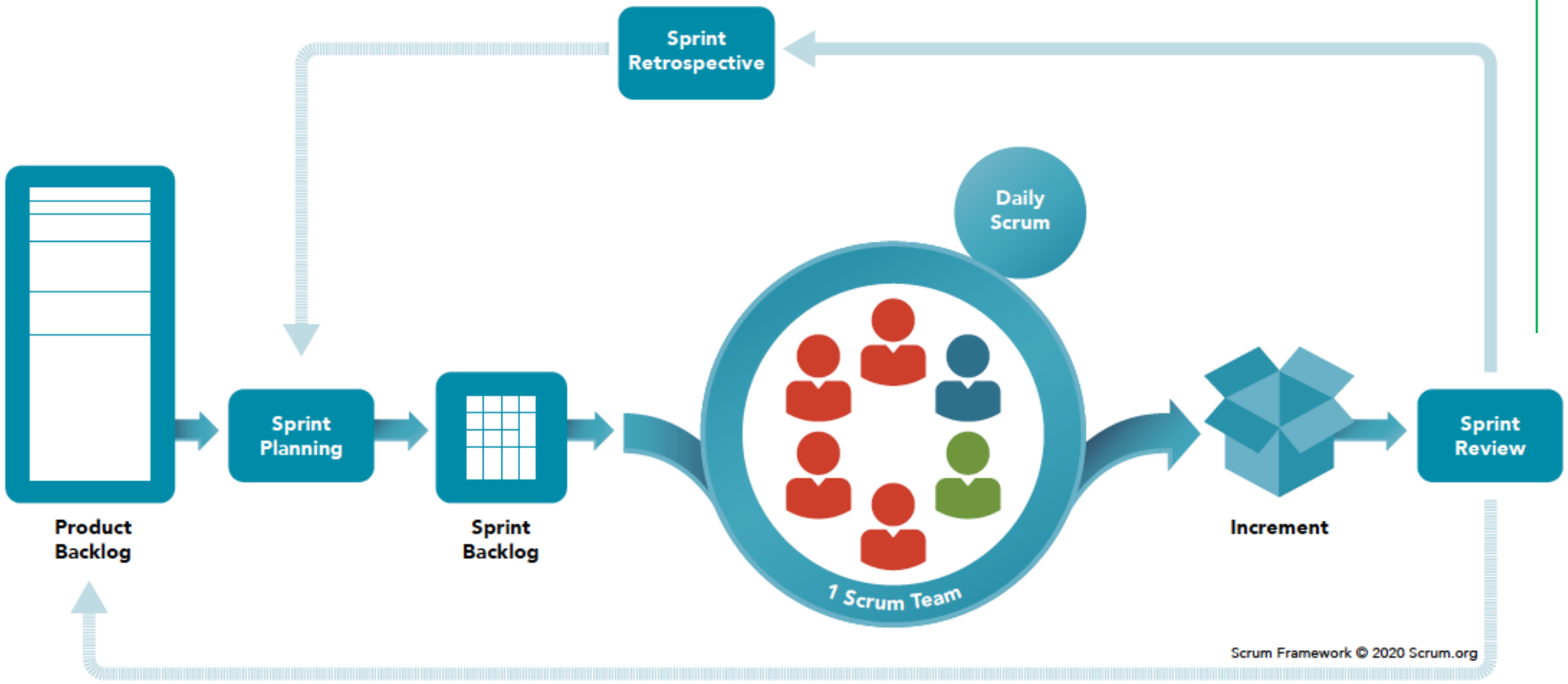
Planira se sledećih 15-30 dana (jedan Sprint).
Biraju se poslovi koji će da se rade (tj. šta se može isporučiti sa inkrementom koji je rezultat ovog Sprint-a)
Priprema se Sprint Backlog, koji elaborira vreme koje će trebati da se urade ti poslovi, učestvuje ceo tim.
Limit sastanka je 8 sati.



Sprint Review (pregledni sastanak Sprinta)

Pregled šta je od posla završeno,
a šta nije (korigovati *Product backlog* po potrebi)

Prezentacija završenog posla zainteresovanim stranama ("demo")



Ovaj sastanak najčešće limitiran na 4h.

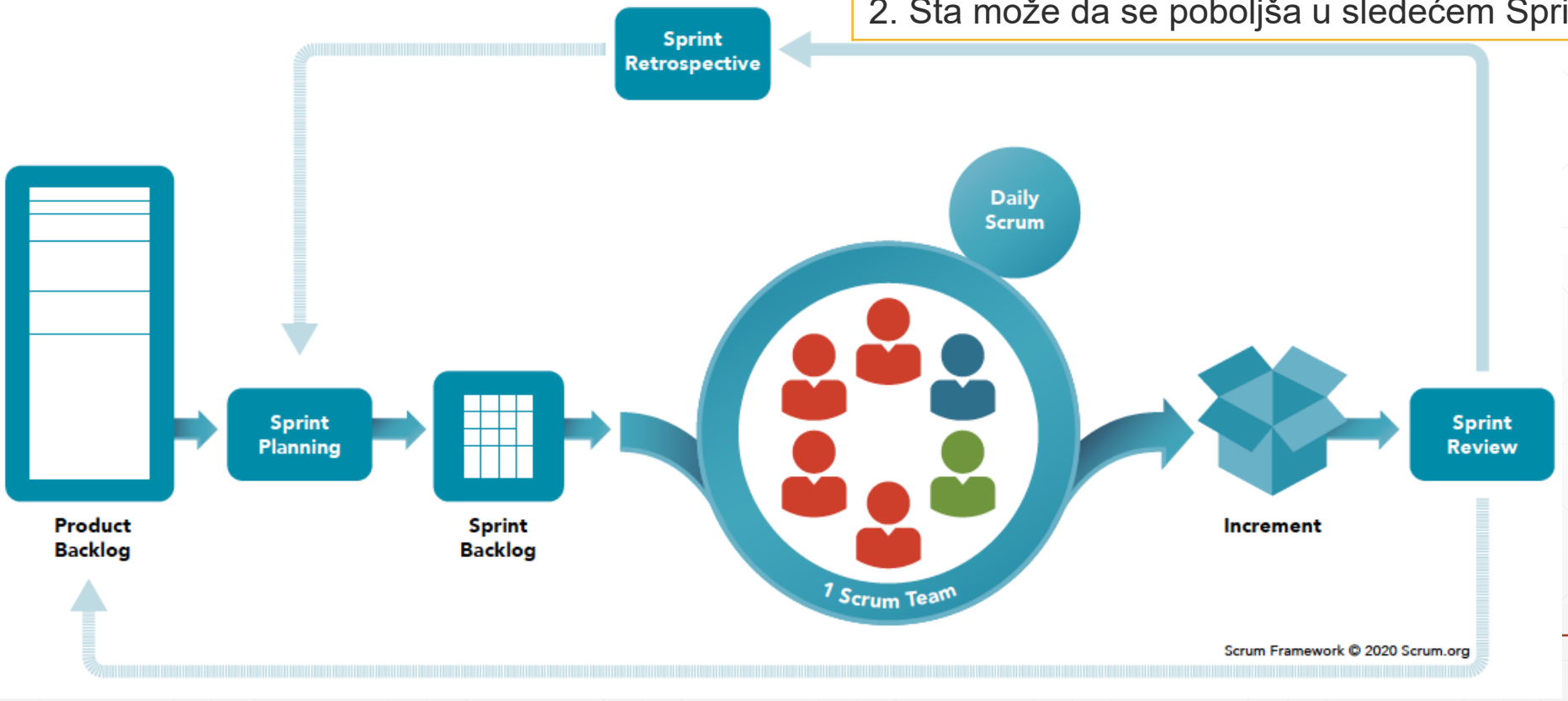
Sprint Retrospective (osvrt na Sprint)

Osvrt na sprint se održava nakon svakog *Sprint Review* sastanka, a pre narednog planiranja *Sprint*-a.

Svi članovi tima daju osvrt na protekli *Sprint*.

Svrha je poboljšanje procesa u smislu odnosa ljudi, procesa, alata.

Dva glavna pitanja tokom osvrta:
1. Šta se dobro odvijalo tokom Sprint-a?
2. Šta može da se poboljša u sledećem Sprint-u?



Ovaj sastanak najčešće limitiran na 3h.

Kanban

popularni radni okvir za implementaciju agilnog i DevOps razvoja softvera

Kanban okvir

- Kanban (japn. „oglasna tabla“) je „pravovremena” (*just-in-time*) metodologija za kontrolu logistike u proizvodnom lancu (nastala iz Lean proizvodnje, u korporaciji TOYOTA, 1988. godine).
- Cilj vizuelizovati tok rada (upravljanje radom).
- Osnovna ideja je da se koriste „kartice“ na svakoj stanici u lancu da bi se predstavio inventar delova na toj stanici. Kada nekoj stanici ponestane delova, njen Kanban se šalje dobavljaču kako bi ukazao na potrebu za dopunom.
- U osnovi, Kanban deluje kao poruka za dostavu novih delova od dobavljača po potrebi.
  U praksi se može zadržati mala rezervna grupu delova, dok se čekaju novi delovi (analogija oznake za vožnju na „rezervi” kod automobila).
- Kanban metodologija se primenjuje i na organizaciju razvoja softvera.

Principi Kanban okvira

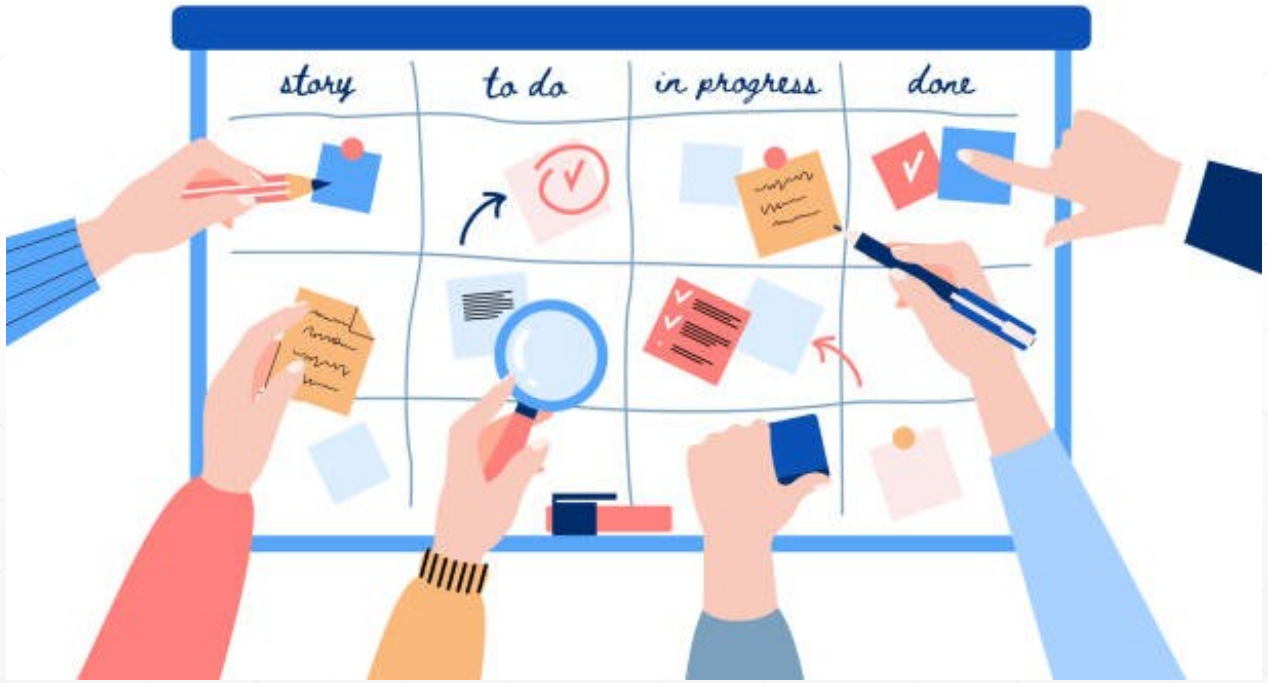
- Poput mnogih agilnih metodologija, Kanban ima skup polaznih principa osmišljenih da stvore zdravo i produktivno radno mesto:
  - **Počinje se sa trenutnom praksom** - Kanban ne diktira eksplicitno svaki deo razvojnog procesa, pa se može uvesti u bilo koji sistem, koji se trenutno koristi.
  - **Uvode se inkrementalne promene** - postepeno dodavanje Kanban principa trenutnim praksama. Pokušaj da se pređe na Kanban odjednom, može se suočiti sa otporom promenama.
  - **Poštuje se postojeći proces** - postojeći sistem verovatno ima neke korisne strane, tako da ga ne bi trebalo odbaciti čak ni ako to može da se uradi.
    Zadržavaju se postojeće uloge i odgovornosti. Kanban sam po sebi ne definiše eksplicitno nijednu ulogu. Vremenom uloge mogu da pretrpe promene (samoorganizovanjem tima).
  - **Podstiče se inicijativa kod svih** - kao i mnogi drugi agilni modeli, Kanban ohrabruje svakog da preuzme vlasništvo nad projektom i svojim zaduženjima. Ne treba čekati da neko drugi odluči da se uhvati u koštac sa zadatkom ili da ispravi grešku. Pokažite inicijativu i uradite to sami.

Kanban prakse

- Namera Kanbana je da se uklopi u postojeće prakse, ali definiše i neke specifične:
- **Vizuelizacija radnog toka** - Kanban omogućava da se vizuelizuje posao koji se radi trenutno u kontekstu drugih zadataka. Ovo ohrabruje slobodnu komunikaciju i saradnju članova tima.
- **Ograničavanje započetog posla** (*Work in progress*) - pravovremena priroda proizvodne linije znači da proizvodni lanac ima što je moguće manje zalihe. U softverskom inženjerstvu, Kanban ograničava količinu posla koji se obavlja u svakom trenutku. => **Uspostavite idealan kapacitet, i održavajte ga!**
- Nasuprot tome, u *Scrum* okviru se može desiti da se povuče previše ili premalo posla u *Sprint*. Smanjujući zadatke koji se rade u paraleli, Kanban uklanja inherentni gubitak vremena pri prebacivanju sa zadatka na zadatak i čini programere produktivnijim.
- **Poboljšanje protoka** - kada pojedinac završi jedan zadatak, povlači sledeću stavku najvišeg prioriteta iz preostalih zadataka. Umesto da koristi *Sprint*-ove koji uključuju planiranje, dizajniranje, procenu i testiranje skupa karakteristika, ti isti koraci se izvode za svaku pojedinačnu stavku kada se dođe do nje.

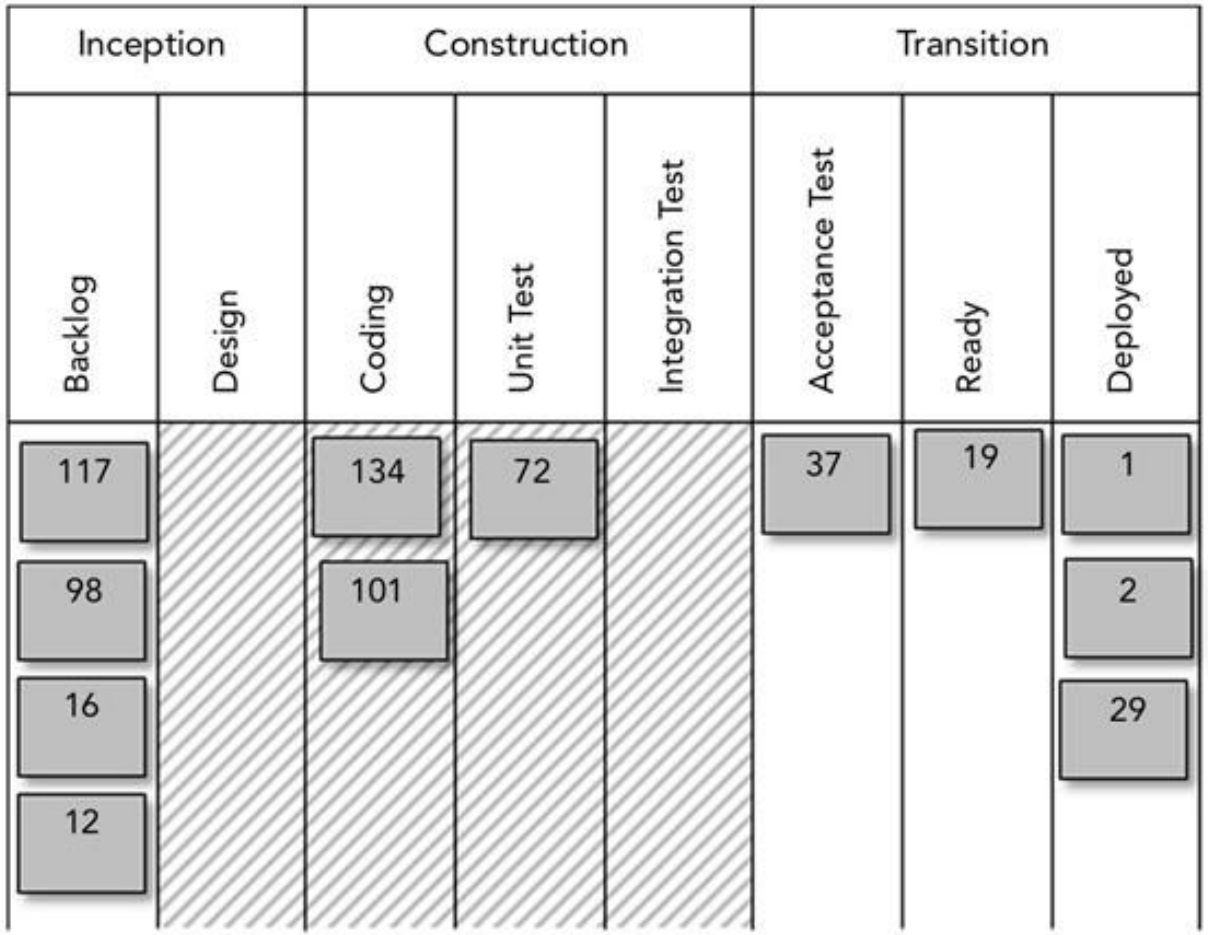
Kanban tabla kod softverskih projekata

- Kanban tabla (eng. *Kanban board*) je glavna karakteristika Kanban okvira.
- Tabla prikazuje sve zadatke projekta i njihove trenutne statuse.



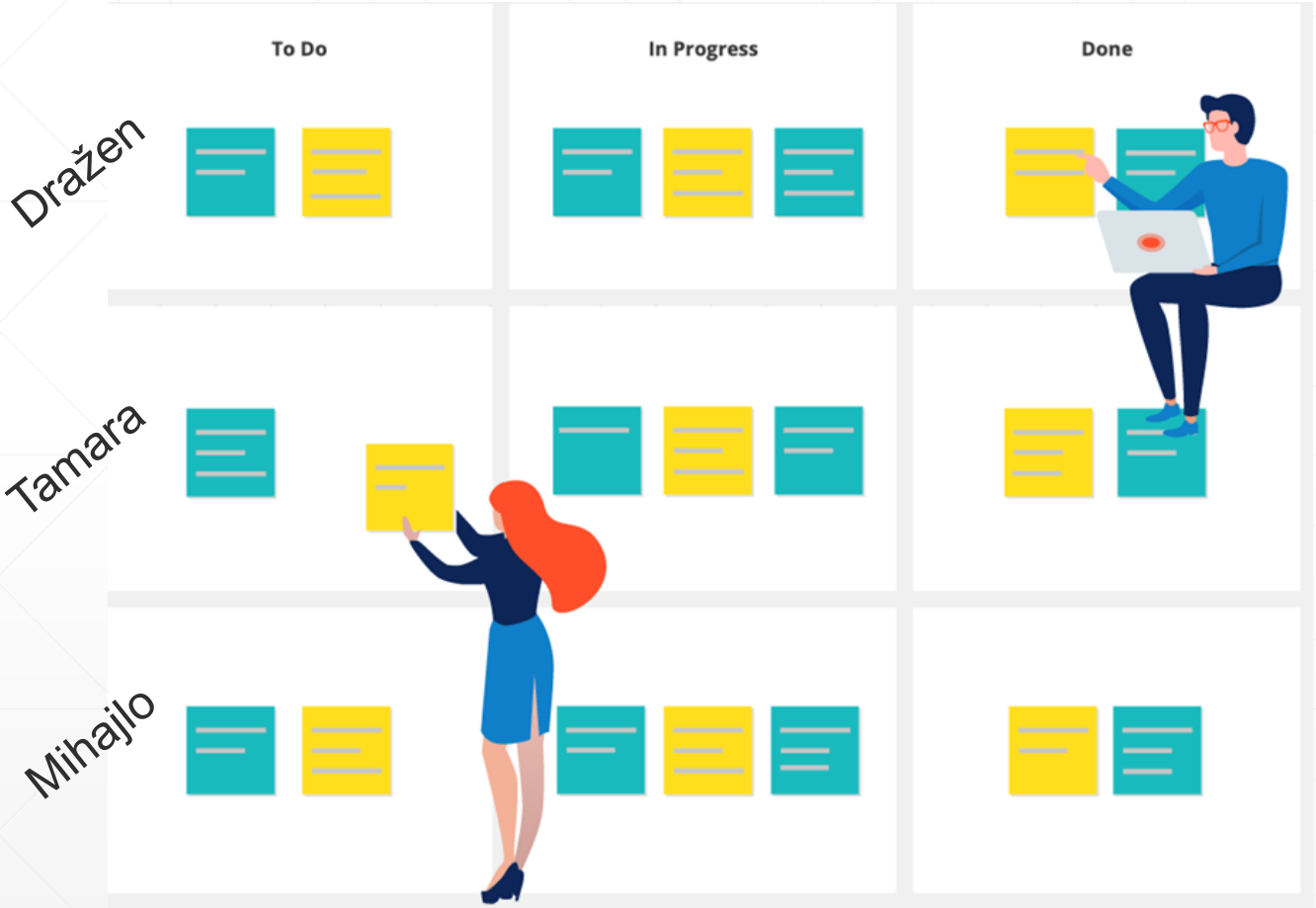
- Tabla može imati više oblika. Najjednostavnija ima samo tri kolone označene kao:
  - Uraditi (*TO DO*),
  - U izradi (*In progress*),
  - Završeno (*Done*).
- Svaki zadatak predstavljen je karticom na tabli.
- Kako se zadaci kreću iz jednog statusa u drugi, pomiču se kartice iz kolone u kolonu, dok sve ne dođu u status završenih.

Kanban tabla sa više kolona



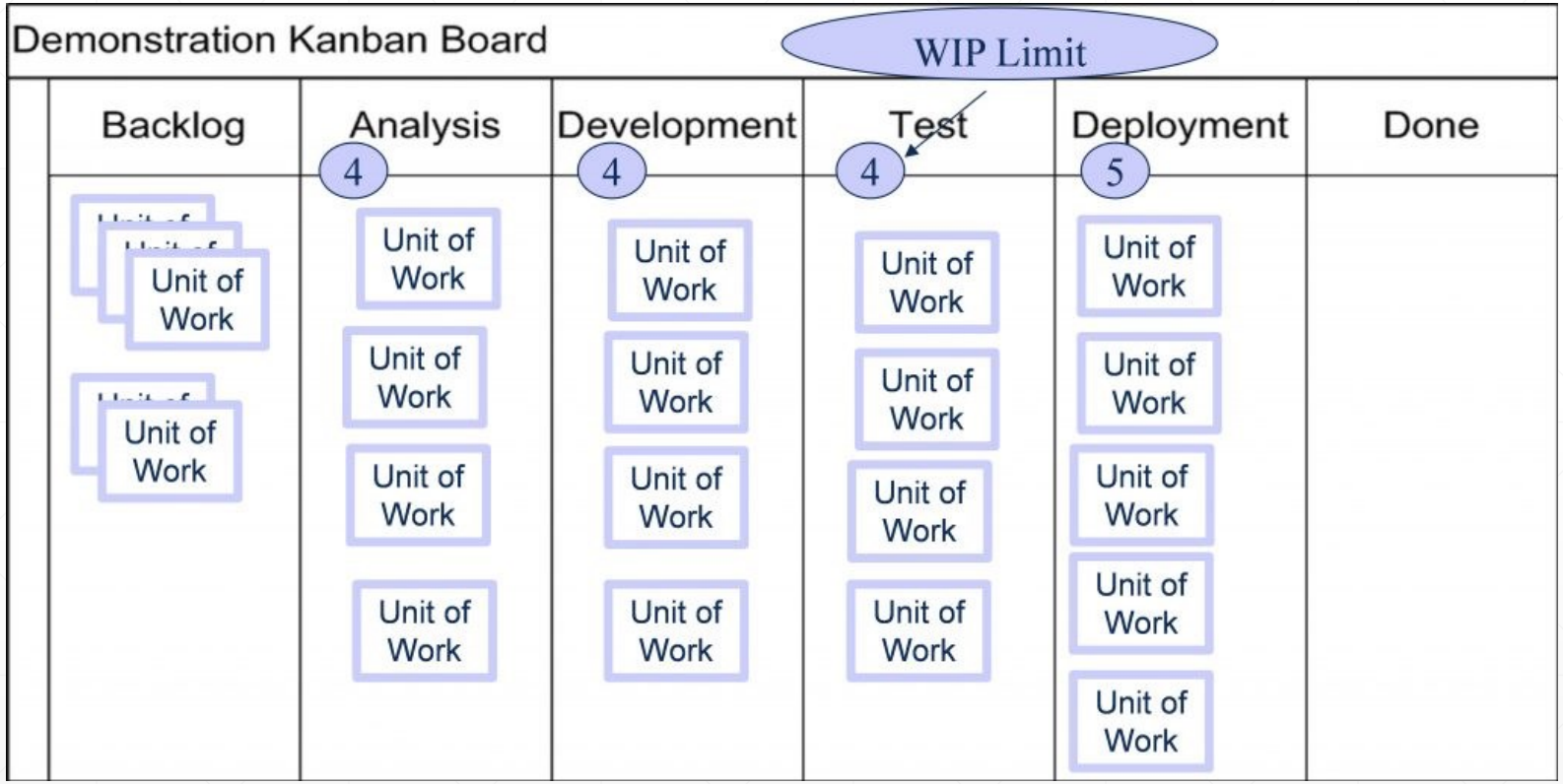
- Svaka kolona (Incepcija, Konstrukcija,...) ima svoje potkolone, npr. kod Ujedinjenog procesa.
- Šrafirane kolone na slici znači da su to kolone za započeti posao (WIP kolone).
- Kartice prikazane na slici samo pokazuju brojeve (ID) zadataka.
- U praksi, kartice mogu uključivati i informacije kao što su naslov zadatka, kratak opis i osoba koja radi na zadatku.

Kanban tabla sa osobama



- Neke Kanban table koriste redove u WIP kolonama da bi se videlo ko radi na kojim zadacima.
- Na primer, zadaci u prvom redu pripadaju Draženu, u drugom Tameri, trećem Mihajlu, itd.
- Zadaci u ne-WIP koloni ne bi pripadali nikome. Na primer, zadaci u koloni Backlog čekaju da ih neko izabere. Zadaci u koloni Acceptance Test čekaju da korisnici provere da li ispunjavaju zahteve.

Kanban tabla sa ograničenjem



- Jedna od Kanban praksi je ograničavanje WIP-a.
- Ako ima previše kartica u WIP kolonama, to ukazuje na potencijalni problem. Na primer, ako projekat ima tri člana tima, ali 17 kartica u progresu, članovi tima možda preuzimaju previše zadataka da bi se pravilno fokusirali na njih.
- Zadaci u koloni *Backlog* čekaju da ih neko izabere.
- Zadaci u koloni *Test* čekaju da testeri provere ispravnost koda.
- Jedan besplatan online alat koji omogućava korišćenje elektronske kanban table je Trello:

  https://trello.com/

Glavni indikatori performansi u Kanban okviru

- Vreme provedeno u svakoj od faza isporuke može se izračunati korišćenjem datuma na karticama zadataka.
- Vreme koje je zadatak čekao je razlika između datuma kada je kartica premeštena u prvu WIP kolonu i datuma kada je kartica prvi put ušla u Backlog.
- **Cycle time** - Vreme kada je zadatak bio u toku (WIP) je razlika između datuma kartice je premešten u kolonu Urađeno i datuma kada je rad prvi put počeo za ovu stavku.

  Cycle time = Time spent in progress

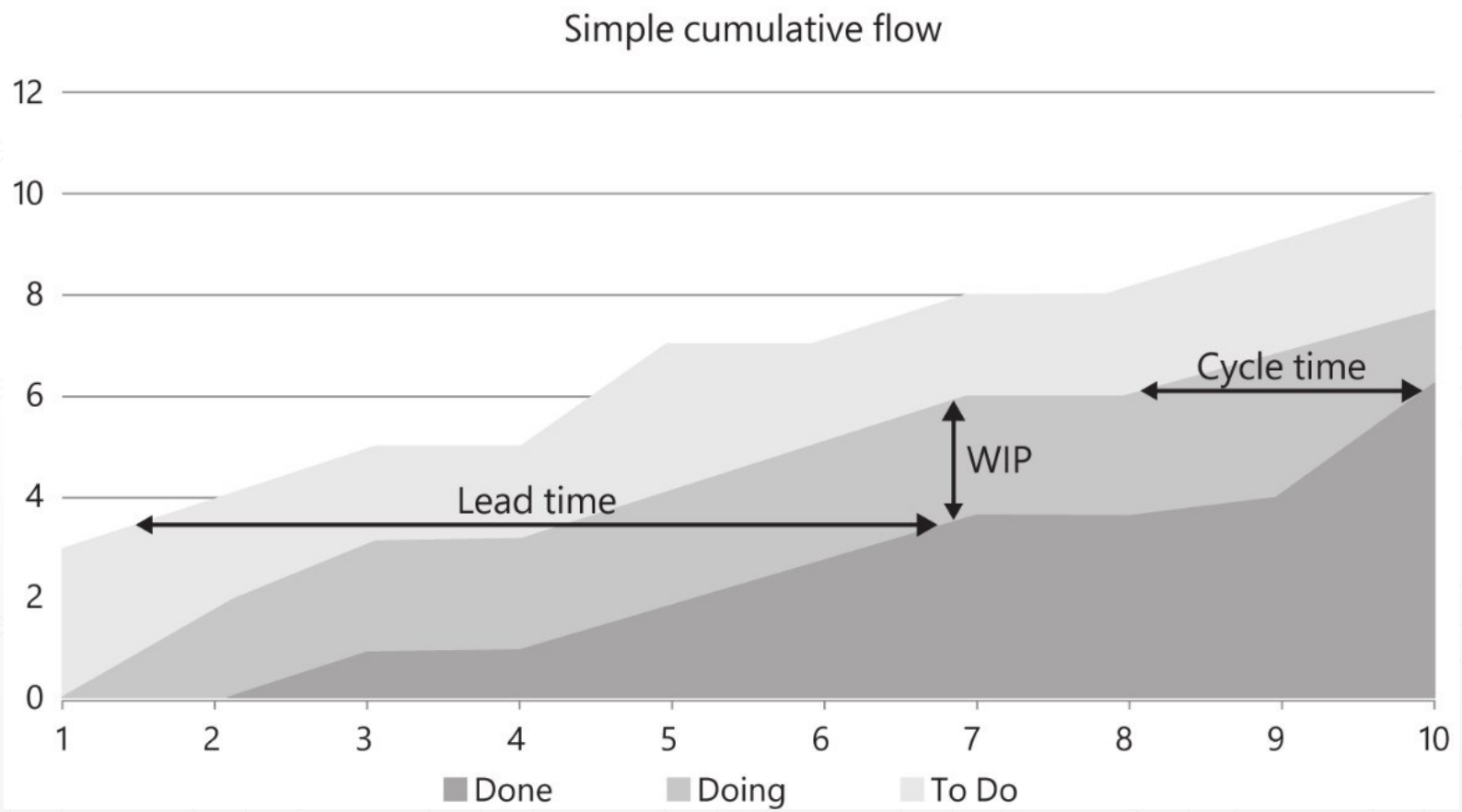
- Dodavanje dva vremena zajedno stvara vreme vođenja zadatka:

  Lead time = Time spent waiting in Backlog + Cycle time

iz knjige Adaptive Code, 2nd edition, Microsoft Press, 2017.

Kumulativni dijagram toka

- Grafik koji na X-osi predstavlja vreme u danima, a na Y-osi broj zadataka u svakoj od kolona (kumulativno).
- Merenja za svaku metriku su sledeća:
  - **Lead time**: razlika na x-osi između područja ToDo i Done.
  - **Cycle time**: razlika na x-osi između oblasti Doing i Done
  - **WIP**: Razlika na y-osi između područja Doing i Done daje ukupan broj zadataka koji su trenutno u toku.
- Iz izgleda grafikona zaključuje se o „zdravlju“ projekta.

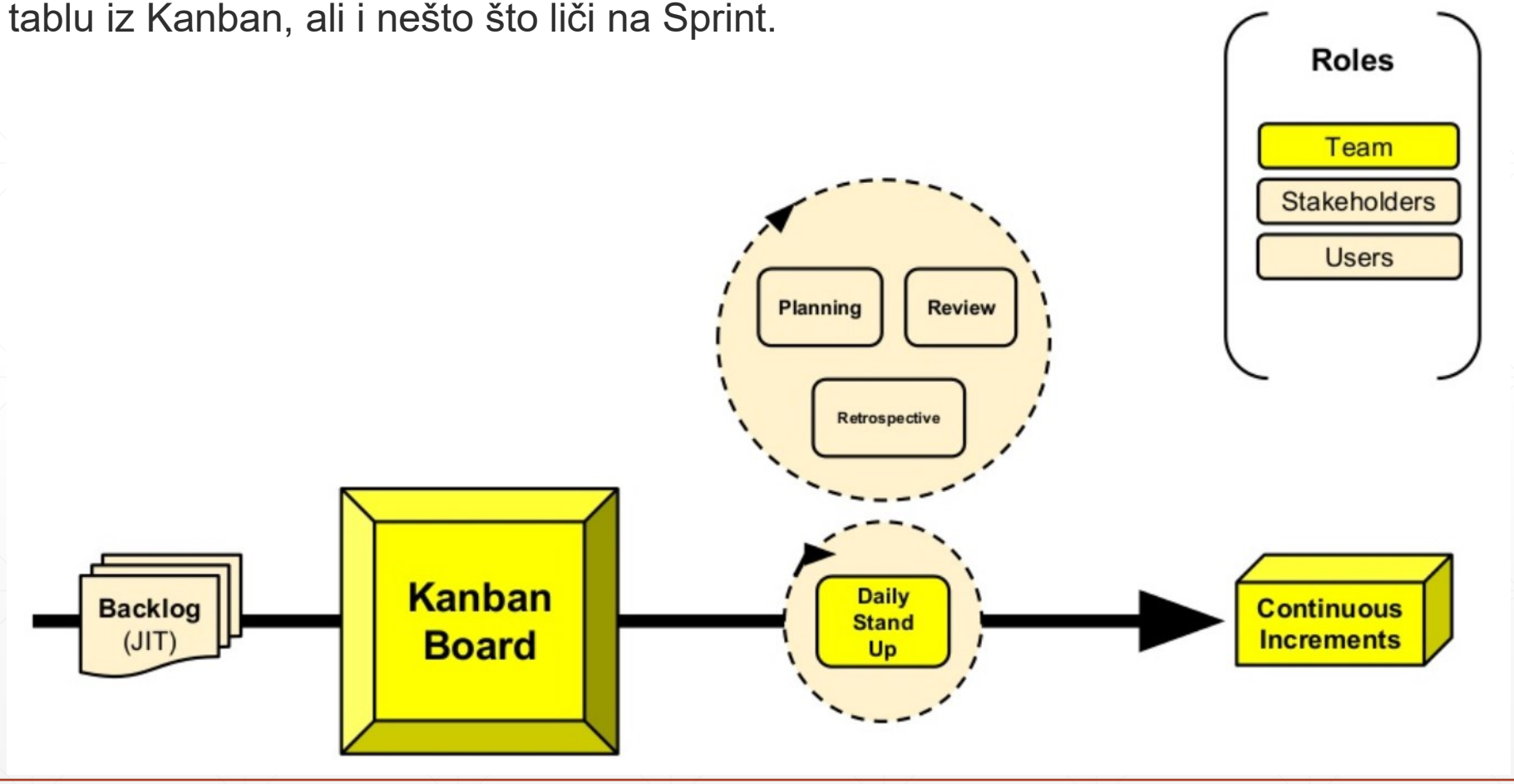


Poređenje Scrum okvira i Kanban okvira

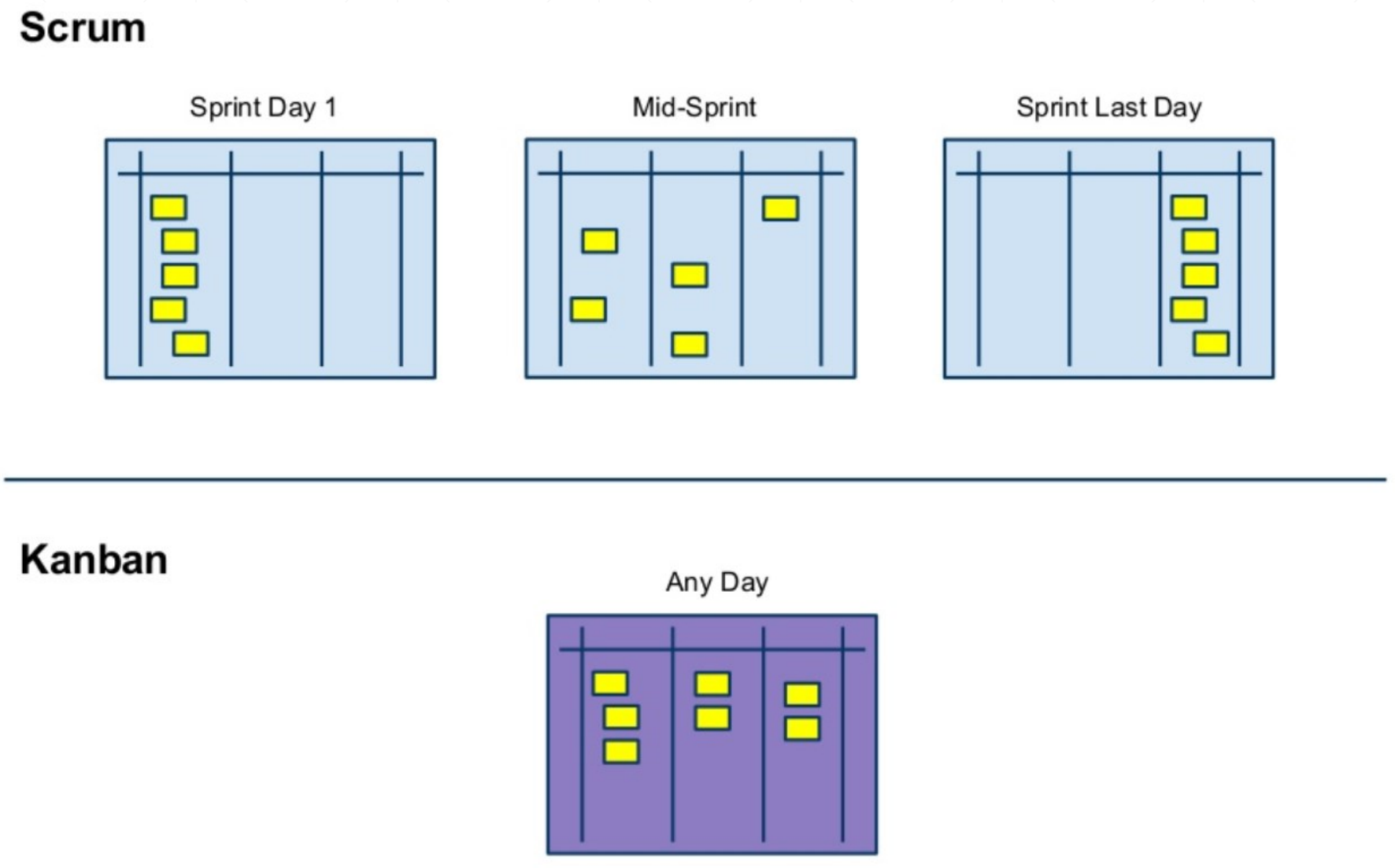
- Uopšteno, Kanban funkcioniše sličnu Scrum okviru.
- Najveća razlika (osim Kanbanovog nedostatka predefinisanih uloga) je u tome što Scrum radi u Sprint-ovima, a Kanban kontinuirano radi jednu po jednu stavku.
- Na početku novog Scrum Sprint-a, sastanak za planiranje Sprint-a povlači odabir visoko prioritetnih stavki iz Backlog-a (preostalih poslova) proizvoda u Backlog Sprint-a. U Sprint-u (koji je obično 15 ili 30 dana) se odrađuje ovaj Backlog da bi se proizveo potencijalno isporučivi inlrement softvera. Kada softver uključi dovoljno novih funkcija, isporučuje se korisnicima.
- Suprotno od toga, u Kanbanu članovi tima povlače stavku najvišeg prioriteta iz projektnog Backlog-a, svaki put kada završe stavku. To je kao da imate Scrum projekat u kojem je svaka stavka u sopstvenom Sprint-u. To Kanbanu omogućava češće isporučivanje inkrementa (kada se isplati). To takođe znači da Kanban može vrlo brzo da odreaguje za na čestu promenu prioriteta kod korisnika.

Kanban + Scrum = Scrumban

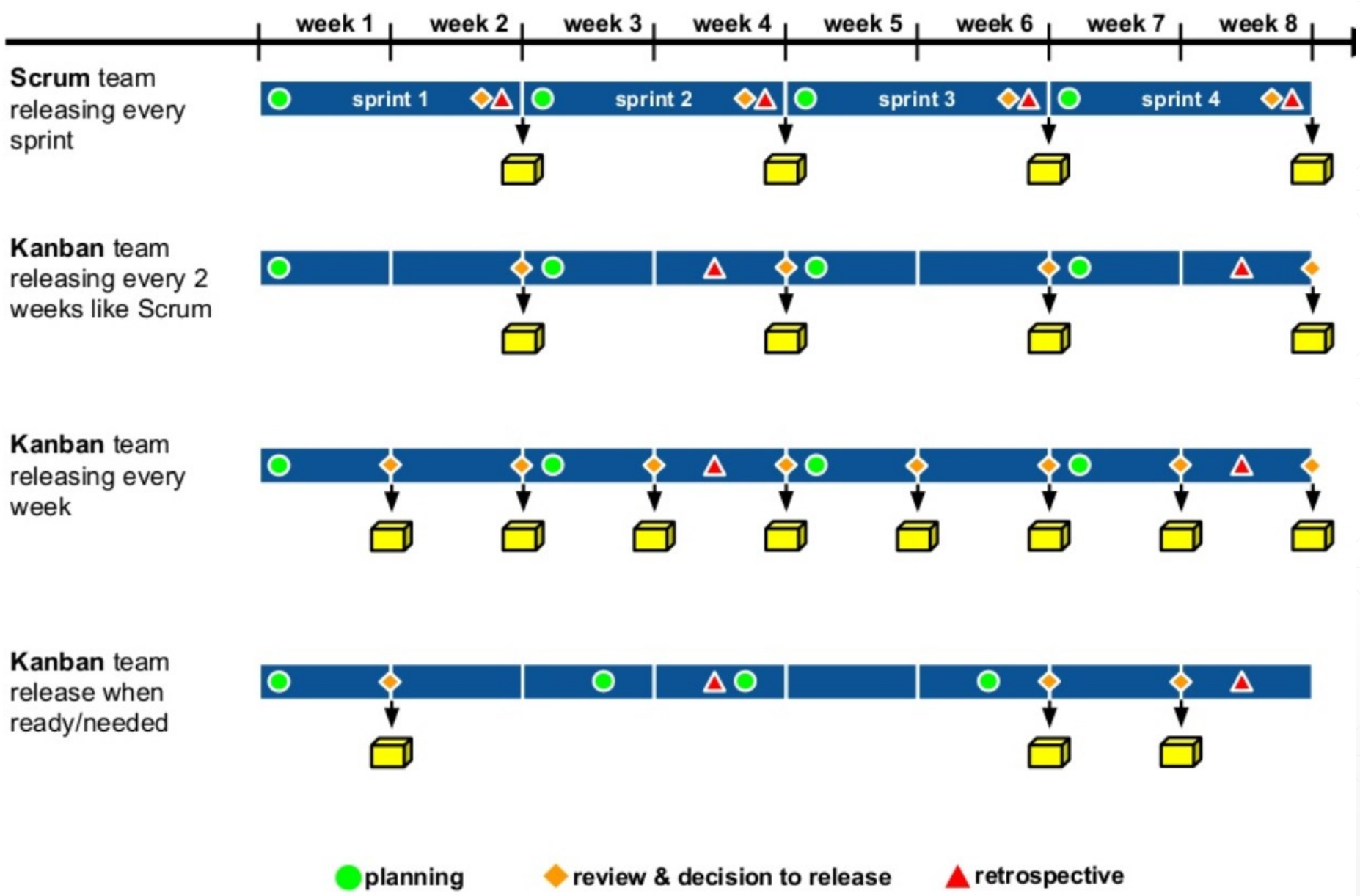
- Imamo tablu iz Kanban, ali i nešto što liči na Sprint.



Scrum vs Kanban



Scrum vs Kanban



Idealno okruženje za Kanban

- Rad vođen događajima
  - dobar za korišćenje kod help-desk / tehničke podrške
  - kod projekata sa frekventnim i veoma neočekivanim korisničkim pričama ili projekata gde se izazivaju česte greške (recimo samo projekat u kome testiramo softver)
  - kod projekata održavanja softvera
- Kompanije koje su uvele *Scrum* su se bazirale na razvoju novih softverskih proizvoda
  - rad je ispred razvoja Sprint-a (R&D)
  - rad je nakon razvoja Sprint-a (sistemsko testiranje, izbacivanje produkcione verzije - *deployment*)
- Unapređuje dobru organizaciju unutar kompanija

Pitanja?

Hvala na pažnji.