Agilni principi razvoja

Boris Matijašević

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Principi Lean proizvodnje

- Filozofija upravljanja (menadžmenta) izvedena iz Toyotinog sustava proizvodnje (TPS)
- Temelji se na eliminaciji svih stvari koje ne stvaraju vrijednost
- 3M:
  - Muda(smanji otpad)
  - Mura(smanji varijacije, imaj ujednačen ritam)
  - Muri (nemoj pregoriti)

3M - Muda

- Smanji otpad, aktivnosti koje ne dodaju vrijednost
- 7 tipova otpada (TIMWOOD)
  - Transportation – trošak koji ne dodaje vrijednost ali povećava rizik da proizvod bude oštećen, izgubljen ili da kasni
  - Inventory – kapitalni izdatak koji nije odmah procesiran ne donosi prihod
  - Motion – svaki trošak koji je nastao tijekom proizvodnje
  - Waiting – proizvodi koji nisu u transportu ili proizvodnji
  - Over processing - kada je napravljeno više posla nego što je potrebno, ili kada su alati previše precizni ili složeni no što je to potrebno
  - Over production - kada je napravljeno više proizvoda nego što je potrebno
  - Defects – gubitak zbog popravka ili nadomještanja neispravnih dijelova ili proizvoda

Vježba

- Navedite primjere za svaki od 7 tipova otpada koji se mogu pojaviti u Vašem okruženju (7 min.).
  - **T**ransportation
  - **I**nventory
  - **M**otion
  - **W**aiting
  - **O**verprocessing
  - **O**verproduction
  - **D**efects

3M – Mura & Muri

- Mura
  - Smanji varijacije, nedosljednost, nepravilnost, nedostatak ujednačenosti
  - Često se može izbjeći boljim planiranjem
- Muri
  - nerazborito, nemoguće, previše odrađeno/napravljeno, preteško
  - Može proizaći iz Mura i iz Muda
  - Kada su strojevi i ljudi iskorišteni više od 100%
  - Rade (previše i predugo) prekovremeno
  - Sve što dovodi do preopterećenja (burn out) ili dosade (bore out)
  - Dovodi do kvarova kada je riječ o strojevima i izostajanja s posla kada je riječ o ljudima

Vježba

- U manifestu agilnog razvoja softvera nadopunite riječi koje nedostaju (4 min.).

Manifest agilnog razvoja softvera

- Tražimo bolje načine razvoja softvera razvijajući softver i pomažući drugima pri njegovom razvoju.
- Takvim radom smo naučili da više cijenimo:
  1. **Ljude i interakcije** nego **procese i oruđa**
  2. **Upotrebljiv softver** nego **iscrpnu dokumentaciju**
  3. **Suradnju s naručiteljem** nego **pregovaranje oko ugovora**
  4. **Reagiranje na promjenu** nego **ustrajanje na planu**
- Drugim riječima, iako cijenimo vrijednosti na desnoj strani, više vjerujemo u one na lijevoj.

Principi agilnog razvoja softvera (1-6)

Rukovodimo se sljedećim načelima:

1. Najvažnije nam je zadovoljstvo naručitelja koje postižemo ranom i neprekinutom isporukom softvera koji nosi vrijednost.
2. Spremno prihvaćamo promjene zahtjeva, čak i u kasnoj fazi razvoja. Agilni procesi upregu promjene da naručitelju stvore kompetitivnu prednost.
3. Često isporučujemo upotrebljiv softver, u razmacima od nekoliko tjedana do nekoliko mjeseci, nastojeći da razmak bude čim kraći.
4. Poslovni ljudi i razvojni inženjeri moraju svakodnevno zajedno raditi, tijekom cjelokupnog trajanja projekta.
5. Projekte ostvarujemo oslanjajući se na motivirane pojedince. Pružamo im okruženje i podršku koja im je potrebna i prepuštamo im posao s povjerenjem.
6. Razgovor uživo je najučinkovitiji način prijenosa informacija razvojnom timu i unutar tima.

Principi agilnog razvoja softvera (7-12)

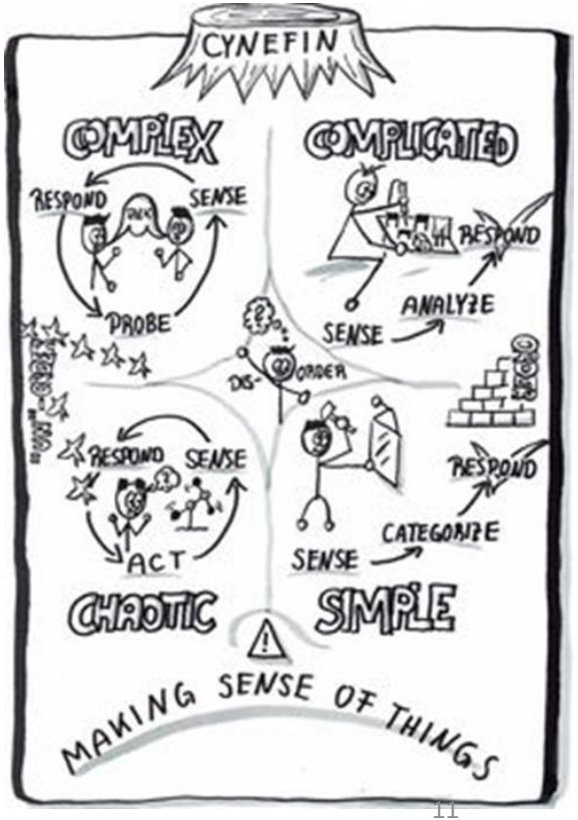
7. Upotrebljiv softver je osnovno mjerilo napretka.
8. Agilni procesi potiču i podržavaju održivi razvoj. Pokrovitelji, razvojni inženjeri i korisnici trebali bi moći neograničeno dugo zadržati jednak tempo rada.
9. Neprekinuti naglasak na tehničkoj izvrsnosti i dobar dizajn pospješuju agilnost.
10. Jednostavnost – vještina povećanja količine posla kojeg ne treba raditi – je od suštinske važnosti.
11. Najbolje arhitekture, projektne zahtjeve i dizajn, stvaraju samo–organizirajući timovi.
12. Tim u redovitim razmacima razmatra načine da postane učinkovitiji, zatim usklađuje i prilagođava svoje ponašanje.

Vježba

- Odaberite 3 principa agilnog razvoja softvera za koje smatrate da su najvažniji (5 min.).

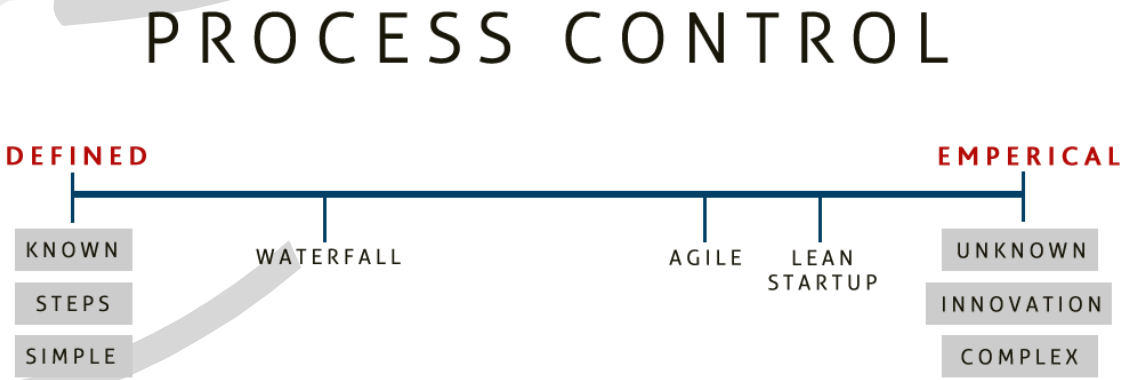
Vrste procesa

- Cynefin-ov okvir (Cynefin framework) [Knevin]
  - Definira načine donošenja odluka kod različitih sustava
- Pet domena:
  - Jednostavna – veza između uzroka i posljedice jasna je svima. Dobra je praksa Opažanje-Kategorizacija-Reagiranje.
  - Komplicirana – veza između uzroka i posljedice zahtjeva analizu ili neku drugu vrstu istraživanja. Dobra je praksa Opažanje-Analiza-Reagiranje.
  - Složena – veza između uzroka i posljedice može se shvatiti samo u retrospektivi. Dobar pristup je Ispitivanje(uzorkovanje)-Opažanje-Reagiranje.
  - Kaotična – u kojoj ne postoji veza između uzroka i posljedice na sistemskom nivou. Težimo stabilizirati stanje pristupom Djelovanje(stabiliziranje)-Opažanje-Reagiranje
  - Nered – u kojem se ne zna koja veza između uzroka i posljedice. Npr. mislimo da je nešto očito ali je zapravo kaotično -> uvod u katastrofu.



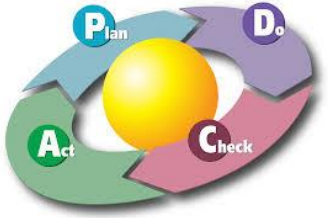
Process control

- Ukoliko imamo proces koji će opetovano imati rezultate koji su zadovoljavajuće kvalitete koristimo *Defined process control*
- Ukoliko se zadovoljavajući ishod ne može postići zbog kompleksnosti sustava koristimo *Empirical process control*



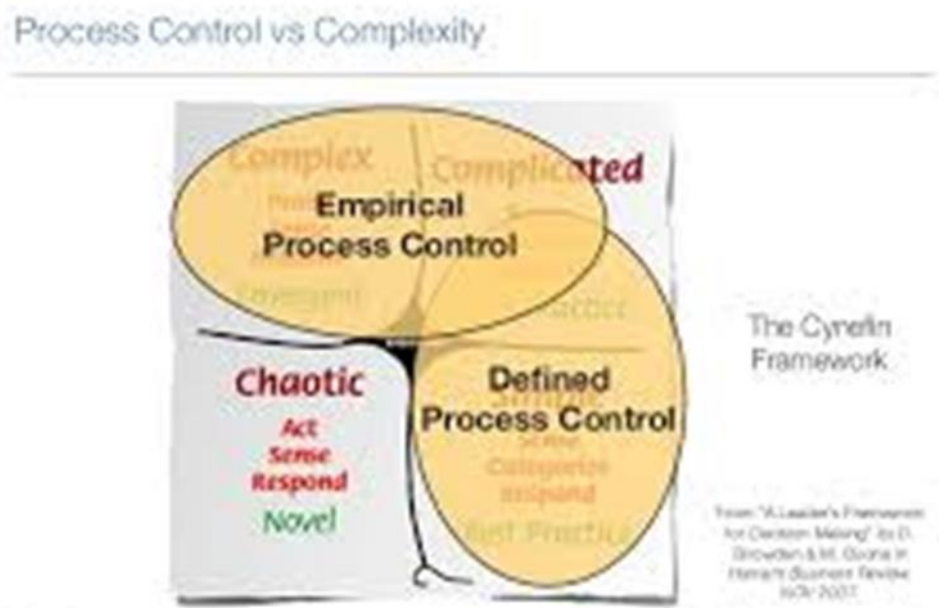
Empirical Process Control

- Tipičan iterativni korak sastoji se od planiranja, izrade, inspekcije i djelovanja (Plan, Do, Check, Act - PDCA)
- Kako bi EPC bio moguć potrebno je da:
  - svi aspekti procesa koji imaju utjecaja na ishod budu dostupni svim dionicima u procesu – **Transparentnost**
  - su svi aspekti procesa dovoljno često analizirani kako bi se otkrile neprihvatljive razlike u procesima - **Inspekcija**
  - onaj dionik koji radi inspekciju napravi prilagodbe procesa ukoliko su jedna ili više aspekata procesa u nedozvoljenom području - **Prilagodba**



Odnos između Process control i kompleksnosti sustava

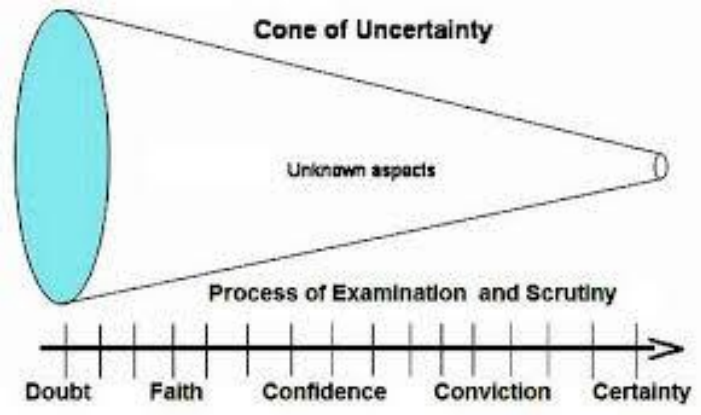
- Defined Process Control primjeren je za jednostavne i složene procese prema Cynedin Framewoorku
- Empirical Process Control također je primjeren kod složenih procesa ali i za kompleksne procese prema Cynedin Framewoorku
- Kakve sustave mi razvijamo?



Cone of uncertainty

- Zadatak:
  - Autocesta ima 4 trake u dva smjera
  - Auto prosječno vozi brzinom od 130km/h
  - Razmak između automobila je najmanje 2s.
  - Koliko će automobila proći autocestom u jednom satu?

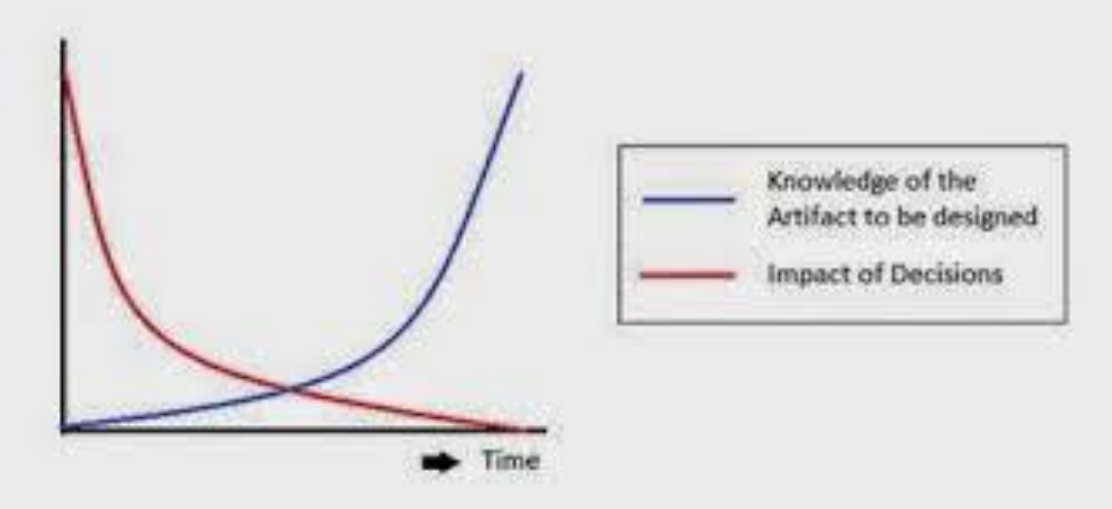
Cone of uncertainty



- Zadatak:
  - Autocesta ima 4 trake u dva smjera
  - Auto prosječno vozi brzinom od 130km/h
  - Razmak između automobila je najmanje 2s.
  - Koliko će automobila proći autocestom u jednom satu?
- Odgovor je teško dati jer se radi o složenom sustavu. Broj automobila na cesti ovisi o dobu dana, o tome je li radni dan ili neradni, ovisi o godišnjem dobu ili o praznicima…
- Najbolje bi bilo napraviti opažanje i izmjeriti broj automobila koji prođu autocestom u 10 minuta. Nakon toga možemo bolje pretpostaviti koliko će automobila proći u jednom satu. Nastavimo li opažanje poboljšat će se i naša procjena.
- Ovo je karakteristika i softverskih razvojnih projekata (ali i nekih drugih) kod kojih je teško procijeniti trajanje razvoja prije no što znamo dovoljno o sustavu.

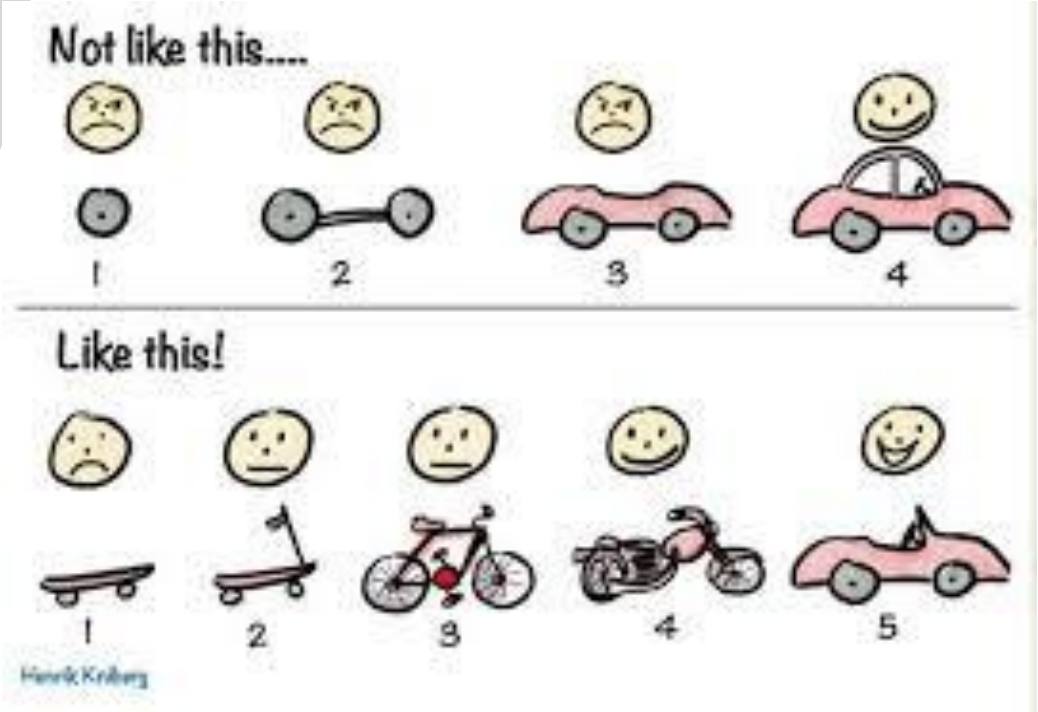
Kada treba donositi odluke?

- S obzirom da na početku malo znamo o sustavu a utjecaj odluke na ishod projekta je velik, logično bi bilo donositi odluke što je moguće kasnije kako se povećava naše znanje o sustavu



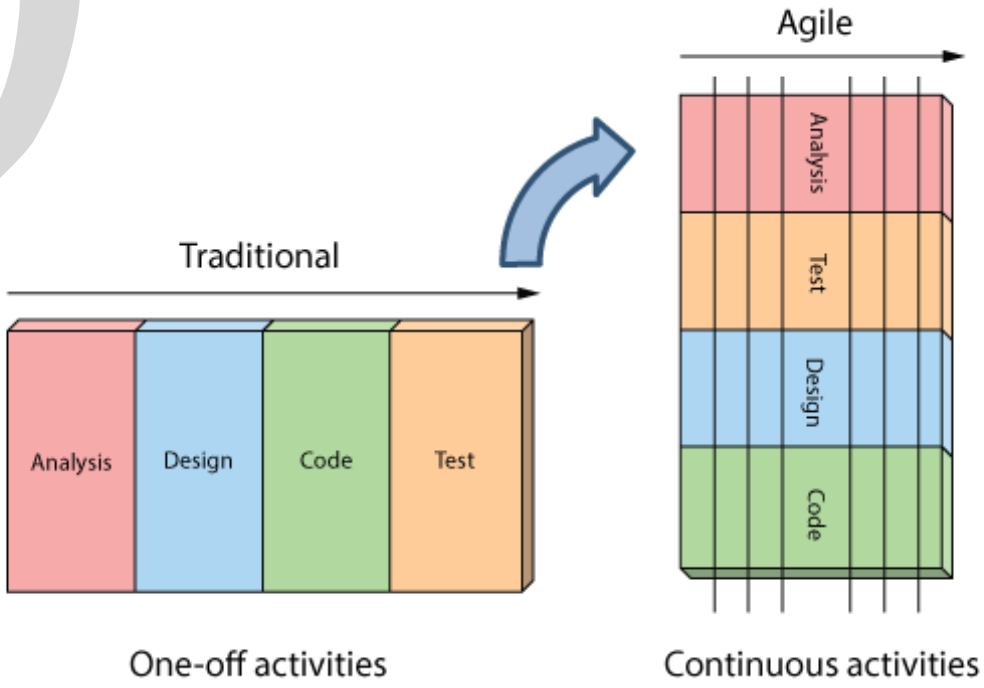
Inkrementalan i iterativan razvoj sustava

- Korisnik od nas zatraži razvoj prijevoznog sredstva
- Radit ćemo **iterativno** i u svakom koraku isporučiti proizvod
- Radit ćemo i **inkrementalno** i u svakom koraku iteracije isporučiti proizvod koji korisnik može koristiti kao prijevozno sredstvo



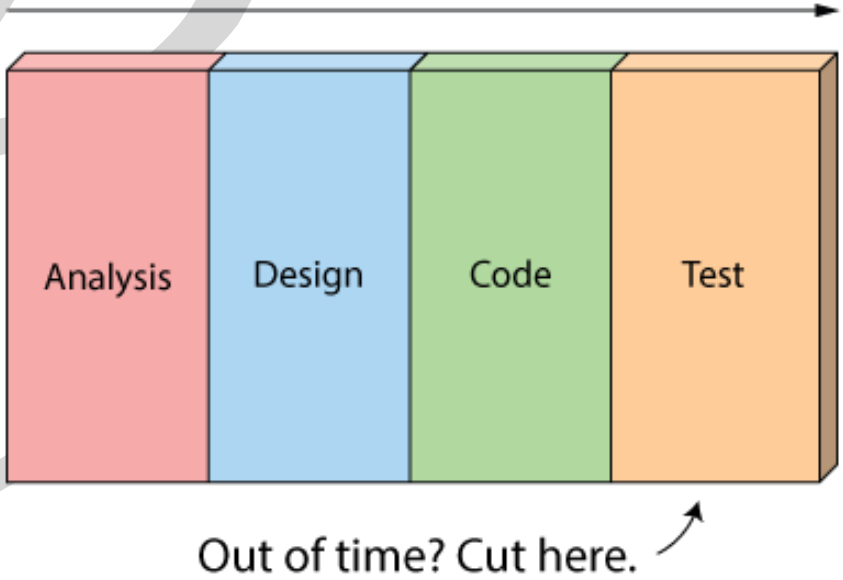
Razlika u odnosu na tradicionalni pristup

- Analiza, dizajn, razvoj i testiranje aktivnosti su koje se provode kontinuirano



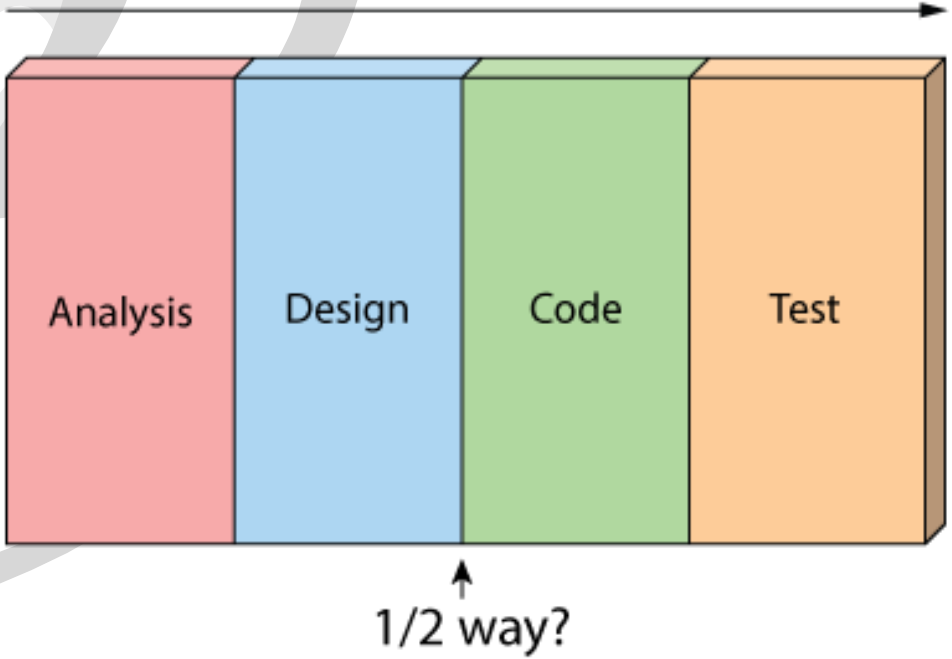
Problemi tradicionalnog pristupa

- Loša kvaliteta tradicionalnih metoda zbog kasnog testiranja.



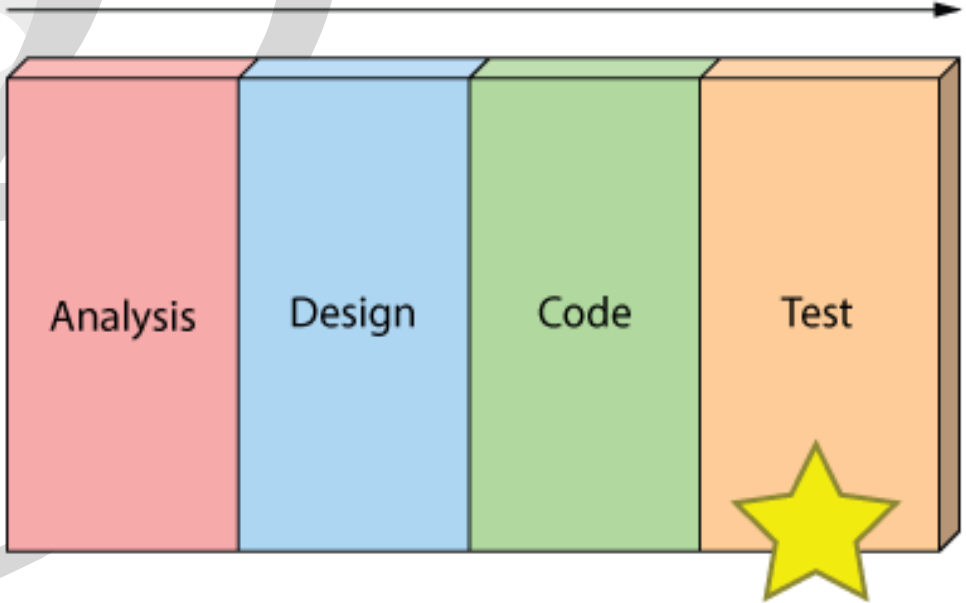
Problemi tradicionalnog pristupa

- Slaba vidljivost



Problemi tradicionalnog pristupa

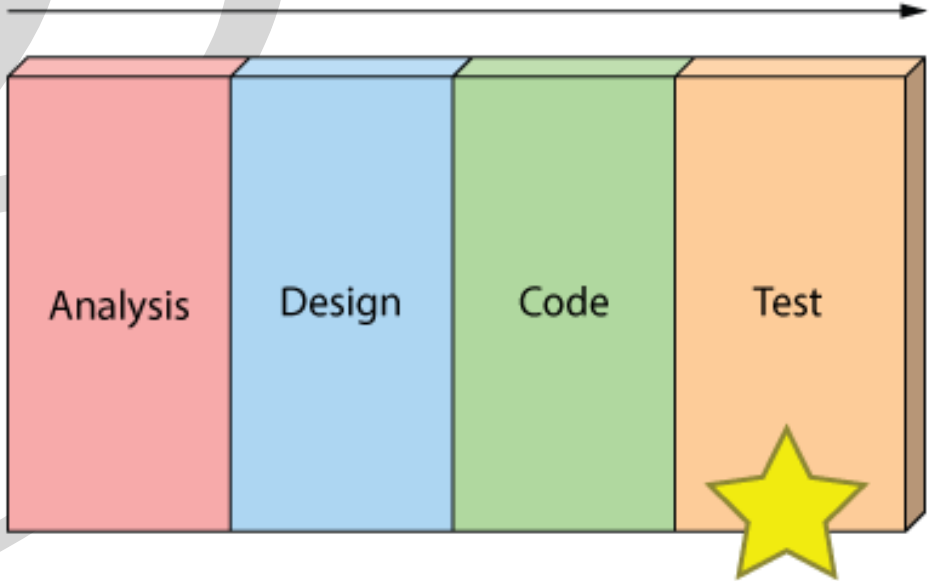
- Preveliki rizik



Houston we have a problem

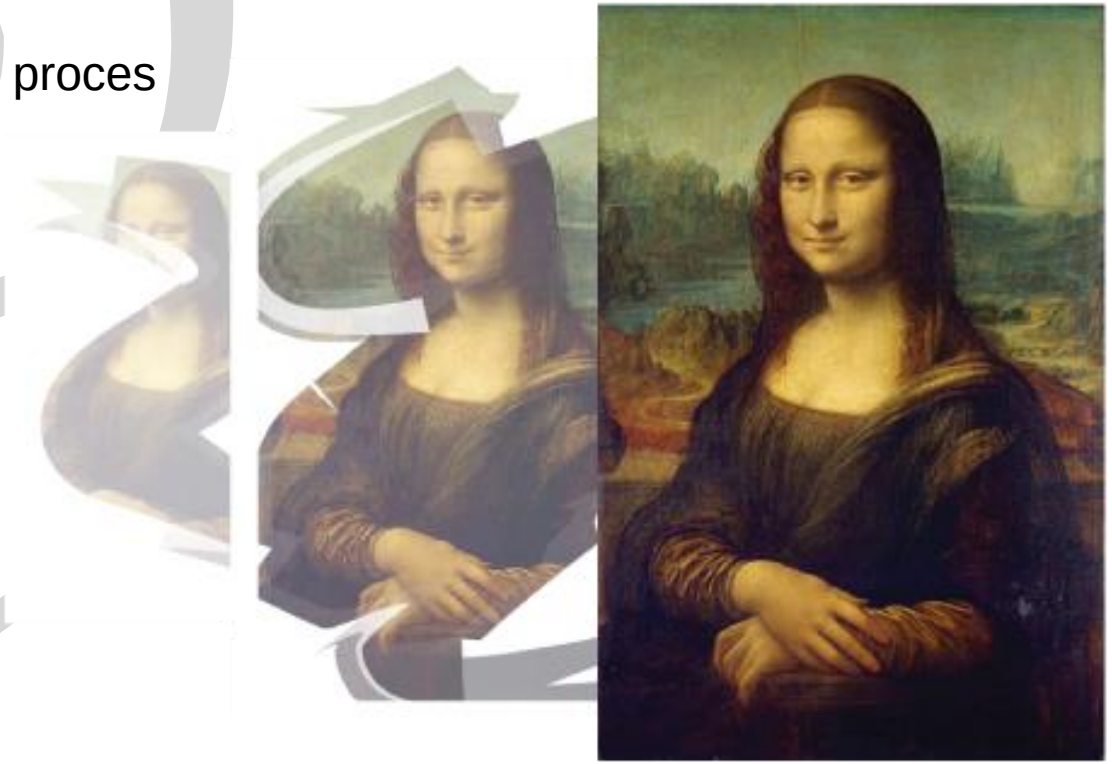
Problemi tradicionalnog pristupa

- Teško se obavljaju promjene



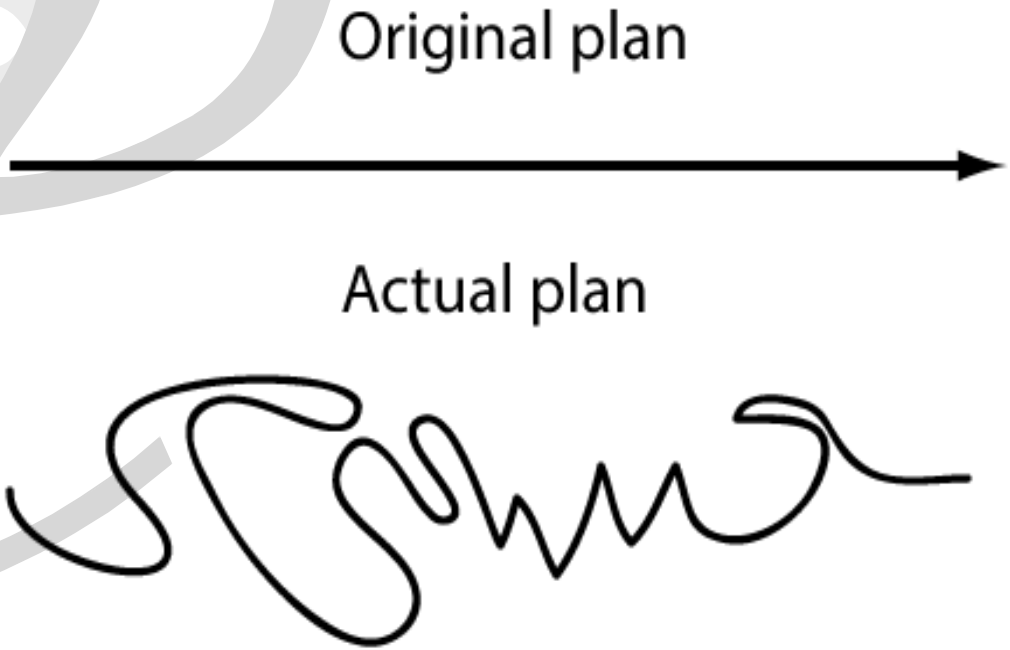
Razlika u odnosu na tradicionalni pristup

- Razvoj je iterativan proces



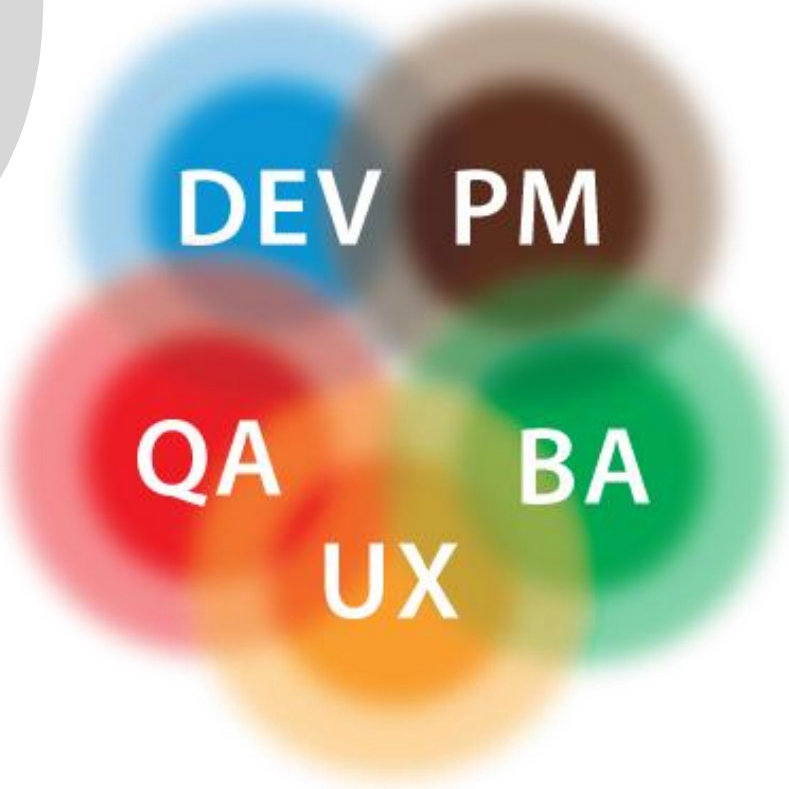
Razlika u odnosu na tradicionalni pristup

- Planiranje je adaptivno



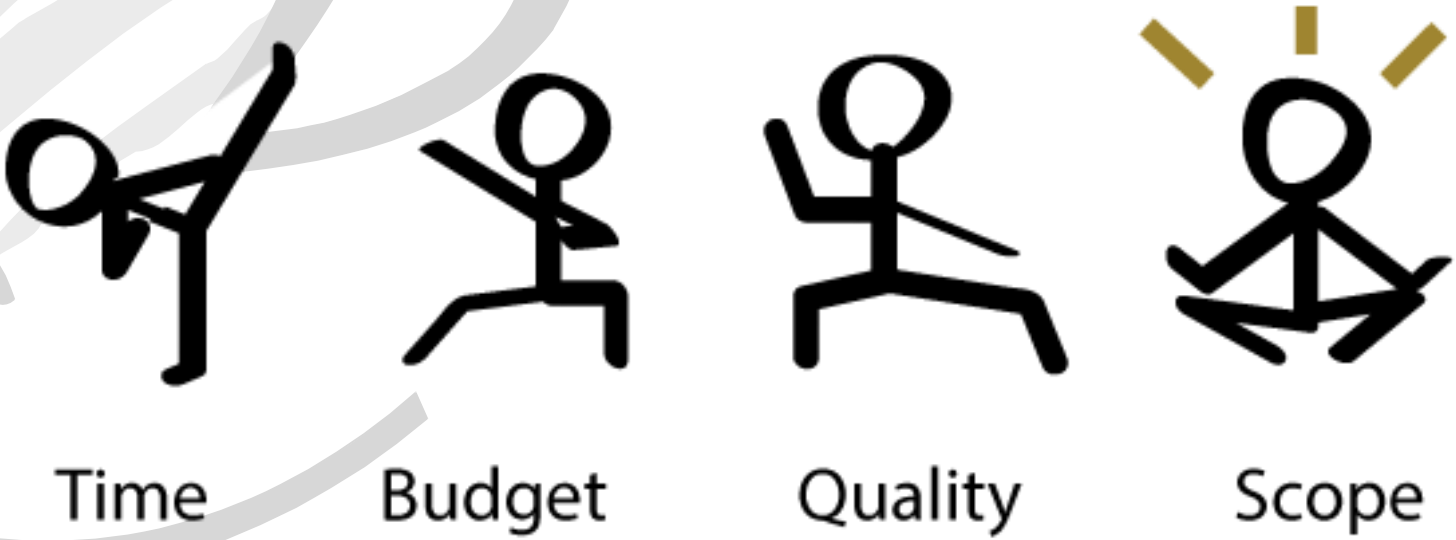
Razlika u odnosu na tradicionalni pristup

- Uloge su nejasne



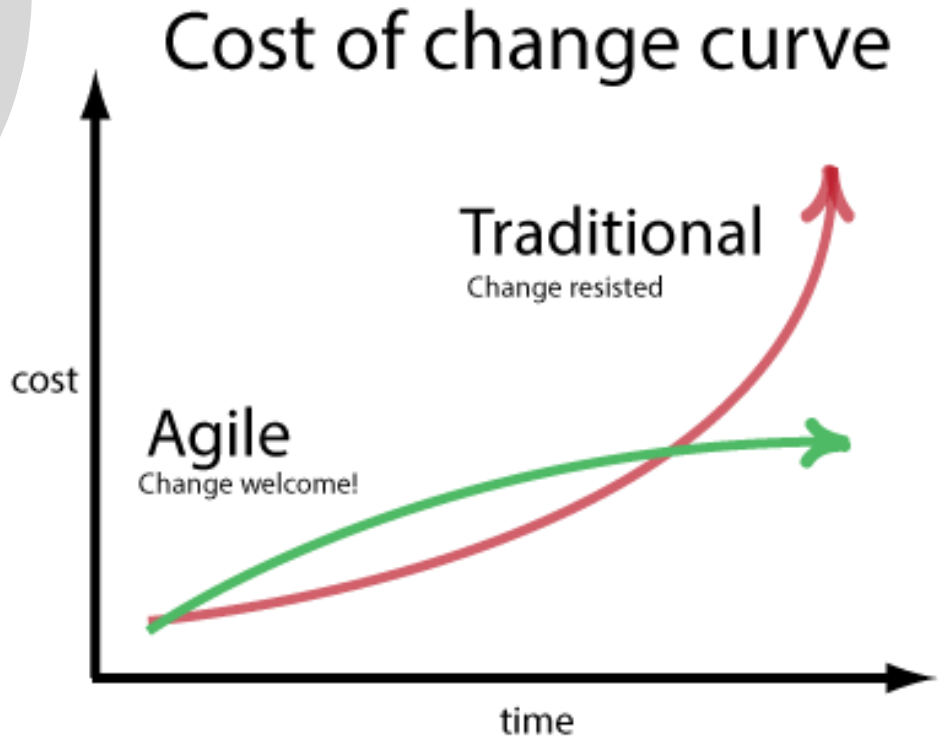
Razlika u odnosu na tradicionalni pristup

- Opseg je varijabilan



Razlika u odnosu na tradicionalni pristup

- Zahtjevi su promjenjivi



Razlika u odnosu na tradicionalni pristup

- Softver koji radi je primarna mjera uspjeha projekta

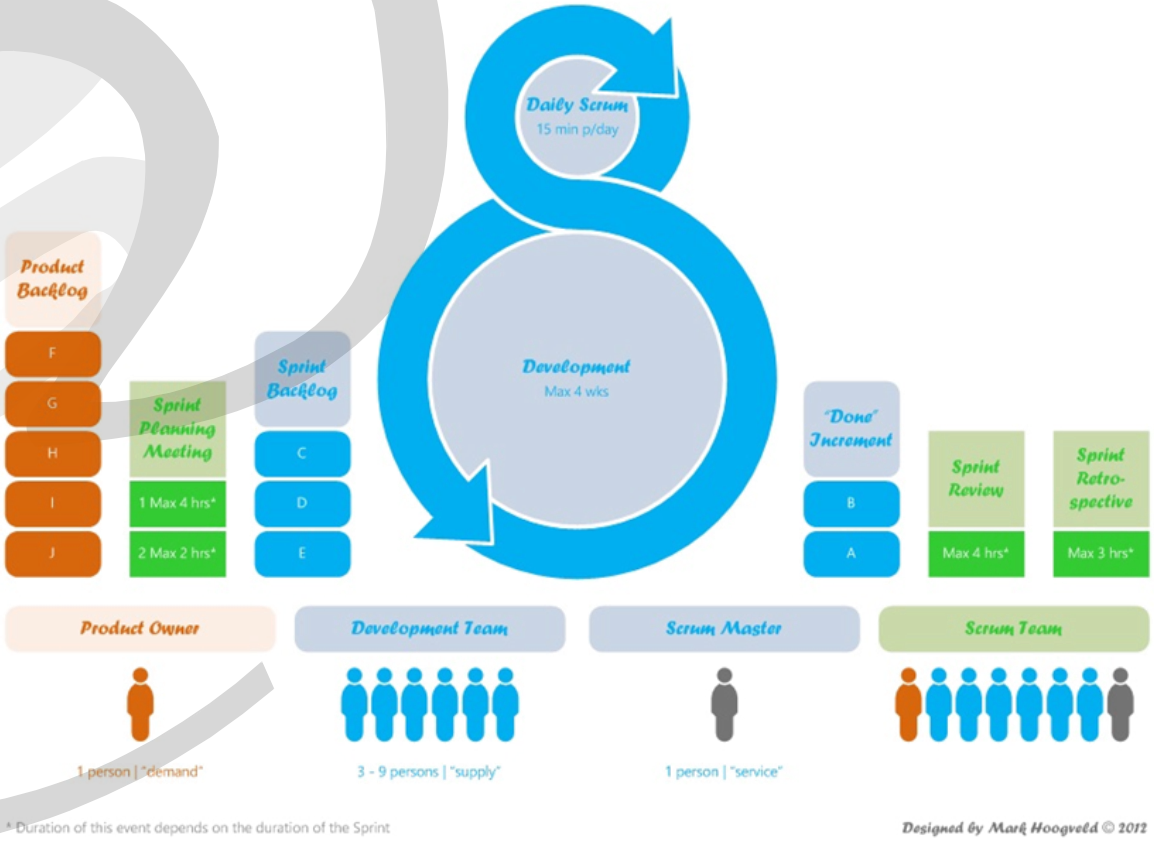
(working software)

OVER

Agilni razvoj (Scrum)

- Osnovni elementi:
  - Uloge
  - Backlog (korisničke priče)
  - Procjena
  - Planiranje
  - Iteracija razvoja (sprint)
  - Retrospektiva
  - Timovi

Scrum Overview



Vizija

- Vizija je važna jer kako bi inače mogli odrediti poslove koje trebamo raditi i koji su nam prioriteti? Plan za što?
- Postavlja usmjerenje za razvoj proizvoda kako bi svi razumjeli koji je krajnji cilj
  - Pomaže donošenju odluka
- Svojstva:
  - Izražava potrebe korisnika
  - Dijeli se sa svim dionicima – rano i često
  - Jasna i privlačna
  - Sažeta

Elevator pitch

- *Elevator Pitch* ili *Elevator Speech* je kratak i uvjerljiv govor o osobi, organizaciji ili grupi, ideji ili proizvodu, usluzi ili projektu. Dobar *Elevator pitch* je često dio načina marketinške komunikacije, brandiranja ili PR programa. Dobri *Elevator Pitch* govori sažeti su i uvjerljivi su ciljnoj publici.
- Primjer predloška:
  - ZA <CILJENOG KORISNIKA>
  - KOJI <IMA NEKU POTREBU> / <JE NEZADOVOLJAN S POSTOJEĆIM>
  - <IME PROIZVODA>
  - JE <KATEGORIJA PROIZVODA>
  - KOJI IMA <KLJUČNA PREDNOST>
  - ZA RAZLIKU OD <GLAVNI KONKURENT>
  - NAŠ PROIZOVD <DALJNJA DIFERENCIJACIJA>

„In your pocket”

- The iPod will be a portable digital music player that will hold 5000 songs. It will have battery life measured in days, not hours. You will navigate the thousands of songs with a single finger. You will sync all your music from your computer to the iPod in minutes automatically, so you can have all your music in your pocket.

- iPod će biti prijenosni digitalni muzički reproduktor koji će pohranjivati 5000 pjesama. Imat će bateriju koja će trajati danima, a ne satima. Navigirat ćete tisućama pjesama s jednim prstom. Automatski će u nekoliko minuta sinkronizirati svu muziku s računala tako da ćete moći imati svu svoju muziku u svom džepu.

Uloge

- Team
  - Članovi tima moraju moći isporučiti svu potrebnu funkcionalnost
  - 2 (ekstremno i dvojbeno) do 9 članova (više od toga nije dovoljno upravljivo)
- Product Owner
- Scrum Master

Product backlog

- Korisničke priče koje još nisu implementirane
- Prije svake iteracije dogovara se koje korisničke priče će se napraviti u sljedećoj iteraciji
- Radi se procjena korisničkih priča i dogovara sadržaj sljedećeg sprinta s obzirom na prioritete razvoja
- Product Owner & Team

Korisnička priča

- Predložak za korisničke priče:
  - Kao <tip korisnika>
  - Želim <napraviti nešto>
  - Kako bih <postigao neki poslovni cilj>
- Primjer:
  - Kao korisnik želim da sustav zapamti moje podatke kako bih mogao brže obaviti kupnju
- Korisnička priča može imati/trebala bi imati uvjete prihvaćanja – acceptance criteria (AC). Na primjer:
  - Korisnički podaci prikazani su prije nego što se obavi kupnja
  - Promjenu korisničkih podataka moguće je pokrenuti direktno na ekranu za pregled korisničkih podataka
  - Moguće je učiniti da sustav zaboravi podatke korisnika

Korisnička priča (2)

- Predložak za AC koju je moguće testirati:
  - Ukoliko je <neki kontekst>
  - Kada <se dogodi neka akcija>
  - Tada <se treba postići neki očekivani rezultat>
- Primjer:
  - Ukoliko sam na Amazonovim web stranicama
  - I prijavljen sam kao korisnik
  - Kada kliknem na „1-Click” gumb
  - Tada bih trebao vidjeti detalje moje narudžbe

Tehnika – procjena veličine i kompleksnosti

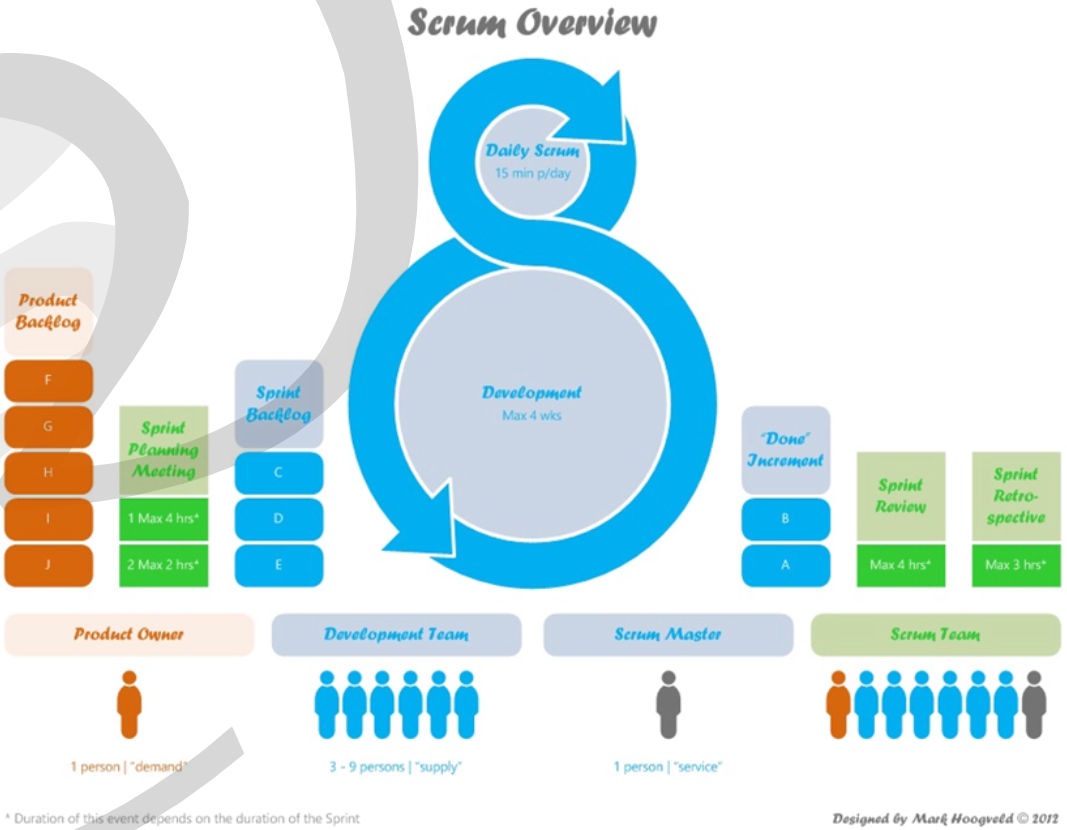
- Tim developera daje procjenu veličine i kompleksnosti nekog posla
- Svakoj korisničkoj priči daju se tzv. story points (SP) za svaki posao
- SP koji se mogu dati slični su Fibonaccijevom nizu (0,1,2,3,5,8,13,20,40,100)
- Nije bitan apsolutan iznos SP nego relativan odnosno njihov odnos, služi kako bi se svi usuglasili oko toga u kakvom su odnosu korisničke priče
- Ne korelira s vremenom potrebnim za obavljanje
- U tri kruga se ponavlja ‘glasanje’ o tome koliko SP vrijedi neki posao, nakon svakog kruga slijedi kratka moderirana rasprava o danim SP
- Product Owner odgovara na pitanja developera oko toga što se očekuje od pojedinog posla
- Sve ovo je procjena temeljena na razumijevanju onoga što je korisnička priča i temeljena na iskustvu članova tima

Iteracija (sprint)

- Sprint uvijek traje fiksno vrijeme koje je između 1 i 4 tjedna
- Sprint je možda pogrešan naziv je asocira na to da se radi intenzivnije nego inače, ali zapravo se cijelo vrijeme treba raditi konstantnim tempom (3M)
- Svaki tim se mora dogovoriti oko toga što se smatra gotovim poslom – mora napraviti svoju definiciju Gotovog
- Samo korisničke priče koje su gotove prema definiciji o kojoj smo se svi složili su Gotove
- Ono što je 90% gotovo nije Gotovo.
- Sve što je u Sprint Backlogu treba biti gotovo prema definiciji Gotovog. To tako treba biti u svakom sprintu.

Retrospektiva

- Nakon svake iteracije treba napraviti tzv. Retrospektivu
- Retrospektiva je možda i najvažniji dio sprinta.
- Koliko je uspješno realiziran cilj sprinta
- Bez Retrospektive to nije EPC!
- Jedan način za obavljanje Retrospektive je da se definiraju kategorije:
  - Nastaviti raditi
  - Raditi više
  - Raditi manje
  - Prestati raditi
  - Početi raditi
- Služi tome da u sljedećoj iteraciji budemo produktivniji/bolji


Scrum Overview
Daily Scrum
15 min p/day
Development
Max 4 wks
Product Backlog
F
G
H
I
J
Sprint Planning Meeting
1 Max 4 hrs*
2 Max 2 hrs*
Sprint Backlog
C
D
E
"Done" Increment
B
A
Sprint Review
Max 4 hrs*
Sprint Retro-spective
Max 3 hrs*
Product Owner
Development Team
Scrum Master
Scrum Team
1 person | "demand"
3 - 9 persons | "supply"
1 person | "service"
* Duration of this event depends on the duration of the Sprint
Designed by Mark Hoogveld © 2012

Tim

- Kako postići da je cjelina veća od sume njezinih dijelova?
- Možemo djelovati kao
  - individualci
  - grupe ljudi
  - timovi

Grupa ljudi

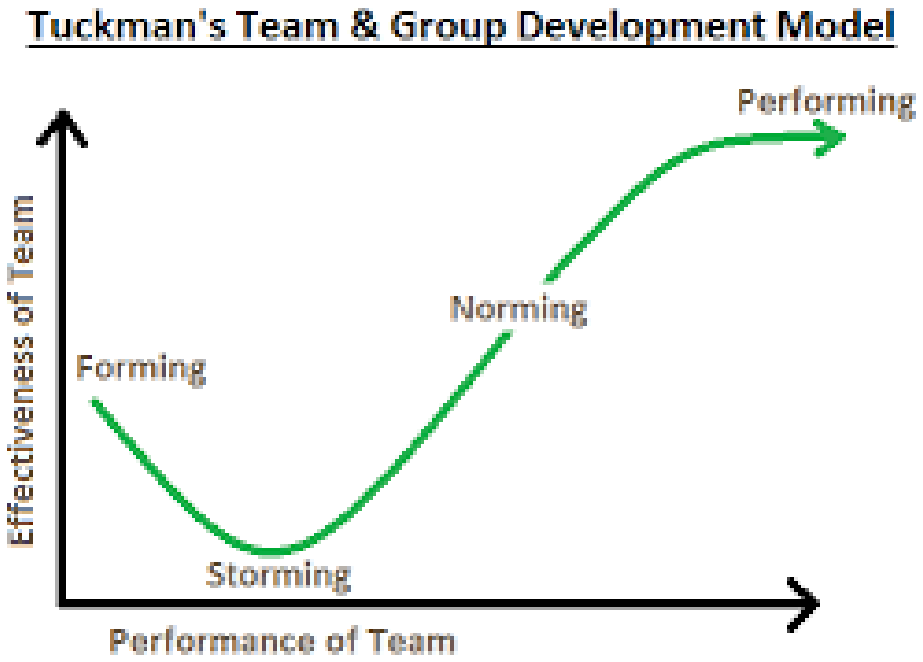
- Grupa ljudi – npr. ljudi na autobusnoj stanici, nema sinergije
- Zajedno su, ako čekaju isti autobus možda i idu u istom smjeru ☺ - imaju isti cilj

Tim

- Tim – npr. tim u pit stopu na F1, ima sinergije
- Djeluju usklađeno kako bi postigli zajednički cilj.

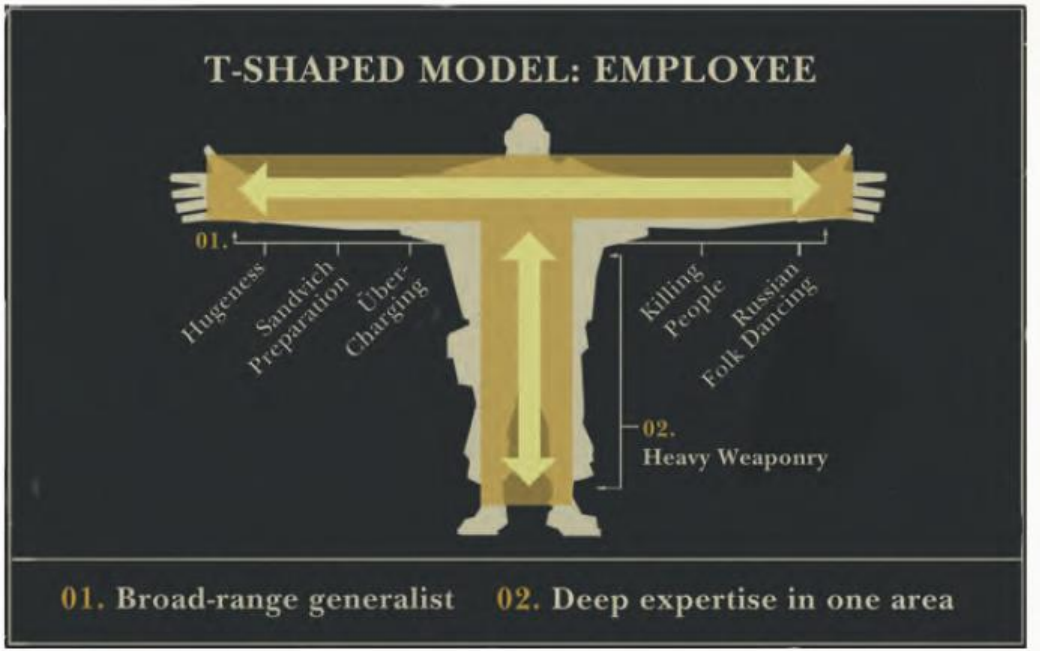
Sinergija

- Očito je da ako želimo biti efikasniji moramo raditi u timovima
- Timovi se ne rađaju, oni se stvaraju, treba vremena i truda da se stvori tim
  - Pseudo team
  - Potential team
  - Real team
  - High performing team



T-shaped people

- **We value “T-shaped” people.** That is, people who are both generalists (highly skilled at a broad set of valuable things—the top of the T) and also experts (among the best in their field within a narrow discipline—the vertical leg of the T). This recipe is important for success at Valve. We often have to pass on people who are very strong generalists without expertise, or vice versa. An expert who is too narrow has difficulty collaborating. A generalist who doesn’t go deep enough in a single area ends up on the margins, not really contributing as an individual.

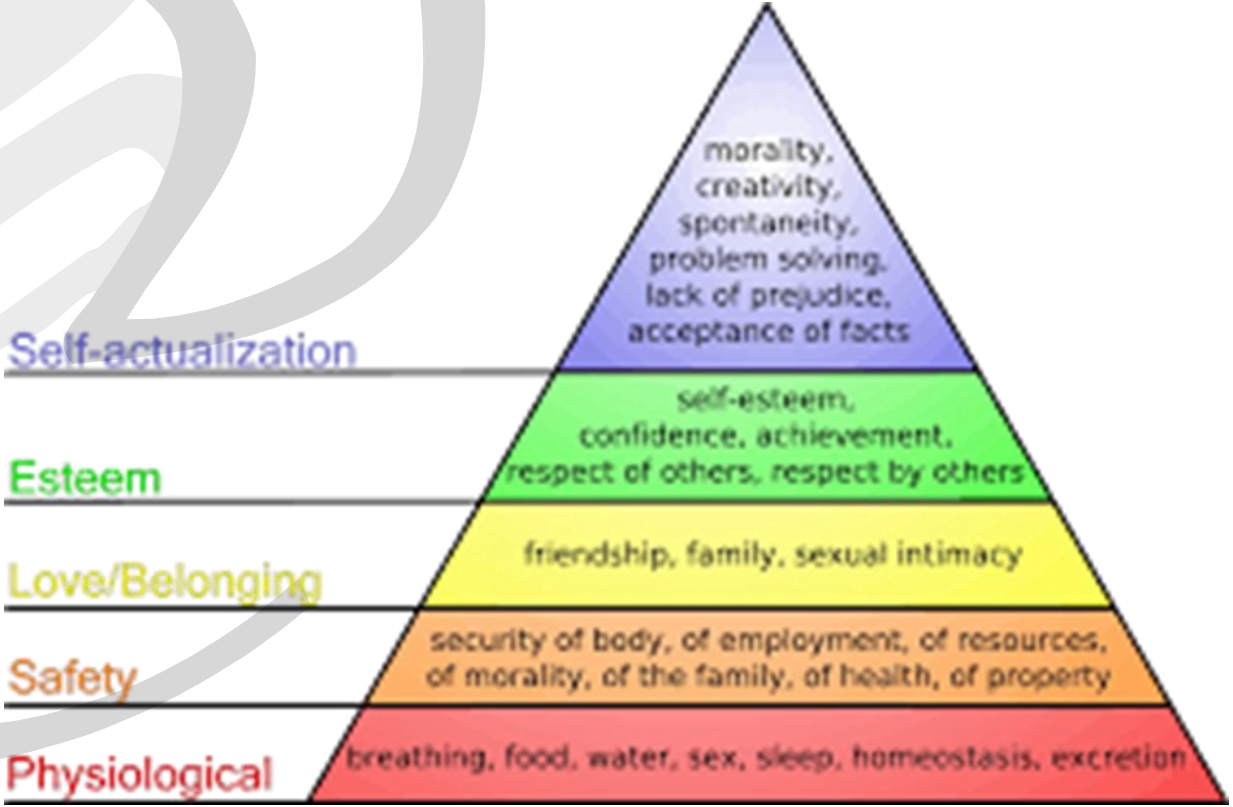


Motivacija članova tima

- 5. princip agilnog razvoja softvera:
  - Projekte ostvarujemo oslanjajući se na motivirane pojedince. Pružamo im okruženje i podršku koja im je potrebna i prepuštamo im posao s povjerenjem.
- Što motivira članove razvojnog tima?

Mrkva i batina…?

Maslowljeva piramida ljudskih potreba



Motivacija članova tima

- Ispostavlja se da motiviranost pojedinca potiče:
  - Autonomnost u obavljanju posla
  - Izvrsnost u nekom području
  - Svrha, odnosno cilj obavljanja posla

Kako Scrum podržava visoku motivaciju

- Autonomy
  - Self organization
  - Sprint planning
  - Daily standup
- Mastery
  - Iterative & incremental
  - Retrospectives
  - Continouous improvement
- Purpose
  - Product vision
  - Review

Agilne tehnike

- Pair programming – ključne dijelove programskog kôda obavljati u paru
- Refactoring – restrukturiranje postojećeg programskog kôda kojim se mijenja njegova unutarnja struktura a vanjsko ponašanje ostaje nepromijenjeno (bolje performanse, lakše održavanje i/ili nadogradnja)
- Test-driven Design – izrada testova kako bi se verificirala hipoteza. Glavna premisa Lean/Agile razvoja je eliminacija otpada – a što je veći otpad nego razvoj nečega što na kraju nećemo koristiti
- Code review – sistematska provjera programskog kôda umjesto odjela za provjeru ili vanjske revizije (jednako učinkovito ili gotovo jednako učinkovito, a brže i jeftinije)

Automatizirani testovi

- Potrebna je bolja suradnja testera i developera
  - možda bi tester trebao postati član tima? (za razliku od izdvojenog tima testera)
  - možda bi programski kôd trebalo pisati tako da ga je lako testirati, pa je pridruživanje testera timu developera jedan od načina da se postigne taj cilj
- Testovi se trebaju održavati
- Bolje imati manji broj testova koji se brzo izvršavaju i kojima se uspijevaju pronaći nedostaci u programskoj podršci nego velik broj testova koji se sporo obavljaju i ne pokrivaju sve slučajeve ili daju pogrešne rezultate
  - Možda ne morate sve testirati
- Važna je brzina izvođenja stoga imajte na umu paralelizam prilikom izgradnje testova
- Obavljanje testova zahtijeva zasebno i izdvojeno okruženje koje je slično, a po mogućnosti i identično, produkcijskom okruženju
  - Imamo dakle razvojno okruženje, testna okruženja, produkcijsko okruženje –> očito je nužno imati mogućnost brze i jednostavne uspostave okruženja u kojem se može pokrenuti sustav.

Continious delivery

- Mogućnost da se promjene u softweru (eksperimenti, opcije, promjene u konfiguraciji, popravci bugova) daju na korištenje na sigurno, brzo i održivo
- Sigurno – trebamo imati deployment pipeline koja će detektirati i odbaciti promjene koje su riskantne, sadrže loše ili neispravne mogućnosti ili dovode do neprihvatljivih performansi
- Brzo – kako bi to mogli raditi često i bezbolno, dobiti brzi feedback
- Održivo – treba biti ekonomski isplativo raditi u malim koracima. Mantra kontinuirane isporuke je „Ako boli, radi to češće i nemoj skrivati bol”

HP primjer

- Razvoj softvera (diver-a/firmware-a) za HP proizvode kočio je izdavanje novih verzija
- Razvoj u 3 vremenske zone nije pomogao
- Analizom poslova uočen je visok udio neproizvodnih aktivnosti
- Do 80% troškova podrške zbog pogreški koje nisu bile ispravljene
  - Loša kvaliteta
  - Riješeno automatiziranim testovi
- Kopija programskog kôda za svaku liniju proizvoda -> mukotrpno popravljanje uočenih pogrešaka
  - Riješeno s jednom verzijom programskog kôda za sve linije proizvoda

% troškova	Aktivnost
10%	Izrada verzija softvera (deployment)
20%	Detaljno planiranje
25%	Integracija programskog kôda
25%	Podrška
15%	Manualno testiranje
~5%	Inovacije

HP primjer

- Promjene nisu nastupile odmah
- Ustanovljeno je željeno stanje i iterirano prema njemu - PDCA
- 8x povećano vrijeme posvećeno inovacijama
- Razvojni troškovi smanjeni ~40%
- Broj programa u razvoju povećan je za 140%
- Troškovi razvoja pojedinog programa pali su za 78%

% troškova	Aktivnost	Prije
2%	Izrada verzija softvera (continious delivery)	10%
5%	Agilno planiranje	20%
15%	Integracija programskog kôda	25%
10%	Podrška	25%
5%	Manualno testiranje	15%
23%	Stvaranje i održavanje automatiziranih testova	0%
~40%	Inovacije	~5%

Lean organization

- Vjerojatno najveće zapreke agilnoj organizaciji su birokracija i menadžment koji su potrebni kako bi se uskladili output kojeg stvaraju timovi s očekivanjima klijenata i korporativnom strategijom
- Pojedinim timovima treba se dati odgovornost i autoritet da komuniciraju s korisnikom i isporuče mu proizvod bez bespotrebnog uplitanja unutar granica postavljenih kroz korisničke zahtjeve
- Najveći je problem povjerenje koje dolazi iz komunikacije i poštovanja (Manifest agilnog razvoja softvera)
- Scrum na nivou timova ima velik naglasak na komunikaciji - instrumentalizirana je. Scrum master bi ju trebao osigurati.
- Na sličan način može se (ako to ne ide neformalnim metodama) zahtijevati na nivou cijele organizacije

Korišteni resursi

- http://agilemanifesto.org/iso/hr/
- http://www.agilenutshell.com/
- http://www.agile42.com
- http://www.adammcfarland.com/2012/12/10/t-shaped-people/
- http://www.thecoachingtoolscompany.com
- https://www.pinterest.com/pascalvenier/complexity-cynefin-framework/

Agilni principi razvoja

Boris Matijašević

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

www.srce.unizg.hr