

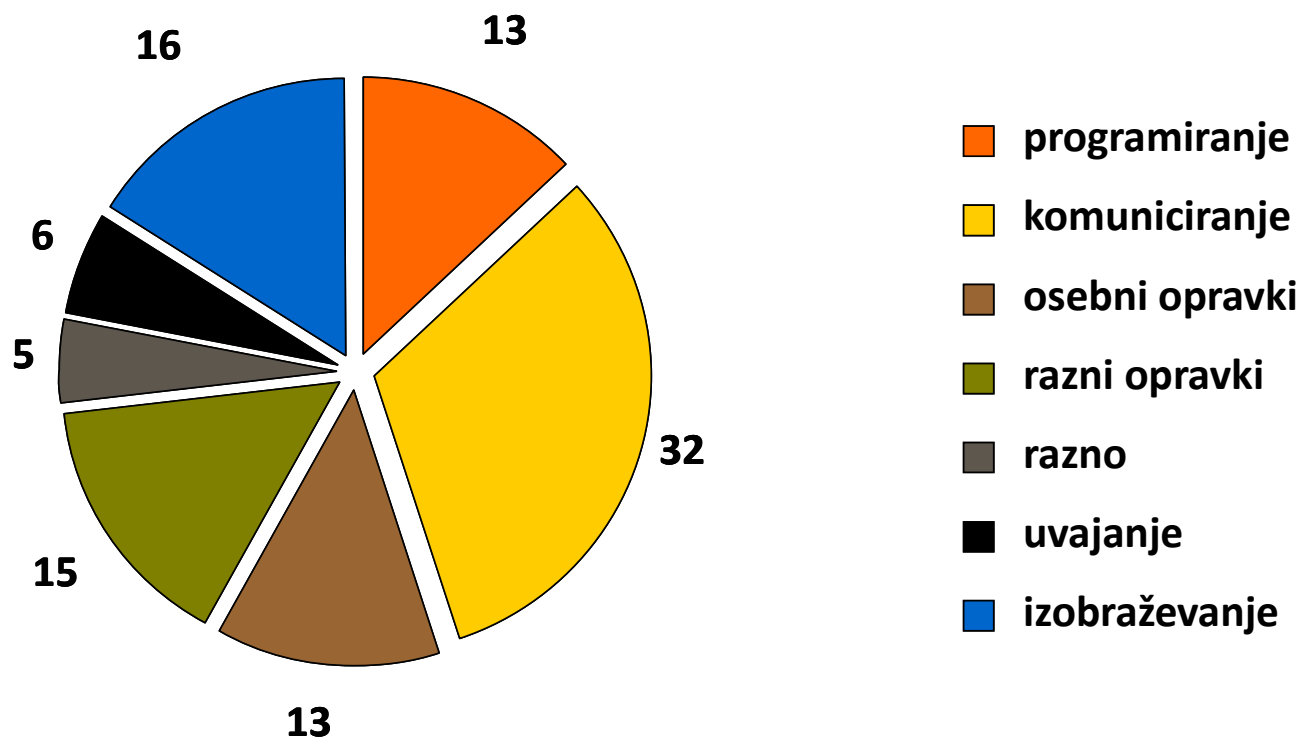
5. Faze razvoja IS

KAKO torej potekajo zadeve?

S čim se ukvarjamo pri razvoju IS?

- V okviru razvoja IS nas zanima, kako razviti računalniške rešitve, ki bodo čim bolje podprle delovanje IS.
- V okviru razvoja IS se tako ukvarjamo z:
 - Razvojem računalniške rešitve
 - Nabavo ustrezne strojne opreme
 - Namestitvijo systemske programske opreme
 - Uvedbo rešitve
 - Vzdrževanjem rešitve

Kaj zajema razvoj IS?



Kaj nas zanima pri razvoju IS?

- Razvoj IS ni zgolj programiranje!
- Ni zgolj tehnično inženirsko delo...

- Pri razvoju IS imajo velik pomen tudi sociološki dejavniki:
 - Kako dojemamo problematiko?
 - Kako razumemo potrebe uporabnikov?
 - Kako uvedemo rešitve v prakso?
 - ...

Vaja dela ...

- Primer: Izdelati želimo IR, ki bo omogočala oddaljeno izvajanje (preko spleta) novega študijskega programa.
- Kako bi opravili zastavljeno nalogo?
 - Česa bi se lotili najprej?
 - Ali bi morali pri tem s kom komunicirati, se posvetovati, sodelovati? Če da, s kom in na kakšen način?
 - Kateri bi bili najpomembnejši mejniki, s katerimi bi preverili, ali vaše delo dobro opravljate?

Faze razvoja IS

- Razvoj IS zajema številna opravila, običajno razdeljena v faze:



KAJ je potrebno narediti? Zahteve IS se pregledajo in strukturirajo.

KAKO bomo to naredili? Zahteve se zapišejo v formalni obliki dokumentacije (diagrami, dokumenti) in pripravi se načrt izvedbe.

Programiranje IS.

ALI obstajajo napake in KAJ ne dela prav?

Prenos IR v realno uporabo. Pravimo, da gre IS v produkcijo – za uporabo končnih uporabnikov.

Popravki in dopolnitve IS. Odpravljanje napak in nadgradnja / dodajanje novih funkcionalnosti.

Procesni modeli razvoja IS

- Procesni model razvoja IS pove, v kakšnem zaporedju in na kakšen način si v okviru razvoja IS sledijo posamezne faze – določa življenjski cikel razvoja IS.
- Kot večina razvojnih procesov sledi tudi razvoj IS določenemu **življenjskemu ciklu** oziroma razvojnemu modelu, ki določa **zaporedje faz razvoja**.

Procesni modeli razvoja IS

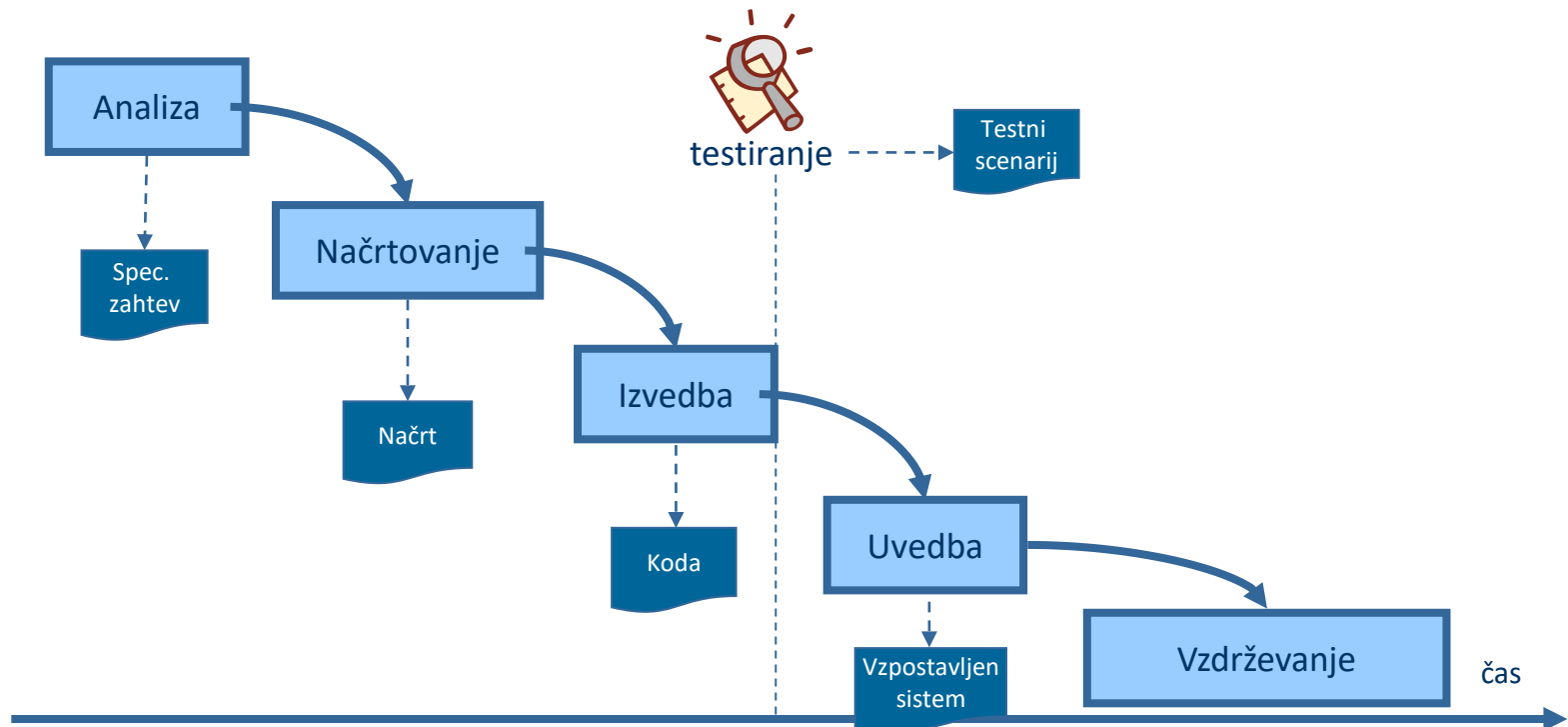
- Zagotovijo osnovo za pridobitev odgovorov na pet zelo pomembnih vprašanj:
 - **Planiranje** - kaj bomo naredili, da dosežemo zadane cilje?
 - **Pooblastila** - kako lahko vplivamo na dogajanje z namenom priti tja, kamor smo namenjeni?
 - **Napovedi** - kam bomo prišli oziroma kam gremo?
 - **Ocenitev** - kje v procesu smo in zakaj?
 - **Sledljivost** - kako smo dosegli rezultat - npr. kako je posamezen del kode povezan s poslovnimi cilji in z modeli v analizi?

Procesni modeli razvoja IS

- Poznamo različne procesne modele oz. načine razvoja IS:
 - Zaporedni pristopi (*waterfall*)
 - Kaskadni model
 - Inkrementalni model
 - Agilni pristopi (*agile*)
 - Iterativni model
 - Prototipni model
- V praksi se večinoma uporablja kombinacija različnih modelov.

Kaskadni model (1/3)

- Zaporedni ali kaskadni model temelji na zaporednem izvajanju faz.
- Ko se ena faza v celoti konča, se začne naslednja.



Kaskadni model (2/3)

□ Značilnosti:

- Najstarejši razvojni model, značilen za prve oblike strukturnega pristopa.
- Faze si sledijo zaporedno.
- Vračanje nazaj ni mogoče.
- Primeren za relativno kompleksne projekte, če zahteve dobro razumemo in se med projektom ne bodo bistveno spreminjale.
- Omogoča dobro in natančno projektno vodenje.

Kaskadni model (3/3)

□ Prednosti:

- Pomaga zmanjševati količino režijskega dela, ki ni v neposredni povezavi z izdelavo programske opreme (npr. vodenje projekta), saj je mogoče načrtovanje v celoti izvesti vnaprej.
- Možno uporabiti v kombinaciji z drugimi modeli.

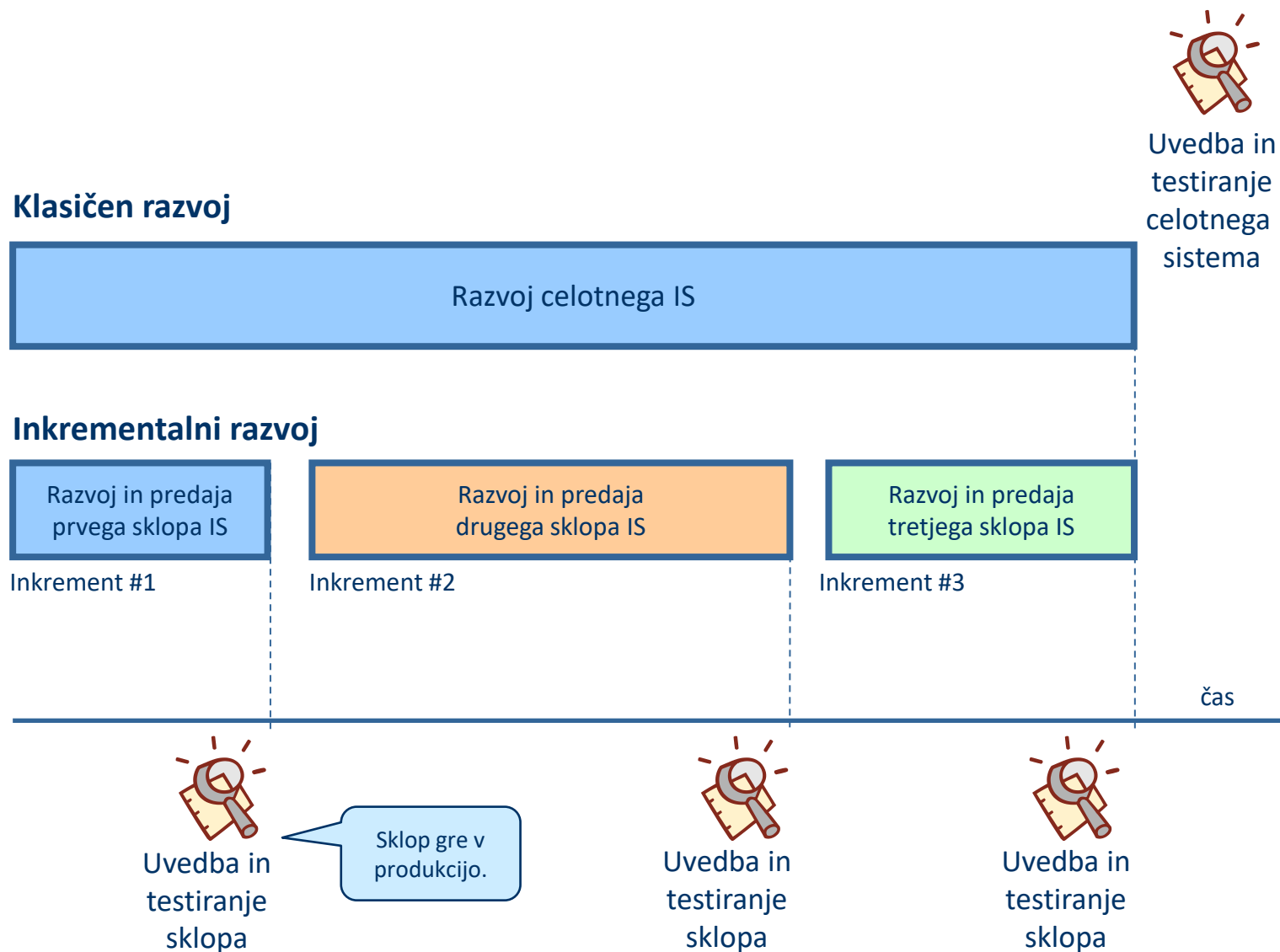
□ Slabosti:

- Ni fleksibilen. Vsaka naknadna sprememba zahteva veliko dodatnega napora.
- Nenaraven: v praksi je težko pričakovati, da se lahko nek postopek v celoti zaključi, preden se začne z naslednjim.
- Ne omogoča paralelnega izvajanja delov postopkov.

Inkrementalni model (1/3)

- Temelji na postopni gradnji celotnega IS in sproti predaji posameznih **inkrementov** (delov) uporabniku.
 - Inkrement predstavlja zaokroženo funkcionalnost sistema (sklop, podsistem, modul).
- Ne razvijamo celotnega IS hkrati.
 - Omejimo se na posamezen sklop, ki ga razvijemo v celoti in predamo uporabniku.
- Ob predaji novi sklop povežemo z ostalimi sklopi.
- Inkremente je moč razvijati tudi vzporedno.

Inkrementalni model (2/3)



Inkrementalni model (3/3)

□ Prednosti:

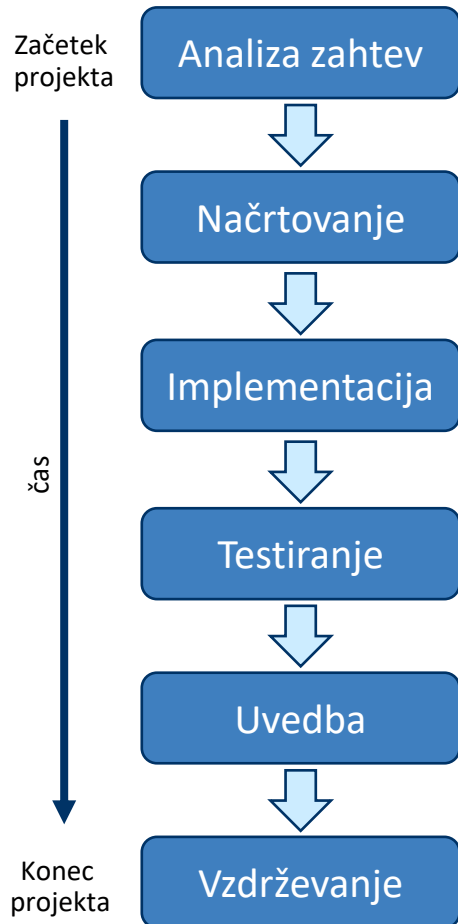
- Uporabnik prej dobi del zahtevane IR, saj se IR razvija po delih.
- Rešitev, ki jo uporablja, se postopoma nadgrajuje, sam pa lahko sodeluje pri testiranju razvitih sklopov.
- Naročnik lažje sledi napredovanju projekta.

□ Slabosti:

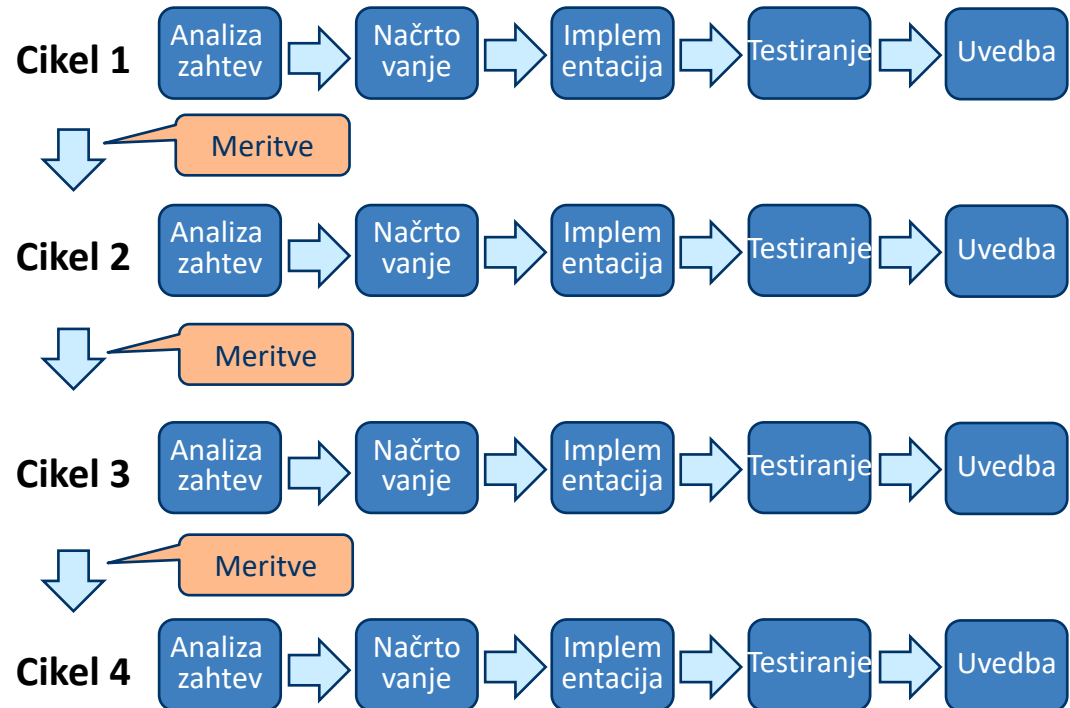
- Ni mogoče uporabiti pri vseh projektih. Nekaterih rešitev ni moč predati v uporabo po delih.
- IR moramo razdeliti na sklope in predvideti odvisnosti med njimi. Pri neustreznem načrtovanju se lahko zgodi, da sklope neustrezno razporedimo.

Zaporedni vs. agilni pristopi

Zaporedni razvoj



Agilni razvoj



Agilni pristopi ^(1/2)

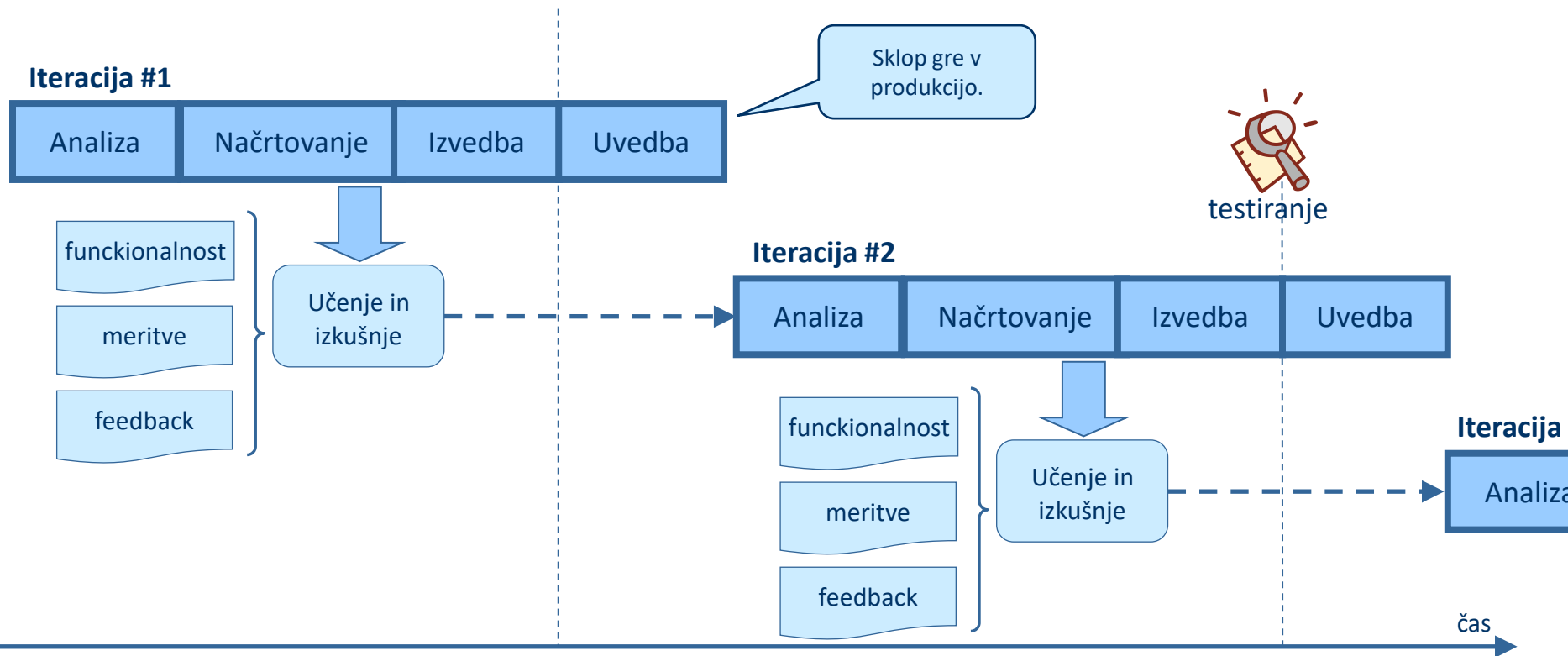
- Osnovna načela oz. vrednote agilnosti:
 - Posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja.
 - Delujoča programska oprema je pomembnejša kot popolna dokumentacija.
 - Vključevanje (sodelovanje) uporabnika je pomembnejše kot pogajanja na osnovi pogodb.
 - Upoštevanje sprememb je pomembnejše kot sledenje planu.

Agilni pristopi (2/2)

- Želimo hiter *feedback* o uporabnosti IS → cilj je izdelati najmanjši uporaben produkt (*minimal viable product*, MVP).
 - Pri zaporednem razvoju delamo IS od začetka do konca in ga šele nato damo v produkcijo.
 - Pri agilnem razvoju damo v produkcijo že delne izdelke.
- Da lahko dobimo *feedback*, moramo spremljati (meriti) uporabo našega IS.
 - Pri zaporednem razvoju ne merimo uporabe.
 - Pri agilnem razvoju je meritev uporabe izredno pomembna!
- Del ali celoten sistem se razvija po ciklih.
 - Bodisi se po ciklih dodajajo funkcionalnosti (iterativni model) ali se po ciklih dodeluje sistem v celoti (prototipni model).

Iterativni model (1/4)

- Razvit kot odziv na pomanjkljivosti kaskadnega pristopa.
- Faze razvoja izvajamo v več **iteracijah**.



Iterativni model (2/4)

□ Značilnosti:

- V vsaki iteraciji razvijemo določen del funkcionalnosti celotnega sistema.
- Iteracija gre navadno čez vse faze razvoja: analizo, načrt, izvedbo, ... vendar ne z enako intenzivnostjo
- Naslednja iteracija se lahko začne šele takrat, ko je prejšnja končana.
- Vsebina naslednje iteracije je določena na osnovi rezultatov prejšnje.
- V začetnih iteracijah razvijemo najbolj tvegane dele sistema.
- Med izvajanjem iteracije ne sprejemamo sprememb.
- Gre za **evolucijski razvoj**.

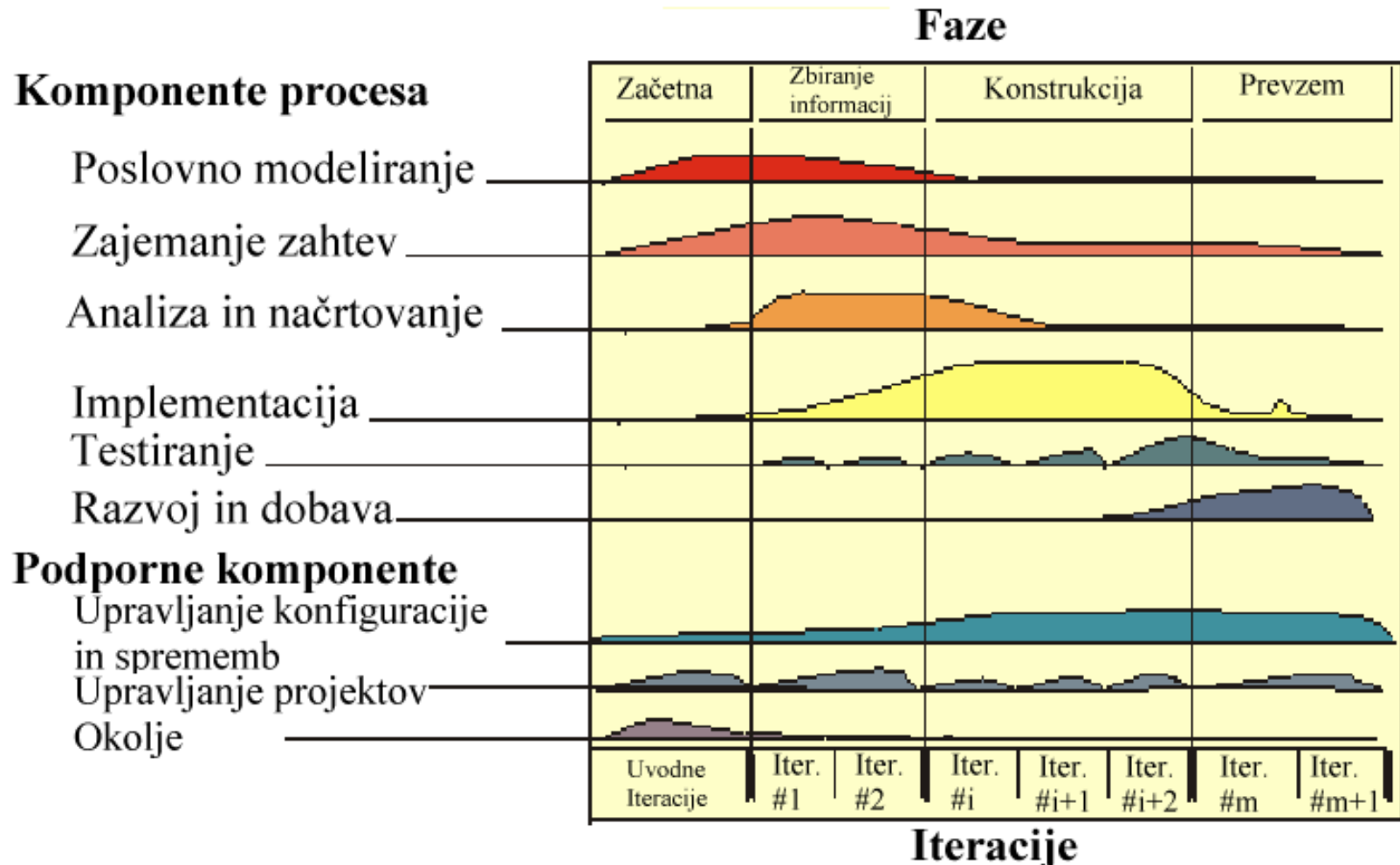
Iterativni model ^(3/4)

- Prednosti (proti zaporednemu):
 - Najbolj tvegani deli so razrešeni še preden postane investicija velika.
 - Začetne iteracije omogočijo zgodnje povratne informacije s strani uporabnikov.
 - Preizkušanje in povezovanje v sistem sta nepretrgana.
 - Ciljni mejniki omogočajo kratkoročno osredotočenje.
 - Možna je predaja izvedenega dela projekta še preden je dokončan celoten projekt.

Iterativni model (4/4)

- Slabosti:
 - Ne omogoča dobrega načrtovanja poteka projekta.
 - Ni mogoče točno predvideti, koliko iteracij bo potrebnih za razvoj dokončnega (dovolj dobrega) izdelka.
 - Vodenje projekta je zahtevno.

Primer: RUP (Rational Unified Process)

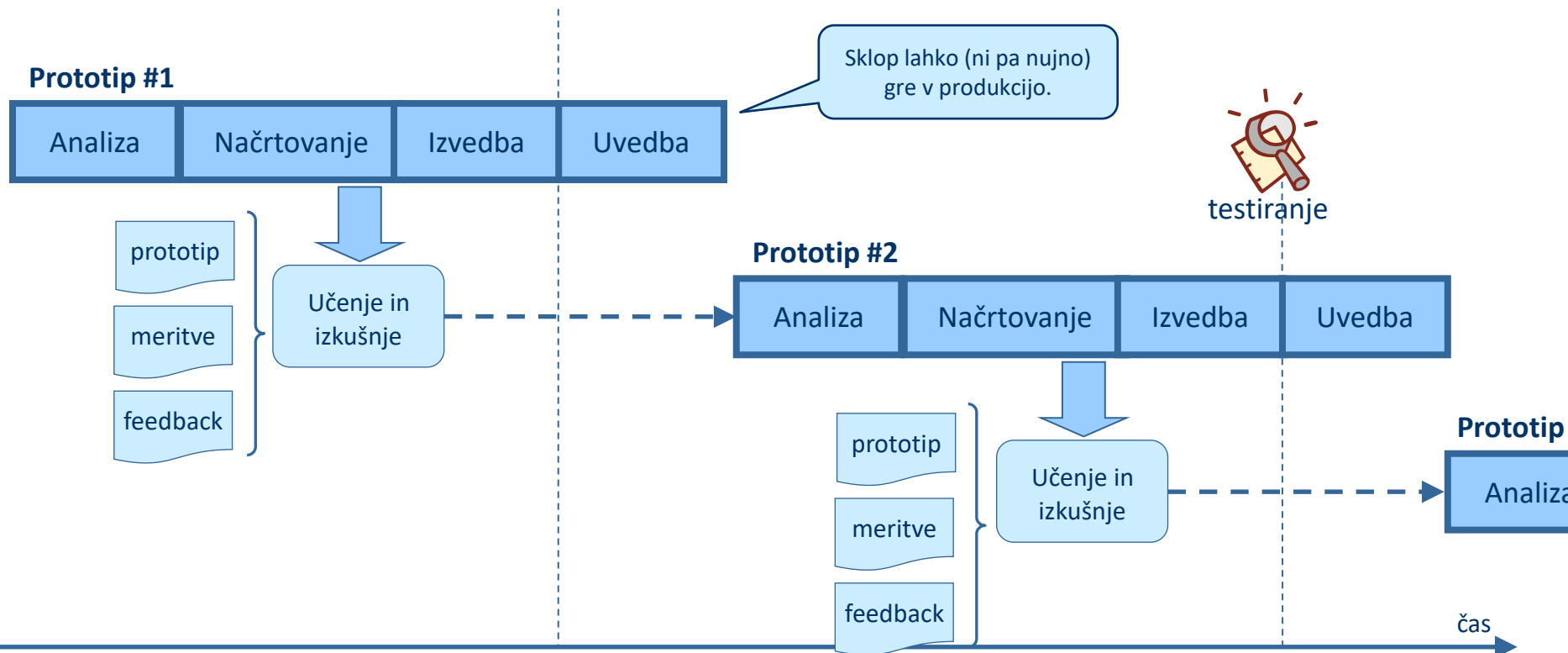


Prototipni model (1/5)

- Gre za različico iterativnega modela.
- Temelji na izdelavi **prototipov** in njihovi postopni izboljšavi, dokler ne dosežemo zadovoljive kakovosti.
- Prototip označuje predhodno izdelane in navadno še nepopolne različice sistema ali dela sistema.

Prototipni model (2/5)

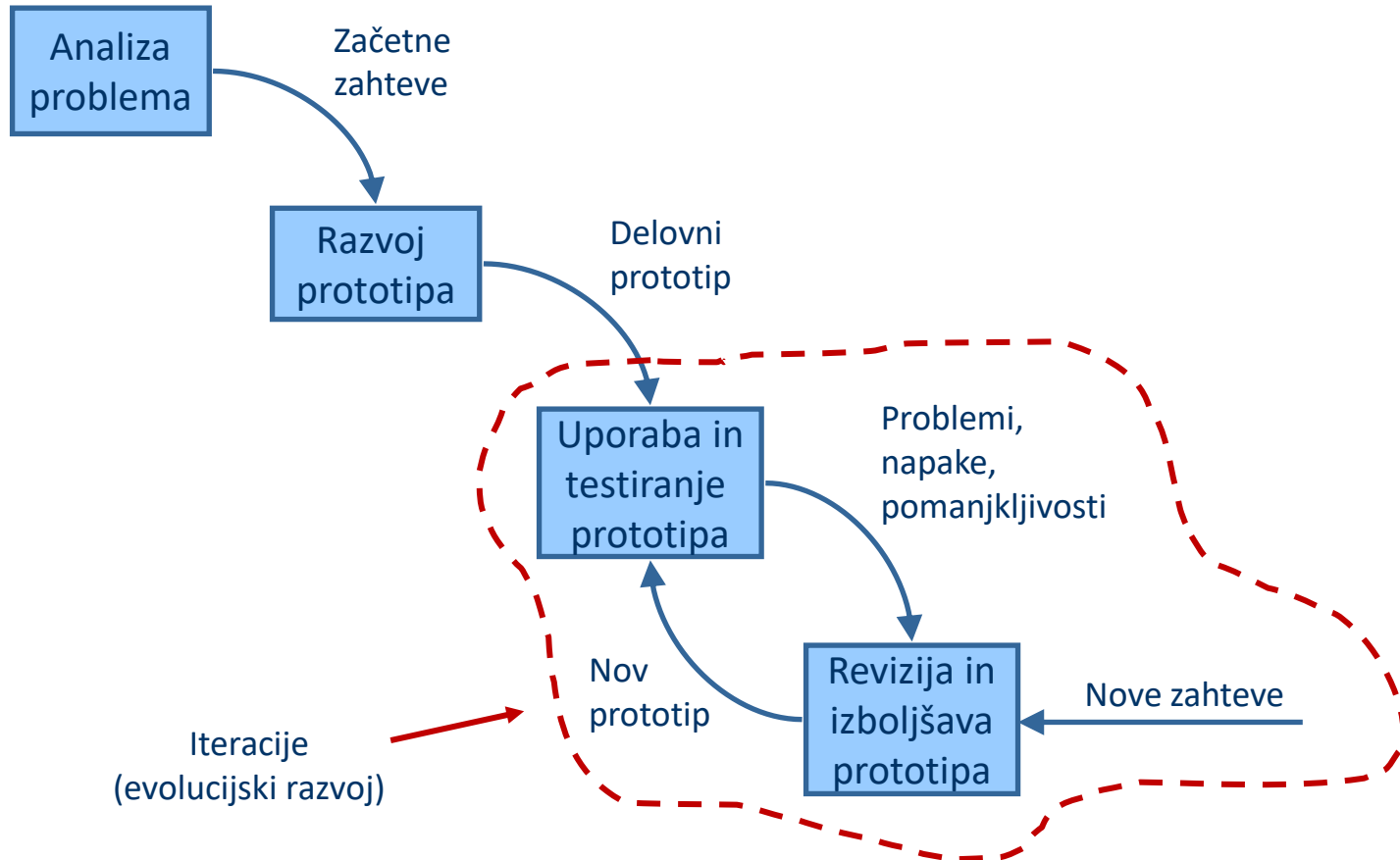
- Faze razvoja izvajamo v več verzijah **prototipov**.



Prototipni model (3/5)

- Značilnosti:
 - Temelji na izdelavi prototipov celotnega IS in njihovi postopni izboljšavi, dokler ne dosežemo zadovoljive kakovosti.
 - Prototip označuje predhodno izdelane in navadno še nepopolne različice IS.
 - Za razliko od iterativnega modela tukaj ni postopnih dodajanj funkcionalnosti, ampak postopno dograjujemo IS v celoti.
 - Prve prototipne verzije se uporabijo le za interno uporabo in testiranje. Kasnejše verzije se že predajo v “produkcijsko” uporabo.

Prototipni model (4/5)



Prototipni model (5/5)

□ Prednosti:

- Začetne iteracije omogočijo zgodnje povratne informacije s strani uporabnikov.
- Hitra ugotovitev, če ima IS v celoti sploh tržni potencial.

□ Slabosti:






- Prve (nepopolne) verzije prototipa lahko odvrnejo potencialne uporabnike IS.
- Zaradi razvoja prototipa je lahko zasnova zgrešena (površna, nedosledna, premalo „inženirska“) in posledično obstaja nevarnost za nujnost ponovnega razvoja od začetka.
- Težko je določiti, kdaj je sistem prišel iz “prototipne” verzije v “produkcijsko” verzijo, ki je primerna za redno uporabo.

Kako izbrati primeren način razvoja?

- Ni enostavnega (in tudi ne edinega pravilnega) odgovora.
- Višja stopnja formalnosti metodologije je primerna za projekte z višjo stopnjo kritičnosti.
 - Zaporedni pristopi → več dokumentacije in s tem tudi več formalnosti.
- Večje razvojne skupine potrebujejo bolj dokumentirane metodologije.
 - Pri majhnih skupinah je nepotrebna dodatna teža metodologije časovno predraga.
- Čim večje so discipliniranost, izkušnje in znanje razvojne skupine, tem manjša je potreba po podrobno definiranem procesu, formalnosti in dokumentaciji.
 - Agilni pristopi = manj dokumentacije.

Primer: Kanban

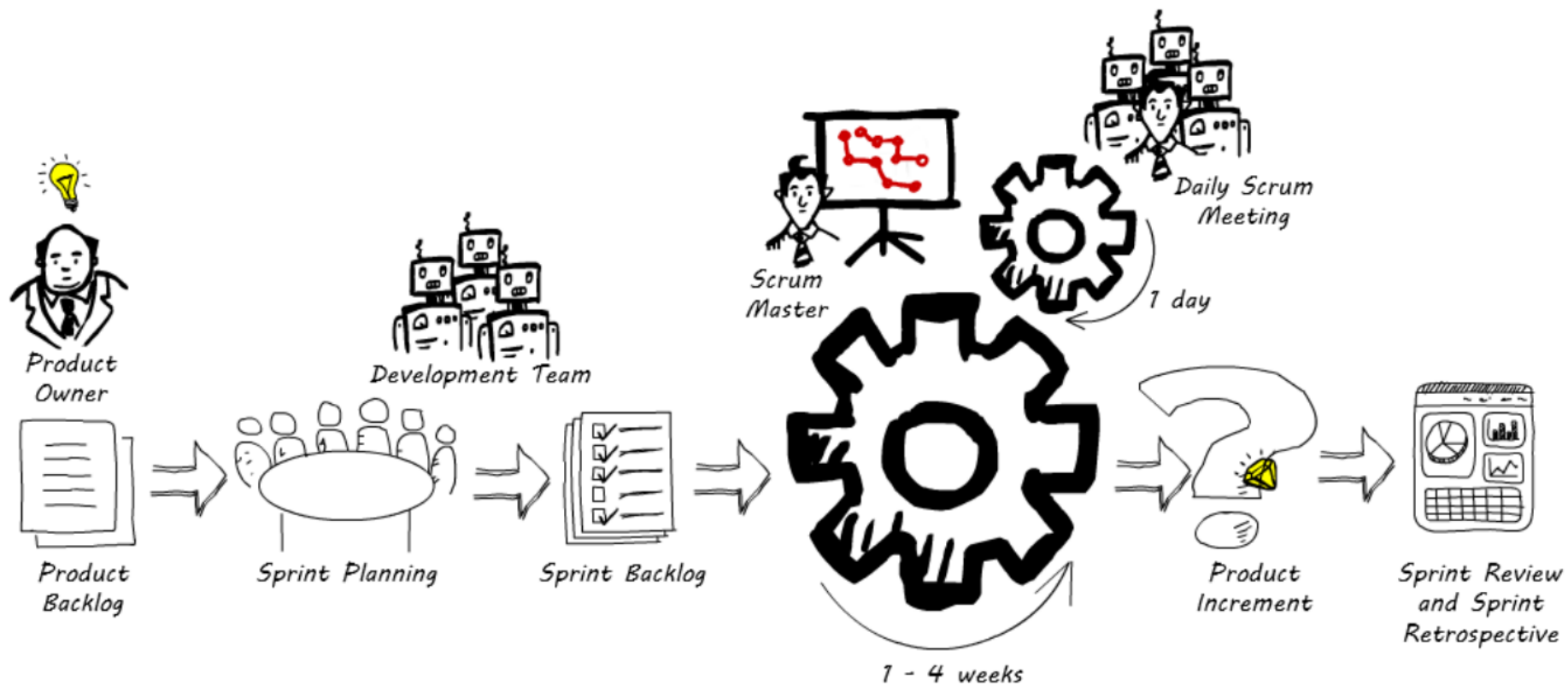
- Gre za **najpreprostejšo agilno metodologijo** iterativnega modela, ki od uporabnika zahteva zgolj tri stvari:
 - Poznati moramo svoj delovni proces do te mere, da se lahko odločimo, katere stvari so v fazi čakanja na izvedbo, katere stvari so v delu in kaj točno pomeni, da je neka stvar zaključena. Seveda je lahko faz poljubno mnogo.
 - Število nalog je v posamezni fazi najbolje omejiti na čim manjše število.
 - Čim boljša definicija stanja “narejeno” na posamezni nalogi.
- Kanban je primeren za ekipe do 10 oseb in tudi za posameznike (kot orodje za osebno produktivnost) ter ko gre za “raziskovalni projekt” ali ko ni točno določenega končnega roka.

TO-DO	SELECTED 5	DEVELOPMENT 3		IN ACCEPTANCE	COMPLETED
		IN PROGRESS	DONE		
					
					
					
					
					

Primer: Scrum

- Je **agilna metodologija iterativnega modela**, ki pomaga ekipam reševati kompleksne probleme.
 - Ekipa vsakih nekaj tednov pripravi novo verzijo produkta, ki ima dodano neko novo funkcionalnost. To verzijo ob zaključku iteracije pokažemo uporabnikom in ostalim deležnikom z namenom, da takoj dobimo povratno informacijo, ki jo upoštevamo pri načrtovanju dela za naslednjo iteracijo.
- Scrum **ekipe so majhne in samoorganizirane**, brez projektne vodje, ki bi sam sprejemal odločitve, ampak se njegove naloge porazdelijo med vse člane ekipe.
 - Ekipi se popolnoma zaupa, da bo dosegla zastavljen cilj na način, za katerega misli, da je najboljši in da bo hkrati prevzela vso odgovornost za svoje delo.
- Za razliko od Kanbana iteracije niso poljubno dolge, ampak so fiksno dolge po vnaprej določenem času (največkrat 4 tedni) – imenujemo jih sprinti.
- Scrum je primeren, ko imamo več neodvisnih majhnih ekip in ko imamo rok končanja razvoja IS točno določen.

Primer: Scrum



Primer: ekstremno programiranje

- Povratne informacije (*fine-scale feedback*)
 - Programiranje v parih; Igra načrtovanja; Razvoj, ki temelji na testiranju; Ekstremne programske prakse
- Stalni (nenehni) proces (*continuous process*)
 - Stalna integracija; Izboljšanje načrtovanja (*refactoring*); Majhne izdaje
- Skupno razumevanje (*shared understanding*)
 - Standardi kodiranja; Skupno lastništvo kode; Enostavna zasnova; Sistemska metafora
- Programerska dobrobit (*programmer welfare*)
 - Trajnostni tempo

