Ohjelmistotuotanto

Luento 4

8.11.2018

Vaatimusmäärittely ketterässä prosessimallissa nopea kertaus

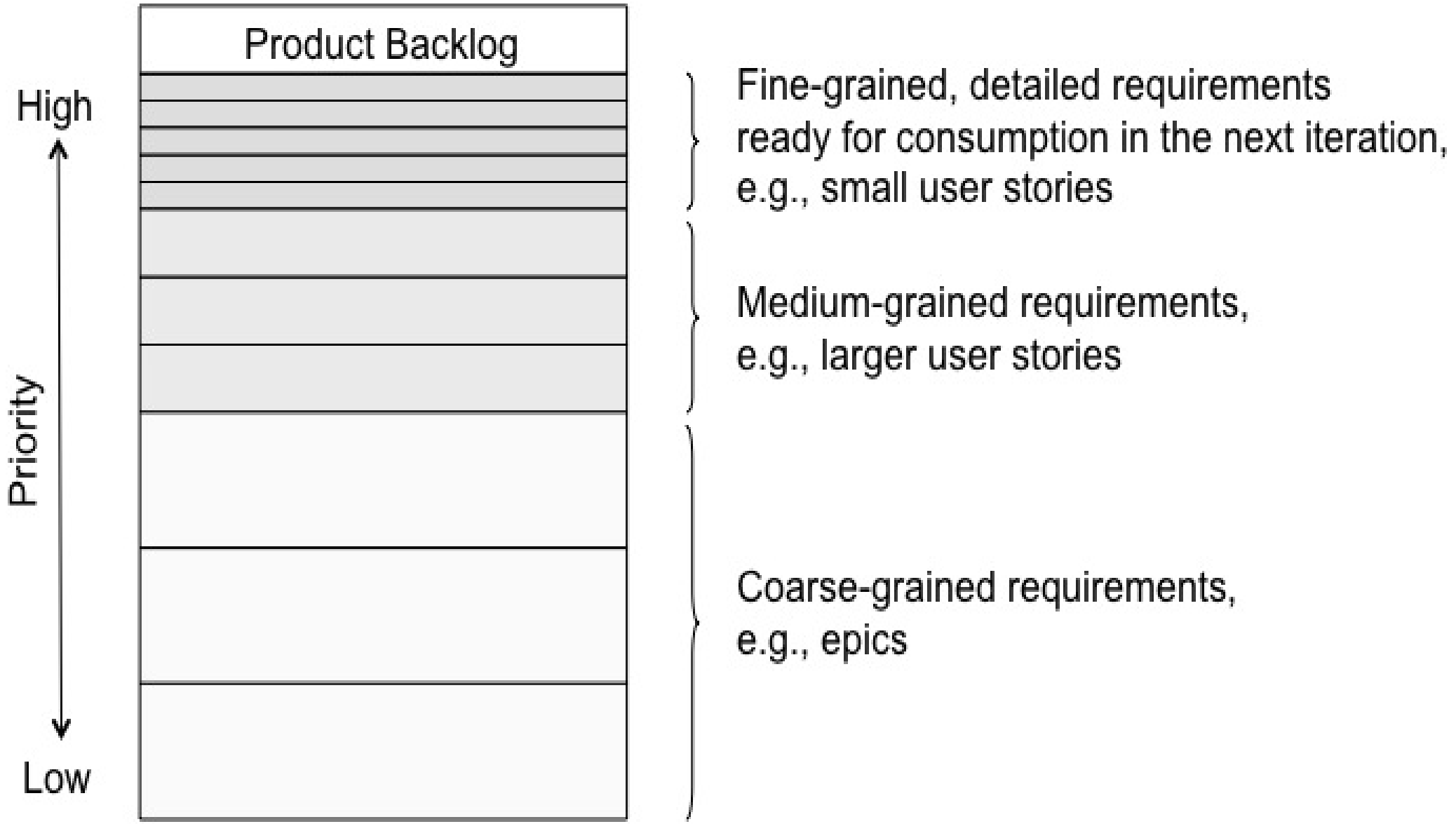
Nopea kertaus tiistailta

- User story
  - ”Määritelmä”:
    - description
    - conversations
    - tests (acceptance criteria)
  - INVEST
- Estimointi
  - Miksi?
  - Miten?
  - Kuka?
- Product Backlog
  - Kuka vastuussa?
  - Miten saadaan projektin alussa muodostettua?

Hyvä product backlog on DEEP

- http://www.romanpichler.com/blog/product-backlog/making-the-product-backlog-deep/
- Mike Cohn lanseerasi lyhenteen DEEP kuvaamaan hyvän backlogin ominaisuuksia
  - **D**etailed appropriatly
  - **E**stimated
  - **E**mergent
  - **P**rioritized
- **D**etailed appropriately eli sopivan detaljoitu:
  - Backlogin prioriteeteiltaan korkeimpien eli pian toteutettavaksi otettavien User Storyjen kannattaa olla suhteellisen pieniä ja näin tarkemmin estimoituja
  - Alemman prioriteetin User Storyt voivat vielä olla isompia ja karkeammin estimoituja

Hyvä product backlog on DEEP



Hyvä product backlog on DEEP

- DEEP ominaisuuksista **estimated** ja **prioritized** ovat meille tuttuja
- **Emergent** kuvaa backlogin muuttuvaa luonnetta:
  - The product backlog has an organic quality. It evolves, and its contents change frequently. New items emerge based on customer and user feedback, and they are added to the product backlog. Existing items are modified, reprioritized, refined, or removed on an ongoing basis.
- Muuttuvan luonteensa takia *backlogia tulee hoitaa* (**backlog grooming)** projektin edetessä
  - Backlogiin lisätään uusia User storyja ja vanhoja tarpeettomaksi käyneitä poistetaan
  - Isoja User storyja pilkotaan tarpeentullen pienemmiksi (erityisesti prioriteetin kasvaessa täytyy isot Storyt jakaa useaksi pienemmäksi)
  - Backlogiin lisättäviä uusia User storyjä estimoidaan ja vanhojen Storyjen estimaatteja tarkastetaan ymmärryksen kasvaessa
  - Backlogin hoitamiseen osallistuu koko ohjelmistotuotantotiimi, pääasiallinen vastuu on Product Ownerilla
- http://www.romanpichler.com/blog/product-backlog/grooming-the-product-backlog/

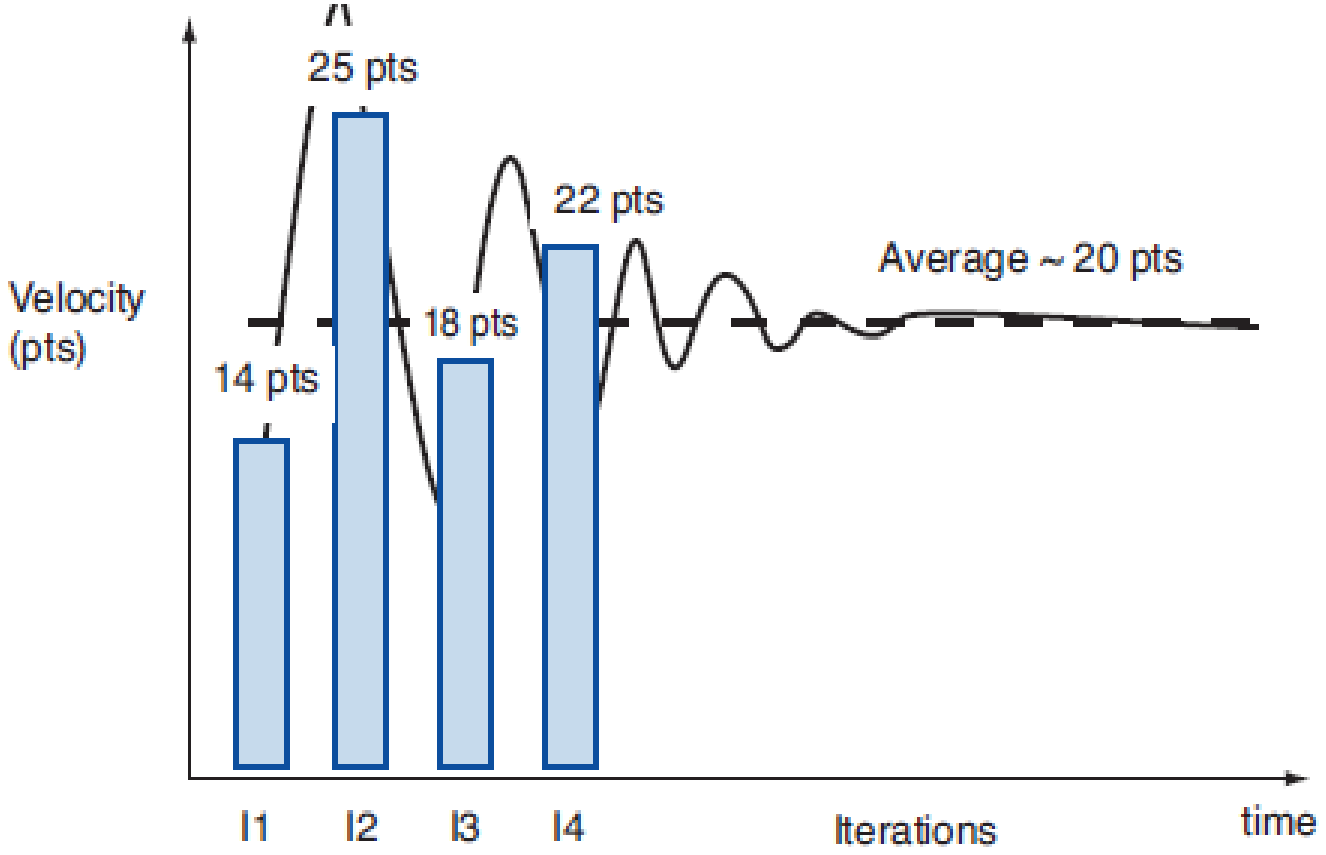
Julkaisun suunnittelu – release planning

- Estimoinnin toinen tarkoitus on, että se **mahdollistaa koko projektin viemän aikamäärän summittaisen arvioinnin** eli julkaisun suunnittelun (engl. release planning)
- Jos estimoinnin yksikkönä kuitenkin on abstrakti käsite *Story point*, miten estimaattien avulla on mahdollista arvioida projektin viemää aikamäärää?
- Kehitystiimin **velositeetti** (engl velocity) tarjoaa osittaisen ratkaisun tähän
- Velositeetilla tarkoitetaan Story pointtien määrää, minkä verran tiimi pystyy keskimäärin toteuttamaan yhden sprintin aikana
- Jos tiimin velositeetti on selvillä ja projektissa toteutettavaksi tarkoitetut User storyt on estimoitu, on helppo tehdä alustava arvio projektin viemästä aikamäärästä

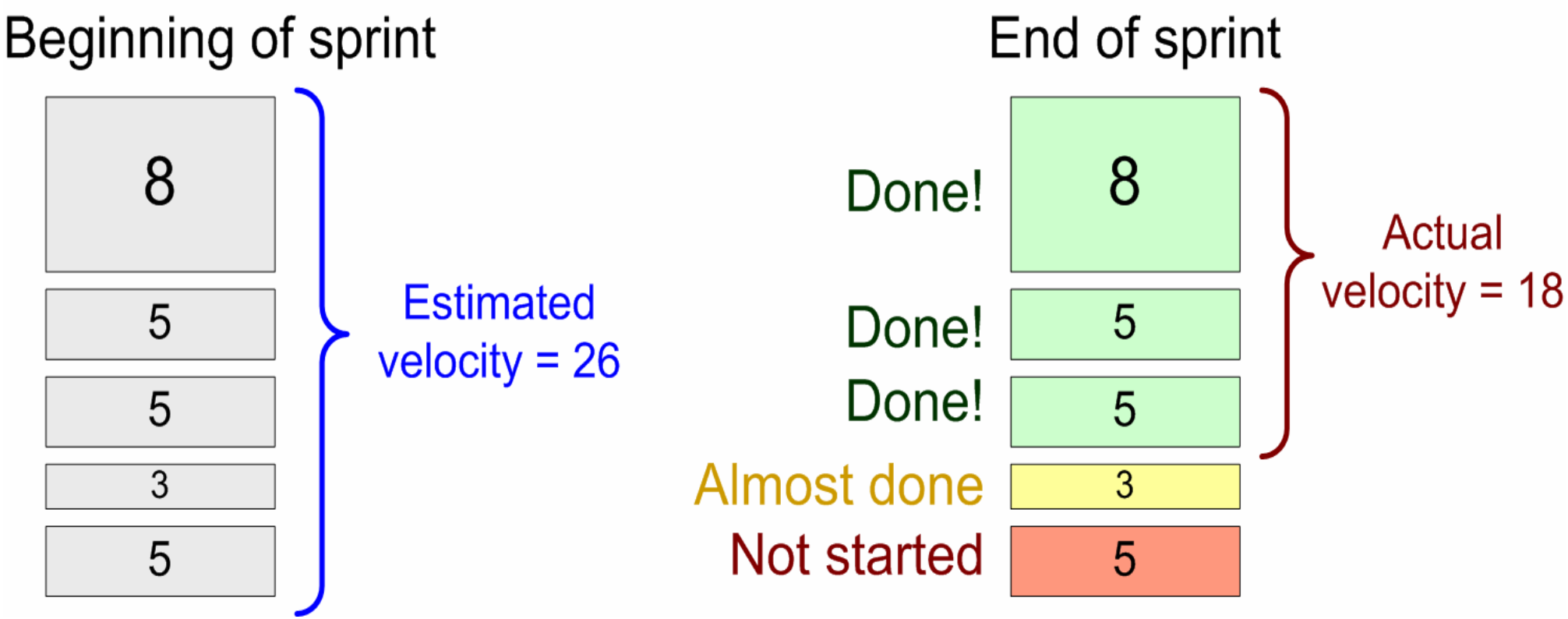
  (User storyjen estimaattien summa) / velositeetti * sprintin pituus
- Projektin alkaessa velositeetti ei yleensä ole selvillä, ellei kyseessä ole jo yhdessä työskennellyt tiimi
- On kehitetty tapoja joiden avulla velositeetti voidaan yrittää etukäteen ennustaa
  - Hyvin epäluotettavia, emme käsittele niitä nyt

Velositeetti

- Velositeetti vaihtelee tyypillisesti alussa melko paljon ja alkaa stabiloitumaan vasta muutaman sprintin päästä
  - Estimointi on aluksi vaikeampaa varsinkin jos sovellusalue ja käytetyt teknologiat eivät ole täysin tuttuja
- Tiimin velositeetti ja siihen perustuva projektin keston arvio alkaakin tarkentumaan pikkuhiljaa

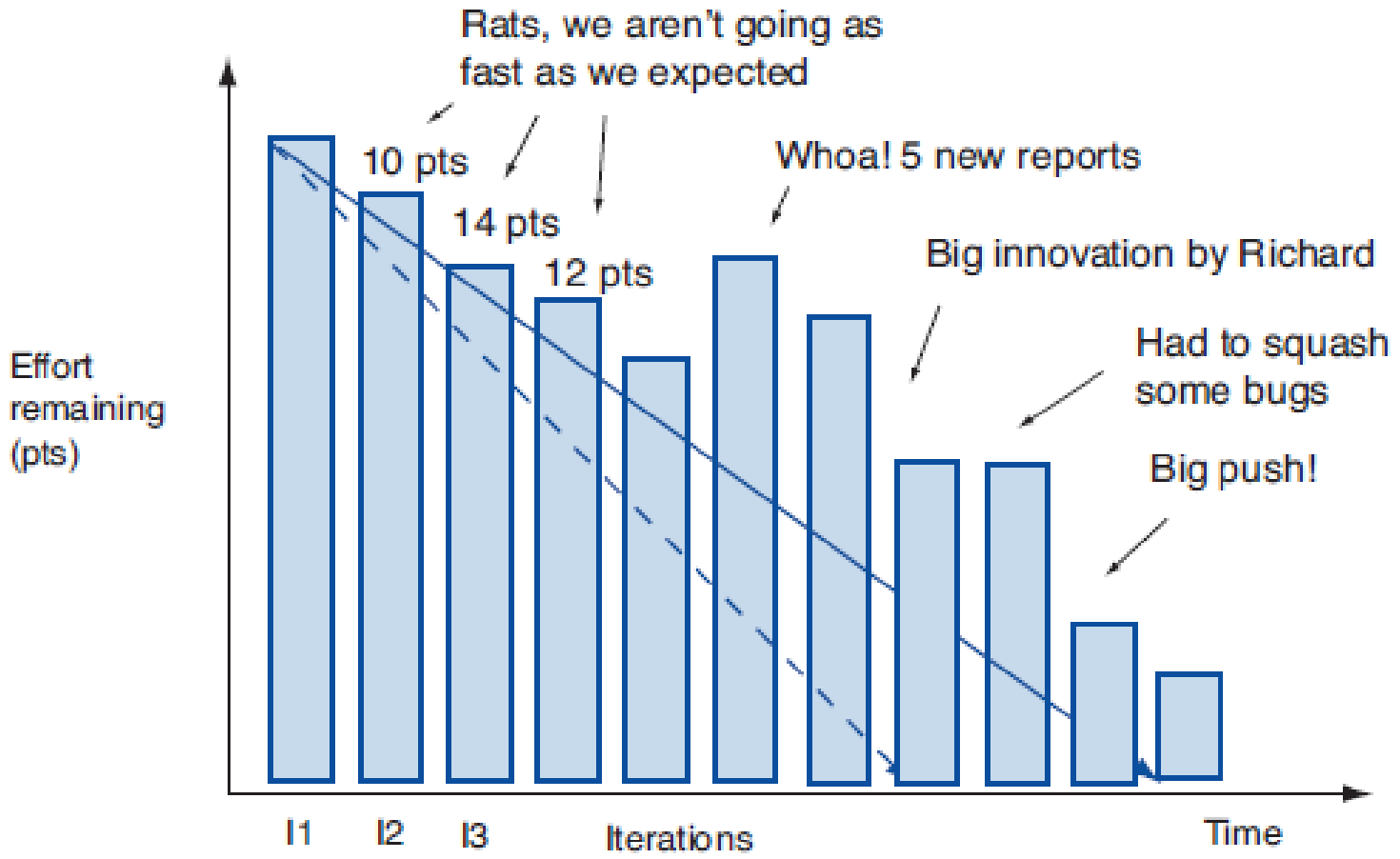


- Ketterissä menetelmissä on oleellista kuvata mahdollisimman realistisesti projektin etenemistä
- Tämän takia velositeettiin lasketaan mukaan ainoastaan **täysin valmiiksi** (eli Definition of Donen mukaisesti) toteutettujen User storyjen Story pointit
  - ”lähes valmiiksi” tehtyä työtä ei siis katsota ollenkaan tehdyksi työksi
  - http://jamesshore.com/Agile-Book/done_done.html



Julkaisun suunnittelu – release planning

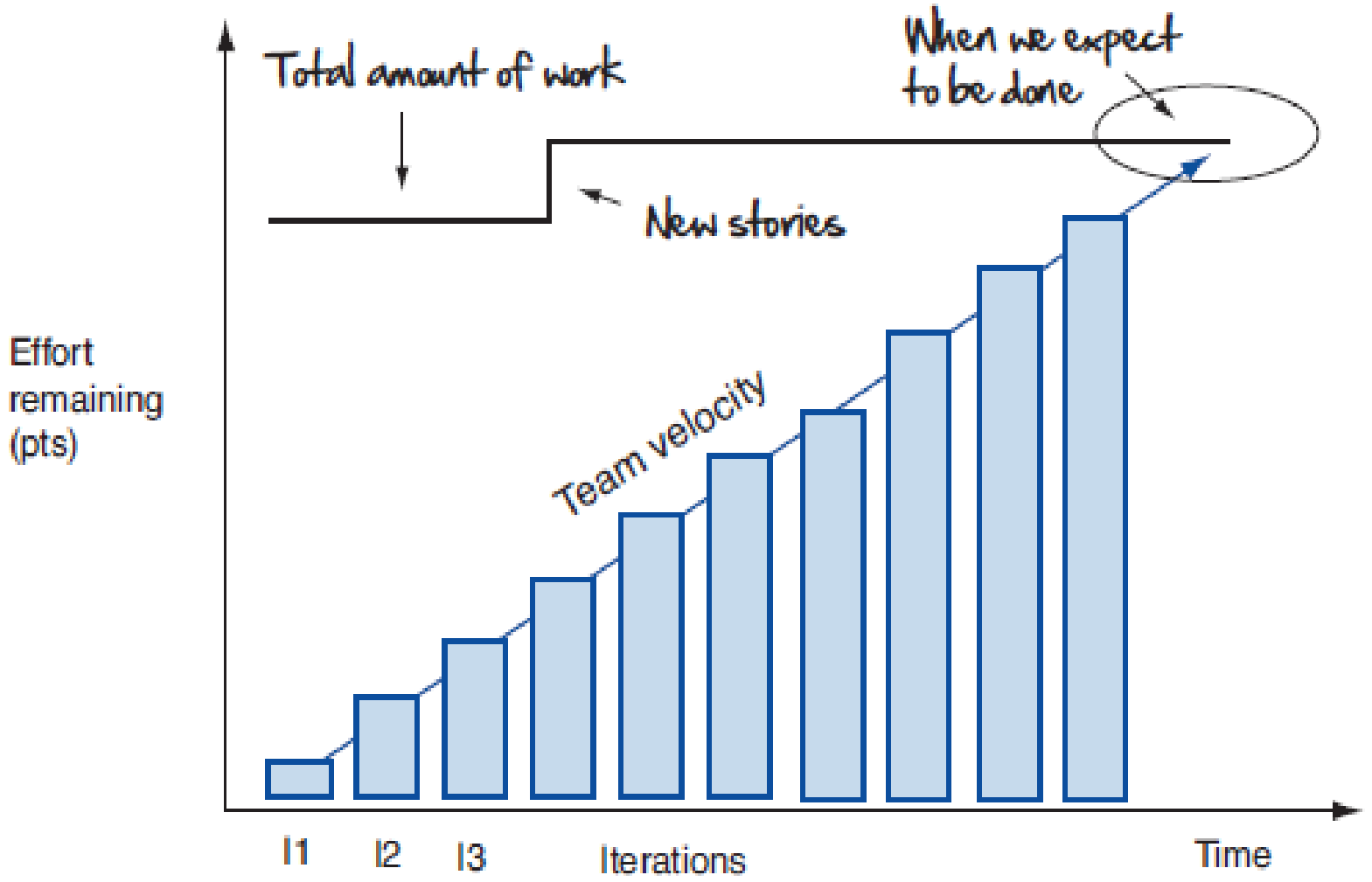
- Ketterän projektin etenemistä kuvataan yleensä Release Burndown -kaavion avulla
  - Aika etenee x-akselissa sprintti kerrallaan
  - y-akselilla on **jäljellä olevan työn määrä** story pointteina mitattuna



- Ketterässä projektissa vaatimukset saattavat muuttua kehitystyön aikana, siksi jäljellä olevan työn määrä ei aina vähene

Julkaisun suunnittelu – release planning

- Joskus käytetäänkin Burn Up -kaavioita joka tuo selkeämmin esiin kesken projektin etenemisen tapahtuvan työmäärän kasvun



Sprintin suunnittelu

Sprintin/iteraation suunnittelu

- Kertauksena viime viikolta: *Scrum määrittelee pidettäväksi ennen jokaista sprinttiä suunnittelupalaverin*
- Palaverin ensimmäinen tavoite on selvittää **mitä sprintin aikana tehdään**
  - Product Owner esittelee Product backlogin kärjessä olevat vaatimukset
  - Tiimin on tarkoitus olla riittävällä tasolla selvillä mitä vaatimuksilla tarkoitetaan
  - Tiimi valitsee tehtäväksi niin monta Backlogin storyistä kuin se arvioi kykenevänsä sprintin aikana toteuttamaan Definition of Donen määrittelemällä laatutasolla
- Sprintin aikana toteutettavien vaatimusten lisäksi asetetaan *sprintin tavoite*
- Suunnittelukokouksen toisena tavoitteena **miten sprintin tavoitteet saavutetaan**
  - Tiimi suunnittelee toteutettavaksi valitut vaatimukset tarvittavalla tasolla
- Tarkennetaan nyt Sprintin suunnitteluun ja läpivientiin liittyviä asioita
  - Lähteenä Kniberg Scrum and XP From the Trenches, luvut 3-6

Sprintin suunnittelu

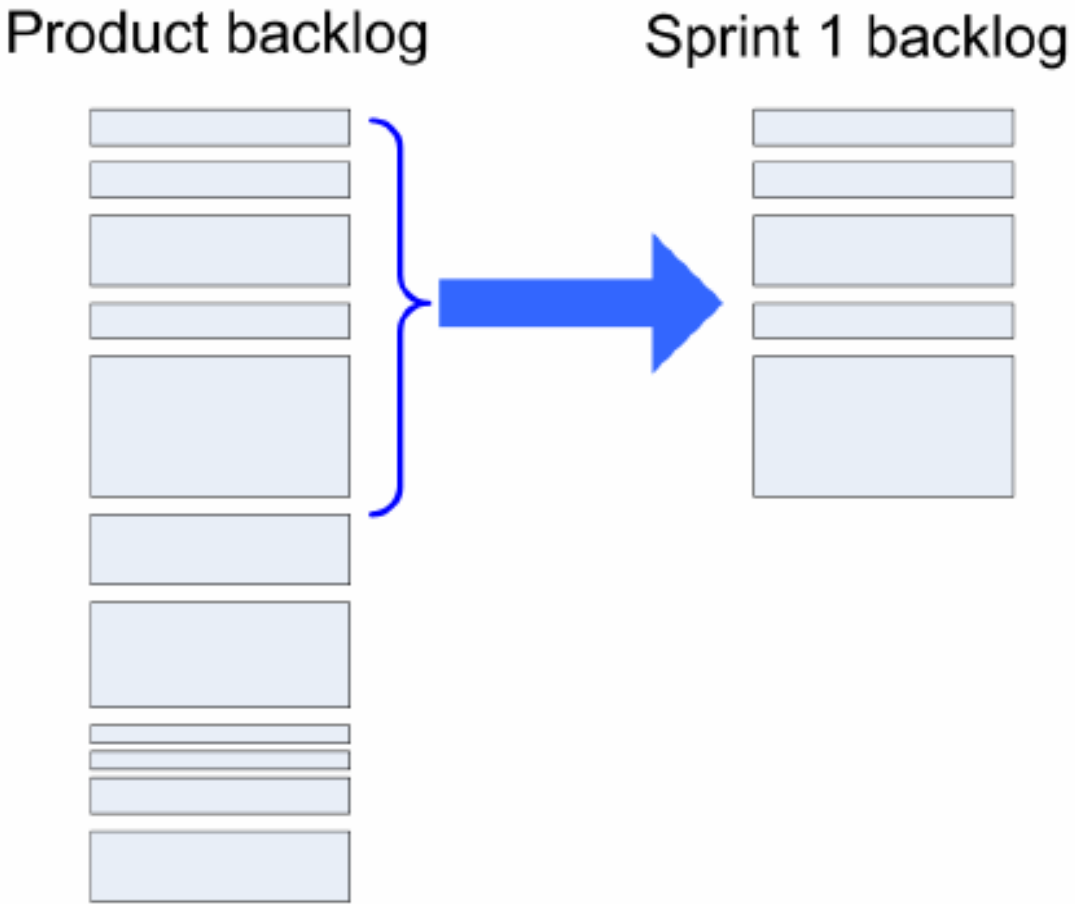
- Suunnitteluun osallistuu Product Owner ja kehittäjätiimi
- Lähtökohtana on sopivassa tilassa oleva eli DEEP Product backlog
  - Priorisoitu ja estimoitu
  - Korkeimman prioriteetin omaavat User storyt tarpeeksi pieniä ja Product Ownerin hyvin ymmärtämiä
- Suunnittelun yhteydessä määritellään **sprintin tavoite** (sprint goal)
  - Tavoite on jotain geneerisempää kuin yksittäisten backlogissa olevien User storyjen toteuttaminen
- Scrumin kehittäjä Ken Schwaber mainitsee 2002 kirjoitetussa kirjassaan asettavansa usein ensimmäisen sprintin tavoitteeksi: *”demonstrate a key piece of user functionality on the selected technology”*
- Seuraavalla sivulla Mike Cohnin määritelmä sprintin tavoitteesta

Sprintin tavoite [Mike Cohn]

- A sprint goal is a short, one- or two-sentence, description of what the team plans to achieve during the sprint
- It is written collaboratively by the team and the product owner
- The following are typical sprint goals on an eCommerce application:
  - Implement basic shopping cart functionality including add, remove, and update quantities
  - The checkout process—pay for an order, pick shipping, order gift wrapping, etc.
- The sprint goal can be used for quick reporting to those outside the sprint
  - There are always stakeholders who want to know what the team is working on, but who do not need to hear about each product backlog item (User story) in detail
- The success of the sprint will later be assessed during the Sprint Review Meeting against the sprint goal, rather than against each specific item selected from the product backlog
- http://www.mountaingoatsoftware.com/scrum/sprint-planning-meeting

Toteutettavien user storyjen valinta

- Sprintin tavoitteen asettamisen lisäksi tulee valita backlogista sprintin aikana toteutettavat User storyt
- Pääperiaate on valita ”sopiva määrä” backlogin korkeimmalle priorisoituja Storyjä



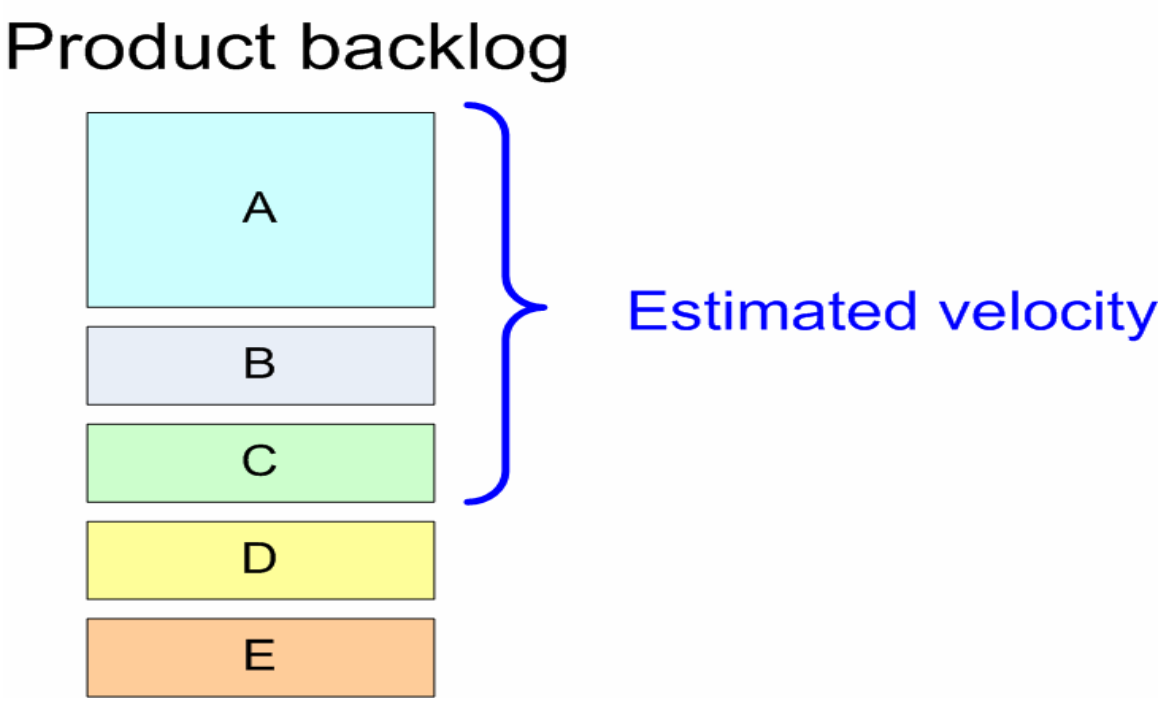
- Valituksi tulevat Storyt siirretään **sprintin backlogiin**

Sprinttiin otettavien User storyjen määrä

- Kehitystiimi siis päättää kuinka monta user storyä sprinttiin otetaan toteutettavaksi
- Tapoja päättää sprinttiin otettavien user storyjen määrä on muutamia:
  - ”perstuntuma”: otetaan niin monta korkeimman priotiteetin Storyä kuin mihin kaikki tiimiläiset tuntevat voivansa sitoutua
  - Jos storyt on estimoitu ja tiimin velositeetti tunnetaan, otetaan sprinttiin velositeetin verran storyjä
  - Edellisten yhdistelmä
- Jos user storyjä ei ole estimoitu tai velositeetti ei ole tiedossa, ”perstuntumamenetelmä” lienee ainoa jota voidaan käyttää
  - Tässäkin menetelmässä tiimi saa valita vain sellaiseen määrän storyjä, jotka se kokee voivansa toteuttaa kunnolla eli ”definition of donen” määrittelemän (eli suunnittelu, toteutus, automaattiset testit, testaus, integrointi, dokumentointi) mukaan valmiiksi
  - Velositeetin käsite ja estimaatithan huomioivat ”definition of donen”

Toteutettavien user stroyjen valinta

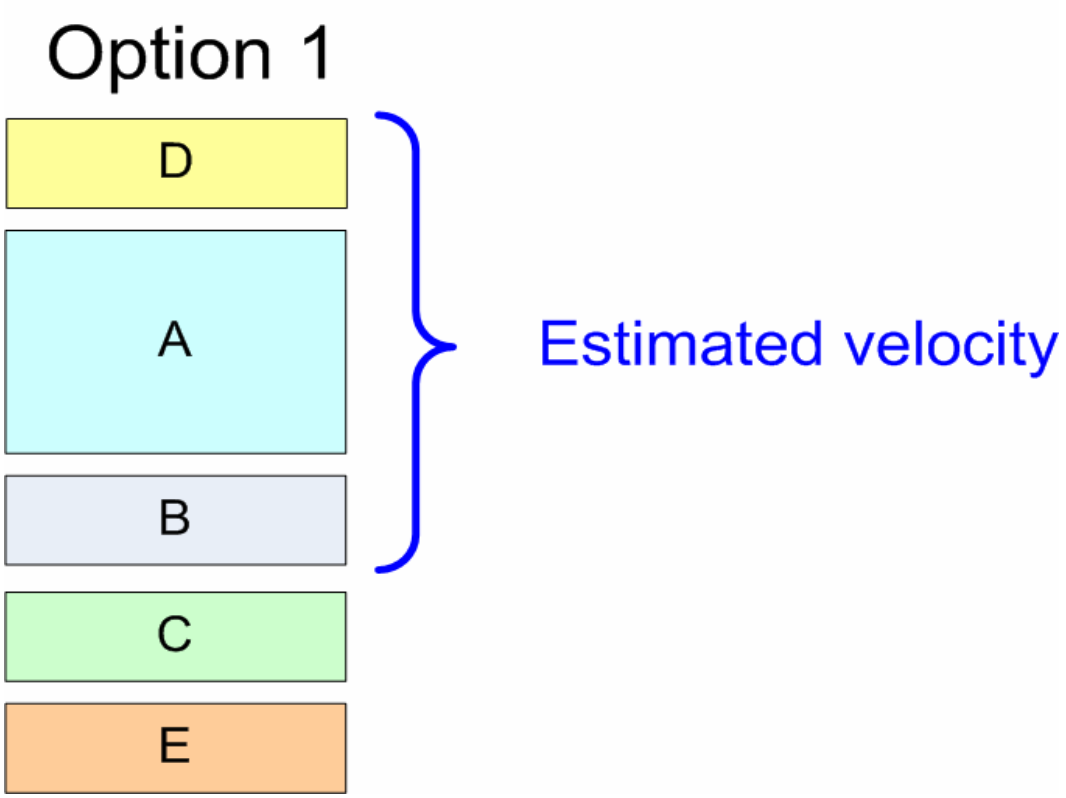
- Jos tiimin velositeetti on tiedossa ja user storyt on estimoitu, otetaan Storyjä mukaan maksimissaan velositeetin verran
- Product ownerilla on mahdollisuuksia vaikuttaa sprinttiin mukaan otettaviin User storyihin tekemällä uudelleenpriorisointia



- Entä jos Product Owner haluaa storyn D mukaan sprinttiin?

Uudelleenpriorisointi

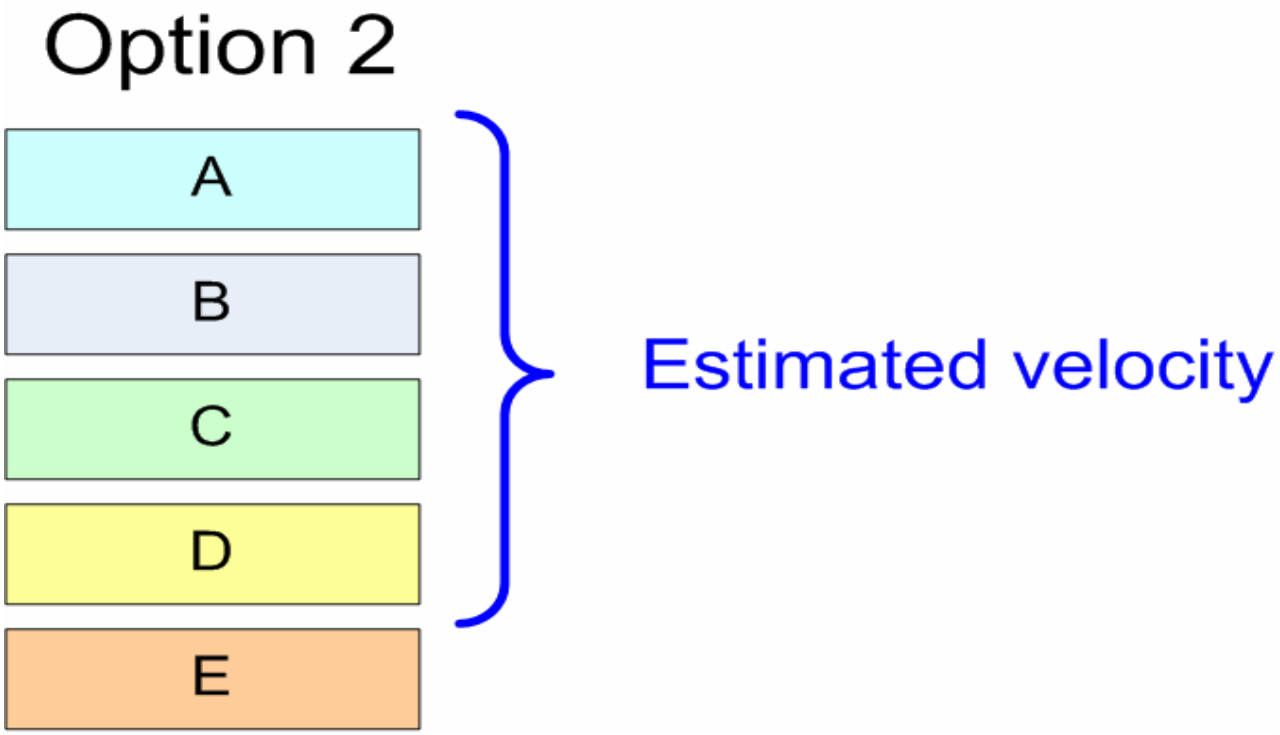
- Product Owner nostaa D:n prioriteettia, C tippuu pois sprinttiin valittavien User Storyjen joukosta



- Entä jos Product Owner haluaa Sprintiin mukaan kaikki user storyt A-D?

User Storyn scopen pienentäminen

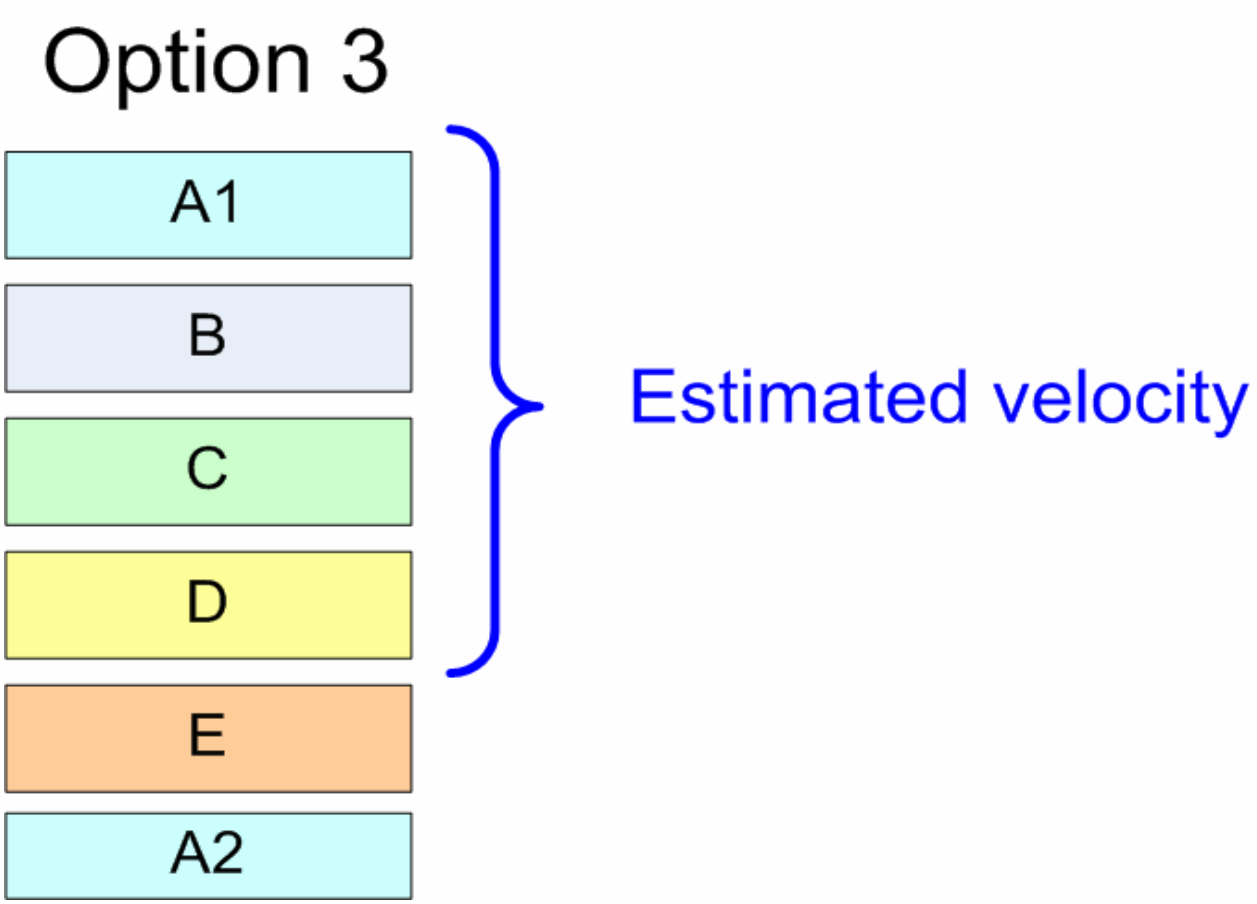
- *Jostain on luovuttava:* Product Owner pienentää user storyn A määrittelemää toiminnallisuutta, kehitystiimi estimoi pienennetyn A:n ja nyt A-D mahtuvat sprinttiin:



- Entä jos A:n toiminnallisuutta ei saa karsia ja silti Product Owner haluaa A-D:n mukaan sprinttiin?

User Storyn jakaminen

- Ratkaisu on jakaa User story A kahteen pienempään osaan A1:n ja A2:n
  - A1 sisältää A:n tärkeimmät piirteet ja otetaan mukaan sprinttiin
  - A2 saa alemman prioriteetin ja jää sprintin ulkopuolelle



User storyjen jakaminen

- Storyjen jakaminen pienemmiksi ei ole aloittelijalle, eikä aina ammattilaisellekaan helppoa
- Seuraavassa Richard Lawrencen ohjeita
  - http://www.richardlawrence.info/2009/10/28/patterns-for-splitting-user-stories/
- Good user stories follow Bill Wake's INVEST model. They're Independent, Negotiable, Valuable, Estimable, Small, and Testable
- Many new agile teams attempt to split stories by architectural layer: one story for the UI, another for the database, etc.
  - This may satisfy small, but it fails at independent and valuable.
- How small should stories be?
  - I recommend 6-10 stories per iteration, so how small is small enough depends on your team's velocity.
- Over my years with agile, I've discovered nine patterns for splitting user stories into good, smaller stories.

User storyjen jakaminen

- Pattern #1: **Workflow Steps**
  - As a content manager, I can publish a news story to the corporate website.

    ==>
  - … I can write and save a news story.
  - … I can edit a saved news story.
  - … I can publish a news story directly to the corporate website.
  - … I can publish a news story with editor review.
  - … I can view a news story on a editor review site.
  - … I can publish a news story from the editor review site to production
- Pattern #2**: Business Rule Variations**
  - As a user, I can search for flights with flexible dates.

    ==>
  - … as “between dates x and y.”
  - … as “a weekend in December.”
  - … as “± n days of dates x and y.”

User storyjen jakaminen

- Pattern #3: **Major Effort**
  - As a user, I can pay for my flight with VISA, MasterCard, Diners Club, or American Express.

    ==>
  - … I can pay with VISA
  - … I can pay with all four credit card types (VISA, MC, DC, AMEX) (given one card type already implemented).

- Pattern #4: **Simple/Complex**
  - As a user, I can search for flights between two destinations.

    ==>
  - … when only direct flights used.
  - … specifying a max number of stops.
  - … including nearby airports.
  - … using flexible dates.

User storyjen jakaminen

- Pattern #6: **Data Entry Methods**
  - As a user, I can search for flights between two destinations.

    ==>
  - … using simple date input.
  - … with a fancy calendar UI.

- Pattern #7: **Operations**
  - As a user, I can manage my account.

    ==>
  - … I can sign up for an account.
  - … I can edit my account settings.
  - … I can cancel my account

User storyjen jakaminen

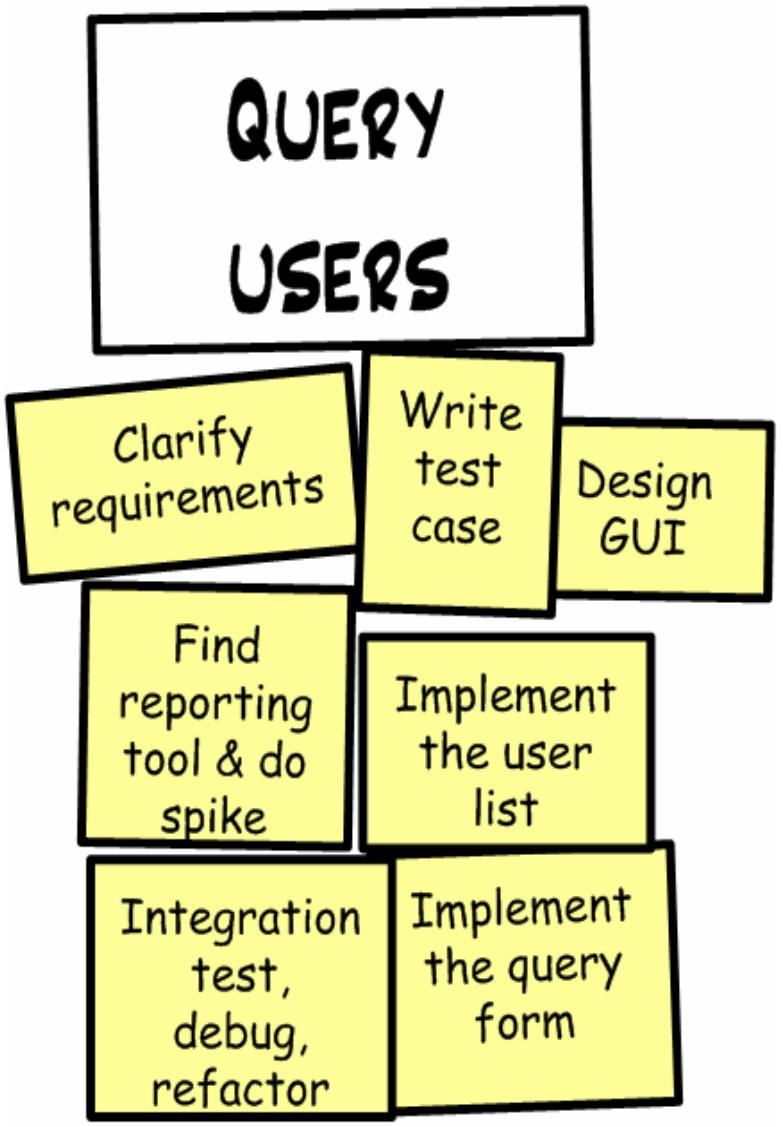
- Pattern #8: **Defer Performance**
  - As a user, I can search for flights between two destinations.

    ==>
  - … (slow—just get it done, show a “searching” animation).
  - … (in under 5 seconds).
- Pattern #9: **Break Out a Spike**
  - A story may be large not because it’s necessarily complex, but because the implementation is poorly understood. In this case, no amount of talking about the business part of the story will allow you to break it up. **Do a time-boxed spike first to resolve uncertainty around the implementation**. Then, you can do the implementation or have a better idea of how to break it up.
  - As a user, I can pay by credit card.

    ==>
  - Investigate credit card processing.
  - Implement credit card processing.

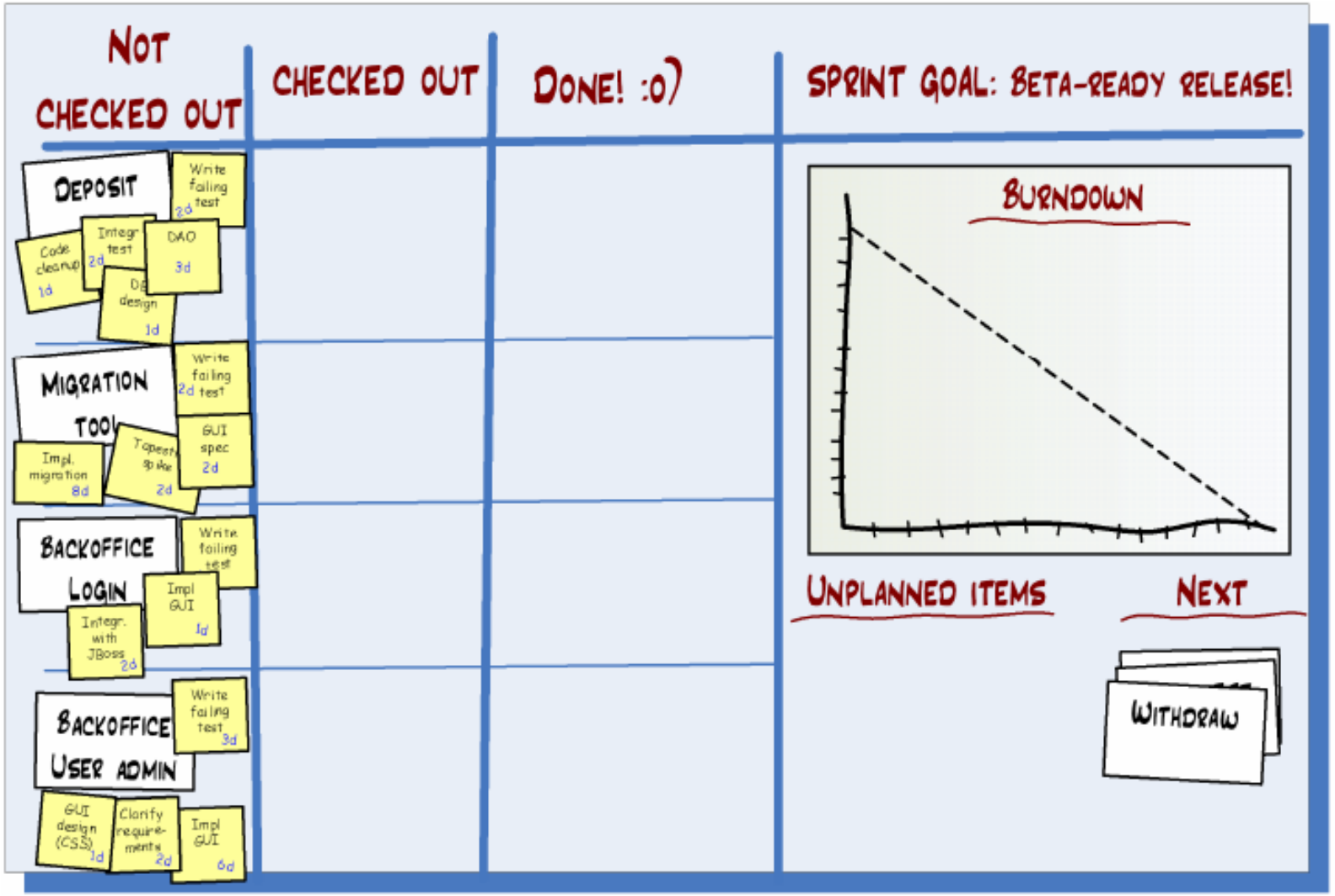
Sprintin suunnittelun toinen vaihe

- Sprintin suunnittelun yhteydessä sprinttiin valituille User storyille tehdään karkean tason suunnittelu
  - Mietitään mitä teknisen **tason tehtäviä** (**task**) on toteutettava, jotta user story saadaan valmiiksi
  - Suunnitellaan komponentteja ja rajapintoja karkealla tasolla
  - Huomioidaan user storyn aiheuttamat muutokset olemassa olevaan osaan sovelluksesta
  - Kaikkia storyyn liittyviä taskeja ei sprintin suunnittelun aikana välttämättä löydetä
  - Uusia taskeja generoidaan tarvittaessa sprintin edetessä



Sprint backlog

- Sprintin tehtävälista eli Sprint backlog koostuu sprintiin valituista user storeista ja niihin liittyvistä tehtävistä eli taskeista
- Backlog voi olla organisoitu ”taskboardiksi”:

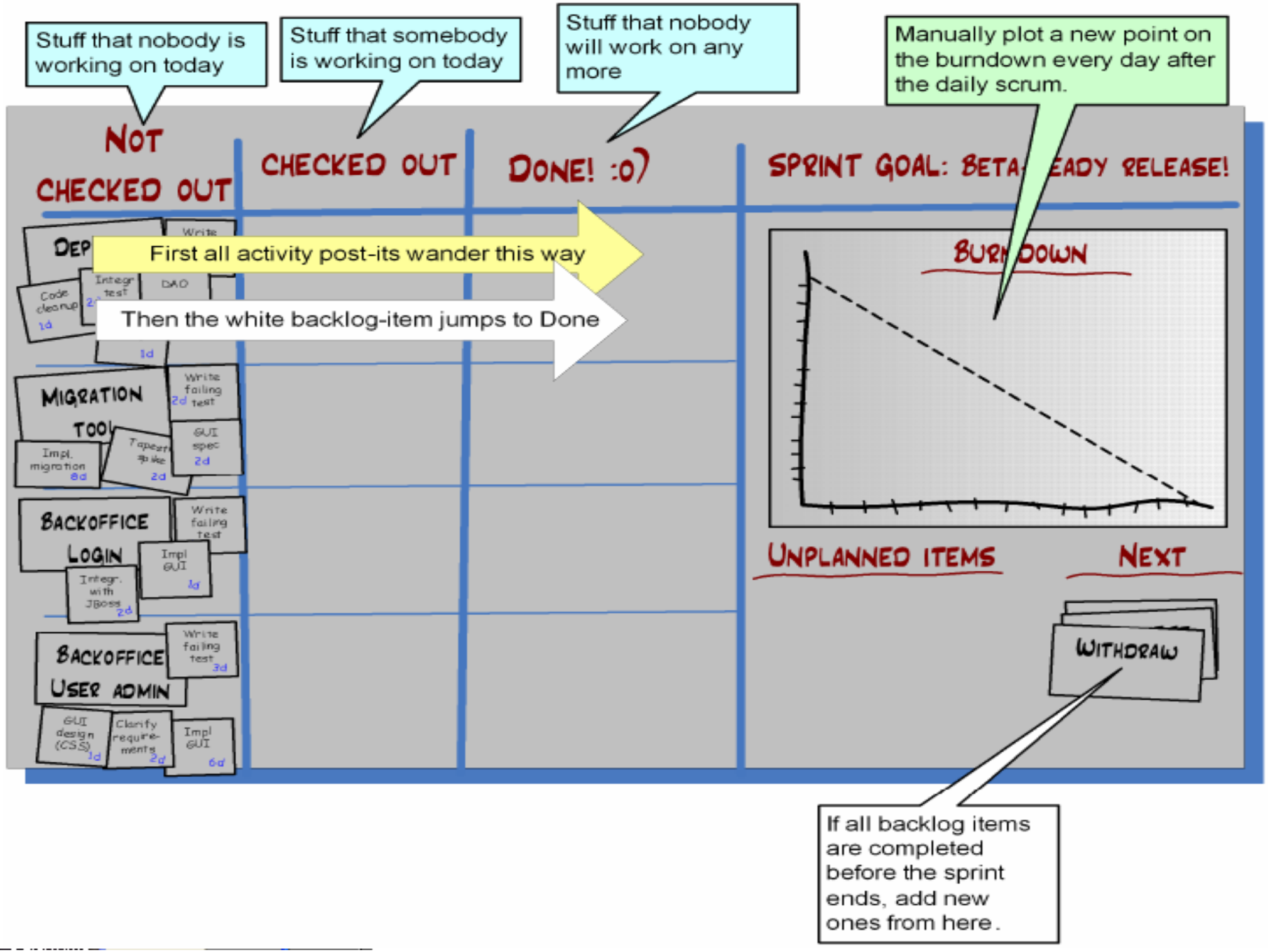


Taskboard

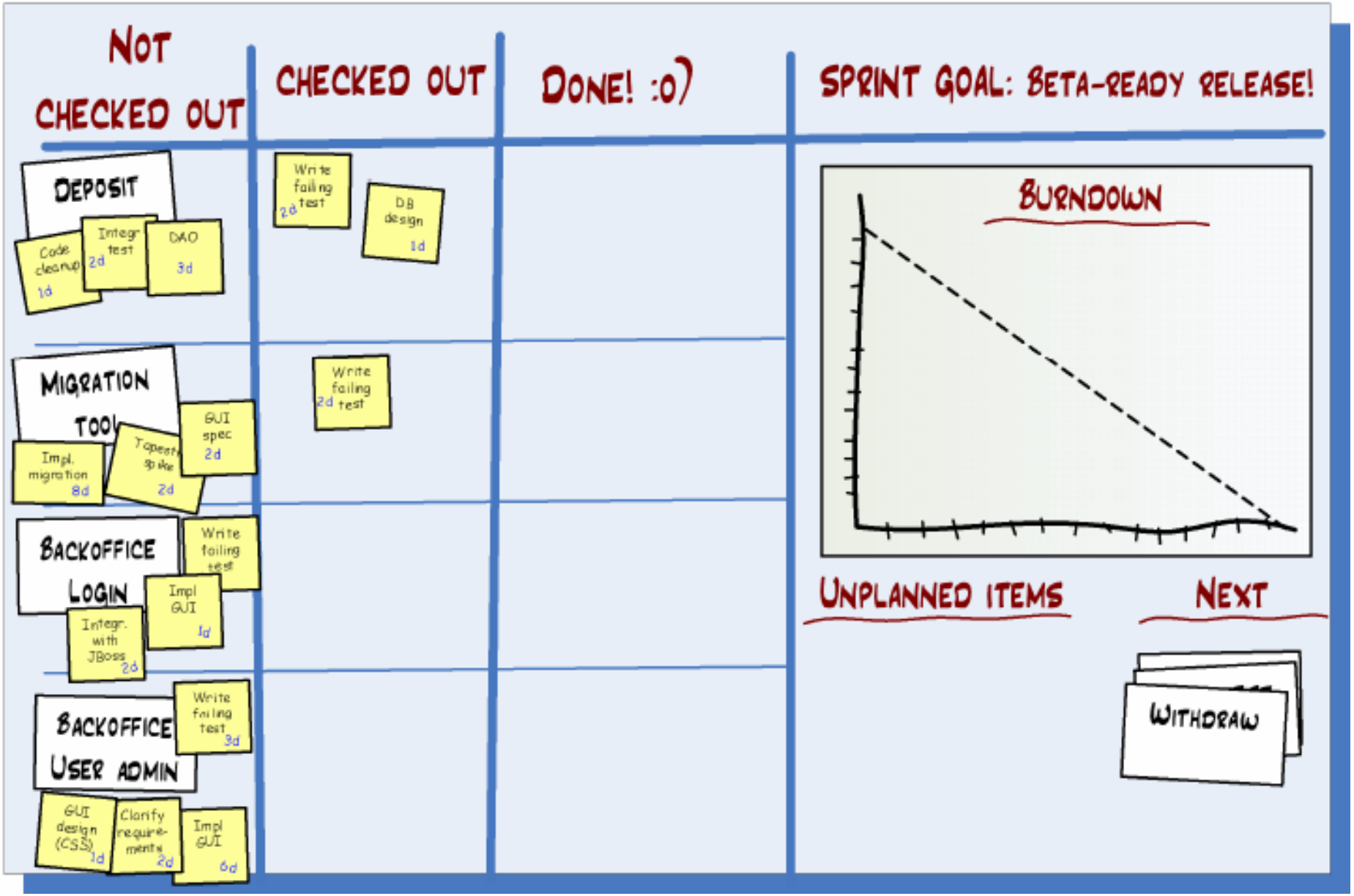
- Kuvassa sprinttiin on valittu 4 user storyä ja taskboard on jaettu neljään kaistaan (swimlane), jokaiselle Storylle oma kaista
- Kuten arvata saattaa, jokaisen taskin on tarkoitus siirtyä sarakkeesta ”not checked out” sarakkeeseen ”done”
- **Sprintissä arvioidaan päivittäin jäljellä olevaksi arvioitua työmäärää**
- Taskien **jäljellä oleva työmäärä** arvioidaan yleensä **tunteina**
- Jokaiseen taskiin kirjataan sen arvioitu vielä jäljellä oleva työmäärä
  - Jos käytössä on ”manuaalinen” taskboard, kirjoitetaan arvio suoraan taskia edustavaan postit-lappuun
  - Arviota päivitetään joka päivä
  - Arvio voi nousta jos taski huomataankin työläämmäksi mitä alun perin ajateltiin
- On varsin tavallista, että uusia taskeja keksitään kesken sprintin
  - Uudet taskit saavat olla ainoastaan kehittäjätiimin itse identifioimia menossa olevaan sprinttiin liittyviä töitä
- Eli sprintissä jäljellä oleva työaika-arvio voi kasvaa kesken sprintin!

Sprintissä jäljellä olevan työmäärän arviointi

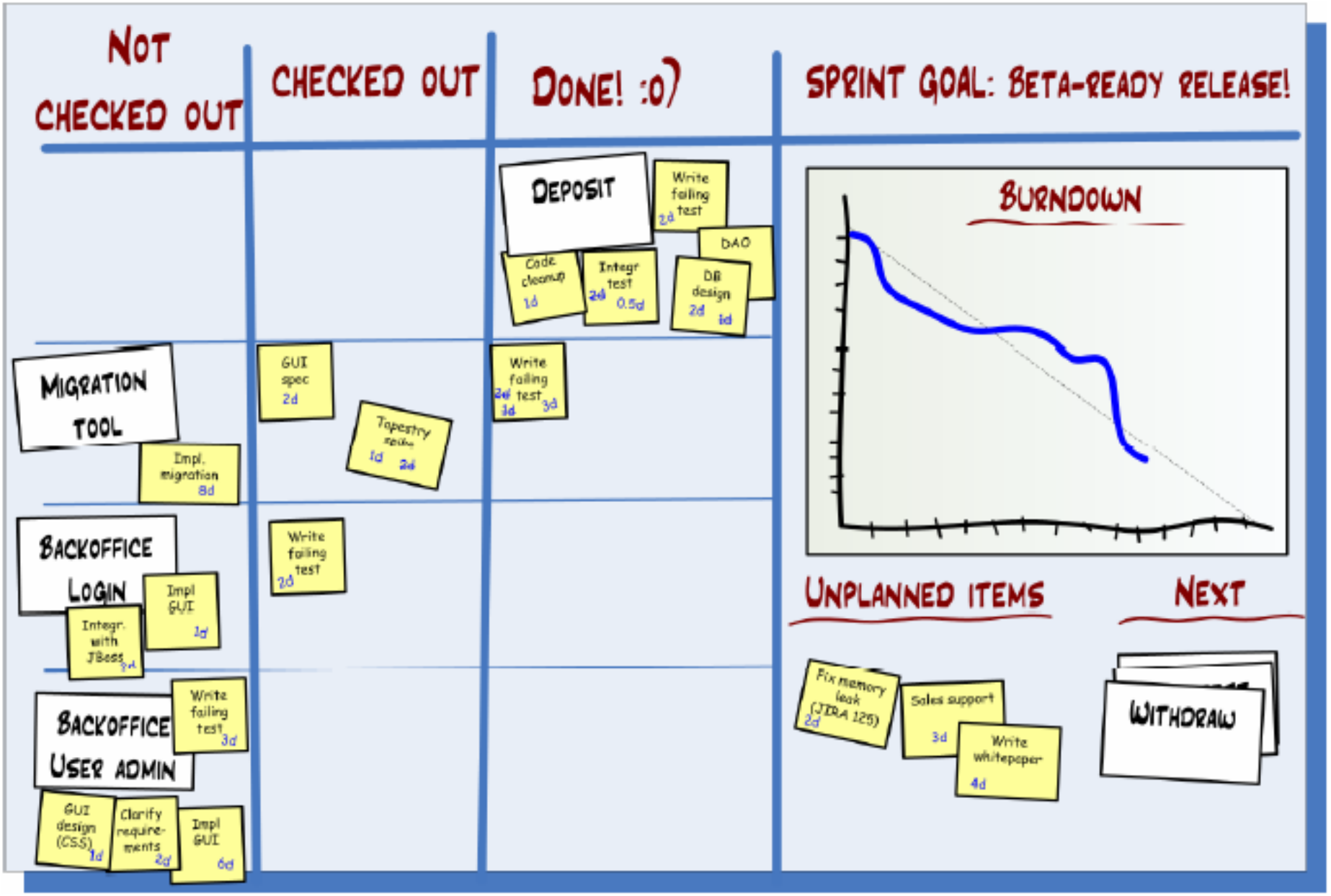
- Jokaisen taskin jäljellä olevan työn määrä arvioidaan esim. päivittäisessä scrum-palaverissa eli daily scrumeissa
- Jäljellä olevaa työmäärää (tunteina mitattuna) visualisoidaan sprintin etenemistä kuvaavalla **burndown-käyrällä**
  - Tätä sprintin burndown:ia ei pidä sekottaa projektin burndown-käyrään!


Stuff that nobody is working on today
Stuff that somebody is working on today
Stuff that nobody will work on any more
Manually plot a new point on the burndown every day after the daily scrum.
NOT CHECKED OUT
CHECKED OUT
DONE! :o)
SPRINT GOAL: BETA-READY RELEASE!
First all activity post-its wander this way
Then the white backlog-item jumps to Done
DEP
Write
Integr test
DAO
Code cleanup
1d
1d
MIGRATION TOOL
Write failing test
2d
GUI spec
2d
Impl. migration
8d
Tapestry spike
2d
BACKOFFICE LOGIN
Write failing test
Impl GUI
1d
Integr. with JBoss
2d
BACKOFFICE USER ADMIN
Write failing test
3d
GUI design (CSS)
1d
Clarify require-ments
2d
Impl GUI
6d
BURNDOWN
UNPLANNED ITEMS
NEXT
WITHDRAW
If all backlog items are completed before the sprint ends, add new ones from here.

Tilanne sprintin alussa



Ja puolen välin jälkeen



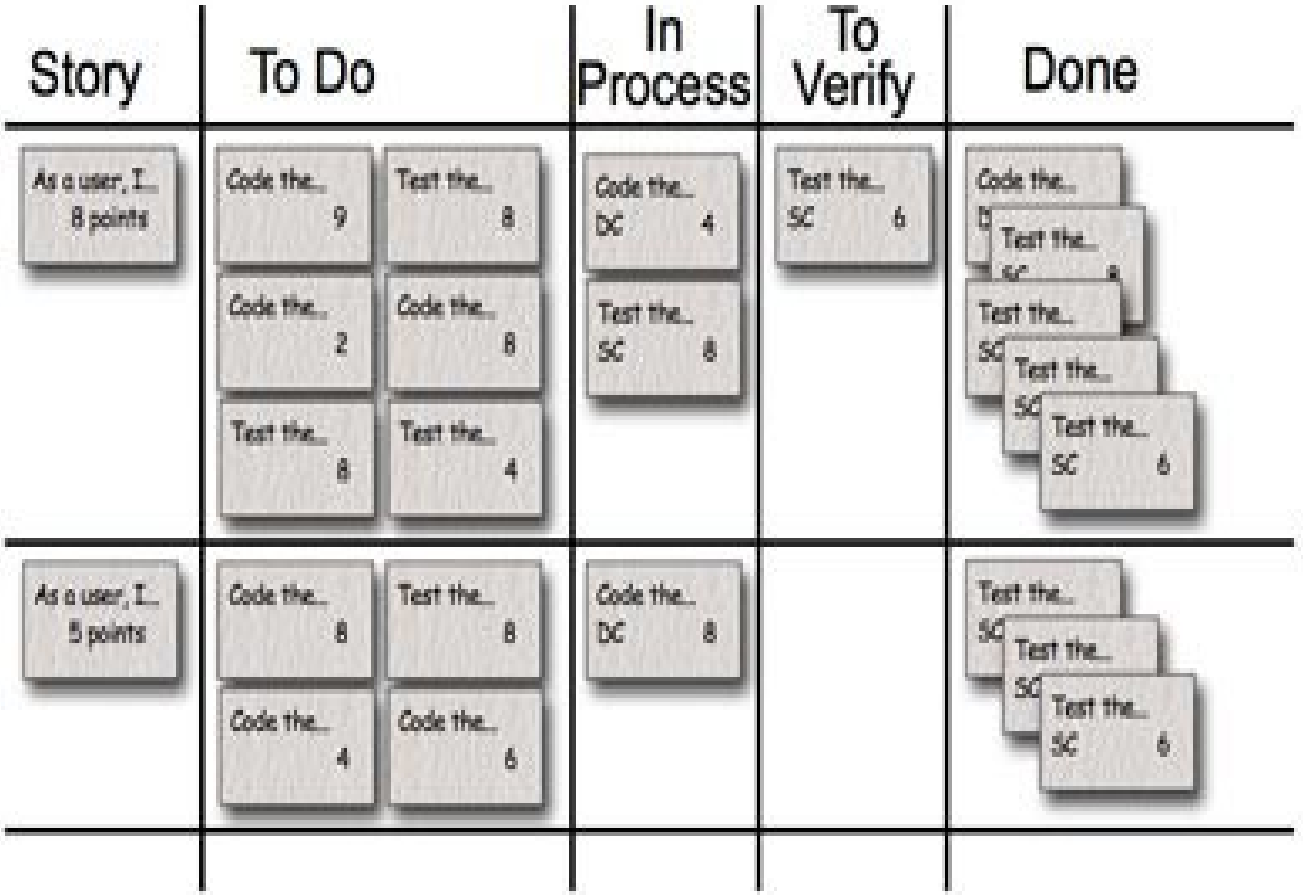
Sprintin seuranta taulukkomuodossa

- Taskboardin sijaan sprintin seuranta hoidetaan usein taulukkolaskennan avulla, erityisesti jos tiimillä ei ole käytössä omaa ”seinää”
- Tällöin sprintin jokaiselle päivälle on oma sarake, johon merkitään kunkin päivän alussa estimaatti taskien jäljellä olevasta työmäärästä (tunteina)

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

- Erään ohtuprojektin product- ja sprintbacklogit:
  - https://docs.google.com/spreadsheets/d/13RzIZI2NFFuV0zdRjrrfoC-CrootK8AZNuHS571WIxo/edit?usp=sharing

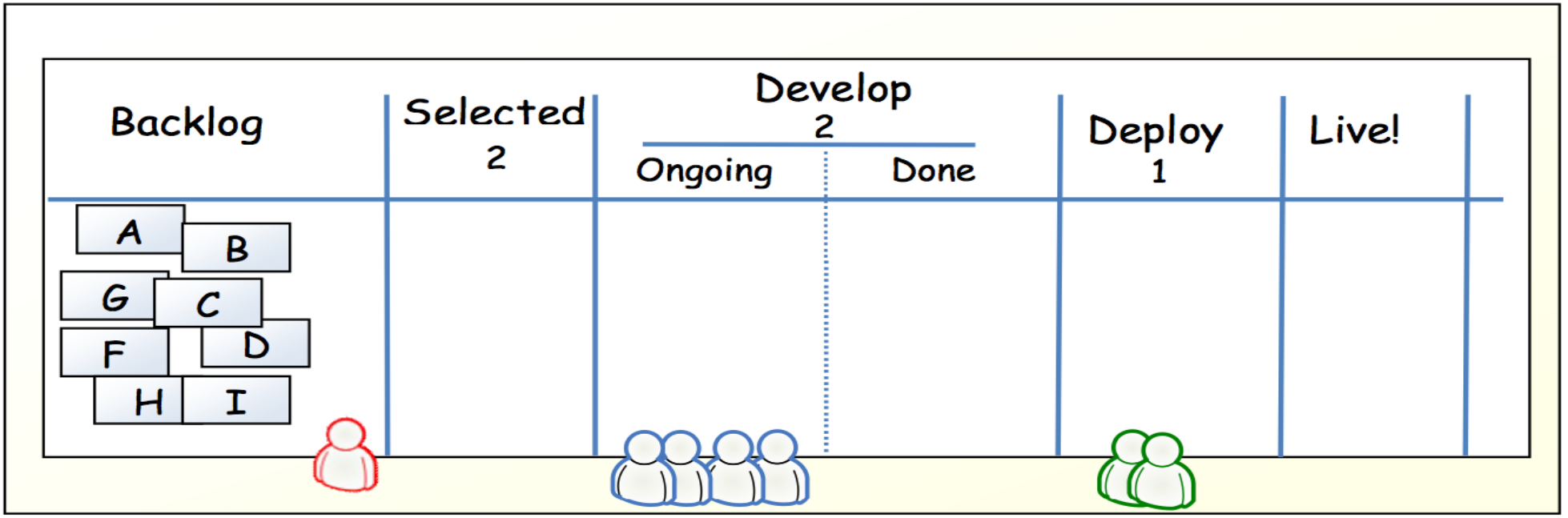
Taskboardissa voi olla merkattu useampiakin työvaiheita

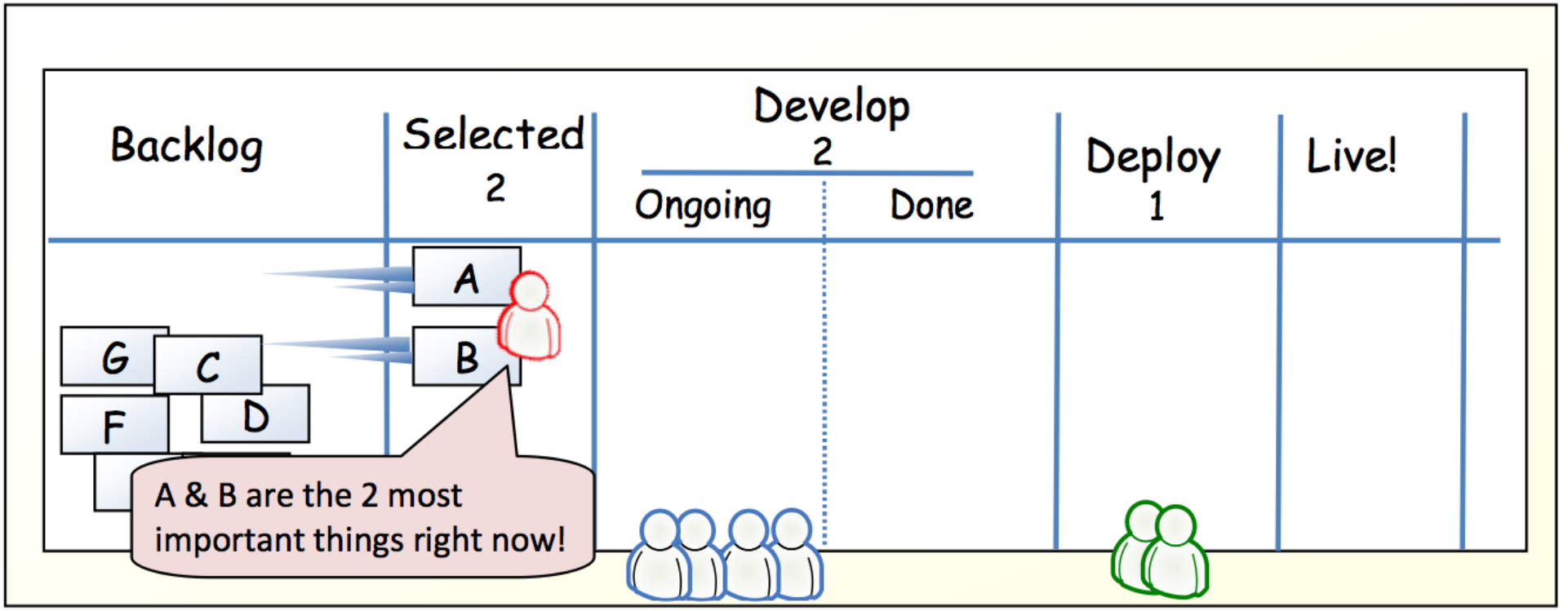


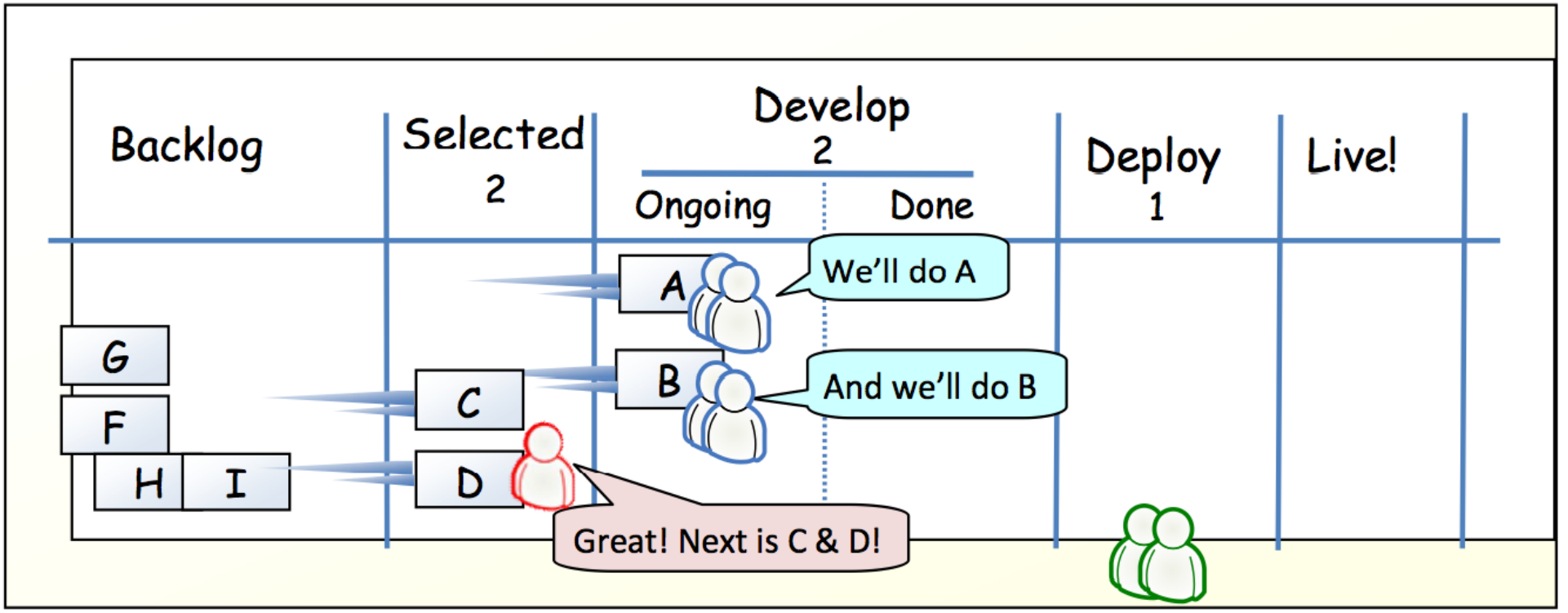
- Yhtä aikaa työn alla olevien taskien suuri määrä voi koitua Scrumissa ongelmaksi, sillä riski sille, että sprintin päätyttyä on paljon osittain valmiita user storyja kasvaa

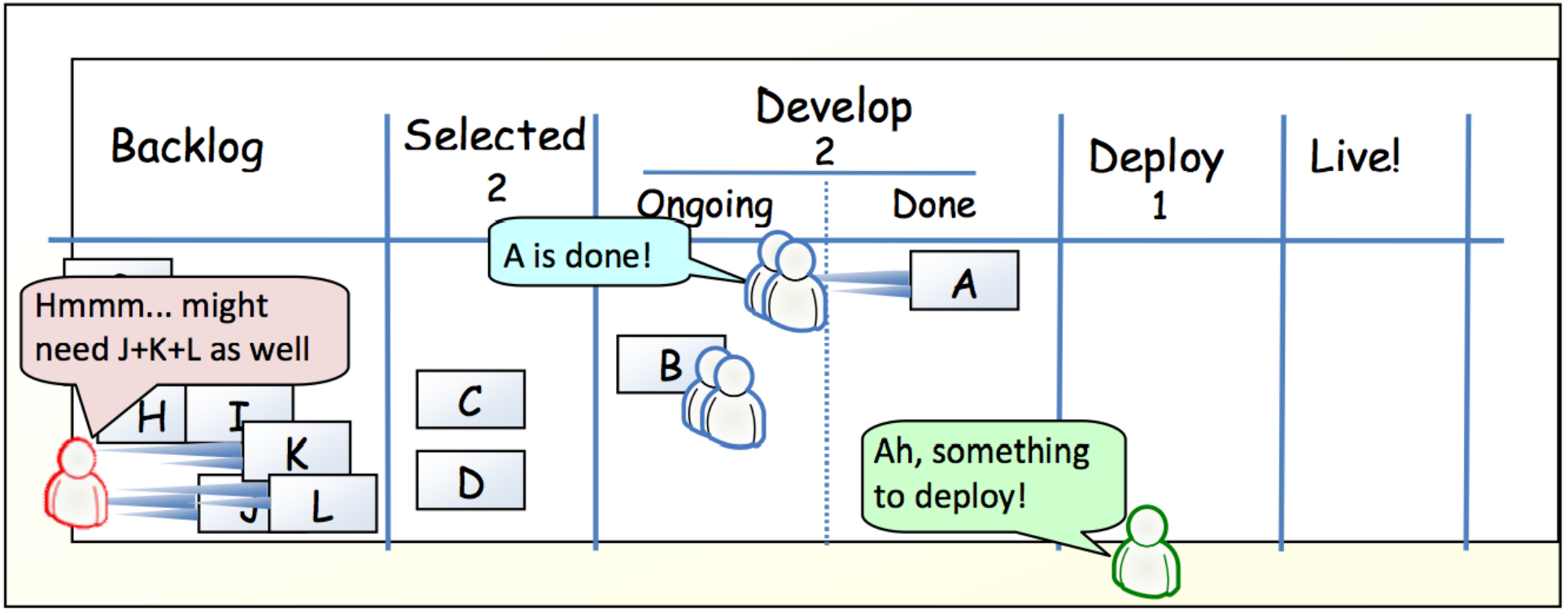
Yhtäaikaa tehtävän työn rajoittaminen

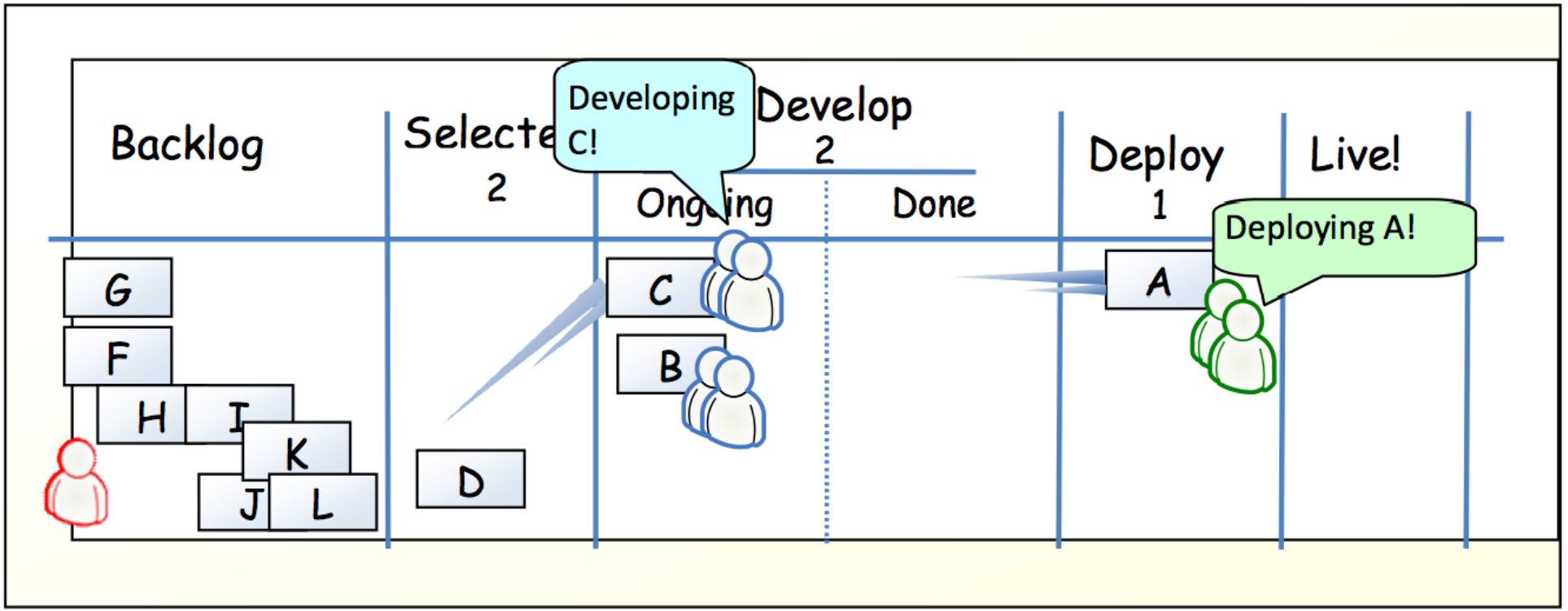
- Voikin olla mielekästä rajoittaa yhtä aikaa tekemisen alla olevien töiden määrää asettamalla **work in progress (eli WIP) -rajoituksia**
  - WIP-rajoitukset on lainattu **Kanban**-menetelmästä
  - Scrumin ja Kanbanin yhdistelmää kutsutaan usein nimellä *Scrumban*
  - Scrumbanissa on tosin muitakin Kanbanista lainattuja elementtejä kuin WIP-rajoitukset
- Esimerkki kirjasta http://www.infoq.com/minibooks/kanban-scrum-minibook
- Kunkin vaiheen WIP-rajoitus on merkitty numerona, eli vaiheessa saa olla yhtä aikaa vain rajoituksen verran taskeja

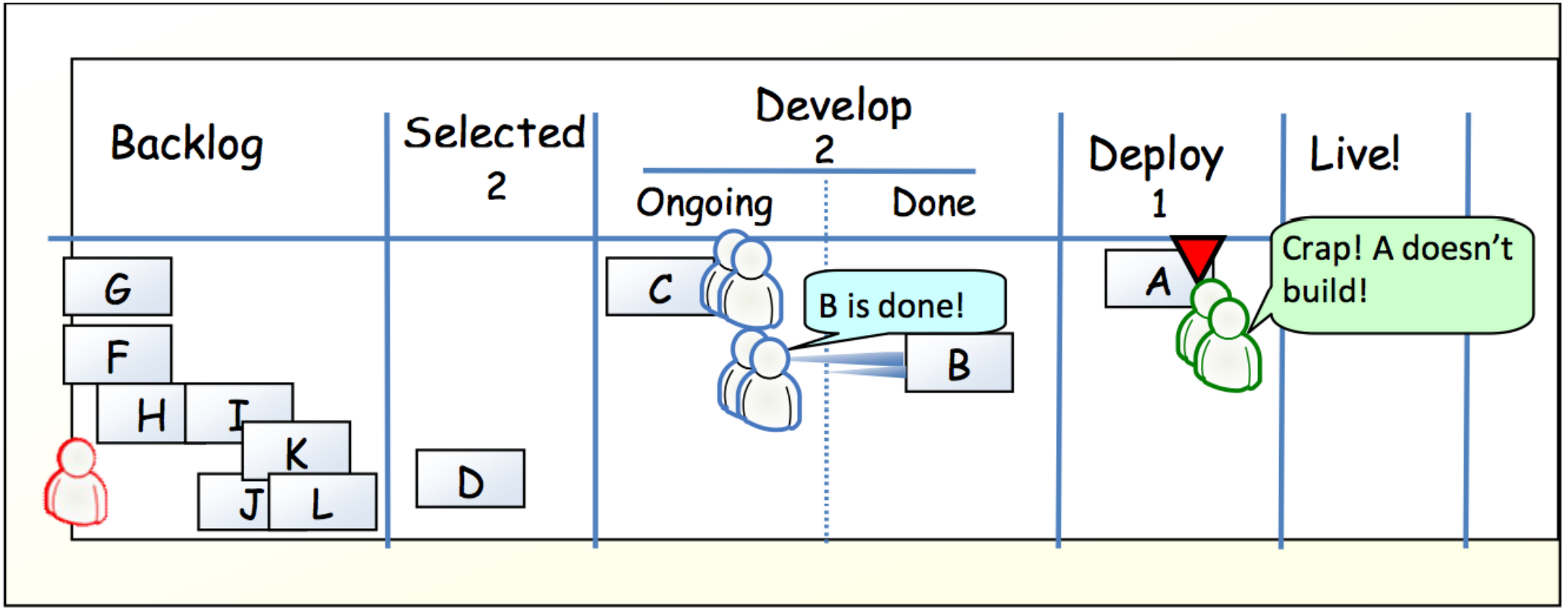


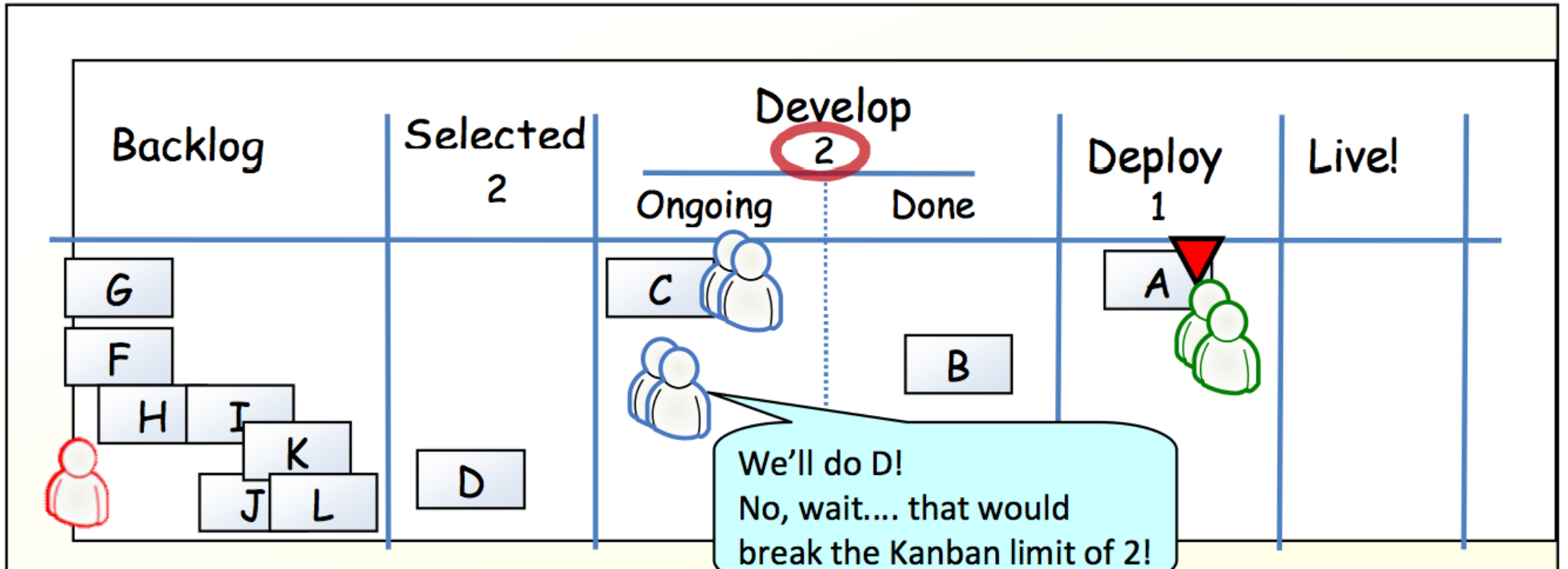

Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
A
B
G
C
F
D
A & B are the 2 most important things right now!

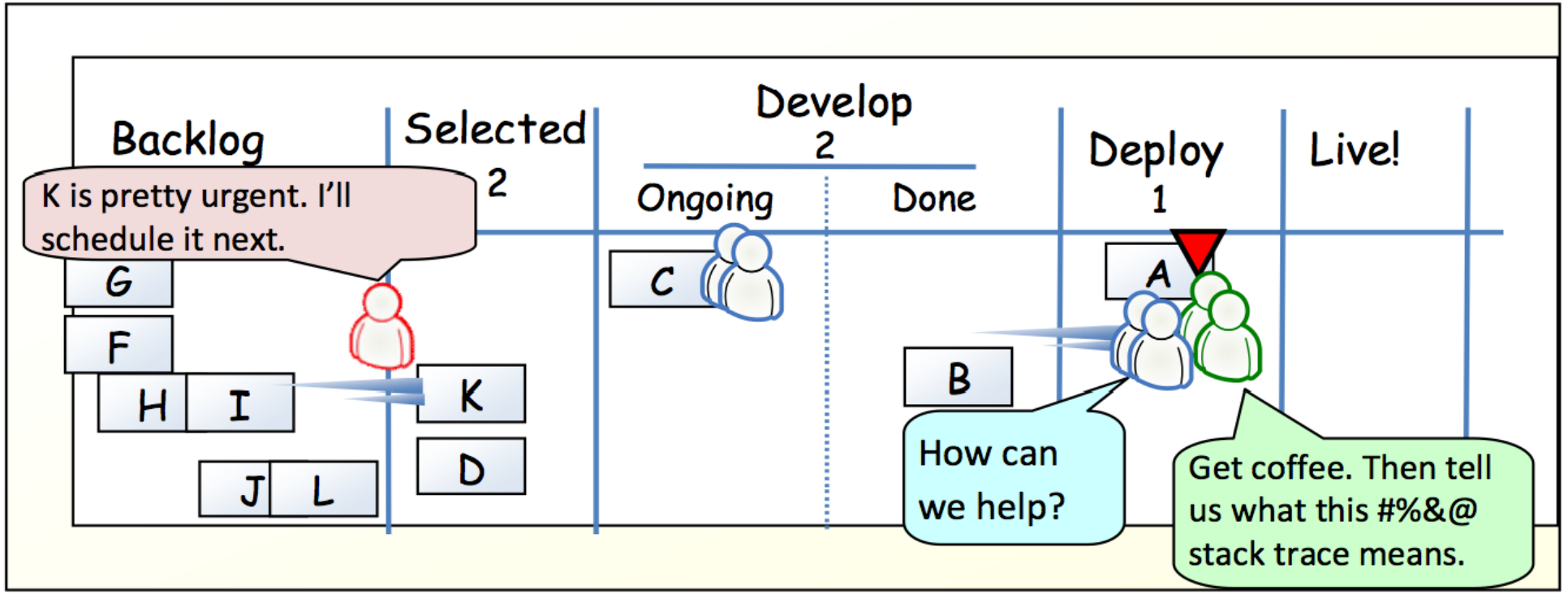

Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
A
We'll do A
B
And we'll do B
G
C
F
H
I
D
Great! Next is C & D!

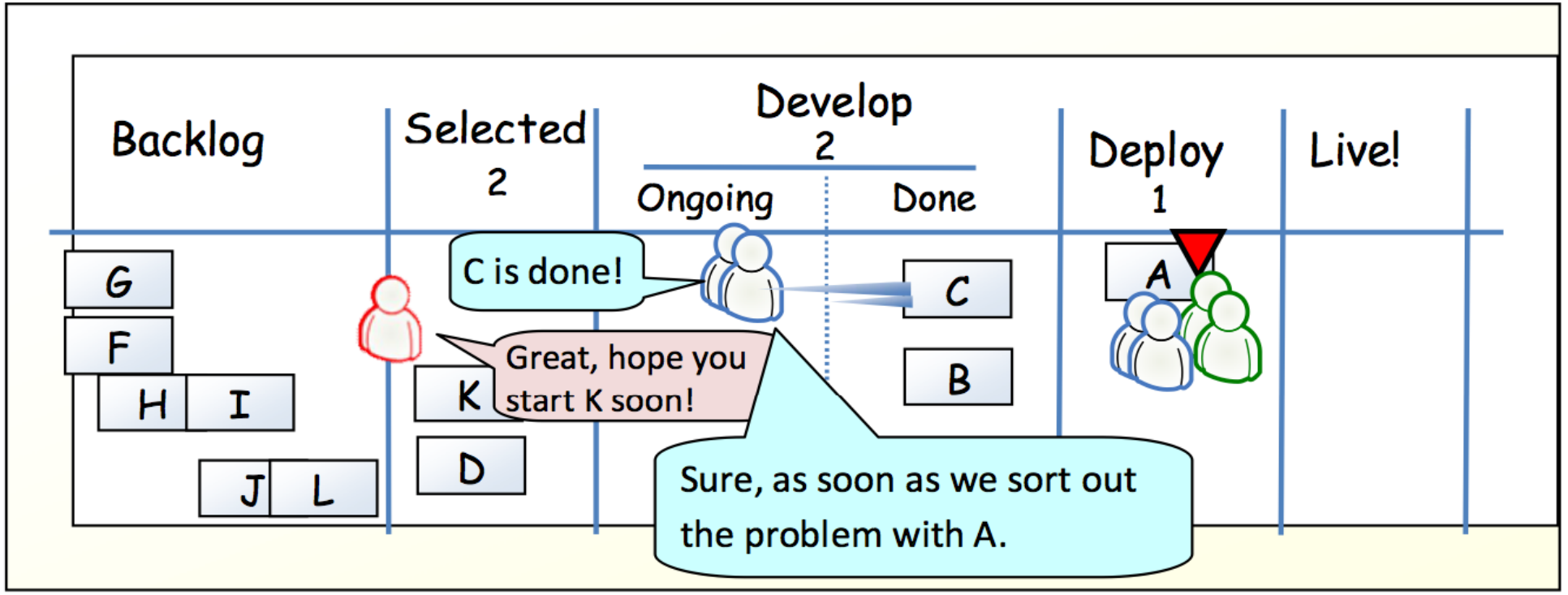

Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
A is done!
A
Hmmm... might
need J+K+L as well
H
I
K
J
L
C
D
B
Ah, something
to deploy!

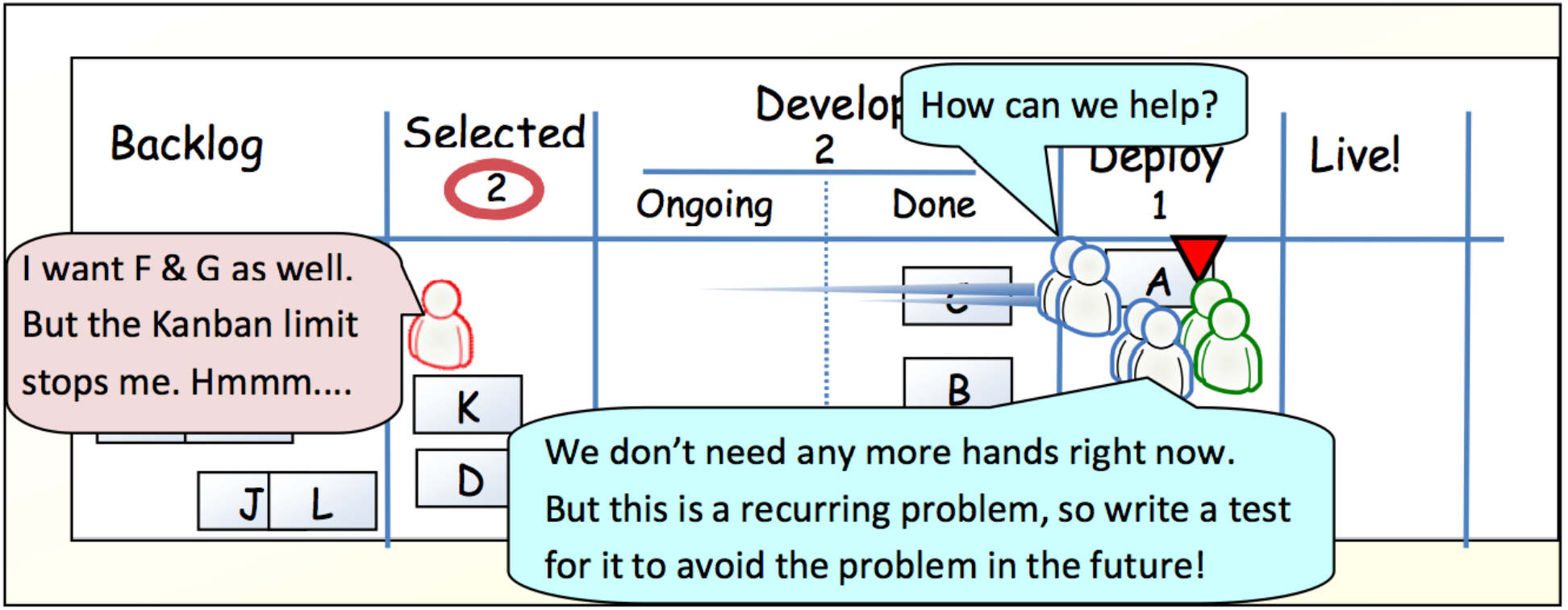

Backlog
Selecte
2
Developing
C!
Develop
2
Ongoing
Done
Deploy
1
Live!
Deploying A!
A
G
F
H
I
K
J
L
D
C
B

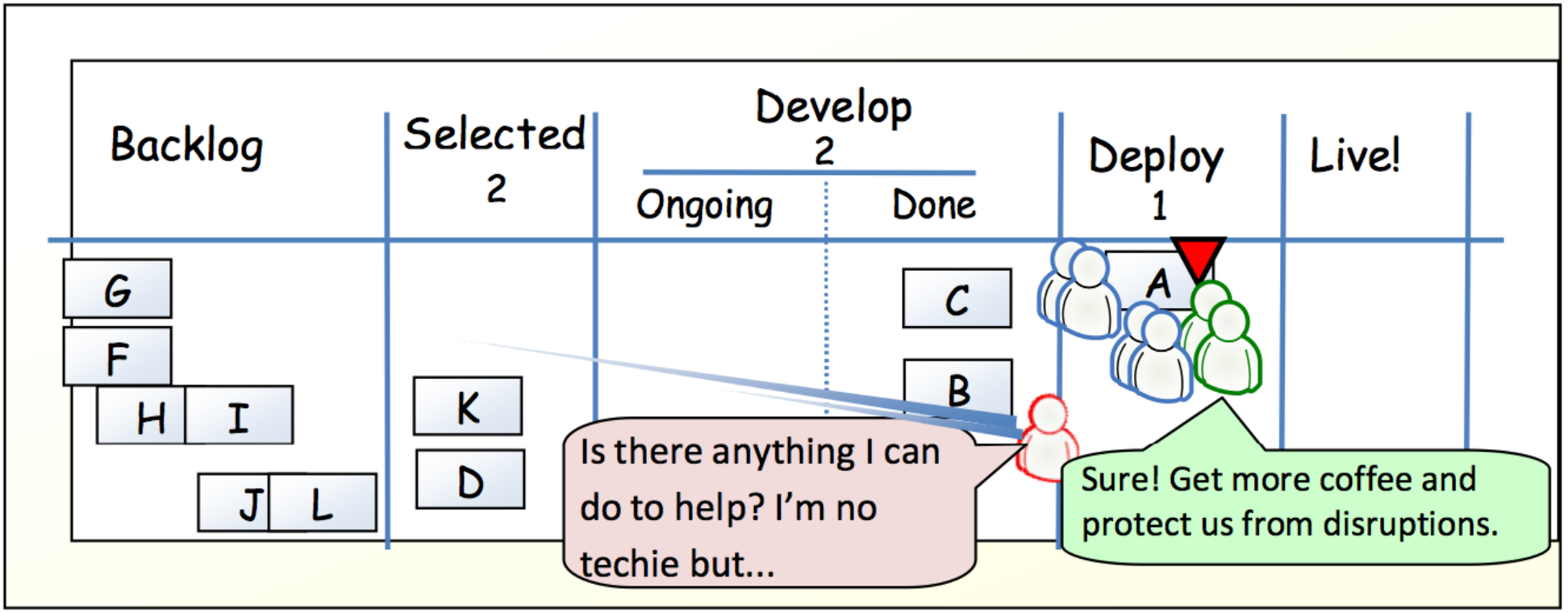

Backlog
Selected 2
Develop 2
Ongoing
Done
Deploy 1
Live!
G
F
H
I
K
J
L
D
C
B is done!
B
A
Crap! A doesn't build!


Backlog
Selected 2
Develop 2
Ongoing
Done
Deploy 1
Live!
G
F
H
I
K
J
L
D
C
B
A
We'll do D!
No, wait.... that would break the Kanban limit of 2!


Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
K is pretty urgent. I'll schedule it next.
G
F
H
I
J
L
K
D
C
B
A
How can we help?
Get coffee. Then tell us what this #%&@ stack trace means.


Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
G
F
H
I
J
L
K
D
C is done!
Great, hope you start K soon!
C
B
A
Sure, as soon as we sort out the problem with A.


Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
How can we help?
I want F & G as well. But the Kanban limit stops me. Hmmm....
A
C
B
K
D
J
L
We don't need any more hands right now. But this is a recurring problem, so write a test for it to avoid the problem in the future!


Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
G
F
H
I
J
L
K
D
C
B
A
Is there anything I can do to help? I'm no techie but...
Sure! Get more coffee and protect us from disruptions.

Yhtäaikaa tehtävän työn rajoittamisien motivaatio

- Yhtäaikaa tehtävän työn määrää kontrolloivien WIP- eli Work in progress -rajoitusten idea on siis peräisin Kanban-menetelmästä
- Kanban-menetelmä taas on eräs keskeisimmistä Lean-ajattelun työkaluista
- Lean-ajattelun taustalla on idea *turhuuden* (engl. *waste*, jap. *muda*) eli arvoa tuottamattomien asioiden eliminoimisessa organisaatioiden toiminnassa
  - Lean-ajattelu on peräisin jo kymmeniä vuosia vanhasta Toyota Production Systemistä
- Lean tunnistaa useita erilaisia turhuuksia (*lean waste*), näiden joukossa ovat esim. *osittain tehty työ* (partially done work), välivarastointi ja turha odottaminen
- Ohjelmistotuotannon kontekstiin sovellettuna työvaiheet, jotka eivät ole vielä definition of donen mielessä valmiina edustavat leanin mukaista turhuutta
  - Esim. testaamista odottavien toiminnallisuuksien (user storyjen) katsotaan olevan ”välivarastoituna”, samoin jo testatut mutta tuotantoon viemistä vielä odottavat toiminnallisuudet ovat ”välivastastossa”
  - Asiakkaalle toiminnallisuudet alkavat tuottaa arvoa vasta kun ne saadaan käyttöön, siinä vaiheessa kun toiminnallisuudet ovat työn alla, ne aiheuttavat ainoastaan kustannuksia ja muodostavat riskin

Yhtäaikaa tehtävän työn rajoittaminen

- Kanban- ja Scrumban-menetelmissä WIP-rajoitteilla rajataan useimmiten yhtä aikaa työn alla olevien User storyjen määrää
  - Kanbanissa ja Scrumbanissa ei yleensä ole olemassa Scrumin sprintin kaltaista kehitystyötä rytmittävää käsitettä
  - Voidaan esim noudattaa periaatetta, missä tiimi tekee yhden user storyn kerrallaan valmiiksi, demoaa sen asiakkaalle ja valitsee product backlogista seuraavan storyn työn alle
  - Joissain tapauksissa asiakastapaamiset ja valmiiden storyjen esittely voi Kanbanissa ja Scrumbanissa tapahtua sovitun aikataulun, esim. 2 viikon välein vaikka itse kehitystyö ei noudattaisi sprinttejä vaan etenisi story kerrallaan
- WIP-rajoitteita voidaan soveltaa Scrumin yhteydessä monella tapaa
  - Järkevintä on rajoittaa sprintin aikana yhtäaikaa työn alla olevien storyjen määrää mahdollisimman pieneksi
  - On myös tavallista rajoittaa eri työvaiheessa, esim. toteutuksen olevien taskien määrää
  - tai yksittäisellä sovelluskehittäjän kerrallaan työn alla olevien töiden määrää
- WIP-rajoitteita säädetään usein retrospektiivien yhteydessä jos kehitystyössä havaitaan ongelmia

Sprintin etenemisen seuranta

- Taskboard ja burndown-käyrä tuovat selkeästi esille sprintin etenemisen asteen ja onkin suositeltavaa, että ne ovat kaikkien tiimiläisten ja projektin sidosryhmäläisten nähtävillä
- Ketterät menetelmät korostavat läpinäkyvyyttä (transparency) ja tiedon maksimaalista kommunikoitumista, näin mahdolliset ongelmatkaan eivät tule yllätyksenä ja niihin on helpompi puuttua ajoissa
- Lisää aiheesta:
  - http://xprogramming.com/articles/bigvisiblecharts/
  - http://blog.mountaingoatsoftware.com/the-ideal-agile-workspace
- Usein toki käytetään myös elektronisia vastineita taskboardista, erityisesti jos kyseessä on hajautettu tiimi, esim:
  - Edellisen sivun tyyliin google docs tai excel
  - Asana, Trello, Github project, Pivotal Tracker, JIRA, trac, bugzilla, ...
- Yleinen konsensus on kuitenkin, että ainakin Sprintin hallintaan manuaalinen postit-lappuja hyödyntävä taskboard on käytettävyydeltään ylivoimainen

Sprint review ja retrospektiivi

- Kuten luennolla 2 mainittiin pidetään sprintin lopussa sprint review eli katselmointi ja sprintin retrospektiivi
- Katselmoinnissa arvioidaan kehitystiimin tekemää työtä
  - Kesken jääneet tai epäkelvosti toteutetut User storyt siirretään takaisin backlogiin
- Retrospektiivissä taas tiimi itse tarkastelee omaa toimintatapaansa ja identifioi mahdollisia kehityskohteita seuraavaan sprinttiin

- Sprintin aikana on product backlogiin tullut ehkä uusia User storyja tai jo olemassa olevia storyjä on muutettu ja uudelleenpriorisoitu
- On suositeltavaa että kehitystiimi käyttää pienen määrän aikaa sprintin aikana product backlogin vaatimiin toimiin eli (backlog groomingiin), esim. uusien User storyjen estimointiin
- Jos product backlog on hyvässä kunnossa (DEEP) sprintin loppuessa, on jälleen helppo lähteä sprintin suunnitteluun ja uuteen sprinttiin