

Ohjelmistotuotanto

Luento 4

18.3.

Vaatimusmäärittely ketterässä prosessimallissa  
nopea kertaus

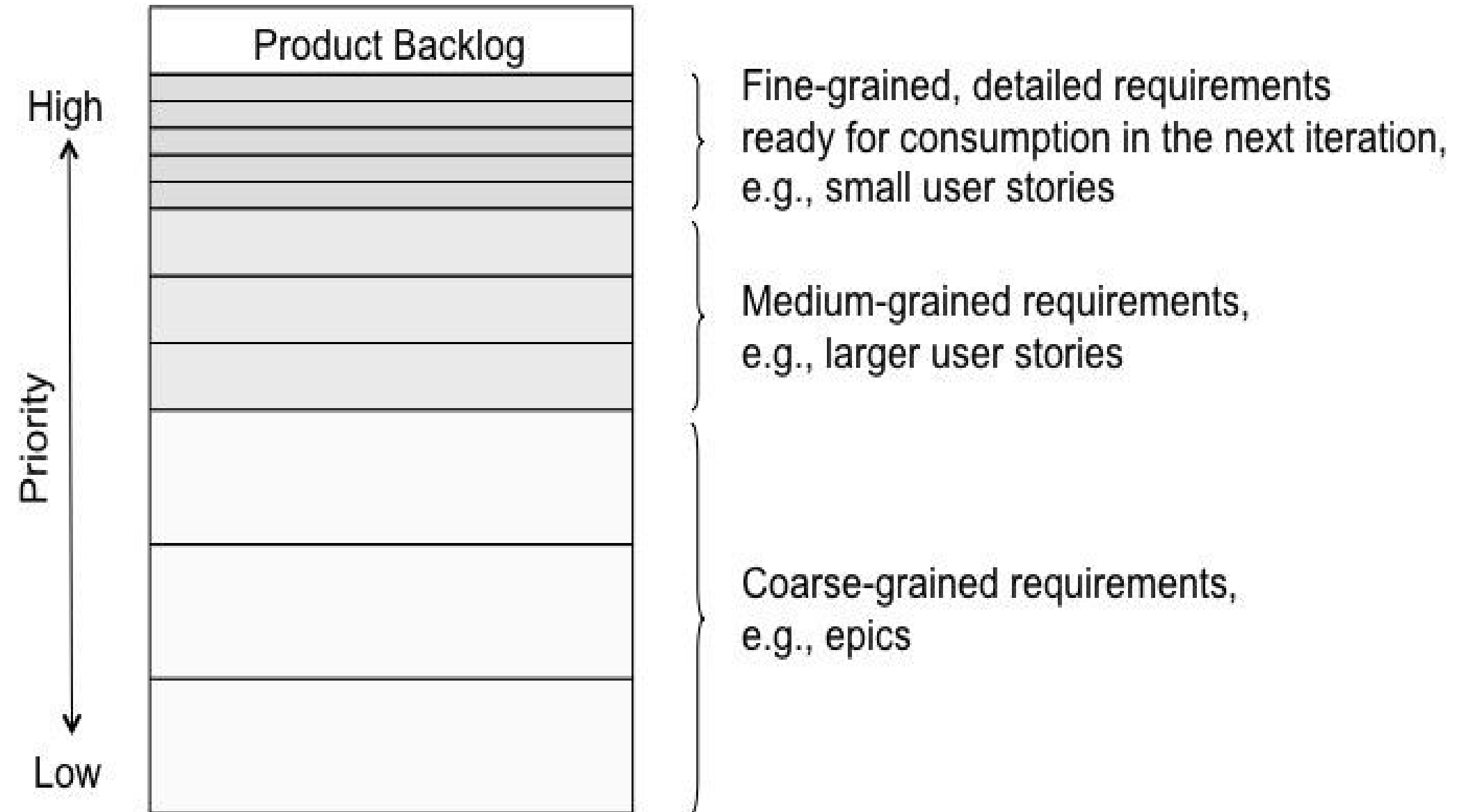
# Nopea kertaus eilisestä

- User story
  - "Määritelmä": CCC (card, conversation, confirmation)
  - INVEST
- Estimointi
  - Miksi?
  - Miten?
  - Kuka?
- Product Backlog
  - Kuka vastuussa?
  - Miten saadaan projektin alussa muodostettua?
- Velositeetti
  - Miten määritellään?

# Hyvä product backlog on DEEP

- **Detailed** appropriatly 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
- **Estimated**
- **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.
- **Prioritized**

# Hyvä product backlog on DEEP



# Sprintin suunnittelu

# Sprintin/iteraation suunnittelu

- Kertauksena viime viikolta: Scrum määrittelee pidettäväksi ennen jokaista sprinttiä kaksiosaisen suunnittelupalaverin
- Ensimmäisen osan tarkoitus 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 sitoutuu niin moneen Backlogin vaatimukseen kuin se arvioi kykenevänsä sprintin aikana toteuttamaan
- Sprintin aikana toteutettavien vaatimusten lisäksi asetetaan sprintin tavoite
- Suunnittelukokouksen toisen osan aikana selvitetään **miten sprintin tavoitteet saavutetaan**
  - Toisen osan aikana 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

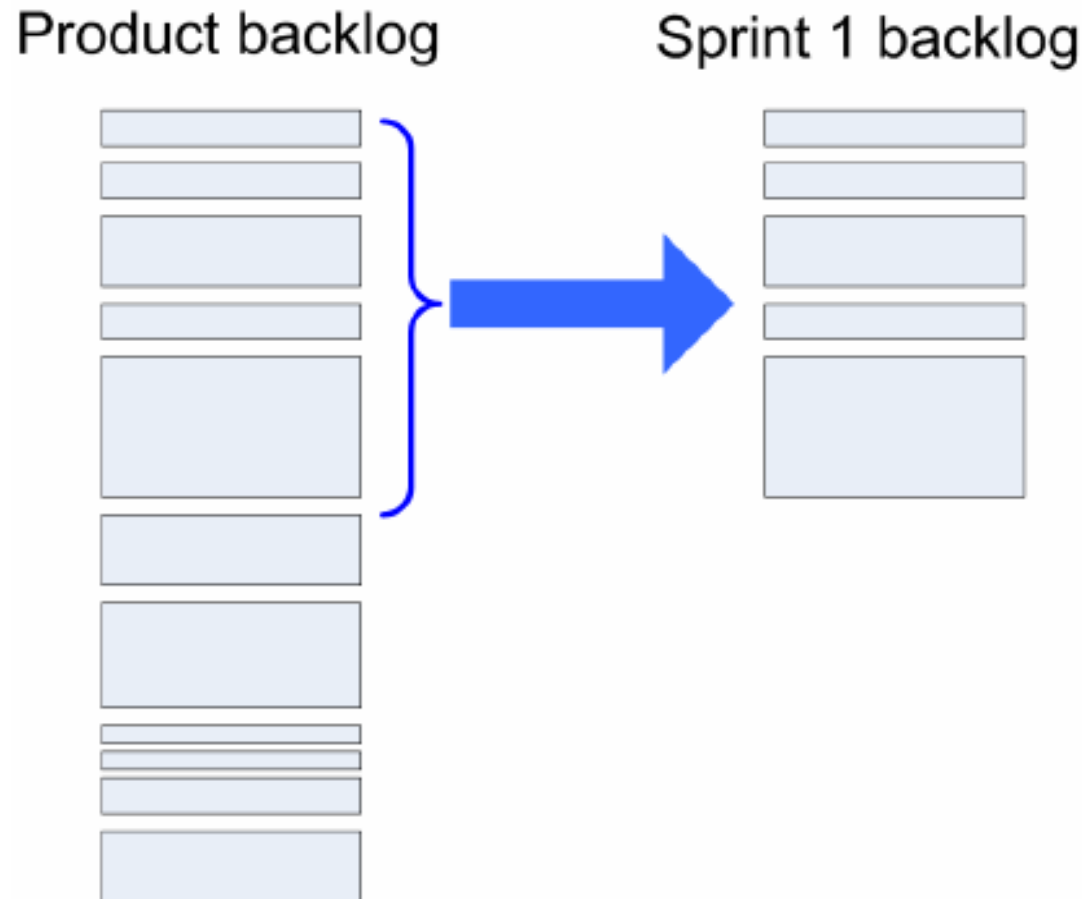


# Sprintin tavoite [Mike Cohn]

- 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.mountangoatsoftware.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**

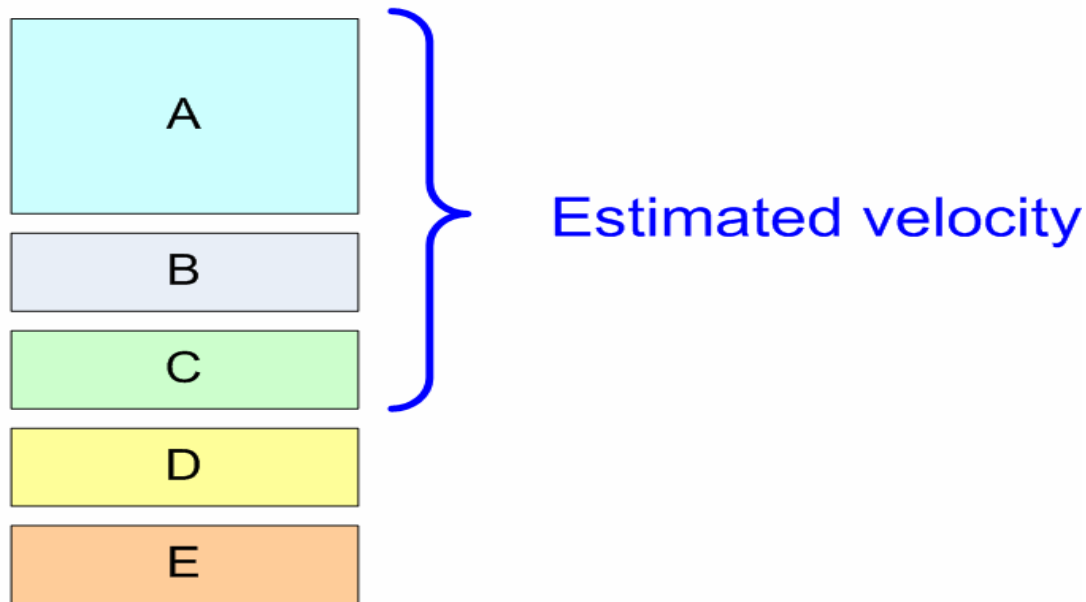
# Sprintiin otettavien User storyjen määrä

- Kehitystiimi siis päättää kuinka monta User storyä sprintiin otetaan toteutettavaksi
- Tapoja päättää kuinka monta User storyä sprintiin otetaan on muutamia:
  - ”perstuntuma”: otetaan niin monta korkeimman prioriteetin Storyä kuin mihin kaikki tiimiläiset tuntevat voivansa sitoutua
  - Jos storyt on estimoitu ja tiimin velositeetti tunnetaan, otetaan sprintiin 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 sitoutua vain sellaiseen määrään jonka se kokee voivansa toteuttaa kunnolla eli ”definition of donen” mielessä (eli suunnittelu, toteutus, automaattiset testit, testaus, integrointi, dokumentointi) valmiiksi
  - Velositeetin käsite ja estimaatithan huomioivat ”definition of donen”

# Toteutettavien user storyjen valinta

- Jos tiimin velositetti 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

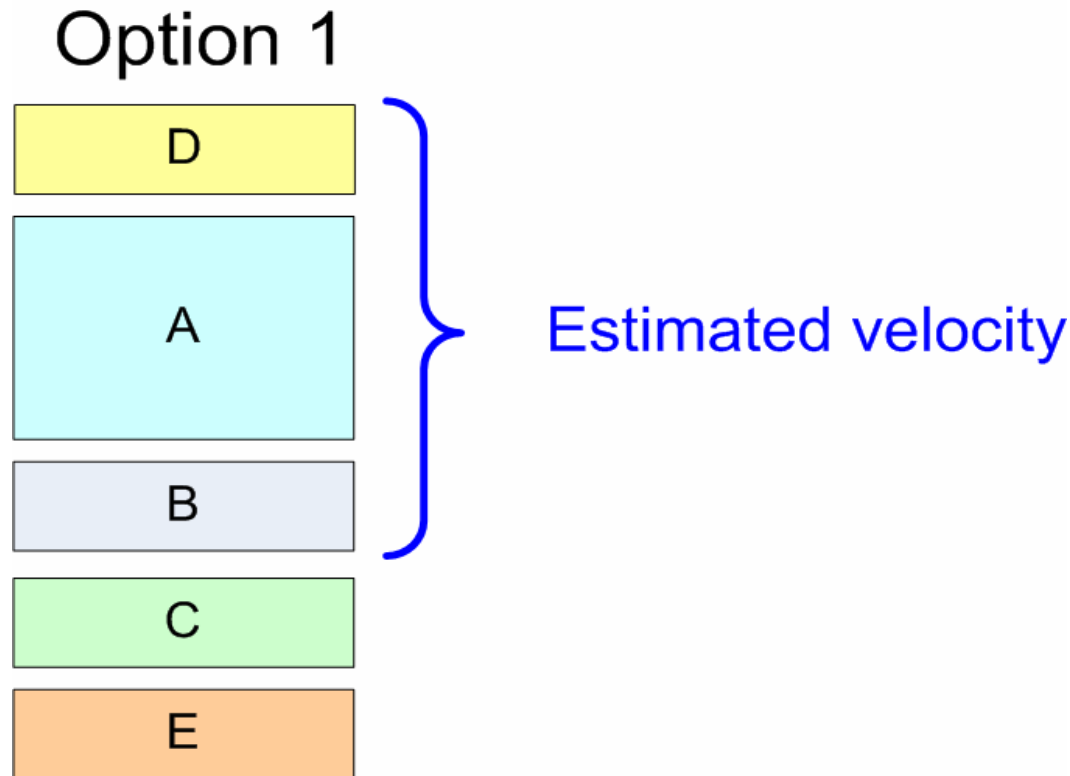
## Product backlog



- Product Owner haluaa storyn D mukaan sprinttiin

# Uudelleenpriorisointi

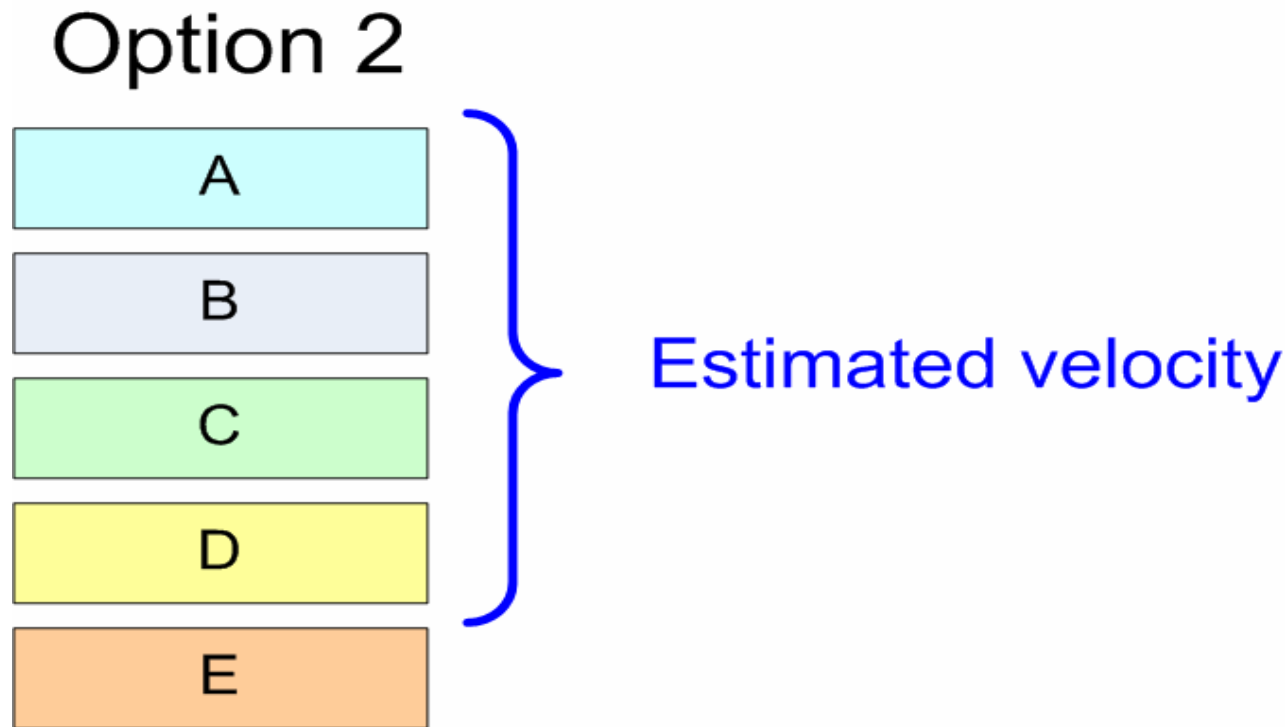
- Product Owner nostaa D:n prioriteettia, C tippuu pois sprinttiin valittavien User Storyjen joukosta



- Entä jos Product Owner haluaa Sprinttiin mukaan kaikki User Storyt A-D?

# User Storyn scopen pienentäminen

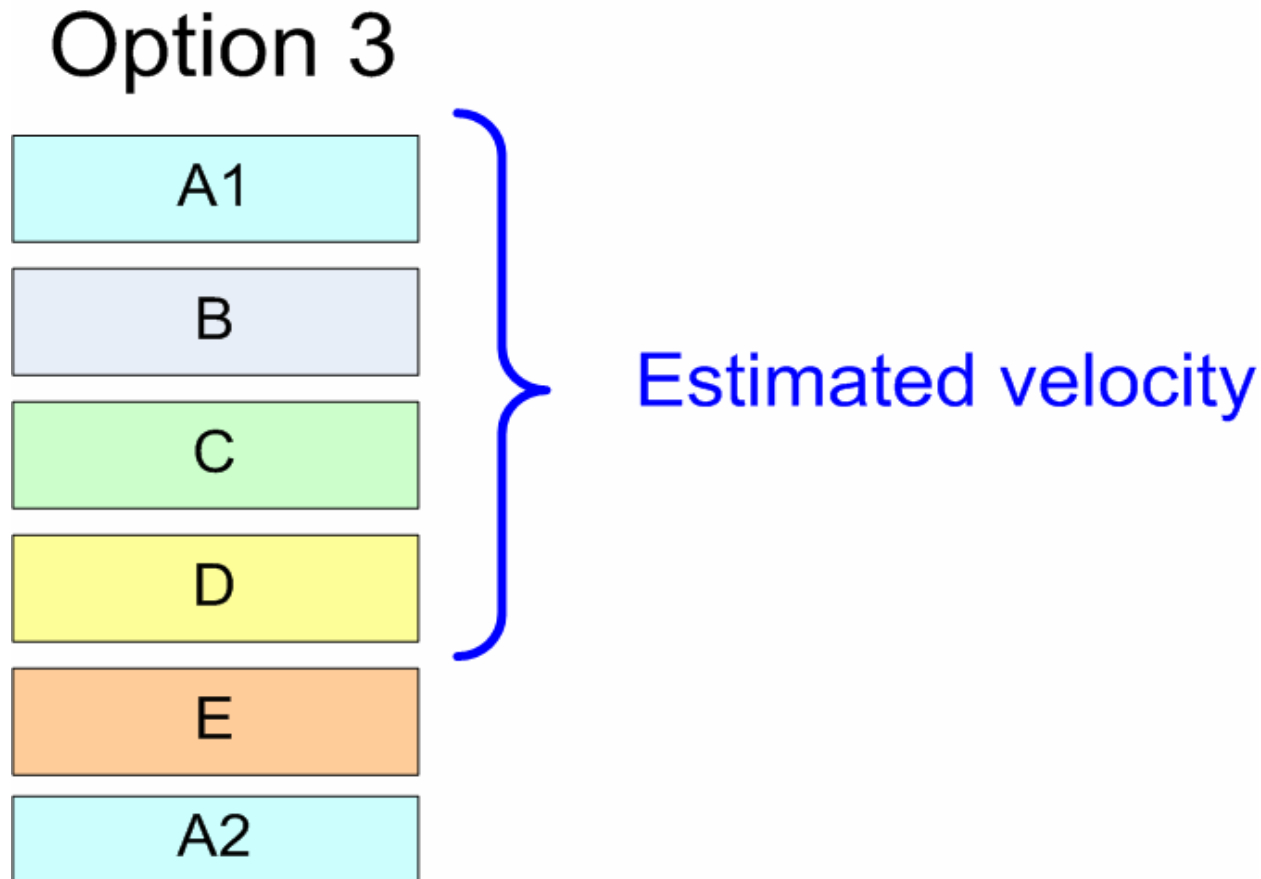
- *Jostain on luovuttava:* Product owner pienentää User storyn A määrittelemää toiminnallisuutta, kehitystiimi estimoi pienennetyn A:n ja nyt A-D mahtuvat sprinttiin:



- 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 helppoa, 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 publish a news story directly to the corporate website.
- ...I can publish a news story with editor review.
- ...I can publish a news story with legal review.
- ...I can view a news story on a staging site.
- ...I can publish a news story from the staging site to production

- Pattern #2: **Business Rule Variations**

- As a user, I can search for flights with flexible dates.
- ==>
- ...as “n days between x and y.”
- ...as “a weekend in December.”
- ...as “ $\pm$  n days of 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 one credit card type (of VISA, MC, DC, AMEX).
  - ...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.
  - ==>
  - ...specifying a max number of stops.
  - ...including nearby airports.
  - ...using flexible dates.
  - ...etc.

# 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**

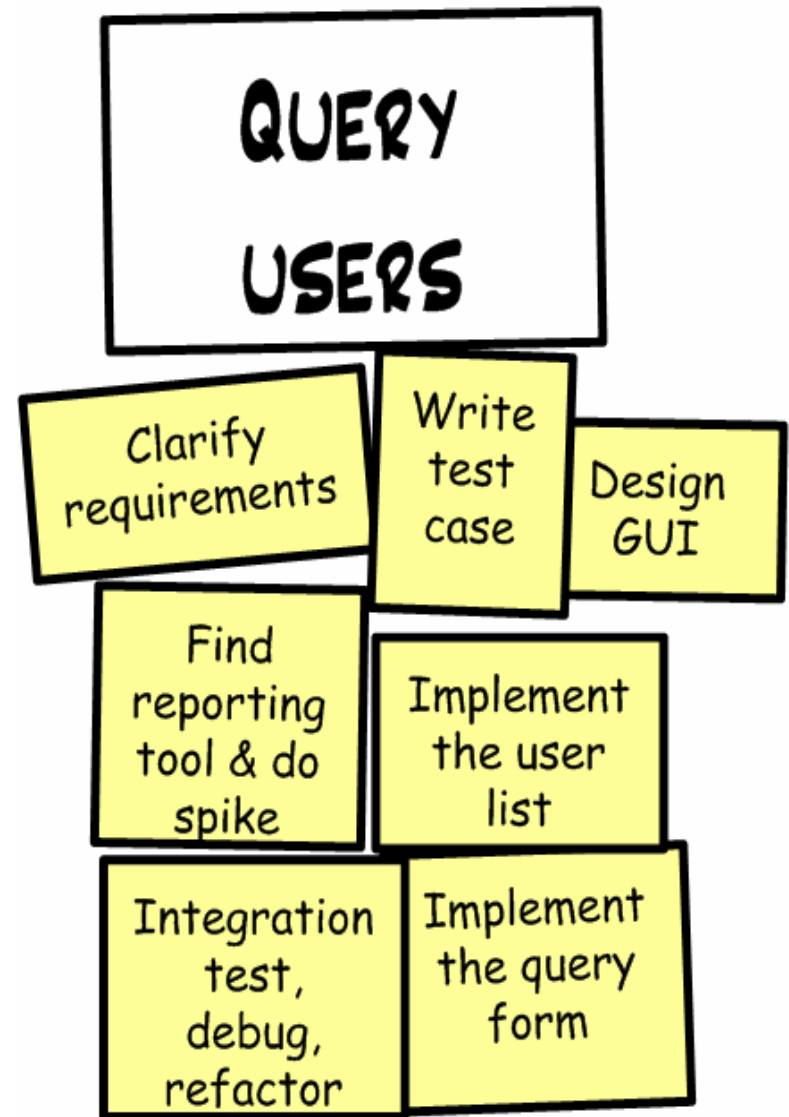
- A story may be large not because it's necessarily complex, but because the implementation is poorly understood. In this case, no amount of talking about the business part of the story will allow you to break it up. Do a time-boxed spike first to resolve uncertainty around the implementation. Then, 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.

# Planning game

- Extreme Programming (XP) kutsuu sprintiin (tai XP:n termein iteraatioon) otettavien User storyjen valintaa nimellä **planning game**:
  - **Customers have the most information about value: what best serves the organization.**
  - **Programmers have the most information about cost: what it takes to implement and maintain those features.**
  - Every decision to do something is a decision to not do something else.
  - The planning game brings together customers and programmers so that they may maximize value while minimizing costs.
  - Anybody may create stories. **Programmers estimate the stories, and customers prioritize them.**
  - Programmers and customers may question each others' decisions, but each group has final say over its area of expertise.
  - The end result is a single prioritized list
- [http://jamesshore.com/Agile-Book/the\\_planning\\_game.html](http://jamesshore.com/Agile-Book/the_planning_game.html)

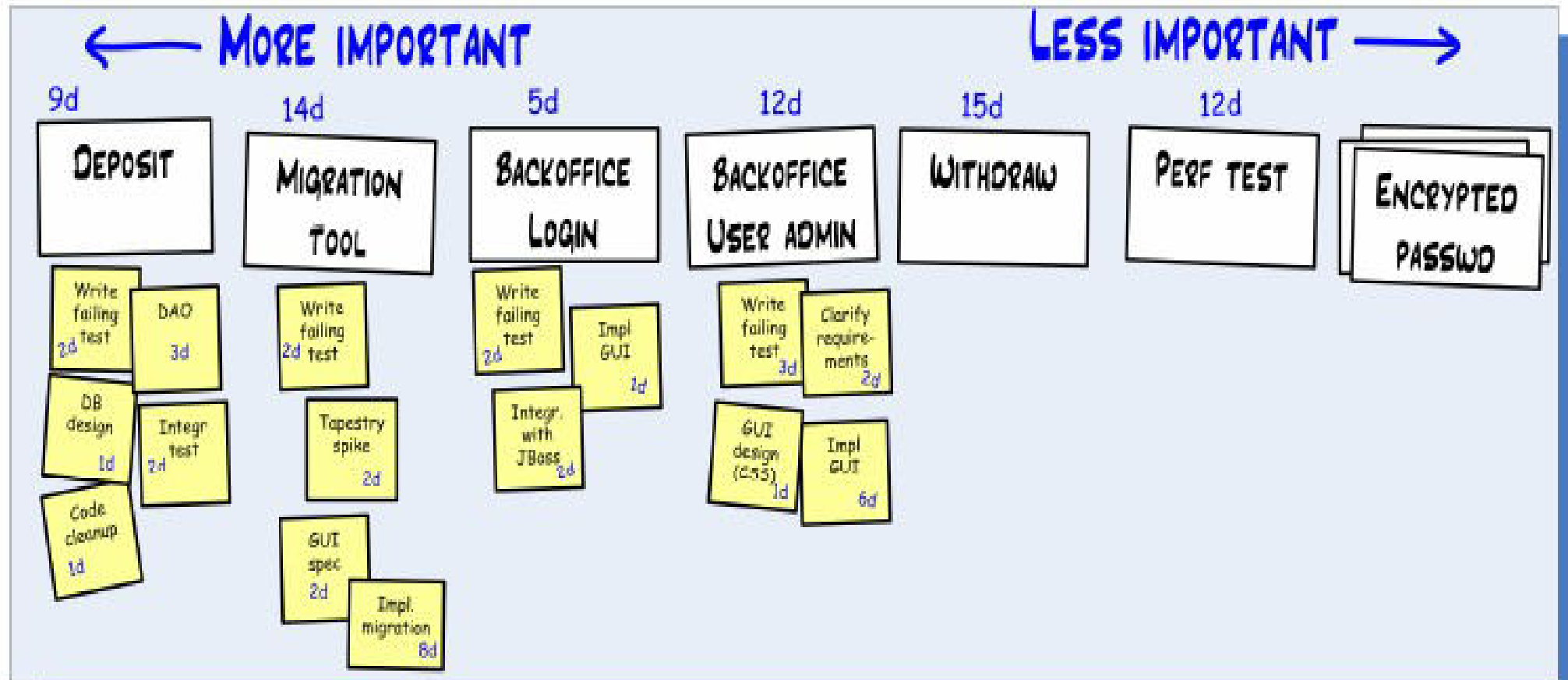
# Sprintin suunnittelun toinen vaihe

- 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



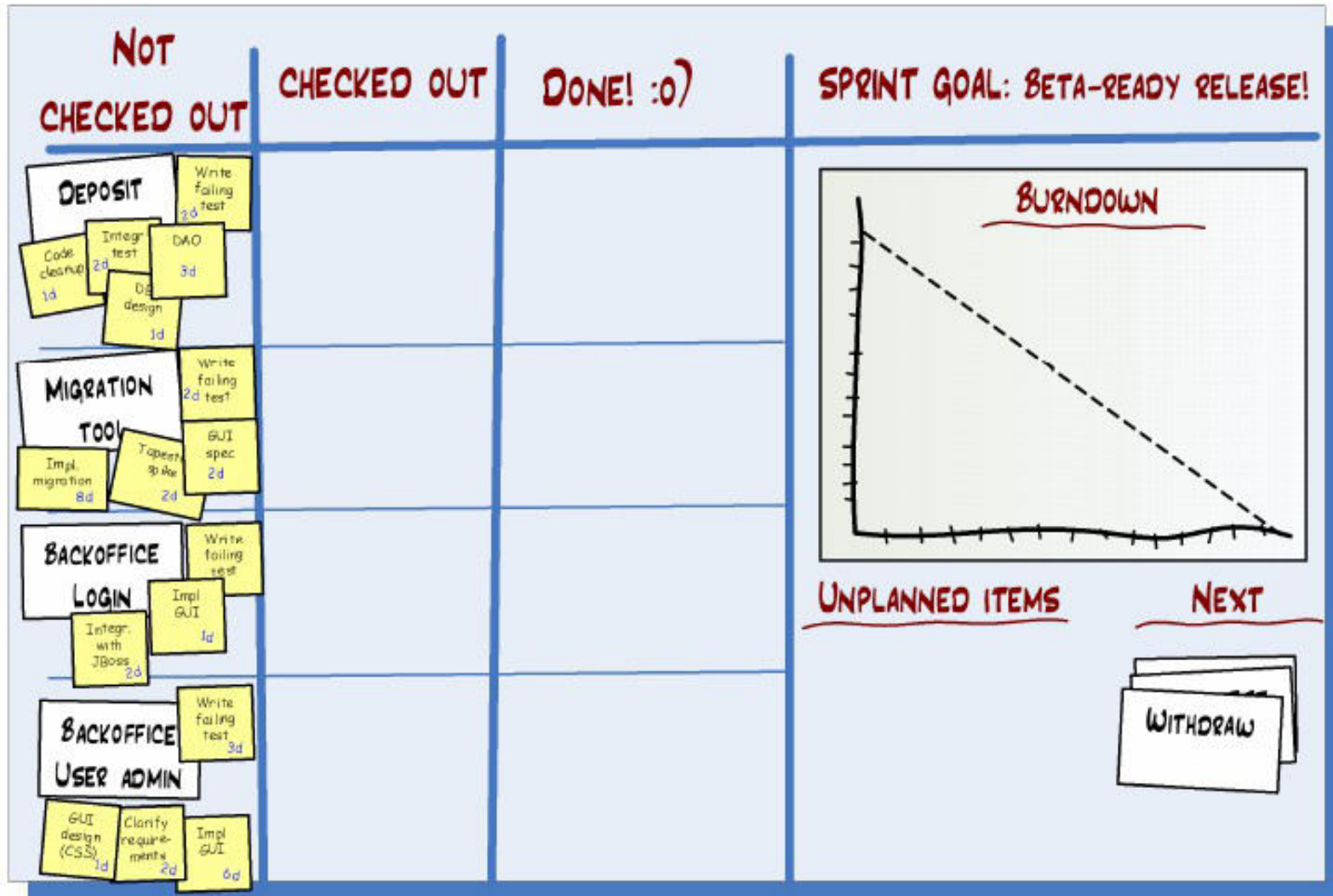
# Sprint backlogin muotoutuminen

- Sprintin suunnittelun "best practise" on monien asiantuntijoiden mielestä käyttää pahvikortteja User storyille ja postit-lappuja teknisen tason tehtäville ja käyttää näitä työvälineenä alustavan sprint backlogin muodostamisessa
  - Näin esim. priorisointi on vaivatonta
  - Elektronisessa muodossa oleva backlog voidaan päivittää tilanteen stabiloiduttua



# Sprint backlog

- Sprintin tehtävälista eli backlog koostuu sprintiin valituista User storeista ja Storeihin liittyvistä tehtävistä eli Taskeista
- Backlog voi olla organisoitu "taskboardiksi":





# Taskboard

- Kuvassa sprinttiin on valittu 4 User storyä ja taskboard on jaettu neljään kaistaan, 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 myös **tunteina**
- Jokaiseen Taskiin kirjataan sen arvioitu vaatima työmäärä
  - Jos käytössä on "manuaalinen" taskboard, kirjoitetaan arvio suoraan Taskia edustavaan postit-lappuun
  - Arviota päivitetään joka päivä
  - Arvio voi nousta jos Task huomataankin työläämmäksi mitä alun perin ajateltiin
- On myös mahdollista 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!

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

BURNDOWN

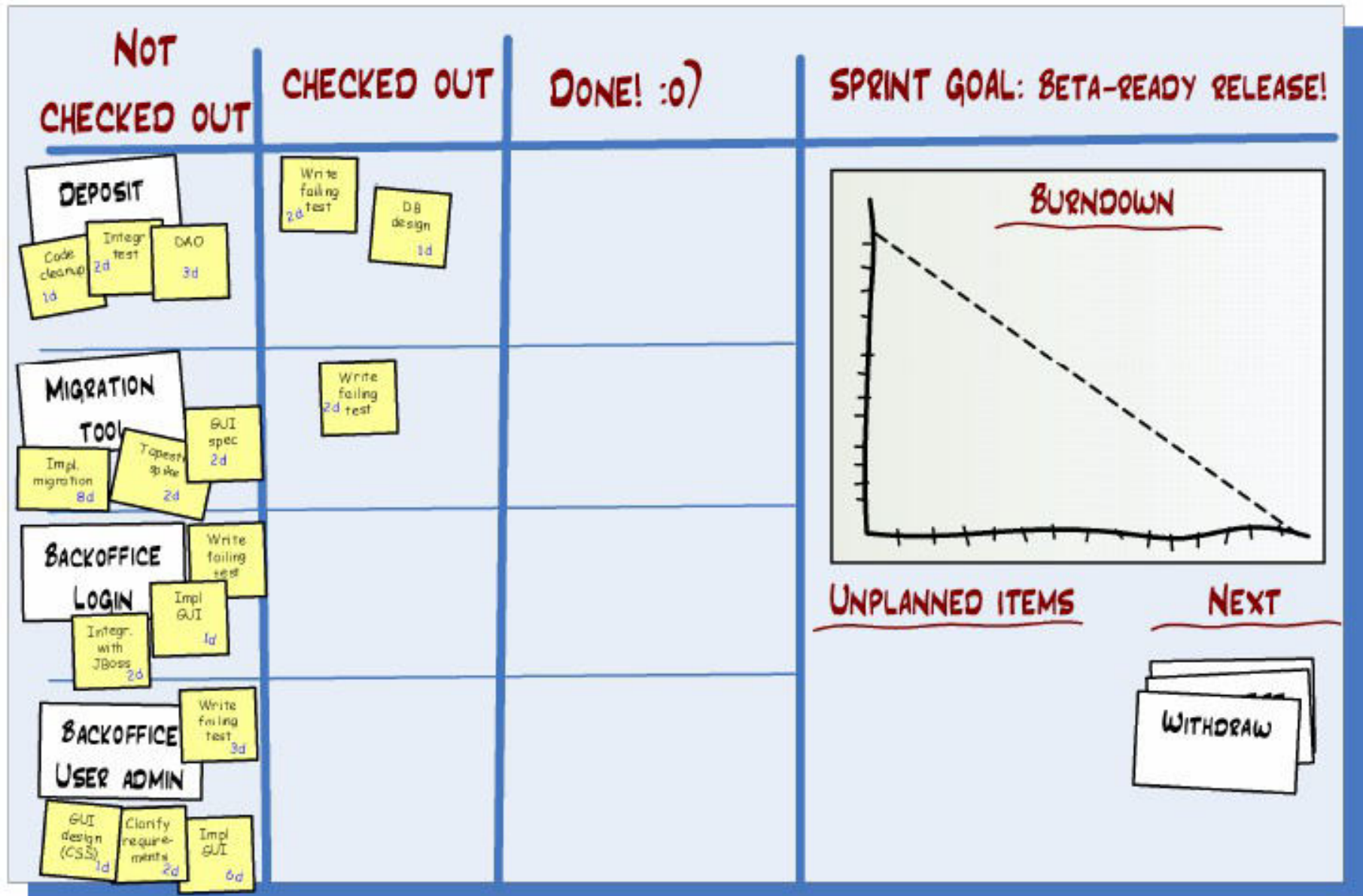
UNPLANNED ITEMS

NEXT

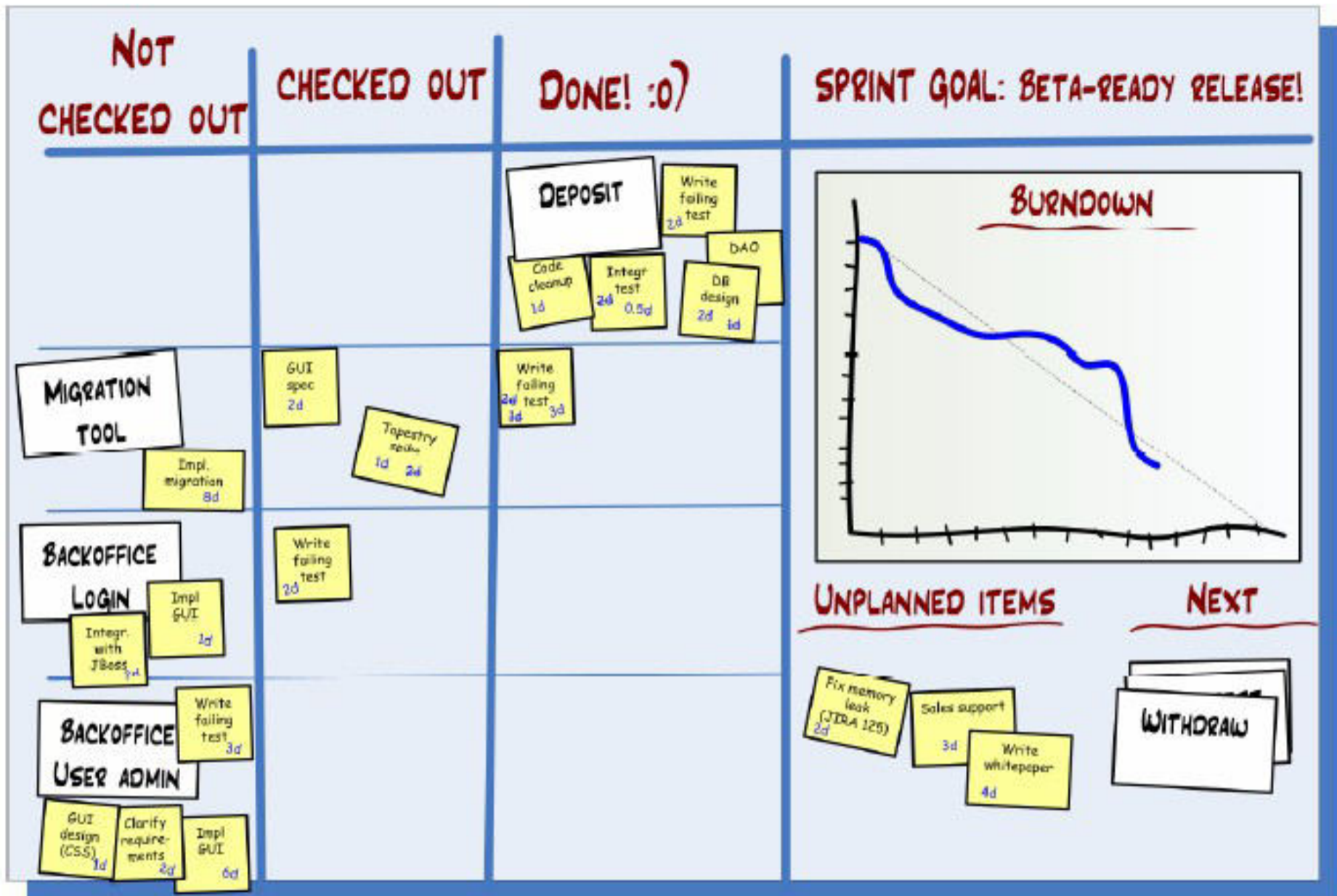
WITHDRAW

If all backlog items are completed before the sprint ends, add new ones from here.

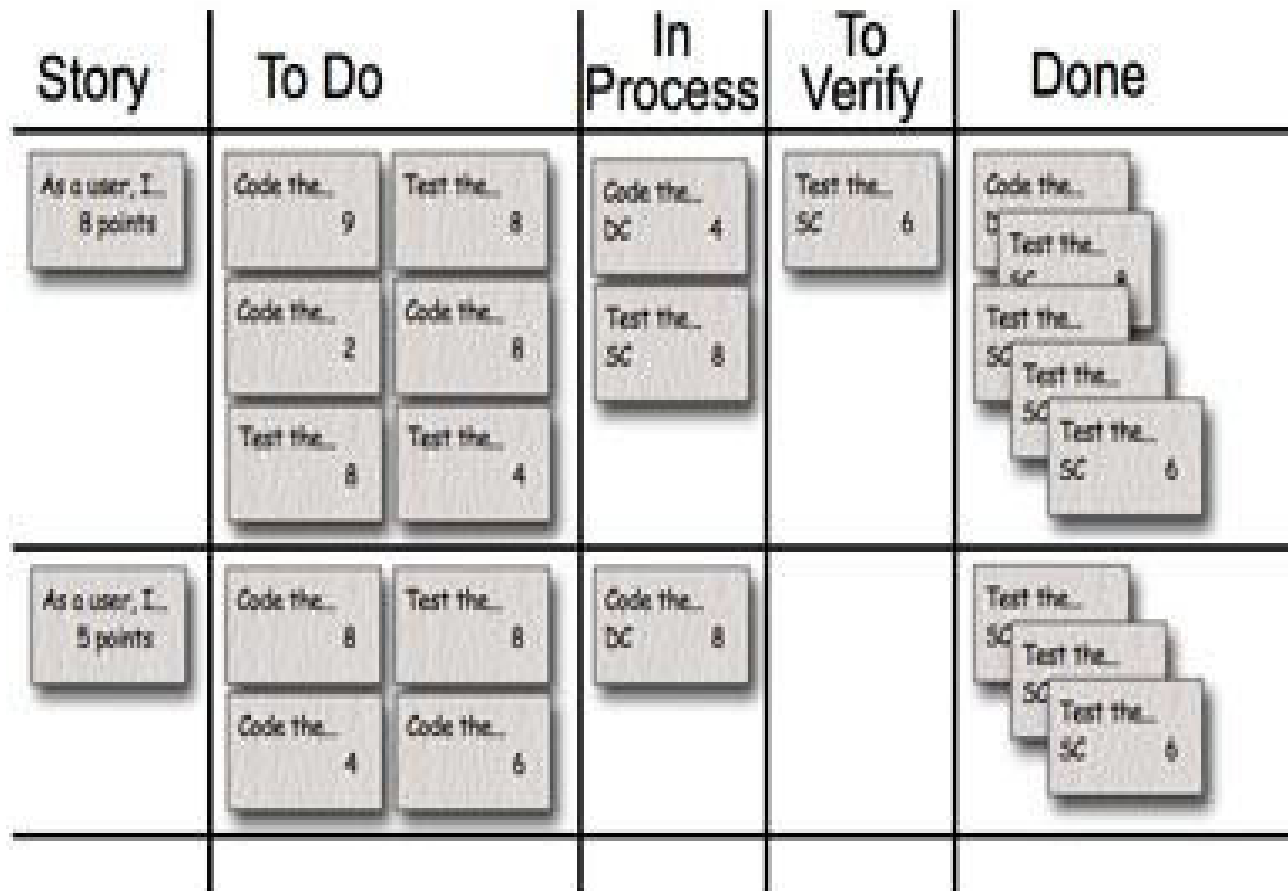
# Tilanne sprintin alussa



# Ja puolen välin jälkeen



# Taskboardissa voi olla merkattu useampiakin työvaiheita

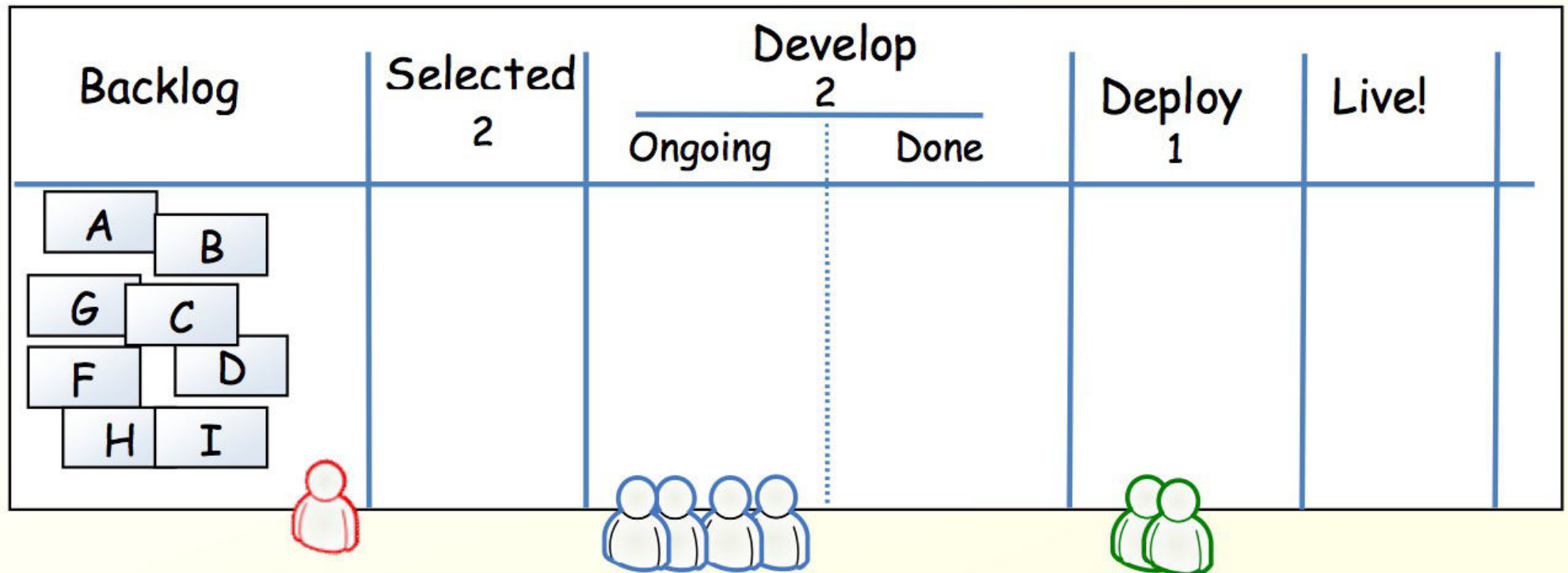


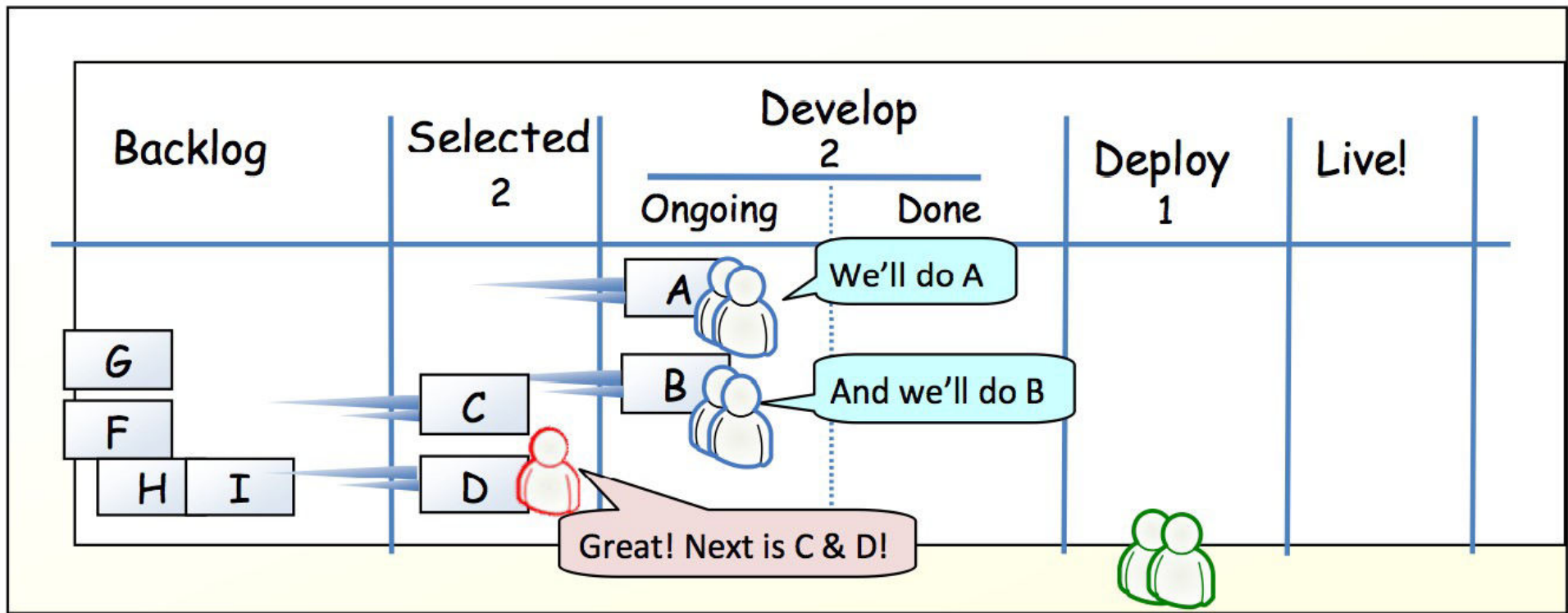
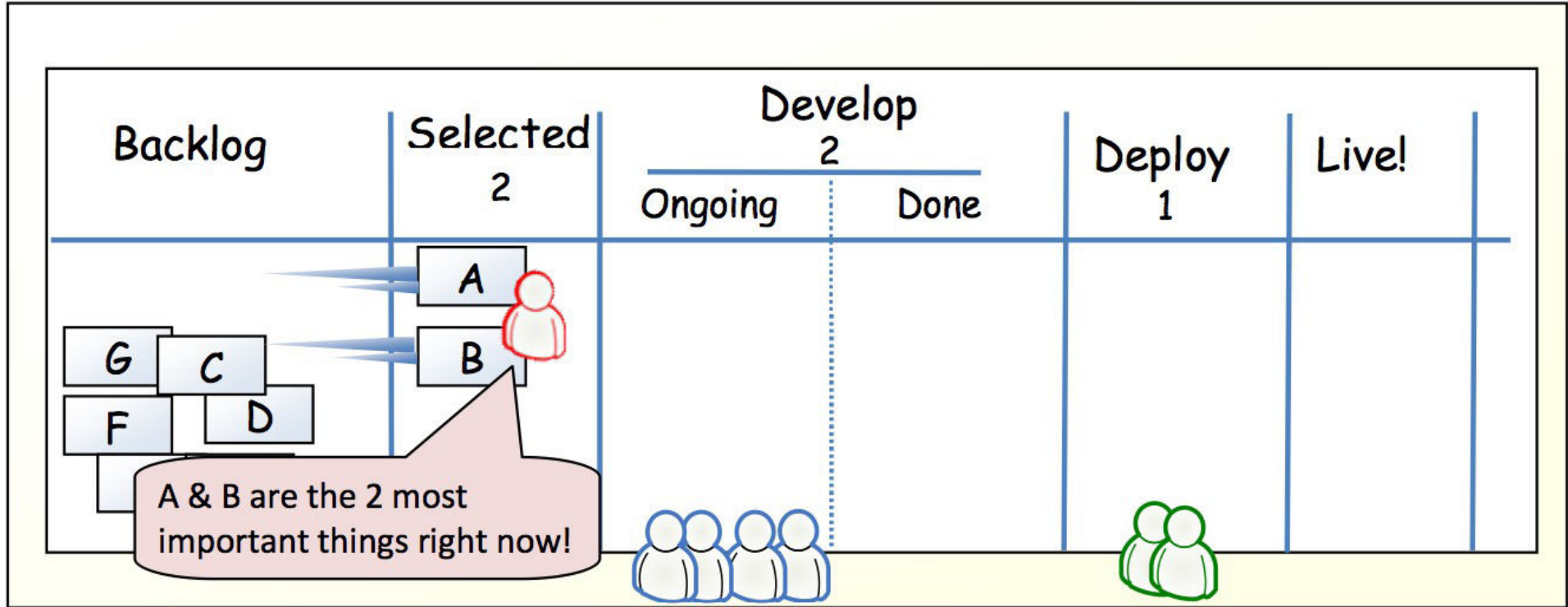
- Yhtäaikaa työn alla olevien taskien suuri määrä voi koitua Scrumissa ongelmaksi sillä riski sille, että sprintin päätyttyä on paljon osittain valmiita User storeja kasvaa

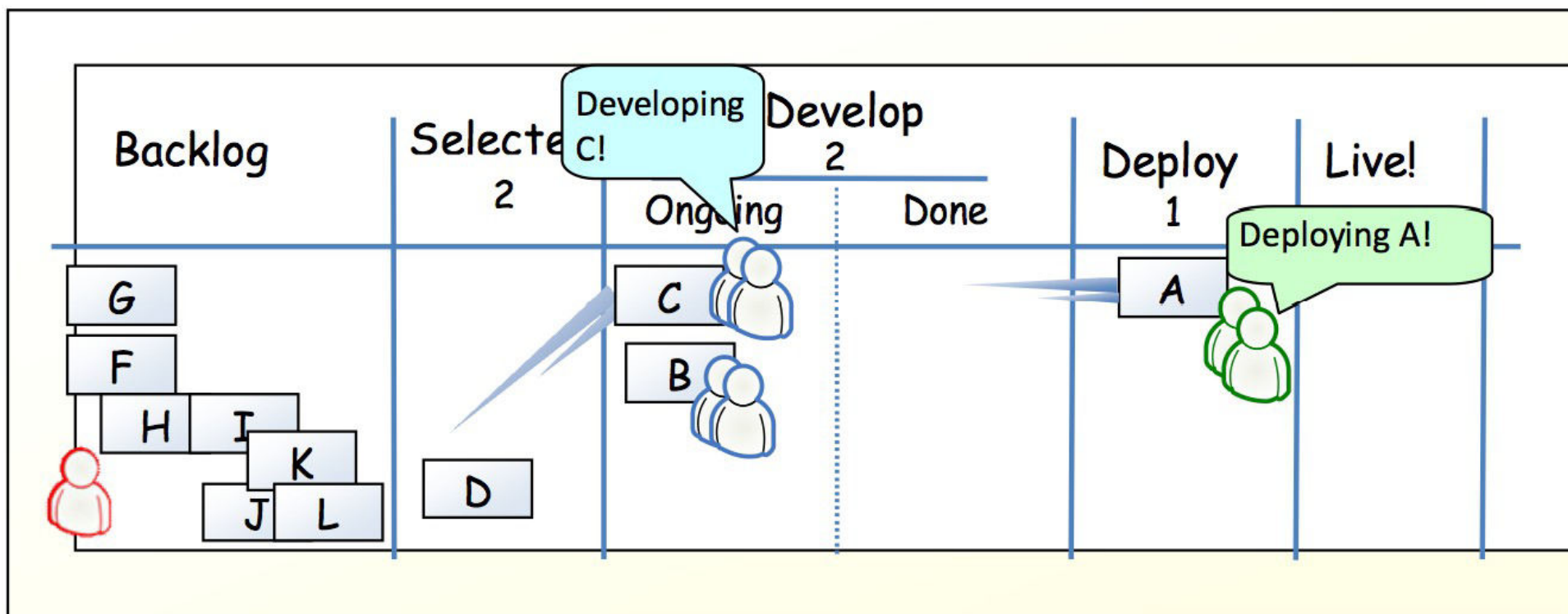
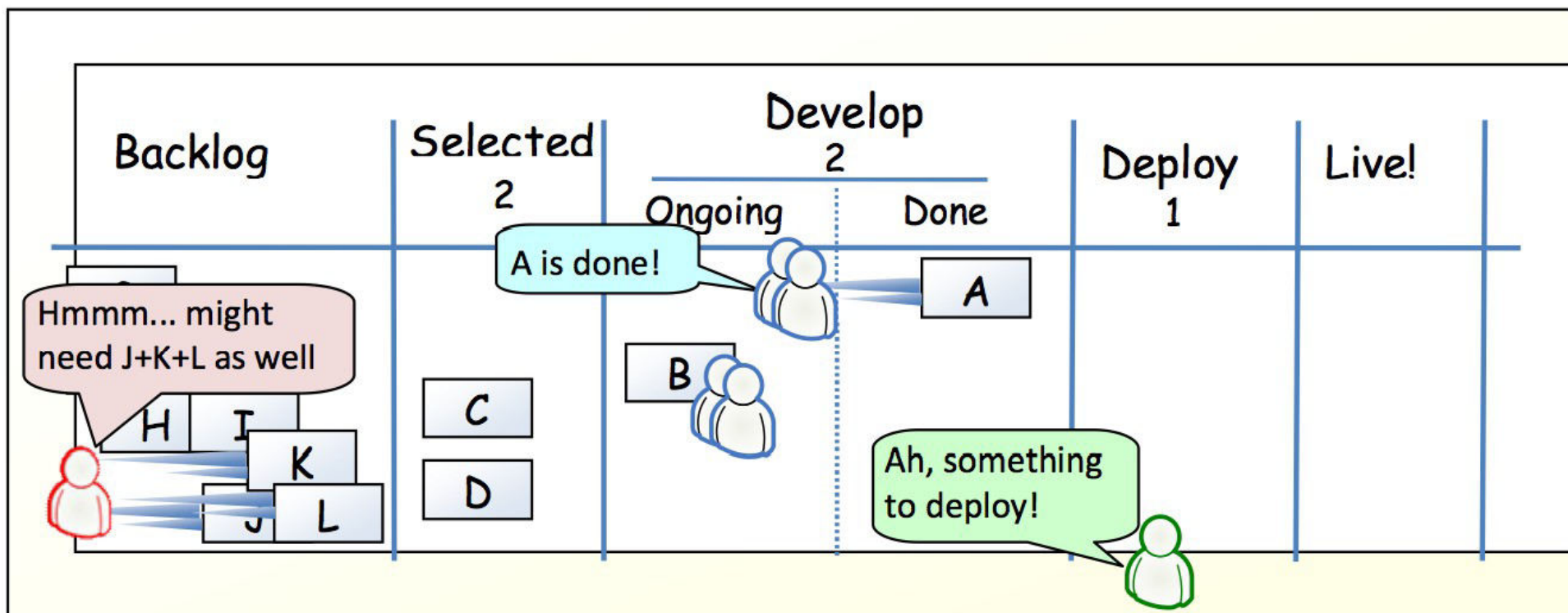


# Scrumban

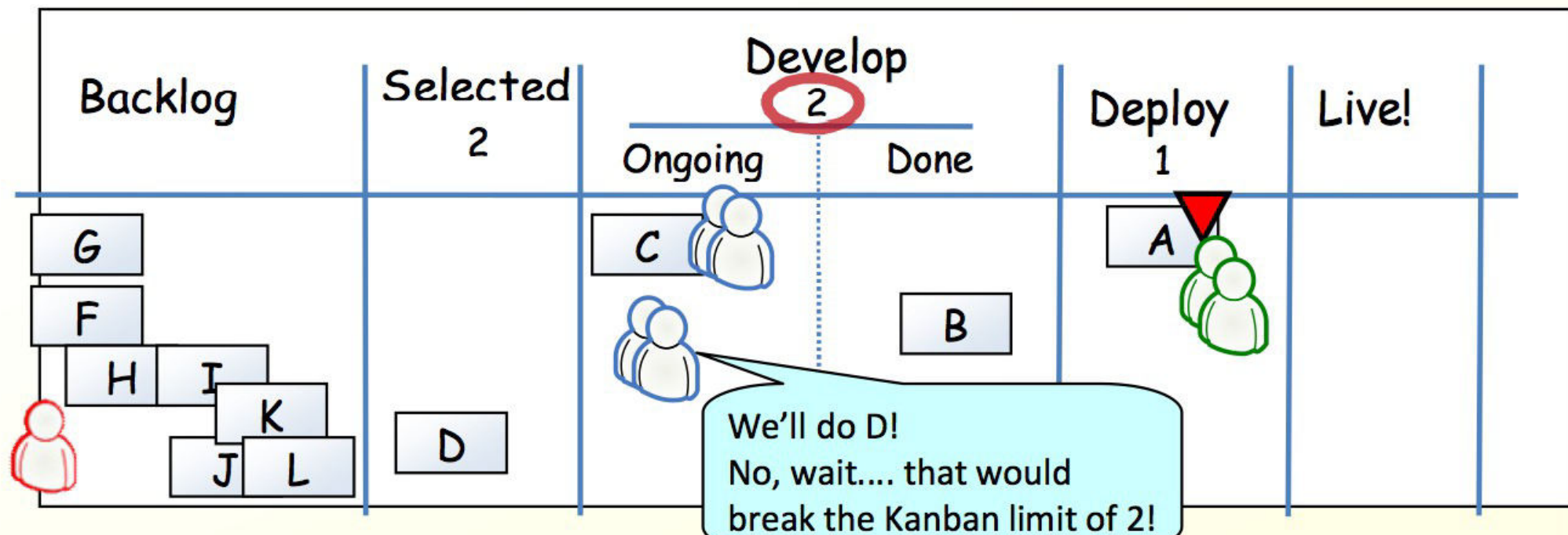
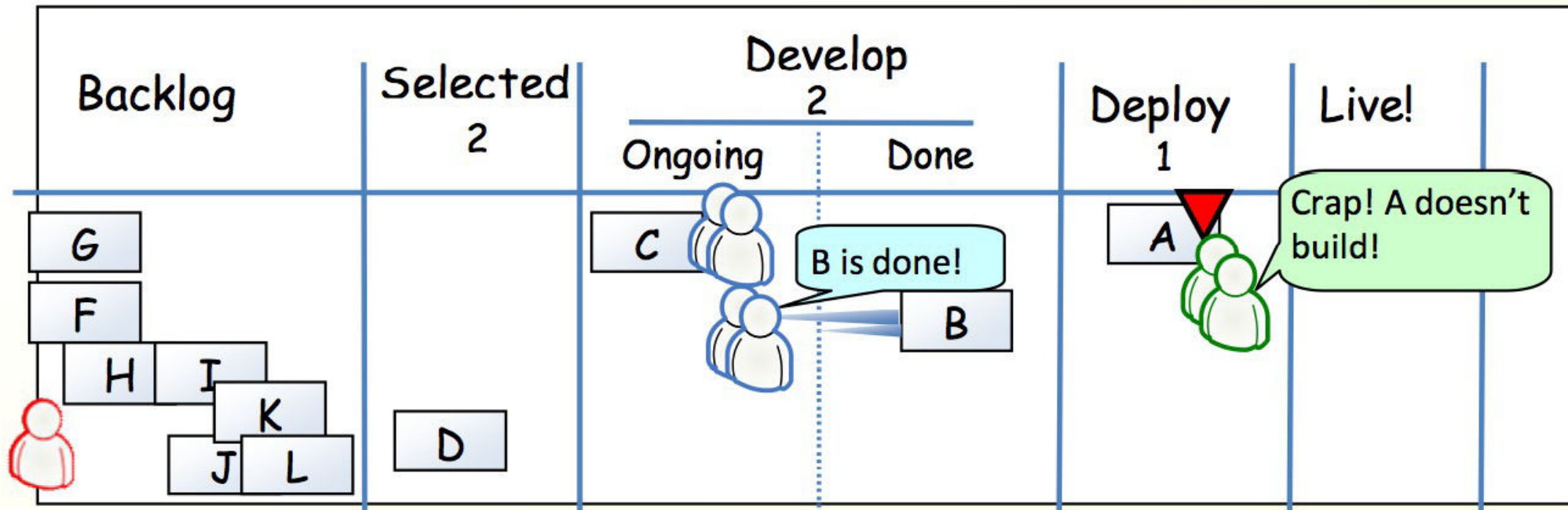
- Voikin olla mielekästä rajoittaa yhtäaikaan tekemisen alla olevien töiden määrää asettamalla "work in progress" (eli WIP) -rajoituksia
  - WIP-rajoitukset on lainattu Kanban-menetelmästä
  - Scrumin ja Kanbanin yhdistelmää kutsutaan nimellä Scrumban
- Seuraavassa esimerkki kirjasta  
<http://www.infoq.com/minibooks/kanban-scrum-minibook>

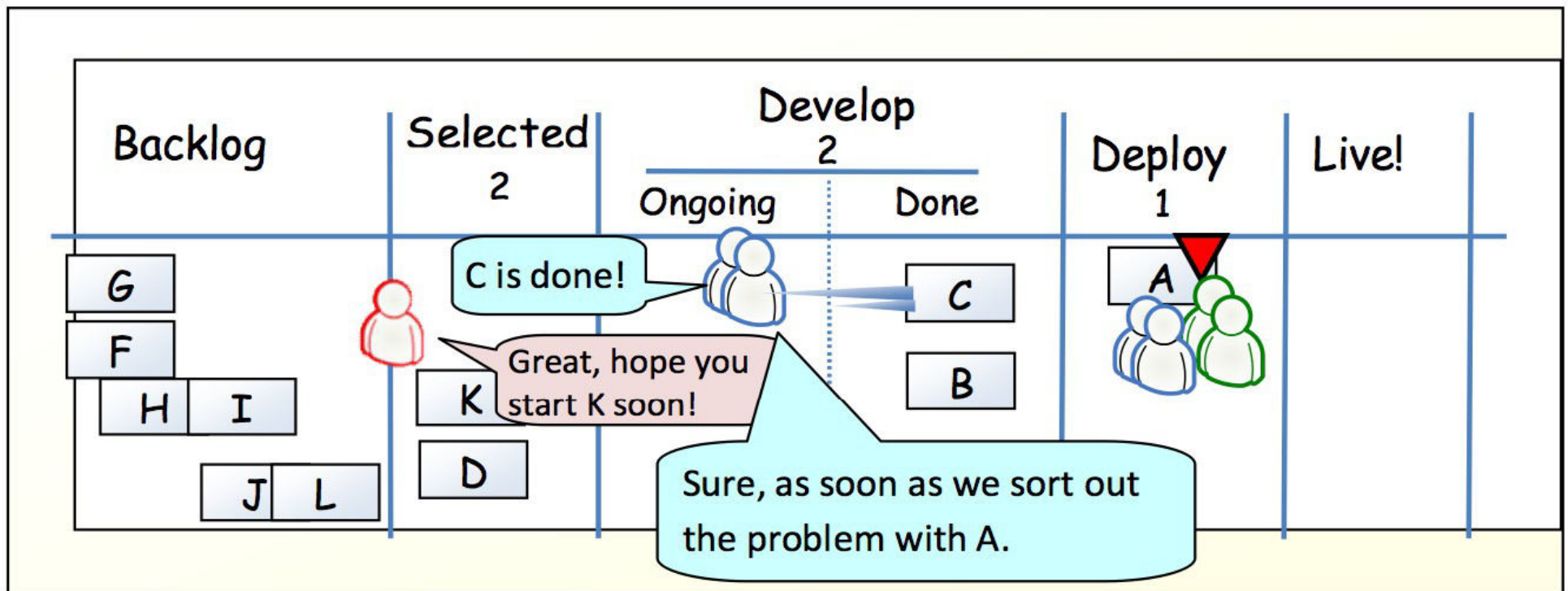
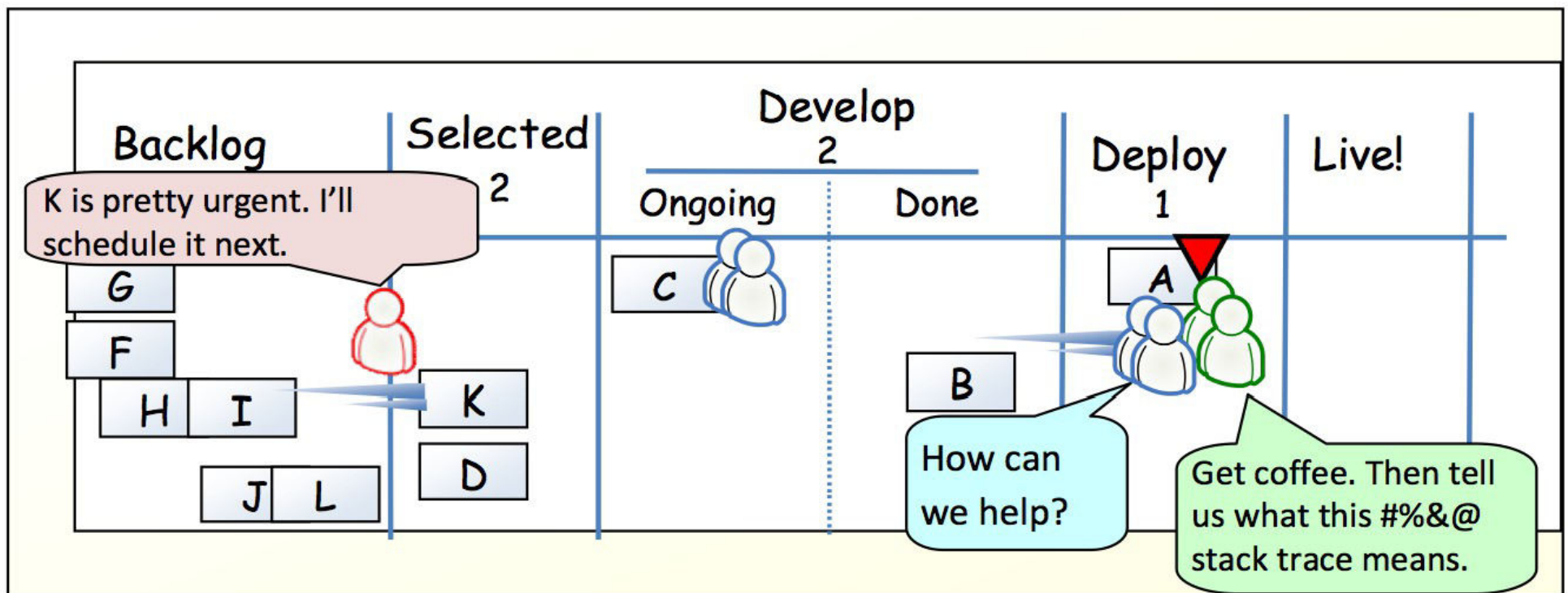


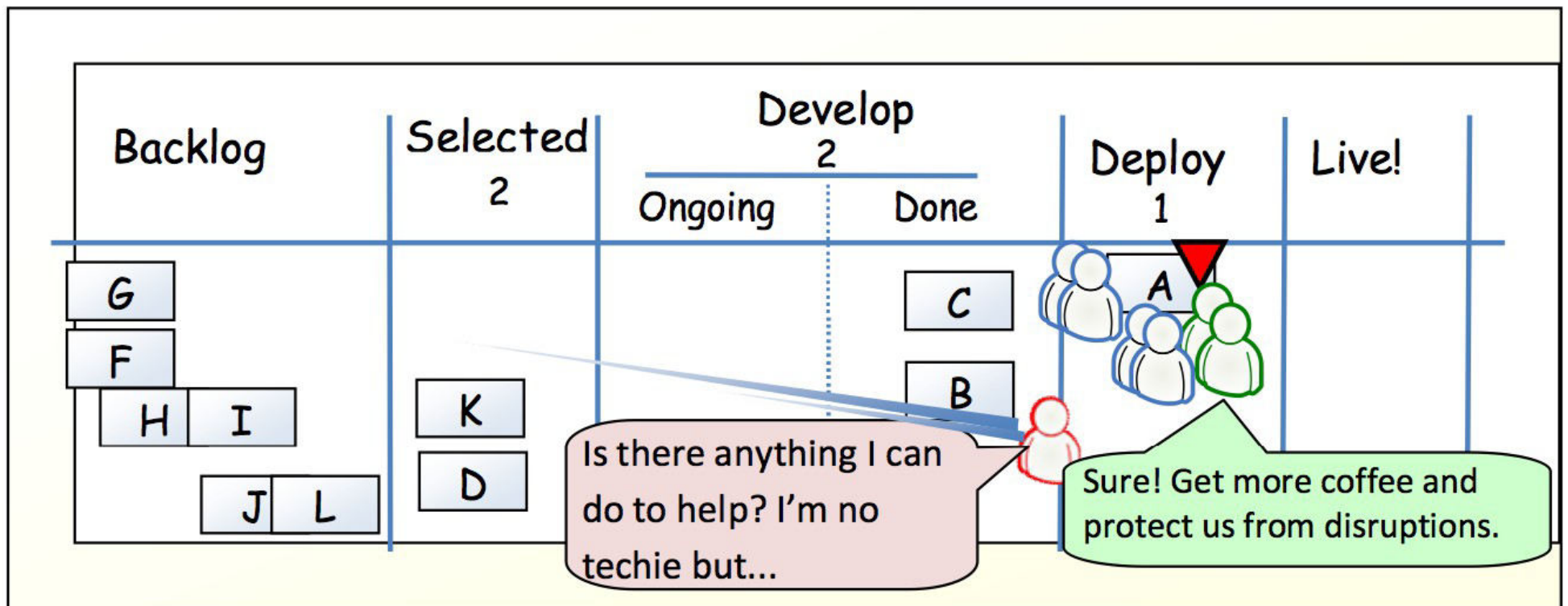
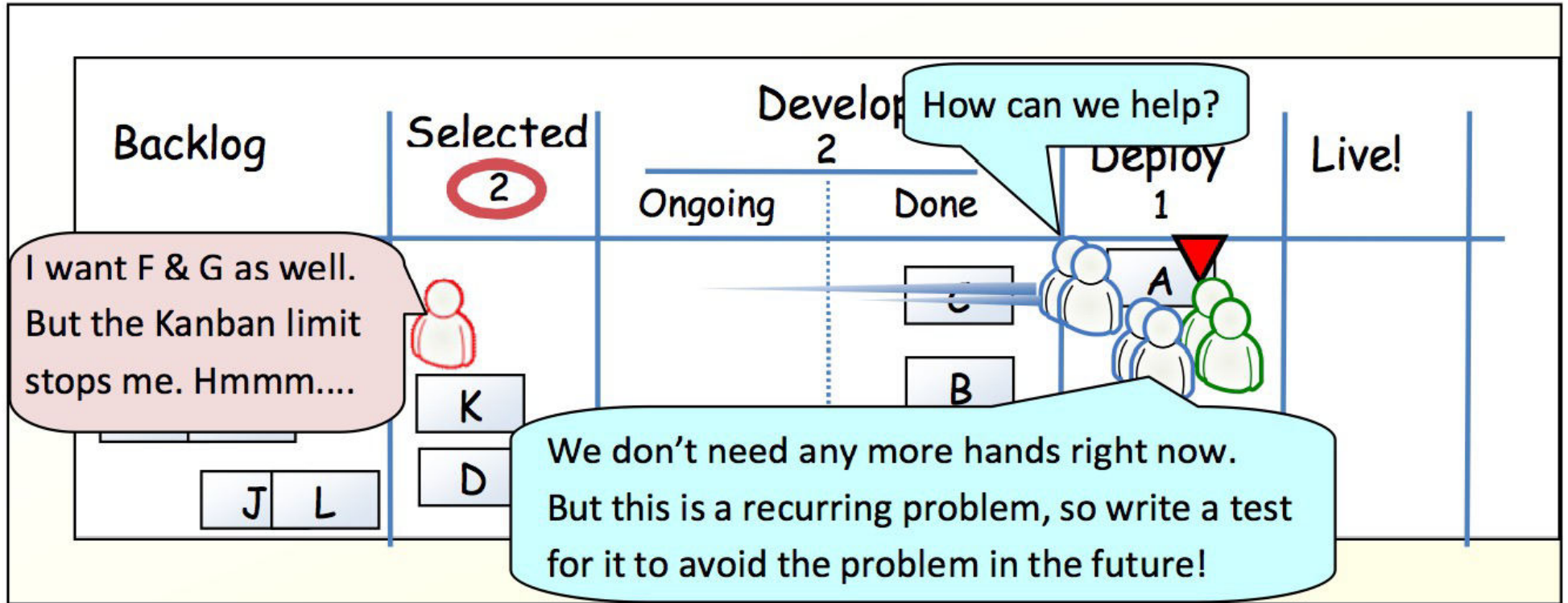












# Sprintissä jäljellä olevan työmäärän arviointi

- 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!





# 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 viime kesän ohtuprojektin product- ja sprintbacklogit:
  - [https://docs.google.com/spreadsheet/ccc?key=0Aul6gm-diqbXdExfcHh0R1RGb18zbUMvblZvR09TUXc&usp=drive\\_web#gid=1](https://docs.google.com/spreadsheet/ccc?key=0Aul6gm-diqbXdExfcHh0R1RGb18zbUMvblZvR09TUXc&usp=drive_web#gid=1)

# 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 avoimuutta 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.mountangoatsoftware.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
  - Pivotal tracker
  - issueträkkerit (esim. trac, bugzilla)
  - [http://en.wikipedia.org/wiki/Comparison\\_of\\_issue\\_tracking\\_systems](http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems)

# 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 olemassaolevia storyjä on muutettu ja uudelleenpriorisoitu
- On suositeltavaa että kehitystiimi käyttää pienen määrän aikaa sprintin aikana product backlogin vaatimiin toimiin, esim. uusien User storyjen estimointiin
- Jos product backlog on hyvässä kunnossa (DEEP) sprintin loppuessa, on jälleen helppo lähteä sprintin suunnitteluun ja uuteen sprinttiin