What is an Agile Tester?

Colombo Agile Conf, June 2014

Henrik Kniberg

henrik.kniberg@crisp.se
@HenrikKniberg

Consultant

crisp
www.crisp.se

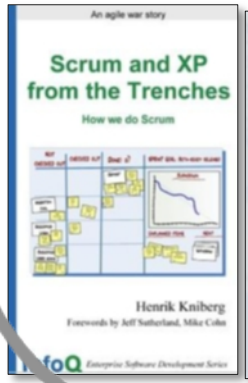
Father

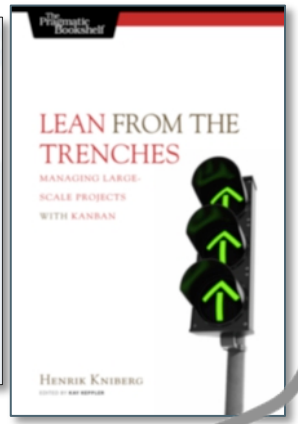
Agile & Lean coach

Spotify

skype

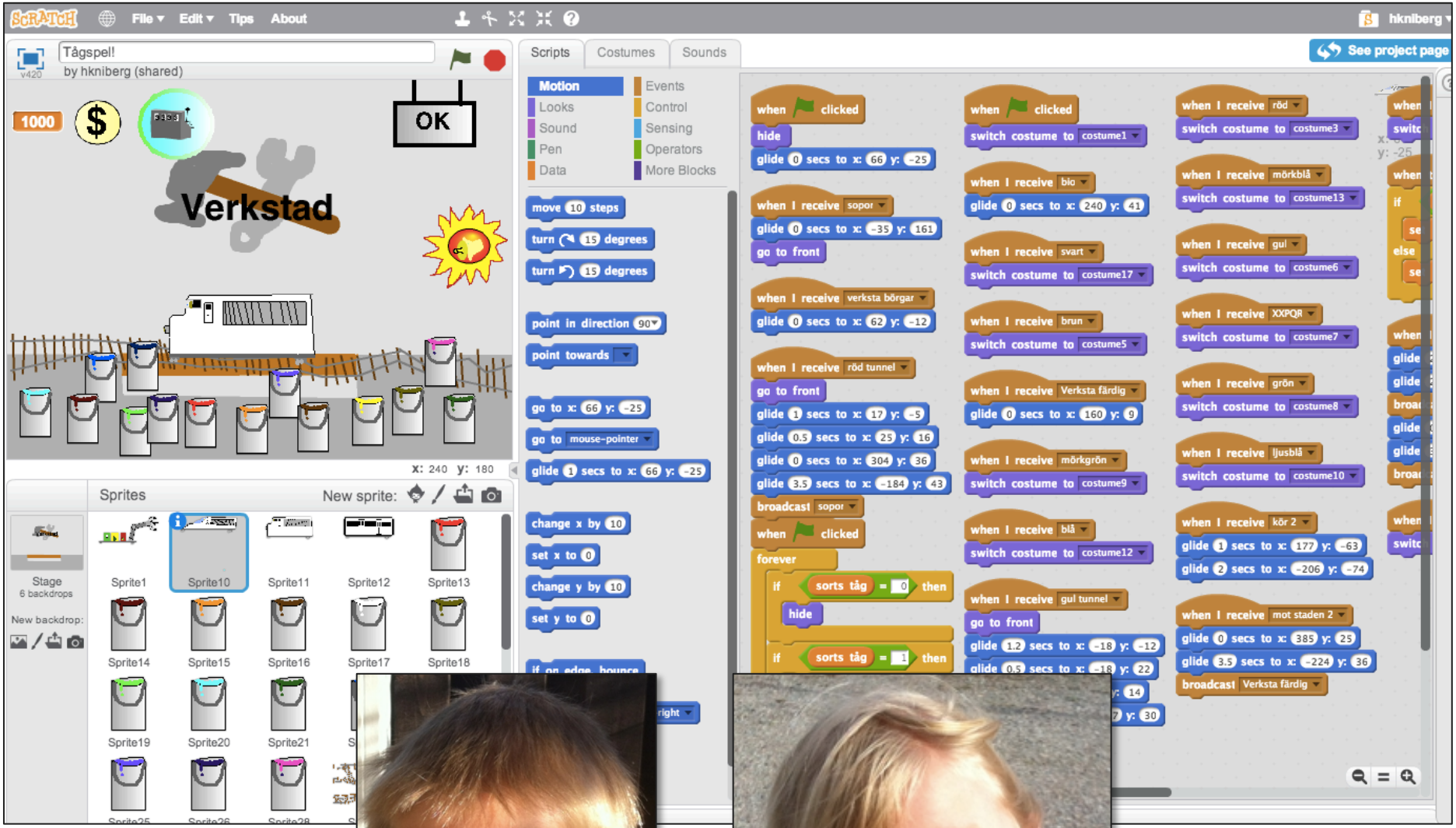
Author





Agile tester mindset

Henrik Kniberg



Henrik Kniberg

COVER FEATURE

Iterative and Incremental Development: A Brief History

Although many view iterative and incremental development as a modern practice, its application dates as far back as the mid-1950s. Prominent software-engineering thought leaders from each succeeding decade supported IID practices, and many large projects used them successfully.

Craig Larman
Valtech

Victor R. Basili
University of Maryland

As agile methods become more popular, some view iterative, evolutionary, and incremental software development—a cornerstone of these methods—as the "modern" replacement of the waterfall model, but its practiced and published roots go back decades. Of course, many software-engineering students are aware of this, yet surprisingly, some commercial and government organizations still are not.

This description of projects and individual contributions provides compelling evidence of iterative and incremental development's (IID's) long existence. Many examples come from the 1970s and 1980s—the most active but least known part of IID's history. We are mindful that the idea of IID came independently from countless unnamed projects and the contributions of thousands and that this list is merely representative. We do not mean this article to diminish the unsung importance of other IID contributors.

We chose a chronology of IID projects and approaches rather than a deep comparative analysis. The methods varied in such aspects as iteration length and the use of time boxing. Some attempted significant up-front specification work followed by incremental time-boxed development, while others were more classically evolutionary and feedback driven. Despite their differences, however, all the approaches had a common theme—to avoid a single-pass sequential, document-driven, gated-step approach.

Finally, a note about our terminology: Although some prefer to reserve the phrase "iterative development" merely for rework, in modern agile methods the term implies not just revisiting work, but also evolutionary advancement—a usage that dates from at least 1968.

PRE-1970

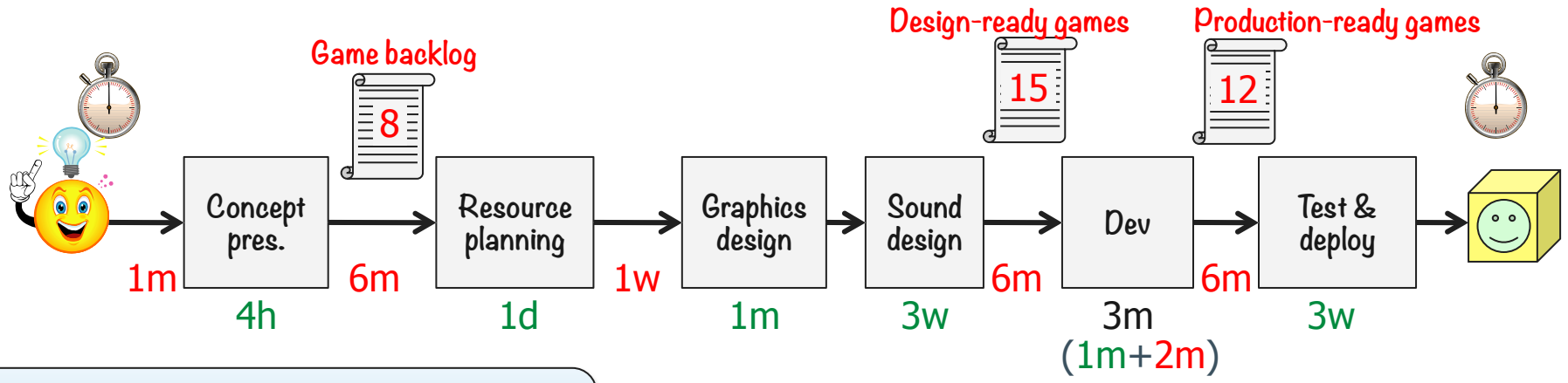
IID grew from the 1930s work of Walter Shewhart,[1] a quality expert at Bell Labs who proposed a series of short "plan-do-study-act" (PDSA) cycles for quality improvement. Starting in the 1940s, quality guru W. Edwards Deming began vigorously promoting PDSA, which he later described in 1982 in *Out of the Crisis*.[2] Tom Gilb[3] and Richard Zultner[4] also explored PDSA application to software development in later works.

The X-15 hypersonic jet was a milestone 1950s project applying IID,[5] and the practice was considered a major contribution to the X-15's success. Although the X-15 was not a software project, it is noteworthy because some personnel—and hence, IID experience—seeded NASA's early 1960s Project Mercury, which did apply IID in software. In addition, some Project Mercury personnel seeded the IBM Federal Systems Division (FSD), another early IID proponent.

Project Mercury ran with very short (half-day) iterations that were time boxed. The development team conducted a technical review of all changes, and, interestingly, applied the Extreme Programming practice of test-first development, planning and writing tests before each micro-increment. They also practiced top-down development with stubs

Henrik Kniberg

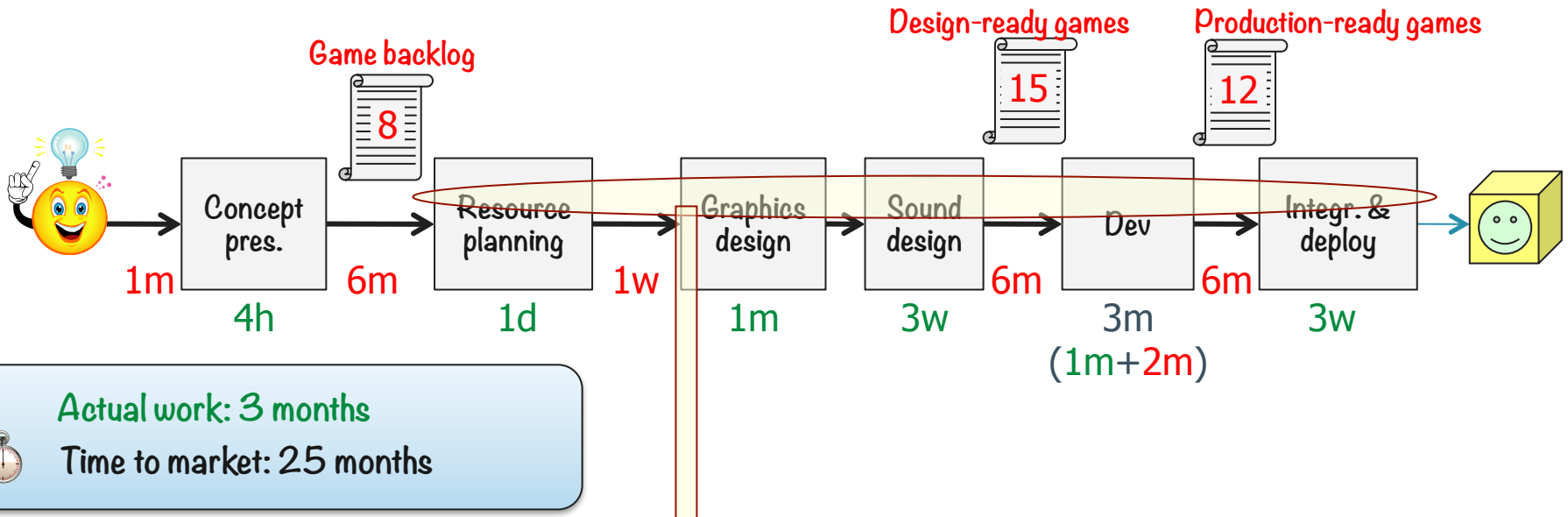
Case study: Game development company



Actual work: 3 months

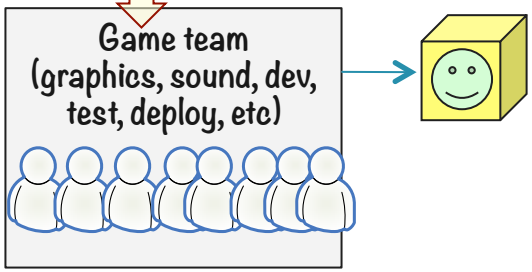
Time to market: 25 months

Before



After

Cross-functional game teams



Time to market: 3-4 months

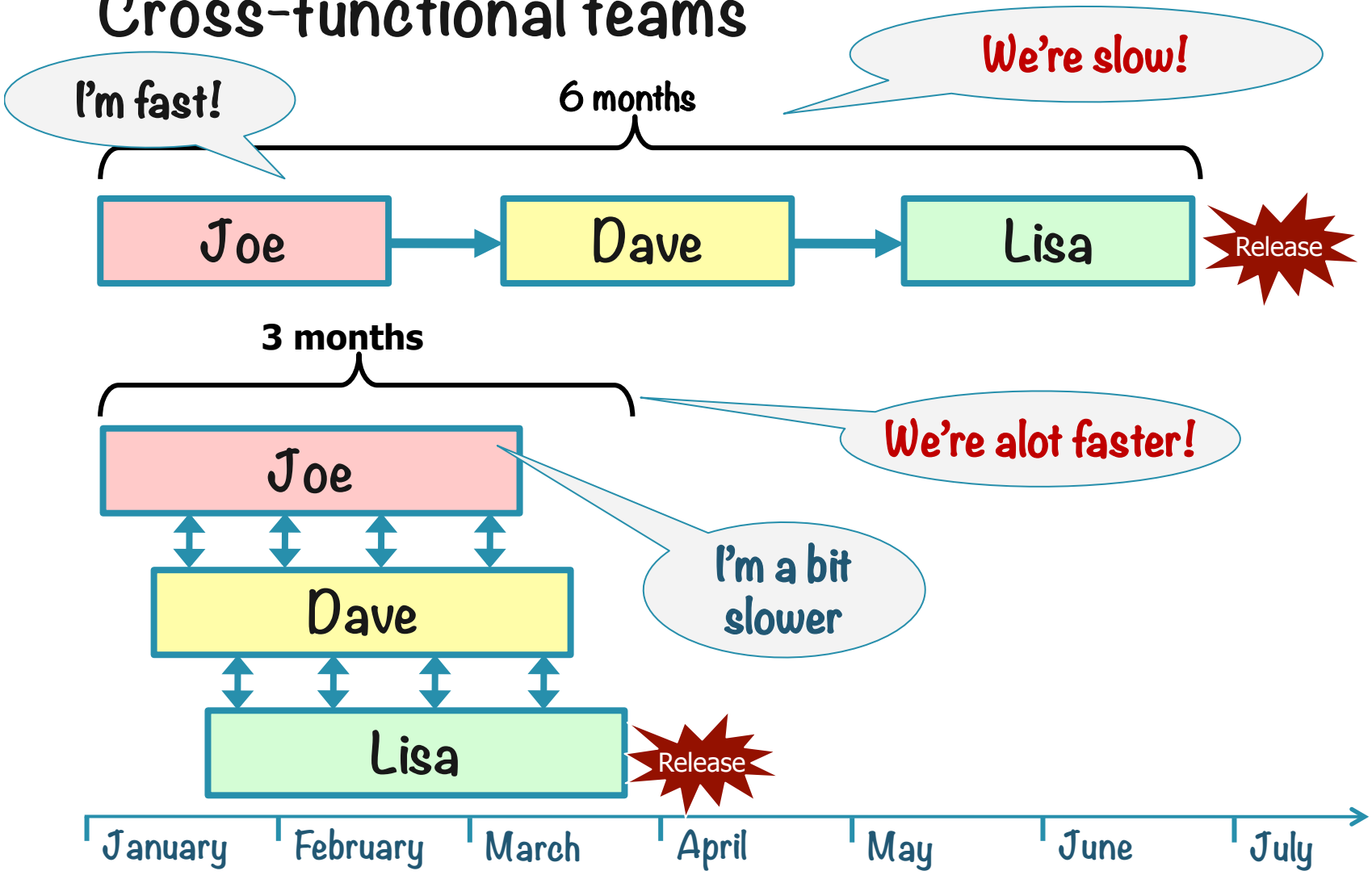
7 times faster

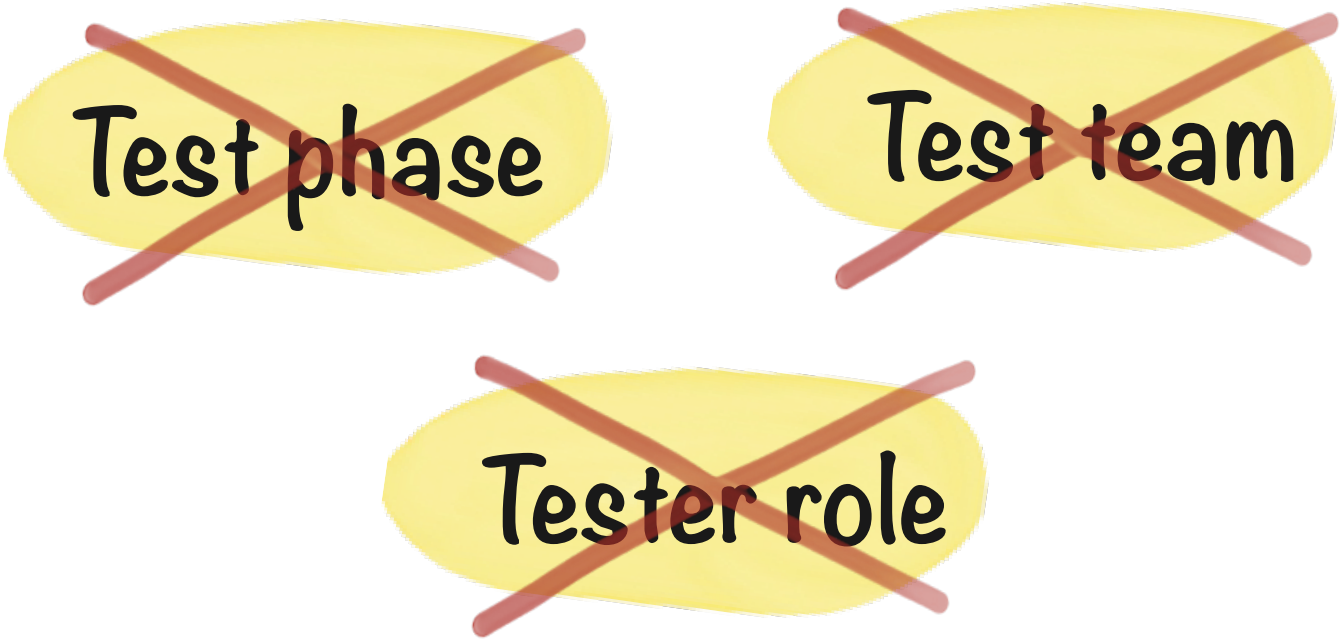
Better games

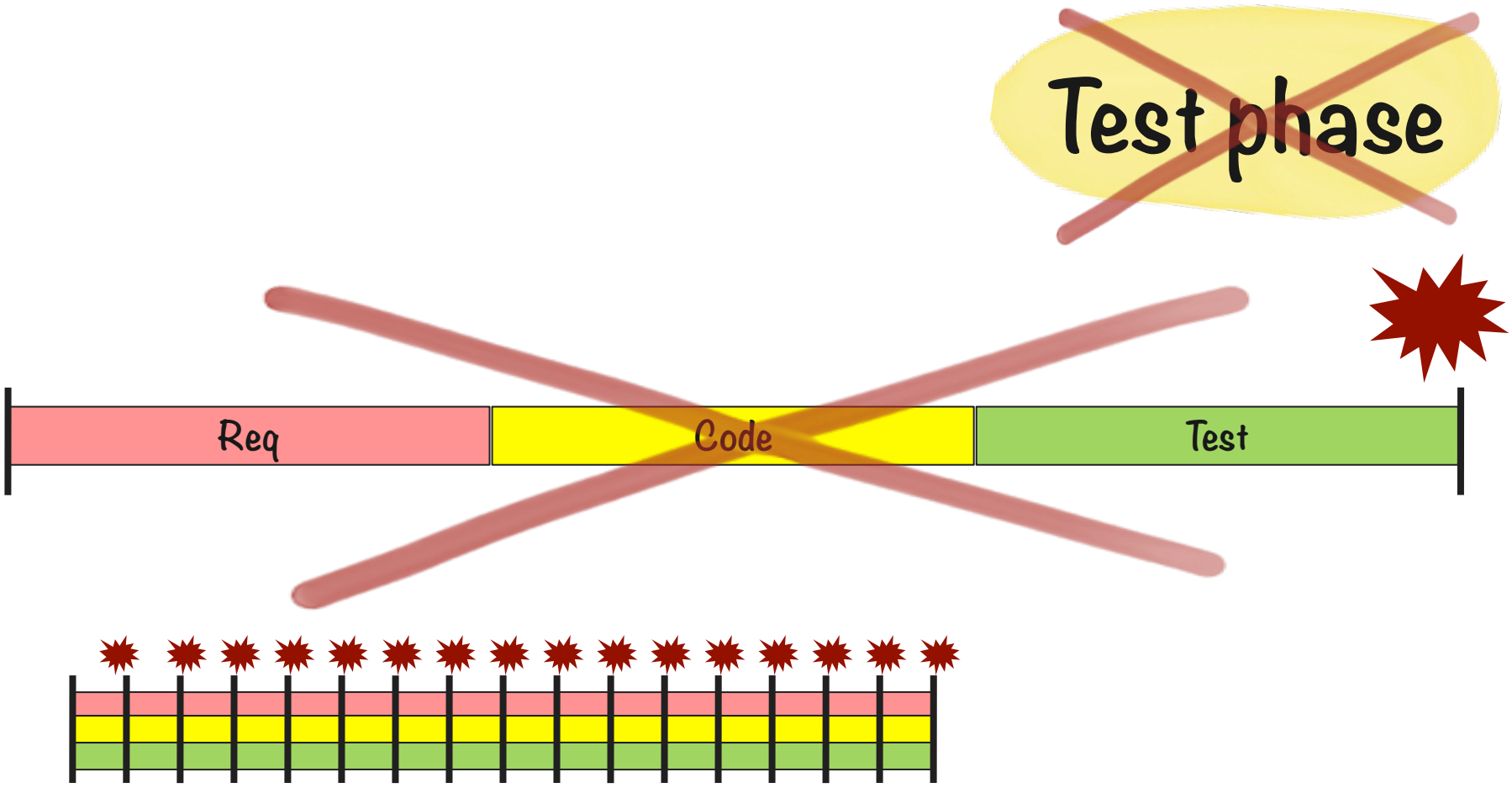
Less Planning

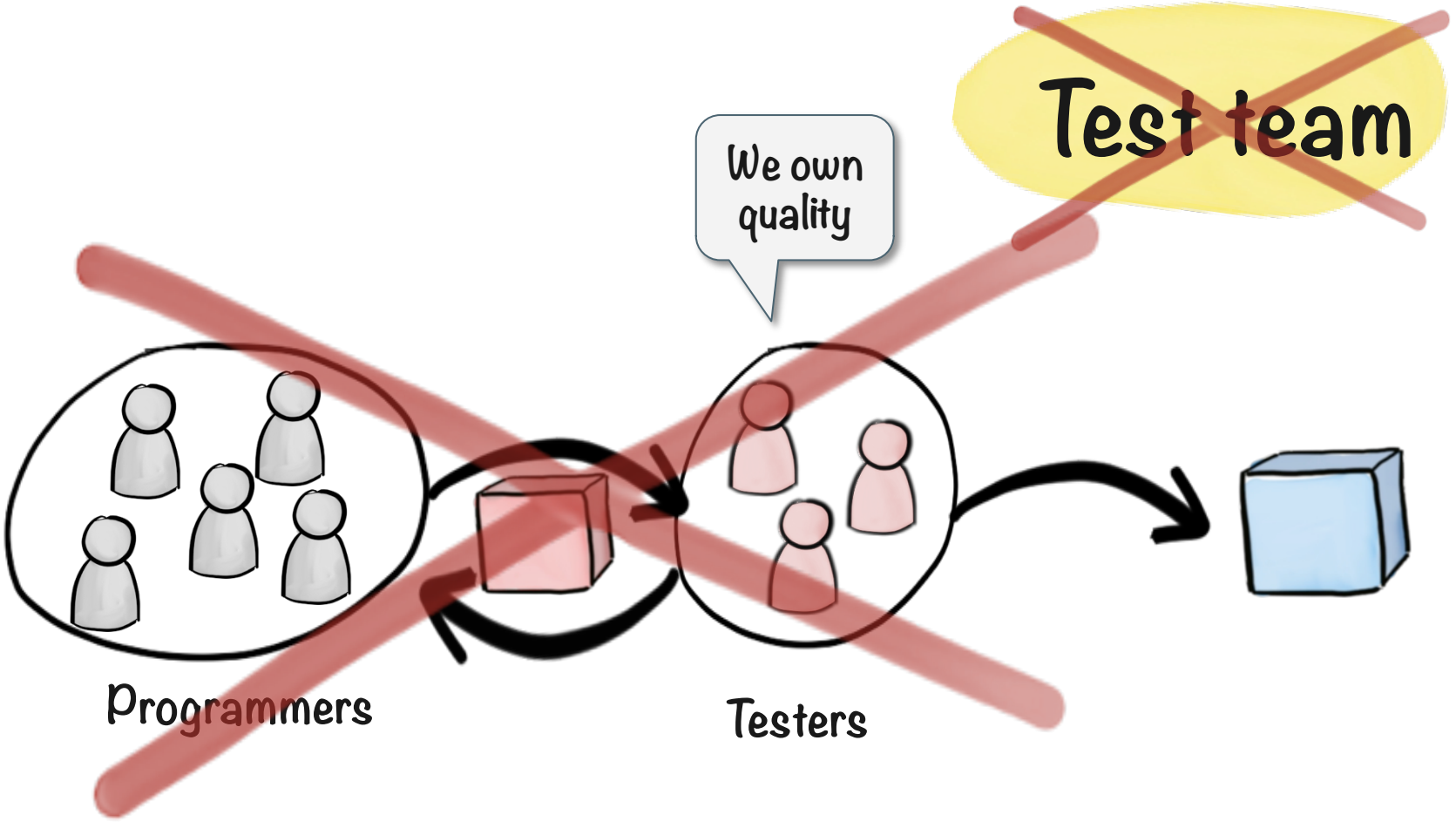
More fun

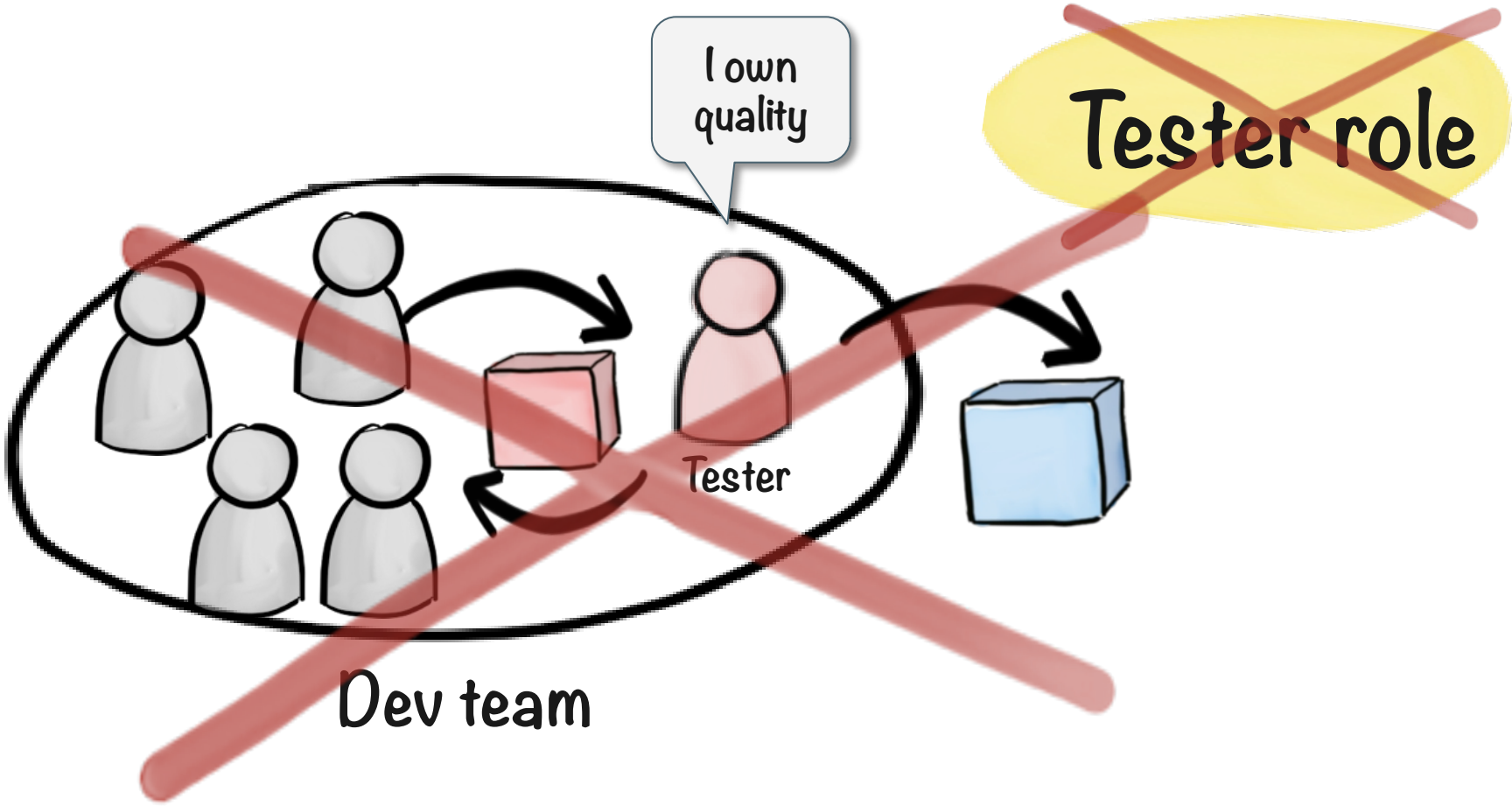
Cross-functional teams



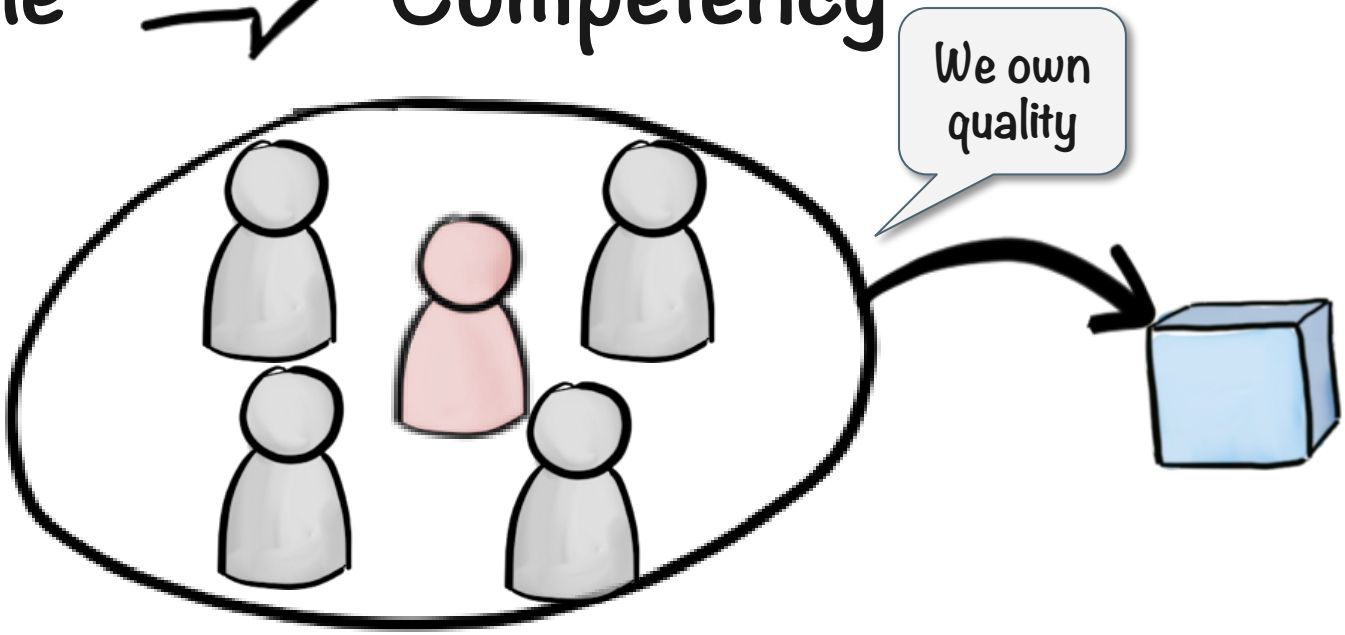

Test phase
Test team
Tester role


Test phase
Req
Code
Test


Test team
We own quality
Programmers
Testers

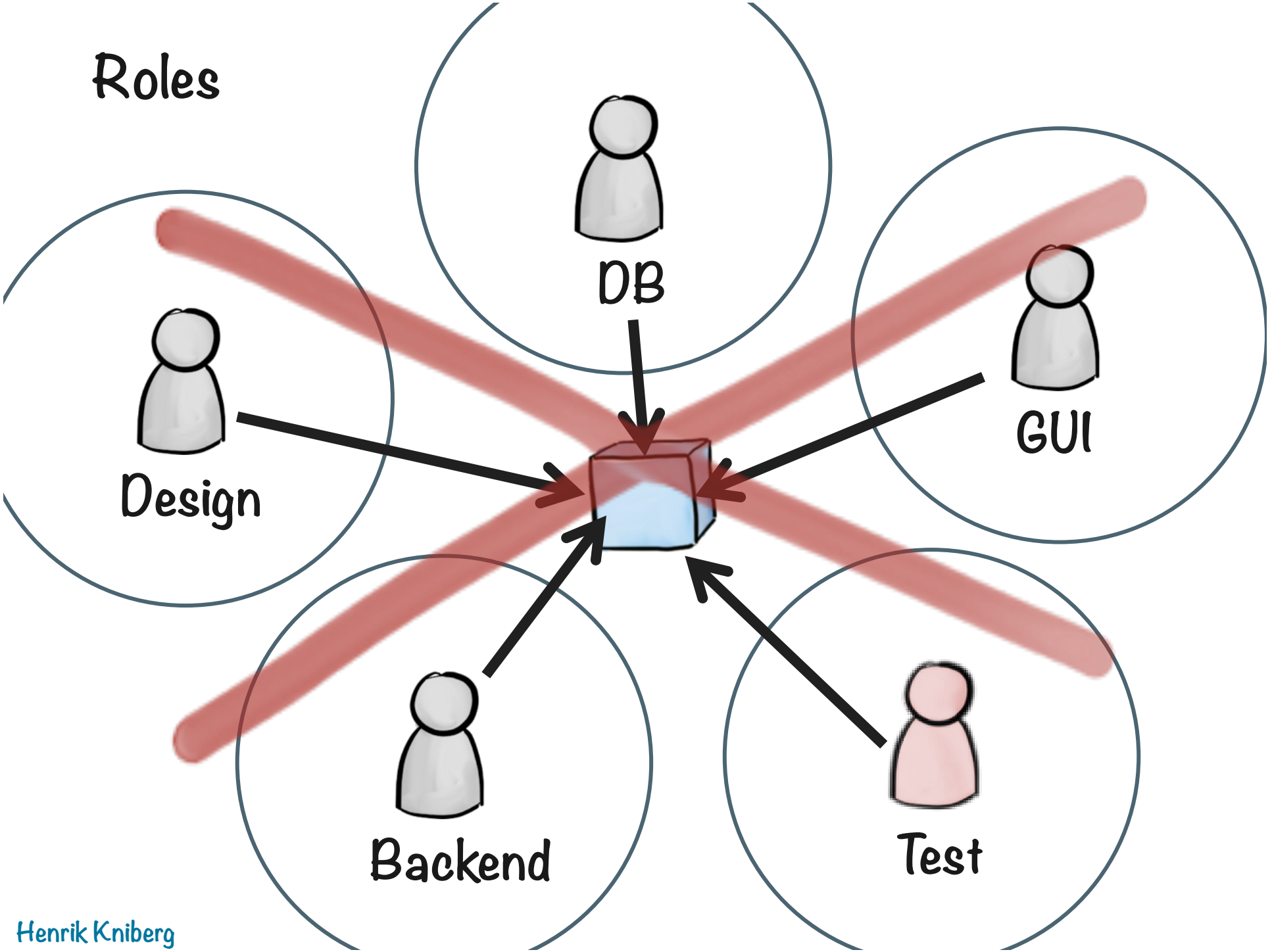

I own quality
Tester role
Tester
Dev team

Role ⇒ Competency

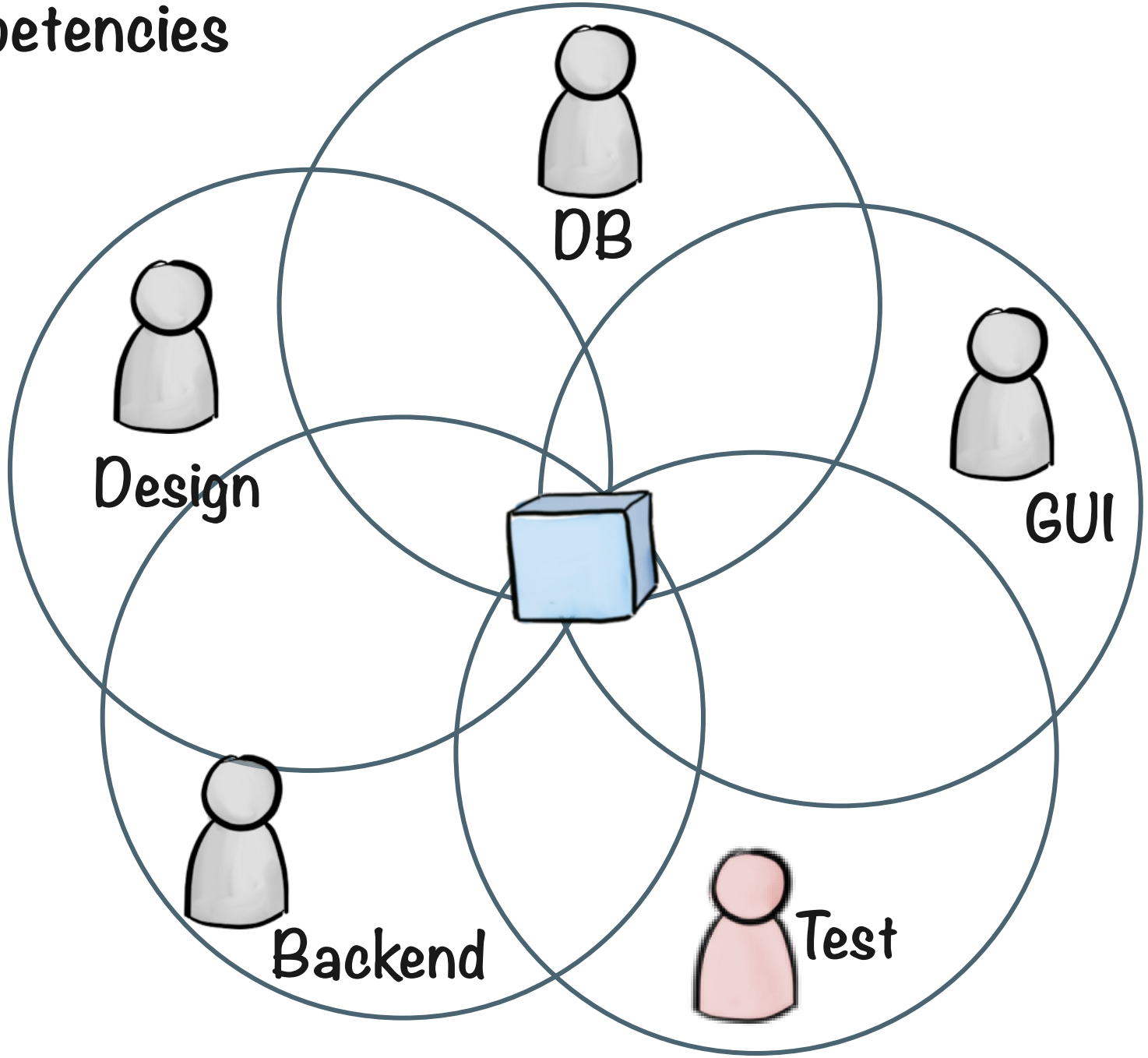


Cross-functional development team

QA = ~~Quality Assurance~~
= Quality Assistance


Roles
DB
Design
GUI
Backend
Test
Henrik Kniberg

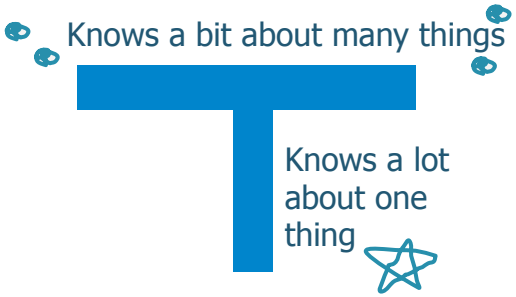
Competencies

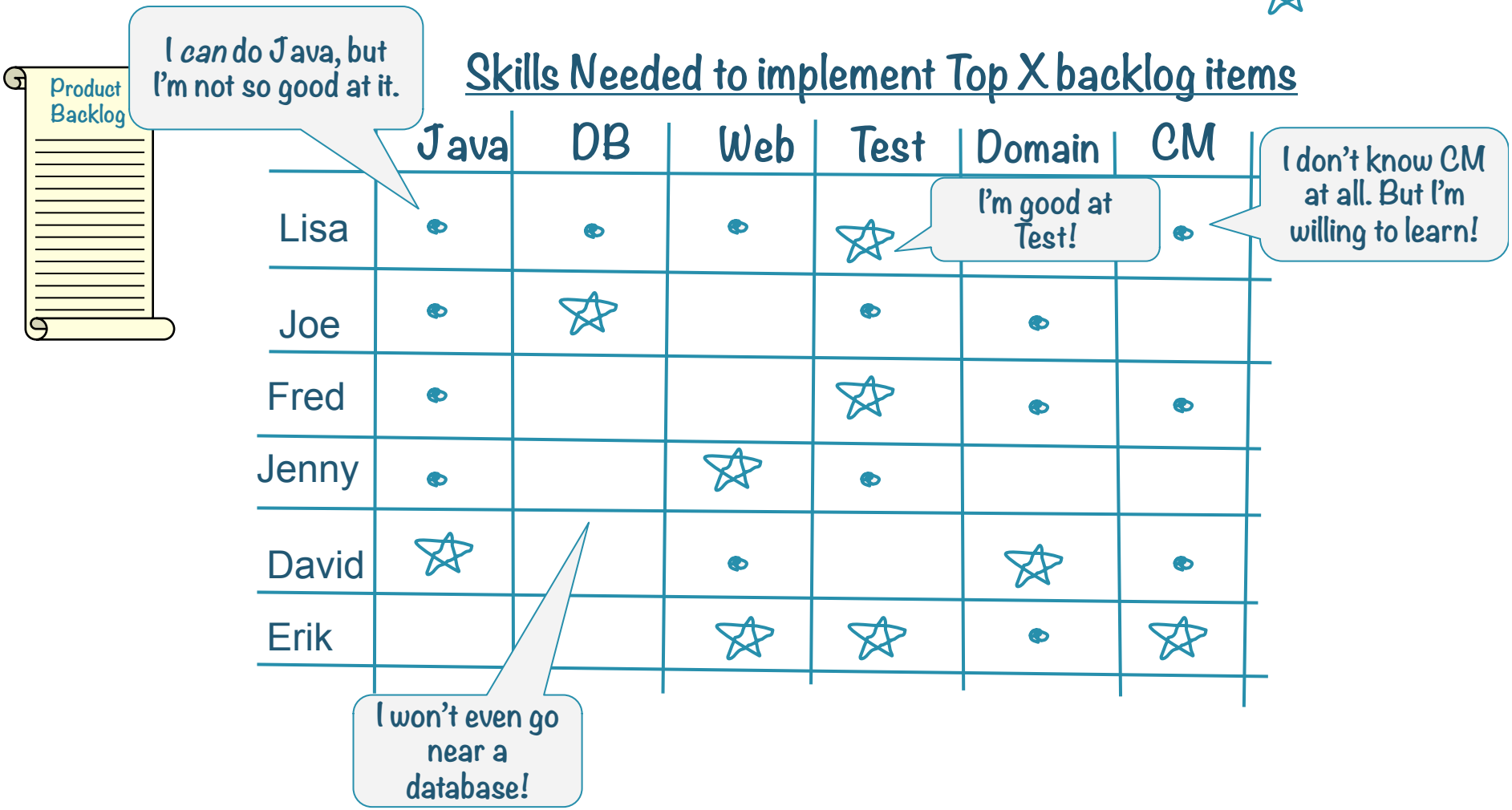


Henrik Kniberg

Cross functional team

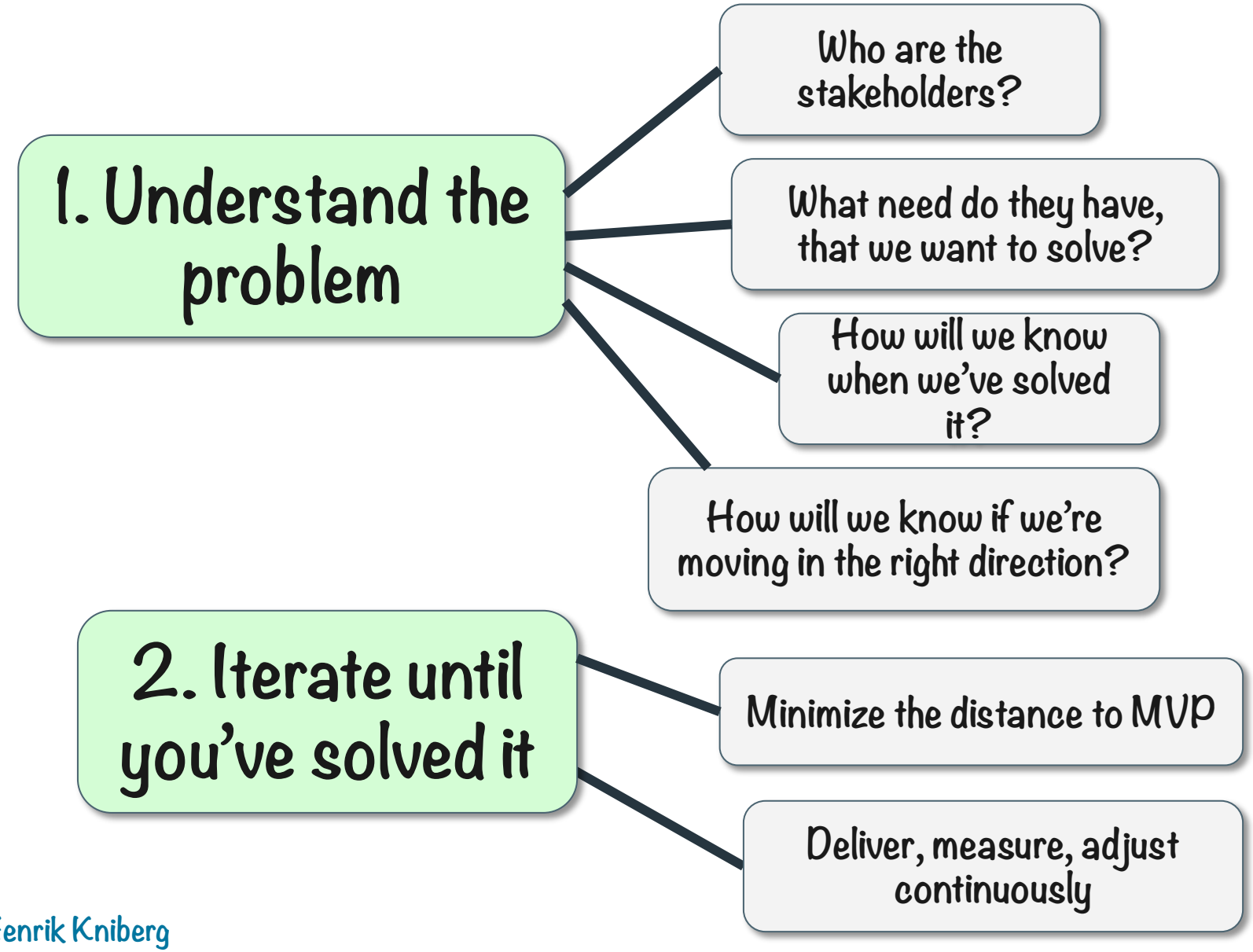
Doesn't mean everyone has to know everything





Henrik Kniberg

How do you ~~know~~ ensure that your product works?



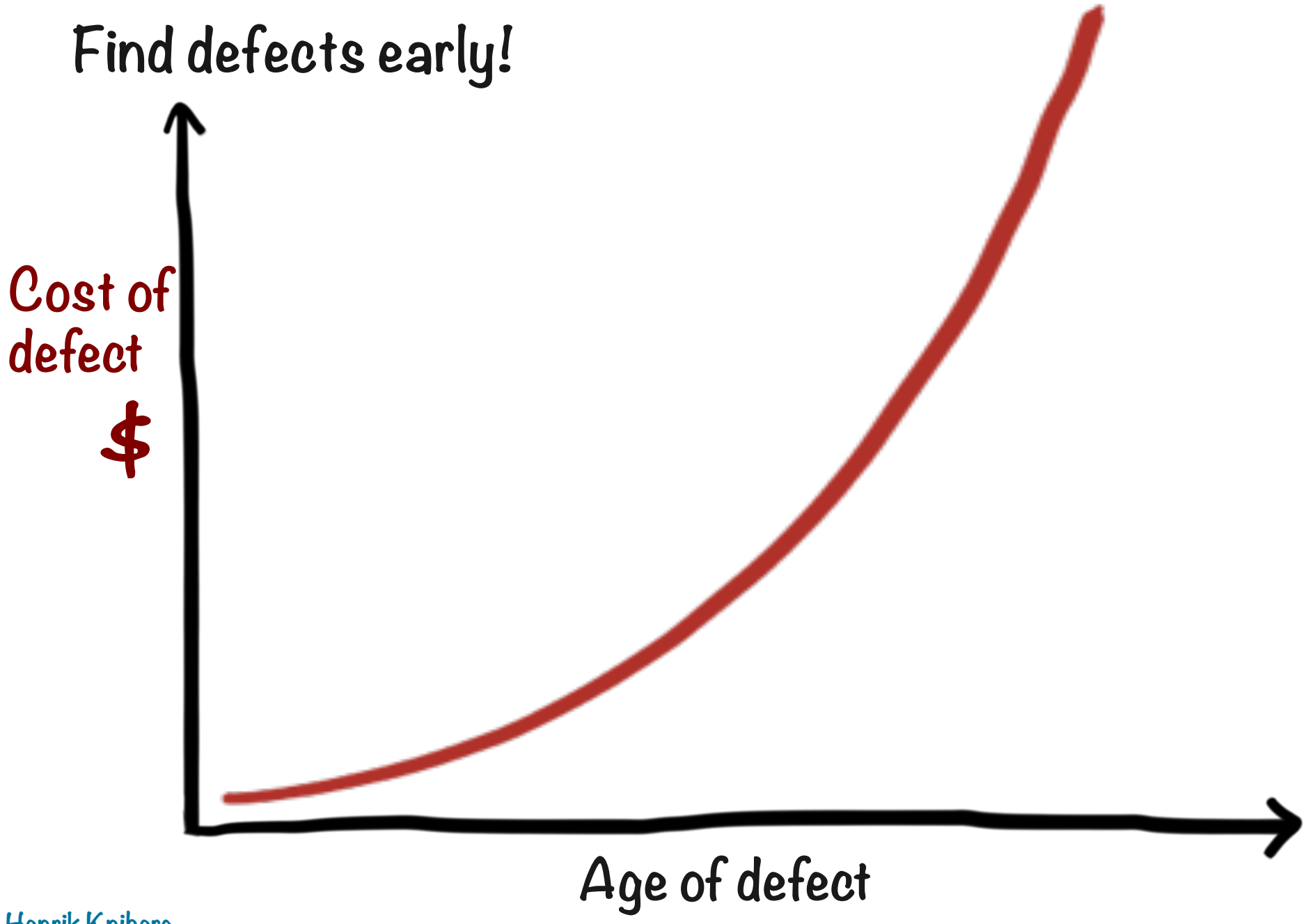
Henrik Kniberg

Typical activities

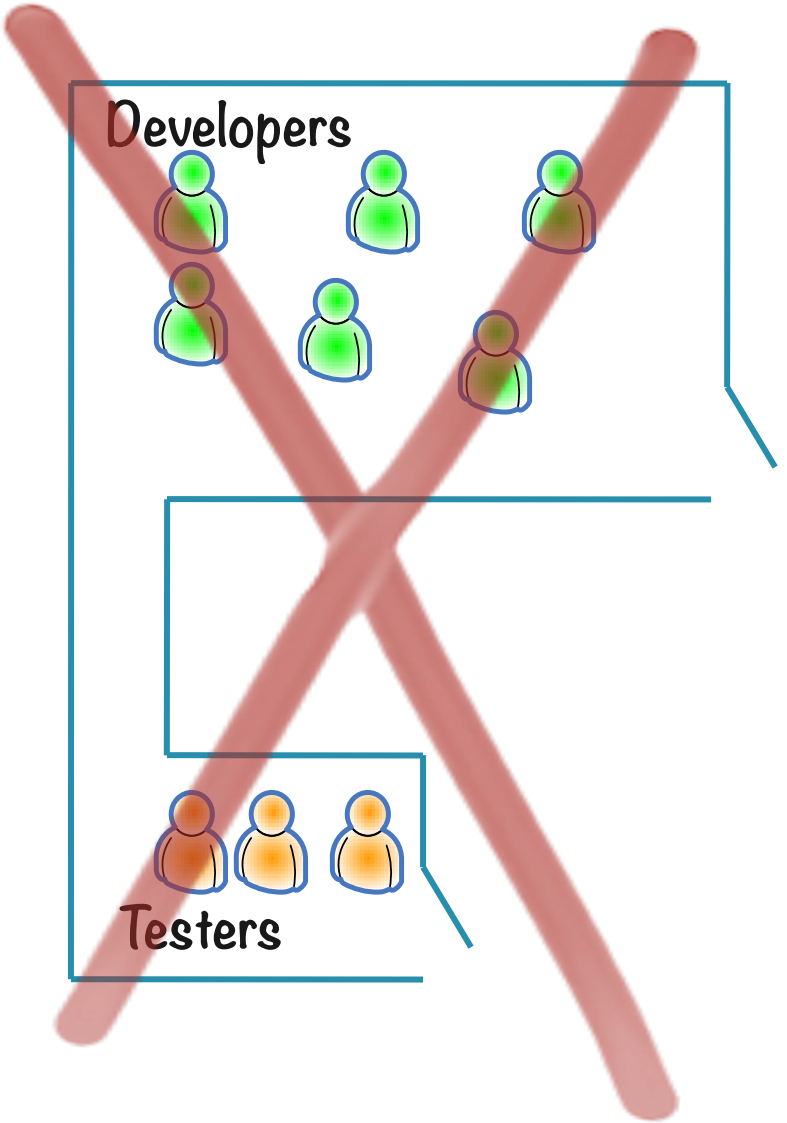
Make sure backlog items are testable & valuable

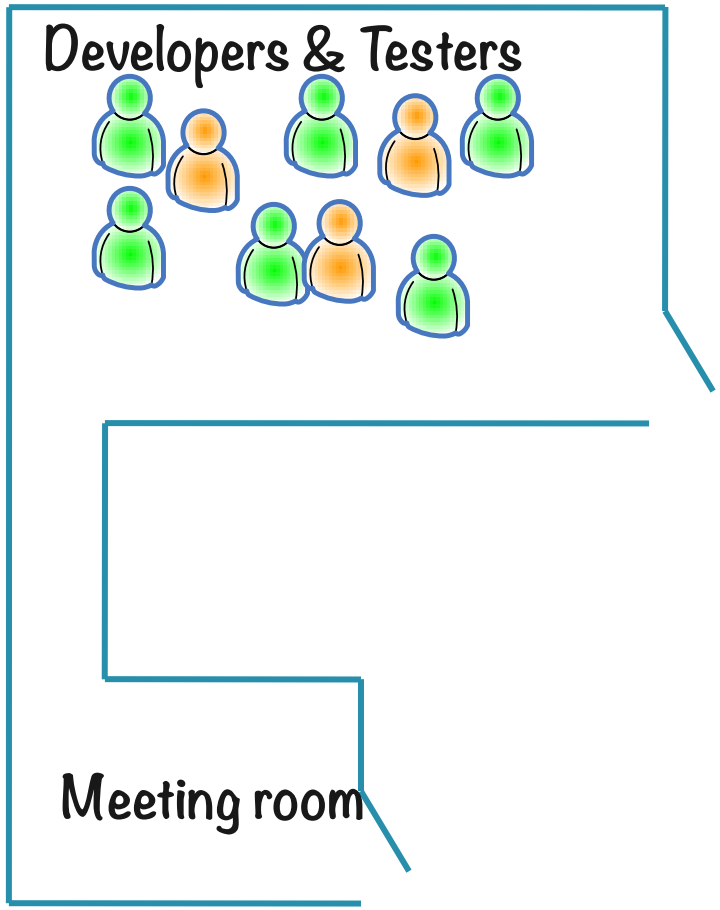
As a buyer
I want to save my shopping cart
so that I can continue shopping later

How to demo:
1) Enter store
2) Put a book in shopping cart
3) Press "save cart"
4) Leave store, and enter it again
5) Check that the book is in my cart

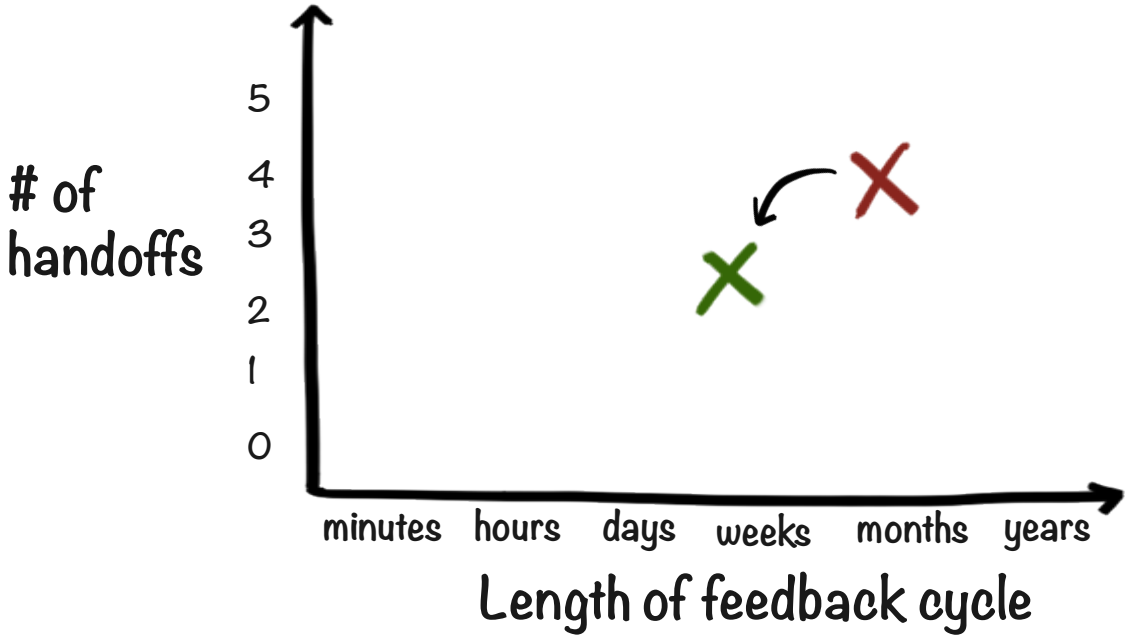

Find defects early!
Cost of defect
$
Age of defect
Henrik Kniberg

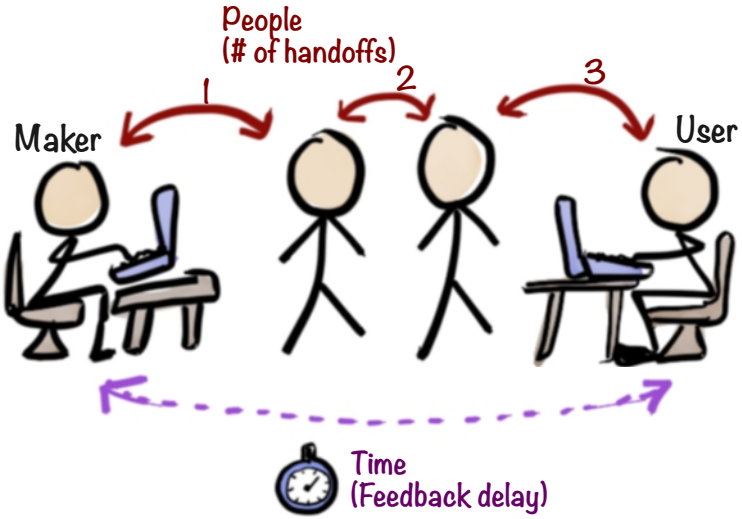
Sit with the developers



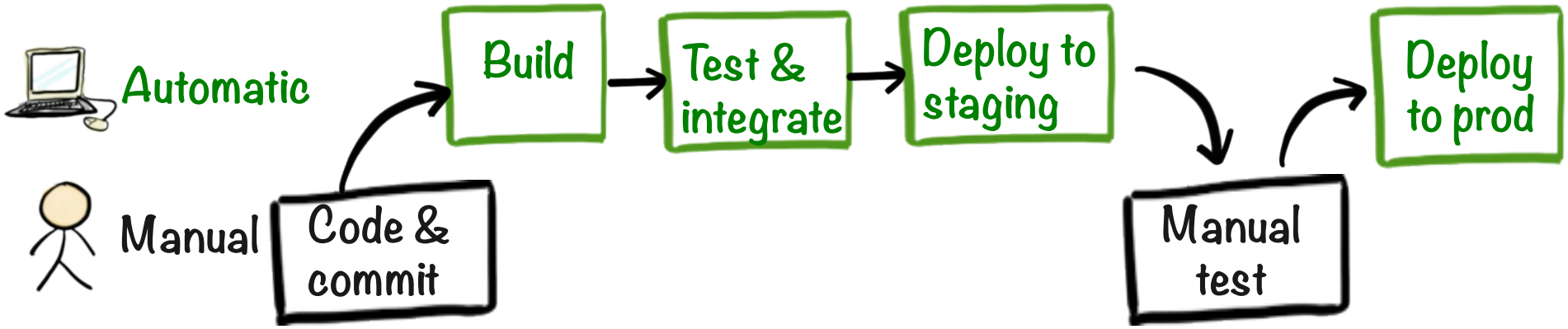


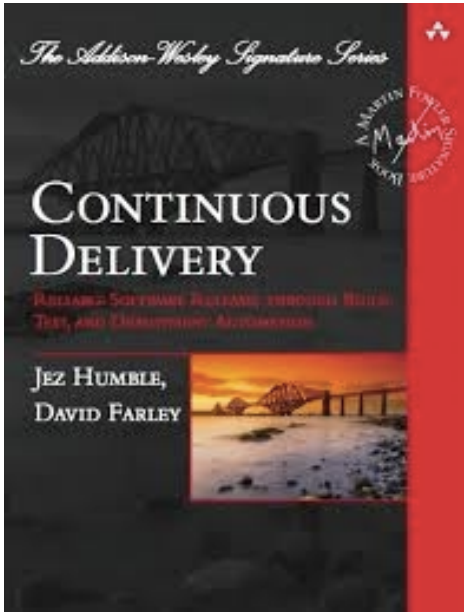
Shorten the feedback loop





Push for Continuous Delivery





Create a shared vocabulary

Unit Test?

Integration test?

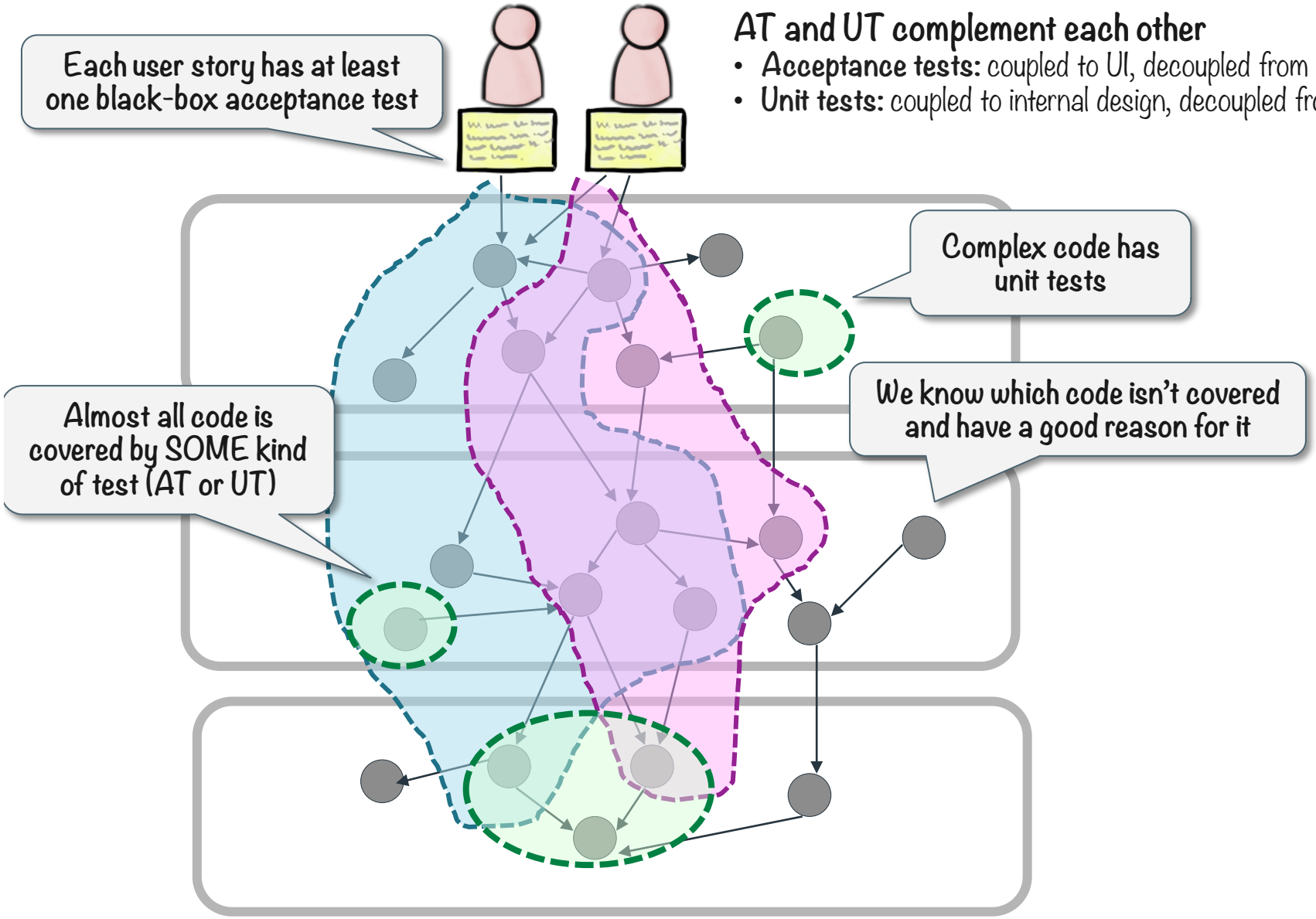
Quality?

Acceptance test?

Technical debt?

Henrik Kniberg

Set working agreements for test automation



AT and UT complement each other

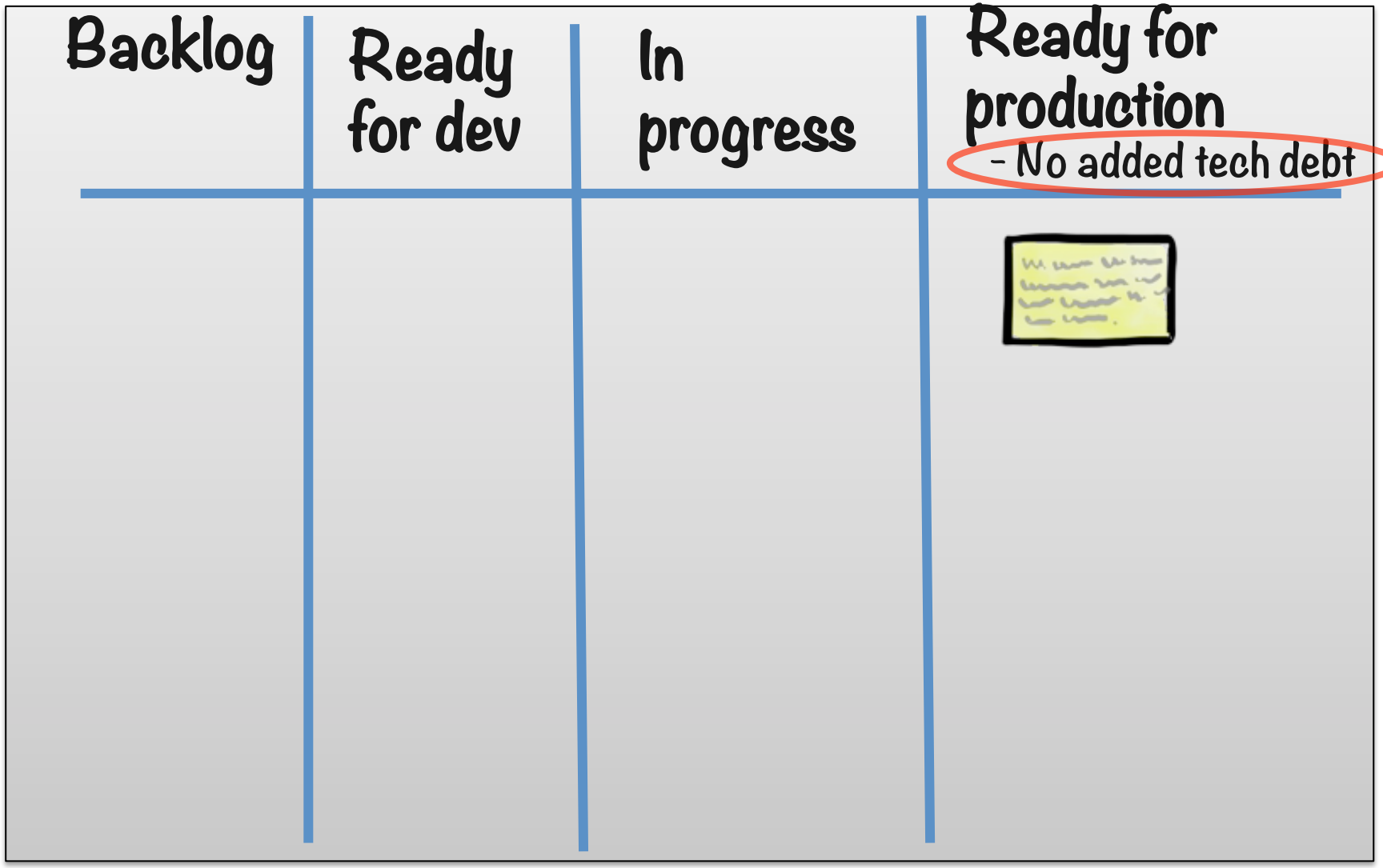
- **Acceptance tests:** coupled to UI, decoupled from internal design
- **Unit tests:** coupled to internal design, decoupled from UI

Henrik Kniberg

Do & teach exploratory testing

Henrik Kniberg

Done includes "no added technical debt"



Henrik Kniberg

Example: Test automation backlog

Step 1: Decide what needs to be tested

- Change skin
- Security alert
- Transaction history
- Block account
- Add new user
- Sort query results
- Deposit cash
- Validate transfer

Step 2: Classify each test

Pay every time

Pay once

Test case	Risk	Manual Test Cost	Automation Cost
Change skin	low	0.5 hrs	high
Security alert	high	1 hrs	high
Transaction history	med	3 hrs	low
Block account	high	5 hrs	low
Add new user	low	0.5 hrs	low
Sort query results	med	2 hrs	medium
Deposit cash	high	1.5 hrs	low
Validate transfer	high	3 hrs	medium

Step 3: Sort the list

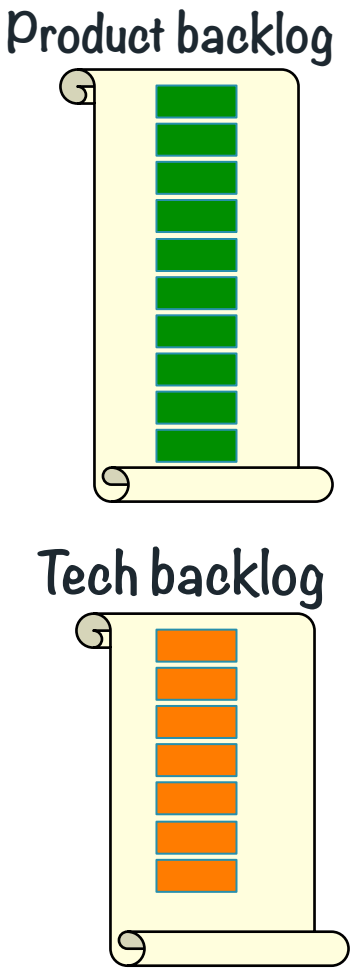
Test case	Risk	Manual Test Cost	Automation Cost
Block account	high	5 hrs	low
Validate transfer	high	3 hrs	medium
Transaction history	med	3 hrs	low
Sort query results	med	2 hrs	medium
Deposit cash	high	1.5 hrs	low
Security alert	high	1 hr	high
Add new user	low	0.5 hrs	low
Change skin	low	0.5 hrs	high

Automate first!

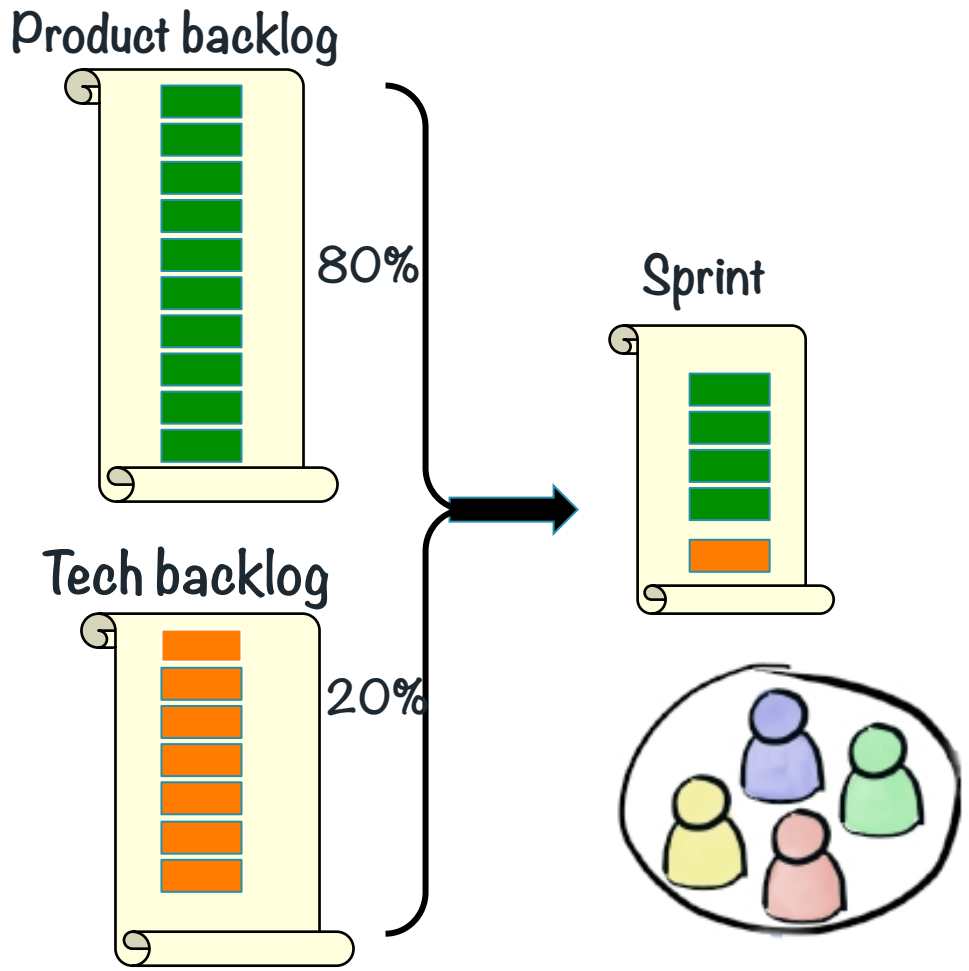
Automate later

Don't bother automating

Example: Tech backlog



Reserve X% of team capacity for the tech backlog

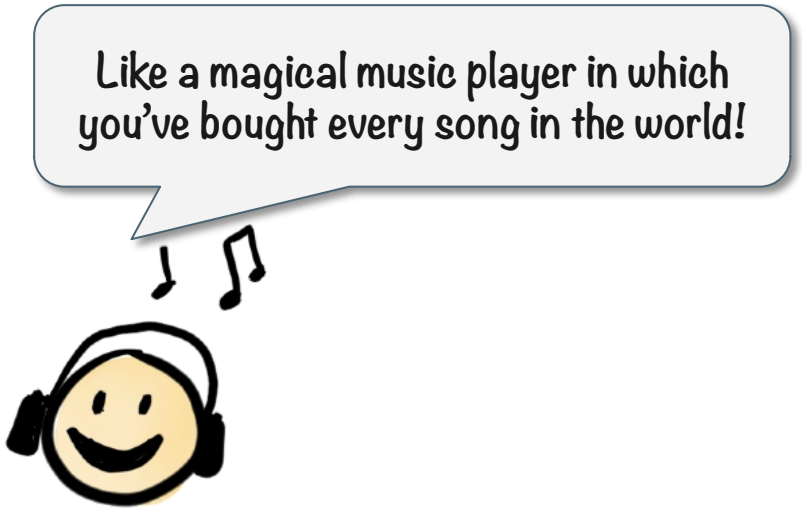


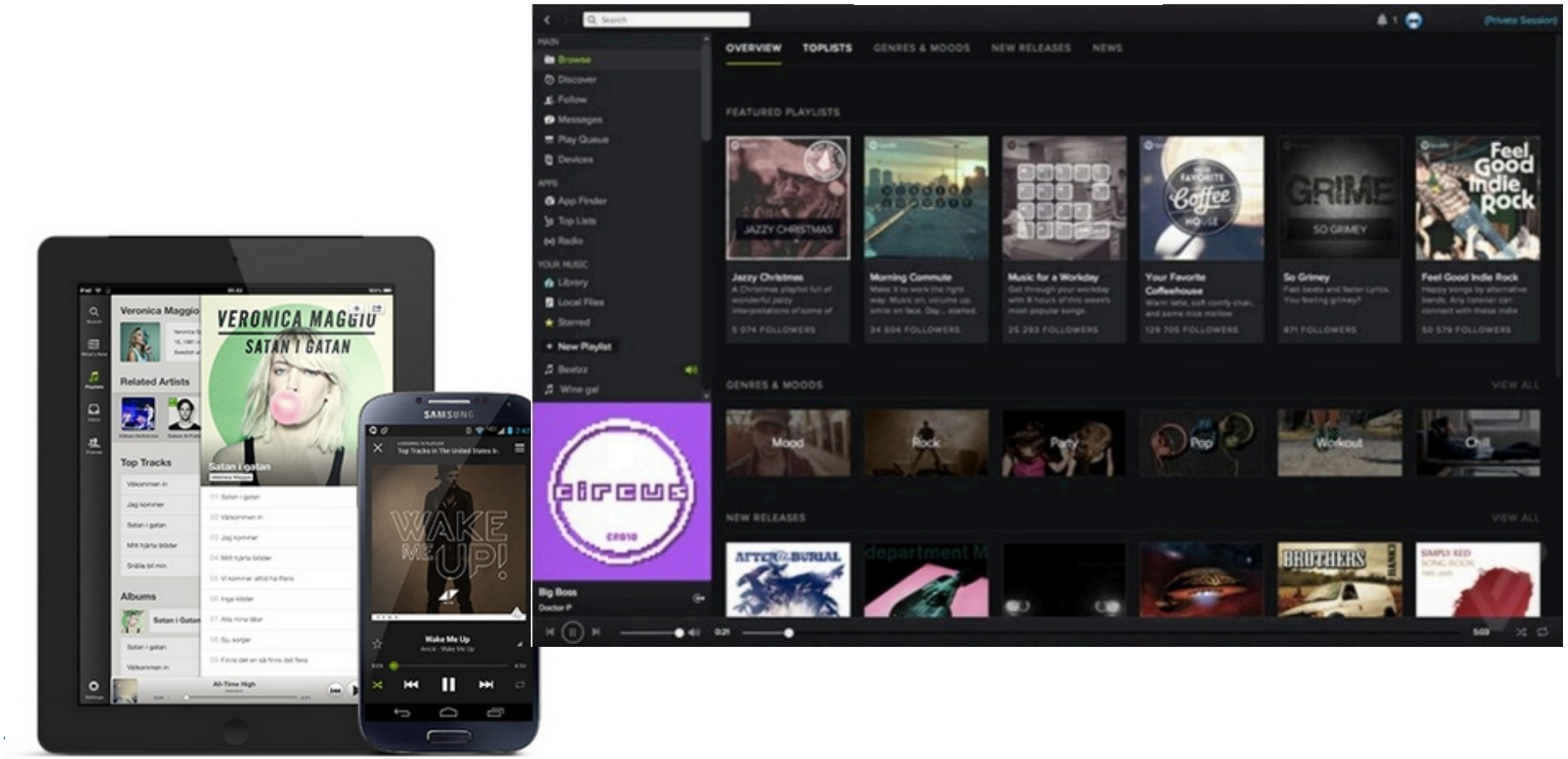
Henrik Kniberg

Example: Spotify

Spotify

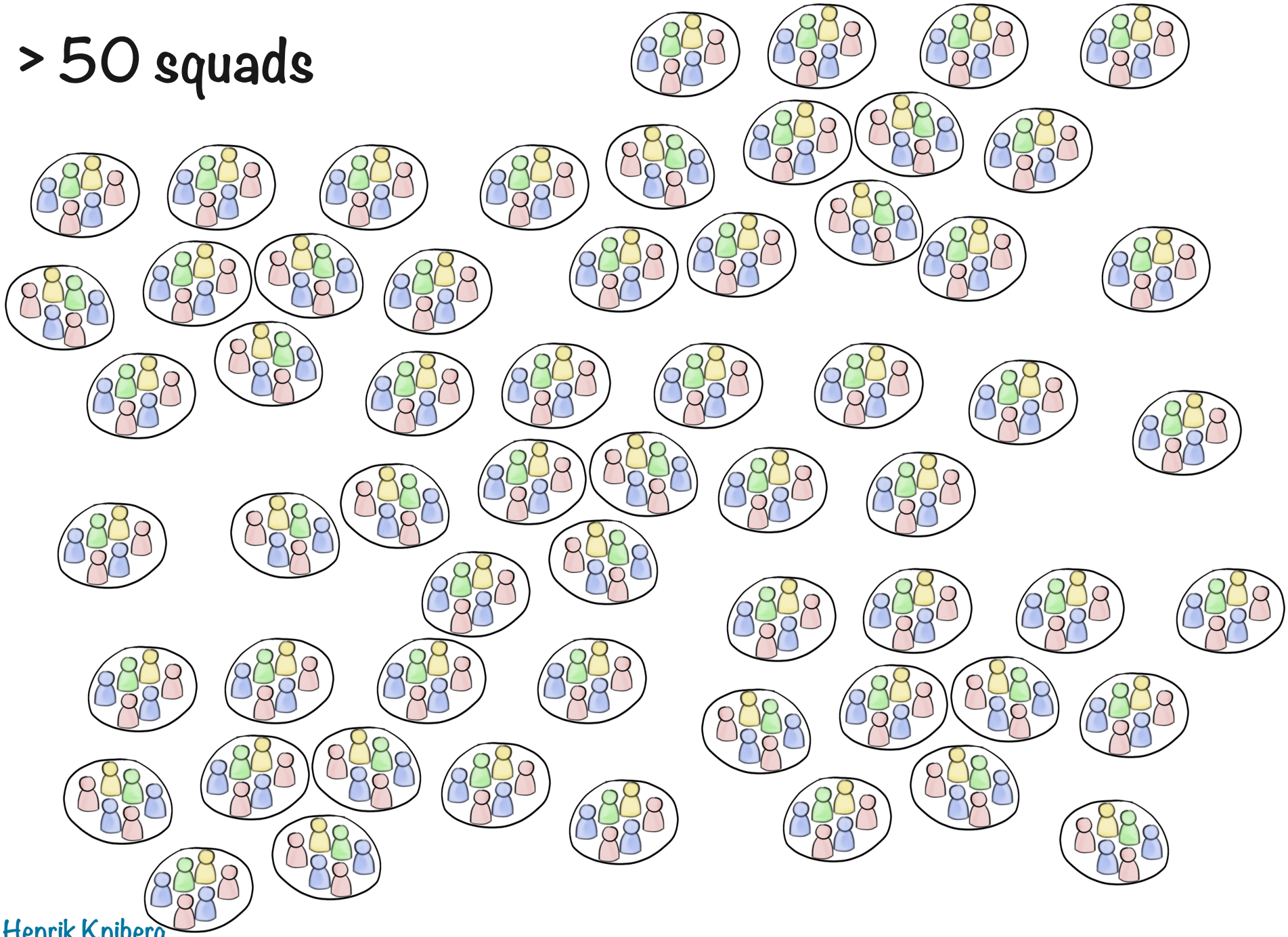
Play Everywhere!





>400 people in tech

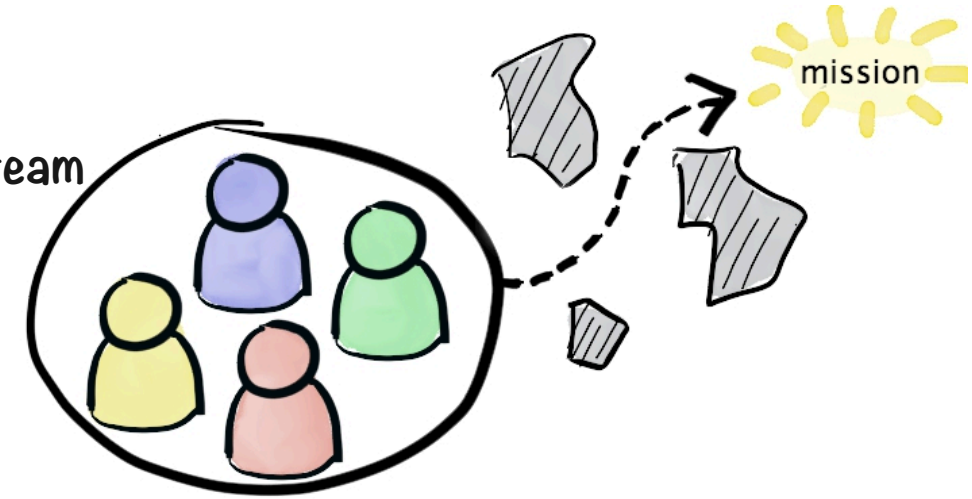
> 50 squads

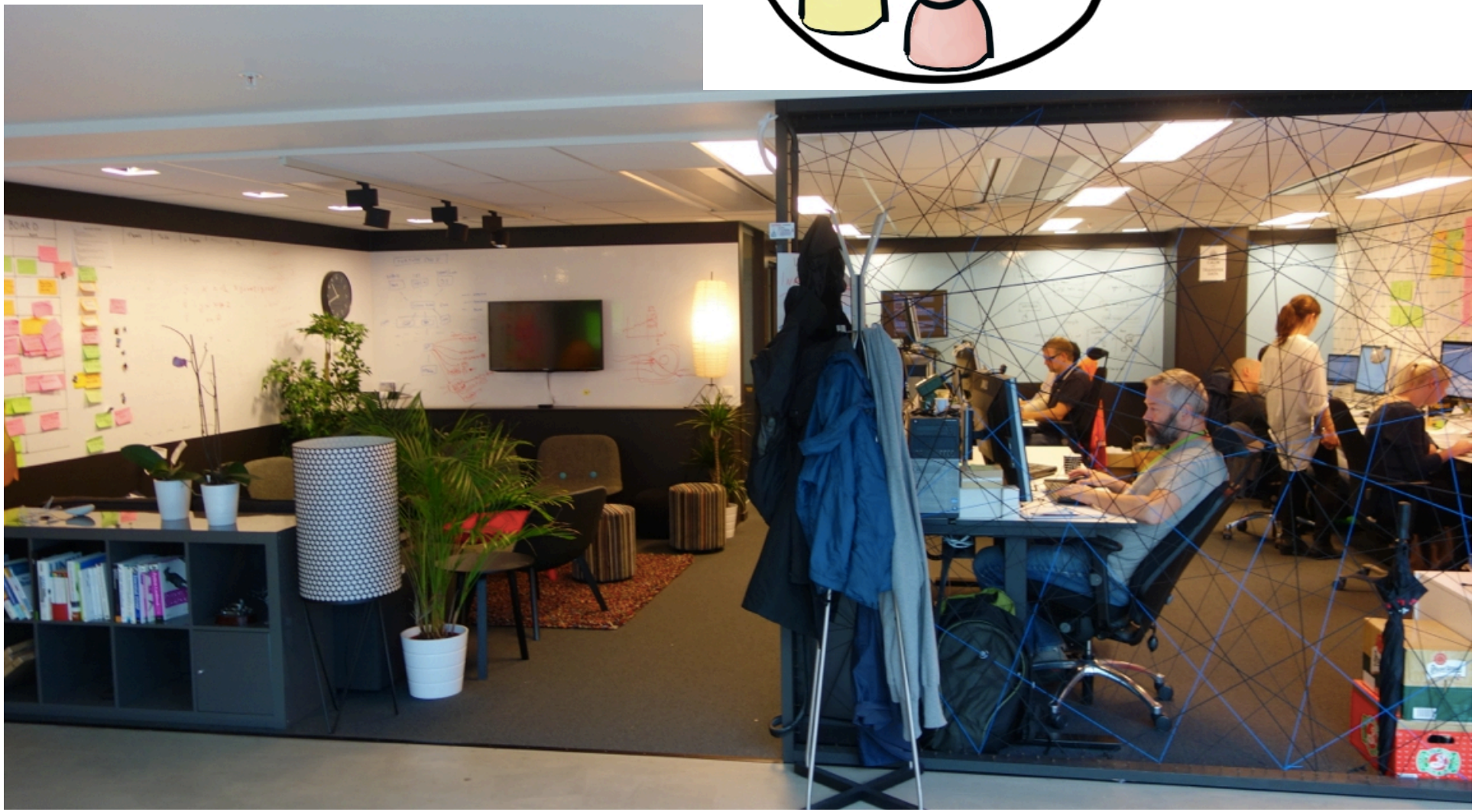


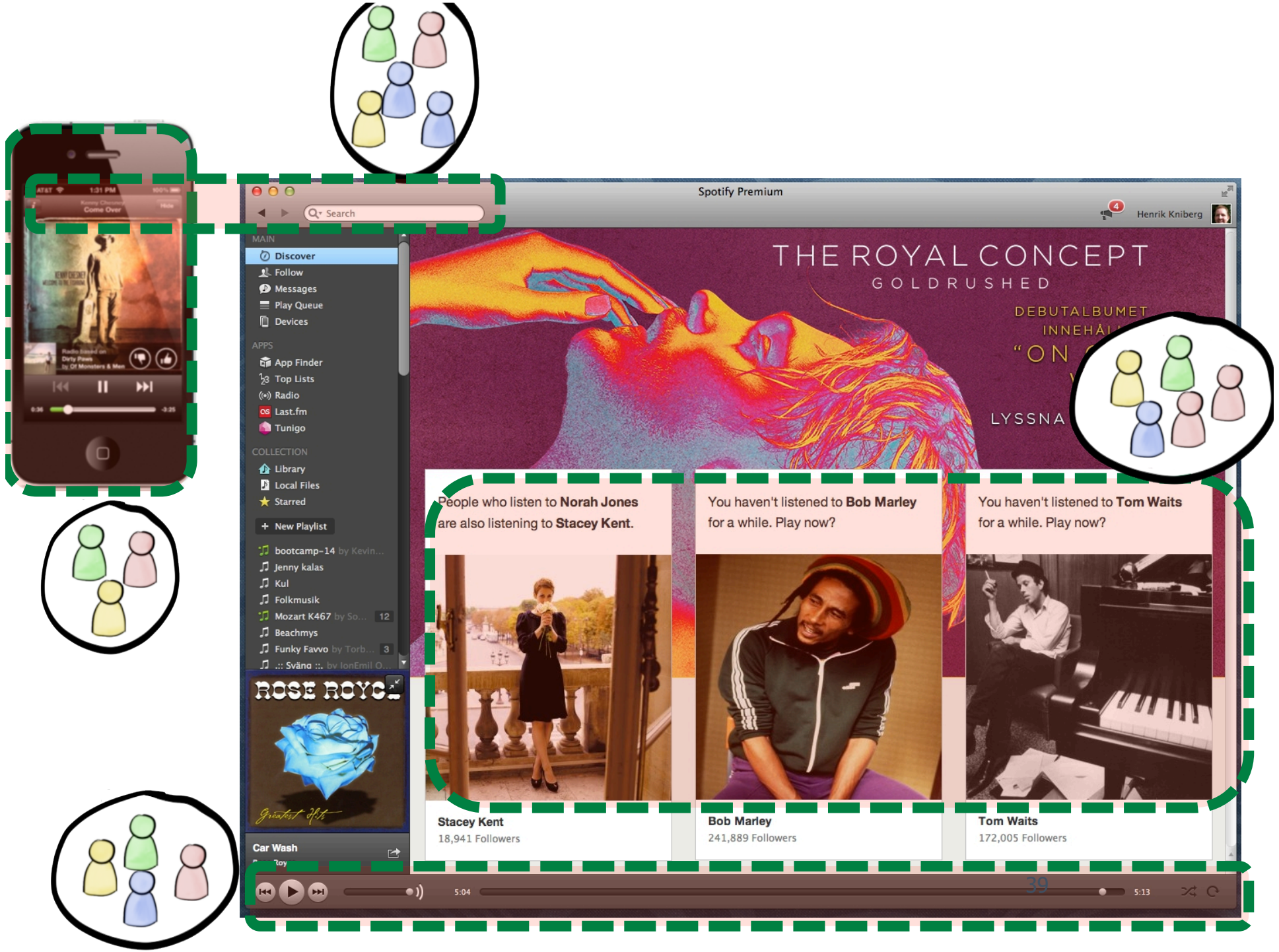
Henrik Kniberg

Autonomous Squad

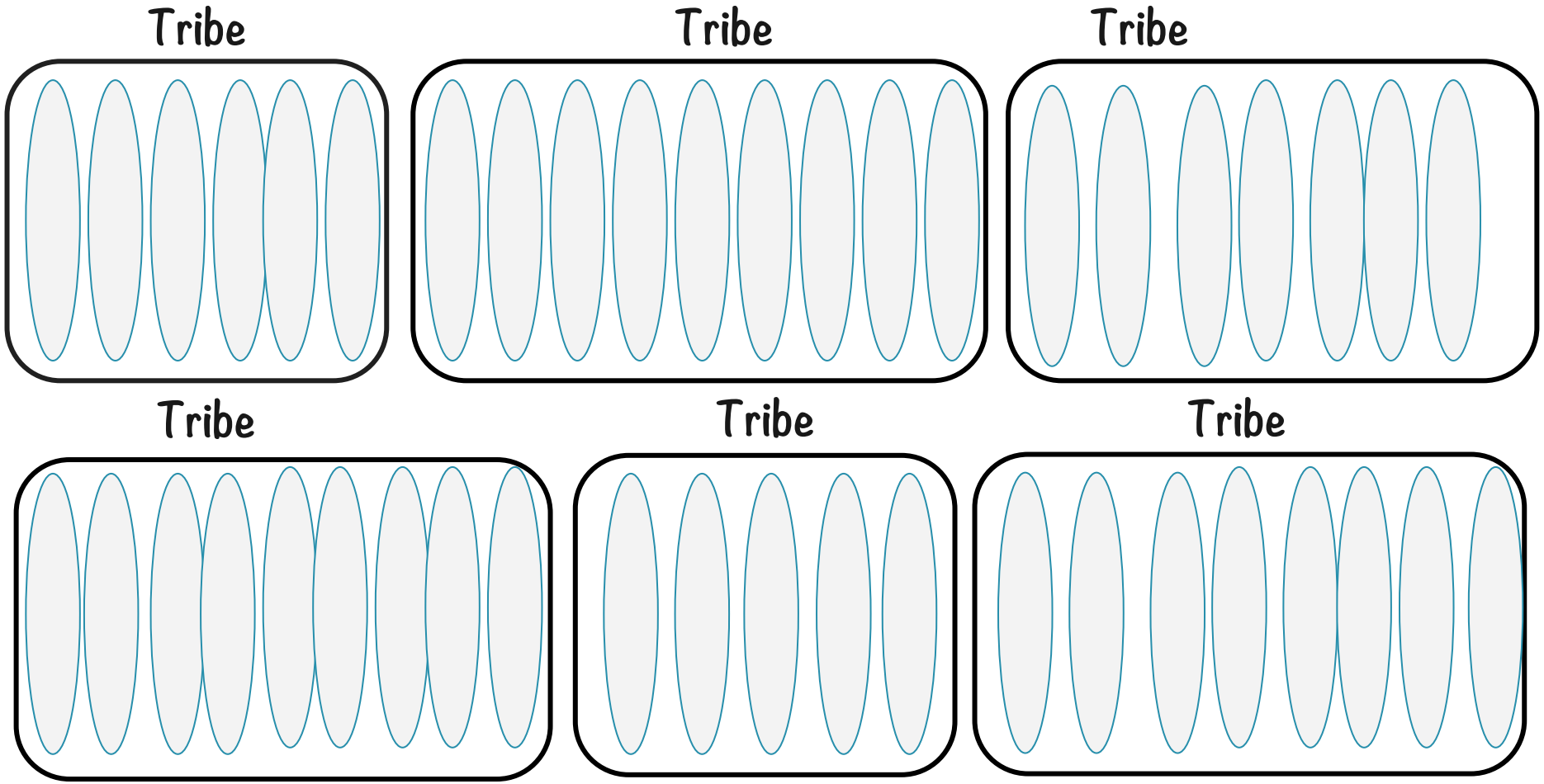
Cross-functional, co-located, self-organizing team






Spotify Premium
Search
Henrik Kniberg
MAIN
Discover
Follow
Messages
Play Queue
Devices
APPS
App Finder
Top Lists
Radio
Last.fm
Tunigo
COLLECTION
Library
Local Files
Starred
New Playlist
bootcamp-14 by Kevin...
Jenny kalas
Kul
Folkmusik
Mozart K467 by So... 12
Beachmys
Funky Favvo by Torb... 3
Car Wash
THE ROYAL CONCEPT
GOLDRUSHED
DEBUTALBUMET
LYSSNA
People who listen to Norah Jones are also listening to Stacey Kent.
Stacey Kent
18,941 Followers
You haven't listened to Bob Marley for a while. Play now?
Bob Marley
241,889 Followers
You haven't listened to Tom Waits for a while. Play now?
Tom Waits
172,005 Followers
5:04
5:13

Squads are grouped into Tribes

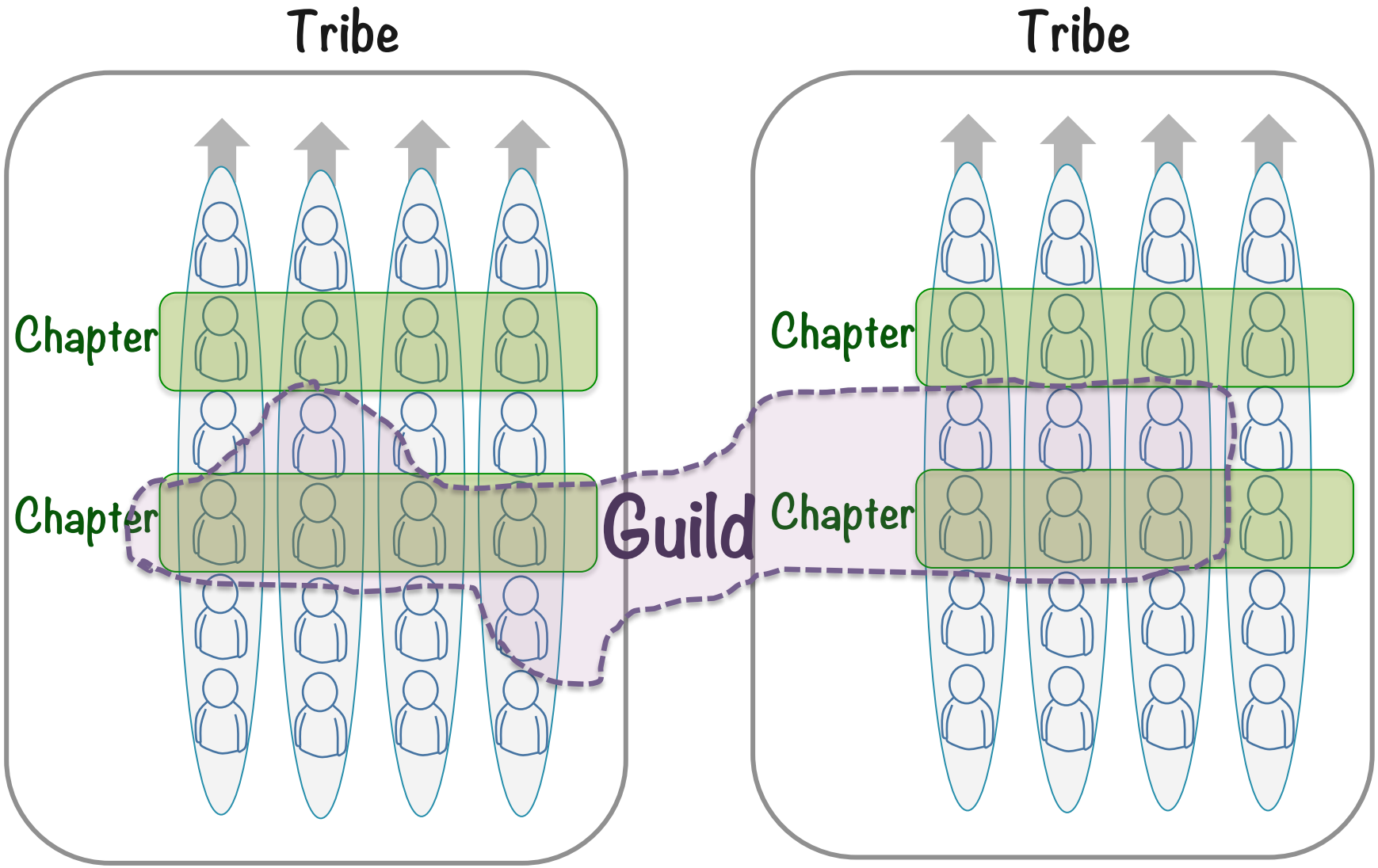


Henrik Kniberg

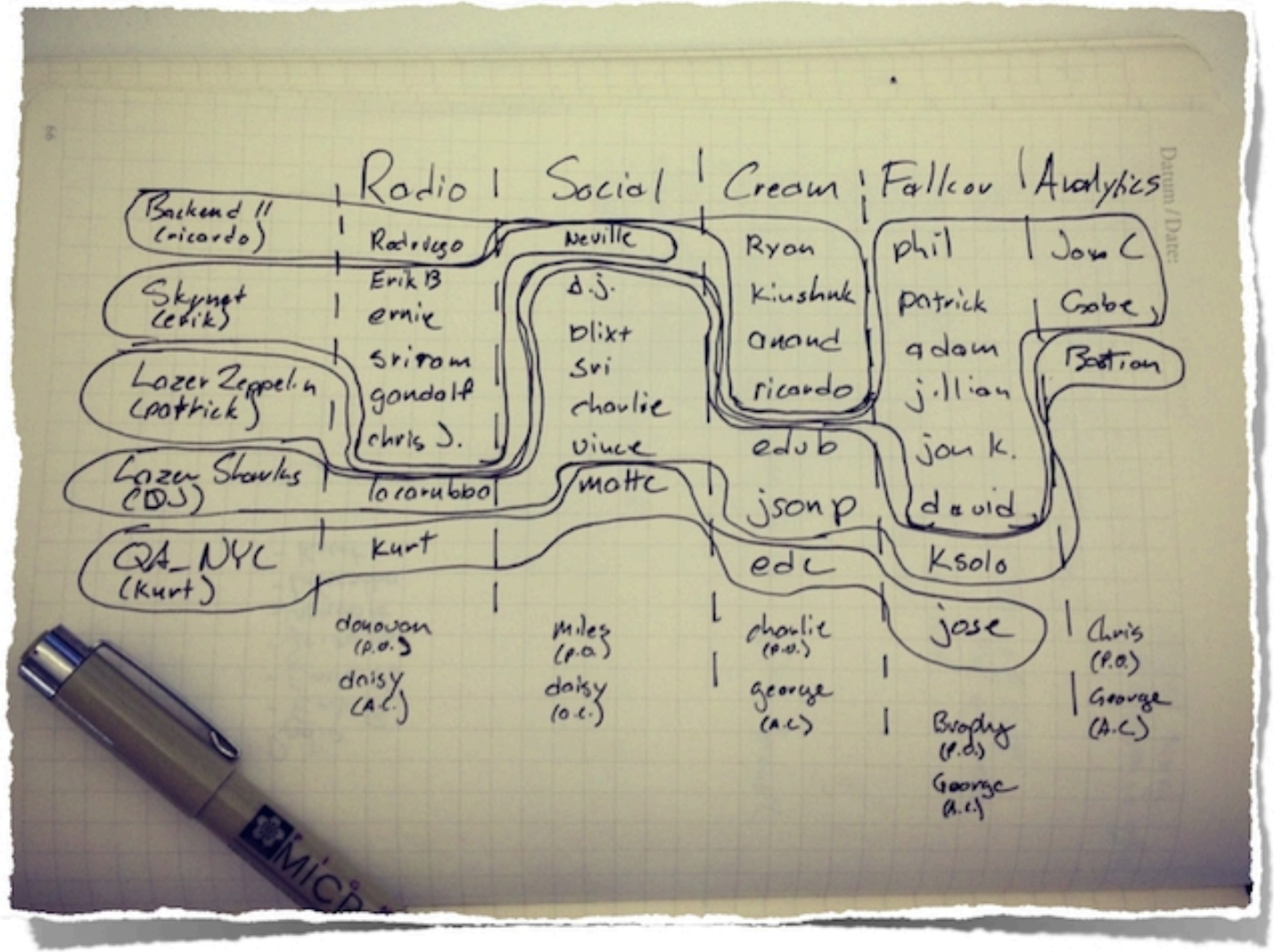
Each Tribe is a lightweight matrix focused on delivery

Vertical = Delivery.

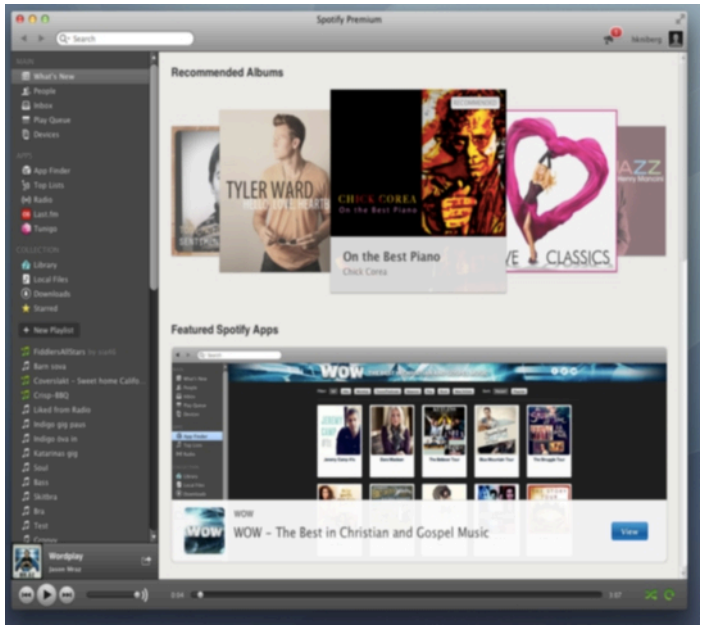
Horizontal = knowledge sharing & personal development

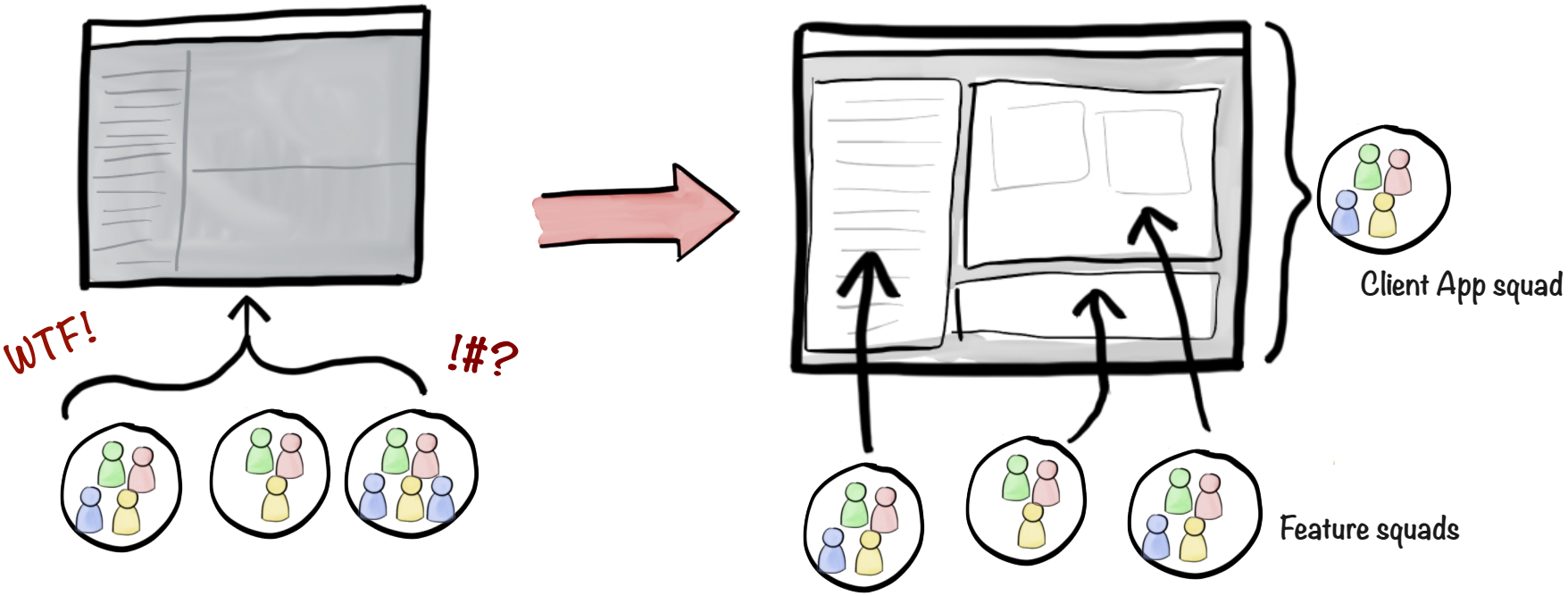


Reality is messy



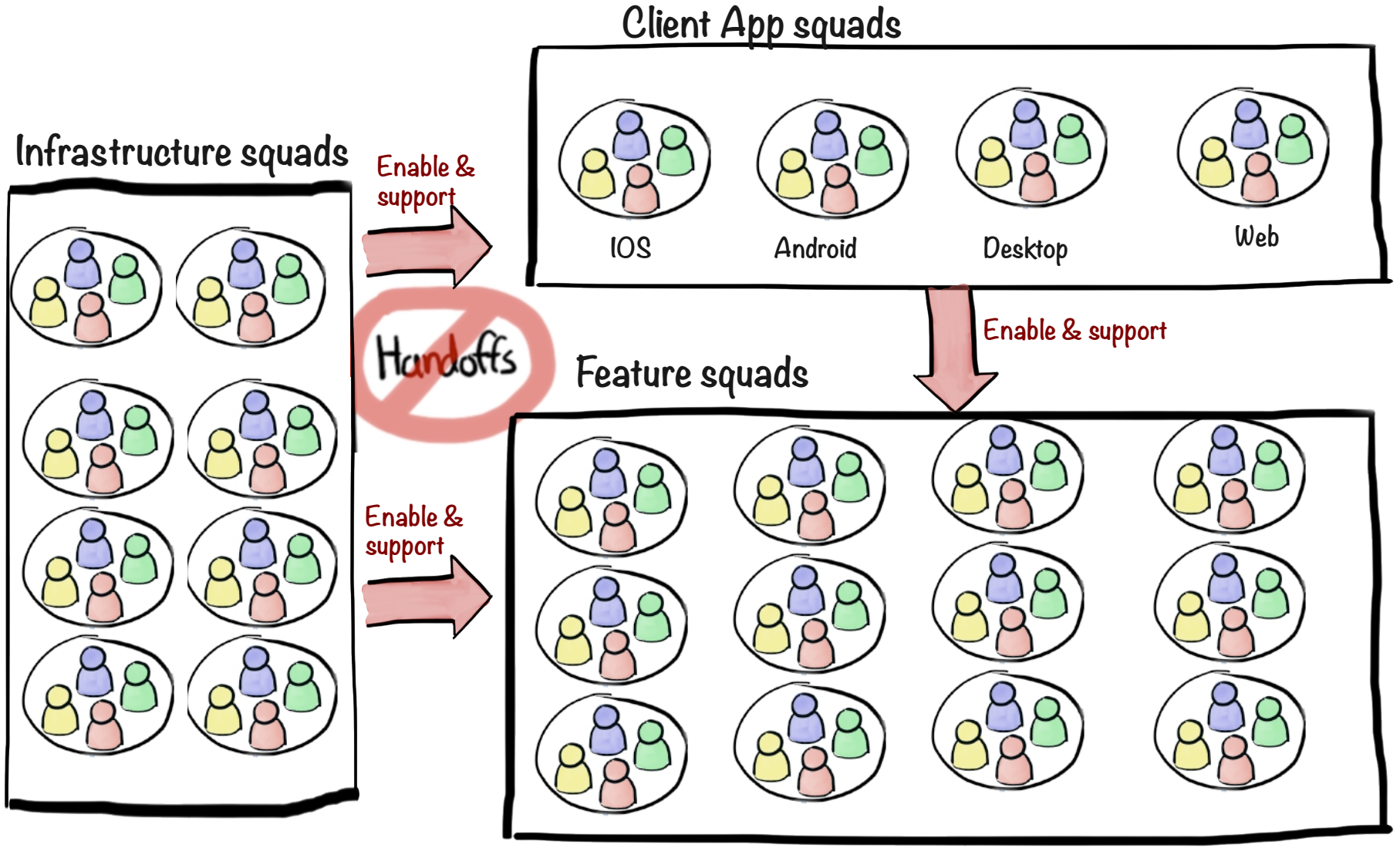
Decoupling to enable frequent releases





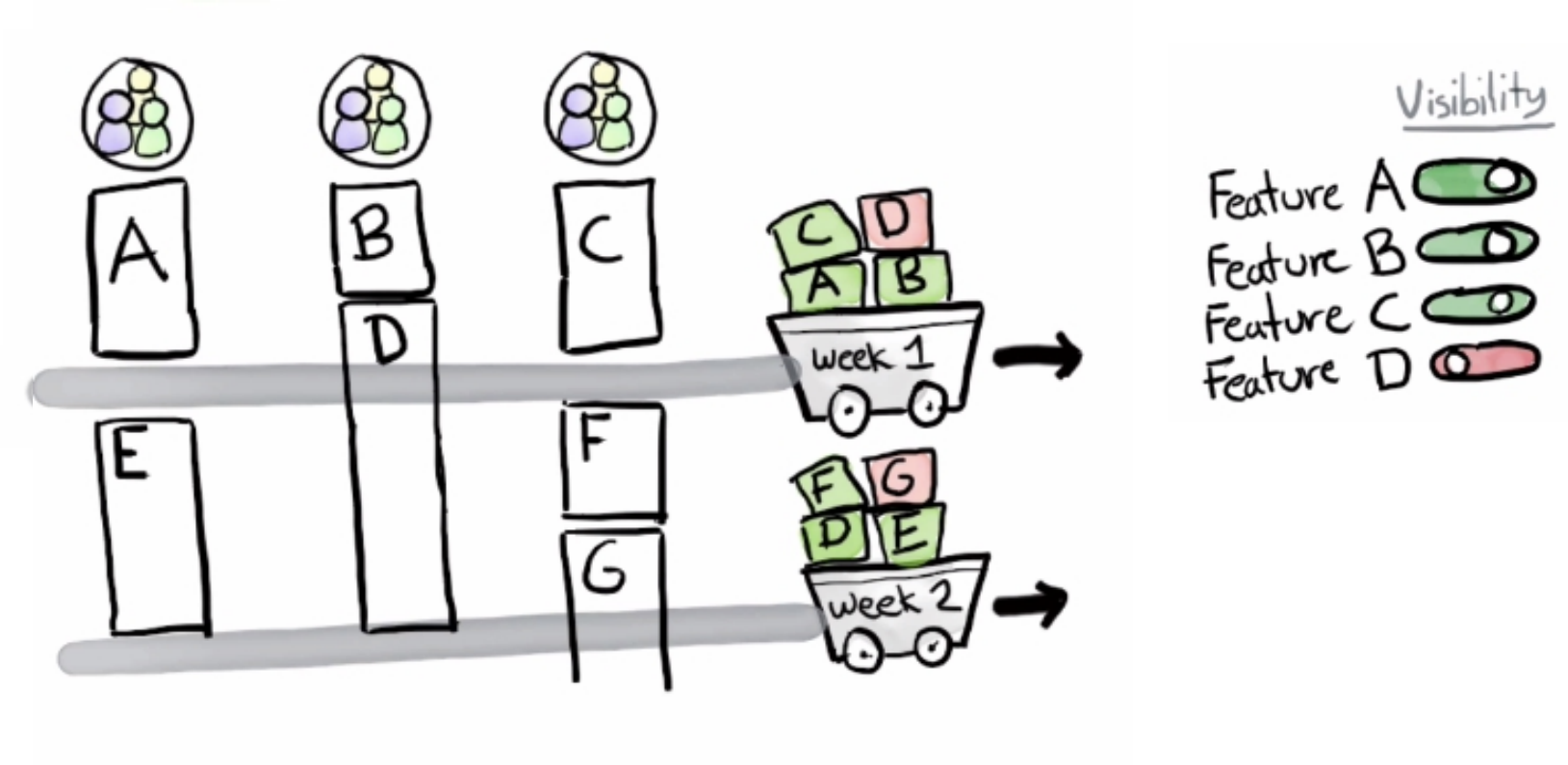
Henrik Kniberg

Self-service model



Henrik Kniberg

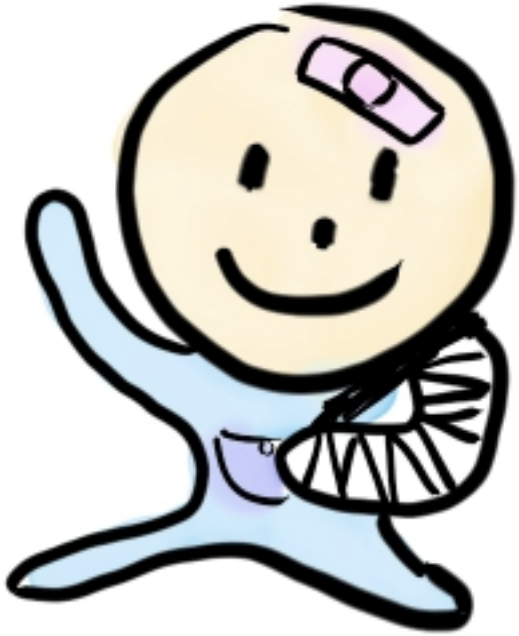
Release trains & Feature toggles



Failure Recovery is more important than Failure Avoidance

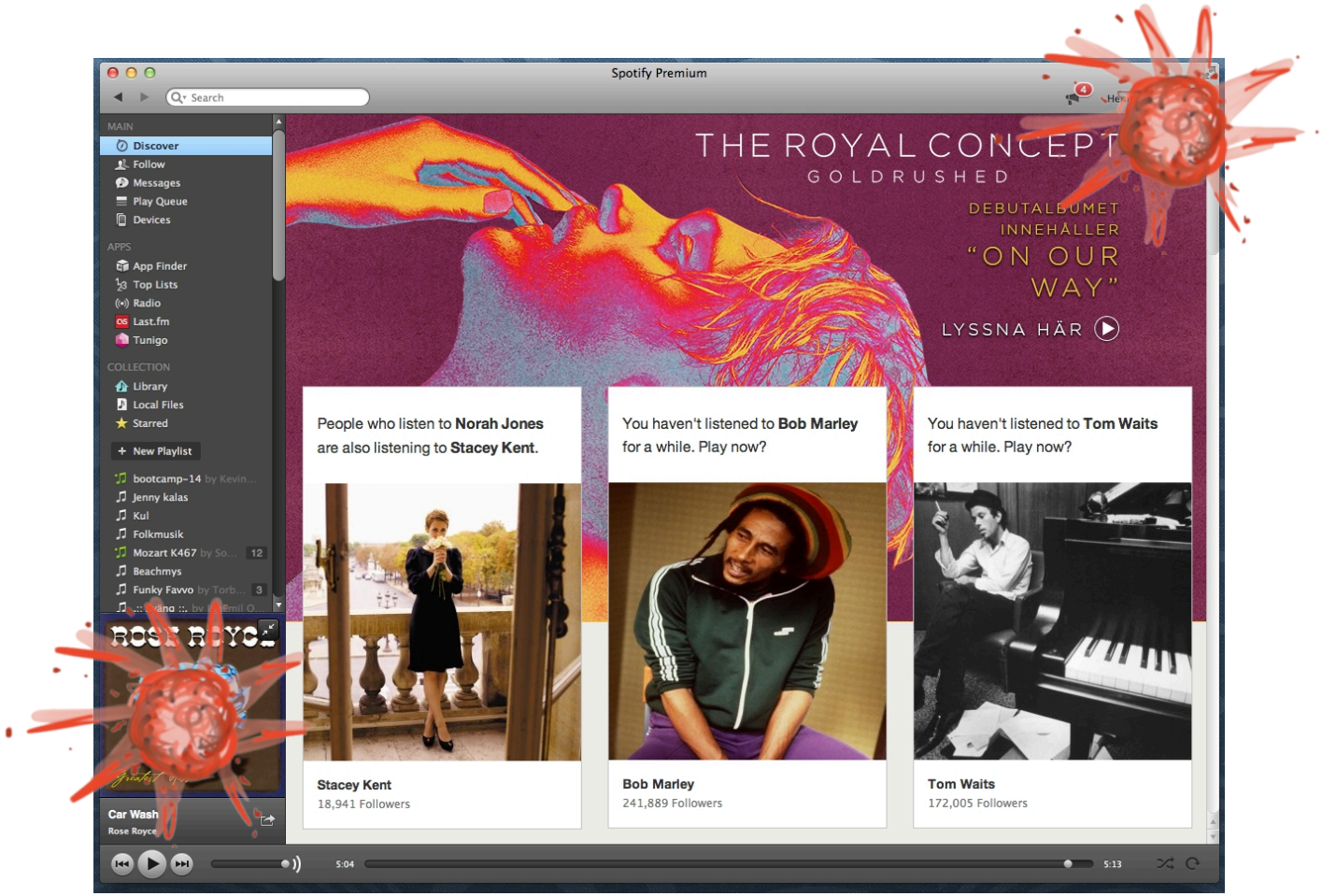
Failure Avoidance

Failure Recovery



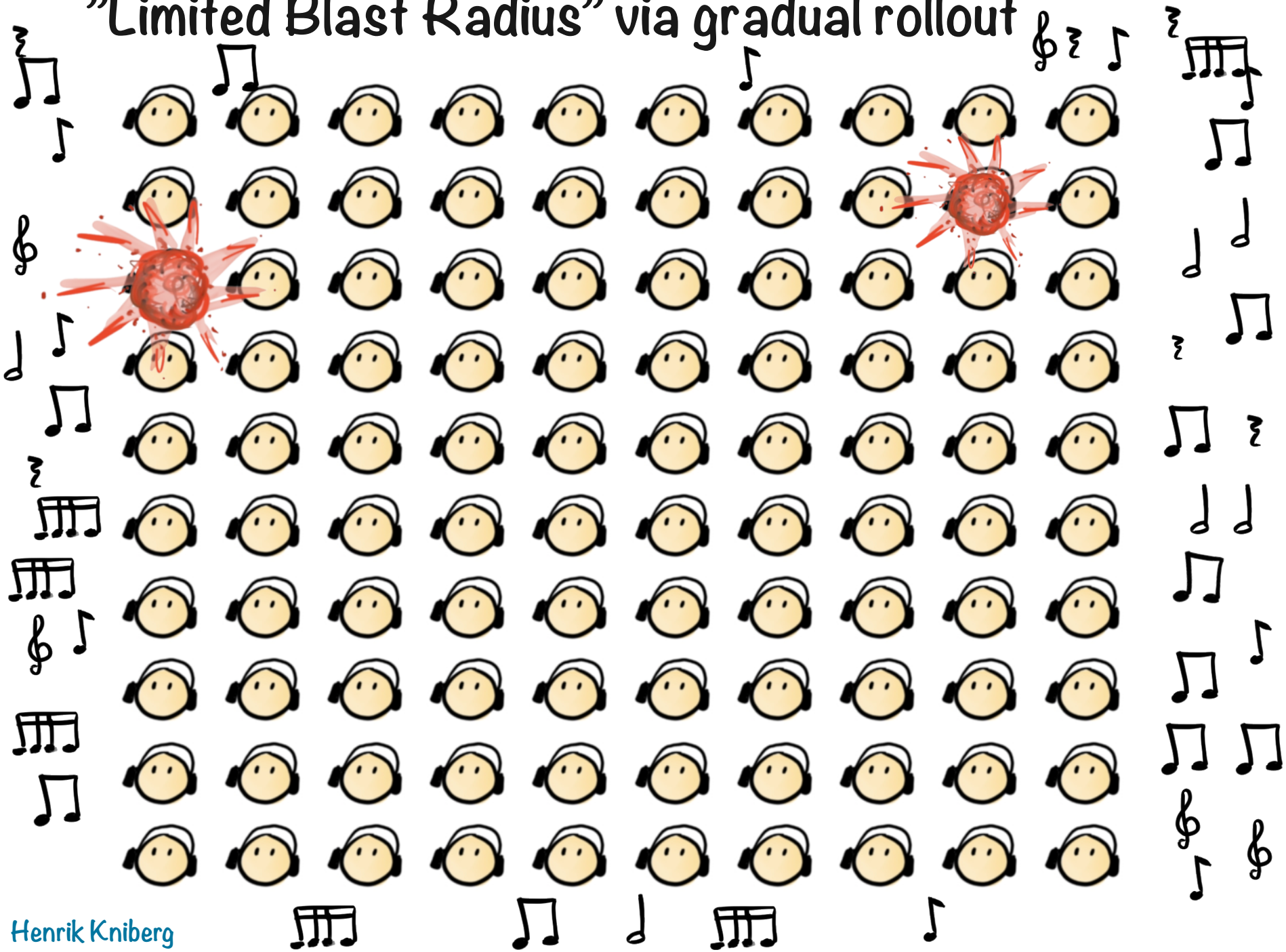
Henrik Kniberg

“Limited Blast Radius” via decoupled architecture



Henrik Kniberg

"Limited Blast Radius" via gradual rollout

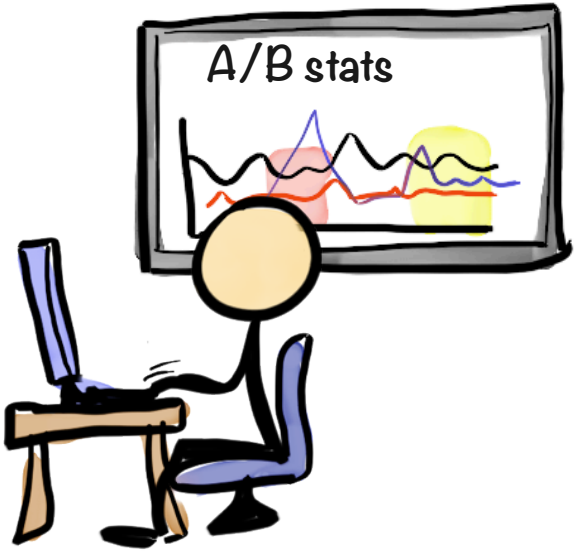


Henrik Kniberg

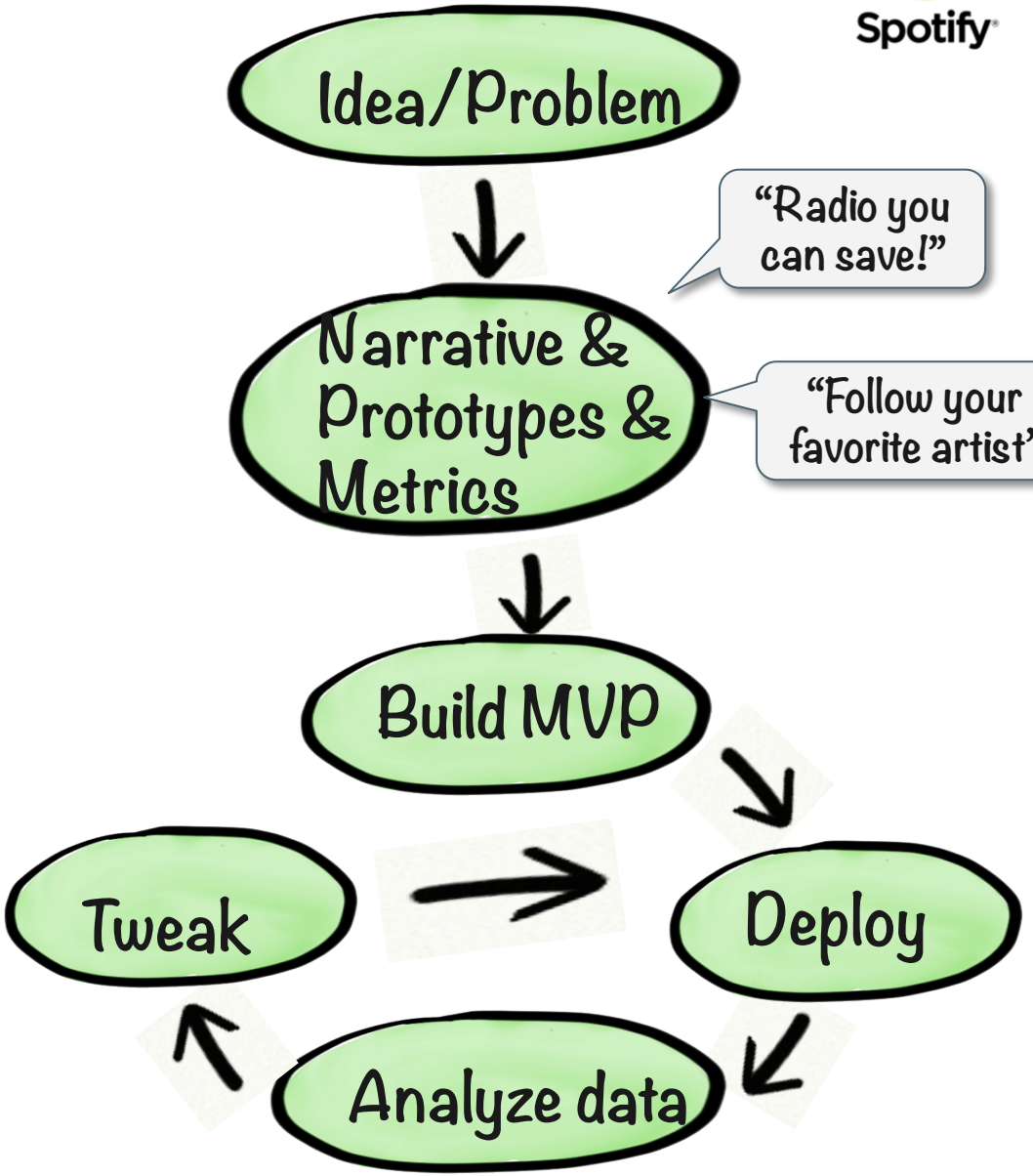
Trust > Control
100% control = 0% motion

If everything's under control,
you're going too slow!

- Mario Andretti



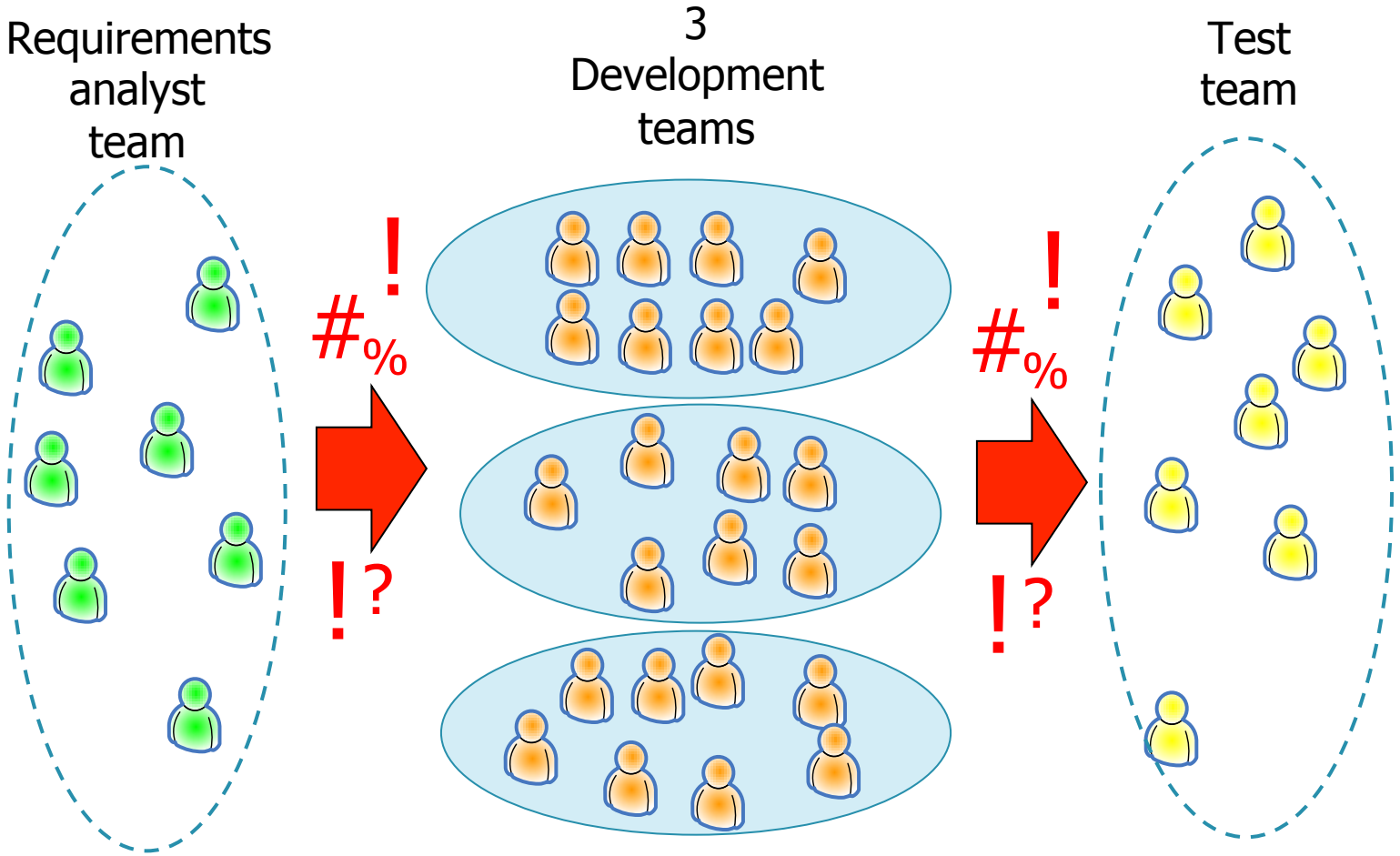
Spotify

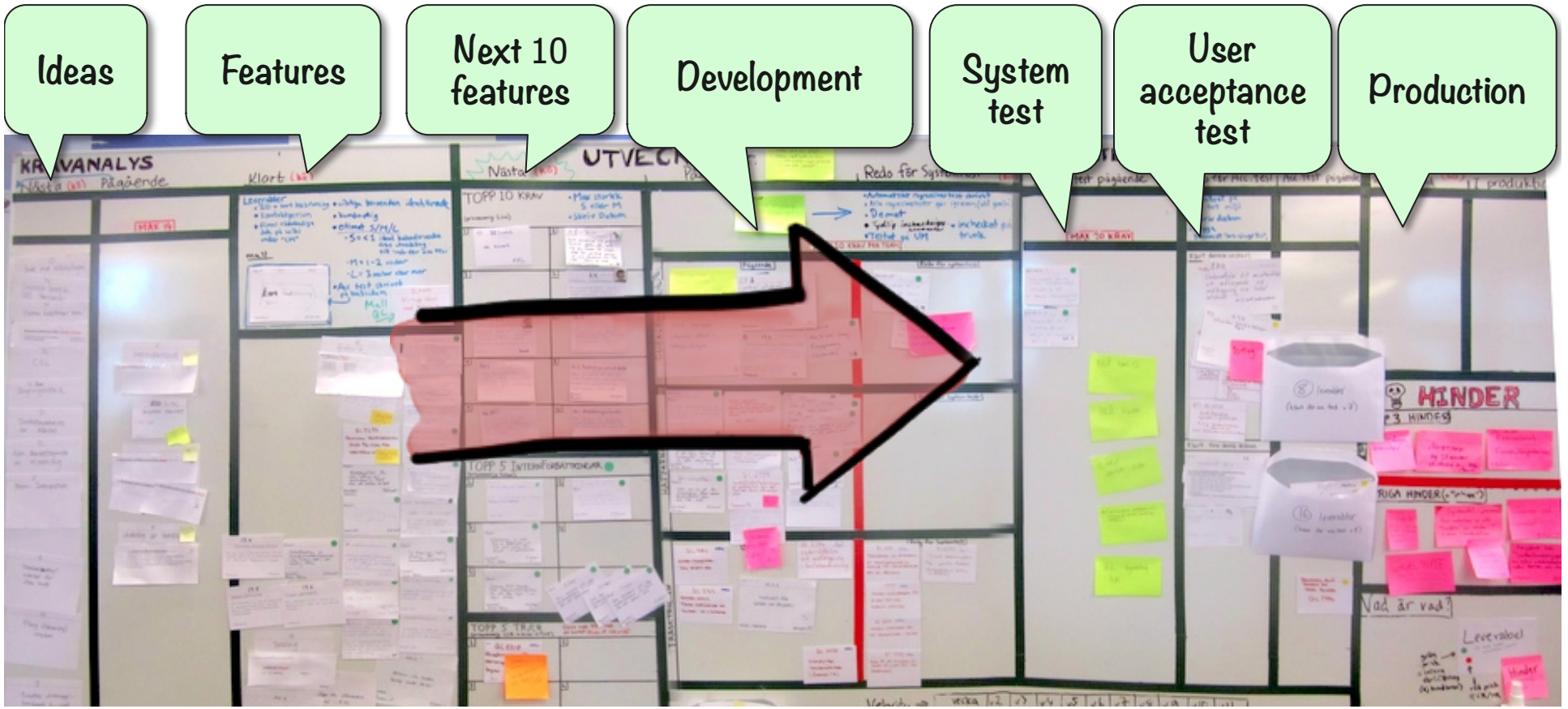


Henrik Kniberg

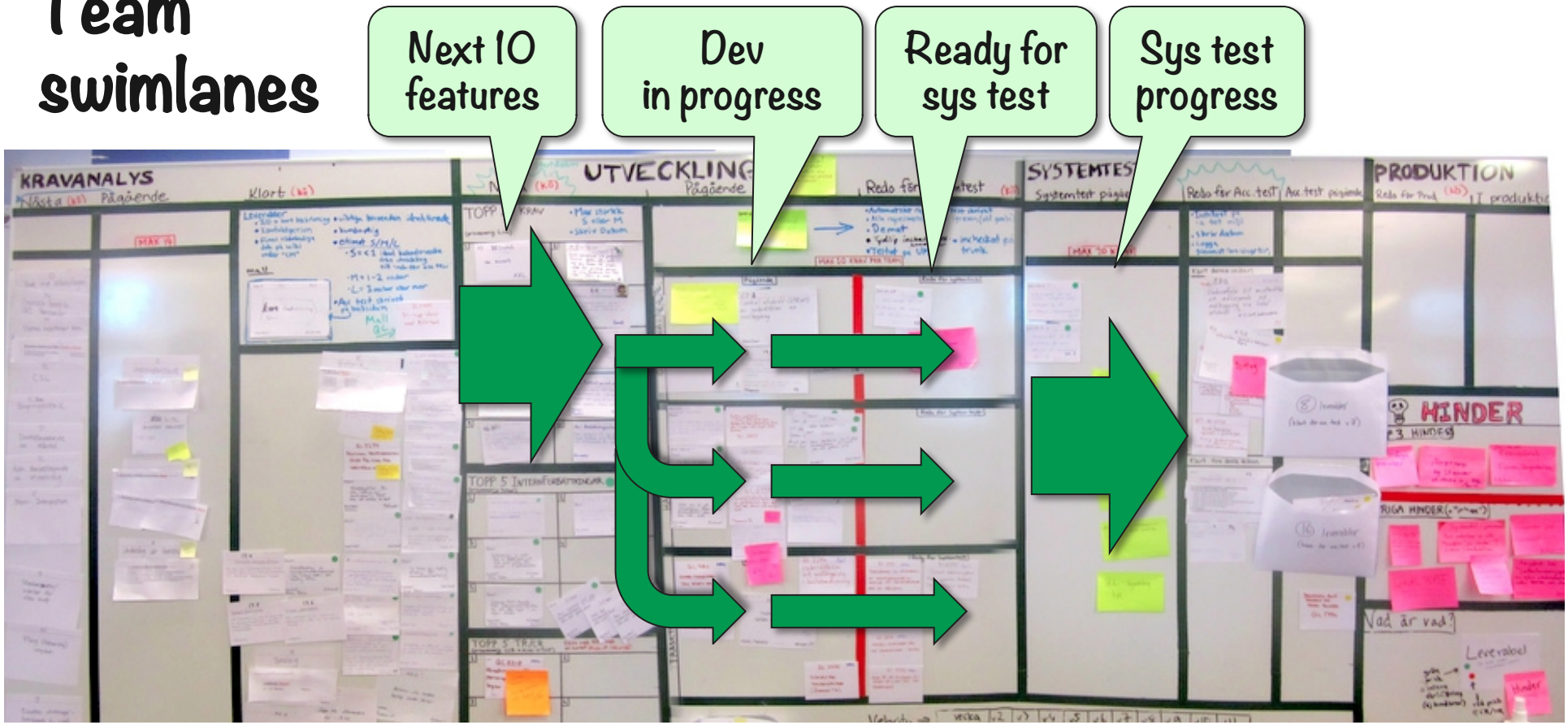
Example:
Big Government Project

Team structure - before




Ideas
Features
Next 10 features
Development
System test
User acceptance test
Production
HINDER

Team swimlanes

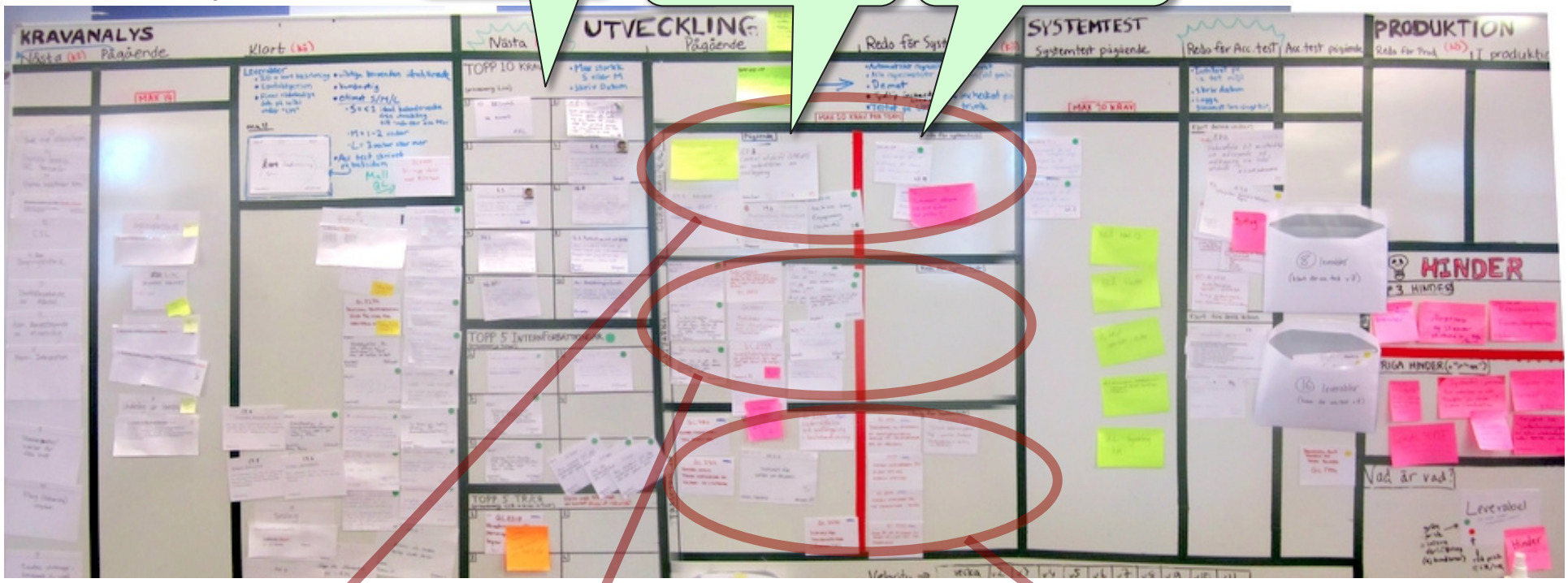


Team swimlanes

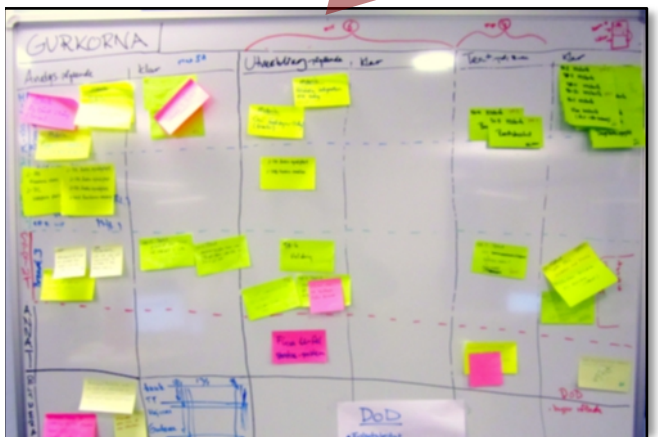
Next 10 features

Dev in progress

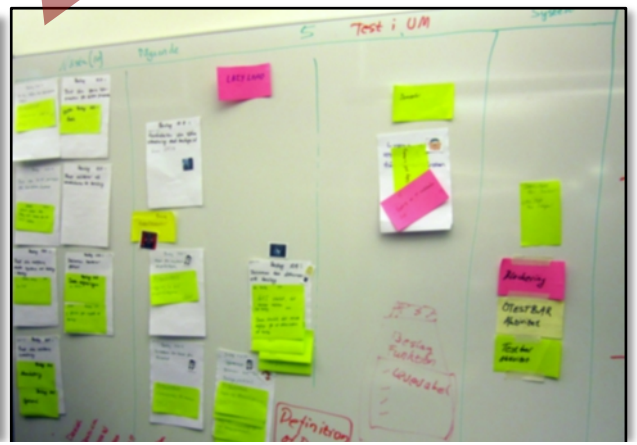
Ready for sys test



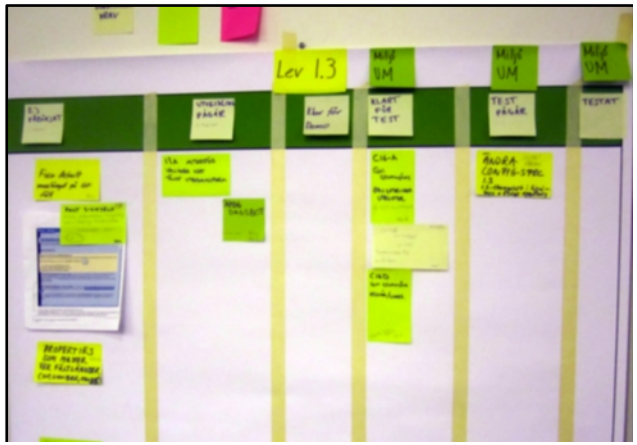
Dev Team 1



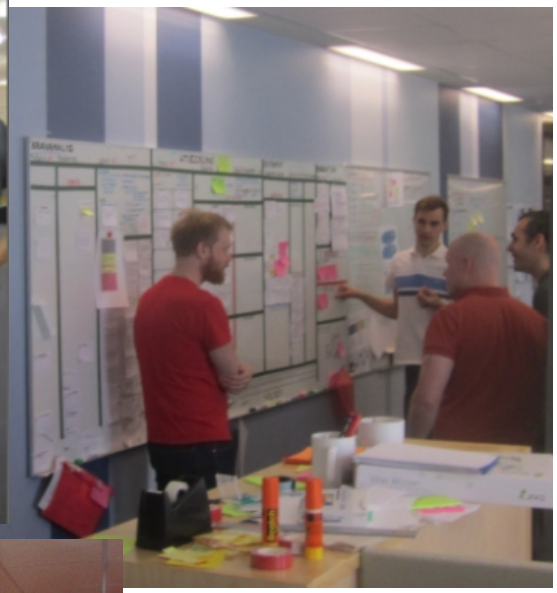
Dev Team 2

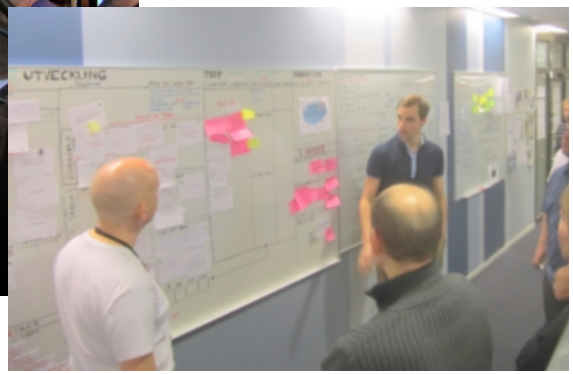


Dev Team 3



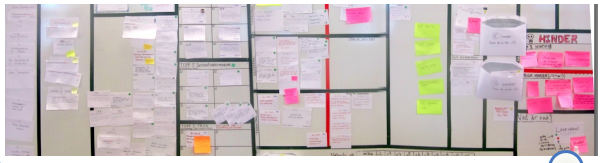
”Daily cocktail party” 9:15 – 10:15





Henrik Kniberg

10:00 – 10:15



Project sync

9:45 – 10:00

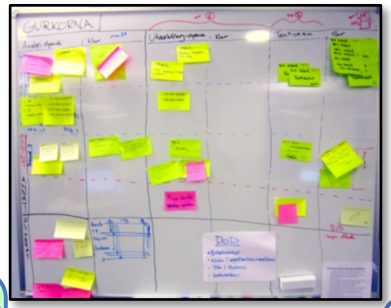
Test sync

9:45 – 10:00

Requirements sync

Dev sync

9:30 – 9:45

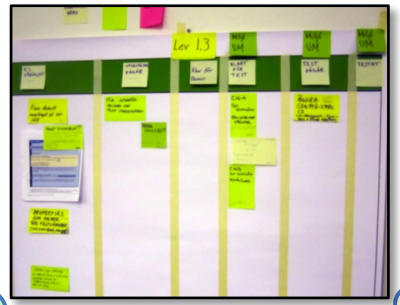


Feature team 1

9:30 – 9:45

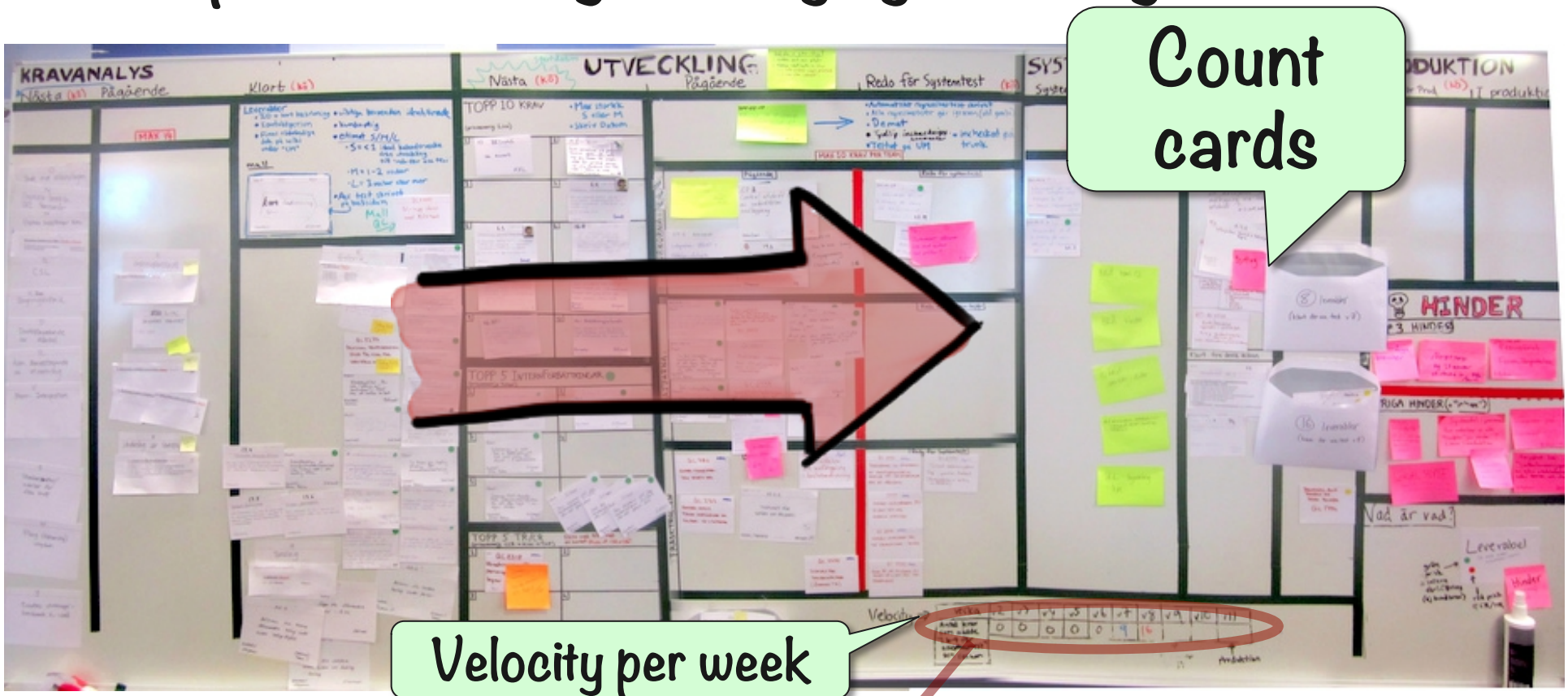
Feature team 2

9:15 – 9:30



Feature team 3

Example: Measuring velocity by counting cards

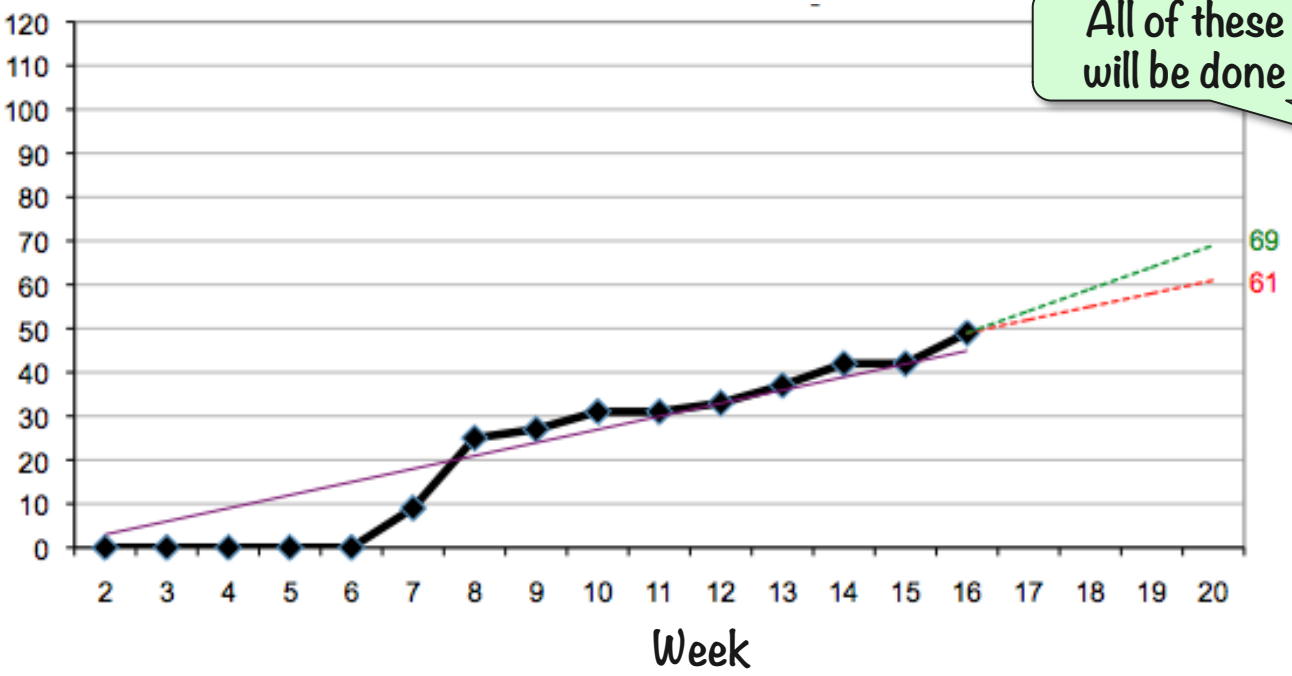


Vecka	v10	v11	v12	v13	v14	v15	v16	v17	v18
Antal nya funktioner som nått till 'Redo för AccTest'	4	0	2	4	5	0			

↑ Prodsättn.

VEL

Example: Release planning using a burnup chart

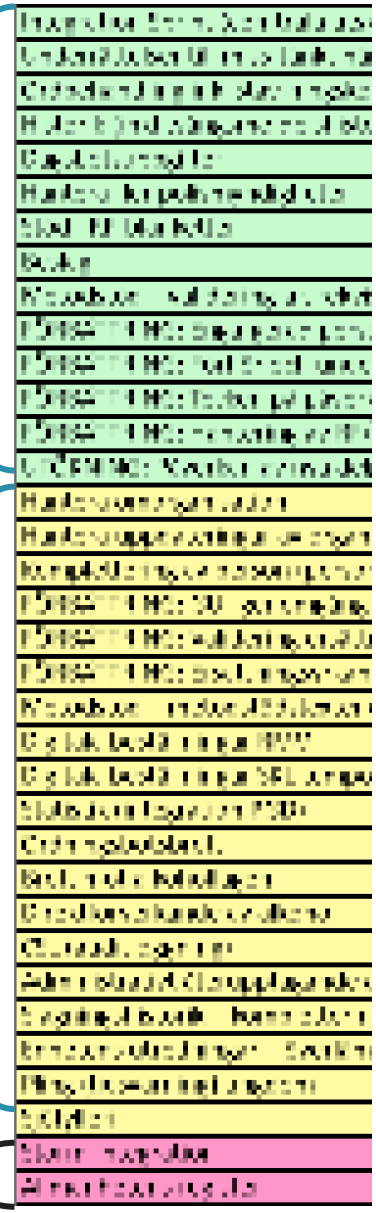


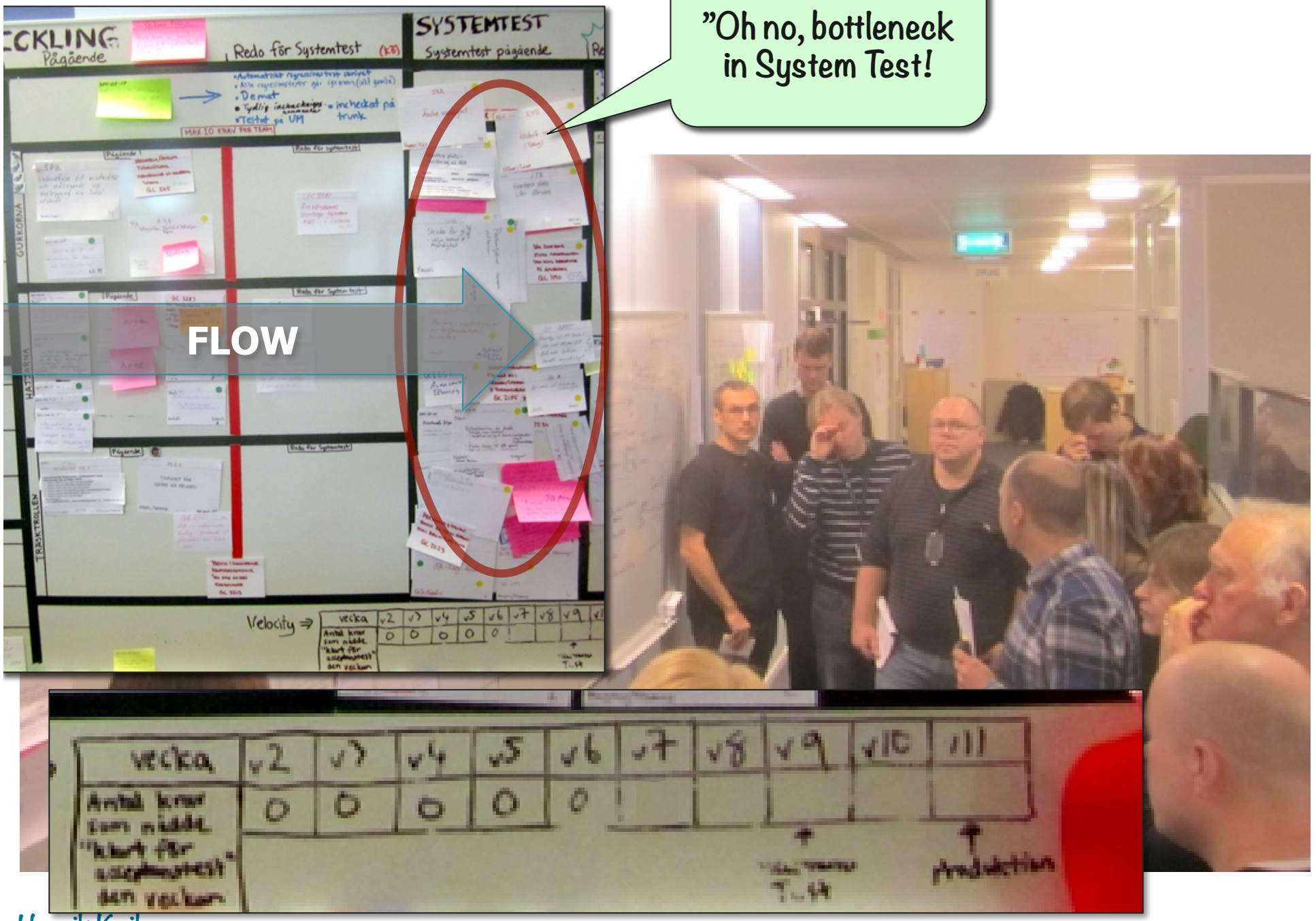
Total # of delivered features

All of these will be done

Some of these will be done, but not all

None of these will be done

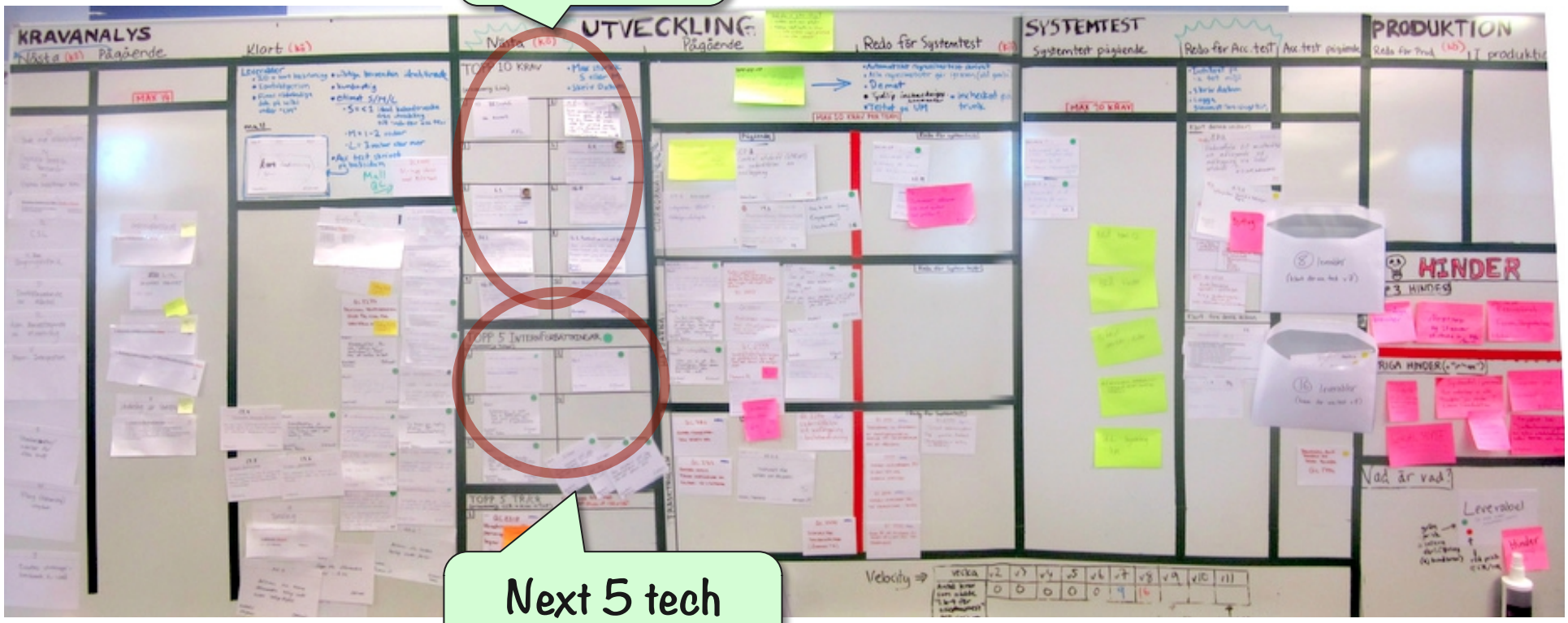




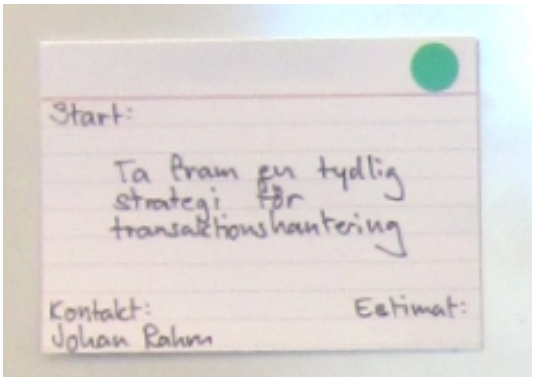
Henrik Kniberg

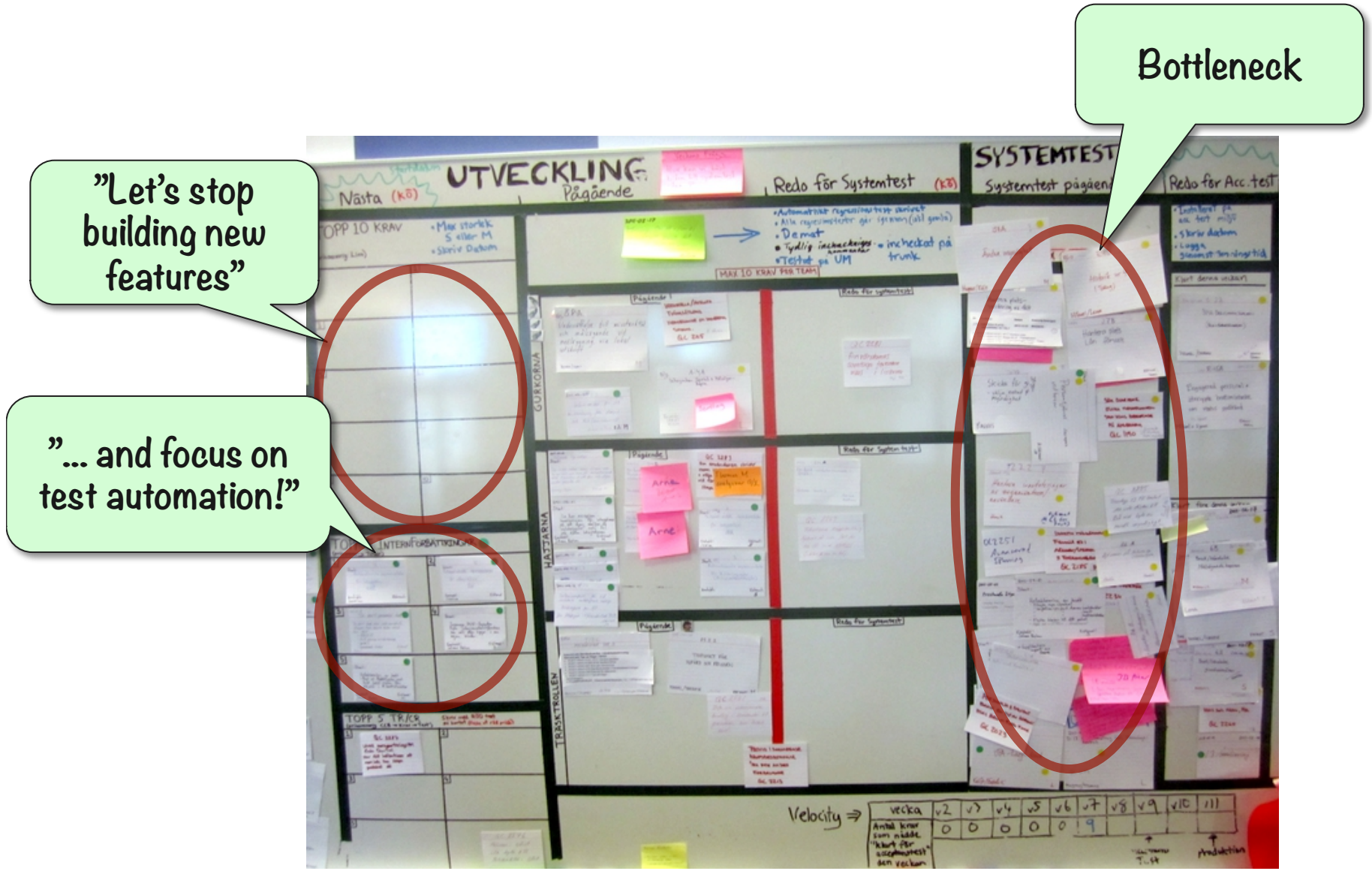
Tech stories

Next 10 features

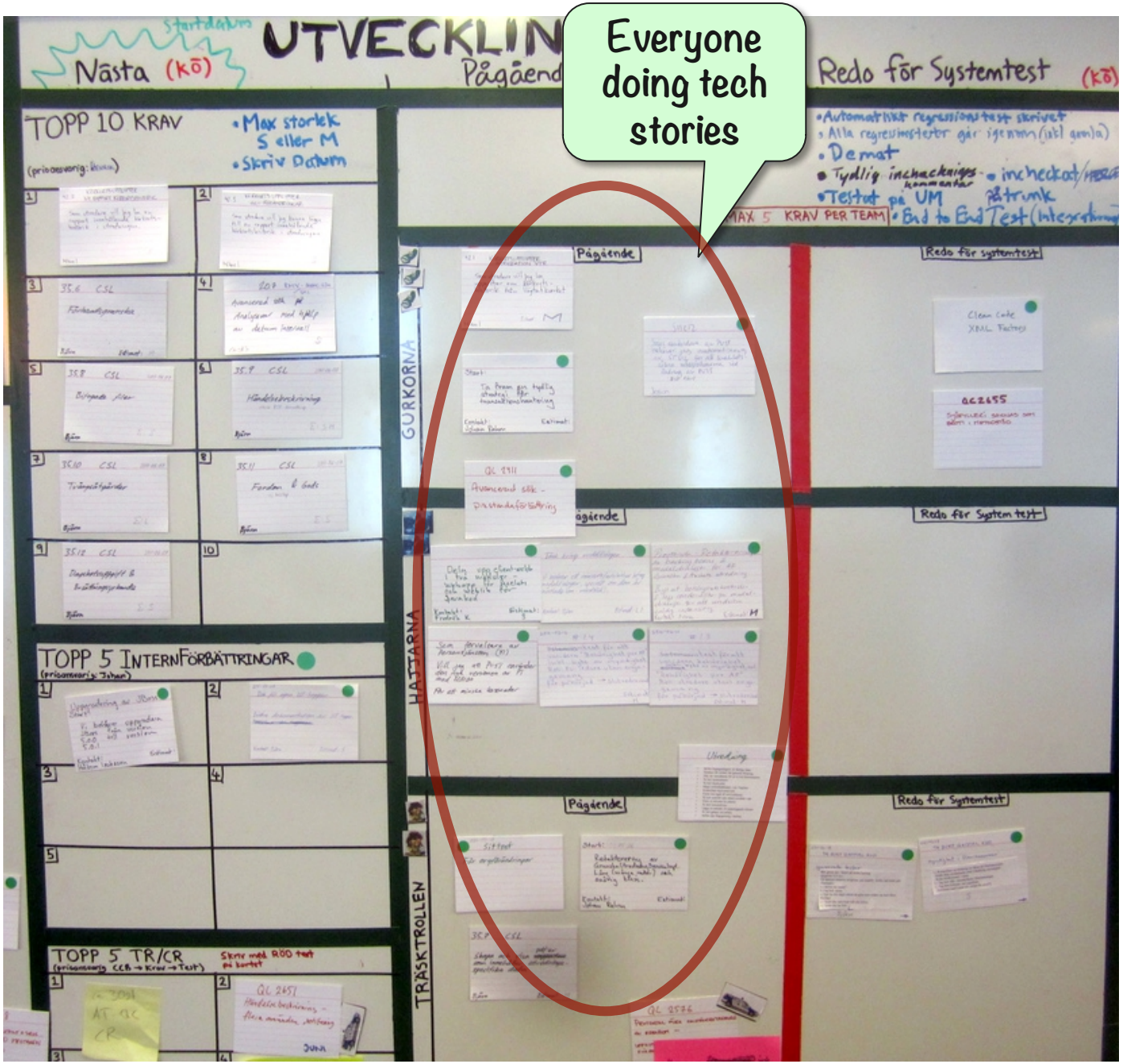


Next 5 tech stories



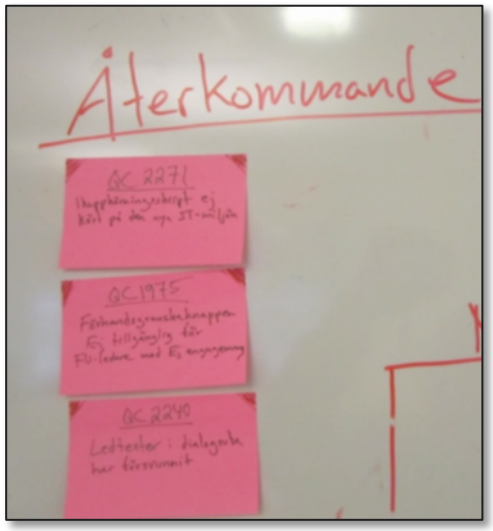


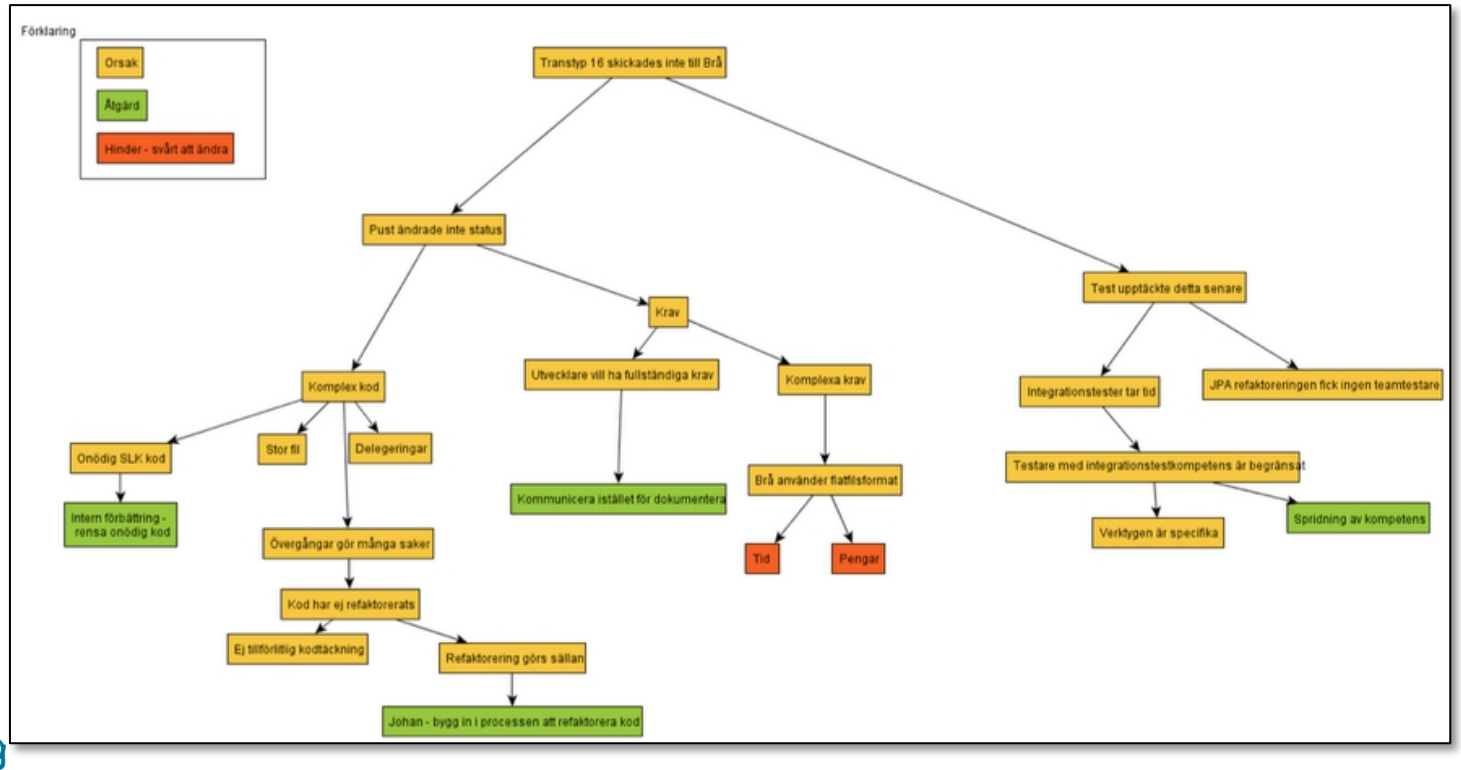
Henrik Kniberg



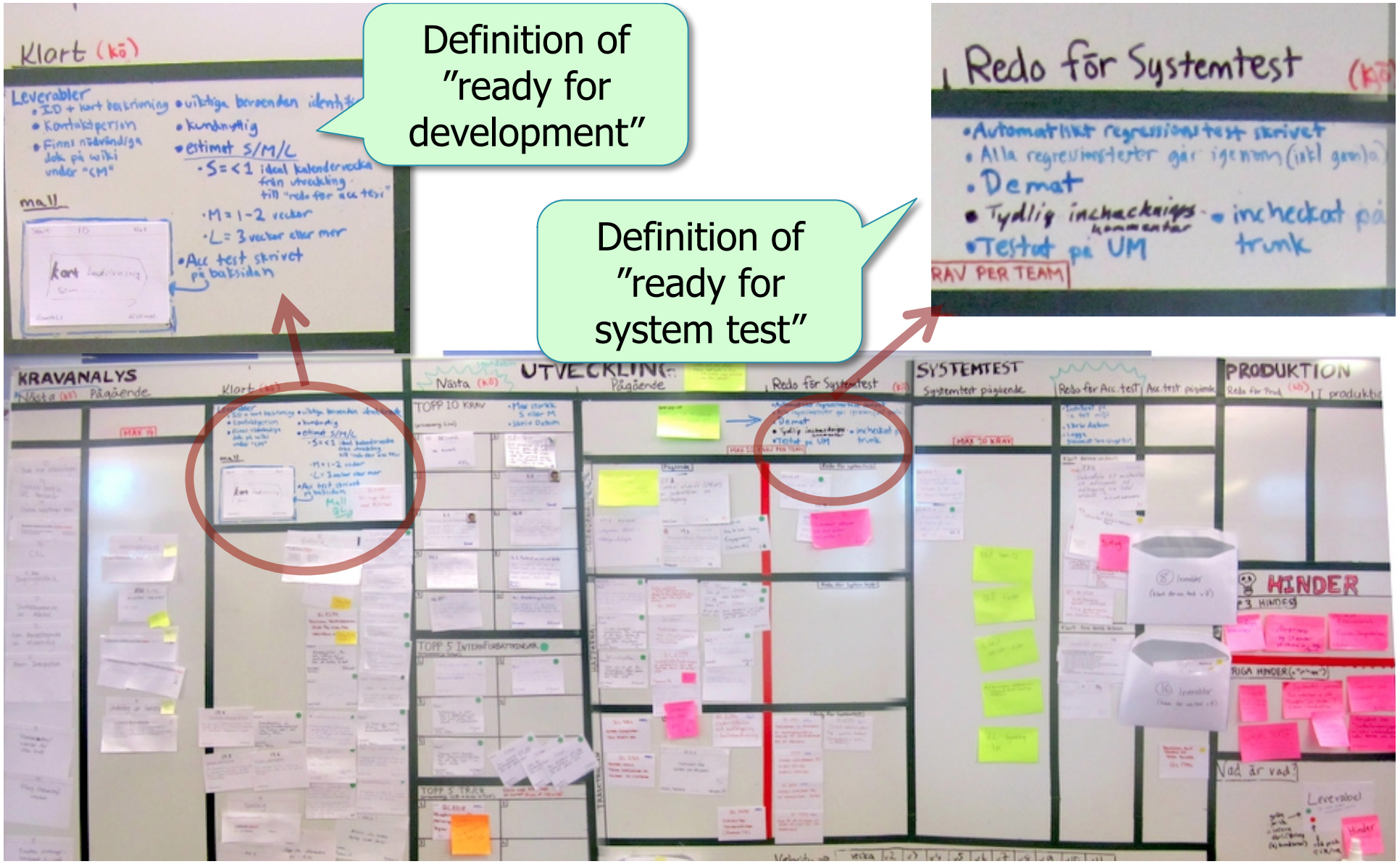
Henrik Kniberg

Top 3 recurring bugs





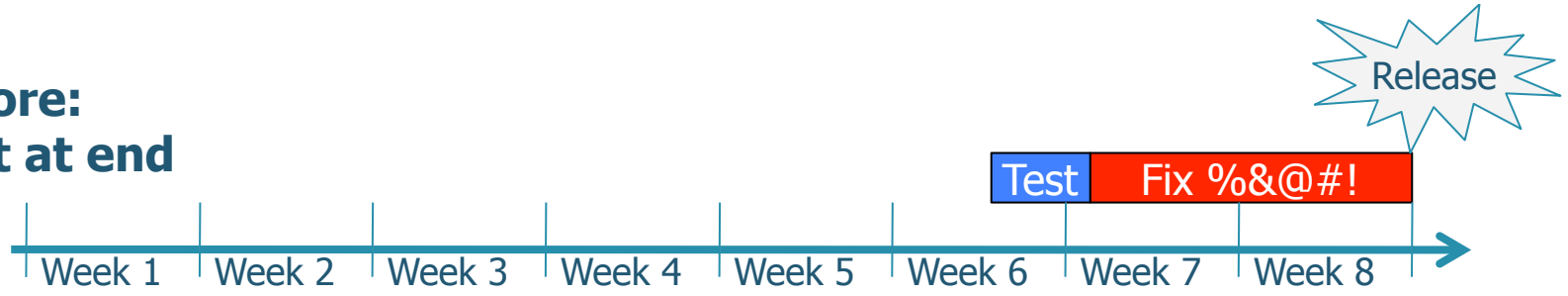
Henrik Kniberg


Definition of "ready for development"
Definition of "ready for system test"
Klart (kö)
Leverabler
mall
Redo för Systemtest
Automatiskt regressionstest skrivet
Alla regressionstester går igenom (inkl gamla)
Demat
Tydlig incheckningskommentar
Incheckat på trunk
Testat på UM
KRAV PER TEAM
KRAVANALYS
Nästa
Pågående
Klart
UTVECKLING
Redo för Systemtest
SYSTEMTEST
Systemtest pågående
Redo för Acc.test
Acc.test pågående
PRODUKTION
Redo för Prod
I produktion
HINDER
Vad är vad?
Leveravel

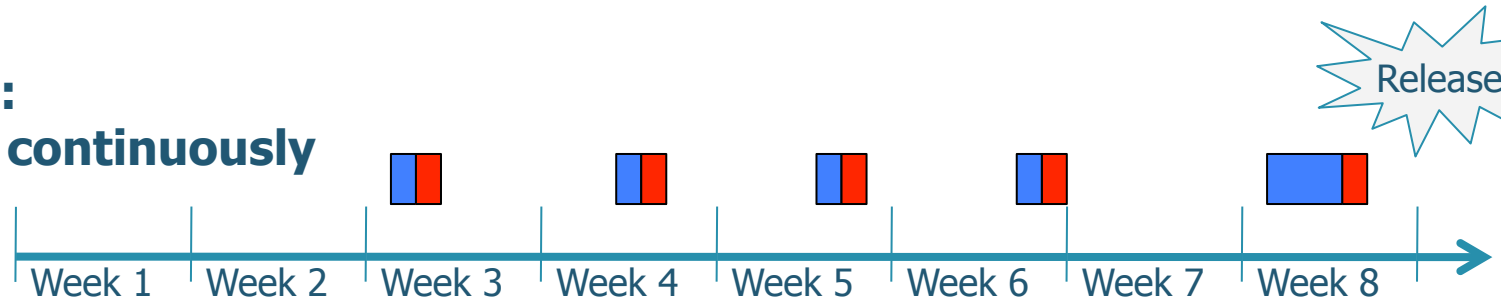
Henrik Kniberg

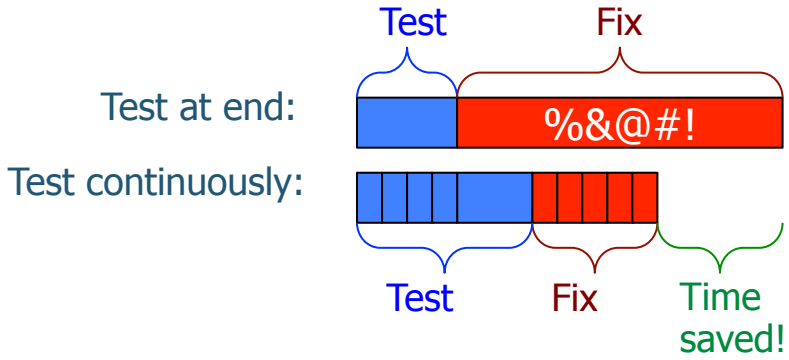
Henrik Kniberg

Before:
Test at end

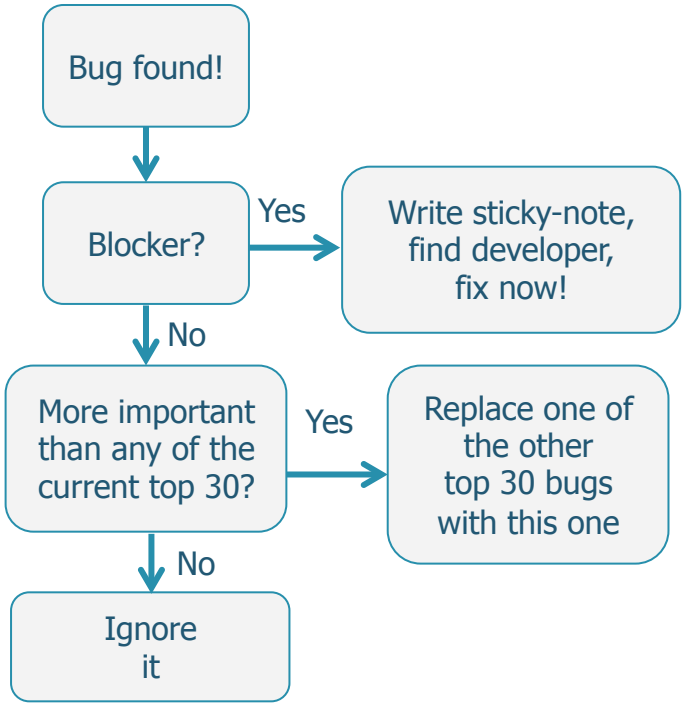


Now:
Test continuously

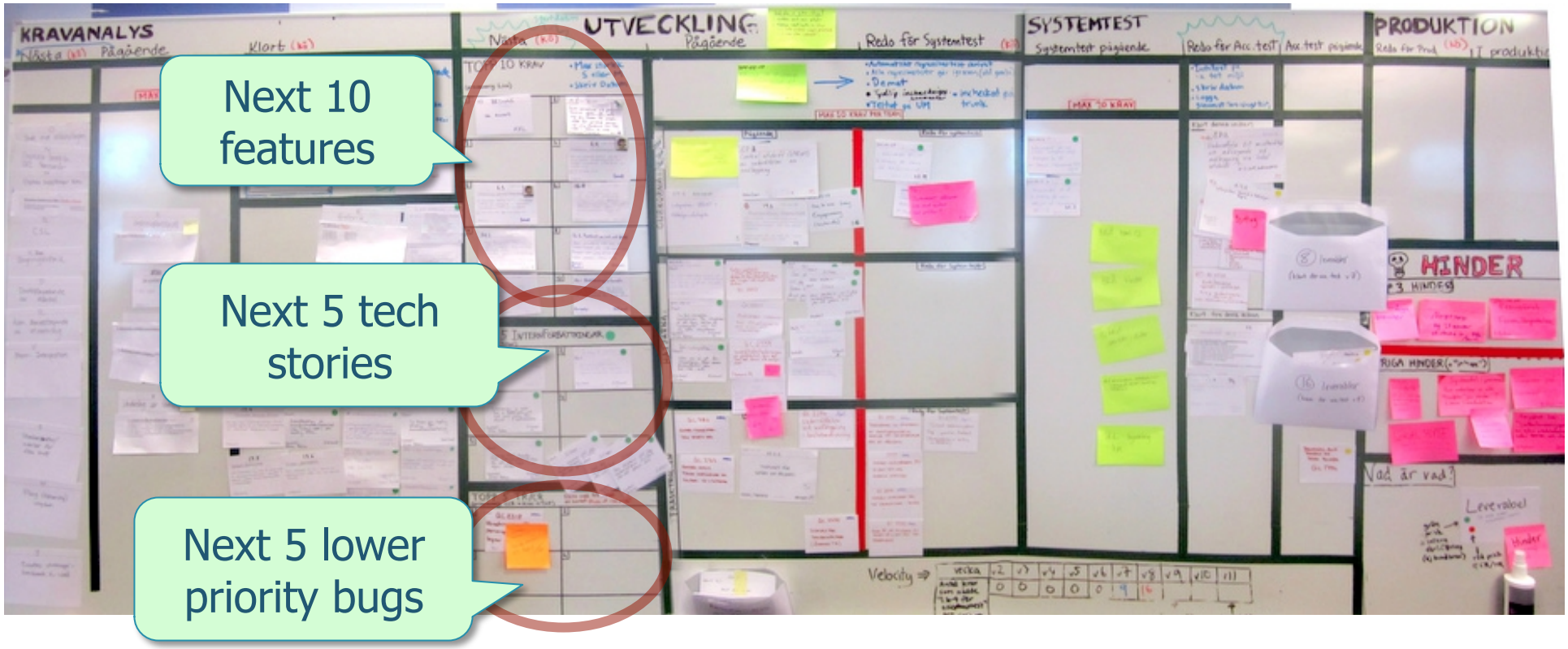




Bug fixing process



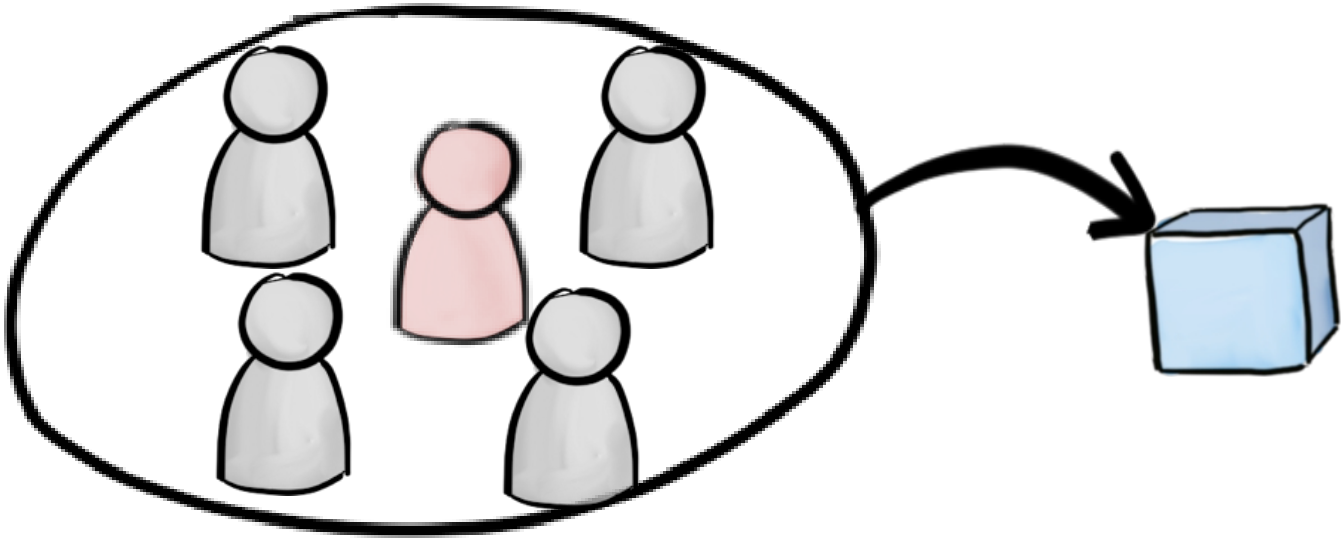
Three input queues



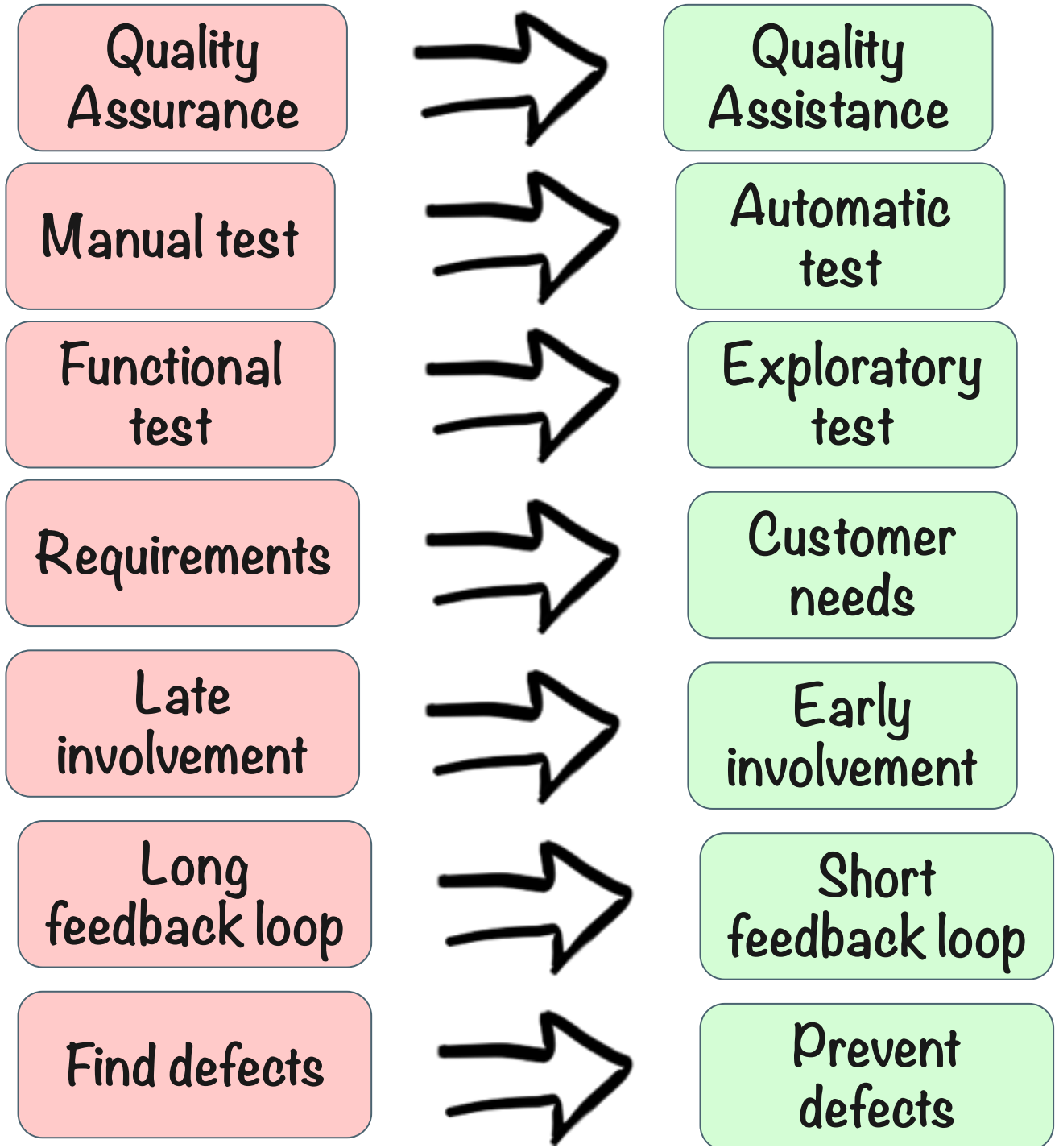
Wrapup

What is an Agile Tester?

- An agile team member with testing expertise
- Helps the team become quality-aware
- … and learn how to deliver better stuff



Mindset



Henrik Kniberg

Agile is a direction, not a place

Henrik Kniberg