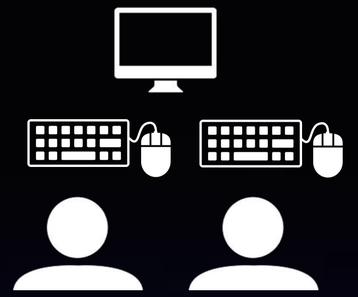
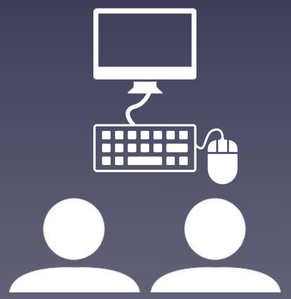
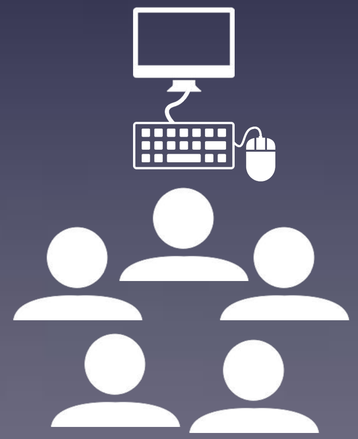


Pair- und Mob-Programming

Essenzielle Zutaten für erfolgreiche, moderne Agilität



Thomas Much
@thmuch
#jfn18



Über…

Thomas Much

Freelancer, Hamburg

@thmuch
#jfn18

Agile Developer Coach

Software Developer (Java et al.)

Es war einmal vor langer Zeit in einer weit, weit entfernten Galaxis….

Neulich im Büro….

„Boah, wer soll denn diesen Schrott pflegen?“

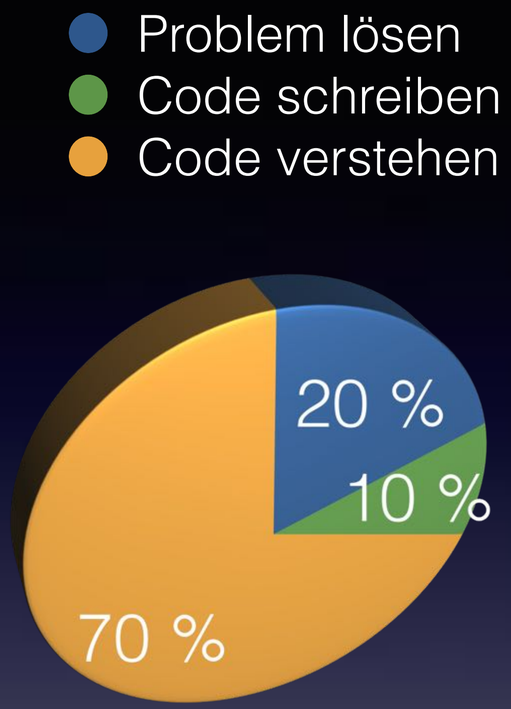
„Wer hat denn den Code geschrieben?“

„Oh. Ich selber.“

„Das muss Kollege X machen,

er hat das mit seinem #!@%&$!? Stil programmiert.“

Problem: Lesbarkeit



- Code wird sehr viel häufiger gelesen als geschrieben
- Verständlichkeit essenziell für Wartung & Pflege!

https://www.slideshare.net/cairolali/langlebige-architekturen

- Wir Entwickler schreiben Code häufig schludrig – oder zu „clever“
- *Wer gibt uns Feedback, bevor es zu spät ist?*

Problem: Einfachheit

„Everyone knows that debugging is twice as hard
as writing a program in the first place.

So if you're as clever as you can be when you write it,
how will you ever debug it?“

– Brian Kernighan

Wer schützt uns davor, zu „clever“ zu sein?

„Dafür machen wir Code-Reviews!“

Code-Reviews?

Reviews oft nicht ehrlich (systembedingt).

Falsche Anreize.

Feedback zu spät.

Wer macht dann noch grundlegende Änderungen?

„Kollege A ist gerade im Urlaub,
der Bugfix muss warten.“

„Kollege B ist nicht mehr im Unternehmen,
sein Programm müssten wir neu schreiben.“

„Das wird Monate dauern, bis der neue Kollege C
das Projekt verstanden hat und mitprogrammieren kann.“

Problem: Wissensverteilung

Fehlende Wissensverteilung.

Kein Collective ~~Code~~Product Ownership.

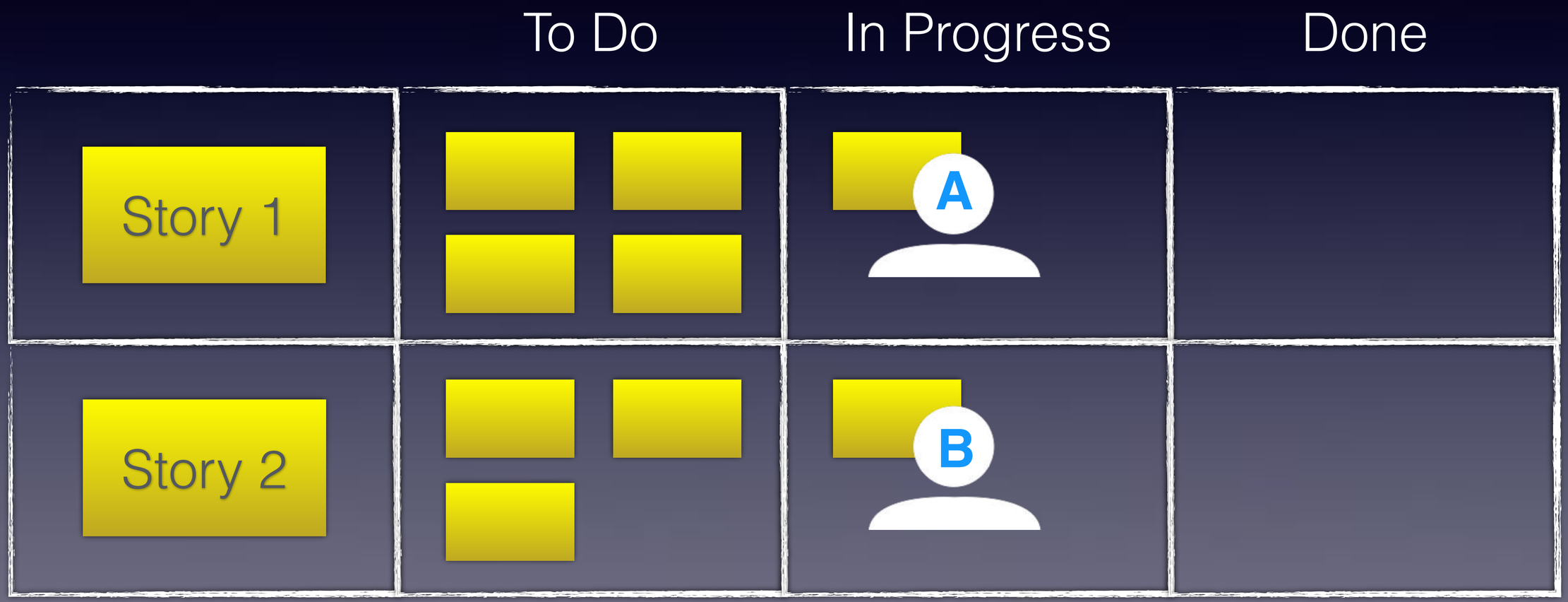
Wie? Doku, Workshops, Schulungen …

Arbeiten wir als Team an unserem Code / unserem Produkt?

„Aber wir sind doch ein Team?!“

„Team“arbeit

Lösung! „Wir werden agil.“

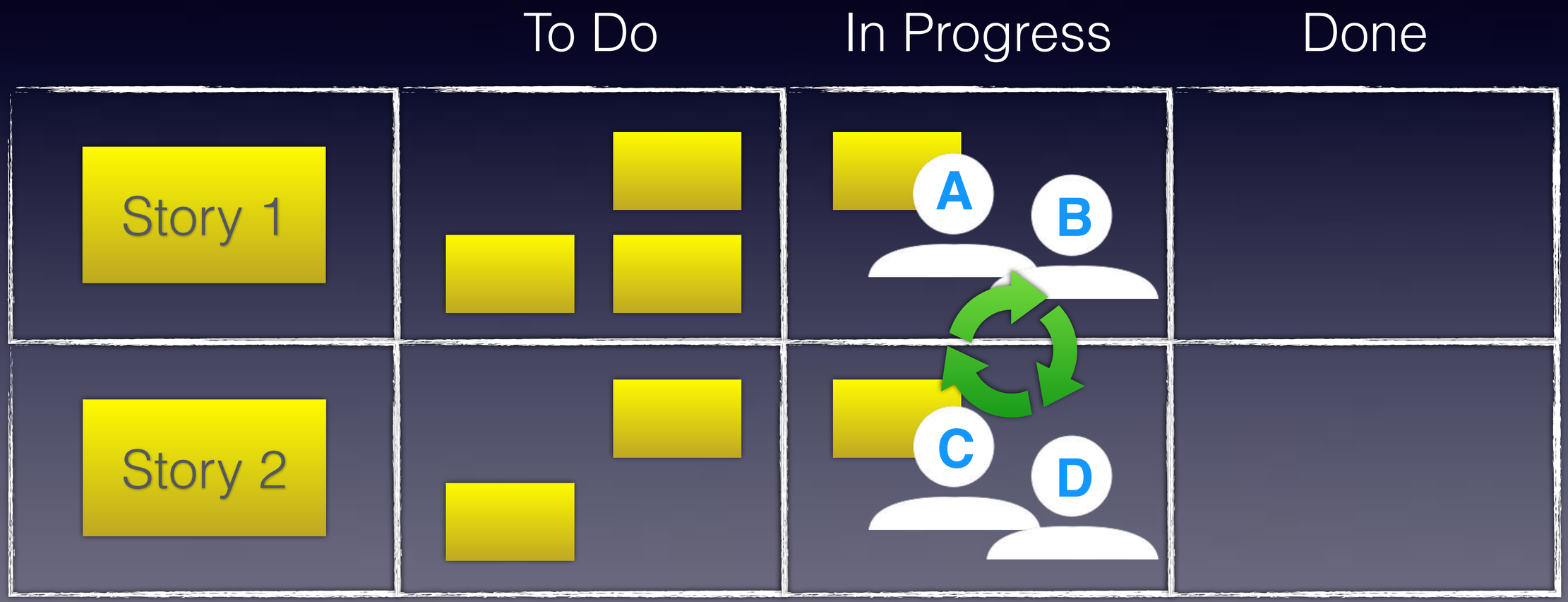


„If your agile Team has individual work assignments,

I suspect it is neither agile nor team.“

– Tim Ottinger

Echte Zusammenarbeit



Problem: Zusammenarbeit

Wie arbeiten wir wirklich *zusammen*

statt nebeneinander her?

Probleme!?

Lesbarkeit / Einfachheit / Verständlichkeit

Wartbarkeit / Wissensverteilung

Zusammenarbeit

Was wollen wir erreichen?

Schnellstens „fertig“ werden?

(„Nach mir die Sintflut“)

Oder einigermaßen wartbare Software entwickeln?

Wartbare Software entwickeln

In „meinen“ Projekten:
Kunden müssen bzw. wollen die Software selber pflegen.

Ziel daher:
Wartbare Software entwickeln.

Unterstützt durch Pair Programming.

Pair Programming Coachin

Idee: Pair Pro

Seit 2013 in za

E-Commerce, E

OTTO

DEV.OTTO.DE

— DEV + OPS BEI OTTO.DE —

About Meet Us GitHub Jobs Impressum

Continuous Delivery

Process Automation and Continuous Delivery at **OTTO**.de

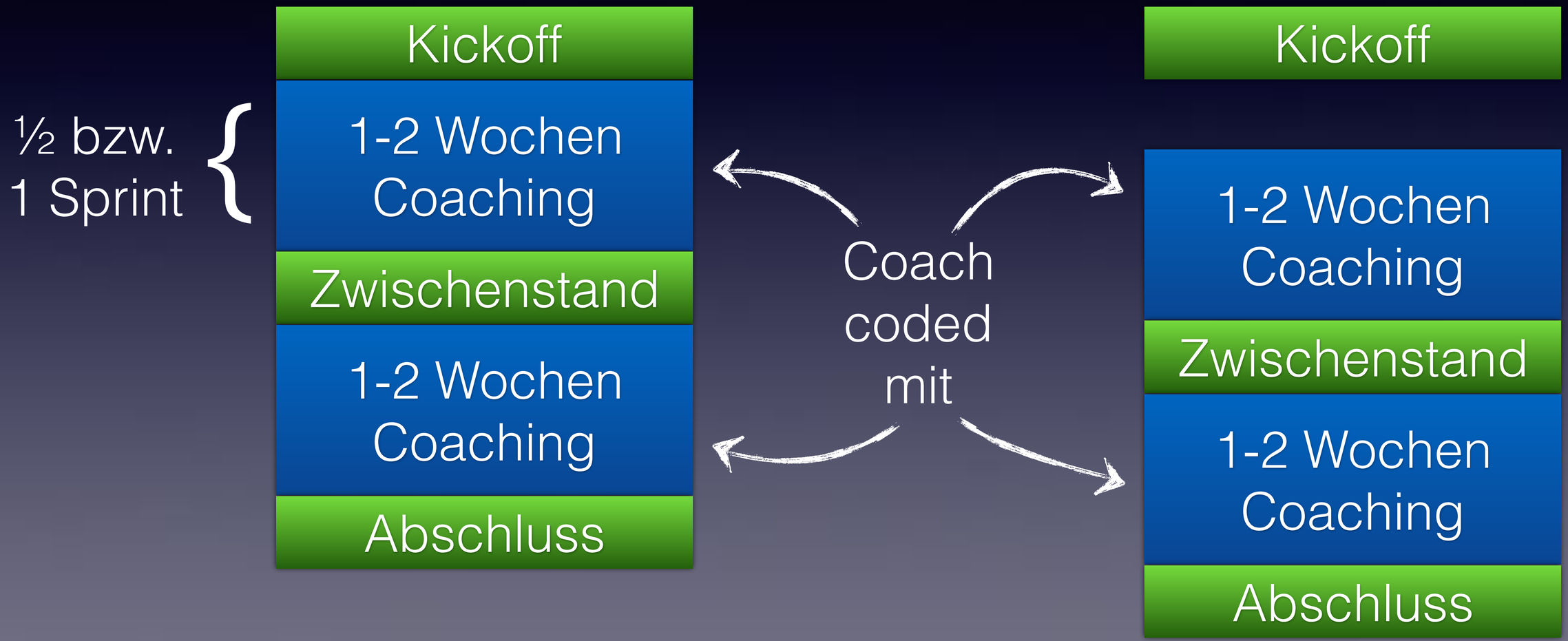
Successful Mountaineering – Reaching the Summit of **LHOTSE** with Agile Software Development

LHOTSE is the internal code name for our project

Continuous Everything: Fast Feedback Driven Development

Fast feedback is a cornerstone of agile software development. When developing the LHOTSE project at Otto, we tried to be as agile as possible

Coaching-Ablauf



Pair Programming: Schnell erklärt

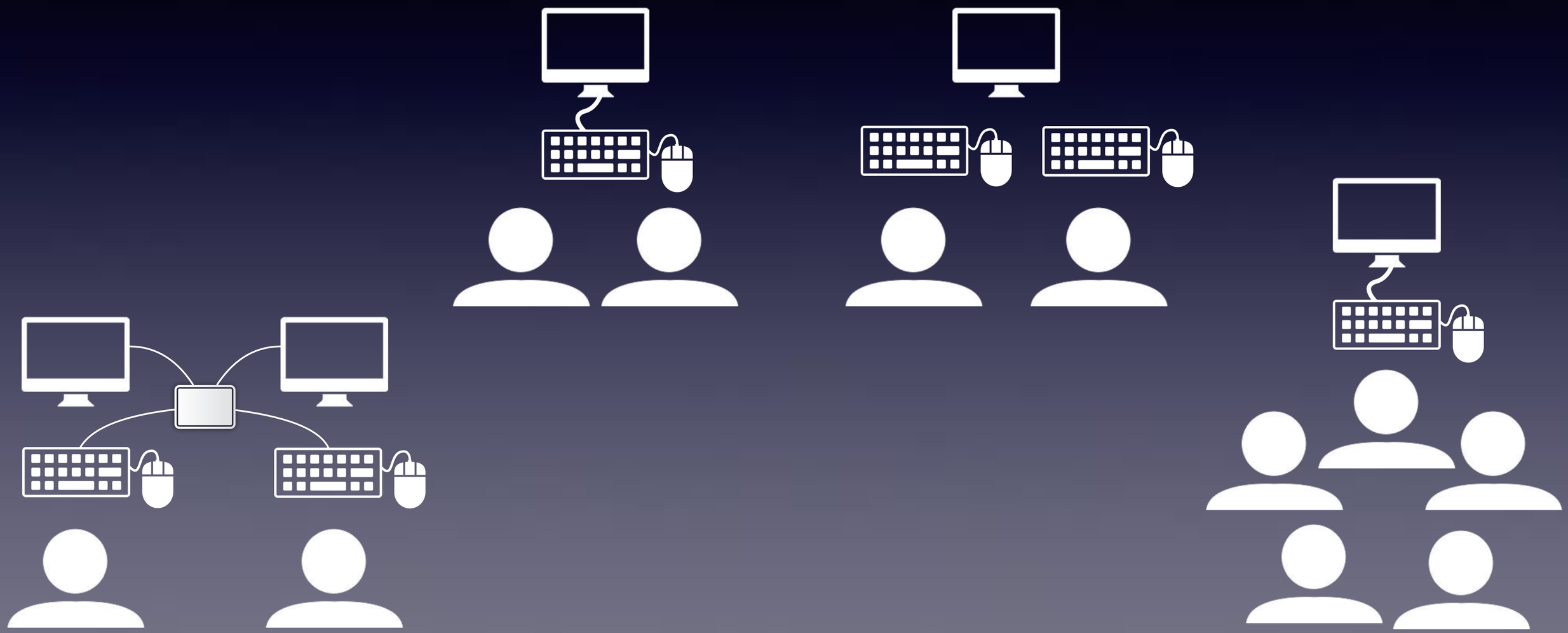
1
Aufgabe

Driver & Navigator

https://commons.wikimedia.org/wiki/File:FORD_Taunus_17M_P2_deLuxe_Steering_wheel.jpg

http://www.marcusvenzke.de/HamburgKarte/

Diverse Varianten



Die Rettung: Pair Programming

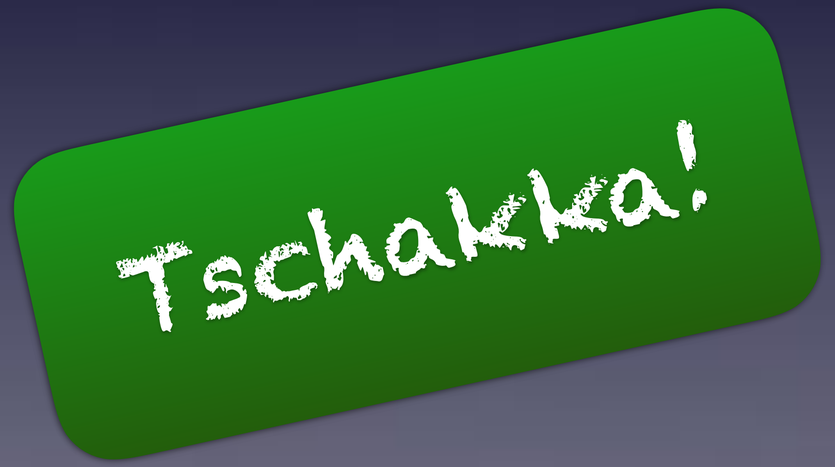
Know-How-Transfer

Collective ~~Code~~Product Ownership

Clean Code

Wartbarkeit

Nachhaltigkeit, Qualität



Keine neue Idee

Pair Programming – ca. 1992? .. 2000 …

Extreme Programming (XP) – ca. 1996 .. 2000 …

„Flaccid Scrum“ (Fowler 2009): Scrum = XP - Practices 🤔

Pair Programming ist „in“

Chef:
„Wir machen jetzt Pair Programming.
Ab sofort sitzt ihr zu zweit vorm Rechner!“

Entwickler A: „Cool, endlich!“
Entwickler B: „Och nö, das mag ich nicht“
Entwickler C: „???“

„Der andere ist aber viel zu schnell für mich“

„Der andere ist zu langsam und kapiert es einfach nicht“

„Das schlaucht total“

„Ich will lieber alleine arbeiten“

Anti-Patterns

Festes Pair sitzt auf einer Story.

Die 4 Wochen dauert.

Eigentlich tippt immer derselbe.

Abwechslung & Kreativität fehlen!

Das Kleingedruckte

Was üben wir?

Passende Kommunikation

Rollenwechsel

Effizient Pausen machen

Pair-Wechsel (Rotation)

Umgang mit unterschiedlichen Wissensständen

Vorbereitung von Stories & Tasks

Passende Kommunikation

Stille ↔ zu viel reden

Als Techniker müssen wir gute Kommunikation üben…

Driver erklärt: „Warum“, nicht „Was“

Navigator sagt nicht zu jeder Kleinigkeit etwas.

Gutes Pair Programming

Richtiges Pair Programming ist

Kommunikation über *geschriebenen* Code.

Nicht nur reden über fiktiven Code.

Warum hilft Pair Programming?

*Wir unterliegen bestimmten „**Brain Patterns**“:*

Interpretation

„Was“ vs. „Wie“

…

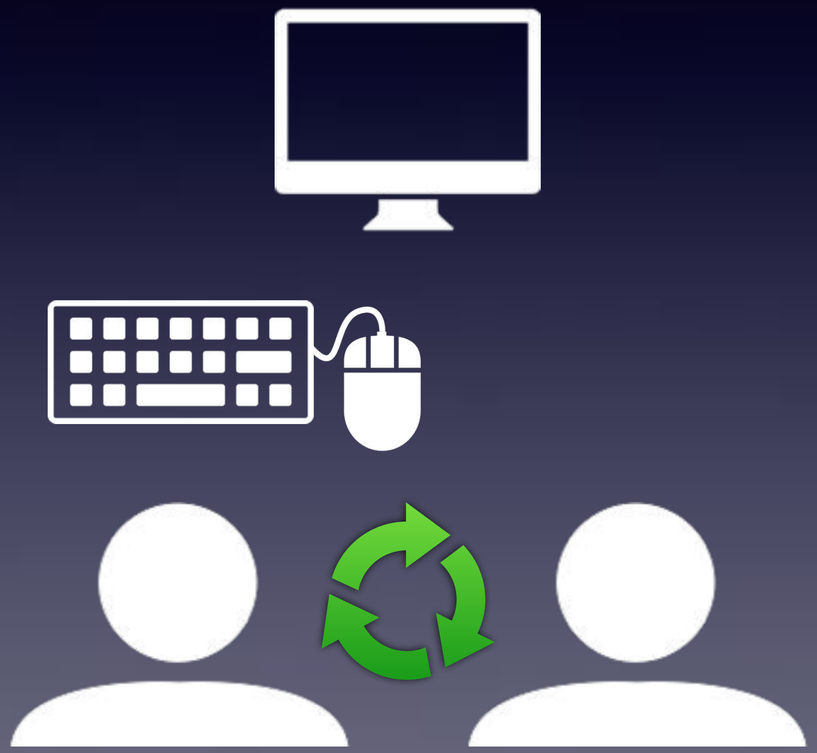
https://www.smidig.de/2016/09/brain-patterns-in-der-softwarentwicklung/

https://javabarista.blogspot.de/2016/06/pair-programming-das-gehirn.html

Rollenwechsel

- Häufig!
- Alle paar Minuten?!
- *Erhält Aufmerksamkeit & Kreativität.*

Ping-Pong-Programmierung
Red-Green-Refactor
TDD



Code-Reviews: Ständig & implizit!

Pair Programming ist eine Review-Technik!

Feedback rechtzeitig.

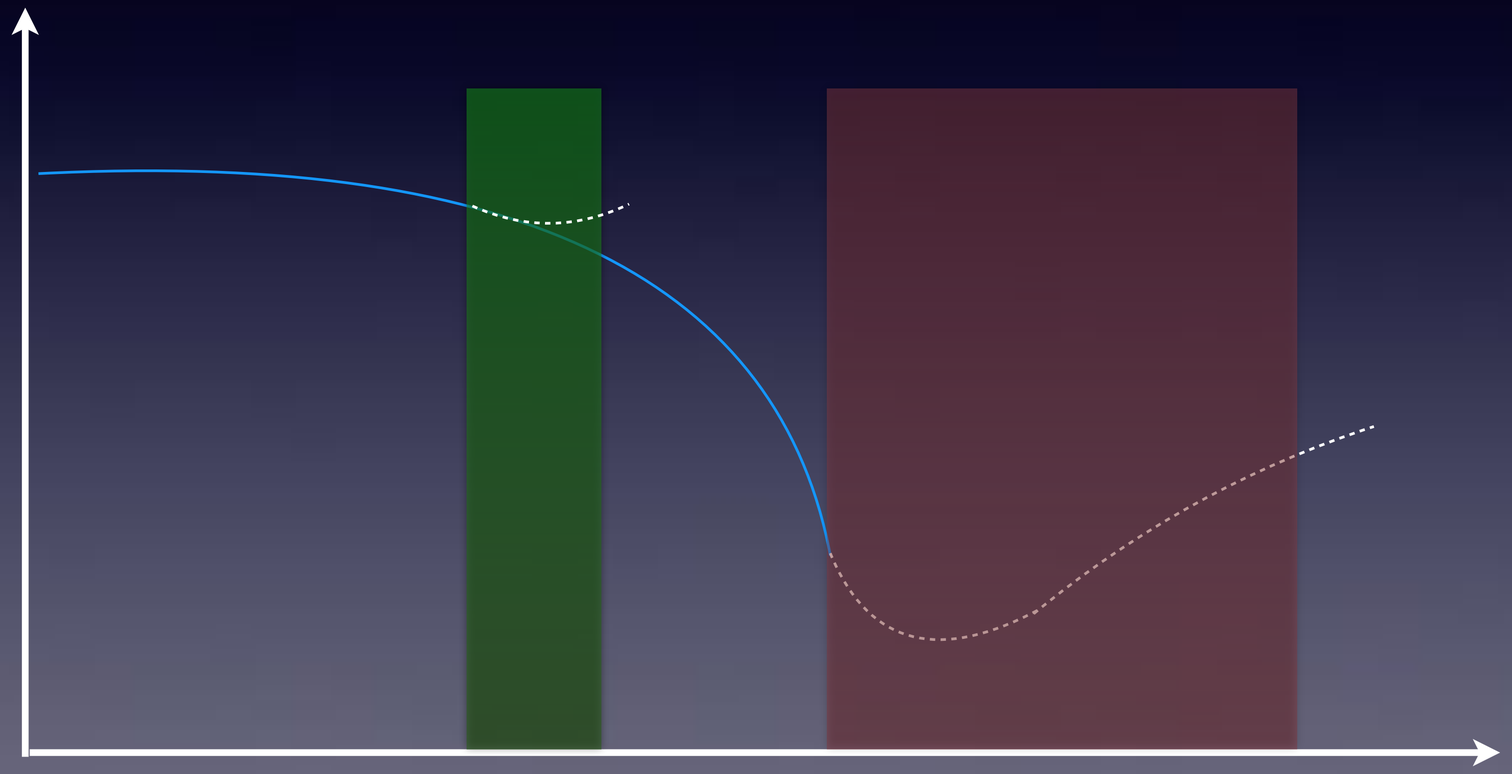
Auch für grundlegende Änderungen.

Explizite Code-Reviews: Optional.

Bei Arbeit im Pair kein Review mehr nötig.

(Kann aber optional eingefordert werden.)

Aufmerksamkeit & Kreativität



Effizient Pausen machen

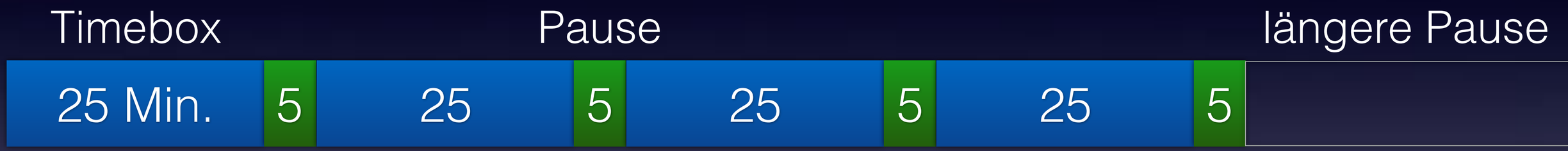
Bevor die Aufmerksamkeit zu stark sinkt.

Z.B. mit „Pomodoro“:

Kreativ-Zeitmanagement-Technik.

https://de.wikipedia.org/wiki/Pomodoro-Technik#/media/File:Il_pomodoro.jpg

Effizient Pausen machen



Fokussiert an der Aufgabe arbeiten!

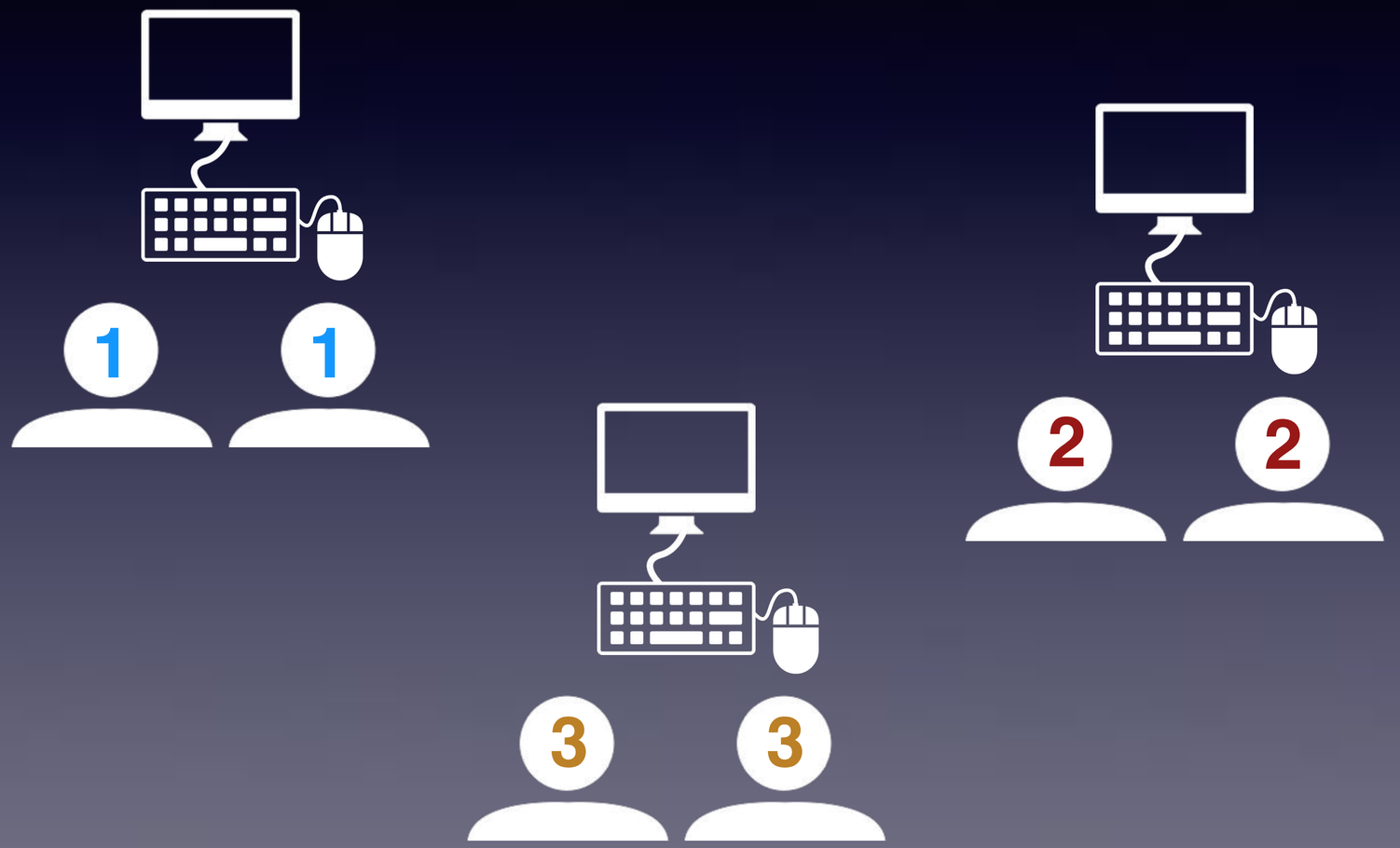
Einhalten!

Timer-App benutzen!

Pair-Wechsel?

Wissensinseln

Wissensinseln 2.0



Mind. 1x pro Tag

Pair-Rotation!

Wer mit wem?

Jeder mit jedem!

Sparrings-Partner

Experte & Experte

Know-How-Transfer.
Beginner’s Mind!

Experte & Einsteiger

Einsteiger & Einsteiger

Projekt entdecken.
Schwachstellen
aufdecken.

Und der Coach?

Coach ist Experte (methodisch, manchmal technisch)

Coach ist Einsteiger (fachlich, oft technisch)

Realistische Mitarbeit!

Akzeptanz

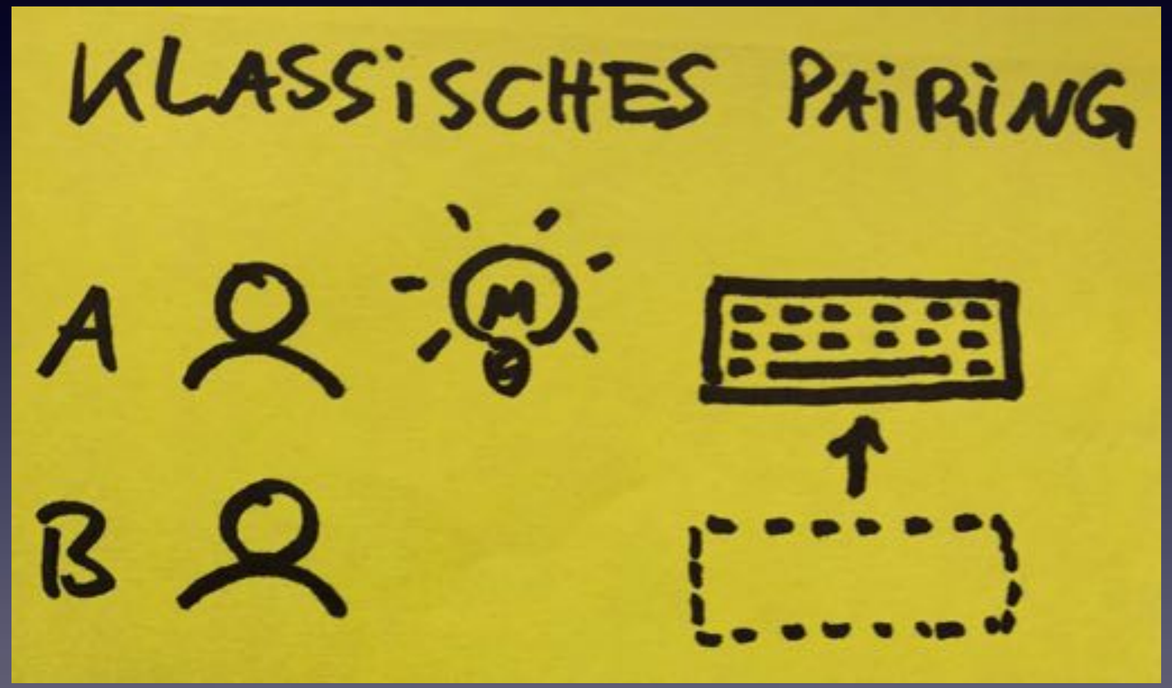
Der Coach …

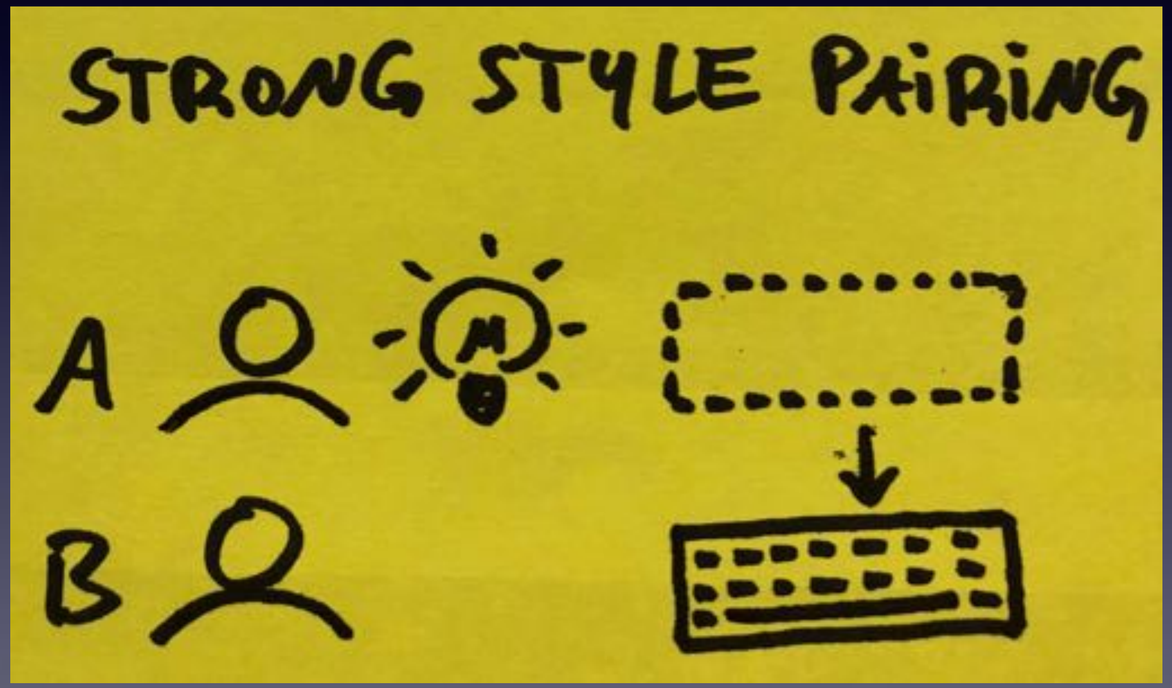
Pairt mit

Beobachtet die Pairs

Übt mit dem Team:
Rollenwechsel. Pair-Rotation. Pausen. Pairing-Varianten.

Pairing-Varianten





@LlewellynFalco

https://twitter.com/thmuch/status/959456902877974528

Remote Pairing

Am besten mit Offline-Erfahrung!

Tools:
Floobits Editor-/IDE-Plugin, AWS Cloud 9 etc.
TeamViewer, appear.in, Tuple.app etc.

Ausprobieren. Hängt viel vom Netzwerk (Proxies etc.) ab.

Gute Vorbereitung muss sein

Gemeinsame Vorbereitung geeigneter, kleiner Tasks.

Discovery, Planning, …

Durch Pair Programming erkennt das Team hier oft „Luft nach Oben“

Zusammenarbeit geht viel weiter!

Rollenübergreifend:
Dev, QS, UX, …

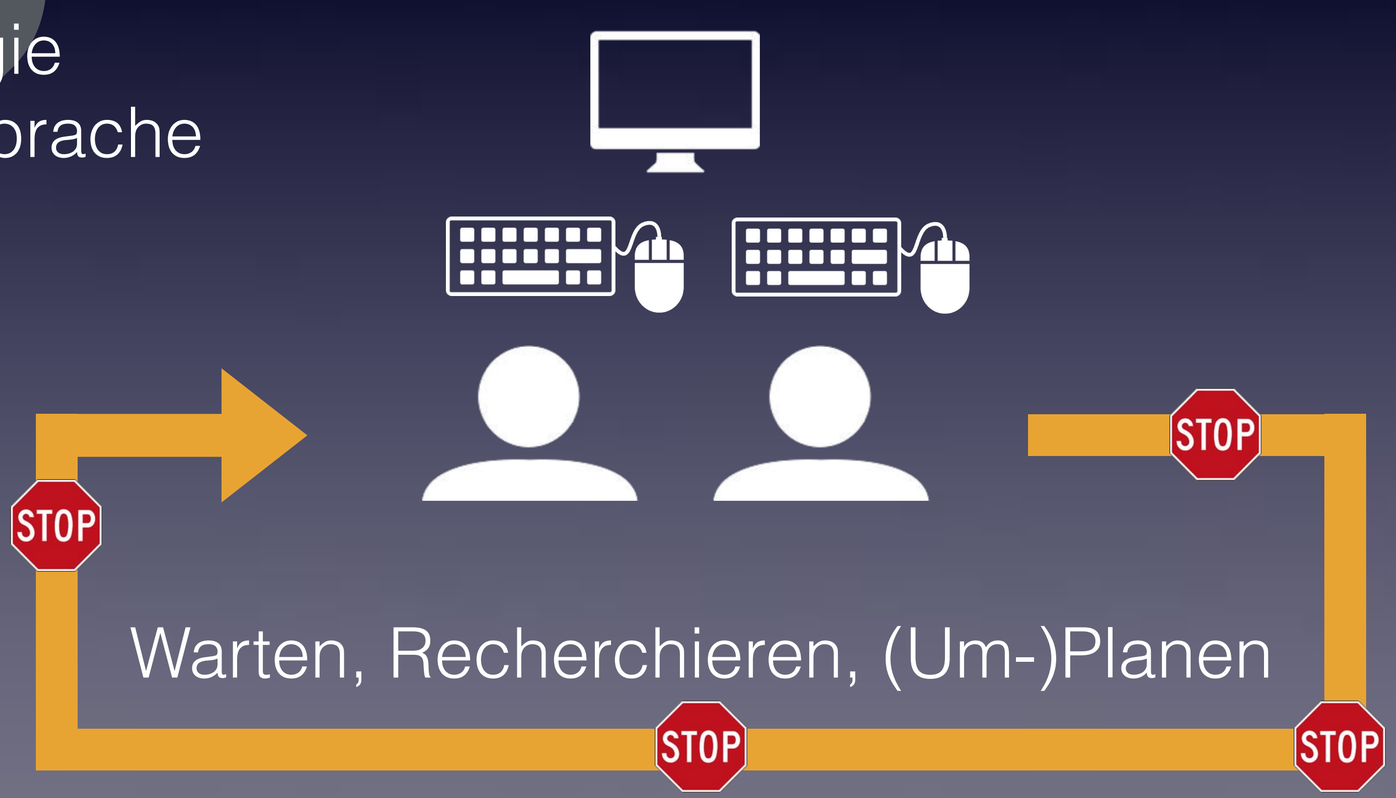
Pair Doing – „Pair on Everything“

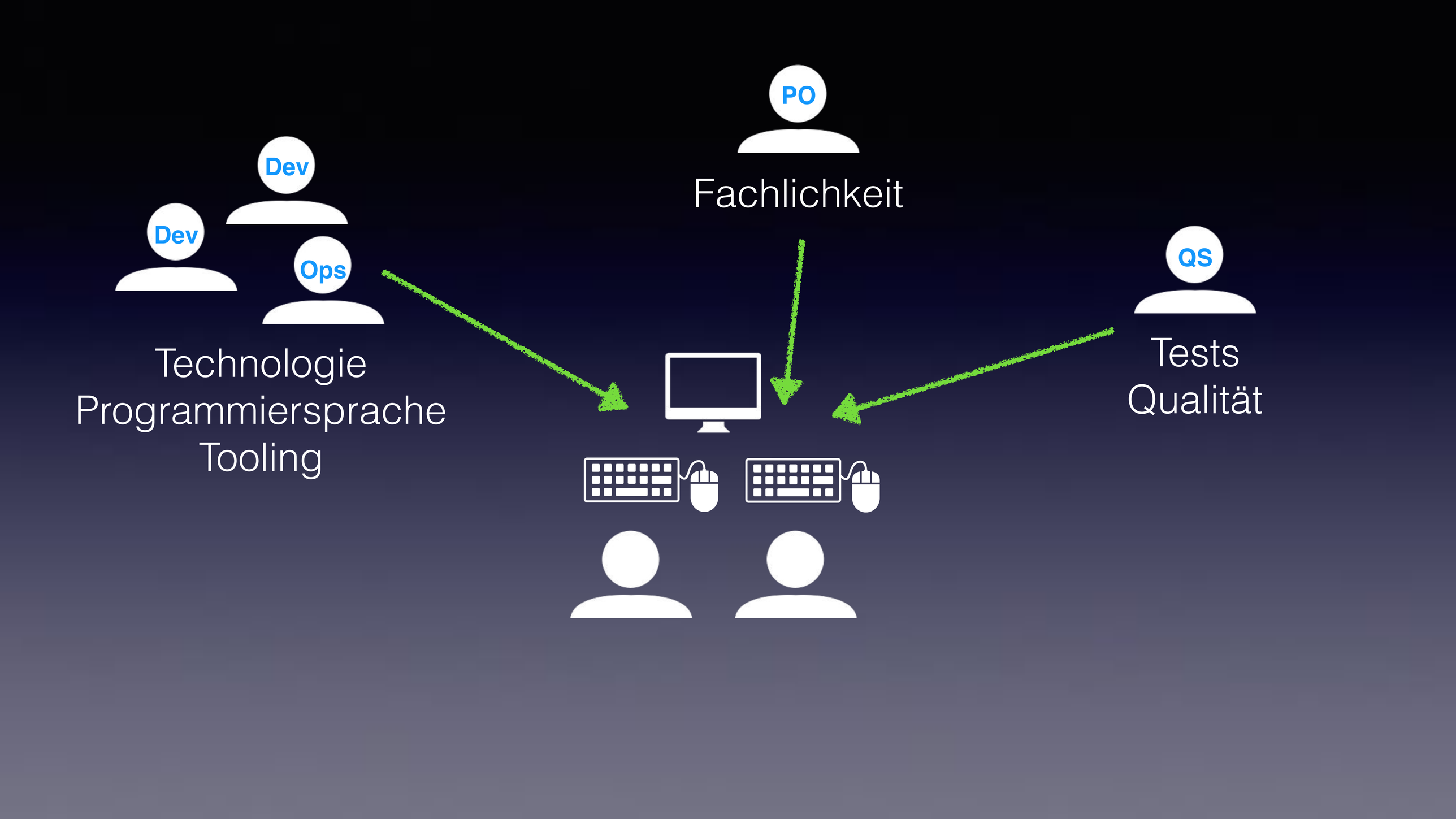
Perspektivwechsel

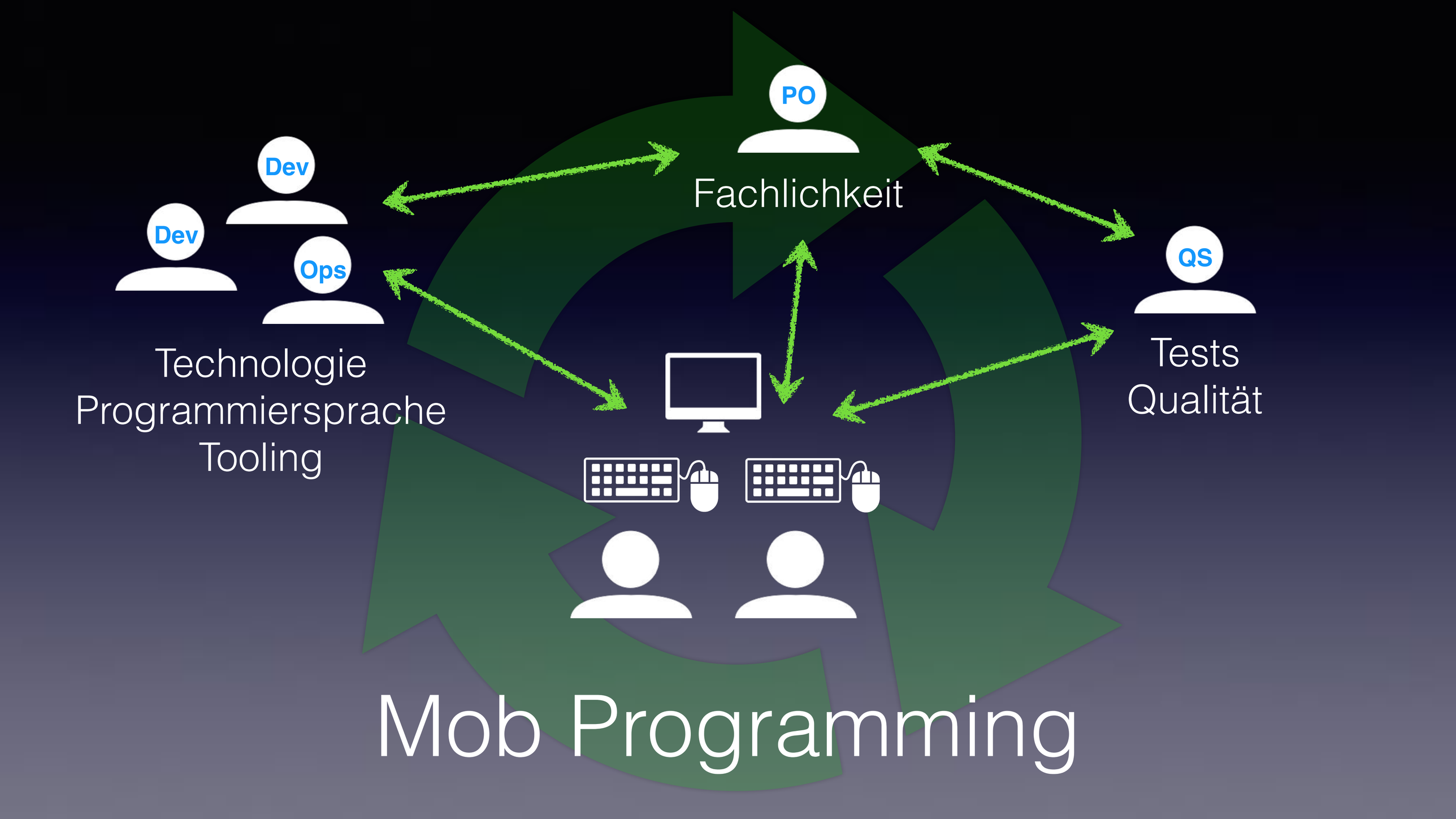
Fachlichkeit

Technologie
Programmiersprache
Tooling

Tests
Qualität




PO
Fachlichkeit
Dev
Dev
Ops
Technologie
Programmiersprache
Tooling
QS
Tests
Qualität



Mob Programming

Mob Programming

„It’s about getting the **BEST** (not the **most**) from your team.“

– Llewellyn Falco

„All the brilliant minds working on the same thing,
at the same time, on the same computer.“

„Continuous Integration of Ideas“

– Woody Zuill

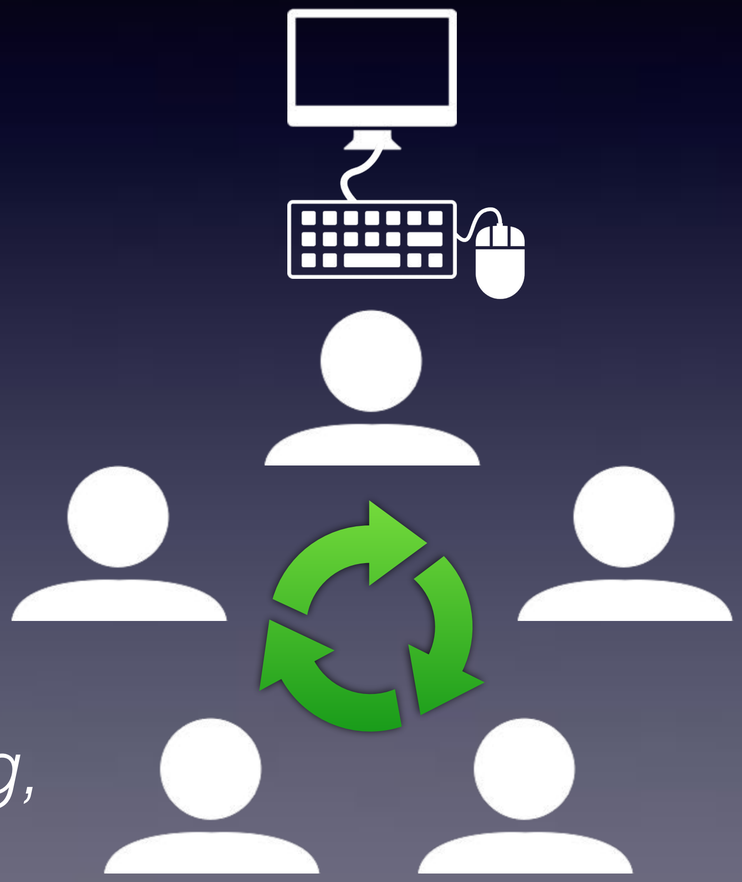
Mob Programming

Rollenwechsel!

Wechsel nach fester Timebox
(z.B. 10 Min.), *http://mobster.cc*

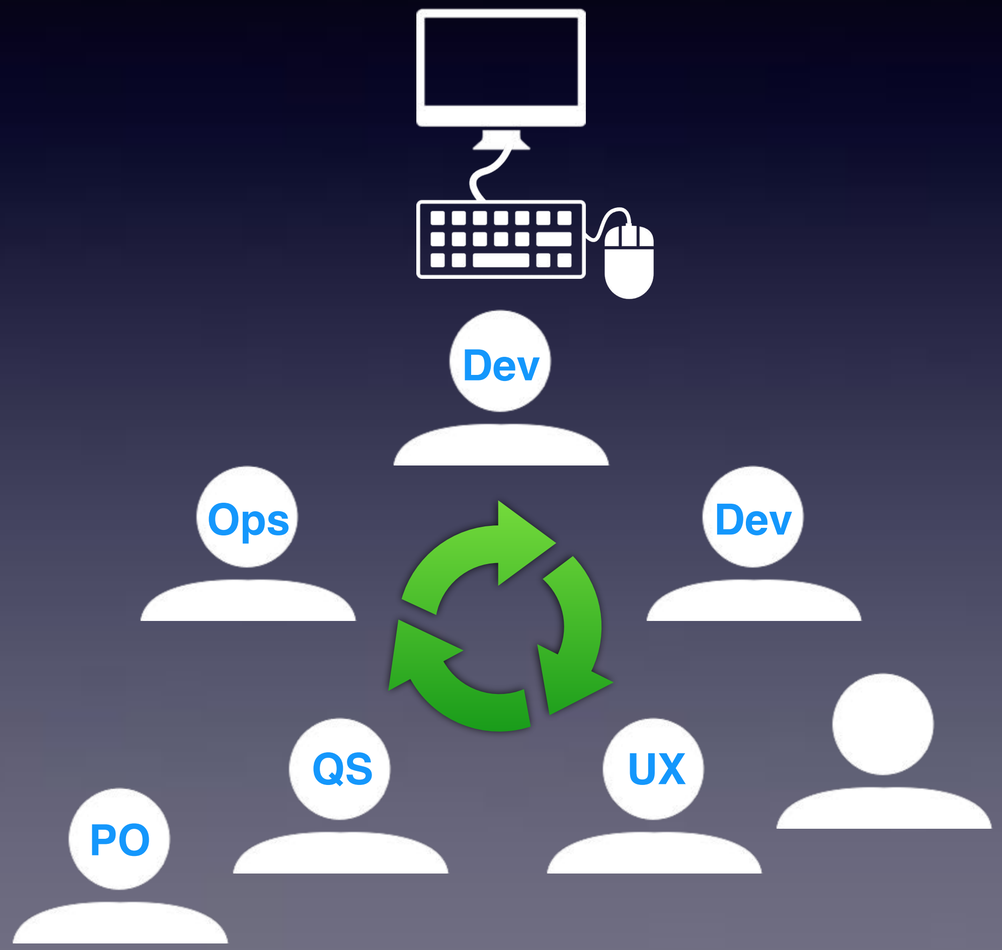
Dynamischer Mob:
Kommen und gehen.

Evtl. angenehmer als Pair Programming,
da nicht so „eng“.



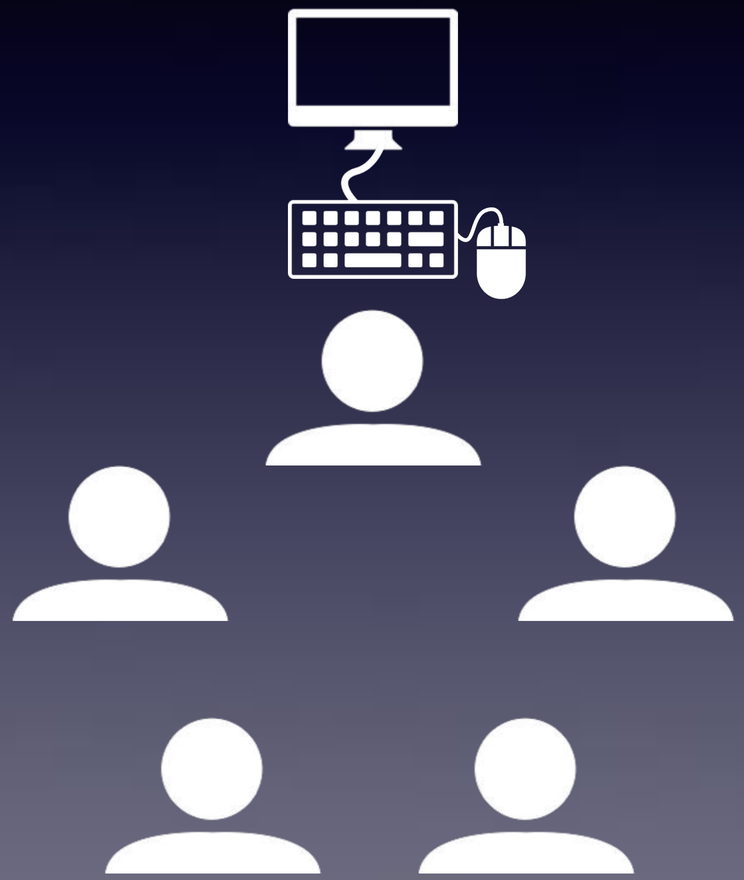
Mob Programming

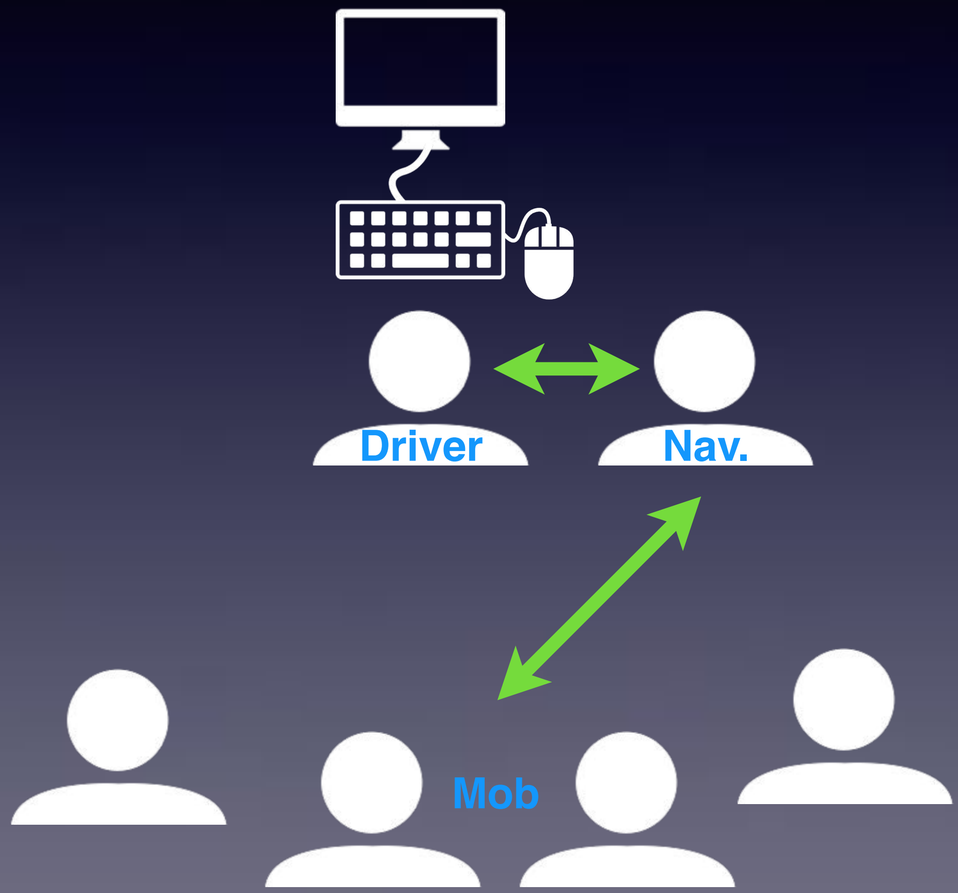
Auch rollenübergreifend!



Die **wichtigste** Aufgabe
zügig und gut fertig bekommen.

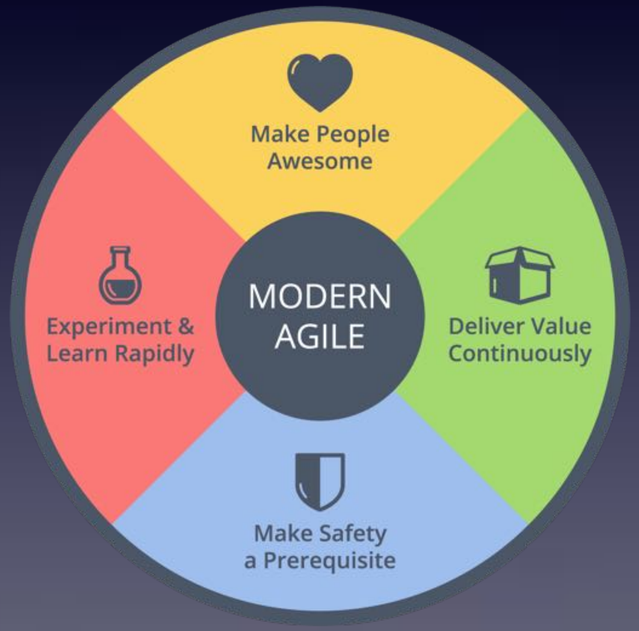
Mob Programming – Setups





Modern Agile

Agile – aber so, dass es funktioniert.



Pair- und Mob-Programming gehören fest dazu!

http://modernagile.org/

Und trotzdem …

„Aber alleine bin ich viel schneller.“

Bewusstsein schaffen

„If you want to go **fast**, go **alone**.

If you want to go **far**, go **together**.“

– Afrikanisches Sprichwort

Kleinigkeiten helfen

Ablehnung hat viele Gründe…

Kompromissbereitschaft bei Befürwortern und Ablehnern nötig.

Nutzen für das Team klarmachen!

Klare Abmachungen aussprechen.

z.B. „Kurzzeit-Pairing“

Kleiner Schritt, große Wirkung!

„Don't think of pair programming
as 2 people doing the work of one.

Think of it as 2 people avoiding the rework of 7.“

– Jason Gorman

Geschwindigkeit ist nicht alles

We follow these principles:

…

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a **constant pace** indefinitely.

…

http://agilemanifesto.org/principles.html

100% Pairing?

Eher nicht. Aber:

Muss normale Arbeitspraktik sein!

Darf keine Ausrede sein, Pair Programmin sen.

Wieviel % vom Tag co

Von der echten Coding-Zeit *Pairing nutzen!*

Solo-Zeit muss erlaubt sein!

Für Lernen von Neuem,
Lesen, Recherche etc.

Fazit

Pair & Mob Programming stärken agiles Vorgehen.

Coaching hilft, Pair & Mob Programming langfristig zu etablieren.

Wir haben gute Erfahrungen gesammelt!

Selber Best Practices *pro Team* herausfinden.

~~Üben, üben, üben.~~ Machen!

Agiles Coaching – wichtig!

Aber:

Coaching für Programmierpraktiken nicht vergessen.

Mob Programming

Pair Programming

Know-How-Transfer

Coaching

Pomodoro

Fragen?

XP

Lesbarkeit

Modern Agile

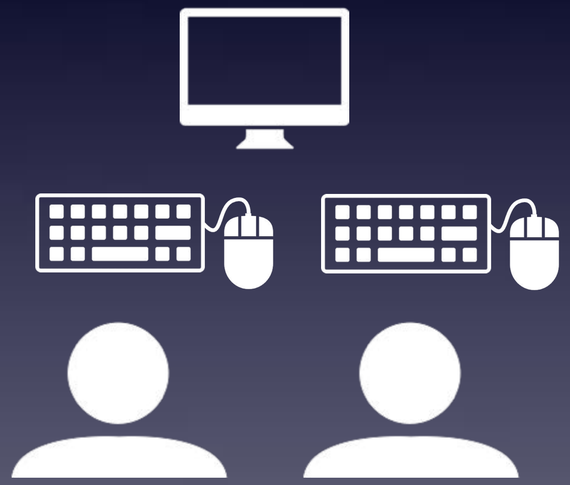
Einfachheit

Strong Style Pairing

Geschwindigkeit

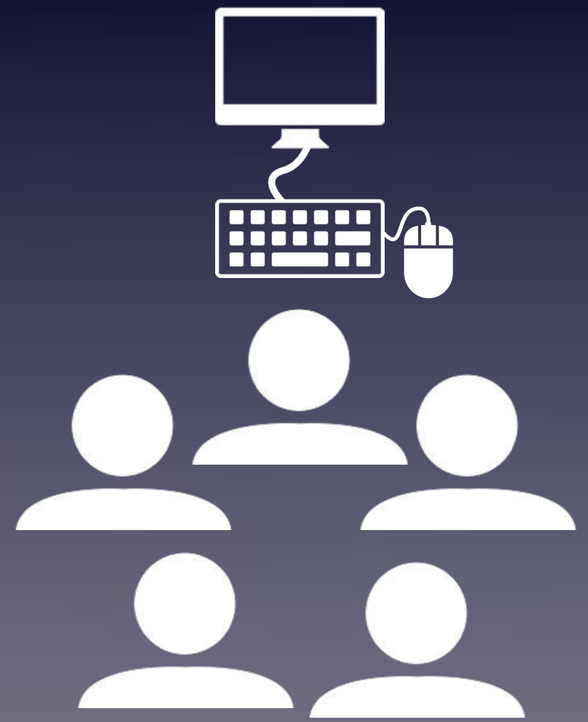
TDD

Collective Product Ownership



JAVA FORUM NORD

Danke!



thomas@muchsoft.com
www.javabarista.de
@thmuch