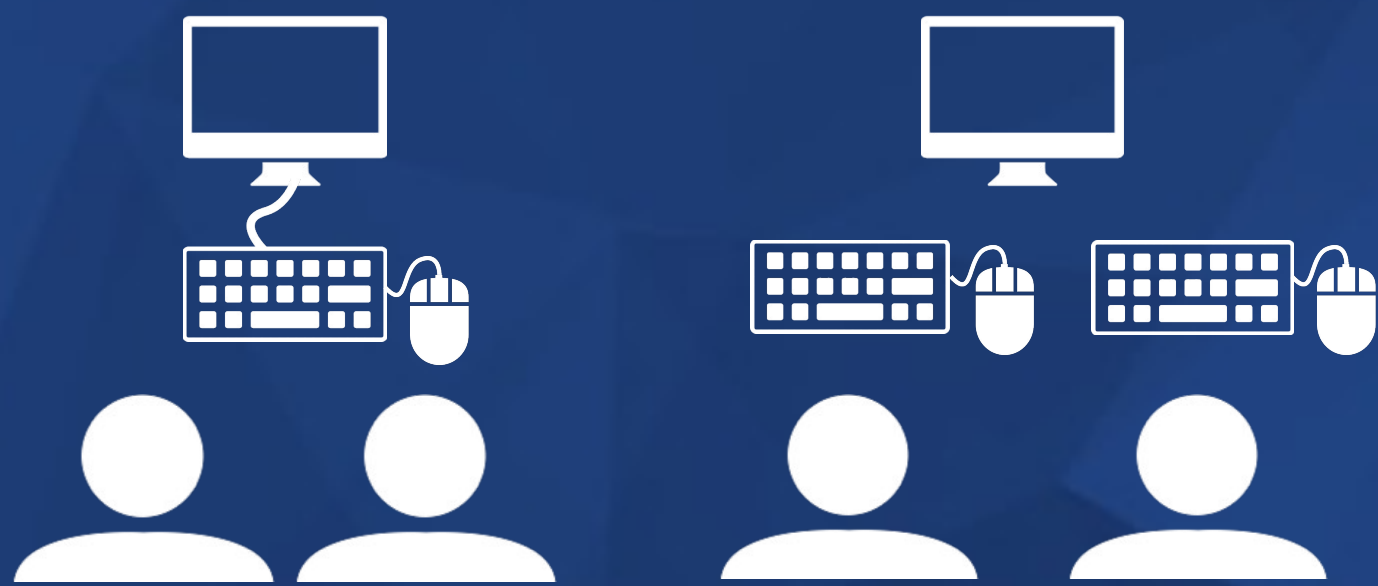


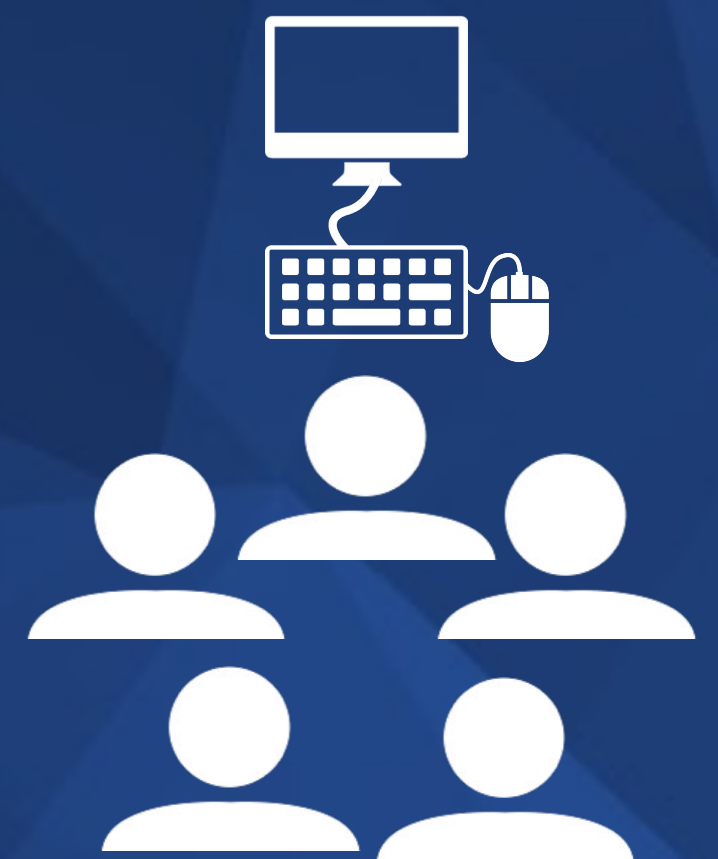


Pair- und Mob-Programming

Damit Agile erfolgreich wird und bleibt!



Thomas Much
 @thmuch



Über...



Thomas Much

Freelancer, Hamburg

 @thmuch
#JAX2018

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?“

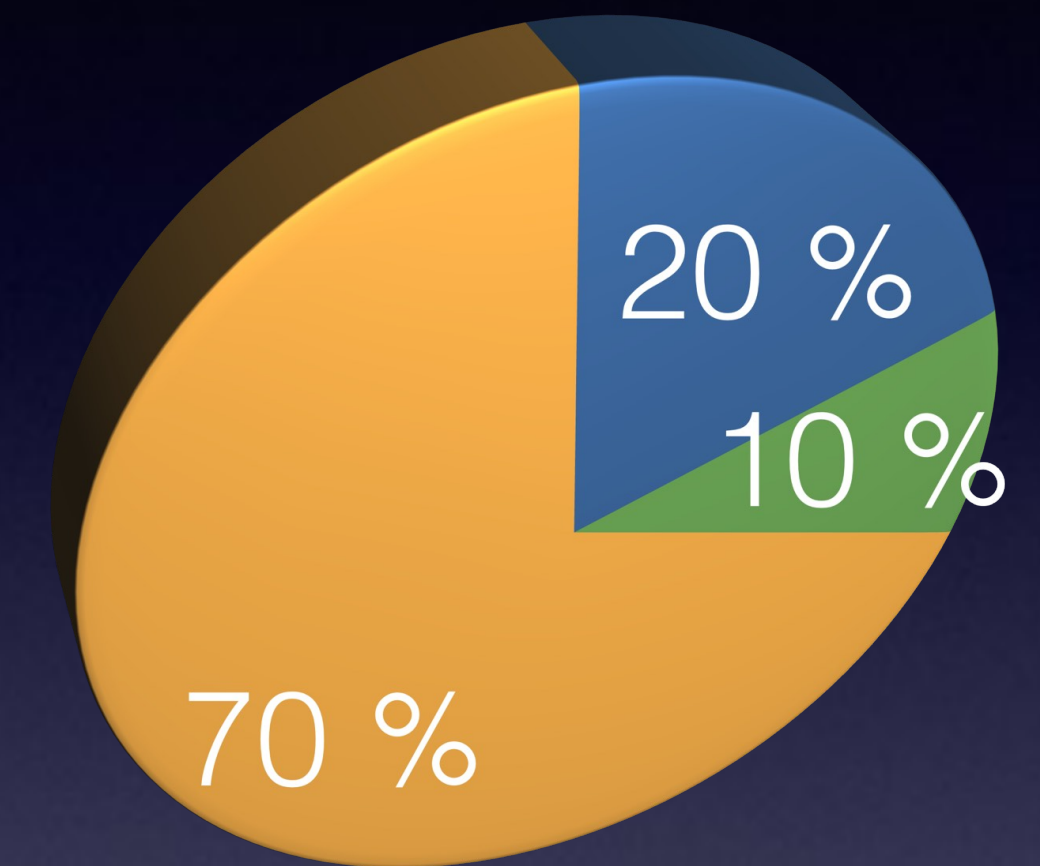
„Oh. Ich selber.“

„Das muss Kollege X machen,
er hat das mit seinem #!@%&\$!?! Stil programmiert.“

Problem: Lesbarkeit

- Problem lösen
- Code schreiben
- Code verstehen

- Code wird sehr viel häufiger gelesen als geschrieben
- Verständlichkeit essentiell 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

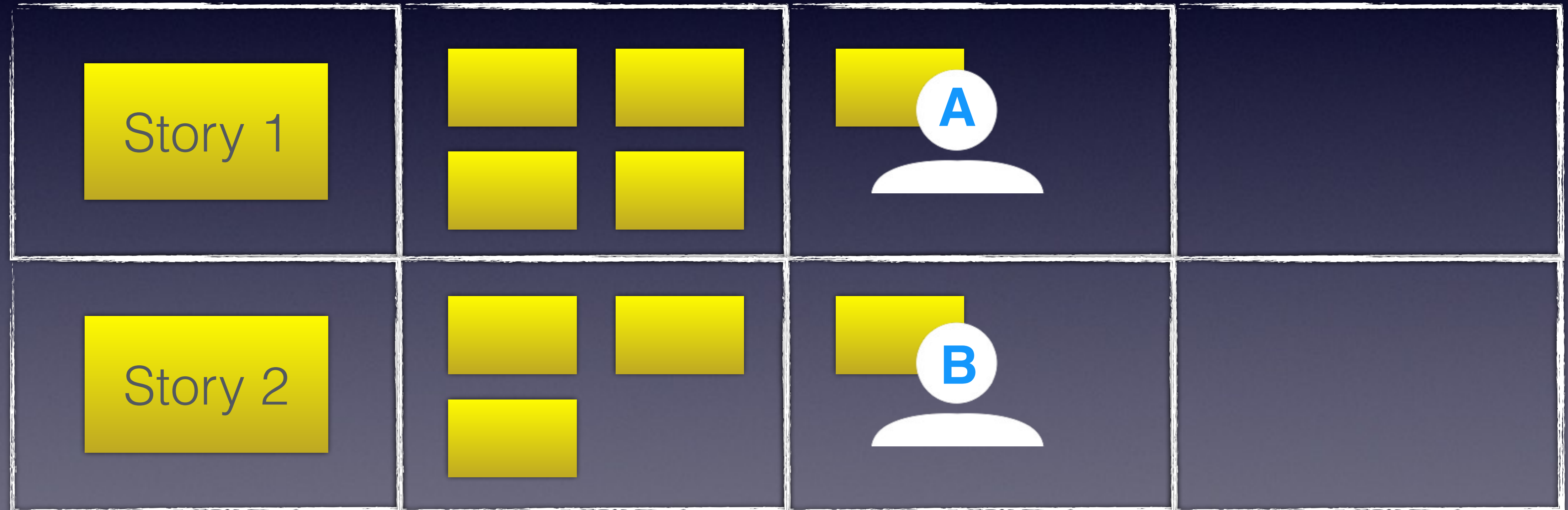


Lösung! „Wir werden agil.“

To Do

In Progress

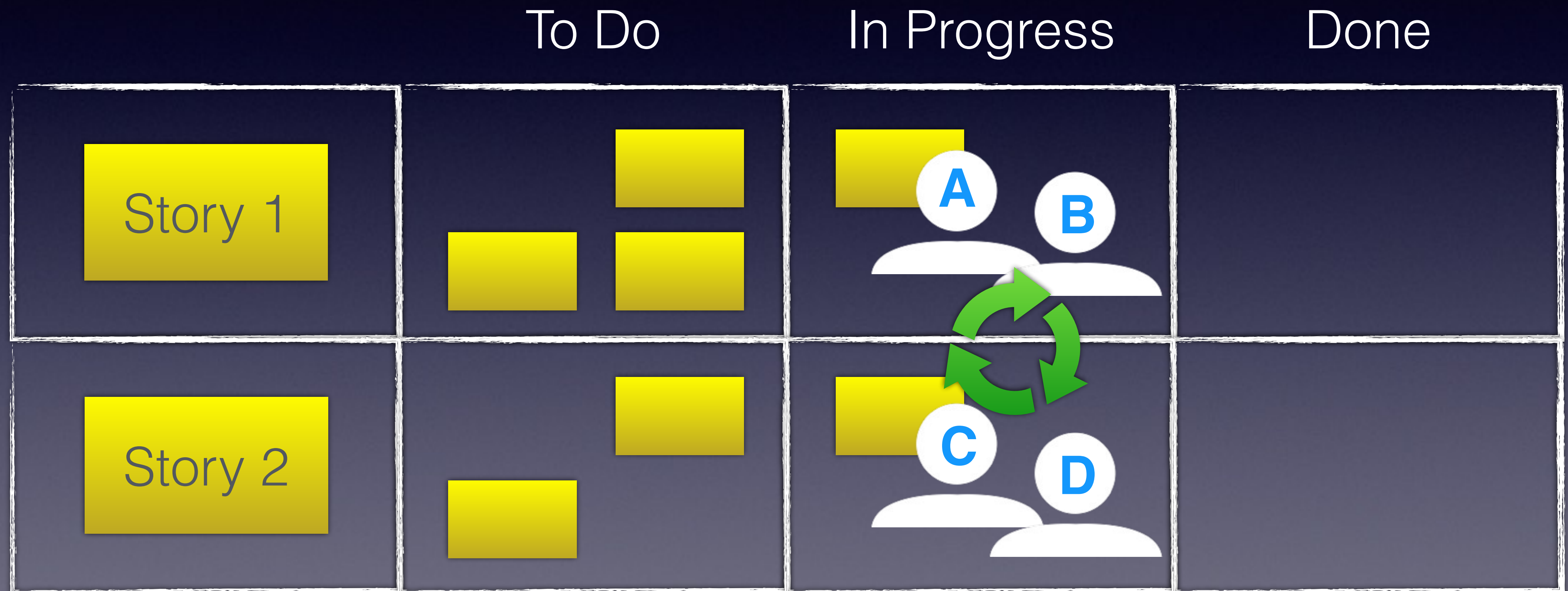
Done



„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 – möglichst quick, egal wie dirty?

(„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 Coaching

Idee: Pair Programming *aktiv* fördern.

Seit 2013 in zahlreichen Teams durch Coaching unterstützt.

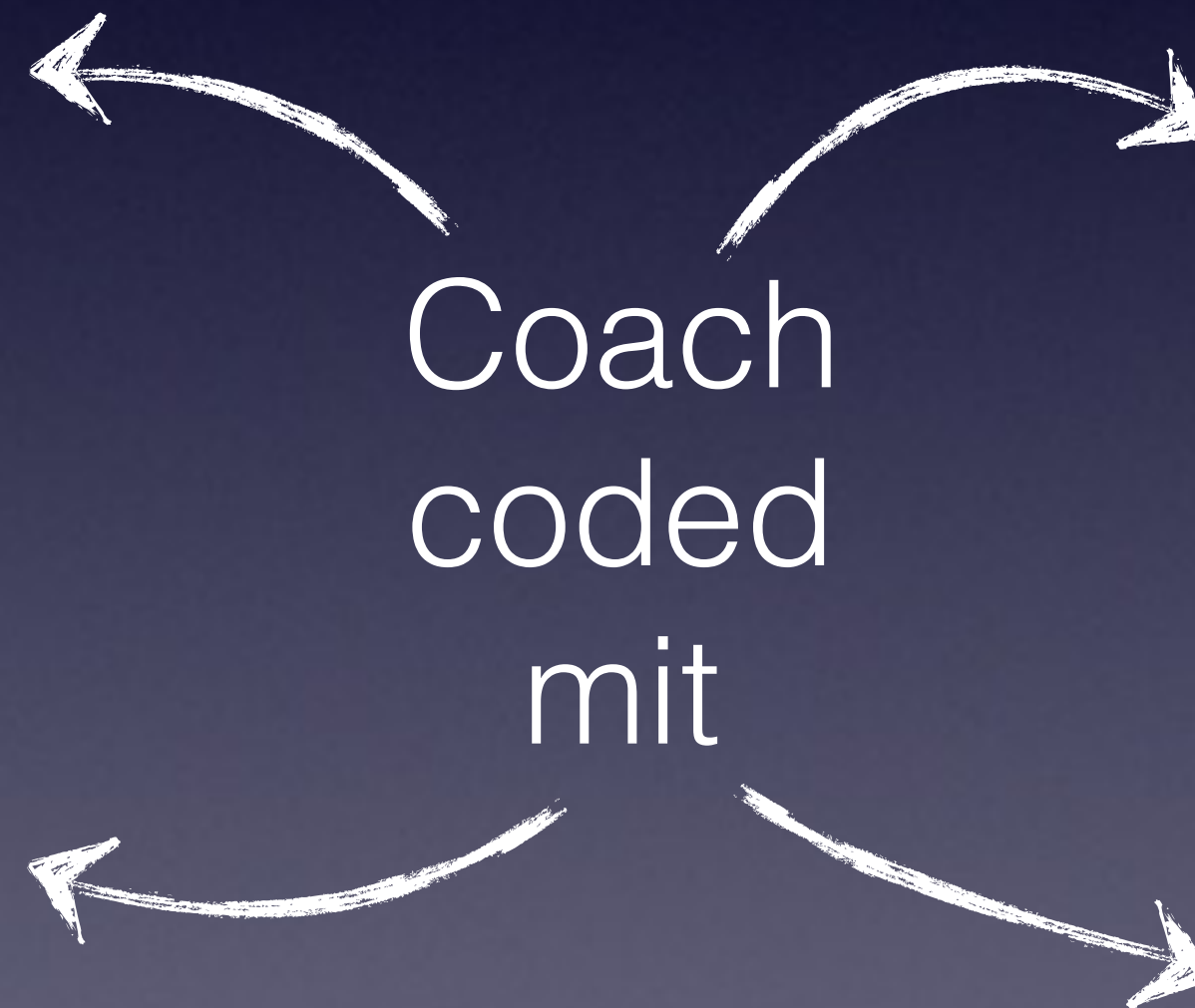
E-Commerce, BI, klassische Enterprise-Backends.

Coach begleitet Team 1-2 Sprints (2-4 Wochen).

Coach arbeitet soweit möglich als Entwickler mit.

Coaching-Ablauf

1/2 bzw.
1 Sprint

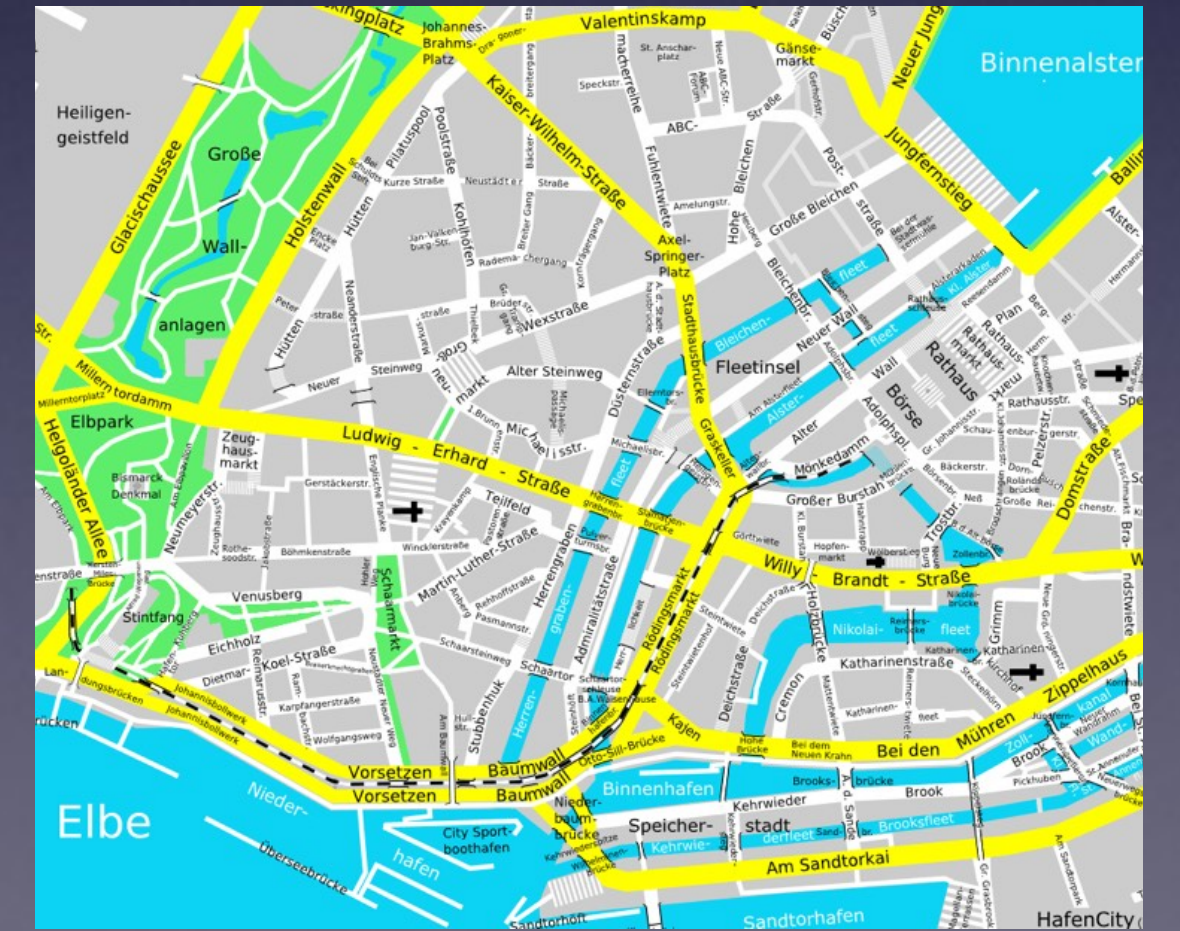


Pair Programming: Schnell erklärt

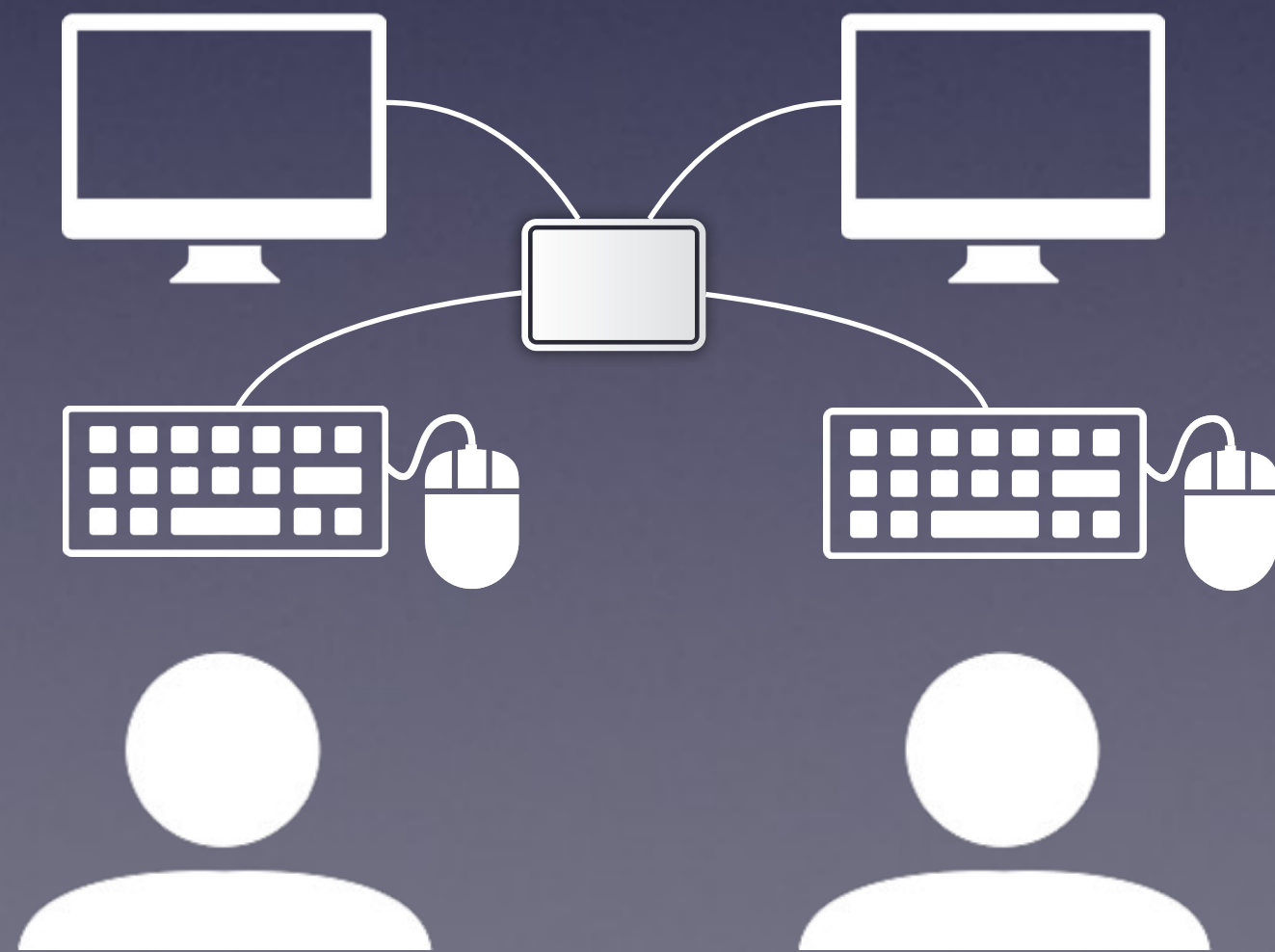
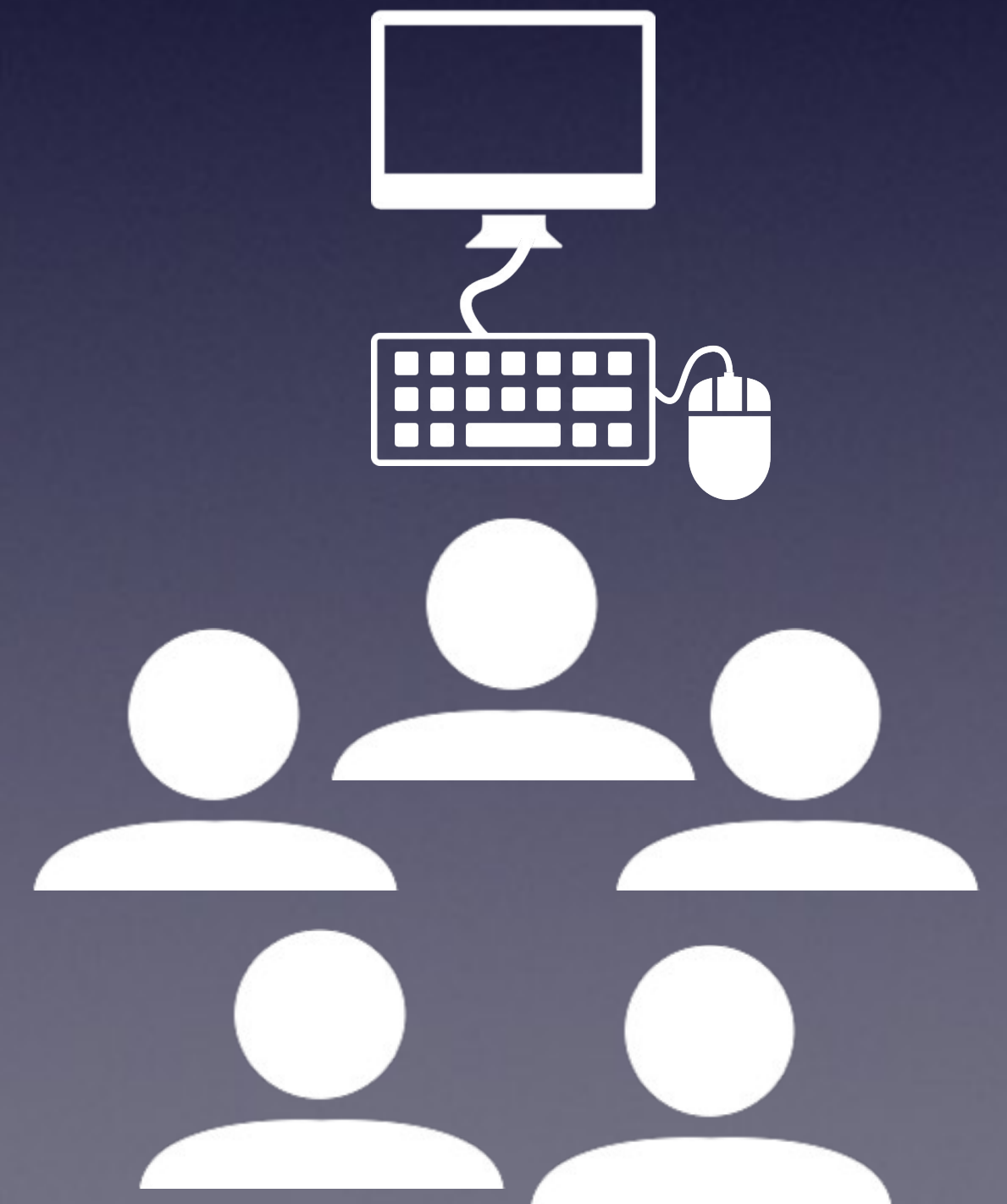
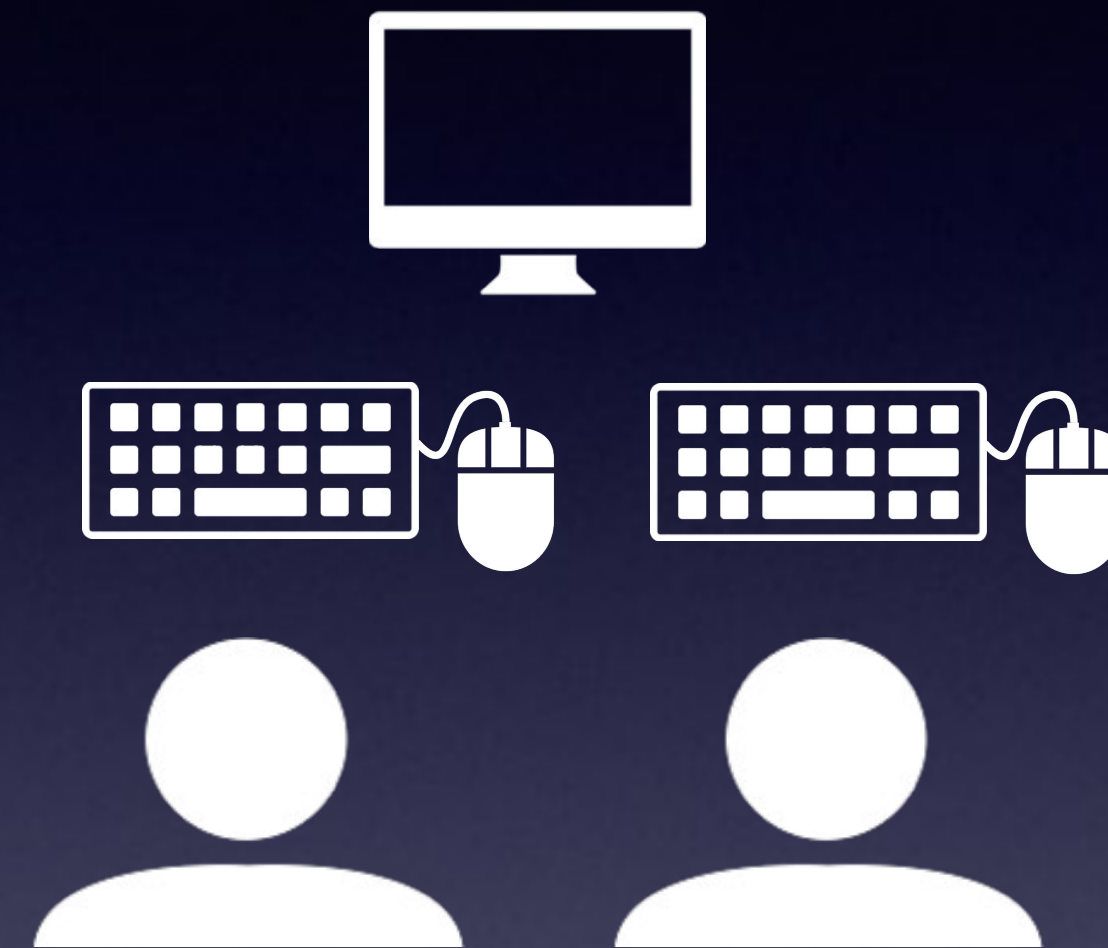
1
Aufgabe



Driver & Navigator



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“

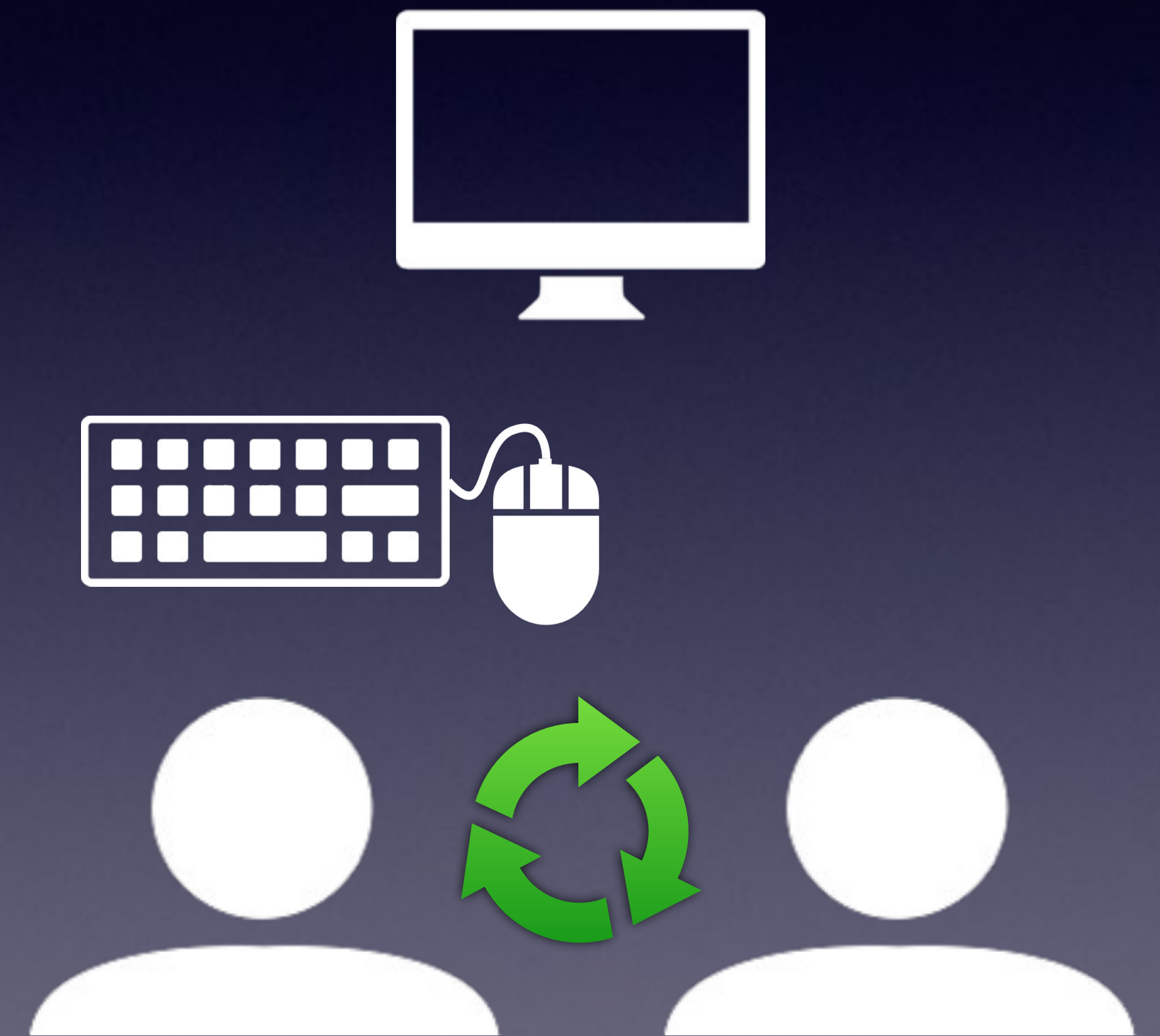
...

<https://www.smidig.de/2016/09/brain-patterns-in-der-softwareentwicklung/>
<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.

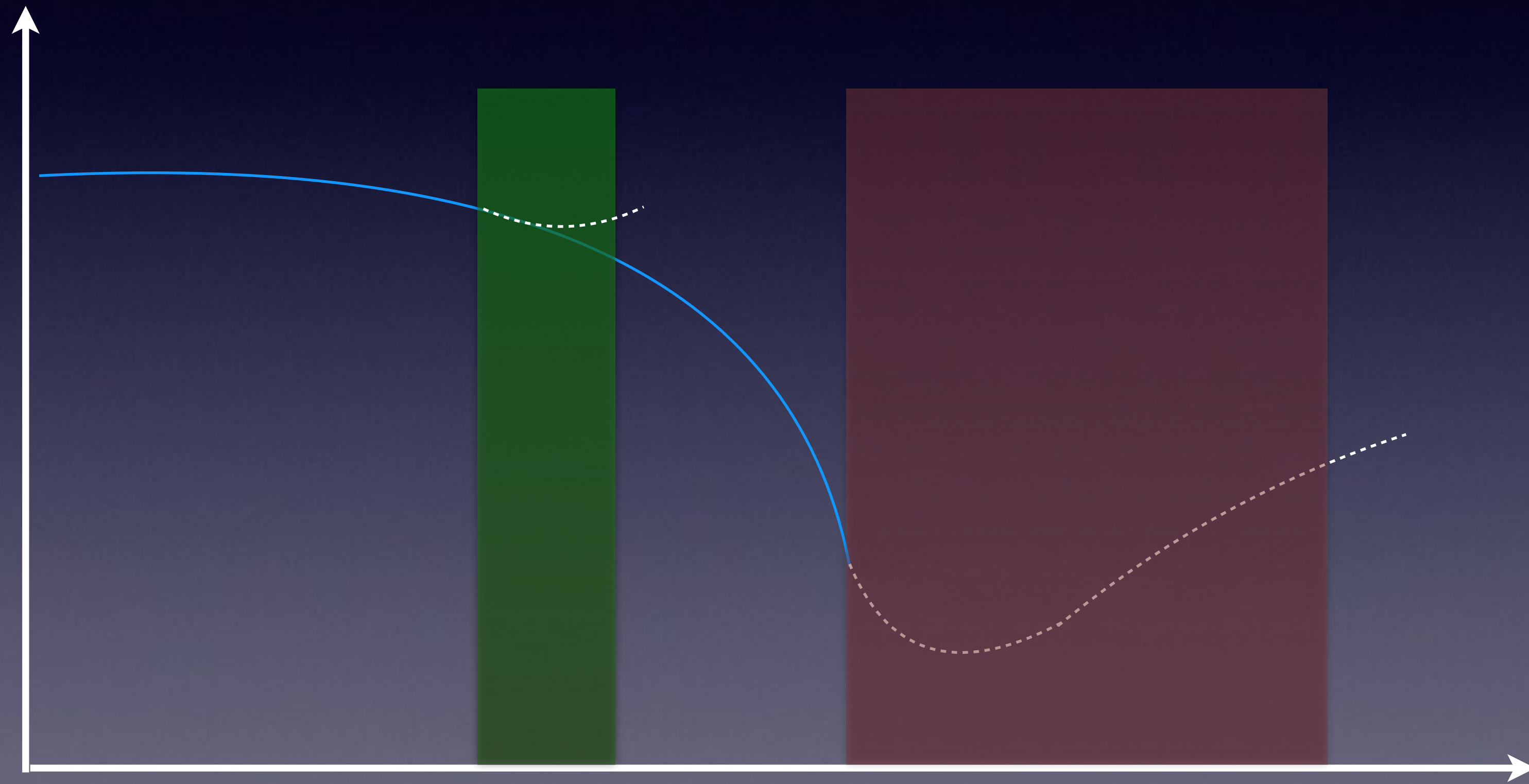
Auch für grundlegende Änderungen.

Explizite Code-Reviews: Optional.

Bei Arbeit im Pair kein Review mehr nötig.

(Kann aber optional eingefordert werden.)

Aufmerksamkeit & Kreativität



Effizient Pausen machen

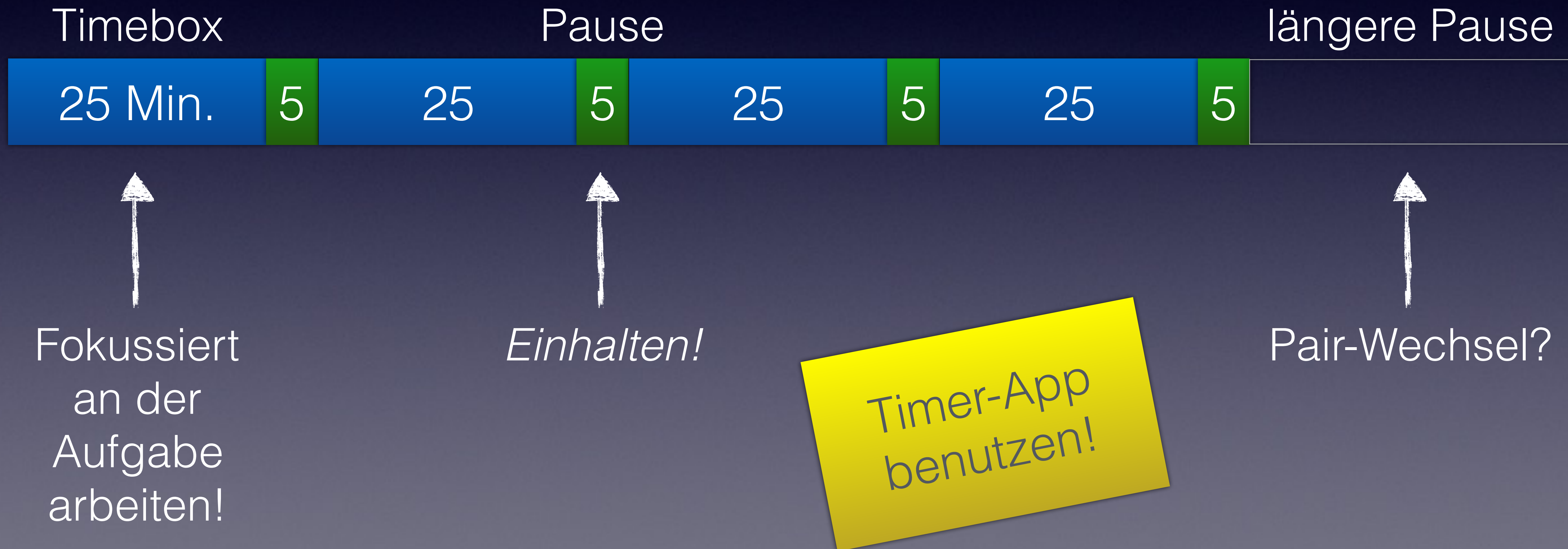
Bevor die Aufmerksamkeit zu stark sinkt.

Z.B. mit „Pomodoro“:

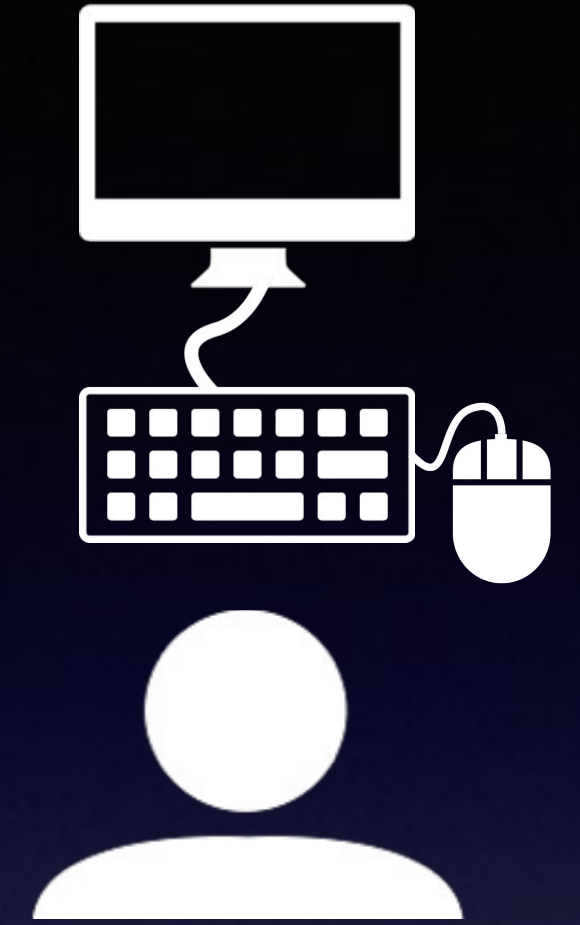
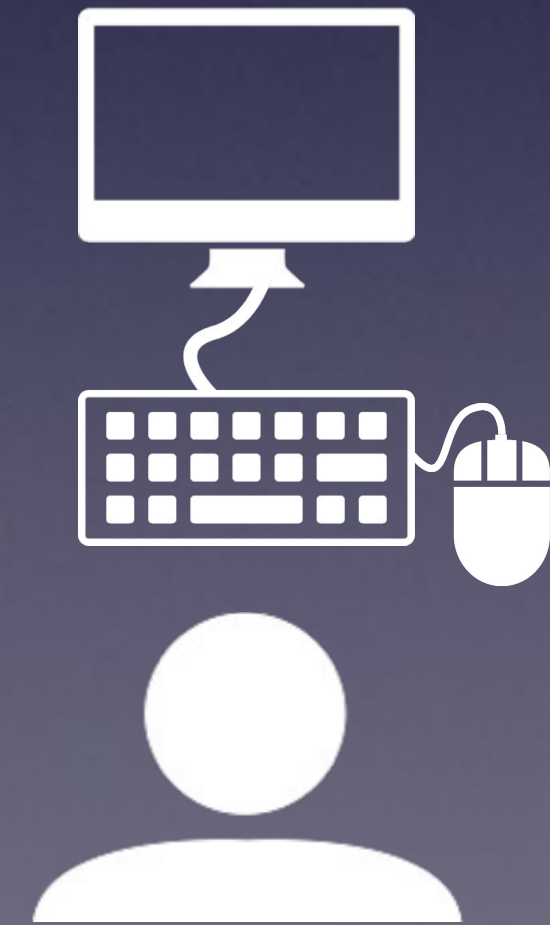
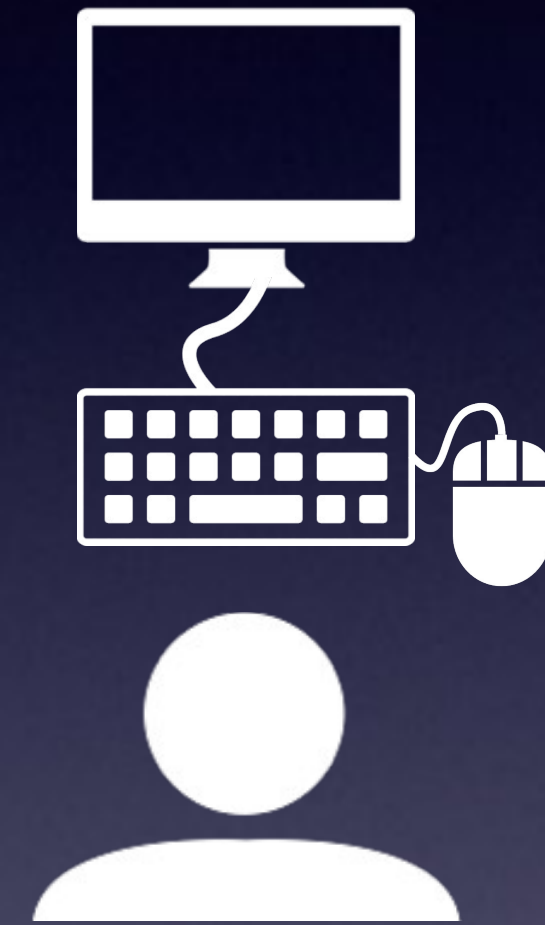
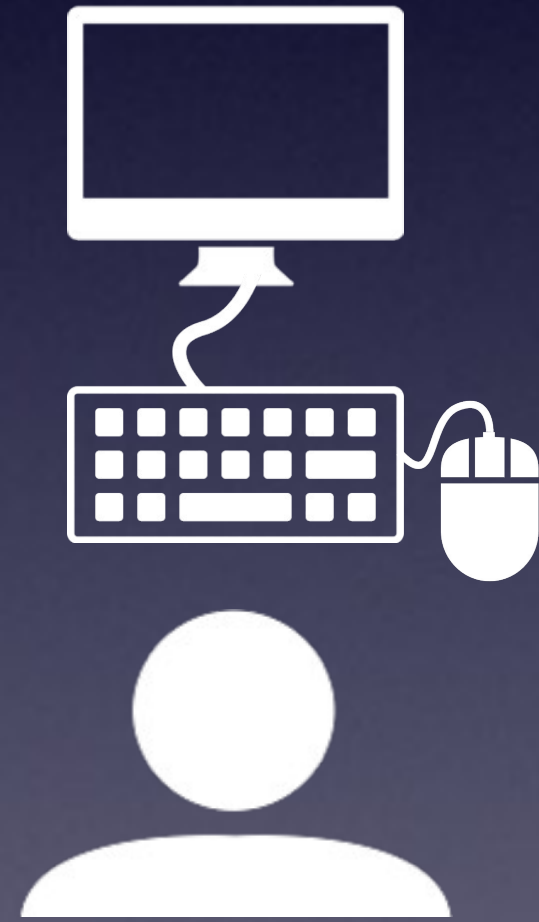
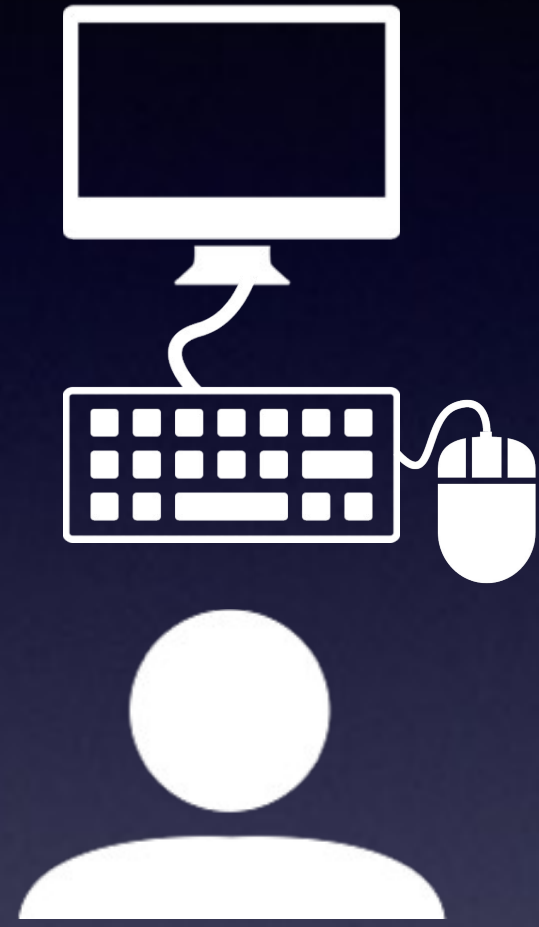
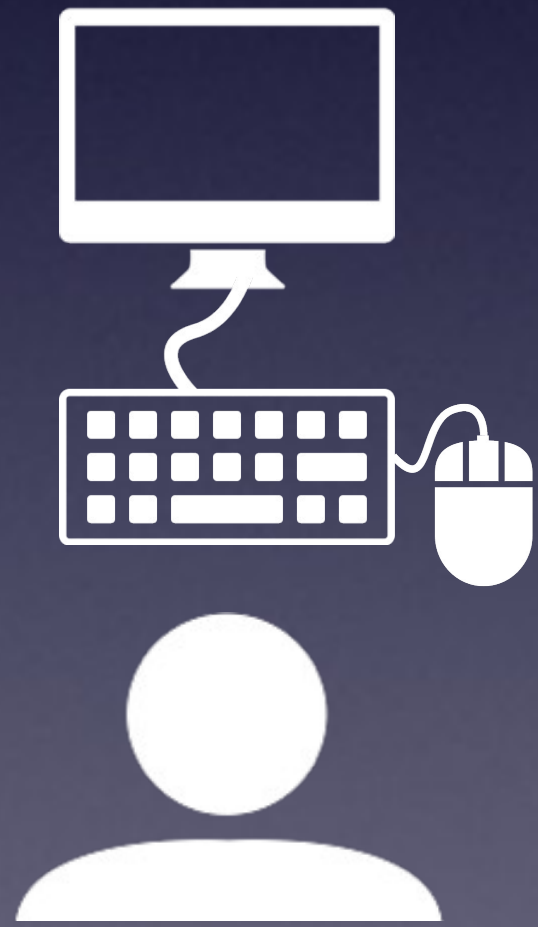
Kreativ-Zeitmanagement-Technik.



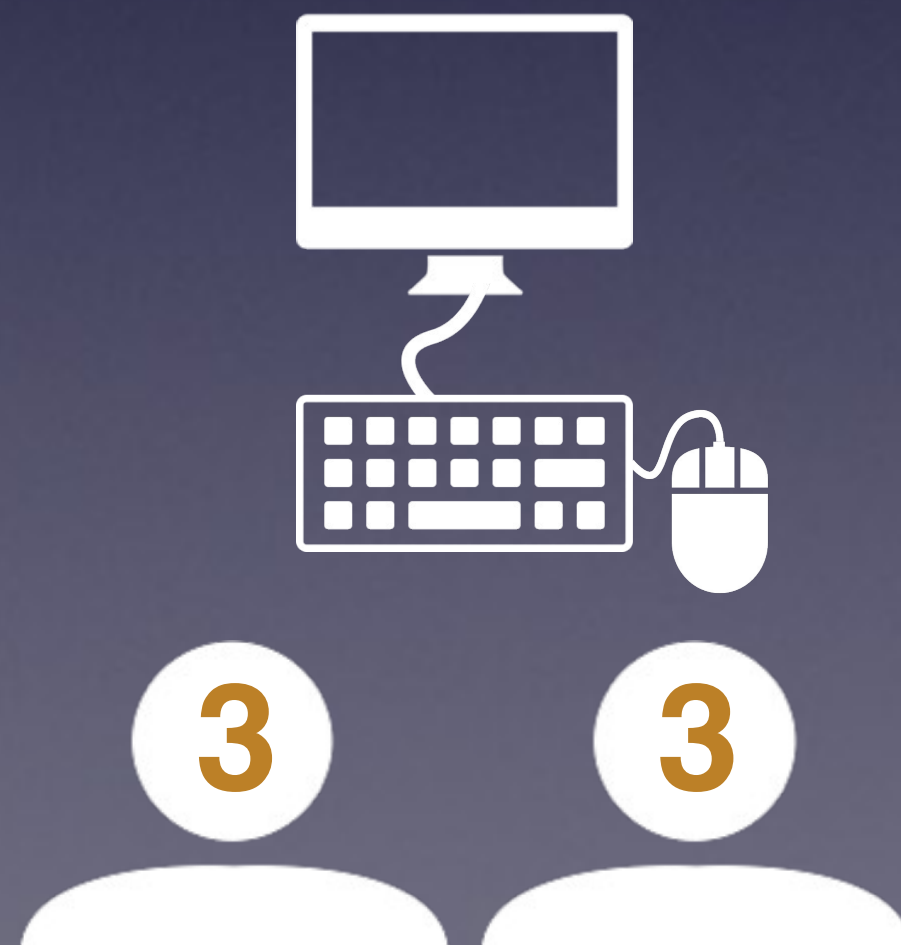
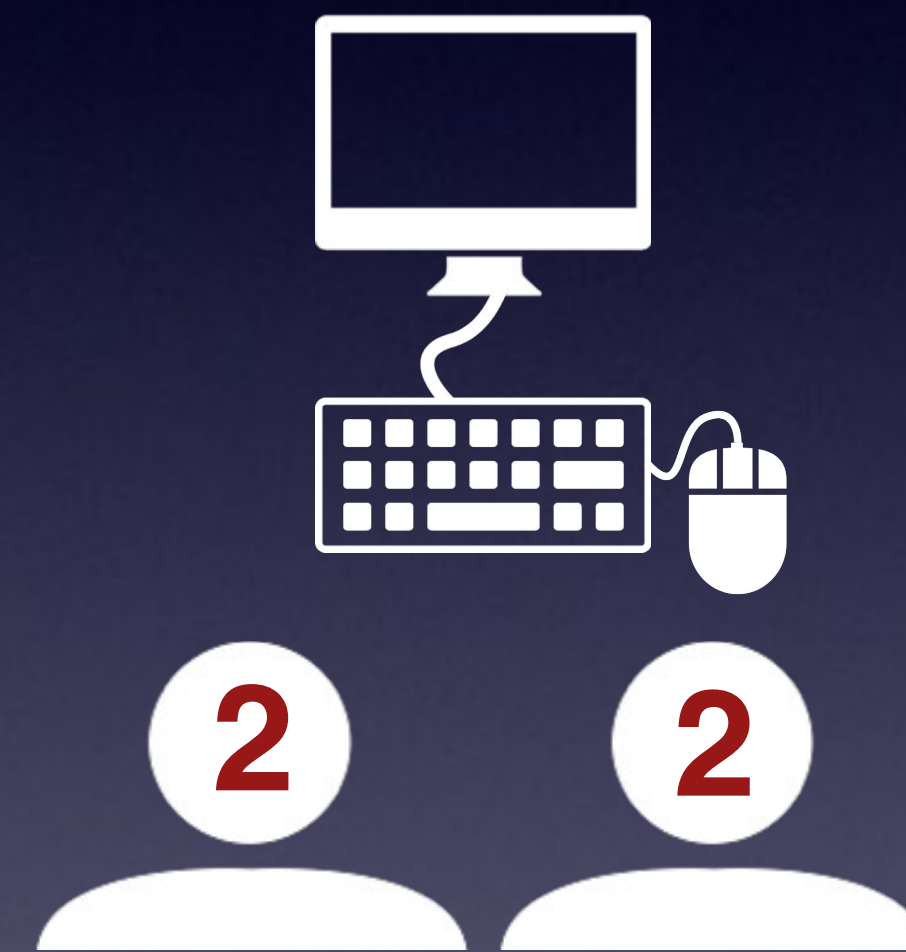
Effizient Pausen machen



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:

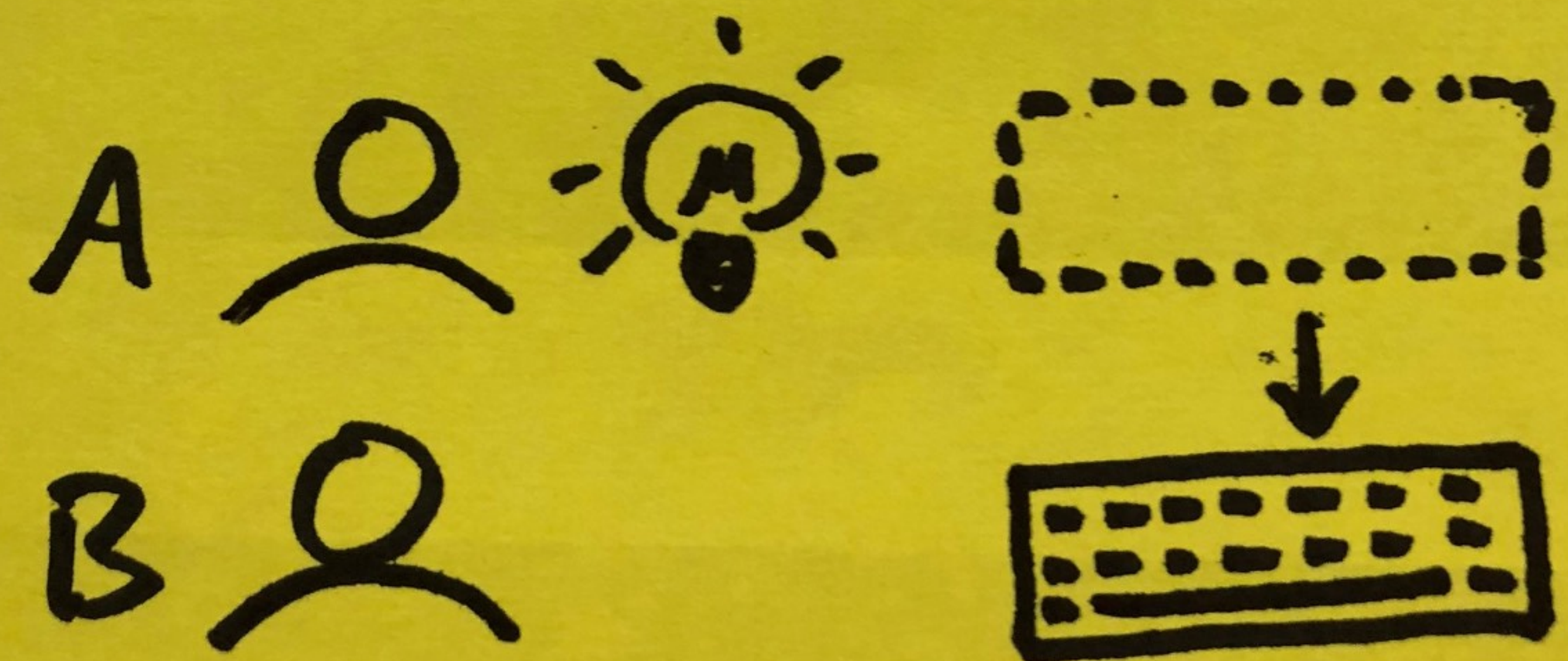
Rollenwechsel. Pair-Rotation. Pausen. Pairing-Varianten.

Pairing-Varianten

KLASSISCHES PAIRING



STRONG STYLE PAIRING



@LlewellynFalco

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!

Geschwindigkeit

Gefühlt langsamer – aber weniger Nacharbeiten?!

Schneller „Done“ – da weniger Rückwärtsbewegung von Tasks?!

So langsam sein, dass man das Richtige richtig machen kann.

„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 Programming ist ein Werkzeug, das eingesetzt werden kann.

Wieviel % vom Tag coden?

Von der echten Coding-Zeit

Solo-Zeit muss erlaubt sein!

Für Lernen von Neuem,
Lesen, Recherche etc.

Pairing nutzen!

Beyond Pair Programming

Zusammenarbeit geht viel weiter!

Rollenübergreifend:
Dev, QS, UX, ...

Pair Doing – „Pair on Everything“

Perspektivwechsel

Mob-Programming

Mob Programming

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

– Llewellyn Falco
– Woody Zuill

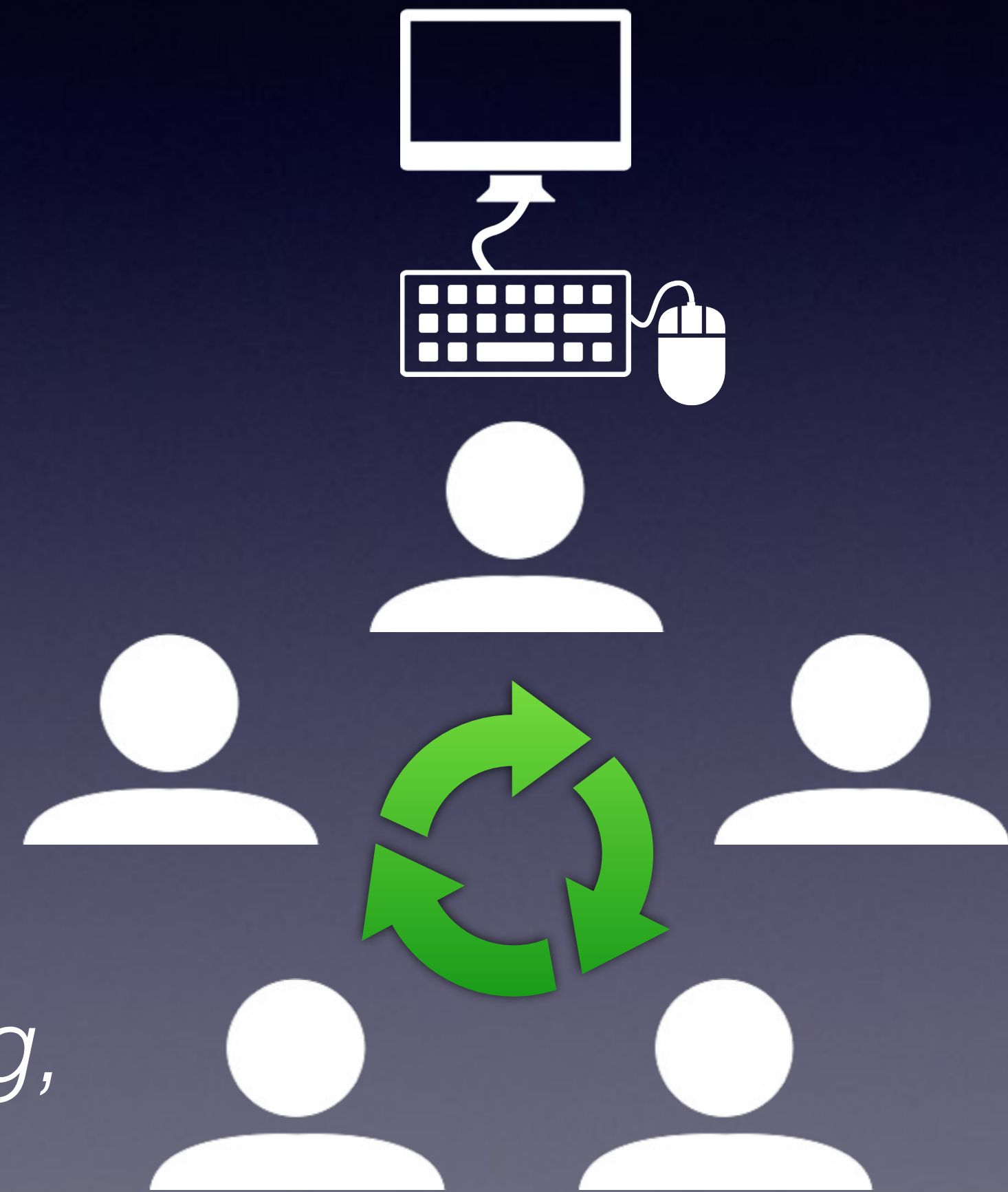
Mob Programming

Rollenwechsel!

Wechsel nach fester Timebox
(z.B. 10 oder 15 Min.)

*Dynamischer Mob:
Kommen und gehen.*

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



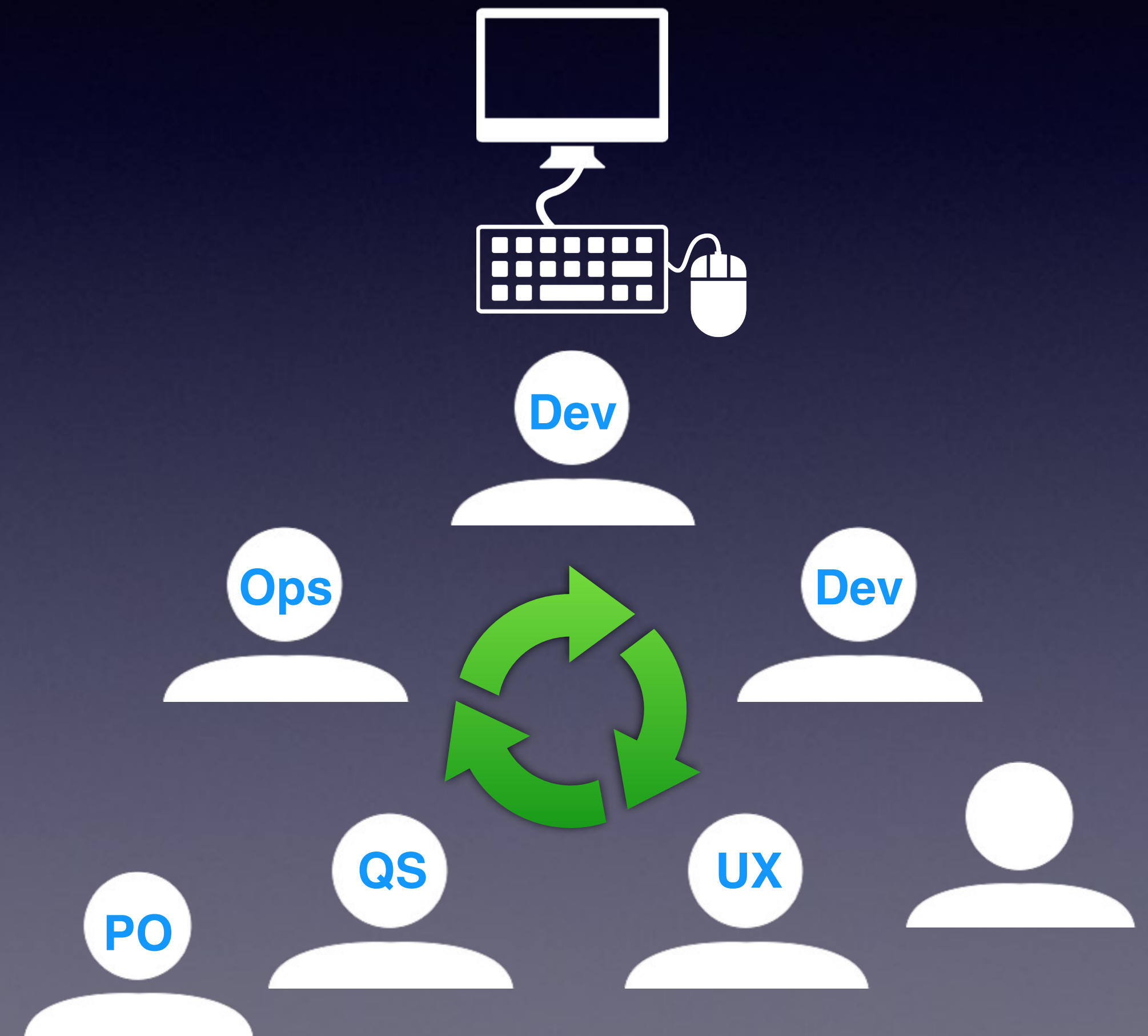
Mob Programming

Auch rollenübergreifend!

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

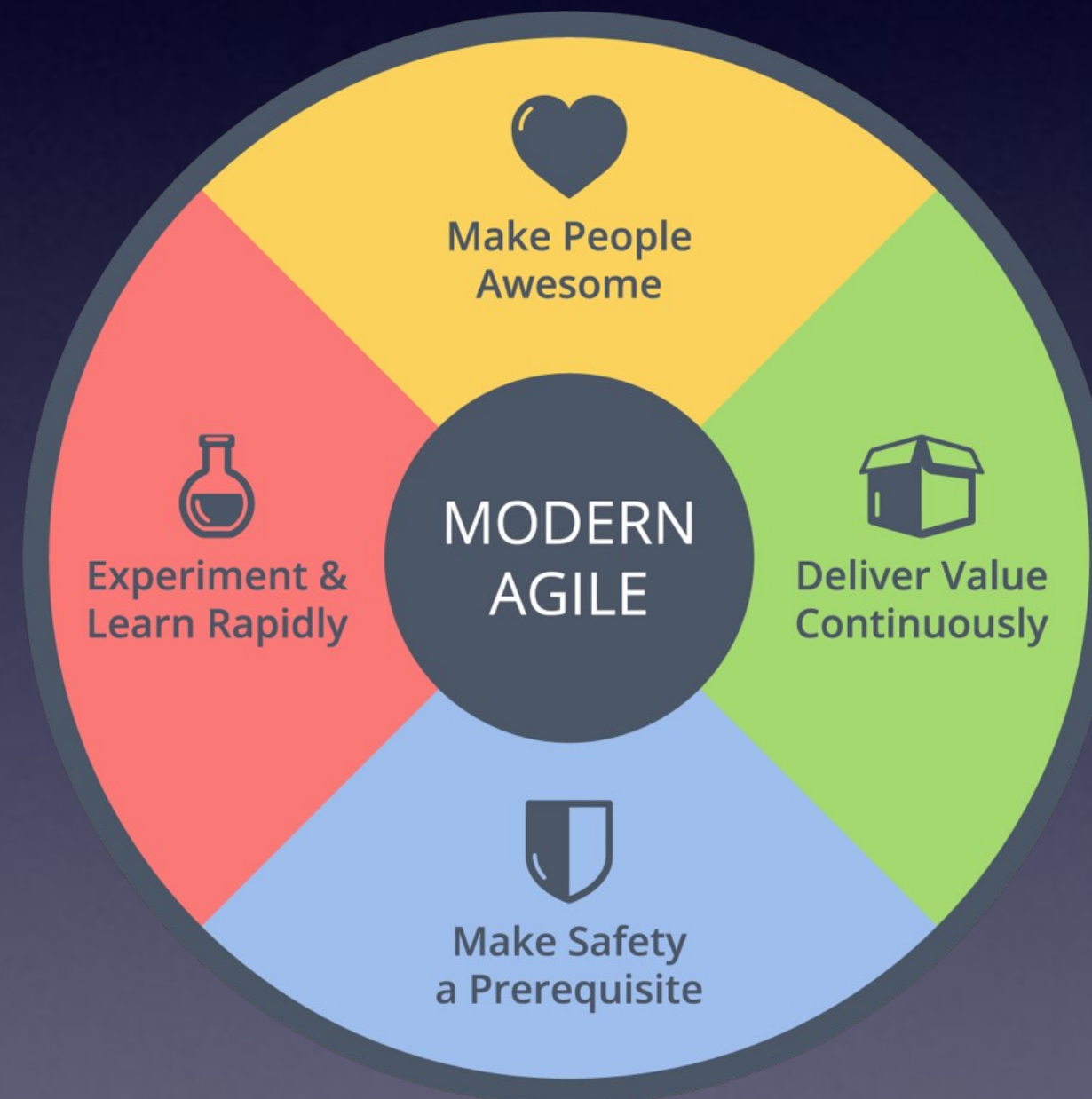
„Continuous Integration of Ideas“

Die **wichtigste** Aufgabe
zügig fertig bekommen.



Modern Agile

Agile – aber so, dass es funktioniert.



Pair und Mob Programming gehören fest dazu!

<http://modernagile.org/>

Gute Vorbereitung muss sein

Gemeinsame Vorbereitung geeigneter, kleiner Tasks.

Discovery, Planning, ...

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



Am besten mit Offline-Erfahrung!

Tools:

*Floobits Editor-/IDE-Plugin, AWS Cloud 9 etc.
TeamViewer etc.*

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

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

XP

Fragen?

Lesbarkeit

Modern Agile

Einfachheit

Strong Style Pairing

Geschwindigkeit

TDD

Collective Product Ownership



Danke!

thomas@muchsoft.com

www.javabarista.de

 @thmuch

