Pair Programming

Auszug aus «Clean Agile»

von Robert C. Martin (übersetzt von Patrick P. Bucher)

01.05.2024

Dies ist eine freie Übersetzung der Seiten 127-131 aus dem Kapitel "Technical Practices" von Robert C. Martins Buch "Clean Agile". [Ergänzungen des Übersetzers stehen in eckigen Klammern.]

Die [technische] Praxis des *Pair Programmings* sah sich über die Jahre einer Menge an Kontroversen und Fehlinformation ausgesetzt. Viele Leute reagieren negativ auf die Idee, dass zwei (oder mehrere) Personen produktiv am selben Problem arbeiten können.

Zunächst einmal ist *Pairing* optional. Niemand sollte dazu gezwungen werden. Weiter findet *Pairing* nur periodisch statt. Es gibt viele gute Gründe dafür, von Zeit zu Zeit alleine zu programmieren. Ein Team sollte ungefähr 50% der Zeit *Pair Programming* betreiben. Die genaue Zahl ist nicht so wichtig. Es kann auch nur 30% oder aber gleich 80% der Zeit sein. Grundsätzlich ist das eine Entscheidung jedes Einzelnen und des Teams.

Was ist *Pairing*?

Von *Pairing* spricht man, wenn zwei Personen gemeinsam an einem einzelnen Programmier-Problem arbeiten. Das Paar kann zusammen am gleichen Computer arbeiten und dabei den Bildschirm, die Tastatur und die Maus teilen. Sie können aber auch an zwei [übers Netzwerk] miteinander verbundenen Computern arbeiten, solange sie den gleichen Code sehen und bearbeiten. Letzteres funktioniert sehr gut mit bekannter *Screen-Sharing*-Software. Diese Software erlaubt es den Partnern auch, sich an entfernten Orten aufzuhalten, solange sie eine gute Daten- und Audio-Verbindung haben.

Die Personen, die sich zum *Pairing* treffen, nehmen manchmal verschiedene Rollen ein. Jemand kann der *Driver* und die andere Person der *Navigator* sein. Der *Driver* hat die Tastatur und die Maus; der *Navigator* hat eine längere Perspektive und gibt Empfehlungen ab. Eine andere Rollenverteilung ist, dass der eine Programmierer einen Test schreibt, und der andere Programmierer diesen zum Laufen bringt und dann den nächsten Test schreibt, den der erste Programmierer dann zum Laufen bringen soll. Dies wird manchmal als *Ping-Pong* bezeichnet.

Meistens gibt es aber gar keine Rollen. Die Programmierer sind einfach gleichgestellte Autoren, welche sich Maus und Tastatur in [enger] Zusammenarbeit teilen.

Pairing wird nicht geplant. Paare bilden und trennen sich nach den Präferenzen der Programmierer. Manager sollten nicht versuchen *Pairing*-Pläne oder Paar-Einteilungen zu erstellen.

Paare sind grundsätzlich kurzlebig. Eine *Pairing*-Session kann einen ganzen Tag einnehmen, dauert aber meistens nicht länger als eine Stunde oder vielleicht zwei. Nur schon *Pairing* von 15 bis 30 Minuten kann hilfreich sein.

[User] Stories werden nicht einem Paar zugeordnet. Einzelne Programmierer (und nicht Paare) sind verantwortlich für das Abschliessen von Stories. Eine Story dauert meistens wesentlich länger als eine *Pairing*-Session.

Im Verlauf einer Woche wird jeder Programmierer etwa die Hälfte seiner *Pairing*-Zeit auf eigene Aufgaben verwenden, wobei er andere zur Hilfe beizieht. Die andere Hälfte der *Pairing*-Zeit wird er darauf verwenden, anderen bei ihren Aufgaben zu helfen.

Senior-Programmierer sollten öfters Paare mit Junior-Programmierer statt mit anderen Senior-Programmierern bilden. Junior-Programmierer sollten öfters Senior-Programmierer als andere Junior-Programmierer um Hilfe bitten. Programmierer mit Spezialgebieten sollten einen bedeutenden Anteil ihrer *Pairing*-Zeit mit Programmierern ausserhalb ihres Spezialgebiets verbringen. Das Ziel ist die Verbreitung und nicht die Konzentration von Wissen.

Warum *Pairing*?

Pairing wird betrieben, um dabei als Team zu wirken. Die Mitglieder eines Teams arbeiten nicht isoliert voneinander, sondern arbeiten im Sekundentakt zusammen. Wenn ein Teammitglied ausfällt, schliessen die anderen Teammitglieder diese Lücke und machen weiterhin Fortschritte in Richtung des Ziels.

Pairing ist bei weitem die beste Möglichkeit um Wissen unter Teammitgliedern auszutauschen und um die Bildung von “Wissenssilos” zu verhindern. Es ist der beste Weg um sicherzustellen, dass niemand im Team unersetzlich ist.

Viele Teams haben berichtet, dass *Pairing* die Fehlermenge reduziert und die Qualität des Designs verbessert. Das dürfte in den meisten Fällen wahr sein. Es ist grundsätzlich besser, mehr als ein Augenpaar auf ein bestimmtes Problem zu richten. Tatsächlich haben viele Teams Code-Reviews durch *Pairing* ersetzt.

Pairing als Code-Review

Pairing ist eine Form des Code-Reviews mit einem bedeutenden Vorteil. Die Programmierer eines Paares sind gemeinsame Autoren während des *Pairings*. Sie betrachten alten Code und prüfen ihn,

doch geschieht dies mit der Absicht, neuen Code zu schreiben. Dadurch ist das Review nicht nur eine statische Prüfung um sicherzustellen, dass die Programmierrichtlinien des Teams eingehalten werden, sondern ein *dynamisches Review* des aktuellen Zustand des Codes aus der Perspektive, wie der Code in naher Zukunft aussehen sollte.

Und was ist mit den Kosten?

Es ist schwierig, die Kosten vom *Pairing* zu messen. Die direkten Kosten liegen darin, dass zwei Leute am gleichen Problem arbeiten. Es sollte offensichtlich sein, dass dies den Aufwand zum Lösen des Problems *nicht* verdoppelt; dennoch führt es zu Mehrkosten. Verschiedene Studien sind zum Schluss gekommen, dass die direkten Kosten ungefähr 15% betragen. Anders gesagt: Es würde 115 Programmierer im *Pairing* benötigen um die Arbeit von 100 einzelnen Programmierern zu leisten (ohne Code-Review).

Unter der naiven Annahme, dass ein Team ca. 50% der Zeit paarweise arbeitet, würde die Einbusse etwas weniger als 8% betragen. Da aber andererseits *Pairings* Code-Reviews ersetzen, ergibt sich daraus wahrscheinlich gar keine Produktivitätseinbusse.

Weiter muss der Nutzen des gegenseitigen Wissensaustauschs und der intensiven Zusammenarbeit beachtet werden. Diese Vorteile sind nicht einfach zu quantifizieren, fallen aber wahrscheinlich bedeutend aus.

Gemäss meiner Erfahrung und der Erfahrung vieler anderer, die *Pairing* betreiben, sofern es informell und von den Programmierern gesteuert stattfindet, ist *Pairing* für das gesamte Team sehr hilfreich.

Nur zwei?

Das Wort “Paar” impliziert, dass nur zwei Programmierer in einer *Pairing*-Session involviert sein können. Auch wenn das normalerweise zutrifft, soll das keine Regel sein. Manchmal entscheiden sich drei, vier oder mehr Programmierer zur Zusammenarbeit an einem bestimmten Problem. (Wie gesagt liegt das im Ermessen der Programmierer.) Dieser Vorgang wird manchmal als *“Mob Programming”* bezeichnet.

Management

Programmierer fürchten oft, dass die Manager *Pairing* missbilligen und sogar verlangen, Paare aufzubrechen um keine [weitere] Zeit zu verschwenden. Ich habe das noch nie beobachtet. In dem halben Jahrhundert, in dem ich Code schreibe, habe ich nie die Intervention eines Managers auf einer solchen tiefen Ebene gesehen. Nach meiner Erfahrung freuen sich Manager darüber, Programmierer beim Zusammenarbeiten zu sehen. Es erweckt den Eindruck, dass Arbeit erledigt wird.

Falls du aber selber ein Manager bist, der dazu neigt, beim *Pairing* einzugreifen, weil du glaubst, dass es ineffizient sei, dann vergiss diese Sorge und lasse die Programmierer damit umgehen. Denn sie sind schlussendlich die Experten. Und falls du ein Programmierer bist, dem der Manager das *Pairing* untersagt hat, erinnere den Manager daran, dass du der Experte [auf diesem Gebiet] bist, und darum du (und nicht der Manager) für die Arbeitsweise verantwortlich bist.

Schlussendlich sollte man niemals um Erlaubnis fürs *Pairing* fragen. Oder fürs Testen [mit automatischen Tests]. Oder fürs Refactoring. Oder… Du bist der Experte. Du entscheidest.