Bug Report für Inhaber von Software-Firmen ab 8 Mitarbeitern

Tagsüber Chef, abends Hotfix.

6 Gründe, warum so viel an Dir hängt, und was Du gegen jeden einzelnen in 30 Minuten tun kannst.

```
~/deine-firma

$ npm test

  ✗ 01  Hotfix-Modus
  ✗ 02  Hardcoded
  ✗ 03  Dead Code
  ✗ 04  Blocking Call
  ✗ 05  Single Point of Failure
  ✗ 06  Ungetestet in Produktion

6 failing  · Welche davon sind es bei Dir? Ab Seite 5.
```

VORAB

Für wen dieser Bug Report ist.

- Du führst eine Software-Firma mit eigenem Entwicklungsteam, ab etwa 8 Mitarbeitern.
- Du kommst selbst aus der Entwicklung und programmierst teilweise noch.
- Deine Woche hat 50 bis 80 Stunden. Die meisten davon gehen an Freigaben, Kundenfragen und Rückfragen aus dem Team.
- Du hast schon einen Teamleiter, ein Buch oder einen Berater probiert. Trotzdem landen die Fragen weiter bei Dir.

Erkennst Du Dich in zwei Punkten wieder? Dann ist dieser Bug Report für Dich. Er ist für Software-Unternehmer geschrieben, die weiter mitarbeiten wollen. Nur muss nicht mehr alles über ihren Tisch.

So ist er aufgebaut

Jedes Kapitel nimmt sich eine Schwachstelle vor, die Du aus dem Code kennst, und zeigt, wie sie in Deiner Firma aussieht. Aufgebaut ist jedes Kapitel wie ein Zyklus in der testgetriebenen Entwicklung:

RED

Woran Du die Schwachstelle erkennst, in 3 Sätzen. Kennst Du zwei davon, ist der Test rot.

GREEN

Der kleinste Fix, als Ticket. Meist reichen 30 Minuten.

REFACTOR

Warum der Fix allein nicht reicht. Und welcher Umbau das Problem dauerhaft löst.

Stefans Fall

Stefan führt eine Software-Firma mit 12 Mitarbeitern. In jedem Kapitel erzählen wir ein Stück aus unserer Arbeit mit ihm.

VORWEG

Es hängt zu viel an Dir. Geplant hat das niemand.

Alles läuft über Deinen Tisch, bis zur kleinsten Freigabe. Abends die Kundenfrage, am Wochenende das Review. Dein Team ist gut, und trotzdem landet jede Kleinigkeit bei Dir, die eigentlich jemand anders entscheiden könnte.

Gewachsen ist das wie Code, der schnell funktionieren musste: erst ein Workaround, dann der nächste. Irgendwann läuft alles über eine Stelle.

Drei Wege, mit einem Fehler umzugehen, der immer wiederkommt

In Clean Code beschreibt Robert C. Martin einen Fehler, der immer wieder auftritt, wenn mehrere gleichzeitig an derselben Sache arbeiten. Er nennt drei Wege, damit umzugehen. In Deiner Firma gibt es dieselben drei.

WEG	IM CODE	IN DEINER FIRMA
1 Tolerate the Failure	Den Fehler abfangen und hinterher aufräumen. Martin nennt das selbst schlampig.	Geht etwas schief, springst Du ein und bringst es in Ordnung. Am nächsten Tag wieder.
2 Client-based Locking	Jeder, der zugreift, muss selbst an die Sperre denken. Vergisst es einer, hängt das System.	„Ihr dürft das entscheiden." Eine Regel, an die jeder selbst denken muss. Sie hält, solange Du danebenstehst.
3 Server-based Locking	Die Regel steckt im System. Wer zugreift, muss an nichts mehr denken.	Wer was entscheidet, wie etwas abläuft und wann es fertig ist, steckt in der Struktur. Nicht in Deinem Kopf und nicht im Gedächtnis jedes Einzelnen.

Der Weg von 1 nach 3 heißt Refactoring. Die Struktur ändert sich in kleinen Schritten, mit einem Test nach jedem. Nach außen bleibt alles gleich: dieselben Kunden, dasselbe Produkt, dasselbe Team. Nur hängt danach nicht mehr alles an Dir.

Die 6 Schwachstellen

01	Hotfix-Modus	Jeden Tag brennt es irgendwo, und Du löschst.	S. 5
02	Hardcoded	Preise, Prioritäten und Prozesse stehen nur in Deinem Kopf.	S. 7
03	Dead Code	Es gibt Prozesse, die keiner lebt.	S. 9
04	Blocking Call	Dein Team wartet jeden Tag auf Deine Antwort.	S. 11
05	Single Point of Failure	Fällt eine Person aus, steht alles.	S. 13
06	Ungetestet in Produktion	Was ohne Dich liegen bleibt, weiß niemand.	S. 15

STEFAN

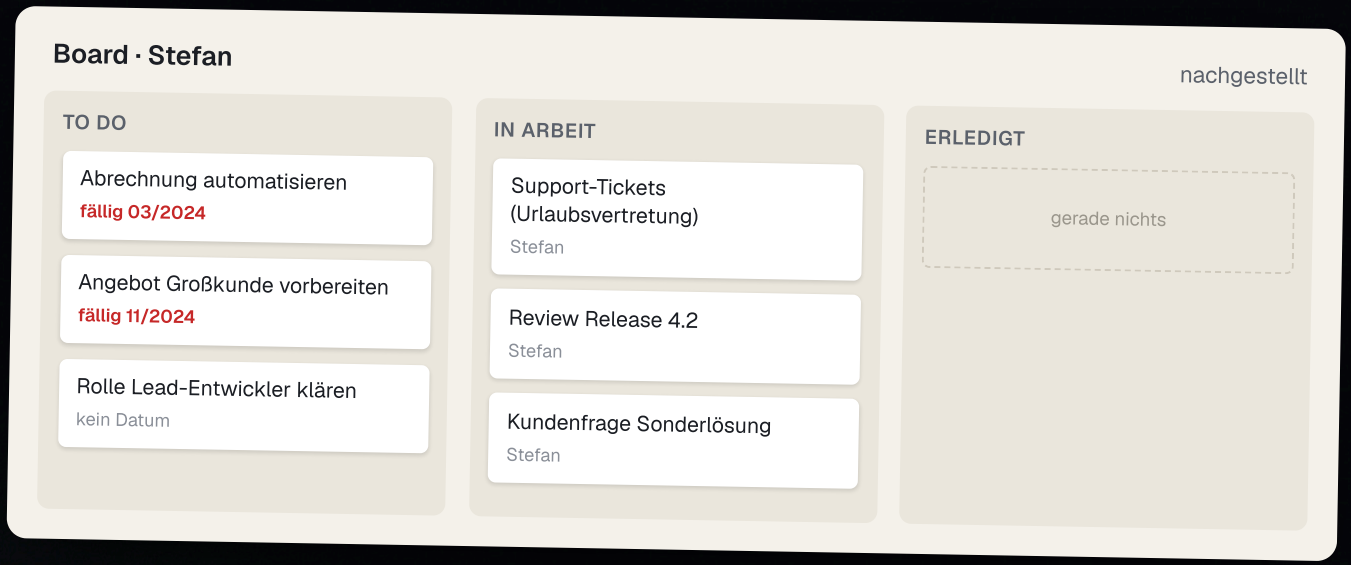
12 Mitarbeiter, die Zahlen stimmen. Und Stefan reibt sich auf.

Im Sommer 2025 kommt Stefan zu uns. Mitte vierzig, spezialisierte Branchensoftware, die Firma hat er vor rund 15 Jahren selbst aufgebaut. Seine Frau führt sie mit ihm. Das Team ist motiviert, die Kunden sind zufrieden. Seine Frau macht sich trotzdem Sorgen um seine Gesundheit.

- Auf seinem Board liegen Aufgaben, die schon 2024 fällig waren.
- Hat der Supporter Urlaub, hütet Stefan selbst das Ticketsystem.
- In seinem Kalender steht jeden Tag „Planung, 15 bis 17 Uhr“. Genutzt hat er den Termin fast nie.
- Er sitzt in jedem Meeting, weil nur er den Überblick hat.
- Nein sagen fällt ihm schwer, bei Kunden wie im eigenen Team.

Letztlich hängt fast alles an ihm.

Dann zeigen 3 Großkunden Interesse. Stefan weiß: Sagen alle 3 zu, platzen die Onboardings, und die Entwicklung liegt Monate zurück.



Rate mal, wer alles in Arbeit hat.

> „Wir reden seit zwei Jahren davon, dass ich aus dem Operativen rauskomme.“
>
> Stefan, vor unserer Zusammenarbeit

01 · HOTFIX-MODUS

Um 9 Uhr hattest Du einen Plan. Um 10 Uhr hat ihn ein Kunde umgeschrieben.

Ein Bug in Produktion. Das Team patcht, und die Arbeit am Release bleibt liegen. Passiert das jeden Tag, ist es kein Notfall mehr, sondern Normalbetrieb. In Clean Code heißt das „Tolerate the Failure“: Der Fehler wird abgefangen und aufgeräumt, behoben wird er nicht.

IM CODE

```
// hotfix: nur kurz
Jeden Tag ein Patch. Das Release
verschiebt sich wieder.
```

IN DEINER FIRMA

Jeden Tag brennt es irgendwo. Du löschst. Was Du eigentlich machen wolltest, bleibt wieder liegen.

RED · KENNST DU DAS?

- Dein Tag gestern lief anders als geplant. Vorgestern auch.
- Auf Deinem Board liegt seit Monaten eine Aufgabe, die Dir jeden Monat Stunden sparen würde. Zum Beispiel eine Abrechnung, die Du immer noch von Hand machst.
- Ist jemand krank oder im Urlaub, springst Du ein.

Kennst Du zwei davon, läuft Deine Firma im Hotfix-Modus.

git log, nachgestellt

```
$ git log --author="chef" --since="1 week" --oneline
a41f2c3  hotfix: Kundenfrage beantwortet              So 22:14
9be07d1  hotfix: Support übernommen (Urlaub Kollege)  Fr 16:40
51c9a8e  hotfix: Abrechnung von Hand, wie jeden Monat  Do 18:05
e02d7b4  hotfix: „nur ganz kurz“ draufgeschaut         Mi 11:20
7c33f90  revert: Wochenplan                            Mo 09:58
```

Feature-Commits diese Woche: 0

> „Du rennst ständig von einem Feuer zum anderen und löschst und löschst und löschst.“
>
> Ralf Armbrüster, intellicon GmbH, 10 Mitarbeiter

Stefans Abrechnung

Bei Stefan liegt seit über einem Jahr eine Aufgabe auf dem Board: die Abrechnung für 8 Kunden automatisieren. 2 Stunden Arbeit, einmalig. Stattdessen macht er die Abrechnung jeden Monat von Hand, weil er die 2 Stunden nie findet. Ist der Supporter im Urlaub, beantwortet Stefan die Standard-Tickets selbst. Seine Begründung: „Das macht ja sonst keiner."

Was Dich das kostet

Deine Stunde ist die teuerste in der Firma, und im Hotfix-Modus geht sie in Themen, die jemand anders für einen Bruchteil machen könnte. Teurer ist, was liegen bleibt: Für den Umbau, der die Brände seltener machen würde, ist nie Zeit. Und Dein Team lernt jeden Tag, dass der Chef einspringt, wenn es eng wird.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-1 — 5 Tage nebenbei — 15 Minuten Freitag — nur Du

Führ eine Woche lang Strichliste.

1. Leg einen Zettel neben die Tastatur. Jede Unterbrechung bekommt einen Strich und ein Stichwort: wer, was, wie lange ungefähr.
2. Schreib daneben, ob das jemand anders hätte machen können.
3. Am Freitag zählst Du nach: Welches Thema kam am häufigsten? Dafür ist ab Montag jemand anders im Team zuständig.

DONE, WENN Du weißt, welches Thema Dich am meisten Zeit kostet. Und jemand anders dafür zuständig ist.

WARUM DAS WIRKT Nach einer Woche weißt Du, was Dich unterbricht, und musst es nicht mehr schätzen. Gibst Du das häufigste Thema ab, fällt ab Montag die größte Gruppe von Unterbrechungen weg.

REFACTOR · DAMIT ES HÄLT

Die Strichliste ist eine kleine Bestandsaufnahme: eine Woche, nur die Unterbrechungen. Sie zeigt Dir, wo Du anfangen kannst. Für den Umbau brauchst Du das ganze Bild: welche Themen über Deinen Tisch laufen, wie viel Zeit sie kosten und wer sie stattdessen machen könnte. Erst dann weißt Du, wo der Umbau am meisten bringt. Und später, ob er gewirkt hat.

Was sofort wegfallen kann, fällt weg. Bei Markus Welter von welante war das ein Support-Meeting am Tag, 5 bis 7 Stunden pro Woche.

WAS DAUERHAFT HILFT

Bestandsaufnahme

Sammle, was über Deinen Tisch läuft. Und kläre, was davon jemand anders machen kann.

Vor dem Refactoring liest Du den Code, nicht nur das Ticket.

02 · HARDCODED

Die wichtigsten Regeln Deiner Firma stehen in Deinem Kopf.

Jeder Entwickler hat schon einmal eine Zahl mitten im Code gefunden, die niemand erklären kann. Sie funktioniert, also fasst sie keiner an. Bis der Code geändert werden muss. Dann geht die Suche los: git blame, alte Tickets, Kollegen fragen. Oft findet sich keiner mehr, der es weiß.

IM CODE

Feste Werte mitten im Code. Warum sie da stehen, weiß keiner mehr.

IN DEINER FIRMA

Preise, Prioritäten, Prozesse stehen in Deinem Kopf. Und nirgends sonst.

RED · KENNST DU DAS?

- Sonderpreise und Absprachen mit Kunden kennst nur Du.
- Neue Leute brauchen Monate, bis sie ohne Rückfrage arbeiten.
- Ob etwas fertig ist, entscheidest Du beim Drüberschauen.

Kennst Du zwei davon, ist bei Dir einiges hardcoded.

angebot.js, nachgestellt

```
function preis(kunde) {
  if (kunde === "Brandt") return 89;   // warum? frag den Chef
  if (kunde === "Hauser") return 72;   // alter Vertrag, glaub ich
  return 95;   // außer der Chef sagt was anderes
}
// TODO: irgendwo aufschreiben (seit 2019)
```

Läuft. Nur weiß keiner, warum.

> „Ich wollte ein Gefühl dafür, wo stehen wir eigentlich in der Entwicklung. Das fehlte mir. 20 Jahre lang."
>
> Matthias Rosenberg, Controllit AG, rund 30 Mitarbeiter

Stefans Tickets

In den ersten Wochen mit Stefan kommt ein Thema immer wieder: die Übergabe zwischen Support und Entwicklung. Der Support erfasst Tickets unvollständig, die Entwickler ärgern sich und fragen nach, und der Support fühlt sich kontrolliert. Beide Seiten arbeiten hart und meinen es gut. Was ein vollständiges Ticket ist, hat nur nie jemand festgelegt. Jeder hat sein eigenes Bild davon, und wenn es hakt, entscheidet Stefan. Am Ende gilt also sein Bild. Aufgeschrieben hat er es nie.

Was Dich das kostet

Was nur in Deinem Kopf steht, musst Du jedes Mal selbst liefern: bei jeder Freigabe, jedem Angebot, jeder Frage eines neuen Kollegen. Nach einer Regel, die es nicht kennt, kann Dein Team nicht arbeiten. Also fragt es. Und Du hältst die Antwort für schneller als das Aufschreiben, seit Jahren. Bei Ralf Armbrüster von intellicon waren so 250 bis 350 Tickets offen. Bis ein Entwickler an ein Ticket kam, vergingen oft 10 Tage. Hatte er dann eine Rückfrage, wusste der Supporter selbst nicht mehr, worum es ging.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-2 — 30 Minuten | diese Woche | nur Du

Schreib Deine 3 Fragen auf.

1. Bei der nächsten Freigabe oder dem nächsten Review schreibst Du mit, worauf Du eigentlich achtest.
2. Mach daraus die 3 Fragen, die Du fast immer stellst. Häng sie ins Ticket-Template oder ins Template für Pull Requests.
3. Ab dann prüft jemand aus dem Team, ob die Antworten da sind und passen. Du nicht mehr.

DONE, WENN das Template Deine Fragen stellt. Und jemand anders prüft, ob sie beantwortet sind.

WARUM DAS WIRKT Die Regel steht jetzt im Template statt in Deinem Kopf. Wer ein Ticket schreibt oder prüft, kann nachlesen, statt Dich zu fragen.

REFACTOR · DAMIT ES HÄLT

Die 3 Fragen holen einen Teil aus Deinem Kopf. Der Rest bleibt drin: wann etwas als fertig gilt, bis wann, was passiert, wenn nicht. Und was Deine Firma zu welchem Preis liefert. Solange das nirgends steht, entscheidet jeder nach Bauchgefühl, und jeder hat ein anderes. Gilt am Ende Deins, landet die Frage wieder bei Dir.

WAS DAUERHAFT HILFT

Standards

Du legst fest, was Du erwartest. Ob etwas fertig ist, entscheidet dann der Standard. Und nicht mehr Bauchgefühl.

Ein Linter diskutiert nicht. Er zeigt an, was nicht passt, und niemand nimmt es persönlich.

03 · DEAD CODE

Im Wiki steht ein Prozess. Gelebt wird er, solange Du danebenstehst.

Jeder Entwickler kennt Code, der im Repo liegt und nie aufgerufen wird. Keiner traut sich, ihn zu löschen, weil er ja vielleicht noch gebraucht wird. Mit Prozessen passiert in Firmen genau dasselbe.

IM CODE

```
// TODO: wird das noch gebraucht?
Steht im Repo. Läuft nie.
```

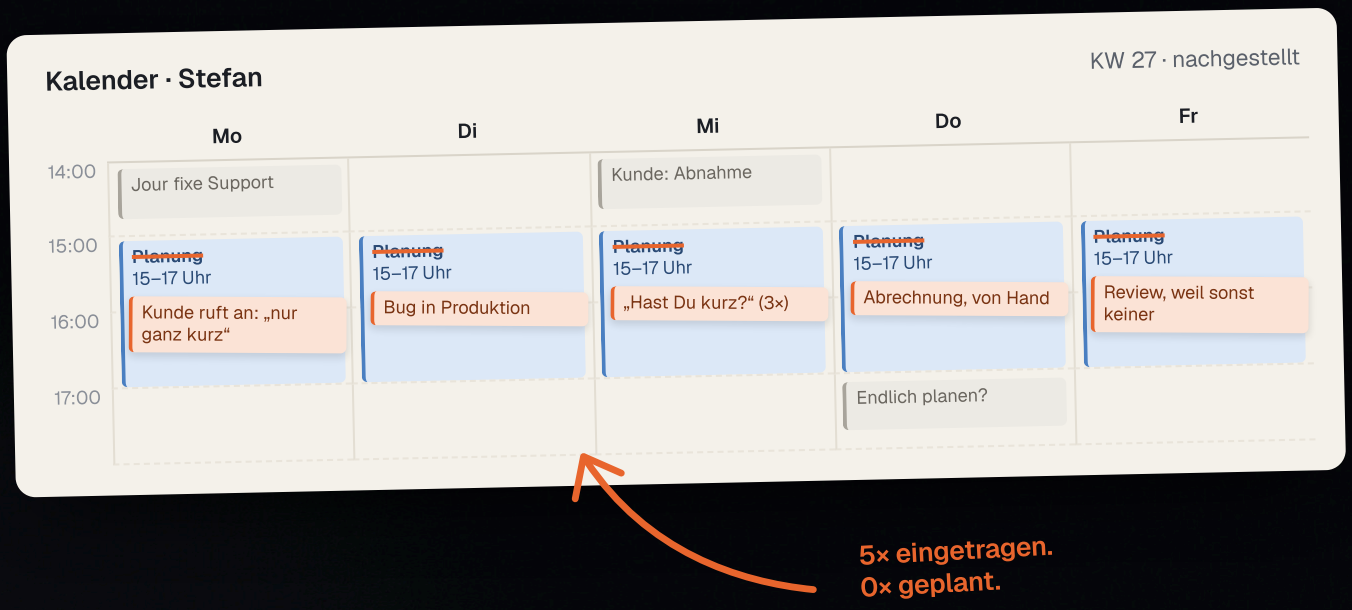
IN DEINER FIRMA

Steht im Wiki. Gilt, solange Du hinschaust.

RED · KENNST DU DAS?

- In Eurem Wiki steht ein Prozess, an den sich außer Dir keiner hält.
- Es gibt eine Checkliste für Releases oder fürs Onboarding. Benutzt wird sie, wenn Du daran erinnerst.
- Wenn es eilt, bist Du der Erste, der den Prozess selbst nicht einhält.

Kennst Du zwei davon, liegt bei Dir Dead Code im Repo.



> „Solange ich da bin und aufpasse, wird das so eingehalten. Aber wenn ich die Tür verlasse, passiert es halt nicht mehr.“
>
> Ein Software-Unternehmer über das, was in seinem Team gilt

Stefans Planung um 15 Uhr

Bei Stefan fängt Dead Code im eigenen Kalender an. Jeden Werktag von 15 bis 17 Uhr steht dort: Planung. Die Regel hat er sich selbst gesetzt, damit endlich Zeit für das bleibt, was die Firma weiterbringt. Genutzt hat er den Termin fast nie. Stehen blieb er trotzdem. Mit Prozessen im Wiki ist es wie mit diesem Termin: Weil es sie gibt, fühlt sich das Thema erledigt an. Sie stören niemanden, nur laufen tun sie nie.

Was Dich das kostet

Ein Prozess, den keiner lebt, kostet mehr als die Stunde, in der er beschrieben und dokumentiert wurde. Dein Team lernt an ihm, wie ernst neue Regeln gemeint sind: Was im Wiki steht, gilt, solange der Chef hinschaut. Führst Du den nächsten Prozess ein, rechnet das Team schon damit, dass er genauso endet. Und Du bist wieder der, der aufpassen muss.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-3 — 30 Minuten | diese Woche | nur Du

Lösch einen Prozess, den keiner lebt.

1. Geh durch Wiki, Checklisten und die Notizen der letzten Retros. Such den einen Prozess, an den sich keiner hält, Du eingeschlossen.
2. Frag Dich: Würde jemand merken, wenn es ihn nicht mehr gäbe? Wenn nein, lösch ihn. Wenn ja, bau ihn dort ein, wo die Arbeit ohnehin vorbeikommt. Beispiel: Die Release-Checkliste wandert aus dem Wiki in den Merge Request, als Pflichtfeld. Ohne Haken kein Merge.
3. Sag es im nächsten Daily: „Den Prozess X gibt es ab heute nicht mehr.“ Oder: „Die Release-Checkliste steht ab heute im Merge Request.“

DONE, WENN der Prozess weg ist oder an einem festen Auslöser hängt. Und Dein Team weiß, welches von beiden.

WARUM DAS WIRKT Jeder Prozess, den alle ignorieren, ist eine Stelle, an der Du aufpassen musst. Hängt er an einem festen Auslöser, läuft er, ohne dass jemand daran denken muss.

REFACTOR · DAMIT ES HÄLT

Mit dem Fix hast Du einen Prozess aufgeräumt. Für die anderen gilt weiter „Client-based Locking“: Sie halten nur, wenn jeder selbst an sie denkt. Und am Ende denkst meistens Du daran.

Dauerhaft hilft, wenn jeder Prozess an einem festen Auslöser hängt, etwa am Merge oder am Sprint-Start. Und eine Person aus dem Team sorgt dafür, dass er läuft.

WAS DAUERHAFT HILFT

Prozesse

Deine Prozesse laufen wie eine Build-Pipeline. Niemand muss sie von Hand anstoßen. Jemand aus dem Team hält sie am Laufen, nicht Du.

Server-based statt Client-based: Die Regel steckt im System, nicht im Gedächtnis jedes Einzelnen.

04 · BLOCKING CALL

Bei jeder Frage an Dich wartet jemand mit seiner Arbeit, bis Du antwortest.

Jeder Entwickler kennt den synchronen Aufruf: Das Programm schickt eine Anfrage und wartet, bis die Antwort kommt. Bei einem Aufruf merkt das keiner. Bei hundert Aufrufen am Tag wird das ganze System langsam.

IM CODE

```
// await chef.antwort()
Der Prozess wartet, bis die Antwort
da ist.
```

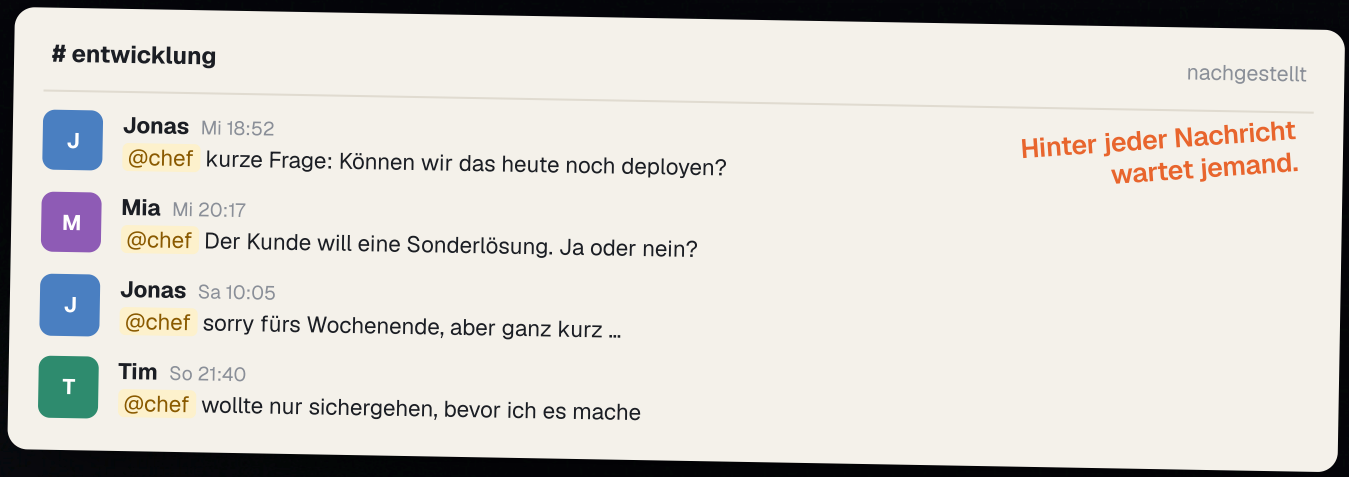
IN DEINER FIRMA

10 Leute warten jeden Tag je 5 Minuten auf eine Antwort von Dir. Das sind rund 4 Stunden pro Woche.

RED · KENNST DU DAS?

- Bist Du einen halben Tag nicht erreichbar, warten danach mehrere Leute auf eine Antwort von Dir.
- Du entscheidest selbst, weil Erklären länger dauern würde.
- Du hast gesagt: „Das dürft Ihr selbst entscheiden.“ Gefragt wirst Du trotzdem.

Kennst Du zwei davon, wartet Dein Team jeden Tag auf Dich.



> „Ich konnte schneller rennen, aber wir haben nicht mehr erreicht.“
>
> Daniel Essl, app-room GmbH, 12 Mitarbeiter

Stefans Daily

Stefan sitzt in jedem Daily, weil er wissen will, was läuft. Einen Lead-Entwickler hat er längst. Gefragt wird trotzdem Stefan, und er antwortet, weil es so schneller geht.

Was Dich das kostet

Jede schnelle Antwort bringt Deinem Team bei, beim nächsten Mal wieder zu fragen. Solange Du am Tisch sitzt, fällt das kaum auf. Bist Du im Kundentermin, im Urlaub oder krank, wartet eine Frage, die in 5 Minuten entschieden wäre, auf einmal Tage. Und die Arbeit dahinter wartet mit. Ralf Armbrüster von intellicon nennt sich für die Zeit davor den „Master der Aufgabenverteilung“: immer rumrennen, immer gucken, wer was zu tun hat.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-4 — 1 Woche | kein Extra-Termin | nur Du

Frag eine Woche lang zurück.

1. Auf jede Frage aus dem Team antwortest Du zuerst: „Was würdest Du machen?“
2. Mach einen Strich, wenn die Antwort gepasst hätte. Einen Kreis, wenn nicht.
3. Am Freitag schaust Du auf die Striche: Bei welchem Thema lag Dein Team fast immer richtig? Dieses Thema entscheidet es ab Montag selbst. Sag es ihm.

DONE, WENN Dein Team weiß, welches Thema es ab Montag ohne Dich entscheidet. Und es tut das auch.

WARUM DAS WIRKT Du siehst, wo Dein Team längst richtig liegt und nur noch auf Dein Okay wartet. Für dieses Thema fällt der Blocking Call weg.

REFACTOR · DAMIT ES HÄLT

Zurückfragen löst ein Thema. Bei allen anderen wird weiter gewartet, solange nicht feststeht, wem welches Thema gehört. Und dann gibt es einen Reflex, den fast jeder kennt: Beim ersten Fehler holst Du die Entscheidung zurück. Auch das ist Tolerate the Failure. Du räumst auf, statt das Problem zu beheben, und die nächste Frage landet wieder bei Dir.

Bei Stefan war es der Lead-Entwickler. Im Oktober übergibt Stefan ihm Daily und Entwicklung: „Du entscheidest, wie Ihr das macht.“ 6 Monate später will er sie nicht wiederhaben.

WAS DAUERHAFT HILFT

Verantwortung

Jedes Thema gehört genau einer Person, mit dem Recht zu entscheiden, ohne Dich zu fragen.

Single Responsibility Principle: Jedes Modul ist für genau eine Sache zuständig. Ein Modul, das für alles zuständig ist, blockiert alles. In Deiner Firma heißt das: Jedes Thema hat einen Zuständigen, und nicht alle Themen denselben.

05 · SINGLE POINT OF FAILURE

Der Laptop fährt mit in den Urlaub. „Nur für den Notfall.“

Jedes Entwicklungsteam kennt den einen Server, ohne den nichts geht, und den einen Kollegen, der als Einziger weiß, wie das Deployment wirklich läuft oder wie dieses eine Modul funktioniert. Single Point of Failure heißt das: Fällt diese eine Stelle aus, steht das Projekt.

IM CODE

```
Ein Server, ohne den nichts geht.
Kein zweiter, der einspringt.
```

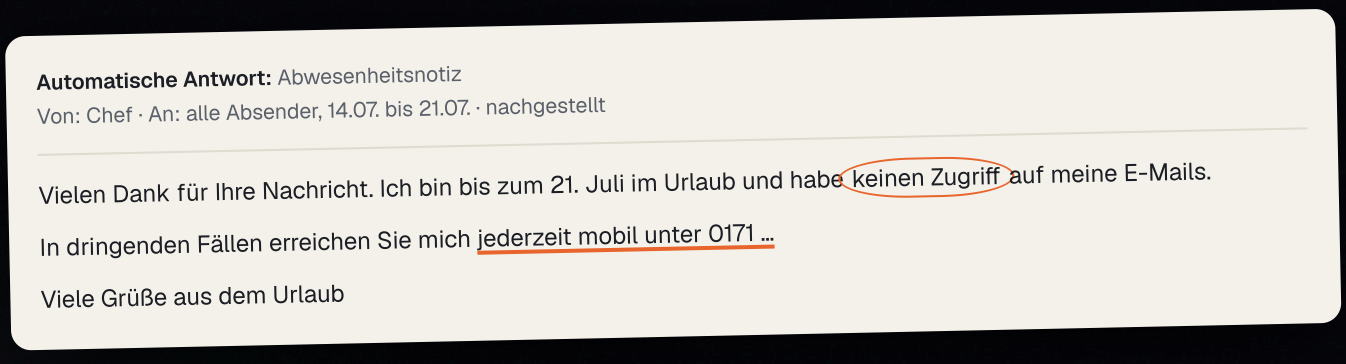
IN DEINER FIRMA

Ein Zugang, ein Kunde, ein Deployment: Das Wissen dazu hat eine Person. Fällt sie aus, steht alles.

RED · KENNST DU DAS?

- Im letzten Urlaub war der Laptop dabei. Aufgeklappt hast Du ihn auch.
- Es gibt Dinge, die nur Du weißt: ein Zugang, ein Kunde, ein Server.
- Fällt ein bestimmter Entwickler aus, steht ein Projekt.

Kennst Du zwei davon, hat Deine Firma einen Single Point of Failure.



Kein Zugriff. Aber jederzeit erreichbar.

> „Jede Anfrage, jede Codezeile, alles lief über mich. Und wenn ich nichts mache, stand das Unternehmen still.“
>
> Markus Welter, welante GmbH, Schweiz, 10 bis 12 Mitarbeiter

Stefans Supporter

In Stefans Firma kennt nur der Supporter das Ticketsystem richtig: welcher Kunde was braucht und welche Tickets eilen. Hat er Urlaub, übernimmt niemand aus dem Team. Dann beantwortet Stefan die Tickets selbst. Den Überblick über die Projekte hat ebenfalls nur Stefan, deshalb sitzt er in jedem Meeting. Fällt einer der beiden aus, bleibt liegen, was nur er weiß.

Was Dich das kostet

Solange alles durch Dich läuft, bist Du der Notfallplan Deiner Firma. Krank im Bett klingelt das Handy, im Urlaub ist der Laptop dabei. Und auf lange Sicht: Was nur in Deinem Kopf steckt, kannst Du an niemanden übergeben, weder an einen Geschäftsführer noch an einen Käufer. Ohne Dich ist die Firma dann weniger wert.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-5 | 30 Minuten | diese Woche | Du und eine zweite Person

Bring einer zweiten Person eine Sache bei.

1. Schreib die 5 Dinge auf, die heute nur Du weißt. Ein Zugang, ein Kunde, ein Prozess, den nur Du kennst.
2. Nimm das, was am meisten kostet, wenn Du morgen ausfällst.
3. 30 Minuten mit einer zweiten Person. Sie schreibt mit, nicht Du.

DONE, WENN es für dieses Thema Redundanz gibt: Zwei Leute können es, und es steht dort, wo man es findet.

WARUM DAS WIRKT Wer mitschreibt, merkt beim Schreiben, wo er etwas nicht verstanden hat, und fragt sofort nach. Und die Notiz setzt nicht voraus, was nur Du weißt.

REFACTOR · DAMIT ES HÄLT

Ein Thema mit Redundanz ist ein Anfang. Die anderen 4 auf Deiner Liste bleiben bei Dir, und jeden Monat kommen neue dazu, solange Übergeben keine feste Aufgabe ist.

Bei welante lief früher jede Codezeile über Markus Welter. Heute reviewt sich das Team gegenseitig.

WAS DAUERHAFT HILFT

Übergeben

Was nur einer weiß, wissen danach zwei. Und es wird dokumentiert. Sonst liegt es in 3 Wochen wieder bei Dir.

Redundanz: Fällt eine Stelle aus, übernimmt die zweite. Danach gibt es die für Deine wichtigsten Themen.

06 · UNGETESTET IN PRODUKTION

Was ohne Dich liegen bleibt, erfährst Du erst, wenn Du weg bist.

Ein Deployment ohne Test geht oft gut. Wenn nicht, findet der Kunde den Bug. Mit der Firma ist es genauso: Was ohne Dich nicht weiterläuft, merkst Du an dem Tag, an dem Du fehlst. Und den suchst Du Dir selten aus.

IM CODE

Deployment ohne Test. Den Bug findet der Kunde.

IN DEINER FIRMA

Niemand weiß, was ohne Dich liegen bleibt. Bis Du weg bist.

RED · KENNST DU DAS?

- Du warst noch nie 2 Wochen weg, ohne dass jemand angerufen hat.
- Nach jedem Urlaub liegt ein Stapel auf Deinem Tisch, der nur auf Dich gewartet hat.
- Fällst Du morgen aus, weißt Du nicht, was als Erstes liegen bleibt.

Kennst Du zwei davon, läuft Deine Firma ungetestet in Produktion.

Testlauf, nachgestellt

```
$ npm test -- --chef=offline --dauer=3w

 PASS  support     ✓ Standard-Tickets ohne Chef beantwortet
 PASS  sprint      ✓ Sprint ohne Chef geplant
 FAIL  freigabe    ✗ Release wartet auf Freigabe (seit Tag 2)
 FAIL  kunde       ✗ Großkunde will „nur kurz den Chef“
 FAIL  angebot     ✗ Sonderpreis unbekannt, Angebot liegt

Tests: 3 failed, 2 passed
```

3× rot = Deine Liste fürs nächste Refactoring

> „Meine Mitarbeiter waren lieb und haben alles zurückgehalten, bis ich wieder da war.“
>
> Ein Software-Unternehmer nach seinem Urlaub

Stefans OP

Im Herbst 2025 muss Stefan operiert werden. Mehrere Wochen fällt er aus, ohne Tagesgeschäft und ohne Zugriff. Ein Jahr früher wäre klar gewesen, was dann passiert. In den Monaten davor hatte Stefan aber umgebaut. Was ein vollständiges Ticket ist, stand im Template. Die Abrechnung lief automatisch. Der Lead-Entwickler hatte die Entwicklung übernommen und führte das Daily. Und für den Support gab es eine Vertretungsregel, also Redundanz. Die OP war der Test, den Stefan nie geplant hatte. Seine Firma hat ihn bestanden, nicht perfekt, aber sie lief.

Was Dich das kostet

Solange niemand testet, lebst Du mit einer Annahme. Entweder Du glaubst, dass alles an Dir hängt, und fährst nie richtig weg. Oder Du glaubst, dass es schon laufen wird, und erfährst das Gegenteil an einem Tag, den Du Dir nicht aussuchst.

GREEN · DER KLEINSTE FIX, DIESE WOCHE

FIX-6 — 1 Woche — im Büro — nur im Notfall eingreifen

Eine Woche Read-only.

1. Kündige es an: „Nächste Woche bin ich da, aber ich greife nicht ein. Entscheidet so, als wäre ich nicht erreichbar."
2. Du bist da, aber Du mischst Dich nicht ein. Kommt eine Frage aus dem Team, gibst Du sie zurück. Eingreifen nur im echten Notfall.
3. Schreib jeden Tag auf, was liegen blieb, was anders lief als bei Dir und wo Du fast eingegriffen hättest. Am Freitag bekommt die erste Zeile ein Ticket.

DONE, WENN Du eine Liste hast, was ohne Dein Eingreifen liegen blieb. Und weißt, was Du als Erstes änderst.

WARUM DAS WIRKT Du siehst, was passiert, wenn Du nicht eingreifst. Das Risiko ist klein, weil Du im Notfall da bist. Und eine Woche ist lang genug, dass nicht alles einfach wartet, bis Du zurück bist.

REFACTOR · DAMIT ES HÄLT

Eine Woche Read-only ist Staging. In Produktion testet die Firma, wenn Du 3 Wochen weg bist. Ohne den Umbau davor fällt sie dabei durch, und auch das ist ein Ergebnis: Es zeigt Dir, wo das nächste Refactoring ansetzt.

Markus Welter hat diesen Test in seinen Flitterwochen gemacht: 3 Wochen, ohne Laptop.

WAS DAUERHAFT HILFT

Test

Die Firma läuft 3 Wochen, ohne dass Du eingreifst. Am besten im Urlaub: angekündigt, ohne Laptop.

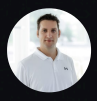
Kein Refactoring ohne Test. Der Test Deiner Firma: 3 Wochen, in denen Du nicht eingreifst.

RELEASE NOTES

6 Monate später.

Im Februar 2026 sitzen wir mit Stefan zur Retrospektive zusammen. Auf die Frage, ob er etwas vom alten Zustand zurückhaben will, das ständige Eingreifen oder das Gefühl, gebraucht zu werden, antwortet er:

> „Ich will es auch nicht zurückhaben."
>
> Stefan, Februar 2026



Daniel und Cindy Essl

app-room GmbH · 12 Mitarbeiter

VORHER „Ich konnte schneller rennen, aber wir haben nicht mehr erreicht."

+ 35 bis 40 Stunden statt 50 bis 80. Mit demselben Team.
+ Preise erhöht, ohne dass Kunden abgewandert sind.



Ralf Armbrüster

intellicon GmbH · 10 Mitarbeiter

VORHER „Du rennst ständig von einem Feuer zum anderen und löschst und löschst und löschst."

+ +38 % Umsatz im ersten Halbjahr.
+ Projekte, die ein halbes Jahr hingen, in einer Woche fertig.



Markus Welter

welante GmbH · Schweiz, 10 bis 12 Mitarbeiter

VORHER „Jede Anfrage, jede Codezeile, alles lief über mich."

+ Das Team reviewt sich gegenseitig.
+ 3 Wochen Flitterwochen, ohne Laptop.



Matthias Rosenberg

Controllit AG · rund 30 Mitarbeiter

VORHER „Ich wollte ein Gefühl dafür, wo stehen wir eigentlich in der Entwicklung."

+ Output pro Sprint verdoppelt.
+ 30 Minuten Daily statt 20 Stunden Meetings pro Woche.



Andreas Andresen

netPension Software GmbH · 8 Mitarbeiter

DANACH „Es hat mich freier gemacht. Den Weg wäre ich alleine nicht gegangen."

+ Software 6 Monate früher am Markt.
+ Den Sprung in die schwarzen Zahlen geschafft.



Dr. Christian A. Müller

wasteant GmbH · 15 bis 20 Mitarbeiter

VORHER „So viele Themen, dass man den Wald vor Bäumen nicht mehr sieht."

+ Mit demselben Team von 2 auf 10 Großkunden, ohne selbst mehr zu arbeiten.
+ Klare Rollen für Projekt und Feature-Entwicklung.

DEIN SPRINT

6 Tickets. Nimm eins.

Nimm das Ticket aus dem Kapitel, in dem Du Dich am meisten wiedererkannt hast. Passen mehrere, fang vorne an: Die Kapitel bauen aufeinander auf.

~/deine-firma

```
$ npm test

  ✓ 01  Hotfix-Modus
  ✓ 02  Hardcoded
  ✓ 03  Dead Code
  ✓ 04  Blocking Call
  ✓ 05  Single Point of Failure
  ✓ 06  Ungetestet in Produktion

6 passing
```

FIX-1 Hotfix-Modus · S. 6

Führ eine Woche lang Strichliste.

5 Tage nebenbei, 15 Minuten am Freitag

FIX-2 Hardcoded · S. 8

Schreib Deine 3 Fragen auf.

30 Minuten

FIX-3 Dead Code · S. 10

Lösch einen Prozess, den keiner lebt.

30 Minuten

FIX-4 Blocking Call · S. 12

Frag eine Woche lang zurück.

1 Woche, kein Extra-Termin

FIX-5 Single Point of Failure · S. 14

Bring einer zweiten Person eine Sache bei.

30 Minuten zu zweit

FIX-6 Ungetestet in Produktion · S. 16

Eine Woche Read-only.

1 Woche, im Büro

Wenn nach 3 Wochen alles wieder ist wie vorher, weißt Du, an welcher Stelle der Fix allein nicht reicht. Genau darüber reden wir im Unternehmens-Review.

DER NÄCHSTE SCHRITT

Wo hängt Deine Firma am stärksten an Dir?

30 Minuten · Zoom · kostenlos

Unternehmens-Review

Der Bug Report kennt Deine Firma nicht. Im Review schauen wir 30 Minuten lang auf Deine.

Unternehmens-Review buchen

go.anne-thorsten.de/review

Du hast einen Geschäftspartner? Bring ihn mit.

1. **Du erzählst, wo es bei Dir klemmt.**
   Hast Du schon einen Fix ausprobiert, erzählst Du, was dabei passiert ist.
2. **Wir schauen auf Deine Firma.**
   An welcher der 6 Stellen hängt es bei Dir am meisten? Und warum hält der Fix bei Dir nicht? Dazu zeigen wir Dir, wie andere Software-Firmen genau diese Stelle gelöst haben.
3. **Du nimmst einen Schritt mit.**
   Einen für nächste Woche, den Du auch ohne uns gehen kannst.



WER MIT DIR ARBEITET

Thorsten Schiffer

Thorsten entwickelt seit über 25 Jahren Software. Seit 2009 führt er Entwicklungsteams, seit 2016 coacht er sie. Er redet mit Dir über Tickets und Deployments, weil er sie selbst macht.

Anne Neuroth

Anne schaut auf das, was nach dem Umbau Geld bringen muss: Angebot, Preis und Markt. Damit aus der Zeit, die Du gewinnst, auch Umsatz wird.

Seit 2023 bauen wir mit Software-Firmen ihre Struktur um. Den Umbau machst Du mit Deinem Team, wir begleiten Dich dabei. So hängt danach auch nichts an uns.