

DER BRIEFING-GUIDE

Wie du Aufgaben stellst, die wirklich funktionieren

1. Das Problem und warum es für jeden wichtig ist

Jeder kennt es aus der eigenen Arbeit. Die Mitarbeiter und Kollegen regen sich permanent auf über unklare Aufgaben. Herr Schmitz hat mal wieder die Aufgabe nicht gut genug erklärt. Es gibt zu viele Rückfragen. Man kann nicht anfangen zu arbeiten. Oder der Chef regt sich auf, dass die Leute nicht eigenständig arbeiten können und an die Hand genommen werden müssen.

Und ich kann kaum glauben, dass das wirklich so ein Ding ist. Nach 15 Jahren weiß ich es einfach, aber wollte nie glauben, dass es etwas so Simples sein kann: **Wie ist die Aufgabe formuliert und wie / wo wird sie kommuniziert?** Wir reden hier über die einfachste Form des Projektmanagements und wir werden nun gemeinsam lernen, wie du als Mitarbeiter oder Chef etwas ändern kannst.

Zwei Bereiche müssen wir abdecken:

Dieser Guide soll funktional sein und nicht wissenschaftlich. Wir müssen lernen, wie man Projekte bzw. Tasks organisiert – von Anfang bis Abnahme – und wie ein einzelner Task formuliert sein muss, damit ihn Leute ohne zu viele Fragen verstehen und bearbeiten können.

Das Problem:

- Ziele unklar
- Kein Kontext
- Output-Format fehlt
- Niemand verantwortlich
- Keine Deadline
- Infos verstreut

Das Ergebnis:

- Rückfragen
- Missverständnisse
- Zeitverschwendung
- Frustrierte Teams
- Schlechte Ergebnisse

2. Prinzipien eines guten Briefings

Bevor wir in die Details gehen, hier die Grundregeln:

1. **Klarheit > Kreativität** – Eindeutigkeit schlägt Eleganz. Lieber präzise als poetisch.
2. **Eine Person verantwortlich** – Nicht "Team" oder "Wir". Ein Name, eine Person.
3. **Ergebnis definiert, nicht nur Tätigkeit**
Nicht: "Konzept entwickeln" / Sondern: "3 Landing-Page-Varianten als Figma-File"
4. **Ausführbar ohne Rückfragen** – Wenn du nachfragen musst, ist das Briefing schlecht.
5. **Arbeitsanweisung, nicht Informationsablage** – Briefings sind Handlungsanweisungen, keine Wikis.

Der Prozess dahinter

Wie werden wir es übermitteln? Innerhalb eines Prozesses, der ganz einfach sein kann oder auch sehr tief – je nachdem, wie deine Strukturen sind.

Simpel

Eine Aufgabe kann "Unbearbeitet" oder "Fertig" sein.

Etwas aufwendiger

Zusätzlich noch "In Bearbeitung".

Professionell

Systeme wie Scrum mit verschiedenen Status, mehreren Projekten und Grundprozessen.

3. Projektmanagement – Die Einzelemente

Wenn du Aufgaben organisierst, gibt es drei Kernbereiche, die du verstehen musst. Jeder dieser Bereiche hat eine klare Funktion und hilft dir, Chaos zu vermeiden.

1) Status der Aufgabe

Der Status zeigt, wo eine Aufgabe gerade steht. Ohne klaren Status weiß niemand, ob etwas schon angefangen wurde, wer dran ist oder ob es fertig ist.

Level 1 – Minimal (2 Status)

- **Offen** Noch nicht begonnen
- **Fertig** Abgeschlossen

Funktioniert für kleine Teams oder einfache Projekte.

Level 2 – Standard (3-4 Status)

- **Offen** Geplant
- **In Bearbeitung** Aktiv
- **Review** Zur Prüfung
- **Fertig** Freigegeben

Der Sweet Spot für die meisten Teams.

Level 3 – Professionell (5+ Status)

- **Backlog** Nicht geplant
- **Ready** Bereit
- **In Progress** In Arbeit
- **QA** Prüfung
- **Testing** Im Test
- **Done** Fertig

Für größere Teams oder komplexe Projekte.

Wichtig:

Egal welches Level – alle müssen wissen, was der Status bedeutet. "In Bearbeitung" heißt nicht "Ich hab mal reingeschaut", sondern "Ich arbeite aktiv daran".

2) Aufteilung der Aufgabe

Eine Aufgabe kann zu groß sein. Dann musst du sie runterbrechen. Aber wie?

Epics, Stories, Tasks – Was ist was?

Typ	Dauer	Beispiel
Epic (Großprojekt)	1-3 Monate	"Neue Website launchen"
Story / Feature (Teilprojekt)	1-2 Wochen	"Kontaktformular bauen"
Task (Einzelaufgabe)	1-3 Tage	"Button-Design erstellen"
Subtask (Mini-Aufgabe)	1-4 Stunden	"Icon für Button suchen"

Warum ist das wichtig?

Wenn du sagst "Bau mal die Website", ist das zu groß. Niemand weiß, wo er anfangen soll. Wenn du sagst "Erstelle das Kontaktformular mit Name, E-Mail, Nachricht und einem Submit-Button", dann kann jemand sofort starten.

WICHTIG: Kein Planungsoverkill!

Die Hierarchie ist kein Selbstzweck. Nutze nur so viel Struktur, wie du wirklich brauchst. Die Frage ist immer: Wie viele Leute arbeiten daran? Wie komplex ist es wirklich?

Beispiel – Kleines Team (2 Personen: Designer + Entwickler):

Ihr baut zusammen eine Landing Page. Hier brauchst du kein Epic und keine Subtasks für jeden Hero-Bereich oder Button.

✗ Overkill:

```
Epic: Website launchen
└─ Story: Herobereich erstellen
    ├── Task: Headline schreiben
    ├── Task: Bild auswählen
    ├── Task: Button designen
    └─ Task: Button programmieren
```

Das ist viel zu granular. Die Administration frisst mehr Zeit als die Arbeit selbst.

✓ Pragmatisch:

```
Task 1: Landing Page Design (Figma)
  → Zuständig: Designer
  → Projekt-Tag: "Landing Page"

Task 2: Landing Page Entwicklung
  → Zuständig: Entwickler
  → Projekt-Tag: "Landing Page"
```

Fertig. Zwei klare Aufgaben, beide mit Projektbezug versehen. Keine unnötige Bürokratie.

Wann brauchst du Epics und tiefe Hierarchien?

Wenn du viele Leute, mehrere Teams oder lange Projekte hast:

- 5+ Personen arbeiten daran
- Mehrere Teams sind beteiligt (Design, Frontend, Backend, Marketing)
- Das Projekt dauert mehrere Monate
- Es gibt viele Abhängigkeiten (Team A muss fertig sein, bevor Team B starten kann)

Die goldene Regel:

Wenn die Administration mehr Zeit kostet als die Arbeit selbst, machst du es zu kompliziert.

3) Projektbezogene Organisation

Tasks existieren nicht im luftleeren Raum. Sie gehören zu einem Projekt, einem Team oder einem Kunden.

Projekte / Bereiche

- "Website-Relaunch"
- "Social-Media-Kampagne Q1"
- "Produktlaunch App 2.0"

Teams / Verantwortlichkeiten

- Team: Design
- Team: Development
- Team: Marketing

Priorisierung

Einfach:

Hoch, Mittel, Niedrig

MoSCoW:

Must have, Should have, Could have, Won't have

Labels / Tags

Bug

Feature

Design

Dringend

Der Punkt ist, du kannst beliebig aufteilen und zuordnen. Hauptsache, ihr setzt für euch fest, wonach. Wie das geht, erkläre ich dir kurz im nächsten Abschnitt.

Element	Funktion	Beispiel
Status	Wo steht die Aufgabe?	Offen, In Bearbeitung, Fertig
Aufteilung	Wie groß ist die Aufgabe?	Epic → Story → Task → Subtask
Projektbezug	Wozu gehört sie?	Projekt, Team, Priorität, Sprint

Wofür entscheide ich mich nun und wie setze ich das um?

Von der Todo-Liste für dich selbst bis hin zum hochkomplexen Ticketsystem mit 10 verschiedenen Status, Epics, Subtasks und 200 Mitarbeitern – es gibt Tools wie Sand am Meer. Die Frage ist: **Wo fängst du an?**

Schritt 1: Finde dein Hauptelement

Überlege dir, was eure Arbeit prägt. Was ist die natürliche Aufteilung?

- **Projekte** (Agentur, Freelancer): "Kunde A", "Kunde B", "Internes"
- **Features** (Produktentwicklung): "User-Login", "Dashboard", "Reporting"
- **Teams / Abteilungen** (große Firma): "Marketing", "Development", "Sales"
- **Zeiträume** (Scrum): "Sprint 1", "Sprint 2", "Backlog"

Das wird deine **Hauptordnung** (Ordner, Epic, Board – nenn es, wie du möchtest).

Schritt 2: Starte simpel mit Kanban

Fang mit einem einfachen Kanban-Board an:

TODO	DOING	DONE
Aufgabe 1	Aufgabe 3	Aufgabe 5
Aufgabe 2		Aufgabe 6
Aufgabe 4		

Warum Kanban?

- Jeder versteht es sofort
- Keine Lernkurve
- Funktioniert für 1 Person genauso wie für 10

Schritt 3: Iterativ verbessern

Arbeite damit. Wenn du an Grenzen stößt, werde detaillierter.

Typische Probleme und Lösungen:

Problem	Lösung
"Ich weiß nicht, wer an was arbeitet"	→ Personen zu Tasks zuweisen
"Zu viele Aufgaben in DOING"	→ Limit setzen (max. 3 pro Person)
"Unklar, was zur Abnahme bereit ist"	→ Status "Review" hinzufügen
"Aufgaben bleiben hängen"	→ Deadlines hinzufügen
"Unklare Prioritäten"	→ Prioritäts-Tags einführen
"Zu viele verschiedene Projekte"	→ Boards pro Projekt

Schritt 4: Nicht overengineeren

Viele Teams bauen sich ein Monster-System mit 15 Status, 200 Tags und komplizierten Workflows – und niemand nutzt es richtig.

Die goldene Regel:

So wenig Administration wie möglich, so viel wie nötig.

Das Wichtigste:

Ein simples System, das alle nutzen, schlägt ein perfektes System, das keiner versteht.

4. Standard-Briefingblock – Die Einzelelemente (der Kern des Projektmanagements)

Wichtig:

Bei allen vorherigen Fragen, wenn nun der Inhalt der Aufgabe nicht passt, bringt die ganze schöne Struktur nichts.

Jedes allgemeine Briefing muss diese 10 Punkte enthalten:

- 1 Ausführende Person**
Wer führt die Aufgabe aus?

- 2 Projekt / Kontext**
Wofür ist das? Warum jetzt?

- Klare Aufgabenformulierung + Titel**
3 Was genau soll entstehen? **Einfach formulieren. Kurze Sätze. Nicht kopieren. Selbst!**

- Deadline + Aufwandsschätzung**
4 Bis wann? Wie viel Zeit? Gibt es einen Stichtag oder soll die Arbeitszeit begrenzt sein?

5

Verantwortliche Person

Wer muss informiert werden? Wer gibt Input? Wer ist Product Owner?

6

Status

Je nach Art des Projektmanagements

7

Ergebnisformat

Was bedeutet "fertig"? (Format, Qualität, Kriterien)

8

Abhängigkeiten

Was muss vorher fertig sein?

9

Materialien / Vorinfos

Wo sind Docs, Links, Assets?

10

Abnahme-Person

Wer entscheidet "fertig"?

Kurzregel: Was, für wen, bis wann, wie, wo, wer entscheidet.

5. Spezifische Briefingfragen

Je nach Art der Aufgabe gibt es zusätzliche Fragen, die ihr oben implizit mit beantwortet habt, aber wenn ihr ein großes Team habt, macht es Sinn, spezifische Vorlagen für diese Bereiche zu erstellen. Hier ein paar Beispiele:

DESIGN

- Welches Format? (Web, Mobile, Print, Dimension)
- Gibt es Brand Guidelines / CI?
- Welche Referenzen / Beispiele?
- Welche Elemente müssen rein? (Logo, CTA, Text)
- Wie viele Varianten?
- Welches Tool / Format für Abgabe? (Figma, PDF, PNG)

DEVELOPMENT

- Welche Funktionen genau?
- Welche Plattform / Stack?
- Gibt es bestehenden Code / API?
- Welche Browser / Devices?
- Performance-Anforderungen?
- Wer testet / deployed?

MARKETING / PERFORMANCE

- Welches Ziel? (KPI: CTR, CPA, Conversions)
- Welche Zielgruppe?
- Welche Kanäle? (Meta, Google, LinkedIn)
- Budget?
- Tracking / Attribution?
- Welche Assets vorhanden?

SOCIAL CONTENT

- Welche Plattform? (Instagram, TikTok, LinkedIn)
- Format? (Reel, Story, Post, Carousel)
- Ton / Stil? (professionell, witzig, sachlich)
- Hashtags / Captions?
- Posting-Zeitpunkt?
- Wer freigeben / postet?

EVENTS

- Wo, wann, wie viele Personen?
- Welches Format? (Workshop, Vortrag, Networking)
- Welche Technik / Location?
- Wer lädt ein / moderiert?
- Follow-up geplant?
- Budget / Catering?

Das sind allgemeine Beispielfragen, ihr werdet am besten für euch einschätzen können, welche Fragen ihr intern oft stellt.

6. Übergabe- & Abnahme-Regeln

Kritisch:

Übergabe und Abnahme sind die kritischsten Momente im Prozess.

① Eine Person prüft

Nicht "das Team schaut mal drüber". Eine Person entscheidet.

② Klare Freigabe

Nur zwei Zustände: Freigegeben oder nicht. "Sieht gut aus" ≠ Freigabe.

Feedback strukturiert, an einem Ort

③ Nicht: "Gefällt mir nicht". Sondern: "Button muss größer sein (48px statt 32px)" im Task.

Änderungen dokumentiert

④ Jede Änderungsrunde wird notiert: Was wurde geändert? Warum? Von wem freigegeben?

Definition of Done

⑤ Was bedeutet "fertig"? Alle Funktionen laufen? Responsive getestet? Texte Korrektur gelesen? Assets exportiert?

6

Abnahme = Final

Nach Freigabe: keine weiteren Änderungen ohne neues Briefing.

7. KI-Assistenz für bessere Briefings

Hinweis:

KI kann helfen, Briefings zu prüfen und zu strukturieren. Und es auch noch schlimmer machen. Ich persönlich bin der Überzeugung, wer dieses Dokument hier liest, sollte erstmal per Hand lernen, wie das geht. KI hilft nur zu unterstützen. Hier trotzdem 2 Miniprompts.

1. Lücken prüfen

Analysiere dieses Briefing. Welche Informationen fehlen, damit jemand die Aufgabe ohne Rückfragen ausführen kann?

[Briefing einfügen]

Die KI zeigt dir, was unklar ist.

2. Strukturierung

Hier ist eine unstrukturierte Aufgabenbeschreibung.
Formatiere sie nach dem 10-Punkte-Briefing-System:

1. Ausführende Person
2. Projekt / Kontext
3. Aufgabe
4. Deadline + Aufwand
5. Verantwortliche Person
6. Status
7. Ergebnisformat
8. Abhängigkeiten
9. Materialien
10. Abnahme-Person

[Infos einfügen]