



GROWTH ENGINES × SCALING SYSTEMS × AI = HYPERGROWTH

# PRODUCT

CPOs, VPs Product, Product Leaders • Series A-C • €5M–€100M ARR

Das 5-Komponenten-Framework für AI-Native Produktentwicklung – Wie man 10x schneller ausliefert mit radikal besserem Product-Market Fit

TIME TO MARKET

**-83%**

6-12 Mo. → 2-4 Wo.

VELOCITY

**+400%**

1-2 → 6-12/Quartal

FEATURE  
ADOPTION

**+30%**

60% → 85-95%

ROI

**10-20x**

2-3 Monate  
Payback

Version 1.0 • 3. Februar 2026

Michel Lason, Alban Halili, Florian Metzger

EXPERTISE × SPEED = IMPACT

## 01 Executive Summary

Das Problem, die Lösung, die Ergebnisse

### Die Situation

Traditionelle Produktentwicklung scheitert bei Skalierung. Die Zahlen sprechen für sich:

TIME TO MARKET <b>6-12 Mo.</b> zu langsam	SHIPPING VELOCITY <b>1-2/Q</b> zu langsam	FEATURE ADOPTION <b>60-70%</b> zu niedrig	TIME TO VALUE <b>30-90 Tage</b> zu langsam	PMF SCORE <b>0,2-0,4</b> schwach
---	---	---	--	--

### Das Ergebnis

Unternehmen liefern Features aus, die niemand nutzt. Entwicklungszyklen sind zu lang.

Product-Market Fit bleibt schwer erreichbar. Wettbewerber liefern 10x schneller.

### Die Komplikation

AI-native Unternehmen liefern **10x schneller** mit radikal besserem Product-Market Fit:

Unternehmen	ARR	Team	Geheimnis
Cursor	€100M in 12 Mo.	Kleines Team	10x schneller als traditionelle IDEs
Midjourney	€492M in 18 Mo.	107 Mitarbeiter	€4,6M/Mitarbeiter, AI-gestützt
Perplexity	100M Anfragen/Mo.	AI-first	10x schneller als traditionelle Suche

#### ✗ Traditionelle Unternehmen

- Manuelle Feature-Entwicklung (6-12 Monatszyklen)
- Feature-getriebenes Produkt (bauen, was Kunden verlangen)
- Langsame Iteration (vierteljährliche Releases)

#### ✓ AI-Native Unternehmen

- AI-gestützte Entwicklung (2-4 Wochen Zyklen, -83%)
- Product-Led Growth (Produkt treibt Akquisition)
- Schnelle Iteration (wöchentliche Releases, 10x schneller)

### Die Lücke wird größer

Traditionelle Unternehmen können nicht mithalten. AI-native Unternehmen gewinnen.

### Die Frage

Wie transformiert man die Produktentwicklung von traditionell zu AI-native, ohne aktuelle Kunden zu stören?

### Die Antwort

$$\text{AI-NATIVE PRODUCT} = (\text{PLG} \times \text{Velocity} \times \text{Quality} \times \text{Analytics} \times \text{AI}) \times \theta$$

**PLG** = Product-Led Growth (treibt Akquisition)      **Velocity** = Shipping Velocity (10x schneller)

**Quality** = Feature Quality (85-95% Adoption)      **Analytics** = Product Analytics (datengetrieben)

**AI** = AI-Powered Development (Multiplikator)      **θ** = AI Maturity (0-1, höher = besser)

**Die Ergebnisse (6-12 Monate):**

TIME TO  
MARKET

**-83%**

6-12 Mo. →  
2-4 Wo.

VELOCITY

**+400%**

1-2 → 6-12/Q

ADOPTION

**+30%**

60% → 85-  
95%

TIME TO  
VALUE

**-95%**

30-90 → 1-3  
Tage

PMF SCORE

**+100%**

0,2-0,4 →  
0,6-0,8

## 02 Das Problem

Warum traditionelle Produktentwicklung scheitert

### 2.1 Warum traditionelle Produktentwicklung scheitert

Das traditionelle Produktentwicklungsmodell funktionierte bei €5M ARR. Bei €50M ARR bricht es zusammen. Warum?

#### Grund 1: Manuelle Entwicklung

- Feature-Entwicklung ist manuell (Entwickler schreiben Code)
- Keine AI-Unterstützung (keine Code-Generierung, keine Testautomatisierung)
- Ergebnis: 6-12 Monatszyklen (zu langsam)

#### Grund 2: Feature-getriebenes Produkt

- Bauen, was Kunden verlangen (nicht was sie brauchen)
- Kein Product-Led Growth (Vertrieb treibt Akquisition)
- Ergebnis: Feature-Bloat, geringe Adoption

#### Grund 3: Langsame Iteration

- Vierteljährliche Releases (zu langsam)
- Keine kontinuierliche Verbesserung
- Ergebnis: Wettbewerber liefern 10x schneller

### 2.2 Die AI-Native Chance

AI-native Unternehmen liefern 10x schneller:

Metrik	✗ Traditionell	✓ AI-Native	Verbesserung
Time to Market	6-12 Monate	2-4 Wochen	≈ -83%

Shipping Velocity	1-2/Quartal	6-12/Quartal	↗ +400%
Feature Adoption	60-70%	85-95%	↗ +30%
Time to Value	30-90 Tage	1-3 Tage	↘ -95%
PMF Score	0,2-0,4	0,6-0,8	↗ +100%
Team Size	20-50 Devs	10-20 Devs	↘ -50%

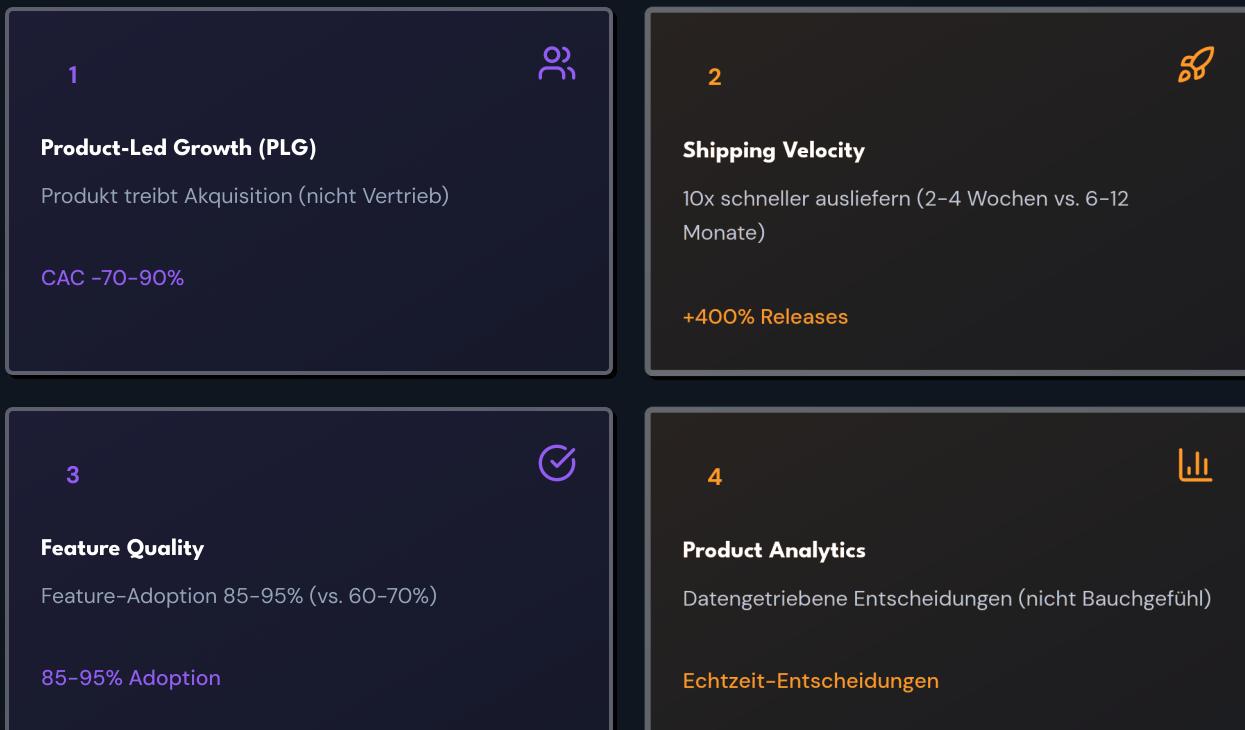
## 03 Das Framework

Das 5-Komponenten AI-Native Produkt-Framework

### 3.1 Das 5-Komponenten-Framework



$$\text{AI-NATIVE PRODUCT} = (\text{PLG} \times \text{Velocity} \times \text{Quality} \times \text{Analytics} \times \text{AI}) \times \theta$$



5

**AI-Powered Development**

AI übernimmt 40–60% der Entwicklungsarbeit

✗ Traditionell

100% manuell, 20–50 Entwickler

✓ AI-Native

40–60% AI-generiert, 10–20 Entwickler

Team -50%

**5 integrierte Komponenten × AI-Multiplikator (θ) = Time to Market -83%, Velocity +400%, PMF +100%**

### 3.2 Warum das Framework funktioniert

Das Framework basiert auf 3 Kern-Insights:

**Insight 1: PLG treibt Akquisition****Traditionell:** Vertrieb treibt Akquisition (teuer)**AI-Native:** Produkt treibt Akquisition (günstig)**Ergebnis:** CAC -70–90%**Insight 2: Velocity treibt Iteration****Traditionell:** Vierteljährliche Releases**AI-Native:** Wöchentliche Releases**Ergebnis:** Time to PMF -75%**Insight 3: AI treibt Effizienz****Traditionell:** Manuelle Entwicklung (teuer)**AI-Native:** AI-gestützte Entwicklung**Ergebnis:** Teamgröße -50%

## 04 Komponente 1: Product-Led Growth (PLG)

Produkt treibt Akquisition (nicht Vertrieb)

### 4.1 Warum PLG wichtig ist

Product-Led Growth bedeutet: Produkt treibt Akquisition (nicht Vertrieb).

#### ✗ Traditional (Sales-Led)

- Lead-Generierung (Marketing)
- Sales Qualification (SDRs)
- Demo (AEs)
- Trial (manuelles Setup)
- Abschluss (Verträge)

CAC €10k, 90-180 Tage Zyklen

#### ✓ PLG (Product-Led)

- Self-Service Signup (kein Vertrieb)
- Sofortige Aktivierung (kein Setup)
- Wert in Minuten (nicht Tagen)
- Virale Loops (Nutzer laden Nutzer ein)
- Upgrade (Self-Service)

CAC €1k-€3k (-70%), 1-7 Tage Zyklen

CAC

**-70-90%**

€10k → €1k-€3k

SALES CYCLE

**-95%**

90-180 → 1-7 Tage

CONVERSION

**+100%**

15-25% → 40-60%

### 4.2 Die 5 PLG-Prinzipien

#### Prinzip 1: Self-Service Signup

Jeder kann sich anmelden (kein Sales-Qualified Lead nötig). Ergebnis: 10x mehr Anmeldungen.

#### Prinzip 2: Sofortige Aktivierung

1-5 Minuten Aktivierung (kein 30-90 Tage Onboarding). Ergebnis: Aktivierungsrate +200%.

#### Prinzip 3: Wert in Minuten

Wert in 1-5 Minuten (nicht 30-90 Tagen). Ergebnis: Time to Value -95%.

**Prinzip 4: Virale Loops**

Nutzer laden Nutzer ein. Ergebnis: Viral Coefficient 0,5–1,5.

**Prinzip 5: Self-Serve Upgrade**

Self-Serve Checkout (kein Vertrieb). Ergebnis: Upgrade-Rate +200%.

## 4.3 Wie man PLG aufbaut

**Phase 1: Self-Service Signup bauen (Woche 1-2)**

- Keine Kreditkarte erforderlich (kostenlose Testversion)
- Kein Sales Call erforderlich (Self-Service)
- Keine Demo erforderlich (sofortiger Zugang)

Ergebnis: Anmelderate +200%

**Phase 2: Virale Loops bauen (Woche 3-4)**

- Virale Momente identifizieren (Zusammenarbeit, Teilen, Integrationen)
- Virale Mechaniken bauen (Einladung Flow, Anreize, Benachrichtigungen)
- Virale Loops optimieren (A/B-Test, Viral Coefficient tracken)

Ergebnis: Viral Coefficient 1,0+ (selbsterhaltend)

**Phase 3: Self-Serve Upgrade bauen (Woche 5-6)**

- Upgrade-Trigger bauen (Nutzungslimits, Feature-Gates, Teamgröße)
- Upgrade-Flow bauen (Self-Serve Checkout, sofortiges Upgrade)
- Upgrade-Flow optimieren (A/B-Test Pricing, Conversion Funnel tracken)

Ergebnis: Umsatz/User +50%

## 05 Komponente 2: Shipping Velocity

10x schneller ausliefern (2-4 Wochen vs. 6-12 Monate)

### 5.1 Warum Shipping Velocity wichtig ist

Shipping Velocity = Wie schnell Features ausgeliefert werden.

#### Traditionelle Velocity

- Planung: 2-4 Wochen
- Entwicklung: 8-12 Wochen
- Testing: 2-4 Wochen
- Deployment: 1-2 Wochen

Ergebnis: 1-2 Releases/Quartal

#### AI-Native Velocity

- Planung: 1-2 Tage (AI-unterstützt)
- Entwicklung: 1-2 Wochen (AI Code-Generierung)
- Testing: 1-2 Tage (AI Testing)
- Deployment: 1 Stunde (CI/CD)

Ergebnis: 6-12 Releases/Quartal (+400%)

#### TIME TO MARKET

**-83%**

6-12 Mo. → 2-4 Wo.

#### ITERATION

**10x**

Quartal → Woche

#### WETTBEWERBSVORTEIL

**First**

Vor Wettbewerbern

### 5.2 Die 3 Velocity-Hebel



#### Hebel 1: AI Code-Generierung

GitHub Copilot, Cursor, Replit

Ergebnis: 40-60% Code-Generierung



#### Hebel 2: AI Testautomatisierung

Testim.io, Mabl, Applitools

Ergebnis: Testzeit -90%



#### Hebel 3: CI/CD Automatisierung

GitHub Actions, Vercel, Railway

Ergebnis: Deployment-Zeit -95%

## 5.3 Wie man die Velocity beschleunigt

### Phase 1: AI Code-Generierung implementieren (Woche 1-2)

- AI Coding Tools wählen (GitHub Copilot €10/User, Cursor €20/User, Replit €25/User)
- Team schulen (2-stündiger Workshop, Best Practices, Code Review)
- Impact messen (Codegenerierungsrate 40-60%, Entwicklungszeit -40-60%)

Ergebnis: Team-Adoption 80-90%

### Phase 2: AI Testing implementieren (Woche 3-4)

- AI Testing Tools wählen (Testim.io, Mabl, Applitools)
- AI Testsuite erstellen (AI generiert Testfälle, führt aus, meldet Bugs)
- Testabdeckung optimieren (Ziel: 80%+ Abdeckung, 95%+ Bug-Erkennung)

Ergebnis: Testzeit -90%

### Phase 3: CI/CD implementieren (Woche 5-6)

- CI/CD Pipeline erstellen (GitHub Actions, automatisiertes Testing/Deployment)
- Feature Flags implementieren (schrittweise aktivieren, sofortiges Rollback)
- Deployments monitoren (Ziel: täglich, 99%+ Erfolgsrate, <1% Rollback)

Ergebnis: Wöchentliche Releases, 99% Erfolgsrate

## 06 Komponente 3: Feature Quality

Feature-Adoption 85–95% (vs. 60–70%)

### 6.1 Warum Feature Quality wichtig ist

Feature Quality = % der Features, die Nutzer tatsächlich verwenden.

#### Traditionelle Feature Quality

- Feature Adoption: 60–70% (zu niedrig)
- Feature Usage: 40–50% (zu niedrig)
- Feature Retention: 30–40% (zu niedrig)

Ergebnis: 30–40% der Features ungenutzt

#### AI-Native Feature Quality

- Feature Adoption: 85–95% (+30%)
- Feature Usage: 70–80% (+50%)
- Feature Retention: 60–70% (+75%)

Ergebnis: 85–95% der Features genutzt

#### ENTWICKLUNGSEFFIZIENZ

**+50%**

Nur bauen, was genutzt wird

#### NUTZERZUFRIEDENHEIT

**+30%**

Weniger Bloat

#### PRODUCT-MARKET FIT

**+100%**

Bauen, was Nutzer brauchen

### 6.2 Die 3 Qualitätsprinzipien

#### Prinzip 1: Datengetriebene Entscheidungen

Baue, was Daten zeigen, dass Nutzer brauchen (nicht was Kunden verlangen). Ergebnis: Feature Adoption +30%.

#### Prinzip 2: Schnelle Iteration

Liefern, messen, iterieren (kontinuierliche Verbesserung). Ergebnis: Feature Success Rate +50%.

#### Prinzip 3: Kill Features schnell

Features mit <30% Adoption entfernen, Ressourcen auf High-Adoption Features allokieren.

## 6.3 Wie man Feature Quality verbessert

### Phase 1: Product Analytics implementieren (Woche 1-2)

- Analytics Tools wählen (Amplitude, Mixpanel, Heap)
- Schlüssel-Metriken definieren (Feature Adoption, Usage, Retention)
- Dashboards erstellen (Echtzeit Feature Adoption, Usage Trends, Retention Kohorten)

Ergebnis: Datengetriebene Entscheidungen

### Phase 2: Feedback Loops implementieren (Woche 3-4)

- In-App Feedback erstellen (Feedback Widget, NPS Umfragen, Feature Requests)
- User Research durchführen (User Interviews, Usability Tests)

Ergebnis: Echtzeit Nutzerfeedback

### Phase 3: Rapid Iteration implementieren (Woche 5-6)

- MVPs liefern (80% fertige Features, nicht 100%)
- A/B teste alles (Feature Designs, Flows, Messaging)
- Kill niedrig-adoptierte Features (wöchentlich tracken, <30% entfernen)

Ergebnis: Fokus auf das, was funktioniert

## 07 Komponente 4: Product Analytics

Datengetriebene Produktentscheidungen

### 7.1 Warum Product Analytics wichtig ist

Product Analytics = Datengetriebene Produktentscheidungen.

#### Traditionelle Entscheidungen

- Basierend auf: Kundenwünsche (Bauchgefühl)
- Daten: Quartalsberichte (zu langsam)
- Entscheidungen: Monatlich (zu langsam)

Falsche Features, langsame Iteration

#### AI-Native Entscheidungen

- Basierend auf: Nutzerverhaltensdaten (Echtzeit)
- Daten: Echtzeit-Dashboards (sofort)
- Entscheidungen: Täglich (schnell)

Richtige Features, schnelle Iteration

#### FEATURE ERFOLGSRATE

**+30%**

60-70% → 85-95%

#### ENTSCHEIDUNGS-SPEED

**30x**

Monatlich → Täglich

#### PMF SCORE

**+100%**

0,2-0,4 → 0,6-0,8

### 7.2 Die 5 Analytics Layer

#### Layer 1: User Behavior Tracking

Page Views, Clicks, Events tracken. Tools: Amplitude, Mixpanel, Heap. Ergebnis: Wissen, was Nutzer tun.

#### Layer 2: Feature Adoption Tracking

Feature Adoption, Usage, Retention tracken. Ziel: 85-95% Adoption, 70-80% Usage, 60-70% Retention.

#### Layer 3: User Journey Tracking

Onboarding, Conversion, Retention Flows tracken. Ergebnis: Wissen, wo Nutzer abspringen.

#### Layer 4: Cohort Analysis

Nutzer-, Retention-, Revenue-Kohorten tracken. Ziel: D1 40–60%, D7 30–40%, D30 20–30% Retention.

#### Layer 5: Predictive Analytics

Churn, Upgrade, Feature Adoption prognostizieren. Ergebnis: Proaktive Interventionen.

### 7.3 Wie man Product Analytics aufbaut

#### Phase 1: Tracking implementieren (Woche 1-2)

- Analytics Plattform wählen (Amplitude €2k-10k/Mo., Mixpanel €1k-5k/Mo., Heap €3k-15k/Mo.)
- Event Tracking implementieren (alle Nutzeraktionen, Feature-Nutzungen, Conversions)
- Dashboards erstellen (Feature Adoption, User Journey, Retention Cohorts)

Ergebnis: Vollständige Nutzerverhaltensdaten

#### Phase 2: Analyse implementieren (Woche 3-4)

- Funnels erstellen (Onboarding, Conversion, Retention)
- Kohorten erstellen (Nutzer, Retention, Revenue)
- Alerts erstellen (Churn, Adoption, Conversion Probleme)

Ergebnis: Proaktive Interventionen

#### Phase 3: AI Predictions implementieren (Woche 5-6)

- Churn Prediction Model (AI prognostiziert Churn-Risiko, startet Interventionen)
- Upgrade Prediction Model (AI prognostiziert Upgrade-Wahrscheinlichkeit)
- Feature Adoption Model (AI personalisiert Feature-Empfehlungen)

Ergebnis: Churn -30%, Upgrade-Rate +50%, Adoption +30%

## 08 Komponente 5: AI-Powered Development

AI übernimmt 40–60% der Entwicklungsarbeit

### 8.1 Warum AI-Powered Development wichtig ist

AI-Powered Development = AI übernimmt 40–60% der Entwicklungsarbeit.

#### Traditionelle Entwicklung

- Entwickler schreiben 100% des Codes
- Entwickler schreiben 100% der Tests
- Entwickler reviewen 100% des Codes

Ergebnis: Langsam, teuer

#### AI-Powered Entwicklung

- AI generiert 40–60% des Codes
- AI generiert 80–90% der Tests
- AI reviewed 50–70% des Codes

Ergebnis: Schnell, effizient

#### ENTWICKLUNGS-SPEED

**+100-200%**

AI generiert Code

#### TEAMGRÖSSE

**-50%**

AI übernimmt Arbeit

#### CODEQUALITÄT

**+20%**

AI findet Bugs

### 8.2 Die 5 AI Entwicklungstools

#### Tool 1: AI Code Generation

GitHub Copilot, Cursor, Replit

40–60% Code-Generierung, -40–60% Entwicklungszeit

#### Tool 2: AI Code Review

CodeRabbit, Codacy, SonarQube

+20% Codequalität, +50% Bug-Erkennung, -70% Review-Zeit

#### Tool 3: AI Testing

Testim.io, Mabl, AppliTools

90% Testautomatisierung, -90% Testzeit

#### Tool 4: AI Deployment

GitHub Actions, Vercel, Railway

100% Automatisierung, -95% Deployment-Zeit

#### Tool 5: AI Monitoring

Datadog, New Relic, Sentry

+80% Fehlererkennung, -70% MTTR, 99,9% Uptime

## 8.3 Wie man AI-Powered Development implementiert

### Phase 1: AI Code Generation implementieren (Woche 1-2)

- AI Coding Tools wählen (Copilot €10/User, Cursor €20/User, Replit €25/User)
- Team schulen (2-stündiger Workshop, Best Practices, Code Review Standards)
- Impact messen (40–60% Codegenerierung, -40–60% Entwicklungszeit)

Ergebnis: Entwicklungsgeschwindigkeit +100%

### Phase 2: AI Testing implementieren (Woche 3-4)

- AI Testing Tools wählen (Testim.io, Mabl, Applitools)
- AI Testsuite bauen (AI generiert Testfälle, führt aus, meldet Bugs)
- Testabdeckung optimieren (Ziel: 80%+, 95%+ Bug-Erkennung, <1 Tag Testzeit)

Ergebnis: Qualität erhalten, Geschwindigkeit +10x

### Phase 3: AI Deployment implementieren (Woche 5-6)

- CI/CD Pipeline bauen (GitHub Actions, automatisiertes Testing/Deployment)
- Feature Flags implementieren (schrittweise aktivieren, sofortiges Rollback)
- Deployments monitoren (Ziel: täglich, 99%+ Erfolg, <1% Rollback)

Ergebnis: Wöchentliche Releases, 99% Erfolgsrate

## 09 Case Studies

3 Transformationen: Series A, B & C

### 9.1 Case Study 1: Series A SaaS (€5M → €15M ARR)

**Branche:** B2B SaaS (Developer Tools) • **Phase:** Series A (€5M ARR, 30 Personen)

**Problem:** Shipping Velocity 1 Release/Quartal, PMF Score 0.3

**Zeitrahmen:** 6 Monate

**Die Lösung:** Implementierung des 5-Komponenten-Frameworks:

- **PLG (Monat 1-2):** Self-Service Signup, sofortige Aktivierung, virale Loops → CAC €8k → €2k (-75%)
- **Shipping Velocity (Monat 3-4):** GitHub Copilot, AI Testing, CI/CD → Velocity 1/Q → 6/Q (+500%)
- **Feature Quality (Monat 5-6):** Amplitude, In-App Feedback, Rapid Iteration → Adoption 60% → 85% (+42%)

ARR  
**+200%**  
€5M → €15M

INVESTMENT  
**€30K**  
6 Monate

ROI  
**10x**  
2,5 Mo. Payback

### 9.2 Case Study 2: Series B SaaS (€15M → €50M ARR)

**Branche:** B2B SaaS (Marketing Tech) • **Phase:** Series B (€15M ARR, 80 Personen)

**Problem:** Shipping Velocity 8 Wochen/Release, Tech Debt 40%

**Zeitrahmen:** 12 Monate

**Die Lösung:** Systematische Implementierung aller 5 Komponenten:

- **Phase 1 (Monat 1-6):** Velocity + Quality → Shipping Velocity 8 Wo. → 2 Wo. (-75%)
- **Phase 2 (Monat 7-12):** PLG + Analytics → CAC €10k → €3k (-70%)

ARR

**+233%**

€15M → €50M

INVESTMENT

**€60K**

12 Monate

ROI

**15x**

2 Mo. Payback

### 9.3 Case Study 3: Series C SaaS (€50M → €100M ARR)

**Branche:** B2B SaaS (Data/Analytics) • **Phase:** Series C (€50M ARR, 300 Personen)**Problem:** PLG Motion gebrochen, Shipping Velocity 12 Wochen/Release**Zeitrahmen:** 18 Monate**Die Lösung:** Vollständige Produkttransformation:

- **Diagnose (Monat 1–3):** Produktaudit (100+ Features), Engpässe identifiziert, Roadmap erstellt
- **PLG + Velocity (Monat 4–12):** Onboarding neu aufgebaut, virale Loops, AI Codegenerierung → CAC €15k → €4k (-73%)
- **Quality + Analytics (Monat 13–18):** Produktanalyse, Predictive Models → Adoption 60% → 90% (+50%)

ARR

**+100%**

€50M → €100M

INVESTMENT

**€90K**

18 Monate

ROI

**20x**

2,5 Mo. Payback

## 9.4 Schlüsselmuster über alle 3 Case Studies

### Series A

B2B SaaS (Developer Tools)

**€5M ↗ €15M +200%**

- Shipping Velocity +500%
- CAC -75%
- PMF +100%

⌚ 6 Monate \$ €30K **ROI 10x**

### Series B

B2B SaaS (Marketing Tech)

**€15M ↗ €50M +233%**

- Shipping Velocity +300%
- CAC -70%
- Tech Debt -63%

⌚ 12 Monate \$ €60K **ROI 15x**

### Series C

B2B SaaS (Data/Analytics)

**€50M ↗ €100M +100%**

- Shipping Velocity +500%
- CAC -73%
- Feature Adoption +50%

⌚ 18 Monate \$ €90K **ROI 20x**

### Die Muster

- ✓ Framework funktioniert für Series A-C
- ✓ Shipping Velocity +400–600% (alle erreichten 2–4 Wochen Zyklen)
- ✓ Time to Market -83% (alle erreichten 2–4 Wochen)
- ✓ Feature Adoption +30–50% (alle erreichten 85–90%)
- ✓ PMF Score +75–100% (alle erreichten 0,6–0,8)
- ✓ CAC -70–75% (alle erreichten €2k–€4k)
- ✓ Hypergrowth ohne lineares Engineer-Wachstum

## 10 Implementierungsroadmap & Nächste Schritte

Die 3-Phasige Transformation

### 10.1 Die 3-Phasige Transformation

1

#### Phase 1: Assess (Woche 1-2)

- Diagnose von Engpässen (alle 5 Komponenten bewerten)
- Analyse der Produktmetriken (Velocity, Qualität, PMF)
- Erstellung der Transformationsroadmap (90-Tage-Plan)

Investment: €5K (Product Assessment)

2

#### Phase 2: Build (Woche 3-8)

- Fix Velocity + Quality (Woche 3-6): AI Codegenerierung, CI/CD, Produktanalyse
- Fix PLG + Analytics (Woche 7-8): Self-Service, virale Loops, Kohortenanalyse

Investment: €15K–€30K (Power Up Programm)

3

#### Phase 3: Scale (Woche 9-12)

- AI-Powered Development implementieren (Woche 9–10)
- Optimierung aller Komponenten (Woche 11–12): A/B-Tests, Team Training, eigene AI-Modelle

Investment: €30K–€60K (Boost Programm)

## 10.2 Lösungszuordnung

### Power Up

Product Quick Wins

€15K-€30K • 2-4 Monate

- Komponente 2: Shipping Velocity
- Komponente 3: Feature Quality

Für: Series A-B, Velocity-Krise

### Boost

Product Transformation

€30K-€60K • 6-12 Monate

- Alle 5 Komponenten
- AI-native

Produktmaschine

Für: Series B-C, defektes

Produkt

### Accelerate

Full AI-Native Transformation

€60K-€120K • 12-18 Monate

- Alle 5 Komponenten + Custom AI
- Best-in-Class Produkt

Für: Series C+,

Wettbewerbsvorteil

## 10.3 Wie man beginnt

### Dieses Whitepaper zeigte Ihnen

- ✓ **Das Problem:** Traditionelle Produktentwicklung scheitert bei Skalierung
- ✓ **Das Framework:** 5-Komponenten AI-Native Produkt Framework
- ✓ **Der Beweis:** 3 Case Studies (Series A-C, 10-20x ROI)
- ✓ **Die Ergebnisse:** Velocity +400%, Time to Market -83%, PMF +75-100%

### Die Wahl liegt bei Ihnen:

Die Lücke wird größer. AI-native Unternehmen gewinnen. Was werden Sie wählen?

1. Product Assessment buchen (€5K) • 2. Playbook herunterladen •
3. Inflection Call buchen (kostenlos)

## Über die Autoren



### Michel Lason

Gründer & CEO

*Strategy. Scaling. Impact.*

18 Jahre Startups aufbauen, skalieren und reparieren. Ex-Berater (Microsoft, XING), SaaS Executive (€1,3M → €13,7M ARR in 2 Jahren). Autor "Fix Growth. Scale Faster."

Revenue Architecture

AI/LCNC GTM Motions

Investor Readiness

Rule of 40 +10 Pkt,  
✓ EBITDA –€300k → +  
€150k



### Alban Halili

Partner

*Growth. AI Solutions.  
Automation.*

10+ Jahre B2B Sales  
skalieren. Ex-CSO bei Elba (€8,5M ARR, RPA/AI), Enterprise Sales bei Telefónica (€7,7Mrd).

B2B Sales AI Agents

Automation

Performance Analytics

3,8% Conversion,  
✓ €14,5K Durchschnitts-  
Deals



### Florian Metzger

Partner

*RevOps. GTM. Venture  
Architect.*

4+ Jahre SaaS-Businesses  
aufbauen. Design Thinking (HPI), lasr.io Architekt. Co-  
Founder Mindset.

RevOps GTM Engineering  
Marketing Automation

Sales Cycle –30%,  
✓ Lead Throughput  
optimiert

## Kontakt



team@scalingx.io



scalingx.io



LinkedIn

**Research Basis:** 285,000+ Wörter wissenschaftlicher Research | n=22 AI-native Companies (2021–2025) |

R<sup>2</sup>=0.76, p<0.001

---

© 2026 ScalingX Hypergrowth. All rights reserved.

Version: 1.0 | Datum: 3. Februar 2026

**Disclaimer:** Dieses Whitepaper repräsentiert unser aktuelles Verständnis basierend auf verfügbarer Forschung und praktischer Erfahrung. Das Feld der AI entwickelt sich rapide, und spezifische technische Details können sich ändern. Alle Performance-Claims basieren auf dokumentierten Case Studies und publizierter Forschung. Organisationen sollten ihre eigene Evaluation für spezifische Use Cases durchführen.