

Frankfurt 29.10.2026

Mehr Wert, weniger Code

Warum Nutzerbedürfnisse die Architektur bestimmen sollten

INNOQ

Aminata Sidibe
Principal Consultant

VALUE

VALUE
Und warum es wichtig ist



https://physics.osu.edu/about-ohio-state





Architektur, basierend auf Nutzerbedürfnissen

Wege wurden auf dieser Basis gebaut.

Nicht umgekehrt

Sie haben damit echten Mehrwert erzeugt

Für die Studenten, aber auch für die Universität





Was wir lernen können

- Geplante Strukturen sind lediglich Annahmen.
- Menschen zeigen ihre Bedürfnisse durch ihr Verhalten.
- Echter Wert entsteht, wenn sich Systeme an die tatsächliche Nutzung anpassen.
- Das Material mag dasselbe sein, aber das Ergebnis ist ein anderes.

Meine These:
Wir modernisieren Strukturen, nicht die gesamte Experience

Wir lieben Technologie

**Technologie sollte Wert schaffen,
nicht die Frage nach dem Wert ersetzen.**

Technology als enabler

- Technologie sollte bessere Ergebnisse ermöglichen.
- Architektur sollte den Nutzen für den Nutzer unterstützen.
- Modernisierung sollte das gesamte Nutzererlebnis verbessern.
- Technologie sollte Wert schaffen.

Wert wird geschaffen durch Produkte und Services, die echte Probleme für echte Menschen lösen.

Kurzer „Break“

Was ist ein Produkt?

Was ist ein Produkt?
Ein Treiber für Mehrwert

Gutes Produkt

-> löst ein reales Problem

-> befriedigt ein reales Bedürfnis

-> passt zum Kontext des Nutzers

-> unterstützt Geschäftsziele

Schlechtes Produkt

-> beginnt mit einer Lösung

-> ignoriert den Kontext

-> fügt irrelevante Funktionalität hinzu

-> schafft keinen messbaren Wert

Gut oder schlecht ist keine Frage des Geschmacks.








1329
949
891



https://www.instagram.com/the_very_nature/



[Via boredpanda.com]



© mrwhitedynamite / reddit



© BKStephens / reddit



© Caesar95 / reddit



https://careerfoundry.com/

Gebaut != nützlich.

- Das sind keine reinen Implementierungsfehler
- Es sind teilweise Kontextfehler
- Es gibt eine Lösung aber der Nutzen ist nicht erkennbar
- Das Ergebnis ist Komplexität ohne Nutzen.

Gleiches gilt für die Entwicklung digitaler Produkte.

Digitale Produktentwicklung

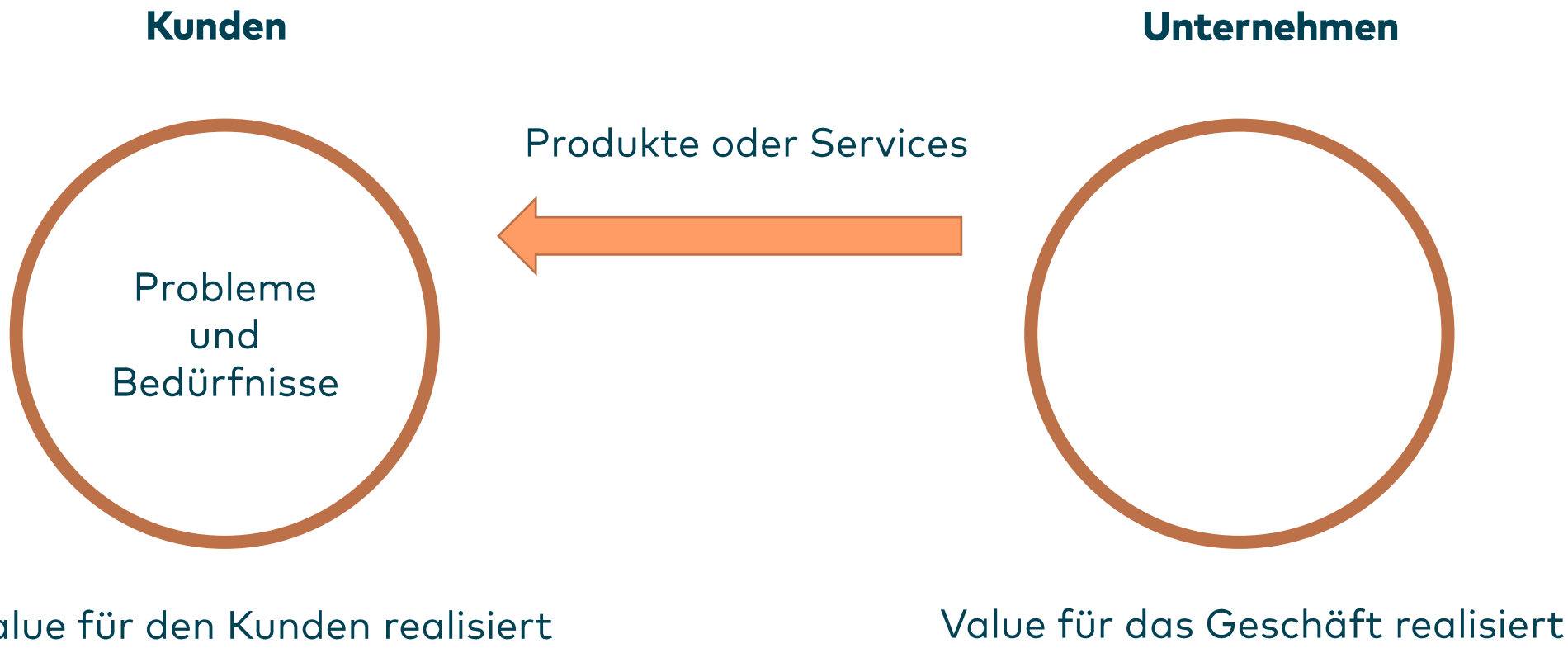
- Mangelhafte Informationsarchitektur
- Barrierefreiheit wird ignoriert
- Design vor Funktion (UI ≠ UX)
- Entwicklung unnötiger Funktionen
- Unlogische/nutzlose Funktionen
- Schlechte Lösungen

- …

Unser Ziel:

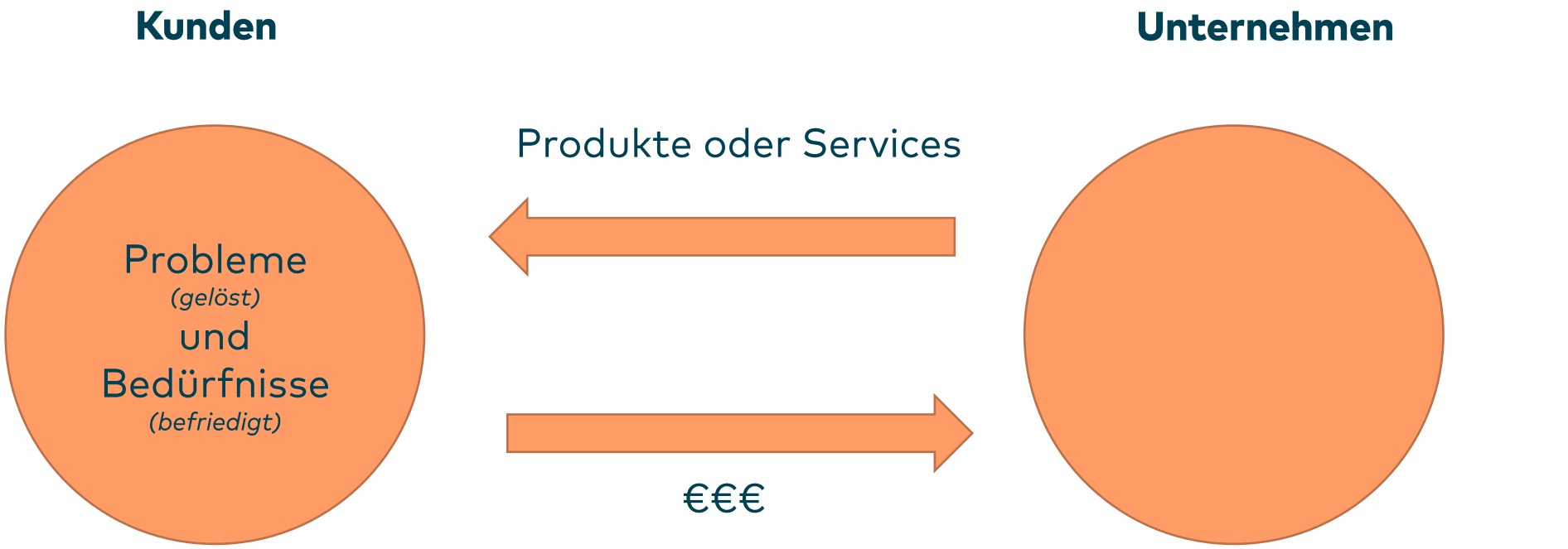
Nicht einfach nur Dinge bauen, sondern Mehrwert schaffen

Value Exchange System



© Escaping the Build Trap by Melissa Perri

Value Exchange System



© Escaping the Build Trap by Melissa Perri

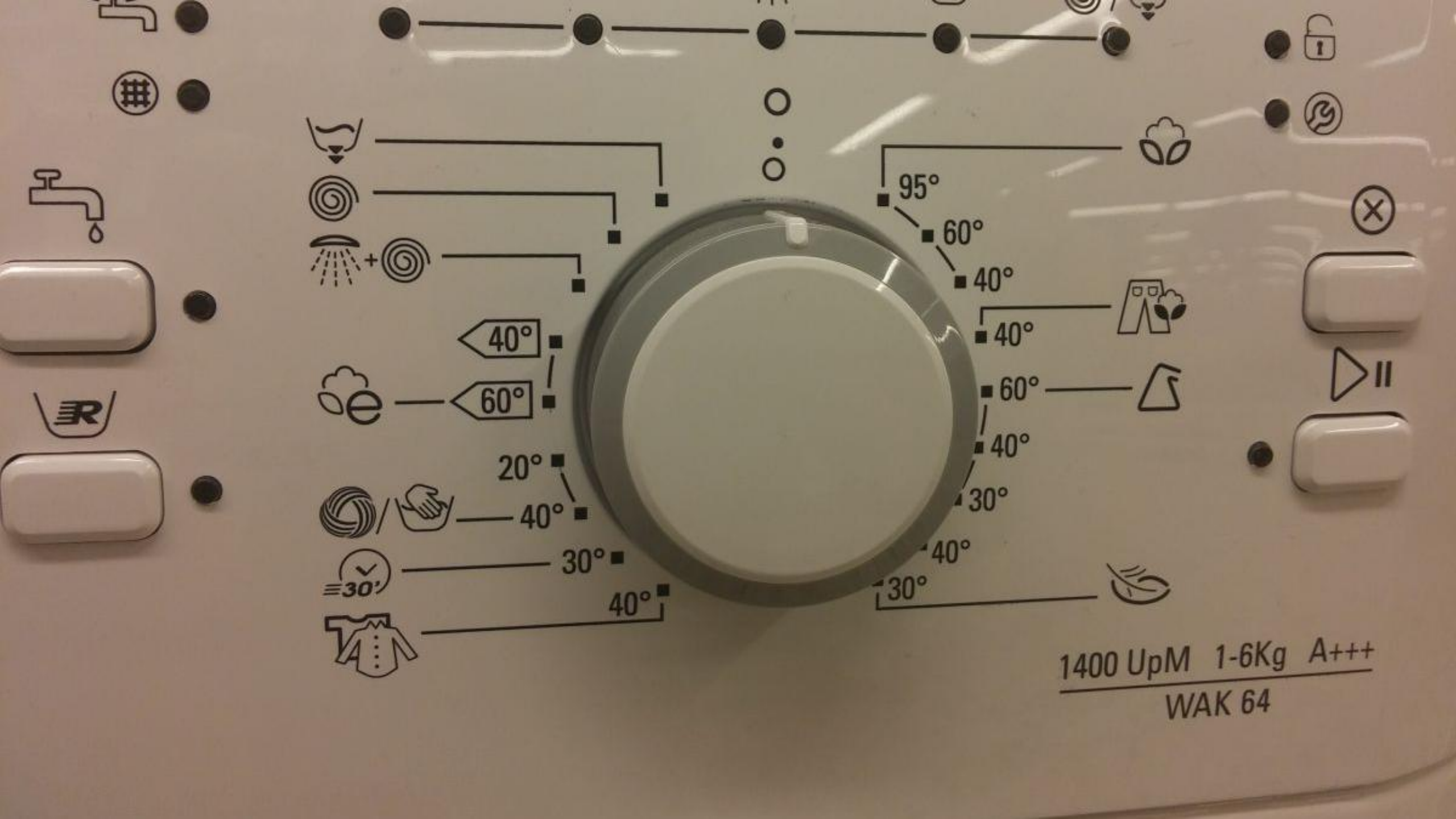
Value != Feature count

Die Anzahl der Features ist ein schwacher Indikator für den Wert

The build trap

Output (Feature Factory)

- Der Erfolg wird am Output gemessen: ausgelieferte Features, Roadmap-Items, Velocity.
- Der Fokus verlagert sich von der Wertschöpfung hin zur Auslieferung.
- Teams sind mit shipping beschäftigt, lernen aber nicht aus den Ergebnissen.
- Technisch einwandfreie Überarbeitungen führen nicht automatisch zu einem besseren Produkt.


95°
60°
40°
40°
60°
40°
30°
40°
30°
40°
60°
20°
40°
30°
40°
1400 UpM 1-6Kg A+++
WAK 64


TV
CABLE
POWER
AUX
VCR
HELP
LOCK
MUTE
VOL
EDIT
INFO
OK
GUIDE
MENU
CHAN
PAGE
LAST
DAY
FAV
1
2
3
4
5
6
7
8
9
TV/VCR
0
ENTER
AT&T
POWER
LIGHT
CH
MUTE
DISPLAY
JUMP
ENTER
CHANNEL GUIDE
REPLAY GUIDE
SELECT
EXIT
PAGE
REC
REW/SR
PLAY/SLOW
FF/SR
PAUSE
STOP
replaytv
SONICblue
DVD
Pioneer
MUTING
SLEEP
POWER
DISPLAY
TV/VIDEO
RESET
MENU
SONY
TV
CHANNEL
MULTI CONTROL
FUNCTION
DIRECT
MODE CHECK
Pioneer
PLAY
REW
FF
STOP
ENTER
VOL
Panasonic
LIGHT TOWER
UNIVERSAL

KI beschleunigt die Feature Factory


We need to improve our product
Add AI
More AI
Understand the customer first

GenAI.works

AI als feature factory accelerator

- Gen AI senkt die Implementierungskosten.
- Ideen können schneller zu Funktionen werden als zuvor.
- Doch eine schnellere Implementierung verbessert das Produkt nicht.
- Ohne Outcome-Fokus hilft uns KI dabei, schneller die falschen Dinge zu entwickeln.

KI macht gute Produktentscheidungen wichtiger, nicht weniger wichtig.

Unser Ziel:

Nicht einfach nur Dinge bauen, sondern Mehrwert schaffen

Output minimieren

Outcome und impact maximieren

Von output zu impact

Von output zu impact

Product discovery

Vonn output zu impact

Outcome-Driven architecture

Outcome-Driven architecture

- Bedeutet: Systeme so zu gestalten, dass sie die angestrebten Ergebnisse (Outcomes) ermöglichen.

- Sie verknüpft architektonische Entscheidungen mit:

  - Nutzerbedürfnissen
  - Produktergebnissen
  - Annahmen
  - Experimenten
  - Lernprozessen
  - Anpassungsfähigkeit

Architektur sollte messbaren
impact erzeugen,
nicht nur Features liefern

7 Prinzipien für größeren impact und weniger feature factory

Principle 1:
Start with outcomes, not features

**Beginne nicht mit der Lösung,
beginne mit dem gewünschten Ergebnis.**

Don‘ts

- „Wir brauchen ein Dashboard“
- „Wir brauchen eine neue Suche“
- „Wir brauchen KI-Funktionen“
- „Wir brauchen einen Chatbot“
- „Wir brauchen eine App“
- …

Das ist bereits die Lösung.

Das führt zu:

- Schnellen Aktivitäten
- Übereilten, technischen Entscheidungen

Welche Frage offen bleibt:

Welche Veränderung soll eigentlich als Ergebnis dabei herauskommen?

Das wollen wir ändern

Von: Was wollen wir bauen?

Das wollen wir ändern

Zu: Was wäre besser?

Für den Nutzer, den Kunden oder das Unternehmen?

Ein Outcome beschreibt eine beobachtbare Veränderung.

Beispiele für outcomes

- Der Nutzer findet relevante Informationen schneller
- Support-Anfragen zu einem bestimmten Prozess nehmen ab
- Mitarbeiter führen weniger manuelle Zwischenschritte aus
- Die Wahrscheinlichkeit, dass Nutzer einen kritischen Prozess abbrechen, sinkt
- Entscheidungen werden schneller oder in höherer Qualität getroffen
- …

Das verändert die Architektur-Arbeit.

Beispiel Outcome:

Nutzer sollen schneller relevante Inhalte finden.

Nicht: „Wir brauchen eine neue Suche“

Fragen die man stellen kann

- Können wir messen, ob Nutzer schneller finden, wonach sie suchen?
- Können wir Suchbegriffe, die Qualität der Suchergebnisse sowie Abbruchquoten nachverfolgen?
- Können wir verschiedene Ranking-Ansätze testen?
- Können wir feststellen, ob die Suchfunktion tatsächlich hilfreich ist oder ob sie nur deshalb genutzt wird, weil die Navigation mangelhaft ist?
- Können wir qualitatives Feedback mit Nutzungsdaten verknüpfen?

Der Outcome verändert,
was für die Architektur relevant ist

Auswirkungen auf die Architektur

Architectural implications

Wenn wir möchten, dass Nutzer relevante Inhalte schneller finden, muss uns das System dabei unterstützen:

- zu beobachten, wonach Nutzer suchen
- zu messen, ob die Ergebnisse nützlich sind
- zu erkennen, an welcher Stelle Nutzer die Suche abbrechen
- verschiedene Ranking-Ansätze zu testen
- Feedback von den Nutzern einzuholen
- die Suche und Navigation im Laufe der Zeit zu verbessern

Die Architektur sollte unterstützen, was wir lernen und verbessern müssen.

Principle 1:
Start with outcomes, not features

Principle 2:
Make impact observable

Wenn ein Impact nicht beobachtbar ist, bleibt es eine Meinung.

Viele Systeme sind technisch beobachtbar.

Wir wissen, ob sie laufen.
Wir kennen Antwortzeiten, Fehlerraten, CPU-Auslastung, Logs, Traces und Deployments, ...

Outcome-Orientierung ergänzt die Frage:

Erzeugt das System
eine relevante Wirkung?

Ein technisch stabiles System kann geringen Value schaffen.

Es kann ohne Relevanz skalieren.

**Es kann viele Features enthalten,
ohne das Problem für den Nutzer zu lösen.**

Systeme müssen beobachtbar sein
- nicht nur aus technischer Sicht,

sondern auch im Hinblick auf
die geschäftliche und produktbezogene
Performance.

Das bedeutet

- Wir erfassen relevante Nutzeraktionen.
- Wir verstehen kritische Prozessschritte.
- Wir erkennen, an welchen Stellen Nutzer aussteigen.
- Wir identifizieren, welche Funktionen genutzt werden – und welche nicht.
- Wir messen nicht nur Klicks, sondern aussagekräftigen Fortschritt.
- Wir verknüpfen Produktkennzahlen mit Geschäftszielen.

Technische Observability verrät uns, ob das System läuft.

Product-Observability verrät uns, ob das System von Bedeutung ist.

Principle 3:
Design for Experiments

From big releases to learning cycles

- Outcome-Driven architecture ermöglicht:
  - kleine Änderungen
  - schnelles Feedback
  - schrittweise Investitionen
  - sicheres Rollback
  - einfaches Entfernen nicht benötigter Features

Die meisten Produktideen sind Annahmen.

Produkt ideen => Annahmen

- „Wir glauben, dass die Nutzer dies brauchen.“
- „Wir glauben, dass sie es verstehen werden.“
- „Wir glauben, dass es das Verhalten verändern wird.“
- „Wir glauben, dass es einen Mehrwert schaffen wird.“

Architekture sollte enablen:

- Prototypen
- „Fake-Door“-Tests
- Feature-Toggles
- Beta-Gruppen
- Kontrollierte Rollouts
- A/B-Tests
- Sichere Rollbacks

Architektur sollte
Lernen nicht teuer machen.

Principle 4:
Optimize for reversibility

**Je unsicherer eine Annahme ist,
desto leichter sollte sie sich ändern lassen.**

Stabile Entscheidungen

- Domänenmodell
- Compliance-Regeln
- Kritische Integrationen
- Plattform-Einschränkungen

Flexible Entscheidungen

- Arbeitsabläufe
- UI-Varianten
- Onboarding-Schritte
- Inhalte
- Funktionsdetails
- …

frühe Produktannahmen sollten
nicht in starre architektonische Vorgaben
verwandelt werden.

Reversibility is Risk Management

- Outcome-Driven teams fragen:
  - Wie sicher sind wir uns?
  - Was ist noch eine Hypothese?
  - Was muss anpassungsfähig bleiben?
  - Was sollten wir nicht zu früh festlegen (bzw. „hart“ implementieren)?
  - ..

Gute Architektur bewahrt Teams davor, sich zu früh auf falsche Annahmen festzulegen.

Principle 5: Connect decisions to assumptions

Architekturentscheidungen sollten nicht nur darlegen:
„Warum ist es technisch sinnvoll?“

What it should also explain

- Welchen Outcome unterstützt diese Entscheidung?
- Welches Nutzerbedürfnis ist damit verbunden?
- Welche Annahme treffen wir?
- Woran werden wir erkennen, ob es geholfen hat?

Architekturentscheidungen sollten anhand von Outcomes überprüfbar sein.

Outcome-driven ADRs

Outcome-driven ADRs

- Angestrebtes Ergebnis
- Nutzerbedürfnis
- Produktannahme
- Erwartetes Signal
- Erfolgsindikator
- Überprüfungszeitpunkt
- Reversibilität

Beispiel

„Wir setzen Event-Streaming ein, da wir Geschäftsereignisse nahezu in Echtzeit beobachten müssen, um Abbrüche im Bewerbungsprozess zu erkennen und zu verringern.“

Architekturdokumentation sollte technische Entscheidungen mit Erkenntnisprozessen verknüpfen.

Principle 6:
Treat Data as Feedback Infrastructure

Daten sind nicht bloß reporting.

Es ist eine Infrastruktur für das Lernen.

Teams brauchen

- Produktanalytik
- Datenqualität
- Datenschutz und Einwilligung von Grund auf (Privacy and Consent by Design)
- Zugang zu relevanten Erkenntnissen
- Verknüpfung von Produkt-, Nutzer- und Geschäftsdaten

Wenn wir Wirkung erzielen wollen,
müssen wir auf Feedback ausgelegt gestalten.

Beyond dashboards

Mehr Daten bedeuten nicht automatisch bessere Entscheidungen.

Beyond dashboards

Frag nicht: was können wir tracken

Beyond dashboards

Frage: Welche Signale helfen uns zu verstehen, ob es den Nutzern besser geht?

Beyond Dashboards

- Beispiele:
  - Haben die Nutzer die Aufgabe abgeschlossen?
  - Haben sie schneller einen Mehrwert erzielt?
  - Sind die Support-Anfragen zurückgegangen?
  - Hat sich das Verhalten in die gewünschte Richtung verändert?

Daten werden wertvoll,
wenn sie mit Outcomes verknüpft sind.

Principle 7: Make it a Team Responsibility

Team responsibility

- Outcomedrive-architecture erfordert Zusammenarbeit
  - Produkt: Welche Auswirkungen sind von Bedeutung?
  - UX & Forschung: Welche Nutzerbedürfnisse sind real?
  - Entwicklung: Welche Änderungen sind technisch umsetzbar?
  - Architektur: Welche Systemfähigkeiten werden benötigt?
  - Daten: Welche Signale zeigen Fortschritt an?
  - Geschäftsbereich: Welche Ergebnisse sind strategisch relevant?

Jede technische Entscheidung verändert, was die Organisation lernen kann.

Architektur in die Product discovery einbringen

It will help teams to ask earlier

- Was müssen wir lernen?
- Welche Annahmen sind riskant?
- Welche Änderungen sollten reversibel bleiben?
- Welche Fähigkeiten muss das System bereitstellen?
- Welche Komplexität sollten wir vermeiden?

THE SHIFT

„Shift from a culture of delivery to a culture of learning."

(Jeff Gothelf)

Shift from

shipping features

Shift from

optimizing output

Shift from

Measuring system health only

to

Enabling outcomes

to

Optimizing learning

to

Measuring product impact

Outcome-Driven Architecture turns architecture into a lever for product impact.

GREAT SYSTEMS
CREATE IMPACT

Good systems are not defined by

How much they ship

How much code they contain

How many features they offer

Good systems are defined by

The value they create

The needs they serve

The learning they enable

The impact they make

Build less noise
Build more value

Lead by example

Entwickelt Software,
Produkte und Services für
Menschen, die sie benutzen

Danke für‘s Zuhören ☺

Fragen? Feedback?

INNOQ
www.innoq.com



Aminata Sidibe
aminata.sidibe@innoq.com

LinkedIn: https://www.linkedin.com/in/aminatasidibe/

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Wendenstraße 130
20573 Hamburg

Spichernstrasse 44
50672 Köln