



ZEIT ZUM MARKT





ZEIT ZUM MARKT





ZEIT ZUM MARKT



[ MVP ]



Erreichen Sie Ihr **Minimum Viable Product** , um Ihr Konzept zu testen.





## ZEIT ZUM MARKT

[ MVP ]

Der **Umfang** deines MVP muss **reduziert** sein, während du dein Produkt vermarkten kannst.

Setzen Sie auf **Early Adopters** und erhalten Sie ein Maximum an **Feedback**.

Ihr MVP ist in **Produktion** implementiert und verwendbar.





ZEIT ZUM MARKT



[ FAIL-SCHNELL ]



**Fail** Fast ist **learn** schnell.





ZEIT ZUM MARKT

[ FAIL-SCHNELL ]

**Erleben Sie schnell** die Lösung (ein paar Wochen), sammeln Sie **Feedback** von Ihren Benutzern und lernen Sie aus Ihren Fehlern.

Hab keine Angst, alles zu ändern \*\*.

Vergiss nicht, **du wirst scheitern!**









\*\* Halten Sie es einfach  
und dumm.

- Warum kompliziert,  
wenn es einfach sein  
kann? \*



ZEIT ZUM MARKT

[ KUSS ]

**Vermeiden Sie zu viel Technik** ,  
wenn ein "Papier" -Modell oder ein  
Google-Formular ausreicht, um Ihr  
Konzept zu testen, gehen Sie nicht  
weiter.

Bleib einfach! Sowohl technisch als  
auch funktional.





ZEIT ZUM MARKT



[ PRODUCTIVITY ]

Geben Sie weniger an,  
**erweitern Sie mehr.**





ZEIT ZUM MARKT

[ PRODUCTIVITY ]

Begrenzen Sie Ihre Spezifikationen auf das Wesentliche,

**konzentrieren Sie sich auf das "Was"** und nicht auf das "Wie".

Das Produkt muss möglichst **selbst-dokumentiert** sein.

Die Dokumentation muss genauso wie der Code versioniert werden.





ZEIT ZUM MARKT



[ SAAS ]

Systematische  
Untersuchung von **SaaS** -  
Lösungen







ZEIT ZUM MARKT

[ SAAS ]

**SaaS** -Lösungen sind **nachhaltig und kosteneffektiv**.

In einigen Fällen kann **SaaS die Implementierung von MVP\*\*\*\*** beschleunigen.

Denken Sie an die ökonomische Vision **in Bezug auf die Alternativen in Bezug auf Gesamtkosten ( TCO : T otal C ost of O \*\* Wernship )** und nicht nur hinsichtlich der Lizenzkosten.





ZEIT ZUM MARKT



[ Herz des Geschäfts ]

Das **Kerngeschäft** sollte den Aufbau neuer Dienste und Anwendungen nicht behindern.





## ZEIT ZUM MARKT

[ Herz des Geschäfts ]

Das Tempo der Entwicklung und Lieferung des Kerngeschäfts muss **mit der Agilität** der Dienste, die es verbrauchen, vereinbar sein.

Das Kerngeschäft muss **Dienstleistungen anbieten.**

Das Kerngeschäft muss das **Event-Driven** -Prinzip übernehmen, es berichtet über Management-Aktionen in Form von Events.





ZEIT ZUM MARKT



[ KONTINUIERLICHE  
VERWENDUNG ]

**Bereitstellung in  
Produktion** ist ein Nicht-  
Ereignis.





ZEIT ZUM MARKT

[ KONTINUIERLICHE  
VERWENDUNG ]

Profitieren Sie von der  
**kontinuierlichen Bereitstellung** ,  
um die \*\*\* Produktion \*an\*  
geschäftliche \*\* Anforderungen  
anzupassen und nicht umgekehrt.

**Bereitstellungen** in Umgebungen,  
bis zu **Produktion** , müssen  
**automatisiert** und **häufig** sein.







ZEIT ZUM MARKT



[ PERPETUAL BETA ]

Der **Perpetual Beta** -  
Ansatz ermöglicht es  
Ihnen, Ihre Benutzer in den  
Entwicklungsprozess  
einzubeziehen.





ZEIT ZUM MARKT

[ PERPETUAL BETA ]

Fühlen Sie sich frei, das ewige Betaprinzip zu verwenden, in dem **Benutzer an der Entwicklung teilnehmen.**

Der Begriff "Perpetual Beta" bezieht sich auf eine Anwendung, die in Just-in-Time entwickelt wurde **sich ständig weiterentwickelt** und kein unvollständiges Produkt.







# BENUTZERERFAHRUNG





# BENUTZERERFAHRUNG







Das vom Benutzer  
wahrgenommene \*\*  
Erlebnis ist grundlegend.

**Ergonomie** ist nicht  
verhandelbar.



## BENUTZERERFAHRUNG

[ WAHRNEHMUNG ]

Vernachlässige nicht die Arbeit von **UX Designern** , es ist grundlegend in der Entwicklung einer Anwendung.

Integrieren Sie das **Feedback** Ihrer Benutzer, dieses ist wichtig.









Verwenden Sie  
**leistungsfähige**  
**Schnittstellen** für interne  
und externe  
Anwendungen.



## BENUTZERERFAHRUNG

### [ PERFORMANCE ]

Die Schnittstellen sind auf **Effizienz** ausgerichtet.

Die **Leistung** einer Schnittstelle spart Zeit , **erhöht** die Benutzerzufriedenheit **und spart somit Frustration.**





BENUTZERERFAHRUNG



[ MOBILE ZUERST ]

Verabschiedung einer  
**Mobile First** Strategie.





BENUTZERERFAHRUNG

[ MOBILE ZUERST ]

Mobile Geräte sind der **wichtigste** Teil des **Marktes**.

Mobile Denken denkt an das **Wesentliche**.

**Responsive Design** ist die Norm, es ist eine Quelle der Einsparungen (**MVP**).







Anpassung an Nutzungen,  
der **Omni-Channel** ist die  
Norm.





## BENUTZERERFAHRUNG

### [ OMNI-KANAL ]

Der Omni-Channel-Ansatz bietet dem Benutzer eine **einheitliche Erfahrung** (Beispiel: Netflix).

Die verschiedenen **Kanäle** sind **synchronisiert** und **kohärent** (im Gegensatz zu Batch-Prozessen).

Alle Akteure (Kunden, Berater) greifen auf dieselben Informationen zu.







\*\*\*\* Benutzer *sind* Besitzer  
\*\* ihrer Daten und ihres  
Kurses.



## BENUTZERERFAHRUNG

### [ SELBST-DATEN ]

Lassen Sie **Personen jederzeit** ihre **persönlichen** Daten \*\* kontrollieren.

Stellen Sie ein **Vertrauen** her, indem Sie Benutzern die Rückverfolgbarkeit und Kontrolle in Echtzeit ermöglichen.

**Subsysteme** müssen die gleichen Anforderungen erfüllen.







Die Kundenbeziehung muss mit einem flexiblen, einheitlichen und ereignisgesteuerten **CRM / SFA vereinheitlicht und kontextualisiert werden.**



## BENUTZERERFAHRUNG

[ CRM / SFA ]

Entscheiden Sie sich für ein **CRM** , das sowohl die Kundenbeziehung als auch die Führung des Verkaufspersonals verwaltet (**SFA** : **S**ales **F**orce **A**utomation) ).

**CRM** muss **offen** für neue Möglichkeiten sein.

**CRM** erzeugt **Ereignisse** , die Verwaltungsaktionen entsprechen, die in die **ereignisgesteuerte** Logik der Plattform passen.









Die Big Data-Plattform ermöglicht Ihnen die Zentralisierung **und die Verarbeitung von Benutzerdaten, um Ihre Reise \*\* optimal zu unterstützen.**



## BENUTZERERFAHRUNG

[ GROSSE DATEN ]

Zentralisieren Sie die **Maif Group** -, **Partner** - und **Vendor** -Daten in einer **Pathway** -Logik.

"Datenvorbereitung" und Verarbeitung können die Daten \*\* konsolidieren.

**Big Data-Teams kooperieren** mit Feature-Teams, um **Daten** - Governance sicherzustellen.







[ Arbeitsplatz ]

Die Workstation ist  
angepasst und an  
**Anwendungen** und  
**moderne Kanäle**  
anpassbar.



## BENUTZERERFAHRUNG

[ Arbeitsplatz ]

Übernehmen Sie den **Identity Federation** für eine einheitliche Erfahrung.

Ein **Portal** ermöglicht eine **Übersicht**, es ersetzt keine Anwendungen.

Die Workstation muss **mobil**, **Mehrkanal** und **Standard** sein, um das Öffnen innerhalb des **erweiterten Unternehmens** zu ermöglichen.







Vergiss nicht, dass deine  
**Mitarbeiter** moderne  
Anwendungen zu Hause in  
UX verwenden.



## BENUTZERERFAHRUNG

[ MITWIRKENDEN ]

Behandle **all deine Nutzer als "Kunden"** : Internetnutzer, Manager, Betreiber, Entwickler, etc ...

Unterschätzen Sie den **UX-Aufwand** nicht, den Sie für interne Verwaltungsanwendungen implementieren müssen.









Alles was gemessen  
werden kann muss sein.

**Ohne Maß, alles ist nur  
Meinung.**



## BENUTZERERFAHRUNG

[ ALLE MESSUNGEN ]

Denken Sie an die Metriken während der **Entwicklung** der Anwendung. **Logs** müssen sowohl eine geschäftliche als auch eine technische \*\* Dimension aufweisen.

Vernachlässige nicht die **Leistungsmetriken** , sie sind fundamental.

Das Feature-Team stellt **Operation** zur Verfügung: Es ist dafür verantwortlich, **die** Anwendung nutzbar zu machen.







[ A / B-Prüfung ]

**A / B Testing** spart Ihnen  
Zeit, indem Sie **Feedback**  
entscheiden lassen.





## BENUTZERERFAHRUNG

[ A / B-Prüfung ]

Anstatt zwischen zwei Lösungen willkürlich zu entscheiden, zögern Sie nicht, **A / B-Tests** einzurichten.

Dieses Muster besteht aus der Darstellung von **zwei verschiedenen Versionen** derselben Anwendung und der Auswahl einer davon basierend auf **objektiven Kennzahlen** der Benutzeraktivität.







**Berücksichtigen Sie die  
Verschlechterung** und  
nicht die  
Betriebsunterbrechung im  
Falle eines Fehlers.





BENUTZERERFAHRUNG

[ VERSCHLECHTERUNG ]

Bei **Ausfall** eines der Subsysteme **muss eine degradierte** Version des Services \*\* in erster Linie als eine Unterbrechung angesehen werden.

Mit **Circuit Breakers** , **Isolieren Sie eine Aufschlüsselung** bis **Vermeiden Sie ihre Auswirkung** und **Aufspreizung** über das gesamte **System**.







HUMAN





HUMAN





HUMAN



[ FEATURE TEAM ]



Das Team ist um ein  
**Produkt** oder einen  
**Service** herum organisiert.





HUMAN

[ FEATURE TEAM ]

Teams sind **Feature Teams**, die um einen zusammenhängenden Funktionssatz herum organisiert sind und aus all den **Fertigkeiten** bestehen, die für diesen Satz notwendig sind.

Zum Beispiel: Business Expert + Webentwickler + Java Developer + Architect + DBA + Operational.

Die **Verantwortung** ist **kollektiv**, das Feature-Team hat die Macht, die für diese Verantwortung notwendig ist







HUMAN



[ 2-PIZZA TEAM ]

Begrenzen Sie die **Größe**  
**von Feature Teams** (von 5  
bis 12 Personen).





HUMAN

[ 2-PIZZA TEAM ]

Begrenzen Sie die Größe eines Feature-Teams: **zwischen 5 und 12 Personen.**

Unter 5 ist sie zu empfindlich für externe Ereignisse und es fehlt ihr an Kreativität. Über 12 verliert es an Produktivität.

Der Begriff "**2-Pizza-Team**" zeigt an, dass die Größe des Feature-Teams die Anzahl der Personen, die mit zwei Pizzen gefüttert werden können, nicht überschreitet.





HUMAN



[ Künstliche Software ]



Wetten Sie auf vielseitige  
Leute, die **wissen** und wer  
**gerne tun.**





HUMAN

[ Künstliche Software ]

Am wichtigsten ist die **Kultur der Entwicklung, Skalierbarkeit** und **Anpassungsfähigkeit**.

Recruiting **Software-Handwerker und Full-Stack-Entwickler** ,  
bringen sie einen echten Mehrwert  
durch ihr Know-how und ihre  
allgemeine Vision.

Mobile Entwickler sind zum  
Beispiel in der Regel **spezialisierte  
Entwickler**.





HUMAN



[ STELLEN ]

Sei **attraktiv** um den  
**besten** zu rekrutieren.





HUMAN

[ STELLEN ]

Stellen Sie Arbeitsweisen vor, die an die Mitarbeiter angepasst sind:

**Mobilität, Heimarbeit, CYOD**

(Choose Your Own Device.

Lassen Sie Zeit für Experimente und machen Sie es **in der Arbeitszeit** möglich.







HUMAN



[ EVE ]

Die Organisation muss eine  
**Schlafmaschine sein**

Der Tag davor ist Teil des  
Jobs.





HUMAN

[ EVE ]

Die Organisation muss eine **Tagespflege** -Maschine sein, indem sie Systeme wie **Weiterbildung** oder **Business-Universitäten** einrichtet.

Fühlen Sie sich frei, sie mit anderen **informellen** Möglichkeiten zu kombinieren, wie: **Coding Dojos**, **Brown Bag Lunchs**, **externe Konferenzen**.





HUMAN



[ CO-KONSTRUKTION ]

Brechen Sie die Barrieren  
zwischen den Trades,  
wetten Sie auf die  
**Konvergenz** -Ziele.





HUMAN

[ CO-KONSTRUKTION ]

Um die Barrieren zwischen den Gewerken zu überwinden, reicht es nicht aus, Menschen an einem gemeinsamen Ort um ein gemeinsames Produkt herum zu gruppieren.

Die **Agile Approaches** beseitigen diese Hindernisse, um eine Konvergenz der Ziele zu gewährleisten \*\*.

Diese Praktiken sind ein wesentlicher Bestandteil der Schlüssel zum Erfolg, die



Organisation ist der Garant



HUMAN



[ DEVOPS ]

**DevOps** -Praktiken ermöglichen es Wänden, zwischen Build und Run zu fallen.







HUMAN

[ DEVOPS ]

Adoptiere **DevOps** , um **Dev** und **Ops** zu einem gemeinsamen Ziel zusammenzuführen: **Diene der Organisation.**

**Die Trades bleiben anders !**

DevOps bedeutet nicht, dass dieselbe Person die Aufgaben von Dev und Ops ausführt. **Entwickler** und **Operational** müssen zusammenarbeiten , **um** von \*\*\*\*

Skills *\*profitieren\** und **Empathie** zu verbessern.





HUMAN



[ PAIN ]



**Schwierige Aufgaben**  
werden **vom Feature-**  
**Team** durchgeführt.

Die Automatisierung folgt.





HUMAN

[ PAIN ]

In einer traditionellen Organisation hängt **mangelndes Verständnis** zwischen Teams normalerweise mit Entfernung und \*\* mangelnder Kommunikation zusammen.

Die **Mitglieder eines Feature-Teams** sind **mitverantwortlich** und **solidarisch** für alle Aufgaben.

**Schmerz** ist ein Schlüsselfaktor für **Kontinuierliche Verbesserung**.





HUMAN



[ CDS ]

Die Servicezentren sind  
schwer mit der **kollektiven**  
**Verpflichtung** zu  
vereinbaren.





HUMAN

[ CDS ]

Feature Teams basieren auf Prinzipien, die stark von **Collaboration** und **kollektivem Engagement** abhängen.

Die Dienstleistungszentren streben eine Rationalisierung und Konsolidierung der IT durch die Unternehmen an, was dieser Vorstellung von kollektivem Engagement widerspricht.







HUMAN



[ VALIDIERUNG ]

Die Organisation hat die  
**Validierungsfunktion**,  
ohne dogmatisch zu sein.





HUMAN

[ VALIDIERUNG ]

Stellen Sie sicher, dass die Organisation ihre **Validierungsrolle** für Tools und Verwendungen beibehält. Insbesondere zu den **Werkzeugen, die das Erbe betreffen** (Beispiel: Verwaltung des Quellcodes).

**Bieten** Feature-Teams mit **bedeutet** , ihre Auswahl zu unterstützen.

Sei nicht dogmatisch und versichere dich, dass du Experimente   regeln kannst.



HUMAN



[  
Transversalitätsbedingung  
]



Von den Feature Teams  
wird erwartet, dass sie  
**kommunizieren** und ihre  
**Erfahrung** und  
**Fähigkeiten** teilen.





HUMAN

[  
Transversalitätsbedingung  
]

Schaffe keine Barrieren zwischen  
**Feature Teams**.

Richten Sie eine **Organisation** und die **Agilität** ein, die für Feature-Teams erforderlich ist, um miteinander zu kommunizieren und ihre Fähigkeiten und Erfahrungen zu teilen.

Die Organisation der Transversalität in **Spotify** (Stämme, Kapitel und Gilden) ist ein beredtes Beispiel







# INTEROPERABILITÄT





# INTEROPERABILITÄT







INTEROPERABILITÄT



[ API für alle ]



**APIs für alle**

**Anwendungen** : intern,  
Kunden und Partner,  
öffentlich.





## INTEROPERABILITÄT

[ API für alle ]

Öffnen Sie Ihre Organisation für neue Anwendungen und neue Kunden mit **Public APIs**.

In **kommerziellen** Partnerschaften , Kunden **als** Provider \*\* sind APIs das Standardaustauschformat.

**APIs** sollen auch für **interne Verwendungen** der Organisation verwendet werden.





INTEROPERABILITÄT



[ SELBSTBEDIENUNG ]



Verwenden Sie eine API  
muss **einfach** und **schnell**  
sein.





## INTEROPERABILITÄT

### [ SELBSTBEDIENUNG ]

Die Verwendung von APIs sollte so einfach wie möglich sein. Denken Sie an die **Entwicklererfahrung**.

Die beste Lösung, um die Angemessenheit mit der Notwendigkeit zu überprüfen, ist, **die API schnell zu testen** : ein paar Minuten müssen genügen!

Die Plattform muss eine **grafische Schnittstelle** bieten, um die API einfach zu testen.







Die Verwendung der APIs  
muss **kontrolliert** und  
**kontrolliert** werden.





## INTEROPERABILITÄT

### [ API-Verwaltung ]

Implementieren Sie eine API-Verwaltungslösung zum Verwalten von **Kontingenten**, **Drosselung**, **Authentifizierung** und **Protokollierung**.

Sammeln Sie Messwerte zur Verwaltung von **Überwachung**, **Filterung** und **Berichterstellung**.







Legen Sie **Anforderungen** für **externe Systeme und Dienste** fest, die in die Plattform integriert sind.



## INTEROPERABILITÄT

### [ ANFORDERUNGEN ]

Erfordern **externe Systeme** erfüllen die gleichen **Anforderungen** wie **interne Systeme**.

Externe Systeme müssen **Ereignisse** veröffentlichen und **technische** Überwachung zulassen.

Für den Fall, dass die externen Systemdaten integriert werden müssen, muss die **Gesamt** Synchronisation **möglich sein**.





INTEROPERABILITÄT



[ MULTI-MIETER ]

Die Architektur muss als  
**Multi-Tenant** betrachtet  
werden.





## INTEROPERABILITÄT

### [ MULTI-MIETER ]

Auch wenn die weiße Markierung an der Basis nicht berücksichtigt wird, richten Sie eine Multi-Tenant-Architektur ein. Ihre **ursprüngliche** Anwendung ist die erste **Halterung**.

Denken Sie von Anfang an an die **multifunktionale Instanziierung** des Systems.







INTEROPERABILITÄT



[ EINSTELLUNG ]

Die Systeme müssen **nativ konfigurierbar** sein.





## INTEROPERABILITÄT

[ EINSTELLUNG ]

**Sprachen, Währungen, Geschäftsregeln, Sicherheitsprofile** müssen einfach einzustellen sein.

Vorsicht vor **Hyper-Generizität**, es ist oft nutzlos und **Kostenquelle**.

Das **Setup** muss **skalierbar** und schnell wie erforderlich sein.





INTEROPERABILITÄT



[ FEATURE FLIPPING ]



Erstellen Sie flexible und generische Systeme mit  
**Feature-Flipping.**





## INTEROPERABILITÄT

### [ FEATURE FLIPPING ]

**Feature Flipping** bedeutet, eine App als eine Reihe von **Funktionen** zu gestalten, die **aktiviert** oder **deaktiviert** heiß, **Produktion** sein können.

In einer **Multi-Tenant** -Anwendung ermöglicht das Feature-Flipping \*\* das Anpassen von Unterstützern.

Das Feature flip **vereinfacht die A / B-Prüfung**.







# Spielregeln





# Spielregeln









Die **technischen Entscheidungen** werden vom **Feature-Team** getroffen und **\*\*** angenommen.



## Spielregeln

### [ TECHNISCHE WAHLEN ]

Das Feature-Team muss **verantwortungsvoll handeln**, um die Auswahlmöglichkeiten zu identifizieren, die sich ausschließlich auf das Feature-Team auswirken, sowie die Auswahlmöglichkeiten, die sich auf das Unternehmen auswirken.

Die **Optionen**, die **den Umfang** des Feature-Teams überschreiten (z. B. Lizenz, seltene Programmiersprache), müssen **von**







Spielregeln

[ Guter Gebrauch ]

Das **richtige Werkzeug** für  
**gute Nutzung** ist eine  
Ersparnisquelle.





## Spielregeln

[ Guter Gebrauch ]

Ein **schlechtes Werkzeug**, das allen auferlegt wird, ist ein **Risiko**. Der **Missbrauch** eines guten Tools kann **sehr** schädliche Folgen haben \*\*. Zum Beispiel sind schlecht verwendete Agile-Methoden gefährlich.

**Werkzeuge** müssen **befragt werden**.

**Excel** ist oft eine vernünftige Wahl, **aber es ist nicht** ein Werkzeug, um alles zu tun \*\* (CRM, ERP, Datamart, ...)







Privileg **Build** für das  
Kerngeschäft.

Betrachten Sie **Buy** für den  
Rest, von Fall zu Fall.





## Spielregeln

### [ BUILD VS. KAUFEN ]

Je mehr ein Tool eine **Funktion zur Differenzierung** für die Organisation aufweist, desto mehr soll es **gebaut** werden. Das Kerngeschäft muss **Spezifität** ermöglichen und **sich schnell und oft anpassen**. Einige **Softwarepakete** werden **manchmal an diesen Bedarf angepasst**.

Für **den Rest** : SaaS, Open Source, Build oder Owner sind von Fall zu Fall zu untersuchen \*\*.







**Machen Sie das Beste aus  
Open Source.**

Alternative  
Auswahlmöglichkeiten  
müssen unterstützt  
werden.





## Spielregeln

[ OPEN SOURCE ]

Die **proprietären Lösungen** sind ein **Risiko** für die Organisation, die in der Lage sein muss, die Wartung bei Bedarf fortzusetzen.

Es gibt wenige proprietäre Tools, die keine **Open-Source-Alternativen** haben.

Die Organisation **profitiert** von der **Open Source Community** und kann **ihre Beiträge zurückzahlen**.







Develop **Stand-alone** und  
**schwach gekoppelte**  
Dienste.



Spielregeln

[ MICRO-DIENSTE ]

**schwache Kopplung** muss die Norm sein.

Jeder Micro-Service hat eine **klar definierte Schnittstelle**.

Diese **Schnittstelle** bestimmt den **Link** zwischen den **Micro-Services**.

**Domain Driven Design** erlaubt, insbesondere mit den **Bounded Contexts**, dieses Problem vorherzusehen.









Jeder Dienst hat sein  
eigenes \*\*  
Datenspeichersystem.



Spielregeln

[ DATA ]

Ein **Data Store** soll **nur** mit **einem einzigen Micro-Service** \*\* gekoppelt sein.

Der **Zugriff auf Daten** von einem Micro-Service zu einem anderen **erfolgt ausschließlich über seine Schnittstelle**.

Dieses Design bedeutet **Konsistenz über die Zeit hinweg** auf der gesamten Plattform. Es muss **auf allen Ebenen** einschließlich UX aufgegriffen werden.





Jeder Micro-Service muss einen angemessenen Funktionsumfang haben, der **"in den Kopf passt"**.



Spielregeln

[ SCOPE ]

Ein Micro-Service bietet eine **angemessene Anzahl von Funktionen**.

**Zögere nicht**, einen Micro-Service zu schneiden, wenn es anfängt zu wachsen.

Ein Dienst von angemessener Größe macht es möglich, **das** Umschreiben \*\* gelassen zu betrachten, wenn die Notwendigkeit besteht.







Spielregeln

[ RESPONSIVE ]

Das **Reaktive Manifest**  
öffnet den Weg zur  
Gestaltung reaktiver  
Architekturen.





Spielregeln

[ RESPONSIVE ]

**Responsive** Programmierung konzentriert sich auf den Datenfluss und die Verbreitung von Änderungen. Es basiert auf dem Muster "**Beobachter**" im Gegensatz zu dem Ansatz "**Iterator**", traditioneller.

Das Reaktive Manifest legt die grundlegenden Achsen fest:

**Verfügbarkeit** und Geschwindigkeit, **Ausfallsicherheit** für Zusammenbrüche, **Flexibilität**, **Elastizität** und **Nachrichtenerkennung**.









Spielregeln

[ ASYNC-ERSTE ]

**asynchrone** Prozesse  
bevorzugen **Entkopplung**  
und **Skalierbarkeit**  
zugunsten **Leistung**.





## Spielregeln

### [ ASYNC-ERSTE ]

Der Austausch zwischen Anwendungen muss zuerst \*\* asynchron sein.

Asynchrone Austausche **erlauben**

\*\*\*\* \* - \* - - - **und** \*\*\*\* - \* - \* - \*\*\*\* - \* - \* -  
- - \*\*\*\* \*\*

**synchrone Kommunikation** sollte nur berücksichtigt werden, **wenn die Aktion es erfordert.**







Das Informationssystem  
muss **Ereignisse** orientiert  
sein.



Spielregeln

[ EVENTS ]

Die \*\*\*\* ereignisgesteuerten  
\**funktionalen*\* Prozesse **sind**  
natürlich \*\*\*\* asynchron \*\*  
implementiert.

Die **Event Orientation** ermöglicht  
die Implementierung von Ansätzen  
wie **C** ommand **Q** sehr **R**  
Verantwortlichkeit **S** egregation  
(**CQRS**) und **Event Sourcing**.







Spielregeln

[ Nachrichtenbroker ]



Privilegieren Sie einen  
**einfachen, robusten und  
leistungsfähigen**  
Message-Broker zu einer  
"Smart Pipe".







## Spielregeln

[ Nachrichtenbroker ]

**ESB** zeigte **Grenzwerte** :  
**Skalierbare Wartung** ist **kritisch** ,  
sowohl von **technischer** als auch  
**organisatorischer** Sichtweise.

\*\*\*\* Broker *\*-Nachrichten wie\** Kafka  
**bieten eine einfache , dauerhafte**  
und **belastbare** Lösung.

**Intelligente Endpunkte** und  
**Einfache Pipes** ist eine Architektur,  
die im Maßstab funktioniert: es ist  
**Internet.**







Die **vollständige Synchronisation** des Systems sollte berücksichtigt werden, sobald es **entworfen wurde**.



Spielregeln

[ STEUER ]

Wenn die **Synchronisation** zwischen zwei Systemen durch einen **Ereignisfluss** sichergestellt wird, muss die **Gesamtsynchronisation** dieser Systeme **zur Entwurfszeit** geplant werden.

Ein automatisches \*\*\*\*

*\*Synchronisations-Audit\** (Beispiel: per Stichprobe) erlaubt es, **mögliche** Synchronisationsfehler **zu messen** und **zu erkennen**.







Die **Konfiguration** der Dienste ist **zentralisiert**, ihre **Entdeckung** wird durch ein **Verzeichnis** gewährleistet.



## Spielregeln

### [ ZENTRALISIERUNG ]

Die **Konfiguration** der **Micro-Services** ist **zentralisiert** für alle **Umgebungen**.

Ein zentrales **Verzeichnis** gewährleistet **dynamische Erkennung** von **Micro-Services**.

Die \*\*\* globale *\*Skalierbarkeit\** hängt von diesem **Verzeichnis** ab.









Spielregeln

[ SAND-FACH ]

Feature-Teams bieten eine  
**Sandbox-Umgebung.**





## Spielregeln

### [ SAND-FACH ]

Feature Teams unterhalten eine **Sandbox** -Umgebung (aktuelle Version und bevorstehende Version), um anderen **Teams das Scale-up** zu ermöglichen.

In **einigen nicht-nominalen** Fällen **können** Funktionen **in der** Entwicklungsumgebung \*\*\*\* deaktiviert sein.







Spielregeln

[ Design für Fehler ]



**Ihr System wird  
abstürzen!**

Entwerfen Sie es so, dass  
es tolerant ist.





## Spielregeln

### [ Design für Fehler ]

Ihr **System wird abstürzen**, es ist unvermeidlich. Es muss dafür ausgelegt sein (**Design For Failure**).

Predict **Redundanz** auf allen Ebenen: **Hardware** (Netzwerk, Festplatte, etc.), **Anwendungen** (mehrere Instanzen von Anwendungen), **geographische Zonen**, **Anbieter** (Beispiel: AWS + OVH)







Spielregeln

[ Werkzeugkits ]

Stellen Sie **Toolkits** zur Verfügung, stellen Sie keine strengen Frameworks auf.





Spielregeln

[ Werkzeugkits ]

## **Achtung auf die technischen Komponenten Häuser und Quer !**

Sie sind restriktiv, teuer und schwer zu warten.

**Beschleuniger , Toolkits ,  
technische Stacks** können  
**zusammengefasst , frei** Feature-  
Teams sein, die einen  
dogmatischen Ansatz vermeiden.









Öffentlich, privat oder hybrid, die **Cloud (IaaS oder PaaS)** ist der Standard für die Produktion.



Spielregeln

[ WOLKE ]

**PaaS** -Dienste sind **bevorzugt**,  
**einfach** und skalieren schnell.

**IaaS** -Dienstleistungen  
ermöglichen es Ihnen, Fälle  
anzugehen, die eine größere  
**Flexibilität** erfordern, aber mehr  
operative Arbeit erfordern.

Eine Private Cloud ist keine  
herkömmliche  
Virtualisierungsumgebung, sie  
beruht auf **Standardhardware**.







Feature Teams verwalten  
die Infrastruktur nicht, sie  
wird **von der Organisation**  
bereitgestellt und  
verwaltet.



## Spielregeln

### [ INFRASTRUKTUR ]

Infrastrukturprobleme gehören nicht zu **Feature Teams**. Die Infrastruktur muss durch einen **funktionsübergreifenden** Dienst **bereitgestellt** und \*\* gewartet werden.







**Container** bieten die Flexibilität, die für heterogene Werkzeuge benötigt wird.





## Spielregeln

### [ CONTAINER ]

Container bieten die **Flexibilität** ,  
die Feature Teams benötigen, um  
**heterogene Werkzeuge** in einem  
**homogenen Kontext** zu aktivieren.







Die Verwendung von **Containern** ermöglicht es, die Probleme der **technischen Umgebung** zu überwinden.



## Spielregeln

### [ ENVIRONMENTS ]

Die Container (Beispiel: **Docker**) ermöglichen \*\* die Befreiung von den Umgebungsdifferenzen.

Der **Bereitstellungsprozess** muss für die Umgebung **agnostisch** sein.

**Einige Komponenten** wie Datenbanken sollten nicht in Containern bereitgestellt werden. Ihr Einsatz ist noch automatisiert.







Maßnahmen müssen  
zentralisiert und für alle  
zugänglich sein.



Spielregeln

[ METRIC ]

Die **Messwerte** sind **für alle Benutzer mit unterschiedlichem Detaillierungsgrad** zugänglich:

Detailansicht für das jeweilige Team-Feature, Aggregationen für andere Mitglieder des Unternehmens.

Der Zugriff auf **Metrik bedeutet nicht den Zugriff auf die Daten der Einheit**, sie muss kontrolliert werden, um die Vertraulichkeit zu wahren.

**Alle Umgebungen** sind betroffen.









Spielregeln

[ QUALITÄT ]

**Softwarequalität ist ein  
Schlüsselfaktor.**





Spielregeln

[ QUALITÄT ]

**Code Reviews** sind **systematisch**.

Sie werden von Mitgliedern des Feature Teams oder anderen Mitgliedern der Organisation im Rahmen von **Continuous Improvement** durchgeführt.

Das **ist nicht auditiert, aber dein Code** : "Du bist nicht dein Code!".

Die **Qualimetrie** kann teilweise automatisiert werden, aber nichts schlägt das **neue Auge**.







**automatisiertes Testen** ist  
eine nicht verhandelbare  
Voraussetzung für die  
kontinuierliche  
Bereitstellung.



## [ AUTOMATISIERTE TESTS ]

### Automatisierte **Tests**

gewährleisten die **Qualität** des Produkts **im Laufe der Zeit**.

Es ist eine **Voraussetzung** für die kontinuierliche Bereitstellung, es \*\*\*\* Änderungen *\*und\** häufige Bereitstellungen \*\* ermöglicht.

Der **Production Rollout** wird zu einem **anekdotischen** Event!







**Tests auf allen Ebenen :**  
Einheit, Integration,  
Funktionalität,  
Belastbarkeit, Leistung.



## Spielregeln

### [ TESTSTUFEN ]

Die Tests **Integration** und **Funktional** sind die wichtigsten, **sie** garantieren **den** effektiven **Betrieb**.

**Einheit** -Tests sind für **Entwicklung** geeignet.

**Leistung** Tests messen die Leistung **im Zeitverlauf**.

**Resilience** -Tests helfen **Fehler** vorwegzunehmen.









**Cover** ist der primäre  
Zielindikator für die  
Testqualität.



Spielregeln

[ COVER ]

Die **Codeabdeckung** der Tests ist eine **gute** Metrik der Codequalität.

Dies ist eine **notwendige Bedingung**, aber **nicht ausreichend**, die Abdeckung einer **schlechten** Teststrategie kann hoch sein, ohne die gute Qualität des Codes zu garantieren.







**Sicherheit** ist ein **Prozess** ,  
sollte nicht als Reaktion  
auf Probleme behandelt  
werden.



## Spielregeln

### [ SICHERHEIT ]

**Sicherheitsexperten** können bei Bedarf **direkt in Feature-Teams** integriert \*\* werden.

**Sicherheitsexperten** sind in der Organisation für **Audit** , **Awareness** und **Forward** verfügbar.



