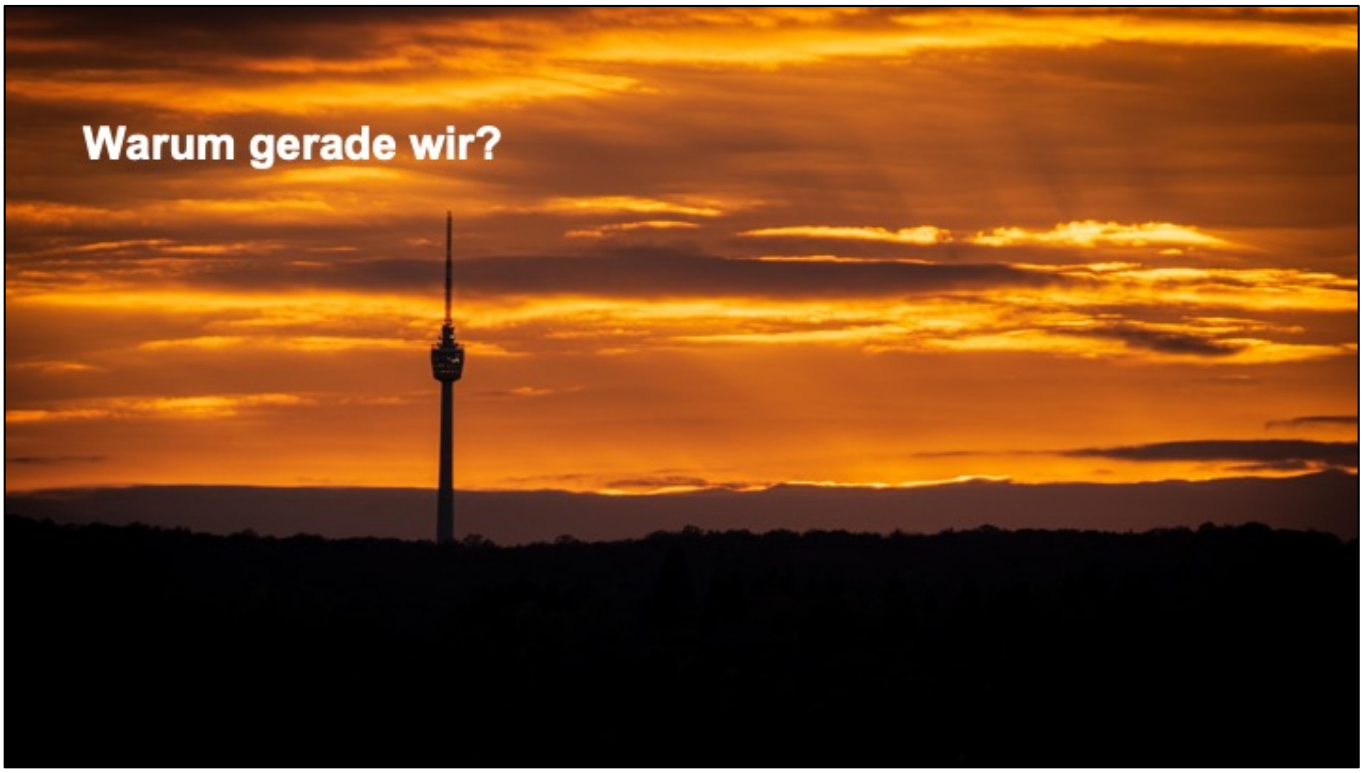
Technik, Menschen, Unternehmen:
Wie Modernisierungsprojekte
(vielleicht) besser laufen könnten.
Kolja Dummann | Everest Systems
Markus Völter | Independent

Kontext:
Modernisierungsprojekte
Technologie-Update
Konkurrenz holt auf
Innovation


Warum gerade wir?

Inhalte
Architektur & Hypes
Prozessanpassungen
Interpretation von "Agil"
immer größere Flughöhe
Kommunikation und Meetingkultur
Führung, Entscheidungsfindung, Produktvision
Firmenkultur und deren Auswirkungen

- **Wir sprechen Fehler und Probleme aus unseren Projekten an, damit ihr diese erkennen und benennen könnt.**
- **Konkrete Ratschläge geben ist schwierig, da die Lösung sehr kontextabhängig ist.**
- **Probleme sind teilweise bewusst vage beschrieben, um niemanden an den Pranger zu stellen.**

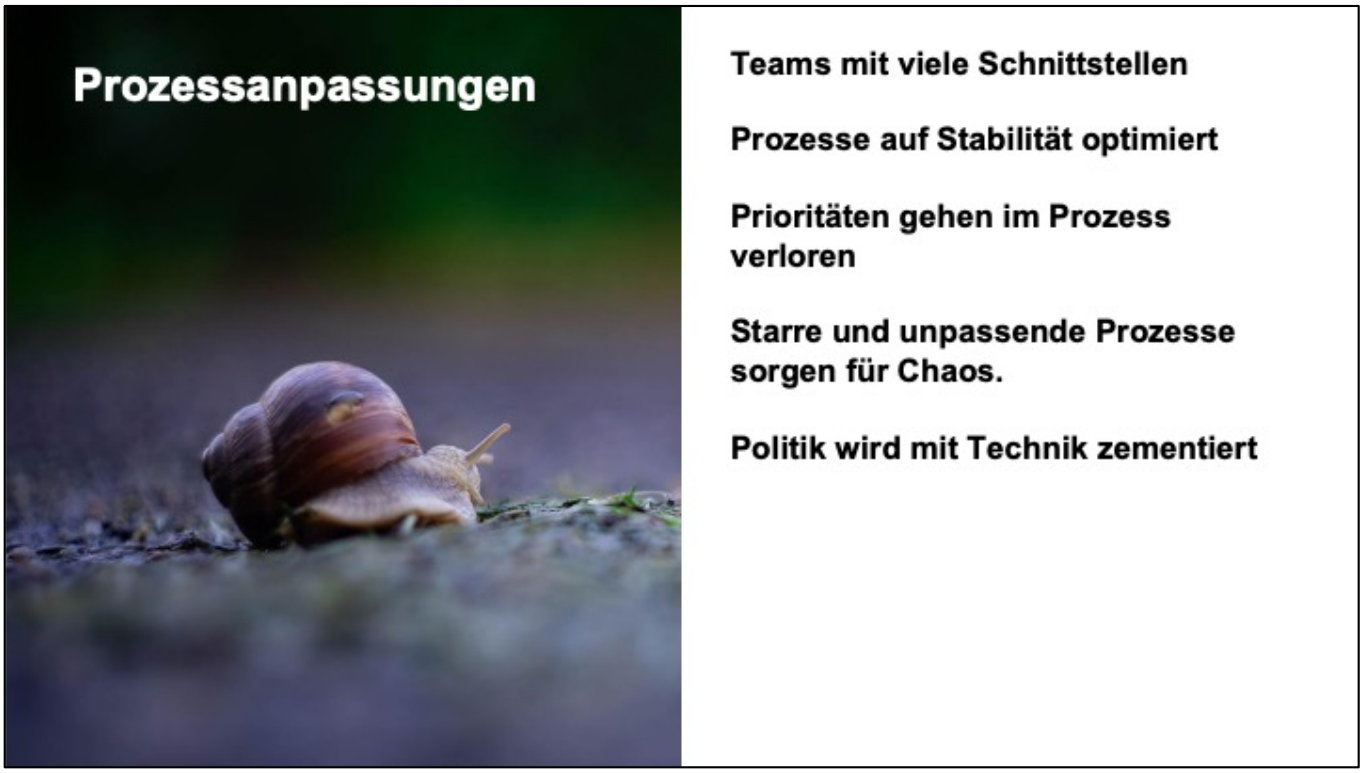
Architektur und Hypes

- **Kontextfreie Buzzword-Gläubigkeit:**
  - Aus Büchern oder Silicon Valley Case Studies
  - Aber: ganz anderer Kontext oder Situation
    - Bspw. sehr viel mehr Fachlichkeit
    - "... then they should also build search engines"
  - Beispiele
    - Microservices
    - mob programming
    - cross-functional teams
    - "Platform contradicts agile"
- **Erfahrung zählt nicht.**
  - Alle Architekturentscheidungen müssen begründet und formal abgewogen werden
    - Erfahrung fließt in die Entscheidung nicht ein
    - Aber wir können halt keine 5 Prototypen bauen um vergleichen zu können
  - Siehe Führung: Konsens ist nötig, sogar mit Leuten die keine Erfahrung haben oder die Sache nicht verstehen

    - False Balance?
        - unerfahrene Kollegen schlagen etwas vor
        - die kleinere Gruppe an erfahrenen Leuten argumentiert dagegen
        - aber wird ignoriert, weil Minderheit
- **"Messbar" als Kampfbegriff**
    - Fordern von Messbaren Vorteilen
    - Unklar, was gemessen werden soll und mit welchen (historischen!) Daten man vergleichen möchte
    - Experimente sind schwierig; das Experiment ist das Projekt an sich!
    - Damit abwürgen von Ansätzen: man kann ja nicht "beweisen" dass es besser ist.
- **Architekturen halbherzig umsetzen**
    - Wir bauen eine Plattform, aber sagen das nicht so, weil "uncool" oder "unagil"
    - Als Folge werden die nötigen organistorischen und Prozessanpassungen nicht umgesetzt
        - Man hat Angst for dem Bottleneck "Platform"
        - setzt sie dann halbherzig um, kein Team, kein PO, keine Resourcen
        - Self-fulfilling Bottleneck.
- **We build cars not tools**
    - Investition in Werkzeuge, Frameworks, Automatisierung wird als "sinnloser Overhead" gesehen
    - Außer bei Buildpipelines -- die sind gut!
    - Ironisch, weil Autohersteller extrem viele "Tools" bauen: Logistik, Produktionsstraßen, ...
- **Architekturpolizei**
    - Es gibt eine Architekturabteilung mit durchaus guten (auch explizit ausgebildeten) Leuten
    - Aber: agieren als High-Level Consultant (Workshops, Präsentationen)
    - ... und Polizei ("this ARC42 document isn't good enough").
    - Kein wirkliches Reindenken in Projektkontext, kein Committttment. Damit nur bedingt hilfreich.

- **Jeder-kann-alles Fallacy**
  - Neue Technologien/Architekturansätze ohne Leute im Projekt die damit Erfahrung haben
  - Klar, jeder kann lernen, aber das dauert -- Konflikt mit Deadlines.
  - Im Effekt: "Buchwissen" beim Systembau
  - Echte Erfahrung ist durch nichts zu ersetzen

Prozessanpassungen



- **Teams mit vielen Schnittstellen**
  - Teams sind für spezialisierte Aufgaben optimiert und können Aufgaben nicht ganzheitlich abbilden weil:
    - Fehlende Erfahrung im Team
    - Formal nicht “befugt” Aufgaben außerhalb der Kernaufgaben wahrzunehmen.
  - Viele Abhängigkeiten machen eigene Planung abhängig von der Planung und Prioritäten anderer Teams.
    - Team A setzt sich Ziele in der Annahme, Team B kann die nötige Zuarbeit leisten.
    - Team B hat andere Prioritäten und erfüllt die Zuarbeit nicht.
    - Team A ist blockiert.
    - Nicht nur offensichtliche Abhängigkeiten über Artefakte relevant, sondern auch z.B. Team B soll ein Artefakt von Team A bewerten, damit dieses weiß ob es die Richtung weiter verfolgen kann.
- **Prozesse auf Stabilität optimiert**

    - Es ist hart die benötigten Tools zu bekommen, z.B. Beschaffung einer Software für 100€ dauert Monate.
    - Prozesse sehen nicht vor, dass “geforscht” wird. Fokus auf stabile Umgebung.
    - Optimierung/Auslegung auf den Worst Case.
        - bspw. Mutwilliger Missbrauch der Freiräume.
- **Prioritäten gehen im Prozess verloren**
    - Ziel der Einkaufsabteilung möglichst günstig etwas einkaufen, Ziel des “Innovations Teams” Consulting/Unterstützung durch einen Partner
    - 100€ Software Lizenz für IT Abteilung irrelevant, für “Innovations Team” wichtig, um weitere Entscheidungen treffen zu können.
- **Starre und unpassende Prozesse sorgen für Chaos**
    - Bürokratische Prozesse sorgen für Resignation/Frustration.
    - Mangelnde Flexibilität in den Prozessen sorgt dafür, dass sie gänzlich ignoriert werden.
        - Führt zu Chaos
        - Resultat ist das Gegenteil von dem, was beabsichtigt ist. Statt Strukturen, die Teams helfen, Entscheidungen zu treffen und Schnittstellen klar zu machen, entsteht Chaos.
    - Der Mitarbeiter muss selbst entscheiden, ob er den Prozess verlässt, um die Projektziele zu erreichen. Moralisches Dilemma.
    - Begriffe im Prozess sind wichtiger als die eigentliche Arbeit.
        - Bugs dürfen nicht als Bug bezeichnet werden, weil sie die Statistik belasten.
        - Für Features fordert der Prozess aber mehr Validierung.
- **Politik wird mit Technik zementiert**
    - Nicht alle vordergründig technischen Probleme sind rein technischer Natur.
    - (Technische) Schnittstellen oft Resultat von Organisationsstrukturen.
    - Änderungen der technischen Architektur zieht eine organisatorische Veränderung mit sich.
    - Teams und Führungskräfte sind dafür oft nicht sensibilisiert genug.
    - Monate damit verbringen, eine technische Lösung für ein Organisationsstruktur-Problem zu finden.
    - Gefühlter Machtverlust durch Veränderung spielt eine große Rolle.
        - Veränderungen müssen begleitet und kommuniziert

werden.
- Conway's Law.

Interpretation von "agil"

- **Vordefinierter “agiler” Prozess:**
  - Etabliertes Agiles vorgehen wird nicht hinterfragt sondern eins zu eins auf das Projekt angewendet
    - Rahmenbedingungen für “etwas Neues machen” und bestehendes weiterentwickeln sind aber grundverschieden.
    - Standardisierung über Projektgrenzen hinweg wird extrem überbewertet
    - “Wir machen Scrum!“
  - Der Teamgröße und Projektreife unangepasster Prozess
    - Viele Meetings zur Synchronisation, während das Team klein ist und sie nicht braucht.
    - Planen wichtiger als funktionierende Software.
  - Falsches Vorgehen während frühen explorativen Phasen des Projekts
    - Etabliertes Vorgehen erwartet Planbarkeit, während im Projekt noch exploriert wird.
    - Explorative Aufgaben werden versucht Planbar zu

machen
                - Erzeugt overhead
                - Plan und Wirklichkeit weichen meist gravierend voneinander ab, da neue Erkenntnisse aus der Exploration den Plan ändern.
    - Das Gegenteil von Agile
- **Fehlende Führung**
    - Wege zur Entscheidungsfindung sind unklar.
    - Konsens zwischen allen Stakeholdern wichtiger als Entscheidungen zu treffen
        - Statt klarer Ziele: “Design by Committee"
        - Personen, die formal Entscheidungen treffen und vertreten können, sehen sich als Beobachter oder Verwalter.
        - Konkurrierende Vorschläge werden nie aufgelöst, sondern immer wieder diskutiert.
        - Grundsatzentscheidungen, die richtungsweisend für das weitere Vorgehen sind, werden extrem hinausgezögert.
    - Entscheidungen sollen immer “aus dem Team kommen”
    - Die Organisation bestraft falsche Entscheidungen härter als Verzögerung oder Komplexität durch keine Entscheidung.
    - Product Owner sieht sich als Moderator.
        - Vertritt keine konsistente Produktvision, sondern ist Sprachrohr der Stakeholder.
- **Agile heißt: wir schreiben Code**
    - Nur Code ist ein valides Artefakt.
    - Konzeptionelle / Architekturarbeit ist “overhead” und muss vermieden werden.
    - Keine Wertschätzung für Aufgaben abseits vom Code
        - Zum Beispiel Product Owner: “der macht ja eh nur meetings” darum wenig Wertschätzung für Konzeptionelle Arbeit.
- **Triviale Definition von Customer Value!**
    - Tools, Frameworks oder Refactorings haben keinen Wert weil:
        - “Endkunde der das Auto kauft hat nix davon”
        - Sind nicht offensichtlich im Endprodukt, ignoriert Aspekte wie z.B.:

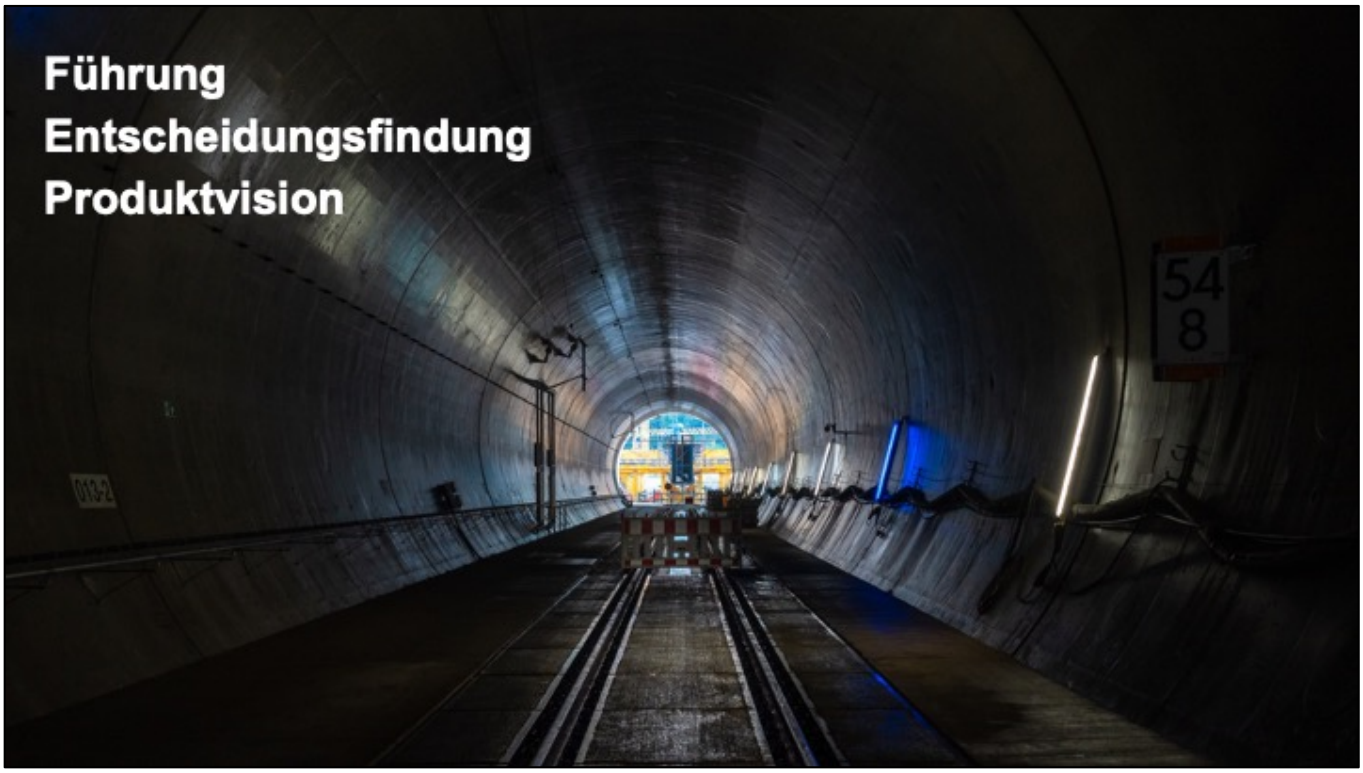
            - Besser Qualität
            - Schnellere Time to market
    - Unklare Definition des MVP
        - MVP wird zur Absicherung technischer Risiken benutzt und nicht zum Absichern der Produktvision.
        - Unklare Ziele
            - Management möchte MVP zur Absicherung der Produktvision.
            - Entwicklerteam baut einen PoC zu Absicherung technischer Risiken.
        - MVP schafft keinen Mehrwert zum aktuellen Vorgehen.
            - Kunde nutzt den MVP nicht, da er (noch) nicht genug Mehrwert schafft.
- **Fehlender Fokus**
    - Prioritäten wechseln ständig, dadurch wird aber nichts wirklich fertig.
    - Oft wird nur von Sprint zu Sprint gedacht. Längerfristige Konsequenzen vom Fokus wechseln, wird ignoriert.
- **Team passt nicht zur Aufgabe**
    - Einzelne Personen im Team sind überlastet, weil die nötigen Skills im Team nicht gleichmäßig verteilt sind.
    - Das Team ist auf dem Papier groß genug, aber das Verteilen der Aufgaben schwer möglich.
        - Aufgaben zu Groß.
        - Benötigte Skills ungleich verteilt.
    - Delegation gewünscht, aber kein Raum für Arbeitsübergabe, weil der Overhead nicht akzeptiert wird.
        - “Alle im Team können das, sind ja Entwickler”

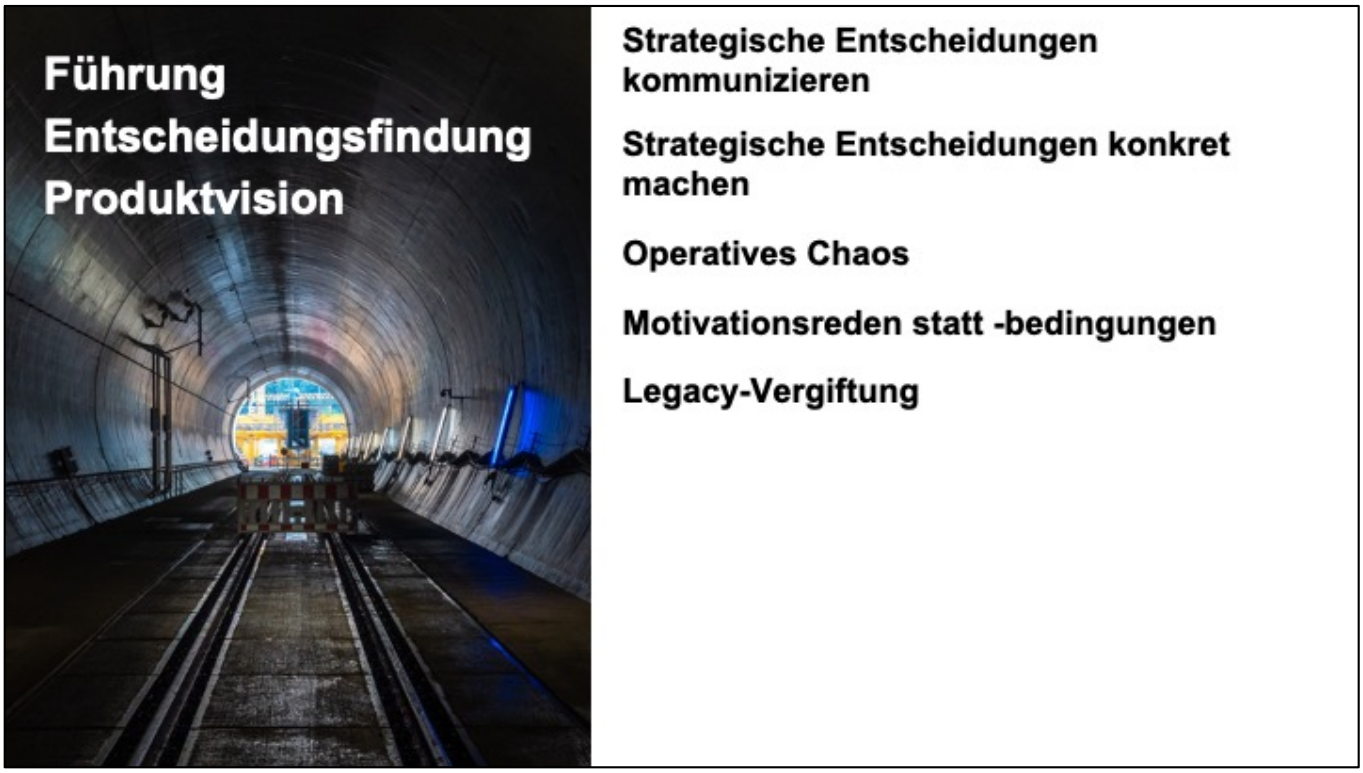
Kommunikation
& Meetingkultur

- **Alles ist Meeting-Driven**
  - Unterbrechen Arbeit (vor allem bei Entwicklern)
  - Man hat keine Nachdenk- und Konzeptionszeit (bei allen)
  - Alle Kalender sind immer voll, deshalb wird ein (jetzt!) nötiges Meeting für Wochen später geplant
  - Gesamtfortschritt ist langsam weil es (brutto) Tage und Wochen dauert bis etwas diskutiert/geklärt ist
  - **Workaround?**
    - Man macht immer sieben Dinge parallel, in der Hoffnung dass immer eines "unblocked" ist
    - Aber das ist ineffizient wegen ständigen Kontextwechseln (im Kopf und in Tools)
    - und "too much work in progress" ist generell keine gute Idee wegen Verzettelung
- **Sogar Arbeits-Koordination ist in Meetings!**
  - Sogar in den technischen Teams!
  - "Hey, wie wollen wir dieses JAR-File nennen"? Dazu braucht's ein Meeting mit vier Leuten, Tage später.

    - Kein spontanes, "Hey, lass uns mal kurz reden/chatten wie wir das machen"
    - Auch Architekten und Technologieexperten (keine Externen, in der Firma!) müssen "gebucht" werden.
- **Abstand zu lang, unterschiedliche Leute**
    - Die meisten Meetings sind <= 1.5 Stunden, da kriegt man viele Dinge nicht zu Ende gedacht/entschieden
    - Also Folgemeeting. Nächste Woche weil Kalender voll sind.
    - Bis dahin haben wir die Hälfte wieder vergessen und brauchen die Hälfte des nächsten Meetings um wieder den "Gedankenstatus" vom Ende des letzten Meetings herzustellen. Also wird alles zweimal besprochen.
    - Und manchmal sind dann auch noch 1/3 andere Leute dabei. Dann ist der Gedankenstatus-Aufwand noch größer
- **Low Quality Meetings**
    - Wenig bis keine Vorbereitung. Keine Agenda.
    - Viel Zeit den Kontext zu erklären weil zu viele Leute dabei sind, und davon die meisten unvorbereitet
    - Oft kein klares Ziel ("Wir sprechen über X" statt "Wir entscheiden X")
    - Entscheidungen und Ergebnisse nicht (sinnvoll) dokumentiert.
        - Entscheidungen nicht nachvollziehbar, damit auch angreifbar
        - Alles wird X Mal neu diskutiert, weil "die Organisation" vergessen hat, dass etwas etwas schon entschieden wurde
        - Haben sich die Umstände so gravierend geändert, dass wir die Entscheidung überdenken müssen?
    - Schlechte Moderation
        - Off-Topic Diskussionen zu anderen (Rand-)themen
        - Oder zu frühes abbrechen fruchtbarer (aber vielleicht "anstrengender") Diskussionen
- **Keine gemeinsame Terminologie**
    - Es gibt kein Glossar oder es hält sich keiner dran
    - Overlap in Begriffen, sogar in "offiziellen" Begriffen
    - Macht effiziente Kommunikation schwierig:
        - Dauernde Wieder-Erklärung

        - Einwände auf Basis von falschem Verständnis dessen was gesagt wurde.
- **Typische "Druck" Kommunikation**
    - "Es ist 16 Uhr an einem Samstag, und ich sehe in Jira nicht viel Fortschritt bei den Aufgaben, die wir diese Woche abschließen mussten. Wenn Sie eine dieser Aufgaben zugewiesen bekommen haben, die bis diesen Freitag abgeschlossen sein sollten, stellen Sie bitte sicher, dass Sie die Ihnen zugewiesene Arbeit bis Sonntagabend abgeschlossen haben. Wenn Sie diesen Zeitrahmen nicht einhalten können, legen Sie bitte einen Plan für die Fertigstellung vor, um uns wieder auf den richtigen Weg zu bringen."
    - Ton unakzeptabel
    - Die Deadline selber war keine echte - kein Kunde. Es ging nur um "Gesichtswahrung" beim Management
    - Führt nur zu Frust und Widerstand.
- **Kein Projektmarketing**
    - Marketing = Do good work and then talk about it
    - Slide decks are where information goes to die!
        - statt dessen (oder ergänzend) Videos, Podcasts.
        - Zahlen, Daten, Fakten, aber auch Anecdotes, Fürsprecher aus der Operativen Ebene
    - Durchaus auch ein bisschen unterhalten. Die Zielgruppe soll es lesen/schauen/hören *wollen*.
    - Aus Sicht des Senders immer Überkommunizieren - aus Sicht der Empfänger ist viel mehr Kommunukation nötig als aus Sicht des Senders
        - die "richtige" Story/Fakten/Wahrheit muss "gewinnen" gegenüber Gerüchten und Kaffeeküchengeschwätz
        - Gleiche Situation wie mit Fake News im echten Leben.
    - Nebeneffekt: Transparenz erzeugt Vertrauen!

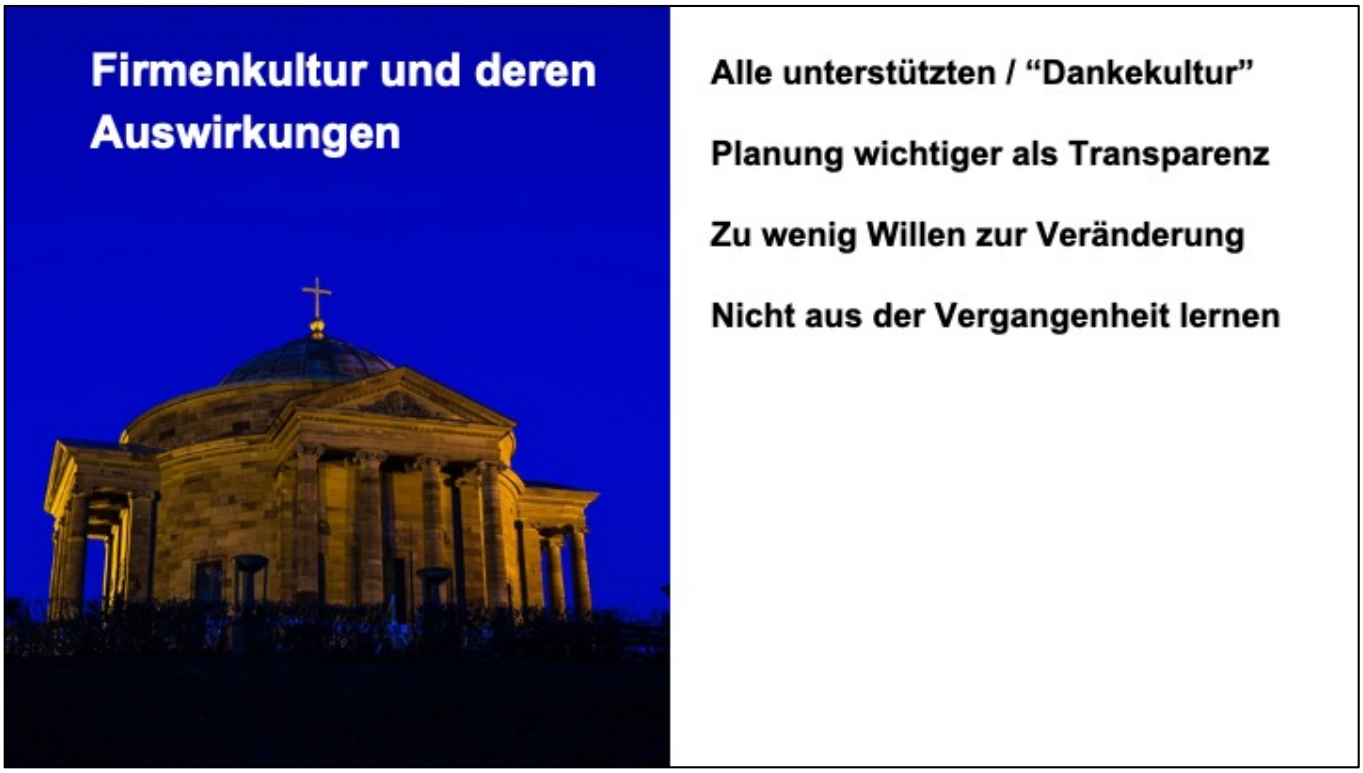

Führung
Entscheidungsfindung
Produktvision



- **Strategische Entscheidungen kommunizieren**
  - Zweck des Produktes (oder Projektziels), Scope, Features und essentielle Architekturentscheidungen werden schlecht oder nicht kommuniziert
    - ans Team
    - und an externe Stakeholder
  - Folge?
    - diese high-level Dinge werden ständig aufs neue hinterfragt
    - technische Teams können Detailentscheidungen nicht am Big Picture ausrichten bzw. damit begründen
    - dadurch werden auch die technischen Entscheidungen immer wieder hinterfragt und aufs neue diskutiert
- **Strategische Entscheidungen konkret machen**
  - Beispielen und Auswirkungen für Menschen und Systeme
  - Als Checkliste bei konkreten entscheidungen nutzen
  - Sonst bleibt es bei "Business Buzzwords" die auf der operativen Ebene keiner ernst nimmt
  - Damit Entkopplung von Management und operativer Ebene

- **Operatives Chaos**
  - In den zwei Wochen vor der Deadline haben wir alle zwei Tage zwischen zwei "Sprintzielen" hin- und hergewechselt.
    - Folge? Keines wurde fertig.
    - Ständige Richtungsänderungen erfordern feingranulare Steuerung -> Bottlenecks
    - Leute können nicht unabhängig einfach "am Ziel" arbeiten
    - Gesamtfortschritt ist langsam
  - PO und sein "Hauptkunde" im Unternehmen reden nicht miteinander
    - Technische Teams bekommen unabhängig (und unterschiedlichen) Input von beiden
  - Managementteam einigt sich nicht auf Deadlines und Release-Termine ("Was? Von der Deadline weiß ich nix")
- **Motivationsreden statt -bedingungen**
  - "Alles-muss-anders-werden"-Gerede, ohne Bezug zu dem was die technischen Teams wirklich tun
    - Unglaubwürdig
    - "when you release the scrum master every three weeks".
  - Teambuilding by "game and beer" statt
    - gute Leute im Team die sich gegenseitig für ihre Skills respektieren
    - gute Arbeitsumgebung, gute Tools, und praktikable, pragmatische Prozesse
- **Legacy-Vergiftung**
  - Legacy-Kompatibilität wird ständig diskutiert und ist unverrückbare Anforderung
    - Verhindert saubere neue Lösungen
    - erhöht Komplexität
    - und Aufwand teilweise signifikant!
  - Management sieht das als Risikoreduzierung, in Wirklichkeit erhöht es aber das Risiko oft.
  - Ganz blöd: als sauberes neues System anfangen, aber kurz vor Ende beschließen, dass man doch kompatibel sein muss mit dem alten System -> es werden jede Menge Krücken in das schöne neue System reingepfuscht die nur so halb funktionieren.

Firmenkultur und deren
Auswirkungen



- **Alle unterstützten / “Dankekultur”**
  - Jeder unterstützt aber keine macht oder entscheidet etwas
  - Thank You meetings wo dem gesamten Team unspezifisch gedankt wird
    - Individuelle Beiträge werden nicht gewürdigt
    - Fehler werden ignoriert und nicht angesprochen, alles ist immer gut.
  - Anerkennung wird nur zum Team aber nicht nach außen gegeben
  - Es wird sich für Belandlosigkeite bedankt
    - Führung weiß nicht, was die Leute tun und/oder was wichtig ist.
  - Anerkennung nur für Außergewöhnliches
    - Anerkennung oft sehr fokussiert auf den der aus seiner Komfortzone geht
    - Keine/Kaum Anerkennung für Leistungsträger im Team die dauerhaft auf hohem Niveau arbeiten
      - “Macht ja immer gute Arbeit”

        - Teams sind Individuen und mache sind mehr für ein Thema Motiviert als andere
            - Durch persönliche interessen or Erfahrungen
            - Unterschiede akzeptieren und beim Feedback berücksichtigen.
        - Auch Anerkennung für offensichtliches geben, Arbeit die im Hintergrund passiert würdigen!
- **Planung wichtiger als Transparenz**
    - Feingranulare Planung wird eingefordert, Resultat sind aber größtenteils irrelevant
        - Solange der Issue Tracker regelmäßig aktualisiert wird ist alle ok
    - Statt Planung besser Transparenz durch
        - Demos und funktionierende Software herstellen
        - Echte Use Cases abbilden und mit (zukünftigen) Benutzern validieren
    - Planung oft gewünscht weil
        - “das Ging ja schon mal schief” (Letztes modernisierungs Projekt ist Fehlgeschlagen)
        - “Das Projekt ist wichtig”
    - Besonders in frühen und explorativen Phasen eines Projekts, in welche Richtung noch nicht klar ist, wird viel Overhead erzeugt.
    - Reporting über Projektfortschritt wird von der primären Datenquelle z.B. Issue Tracker, entkoppelt z.B. Excel Report
        - Verantwortung geht vom Team an den Reporter über
        - Team fühlt sich weniger verantwortlich Status zu reflektieren
- **Zu wenig Willen zur Veränderung**
    - Extrem lange Projektlaufzeiten und Ziele sorgen für “Gelassenheit”
    - Komfortable Position im Markt erfordert keine Veränderung von außen aber intern klemmt es überall
    - Bürokratie und alte Prozesse werden nicht verändert weil kein Wille da ist
    - **Beschäftigt sein ist wichtiger als das richtige zu machen.**
- **Nicht aus der Vergangenheit lernen**

- Man spricht nicht über Fehlschläge aus der Vergangenheit und analysiert diese nicht systematisch um daraus zu lernen
- Selbst wenn diese Analyse passiert, ist teilweise kulturell nicht gewollt, dass ein neues Team beim nächsten Versuch Zugang zu diesen Informationen hat.
    - Da Fehler als persönliche Niederlage gesehen werden will man die Beteiligten des letzten Versuchs schützen
    - Nimmt die Möglichkeit aus Fehler zu lernen und zu kommunizieren, warum sie sich nicht wiederholen.
        - Erzeugt die Wahrnehmung: “Das geht wieder schief!”.

Zwei Arten von Ursachen

- Unbeabsichtigt/Beabsichtigt
- **Überwiegenden Mehrheit ist unbeabsichtigt**

Unbeabsichtigte Probleme

- Ursache:
  - Fehlende Informationen.
  - Falsches Rollenverständnis auf Grund neuer Situation.
  - Mangelnde Kommunikation.
- Reden/ansprechen hilft oft unmittelbar, weil das Problem bisher "unsichtbar" war.

Beabsichtigte Probleme

- Ursachen:
  - Aktives Unterdrücken von Motivation/Informationen.
  - Gezieltes Erzeugen von organisatorischem Chaos.
- Ansprechen hilft oft nicht unmittelbar.
- Im Zweifelsfall Organisatorische Konsequenzen ziehen
  - Toleranz-Paradoxon