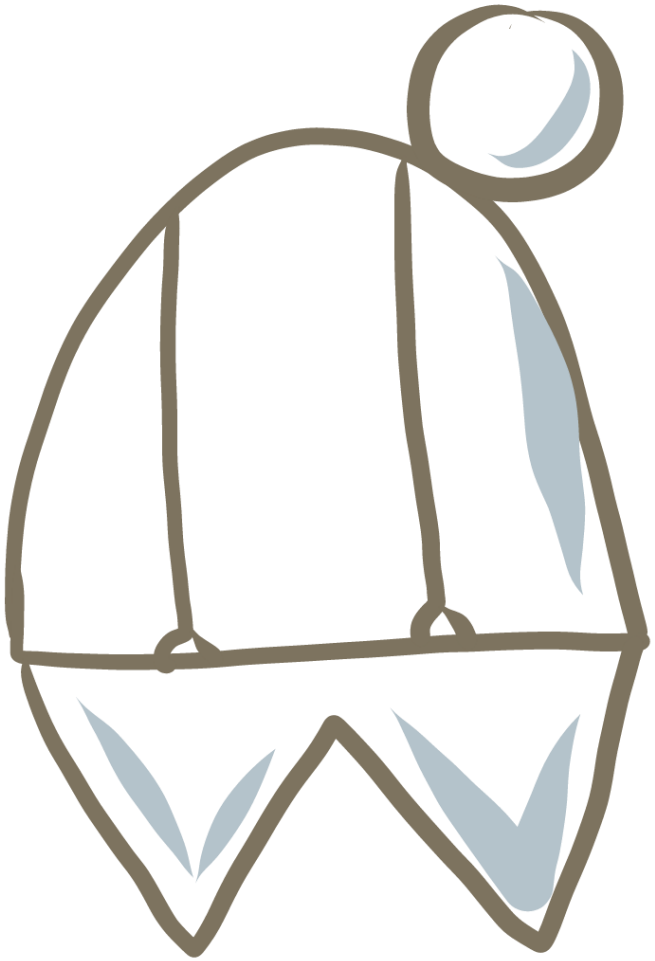
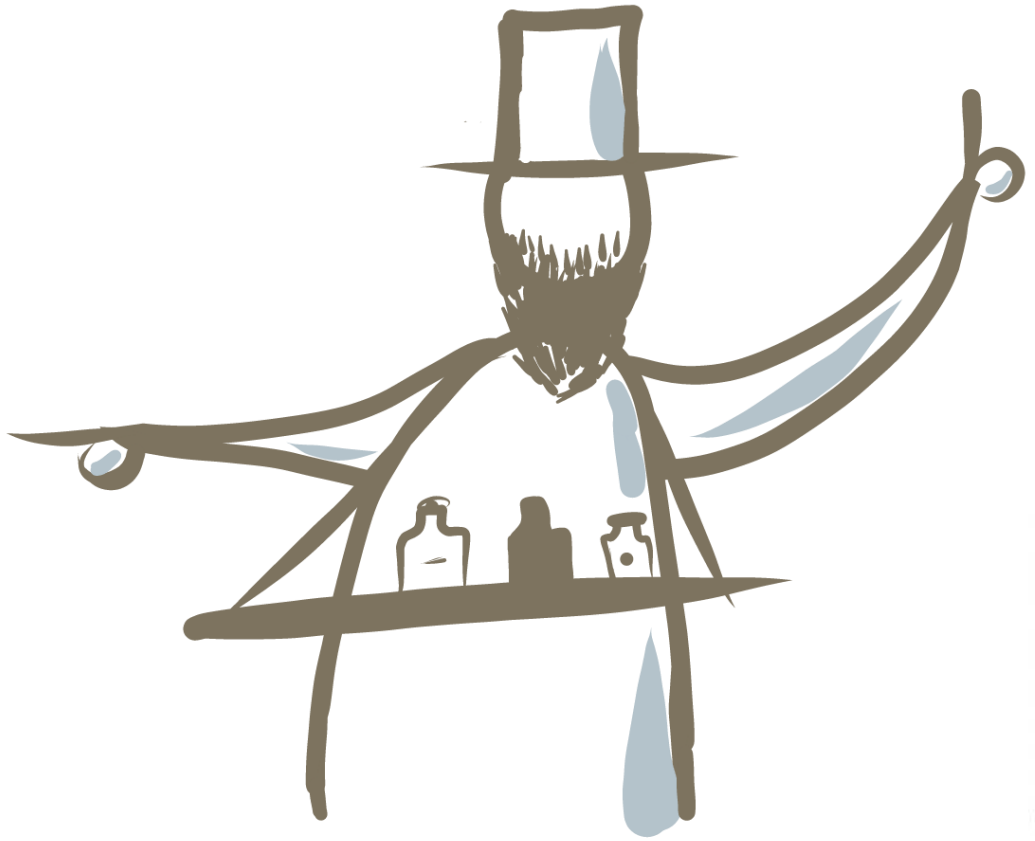
Der Dicke und sein Wunderheiler



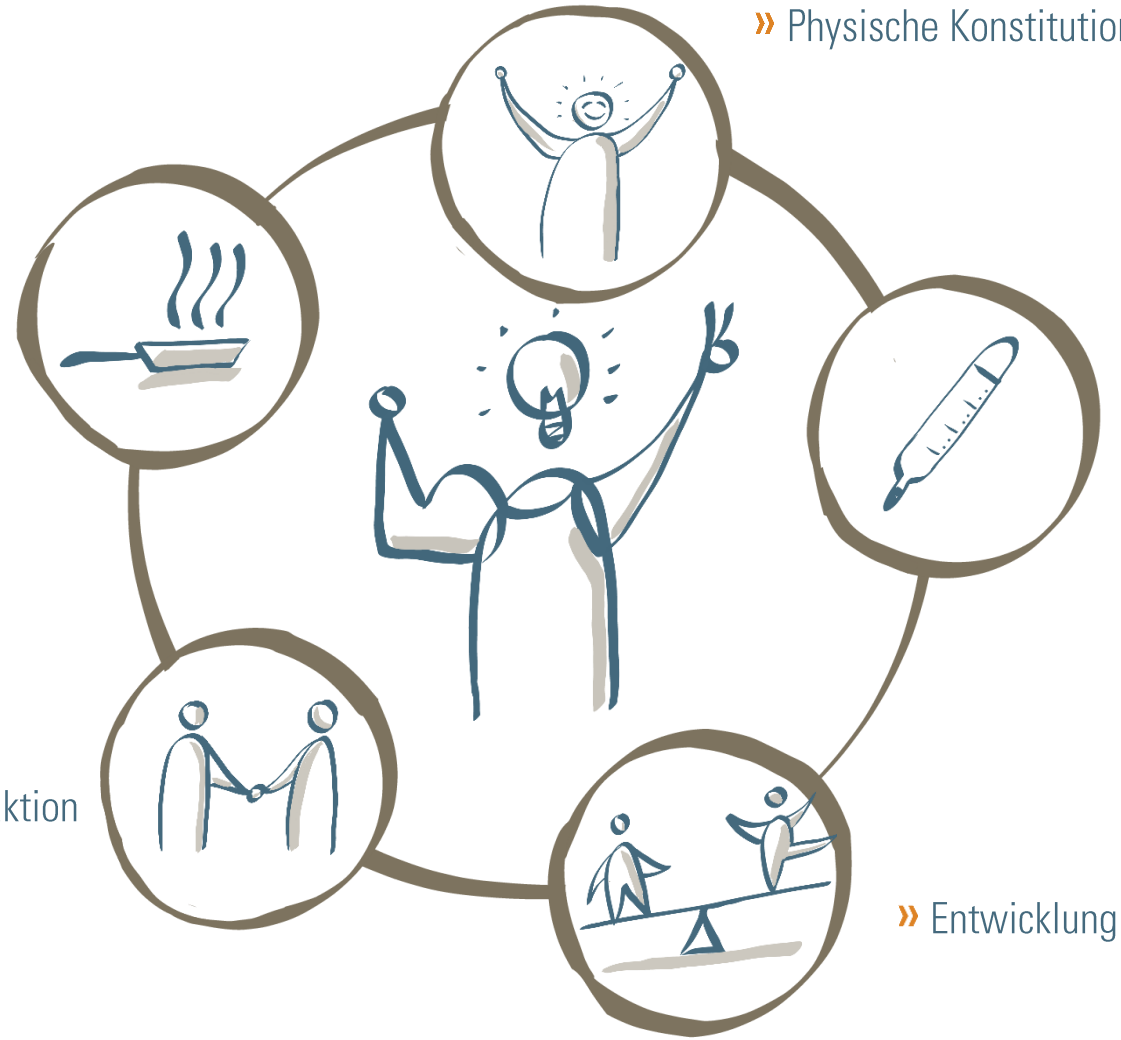


» Tilmann Glaser

» @TGnowledgy

Wie Architektur die Gesundheit unserer Organisation beeinflusst

Gesundheit fußt auf fünf Faktoren



» Physische Konstitution

» Ernährung

» Umgang mit Krankheiten

» Soziale Interaktion

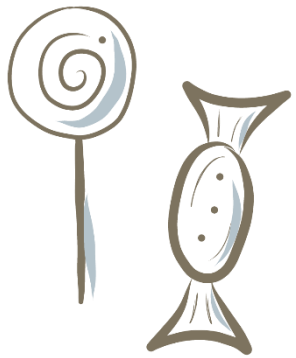
» Entwicklung im Gleichgewicht

Convenience Food und Süßigkeiten sind verlockend

Ernährung

Die Digitalisierung erhöht den Entwicklungsdruck bei Softwareprojekten stetig. Die Entwickler-Ressourcen werden knapper.

Der Wunderheiler empfiehlt



Convenience Code

- Zeitersparnis durch „Fertigprodukte“
  - Copy & Paste nutzt erprobten Code
  - Heavyweight-Codegeneratoren erstellen die Anwendung „fast von selbst“
- Keine weitergehenden Entwickler- / Architektenkenntnisse notwendig
- Multiplattformfähigkeit
- Dokumentationsarm

Magic Tech-Candy

- Technologie als Problemlöser
- Technologie als Motivator
- Einschätzung: Auswirkungen auf Business-Case gering

Bewusste Ernährung hält langfristig schlank

Ernährung

Maßvoller Einsatz von Fertigprodukten

- „Küchenfertige“ Hilfsmittel, die mühsame, stupide Arbeit erleichtern
  - Boilerplate-Code
  - Dependency Injection / Laufzeit-Container
  - O/R-Mapping
- Am Rand der Anwendung nutzen
  - Anwendung muss auch ohne die Hilfsmittel betreibbar sein / Hilfsmittel austauschbar
  - Libraries over Frameworks
- Technische Schulden != Cruft
  - Kent Beck‘s „Four Rules of Simple Design“ beachten

Nachtisch und Sonntagskuchen

- Architektur so aufbauen, dass Tech-Entscheidungen bewusst lange offen bleiben und revidiert werden können.
- „Austoben“ in der Peripherie, den Kern der Anwendung in vertrauter, reifer Technologie
- Vom Konzept zur Technologie, nicht andersherum
  - z.B. Welche Ausprägungen des CAP-Theorems sind relevant? → AP - Hadoop, CP - MongoDB oder CA - mySQL?

Ein Jogginganzug alleine macht keinen Sportler

Physische Konstitution

Unsere Anwendungen müssen dynamisch und schnell deployed werden können. Wir brauchen strapazierfähige Laufzeitumgebungen. Um langfristig erfolgreich zu sein, müssen unsere Anwendungen schlank bleiben.

Der Wunderheiler empfiehlt

Modische Sportbekleidung

» Containervirtualisierung à la Docker

» Deployment in der Cloud

Mitgliedschaft im Fitnessstudio

» Persönliche Trainer

- Ein „Ops-Team“ bringt das Inkrement des „Dev-Teams“ in kürzester Zeit in Form
- Nachträgliche Erhöhung der Testabdeckung strafft unsere Anwendung

Diätpillen

» Qualitätsinitiativen nach jedem Release

» Refactoring Sprints

Regelmäßiges Training führt zu gesunder Routine

Physische Konstitution

Tun!

» Gedanken zu Deployment und Betrieb beginnen bei den Architekturzielen
» DevOps beginnt auf dem Entwicklerrechner

Sportverein

» Regelmäßiges Training zu festen Zeiten im Team
» Ausgewogene Bearbeitung identifizierter Schwachstellen
» Gemeinsame Auseinandersetzung mit Code und Codequalität
  • Alignment des Code an die Architekturziele

Treppe statt Fahrstuhl

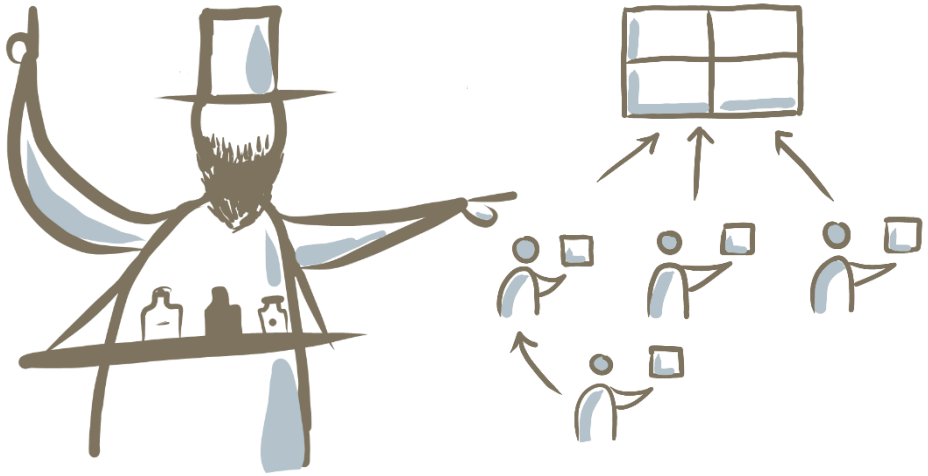
» Outside-In TDD fördert stetige Refactorings
  • Ein kleines, sinnvolles Refactoring zahlt sich oft schon vor dem nächsten Release aus
» Kleinschrittig vorgehen statt lange an ganzheitlicher Umsetzung von Features zu arbeiten
» Mut zu kleinen Experimenten

Zusammenarbeit löst nur scheinbar alle Probleme

Soziale Interaktion

Unsere alten Monolithen sind träge bei Erweiterungen, unperformant im Betrieb und voller „Single Points of Failure“. Unsere neuen Anwendungen müssen sich dieser Probleme annehmen

Der Wunderheiler empfiehlt



Viel hilft viel

» Probleme mit Komplexität erschlagen
» Bei Fehlschlag, Lösung größer machen

Herdenschutz

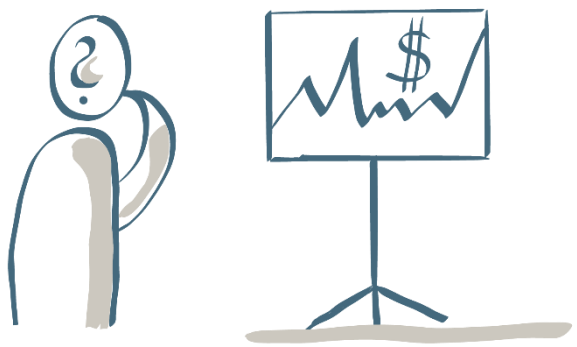
» Verteiltes System
» Microservices

Gesundheitspolizei

» Verfügbarkeitspflicht
» Detaillierte Umsetzungsvorgaben

Eigenverantwortlich handeln – über den Tellerrand hinaus

Soziale Interaktion



Deal with Reality

» Technische Restriktionen sind (auch) Business-Probleme

» Fokussieren aufs Wesentliche

- Gute Architekten beherrschen die Kunst, Systeme zu vereinfachen

Richtlinien aufstellen

» Welche beobachtbaren Kriterien soll ein Service erfüllen

z.B.

- Unabhängigkeit zu anderen Services (Anti-Corruption Guideline)
- Änderungsaufwand wenn sich Input / Output Strukturen ändern
- Log-Format und –Ziel

Erfahrungsaustausch

» Best-Practices

» Epic Fails

Gesundheit erzwingen führt zum Gegenteil

Umgang mit Schmerzen und Krankheiten

it-economics
a Sopra Steria company

Wenn der Projektstress steigt wird das Immunsystem unserer Anwendung strapaziert. Gerade jetzt ist es wichtig, aufkommenden Leiden entgegenzuwirken

Der Wunderheiler empfiehlt

Schmerztabletten

- Mocking-Frameworks
- Dependency-Stubbing Tools

ACE-Hemmer

- Anzahl der Builds reduzieren
- Seltener testen

Ignorieren

- Fehlschlagende Tests auskommentieren
- Temporär „Abkürzungen“ in den Code einbauen

Auf Anzeichen achten und rechtzeitig entschleunigen

Umgang mit Schmerzen und Krankheiten

Schmerzen auf den Grund fühlen

» Ein allgemeines Problem (z.B. Fieber) kann viele Ursachen haben
» Überlastung durch Code-Rushing?
» Zu enge Kopplung an Peripherie?
» Konzeptioneller Fehler im System?
» Vereinfachungen möglich?
» Team zu groß?

Individuelle, passgenaue Lösung suchen

» Zu einer allgemeinen Ursache kann es sehr unterschiedliche Lösungen geben
» Probleme transparent machen
» Gemeinsam mit Business-Seite Gegenmaßnahmen planen

Ausgleich

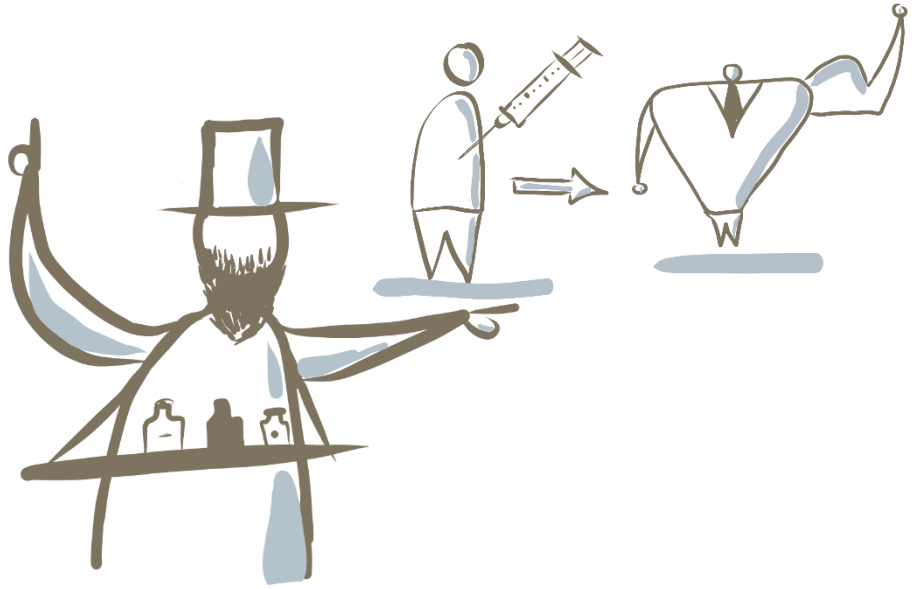
» Sport baut Stress ab und senkt den Blutdruck

Wachstumsbeschleunigung hat Quereffekte

Entwicklung im Gleichgewicht

Früher haben wir am Kundennutzen vorbeientwickelt. Zukünftig müssen wir inkrementell vorgehen um schneller zu sein

Der Wunderheiler empfiehlt



Sequenzielles Wachstum

» Schrittweise Umsetzung der einzelnen Anwendungsteile

» Vertikaler Anforderungs-Schnitt

- Eine Anforderung pro Prozessschritt
- Ganzheitliche Umsetzung des Schritts

Frühzeitig Komponenten festlegen (BDUF)

» Klare, stabile Struktur ins System bringen

» Skalierbarkeit des Teams entlang des Komponentenschnitts sichern

Ausbalancierte Entwicklung führt zu gesunden Systemen

Entwicklung im Gleichgewicht

Ende-zu-Ende denken

» Relevante Architekturziele festlegen (RGUF)
  - Performance
  - Verfügbarkeit
  - Skalierbarkeit
  - Traffic
  - Betriebskosten
» Horizontaler Anforderungsschnitt
  - Minimaler Prozess
  - Happypath
  - Erste Restriktionen
  - Entscheidungszweige

System aufwachsen lassen

» Beginn als Minimonolith
» Regelmäßiges Refactoring
» Weiter Aufteilen immer nur bei Bedarf

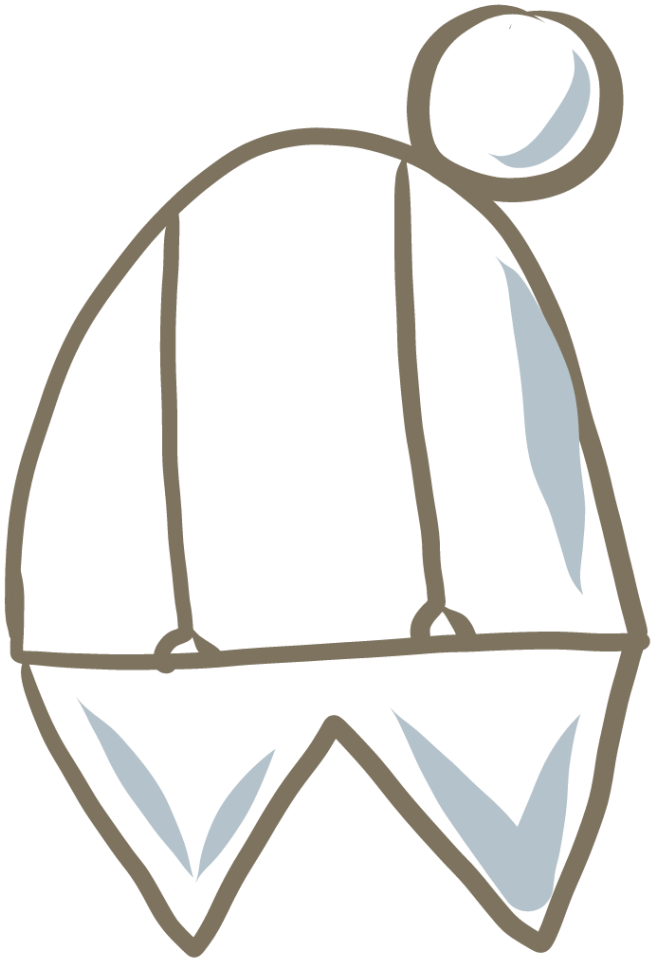
Architektur-Guidelines für eine gesunde Organisation

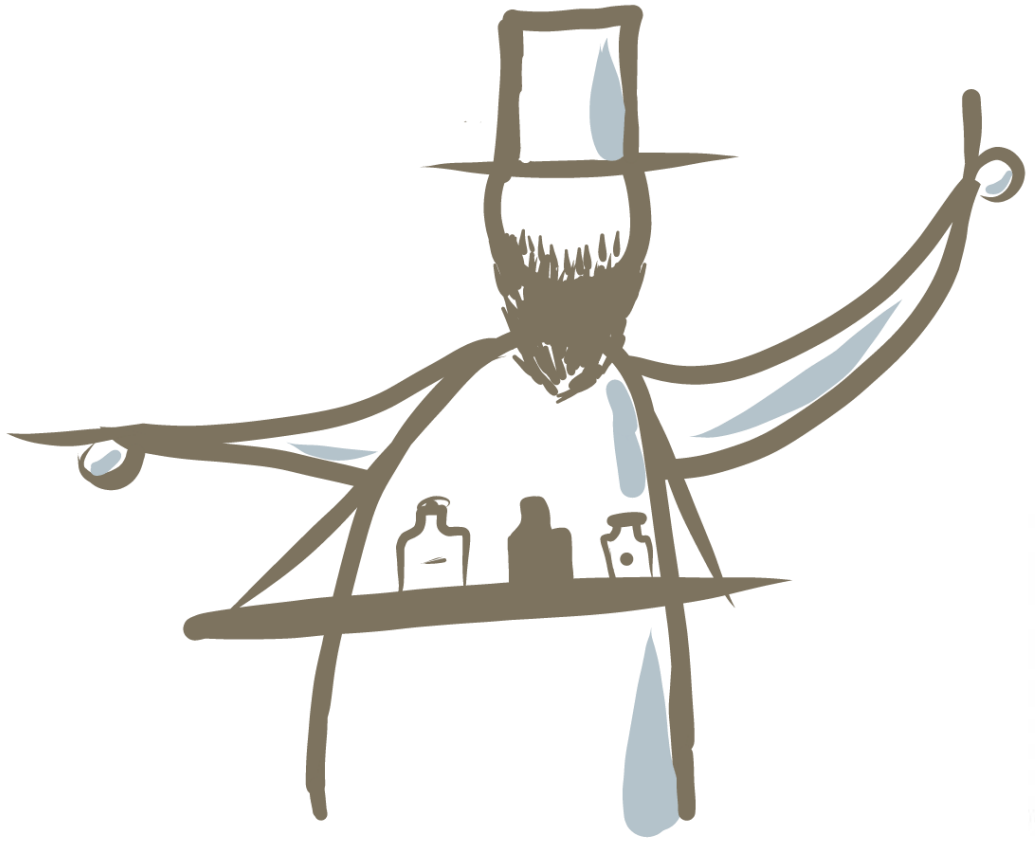
Fazit

- Wir sind für die Gesundheit unserer Anwendungen selbst verantwortlich
- Wer keine Zeit für seine Gesundheit hat, wird später viel Zeit für seine Krankheiten brauchen.
  - (Sebastian Kneipp 1821 – 1897)

- Lasse den Business-Value deiner Anwendung von den besten Leuten deines Teams programmieren
  - Sündige bewusst
- Mache Verbesserung zur Routine
- Konzeptuelle Herausforderungen lassen sich nicht durch den Einsatz von Tools und Frameworks lösen.
  - Aber Tools und Frameworks können gute Konzepte sinnvoll flankieren
- Technische Restriktionen sind Business-Probleme
  - Es wird eine fachliche Lösung brauchen, die im Dialog mit der Business-Seite entsteht.
- Beachte deine Codebasis, dein Team und das Umfeld
  - Lenke frühzeitig ein. Nutze Energie für konstruktive Verhandlungen und nicht um Dinge zu erzwingen
- Sei ein guter Entwickler, sei ein Servant Leader
  - Halte Kontakt zur Basis
- Lasse dein System aufwachsen
  - Trainiere deine Anforderer, Stories richtig zu schneiden

Der Dicke und sein Wunderheiler





» Tilmann Glaser

» @TGnowledgy

Wie Architektur die Gesundheit unserer Organisation beeinflusst