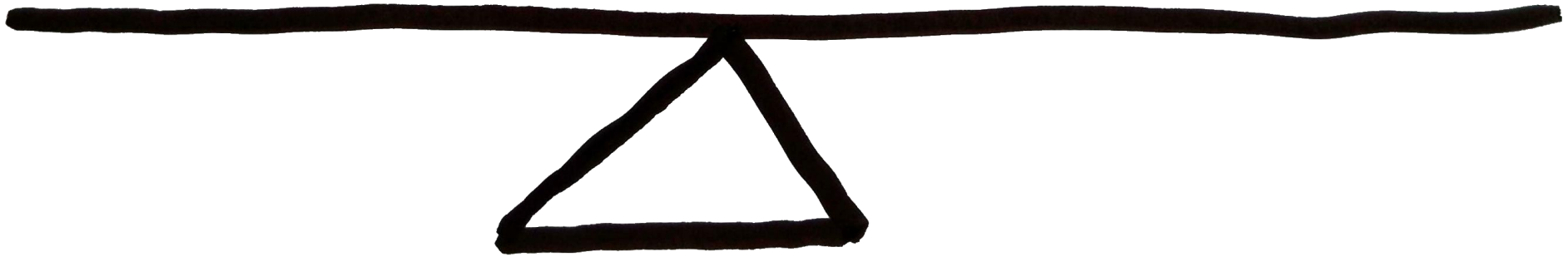
Peter Götz
Agile Coach, Trainer & Software Architect

Sprintlänge

Wir haben so kurze Sprints, dass wir uns kaum auf das Abschließen von Arbeit fokussieren können

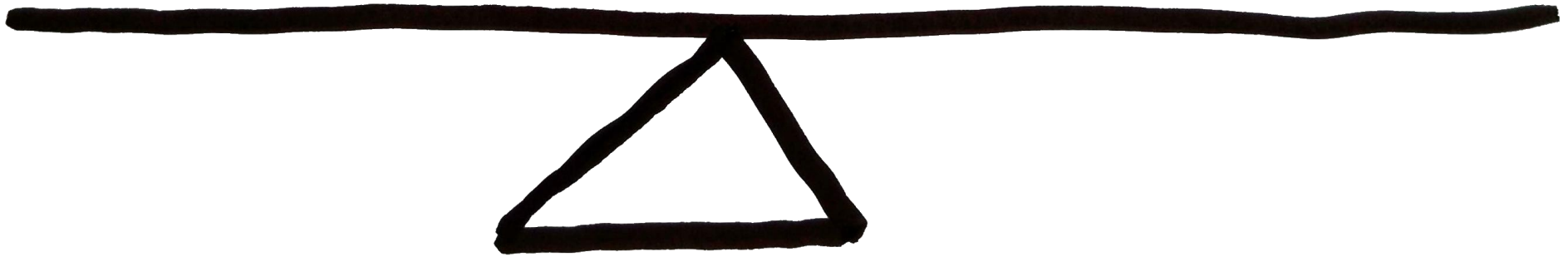


Wir haben so lange Zyklen, dass wir viele Annahmen anhäufen, bevor wir Feedback erhalten

Peter Götz
Agile Coach, Trainer & Software Architect

Planungshorizont

Wir planen das gesamte Produkt mit allen Details im Voraus

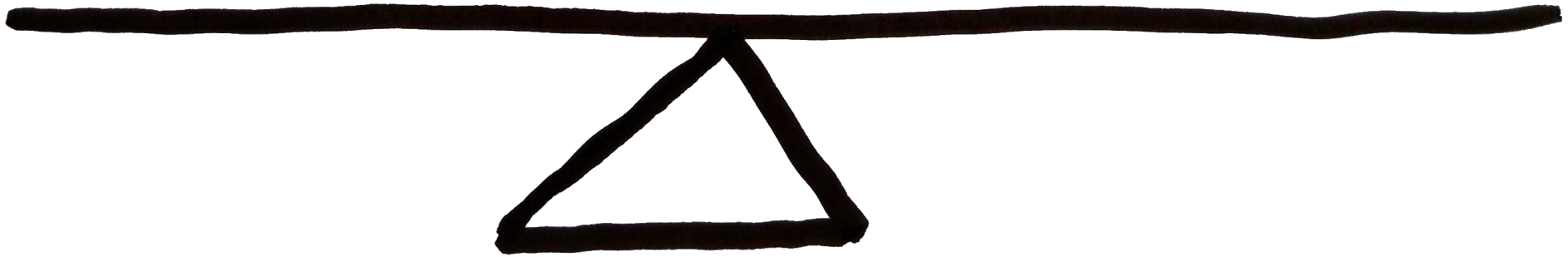


Wir planen höchstens den nächsten Sprint, alles andere ergibt sich dann

Peter Götz
Agile Coach, Trainer & Software Architect

WIP & Fokus

Wir limitieren gleichzeitige Arbeit gar nicht, jedes Teammitglied kann jederzeit eine neuen Aufgabe beginnen

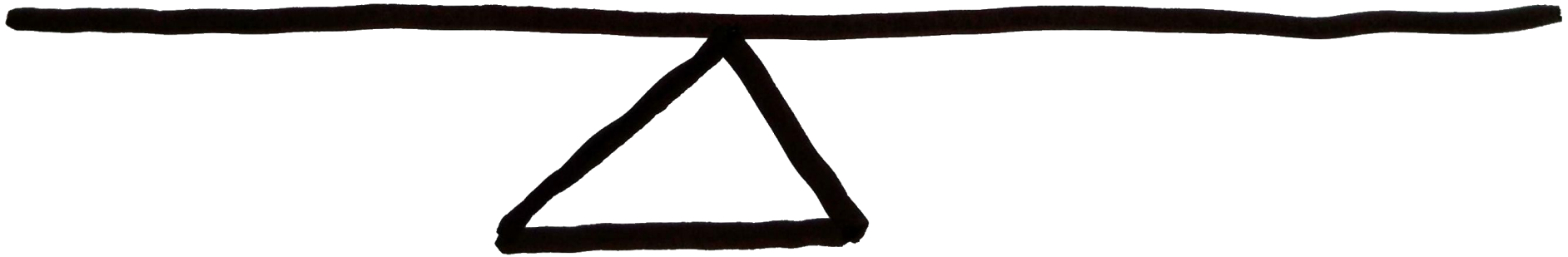


Wir limitieren gleichzeitige Arbeit so stark, dass oft Leerlauf entsteht

Peter Götz
Agile Coach, Trainer & Software Architect

Release-Frequenz

Wir liefern nur 2 mal im Jahr ein neues Release, dazwischen höchstens kritische Fixes



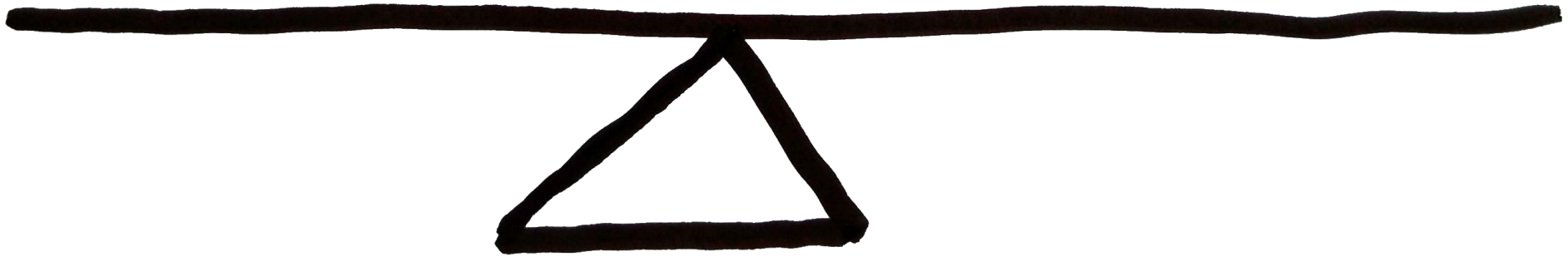
Wir releasen ständig, was unsere Nutzer überfordert und die Qualität beeinträchtigt

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Time-to-Market vs. Stabilität

Geschwindigkeit ist alles, Stabilität ist zweitrangig



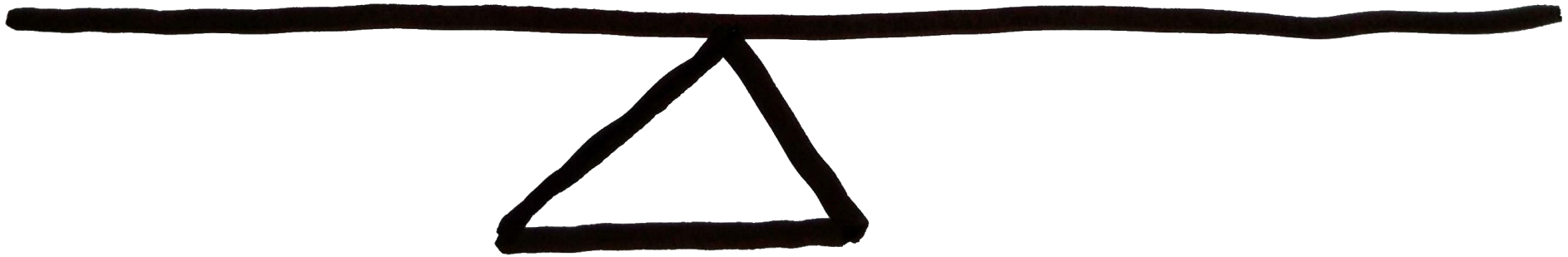
Stabilität ist alles, neue Features sind zweitrangig

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Arbeitsbelastung

Wir arbeiten dauerhaft am Limit, gefühlt brennt es immer irgendwo

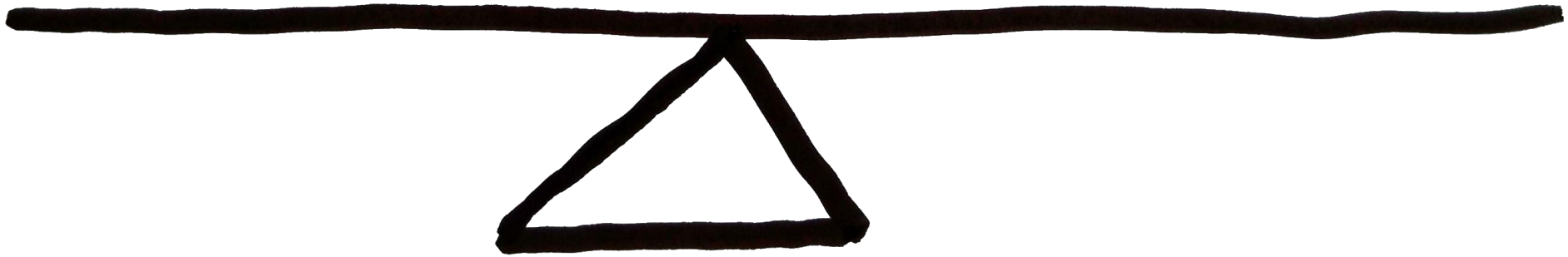


Es gibt keinen Druck, Dinge dauern so lange, wie sie eben dauern

Peter Götz
Agile Coach, Trainer & Software Architect

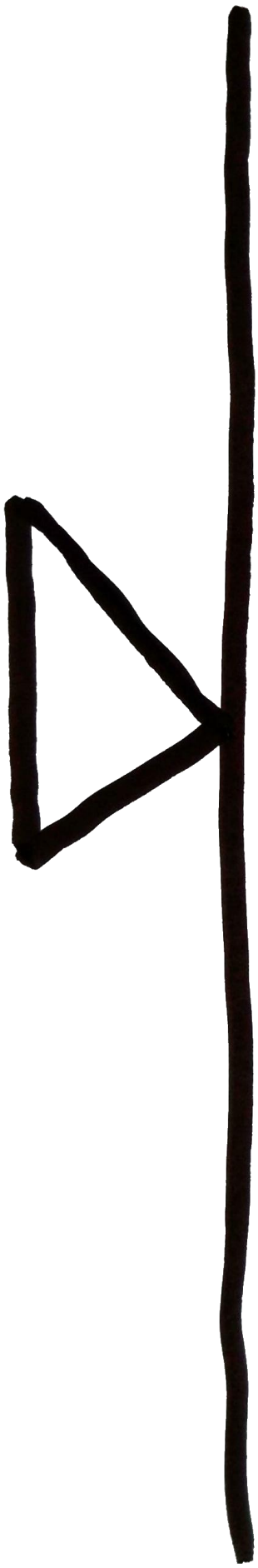
Code Refactoring

Wir sind ständig am Anpassen von Code, weil wir immer noch bessere Patterns finden



Wir ändern bestehenden Code nie, weil das zu riskant oder aufwändig ist

Wir achten peinlichst darauf, immer alle Clean-Code Regeln einzuhalten. Code, der nicht unseren hohen Standards genügt darf nicht eingecheckt werden.



Code-Qualität

Um möglichst schnell zu sein, achten wir zunächst gar nicht auf Code-Qualität, Code wird erst dann optimiert, wenn es Probleme damit gibt

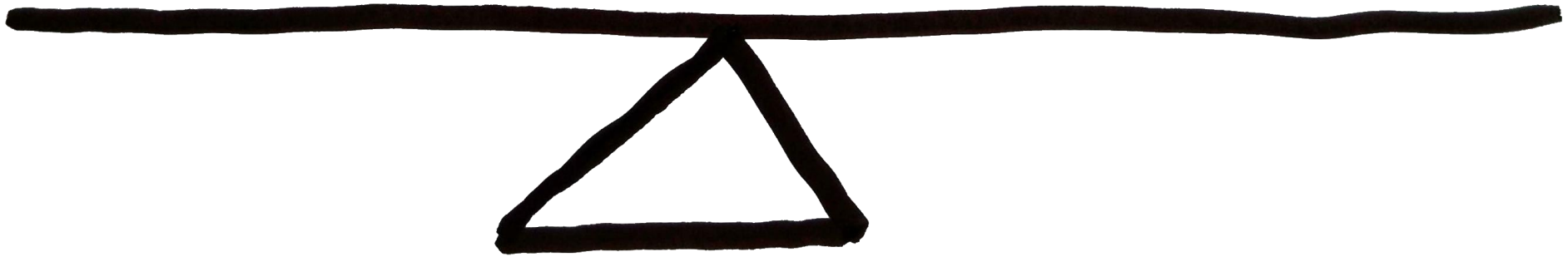
agileMax
Maximize Your Agility
Peter Götz
Agile Coach, Trainer & Software Architect

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Architektur-Entscheidungen

Architektur wird upfront für Jahre im Voraus festgelegt

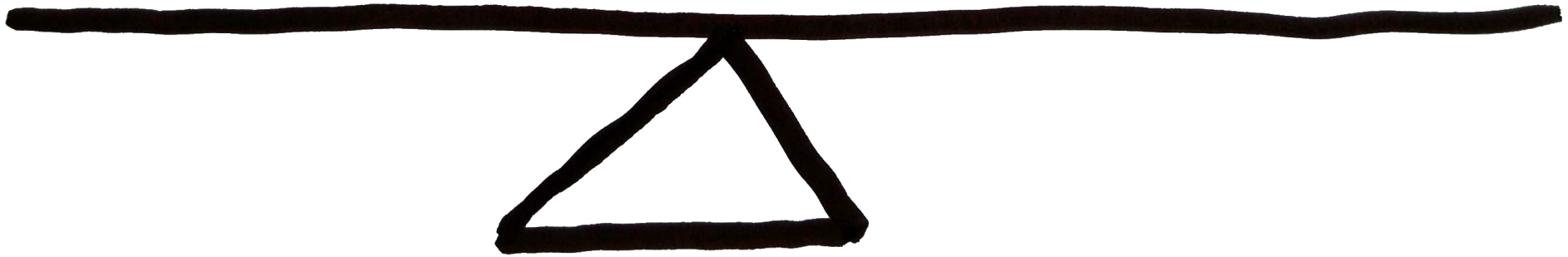


Architektur entsteht ausschließlich reaktiv, wenn Probleme auftreten

Peter Götz
Agile Coach, Trainer & Software Architect

Technologieauswahl

Wir greifen jeden neuen Technologie-Trend sofort auf und arbeiten gerne auch mit Beta-Versionen



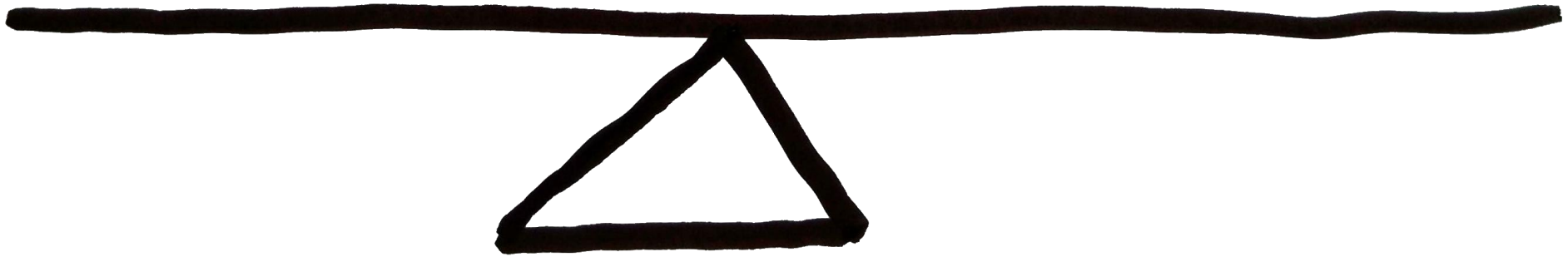
Wir bleiben strikt bei bewährten Technologien und vermeiden jeden Wechsel

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Monolith vs. Microservices

Wir packen alle Funktionalitäten in ein einziges monolithisches System



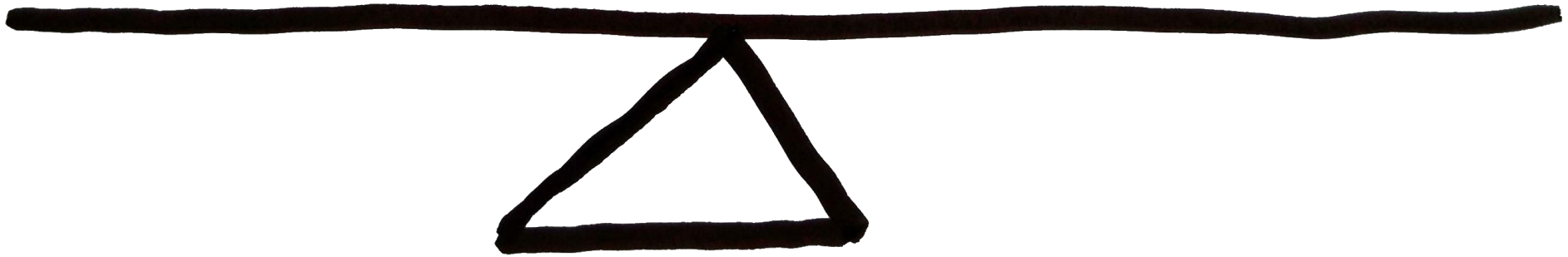
Das System besteht aus sehr vielen Microservices, selbst für kleinste Funktionen

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Einfachheit vs. Erweiterbarkeit

Wir gestalten das System so spezifisch wie möglich, um Komplexität zu minimieren

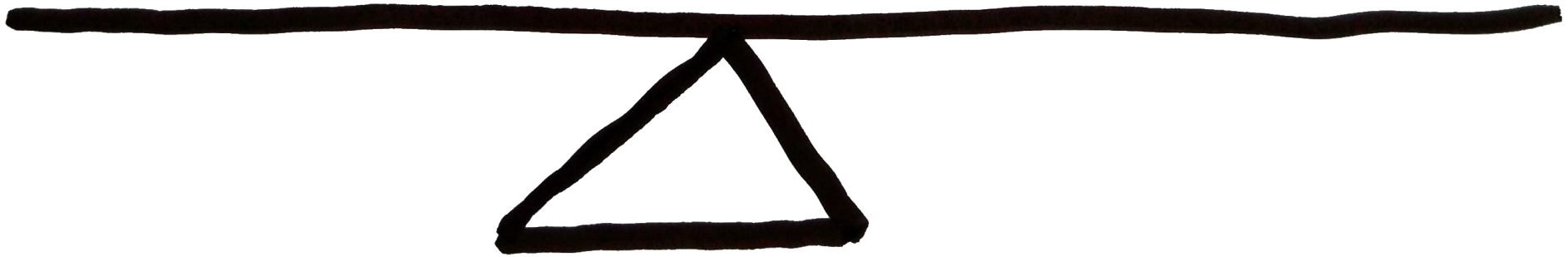


Wir versuchen, das Design maximal generisch und flexibel zu halten, um zukünftige Anpassungen schneller umsetzen zu können

Peter Götz
Agile Coach, Trainer & Software Architect

Bibliotheken vs. Eigenentwicklung

Für alle wichtigen Bereiche entwickeln wir eigene Bibliotheken, um volle Kontrolle zu haben

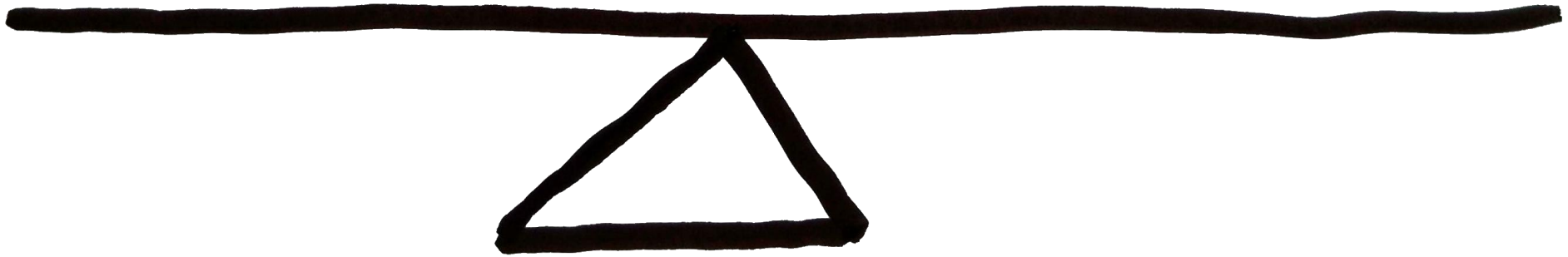


Wir nutzen überall bestehende Bibliotheken, um Zeit zu sparen

Peter Götz
Agile Coach, Trainer & Software Architect

Security

Sicherheit wird erst betrachtet, wenn Probleme auftreten

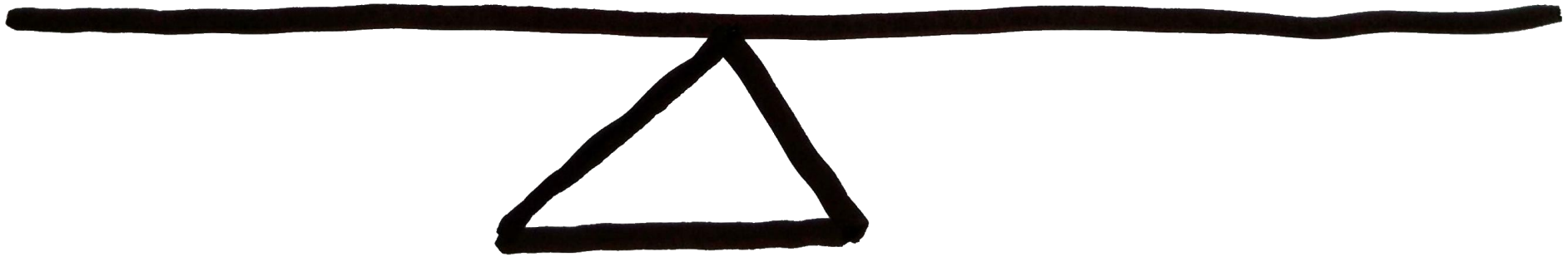


Sicherheitsanforderungen dominieren alle Entscheidungen

Peter Götz
Agile Coach, Trainer & Software Architect

Testabdeckung

Wir testen gar nicht und reagieren nur auf Feedback der Anwender

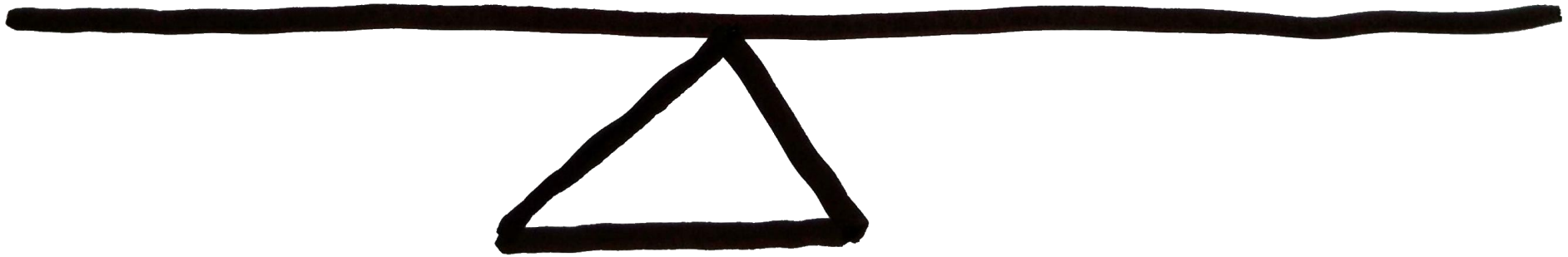


Ein Release kann nur freigegeben werden, wenn alle bestehenden Funktionen eingehend getestet wurden

agileMax
Maximize your Agility
Peter Götz
Agile Coach, Trainer & Software Architect

Manuelles vs. automatisiertes Testen

Alle Tests werden vollständig automatisiert, die Testanzahl wächst ständig

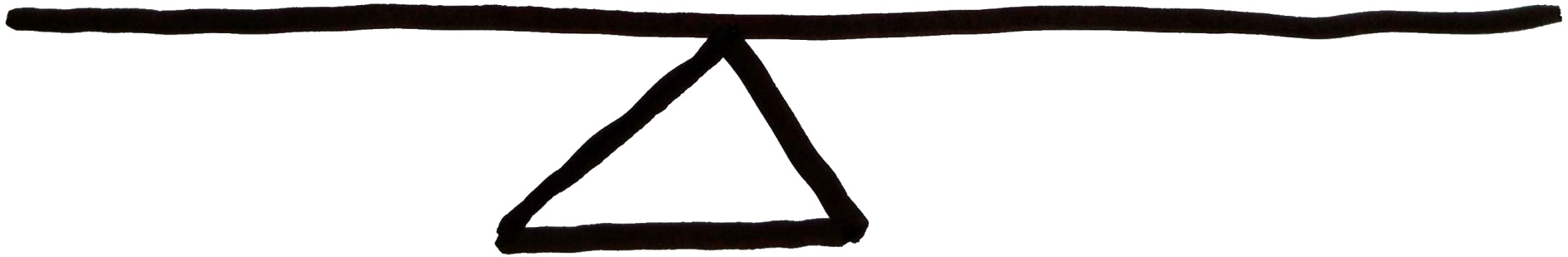


Alle Tests werden ausschließlich manuell durchgeführt

Peter Götz
Agile Coach, Trainer & Software Architect

Qualitätssicherung

Qualitätssicherung ist ausschließlich Aufgabe dedizierter QS-Experten

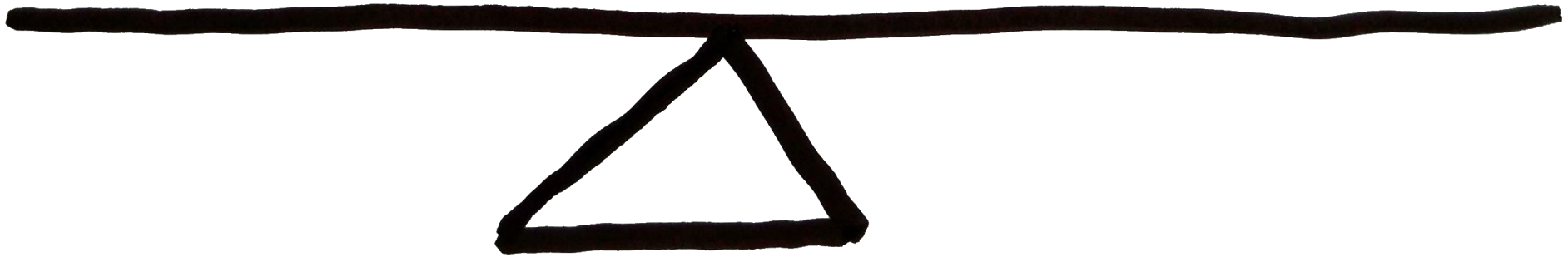


Jede Entwicklerin ist allein für die Qualität des eigenen Codes verantwortlich

Peter Götz
Agile Coach, Trainer & Software Architect

Versionsunterstützung

Wir supporten auch noch sehr alte Versionen, auch wen das viel Aufwand bedeutet

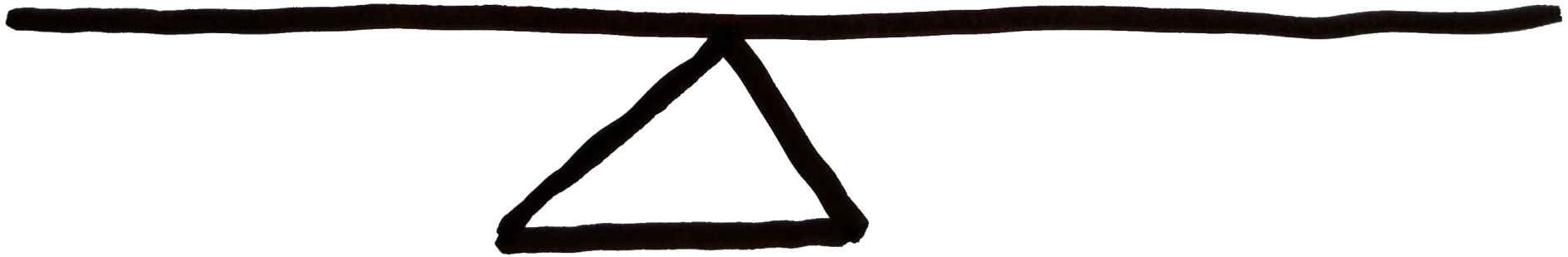


Nutzer werden immer zur neuesten Version gezwungen

Peter Götz
Agile Coach, Trainer & Software Architect

Dokumentation

Wir schreiben keine Dokumentation, der Code ist Dokumentation genug

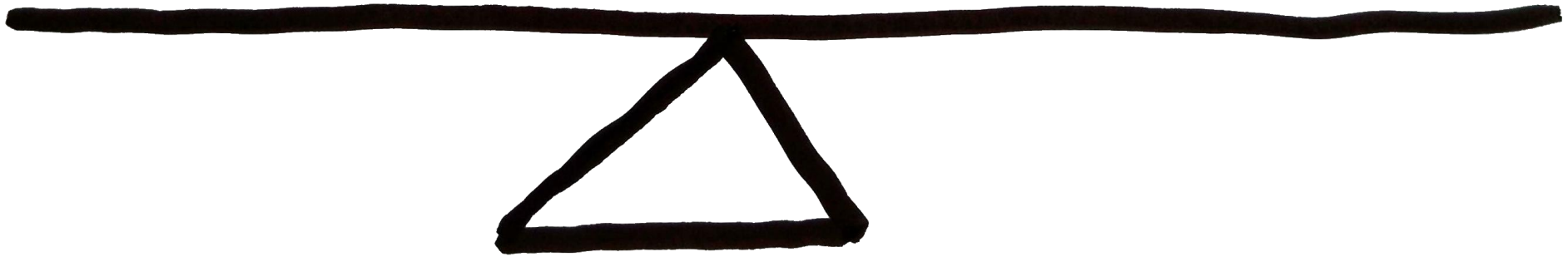


Wir dokumentieren alles, was irgendwann einmal hilfreich sein könnte

Peter Götz
Agile Coach, Trainer & Software Architect

Automatisierung

Wir automatisieren alles, was sich irgendwie automatisieren lässt



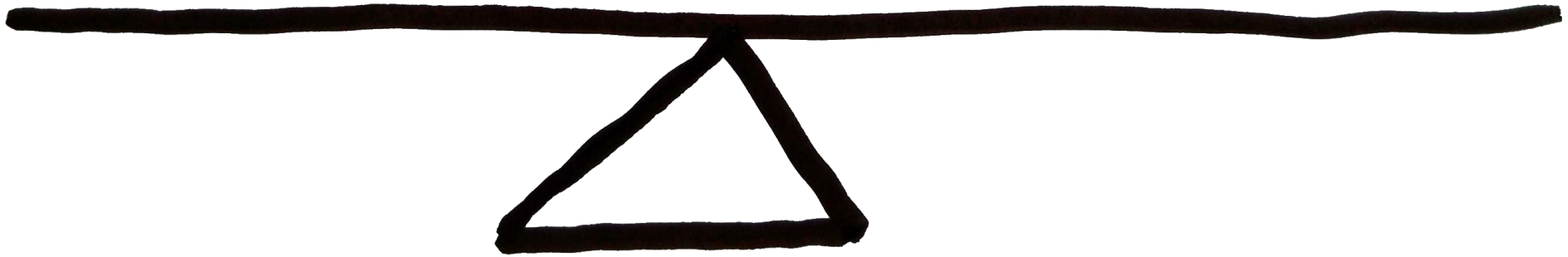
Wir automatisieren nur Dinge, die sich mit ein paar Klicks automatisieren lassen

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Kundenwünsche vs. Produktvision

Wir versuchen, jeden einzelnen Kundenwunsch sofort umzusetzen



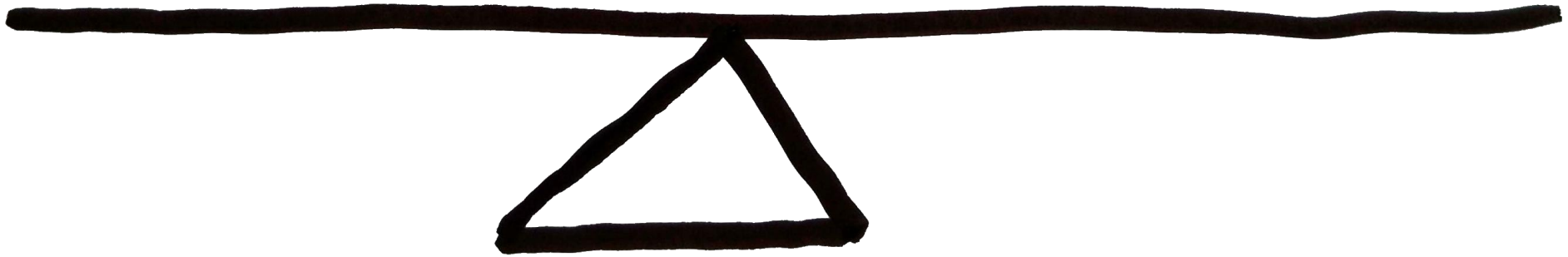
Wir halten stur an unserer internen Produktvision fest und ignorieren das Feedback der Anwender

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Hypothesen vs. Anforderungen

Alles ist eine Hypothese, alles kann sich jederzeit ändern

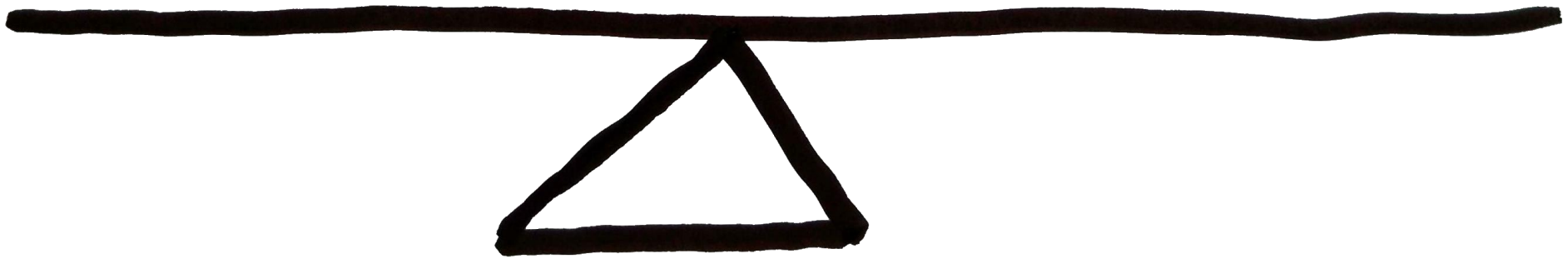


Alles sind fixe Anforderungen, die stur nach Plan abgearbeitet werden

Peter Götz
Agile Coach, Trainer & Software Architect

Feedback & Lernen

Feedback entsteht nur über formale Reviews und Umfragen

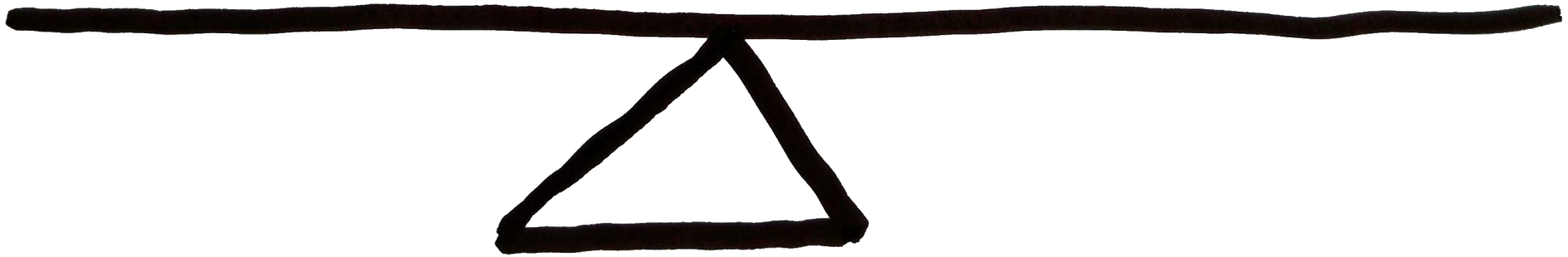


Feedback entsteht ausschließlich informell und ungeplant

Peter Götz
Agile Coach, Trainer & Software Architect

Experimente

Wir experimentieren ständig und versuchen immer neue, bessere Lösungen zu finden

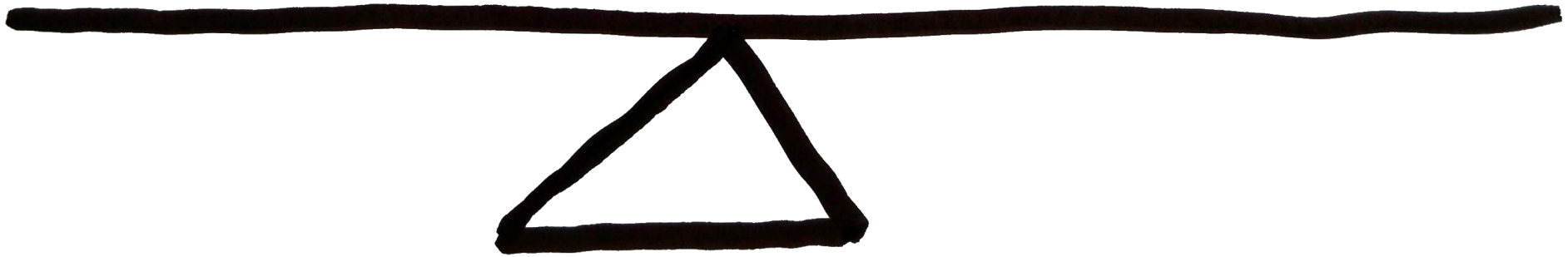


Wir vermeiden Experimente um Aufgaben schnell und planbar abzuarbeiten

Peter Götz
Agile Coach, Trainer & Software Architect

Impediments

Hindernisse werden grundsätzlich immer dem Scrum Master übergeben

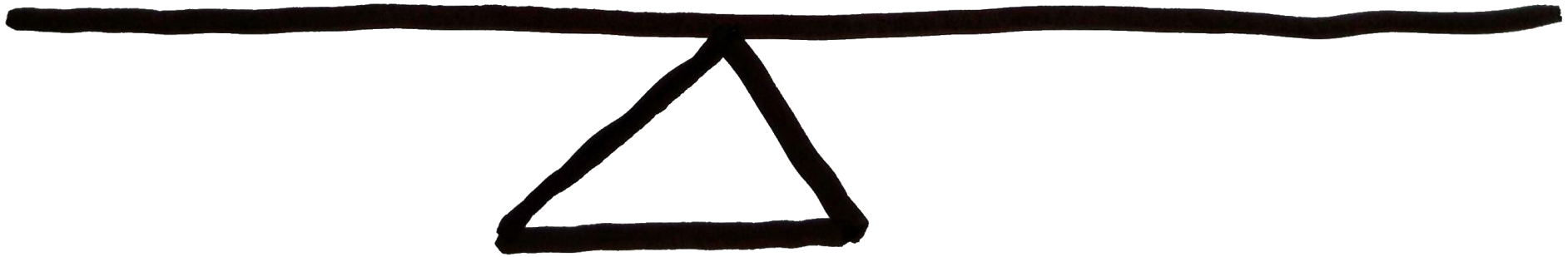


Jedes Teammitglied kümmert sich um seine eigenen Probleme / Hindernisse

Peter Götz
Agile Coach, Trainer & Software Architect

Generalisten vs. Spezialisten

Bei uns kann jedes Teammitglied alle anfallenden Aufgaben erledigen



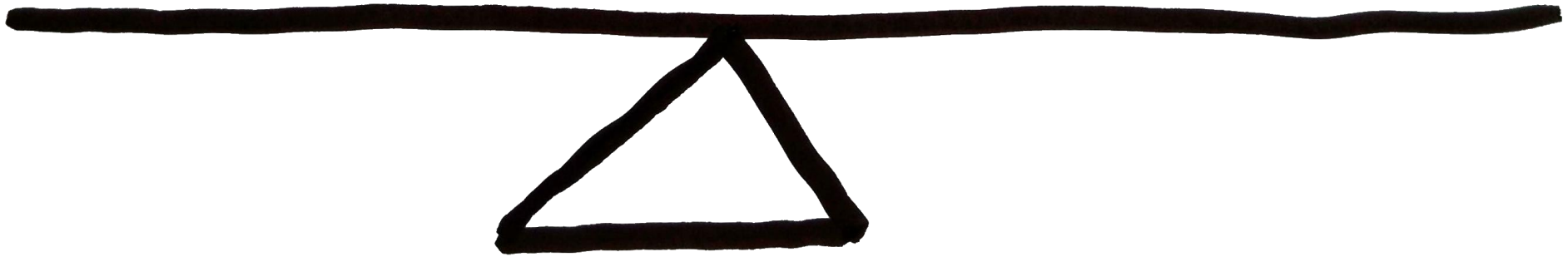
Bei uns werden Aufgaben immer von der Person umgesetzt, die sich in dem Thema am besten auskennt

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Wissensaustausch im Team

Wissens- und Erfahrungsaustausch kostet sehr viel Zeit, Zeit, die wir aktuell nicht haben

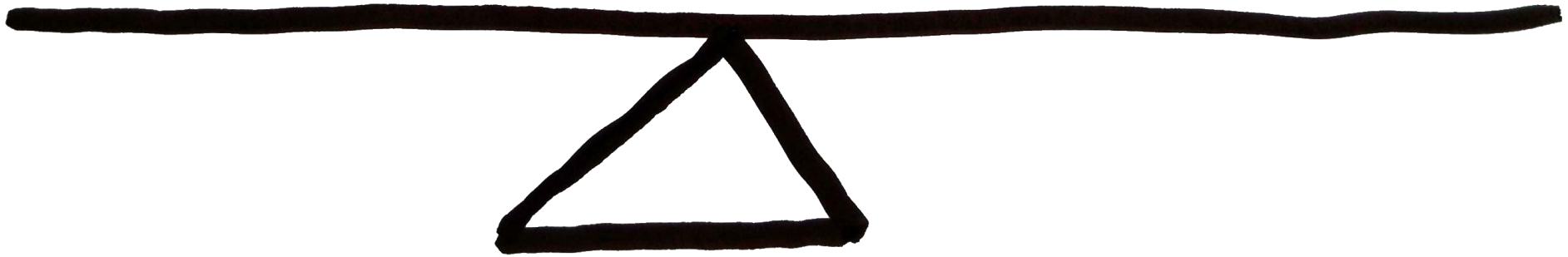


Bei uns wissen immer alle über alles bescheid

Peter Götz
Agile Coach, Trainer & Software Architect

Kommunikation

Wir haben viele Meetings / Diskussionen damit alle informiert und in Entscheidungen einbezogen sind

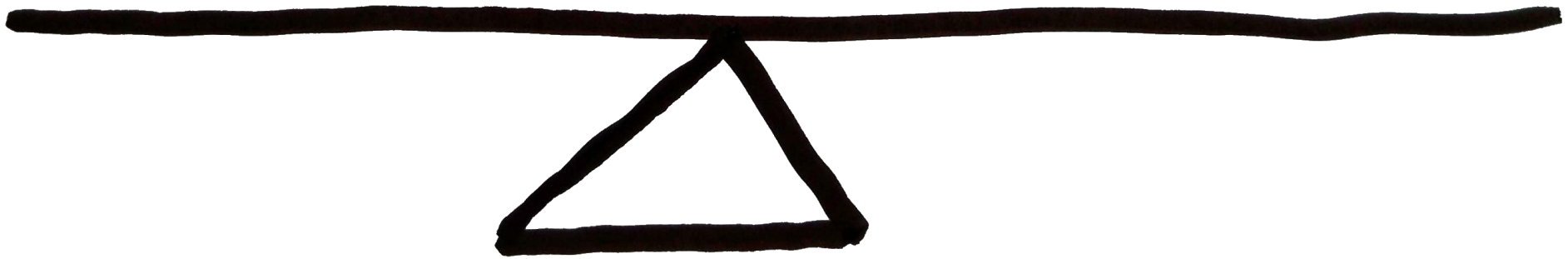


Wir kommunizieren fast gar nicht, Teammitglieder arbeiten isoliert

Peter Götz
Agile Coach, Trainer & Software Architect

Prozessdisziplin

Wir halten uns strikt an vorgeschriebene Prozesse und Regeln, selbst wenn sie nicht sinnvoll sind

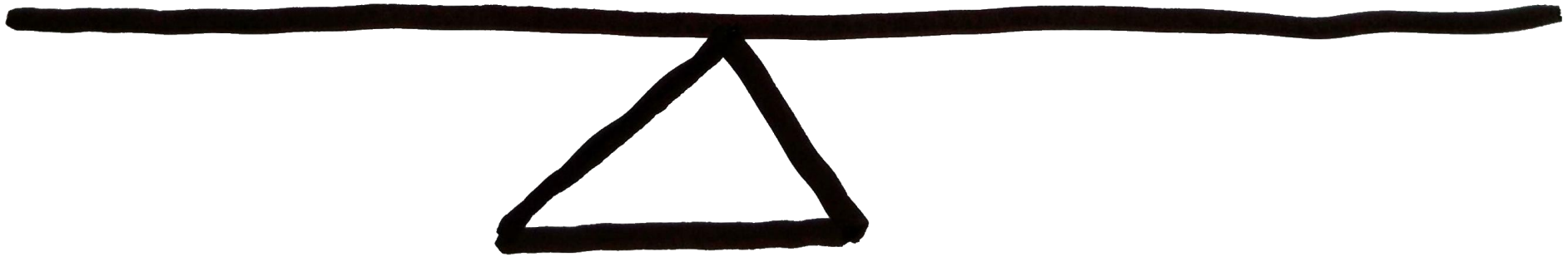


Wir haben keine festen Prozesse, jede Person entscheidet individuell, wie Aufgaben erledigt werden

Peter Götz
Agile Coach, Trainer & Software Architect

Selbstorganisation

Wir haben eine streng hierarchische Teamstruktur, in der alle Entscheidungen von oben kommen



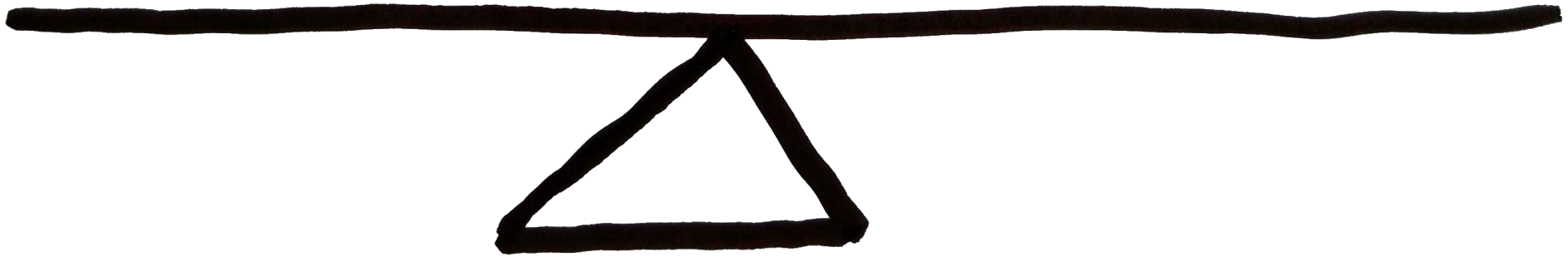
Wir sind vollkommen hierarchiefrei und selbstorganisiert

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Solo-Entwicklung vs. Collaboration

Wir arbeiten ausschließlich im Mob-Programming oder Pair-Programming, Änderungen dürfen nie von Einzelpersonen durchgeführt werden

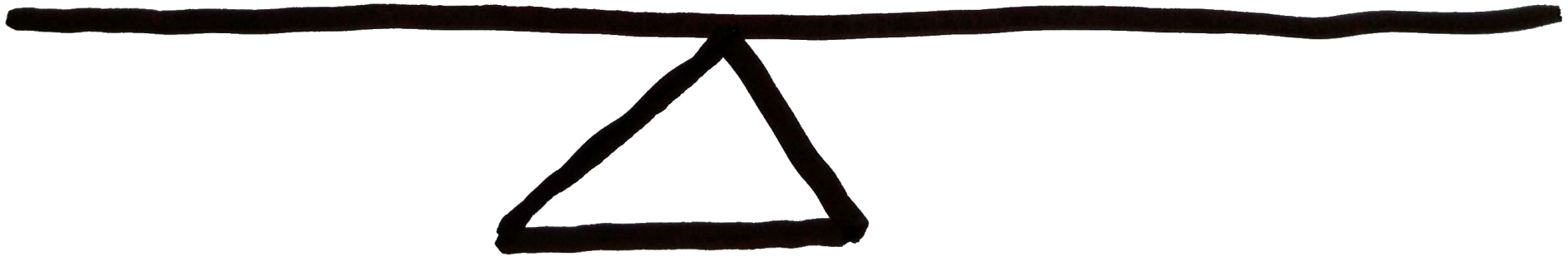


Wir haben möglichst abgegrenzte Aufgaben- und Themenbereiche für die verschiedenen Teammitglieder

Peter Götz
Agile Coach, Trainer & Software Architect

Führung

Führung bedeutet Kontrolle

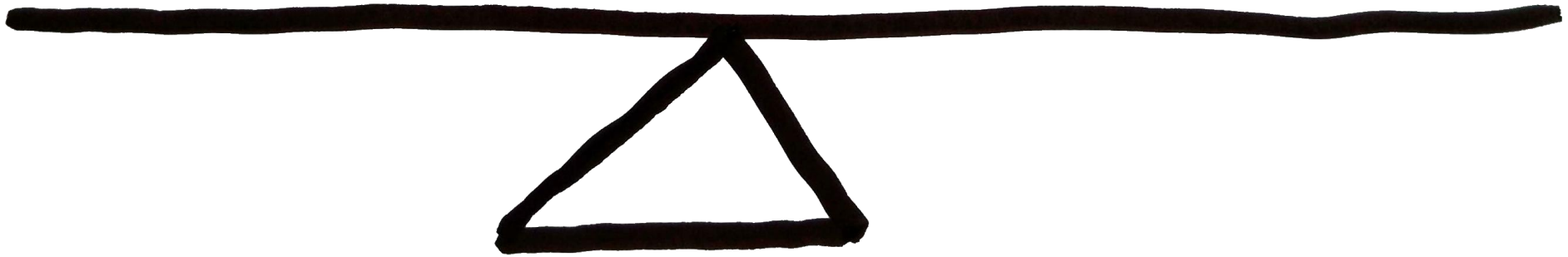


Führung bedeutet Orientierung

Peter Götz
Agile Coach, Trainer & Software Architect

Refinement

Anforderungen werden so lange diskutiert, bis jedes Detail geklärt ist

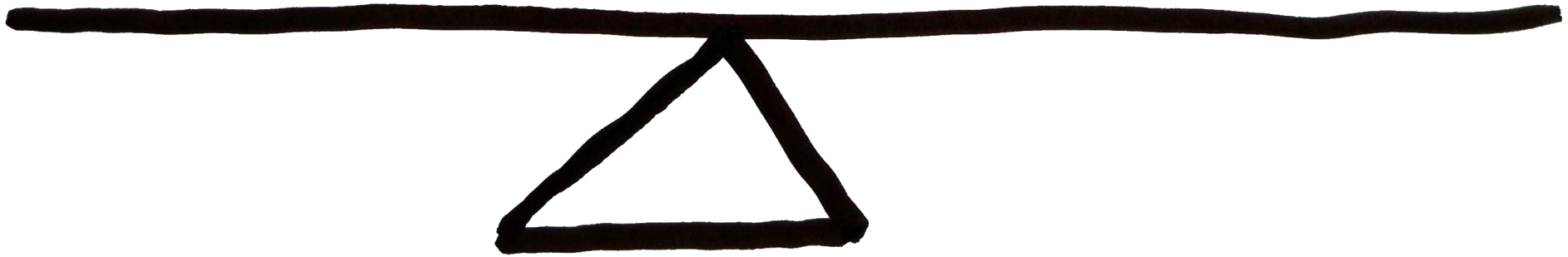


Wir starten mit der Implementierung und klären alle offenen Punkte dann während der Umsetzung

Peter Götz
Agile Coach, Trainer & Software Architect

Schätzung

Wir versuchen möglichst präzise zu schätzen, um damit zu exakten Prognosen zu kommen



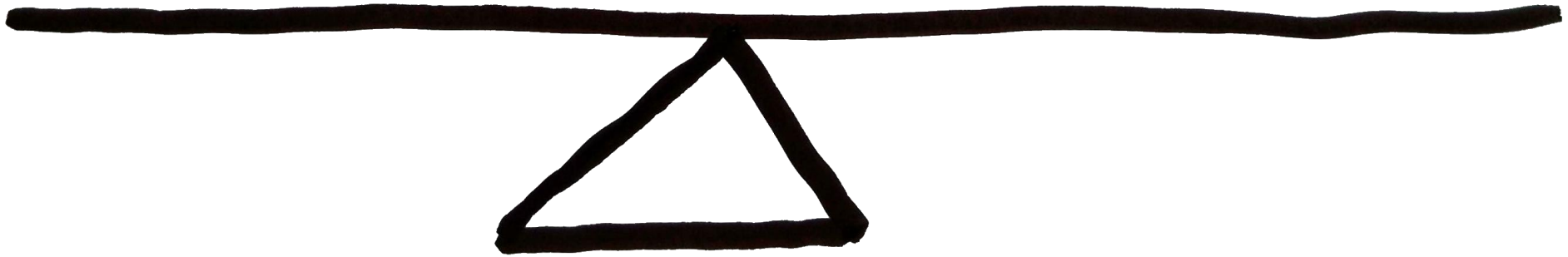
Wir führen gar keine Schätzung durch, Aufgaben dauern so lange, wie sie eben dauern und Schätzungen sind eh Lügen

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Outsourcing vs. Inhouse-Entwicklung

Wir vergeben nahezu alle Entwicklungsaufgaben an externe Dienstleister

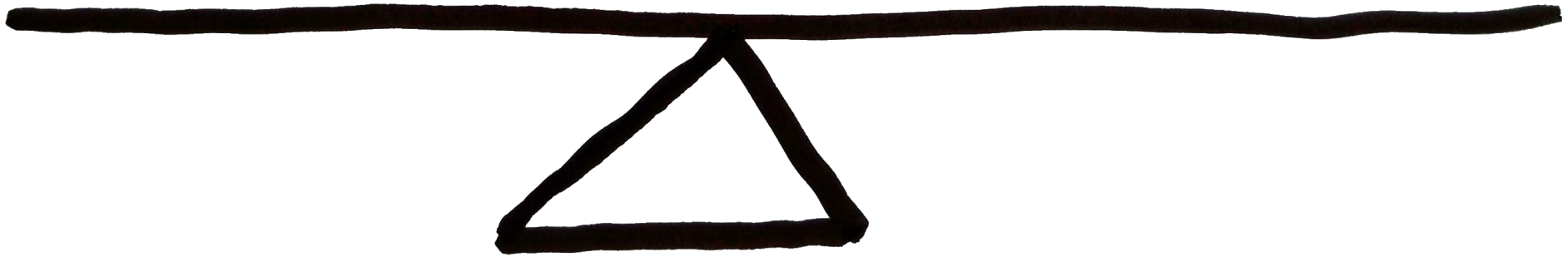


Wir entwickeln alles strikt im eigenen Haus

Peter Götz
Agile Coach, Trainer & Software Architect

Produkt-Ziele

Ziel ist es, die Aufgaben im Backlog abzuarbeiten



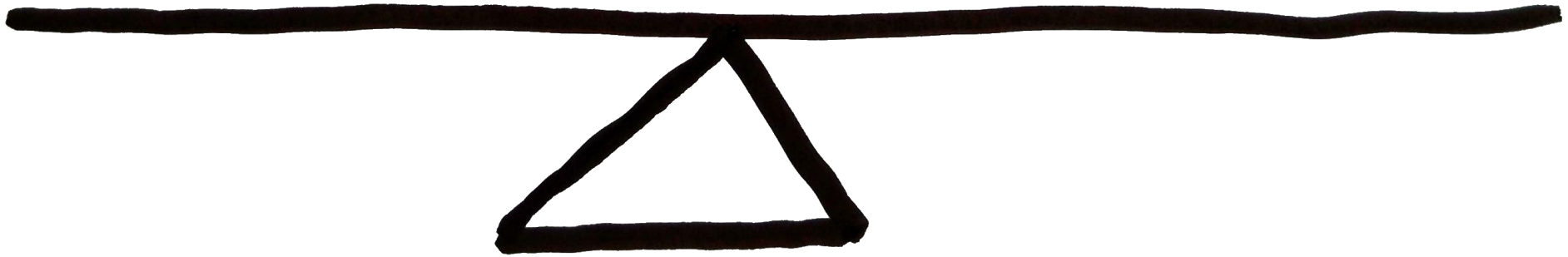
Wir haben eine grobe Produkt-Vision, daraus ergibt sich alles völlig dynamisch

agileMax
Maximize your Agility

Peter Götz
Agile Coach, Trainer & Software Architect

Künstliche Intelligenz

Wir nutzen konsequent künstliche Intelligenz weil uns das wesentlich produktiver macht



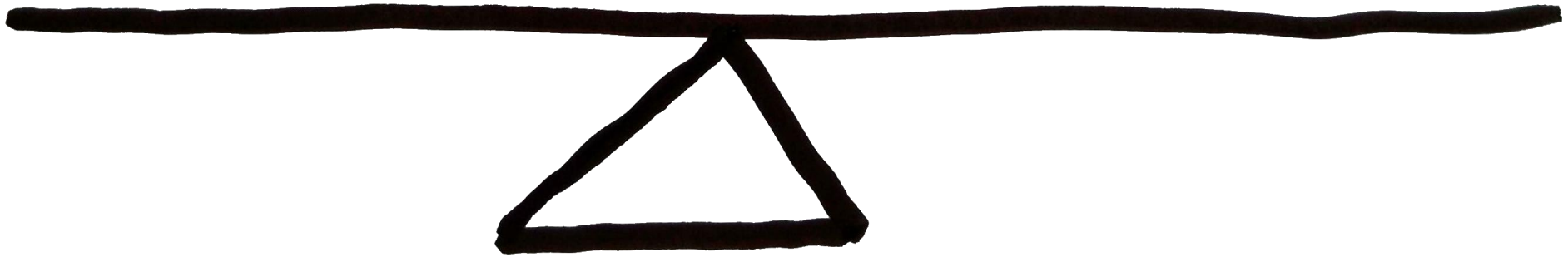
Bei uns ist der Einsatz von künstlicher Intelligenz verboten, weil wir sicherstellen wollen, dass die Entwickler:innen ihren Code beherrschen

agileMax

Peter Götz
Agile Coach, Trainer & Software Architect

Nachhaltige Entwicklung

Wir entwickeln "qucik & dirty" um schnell liefern zu können

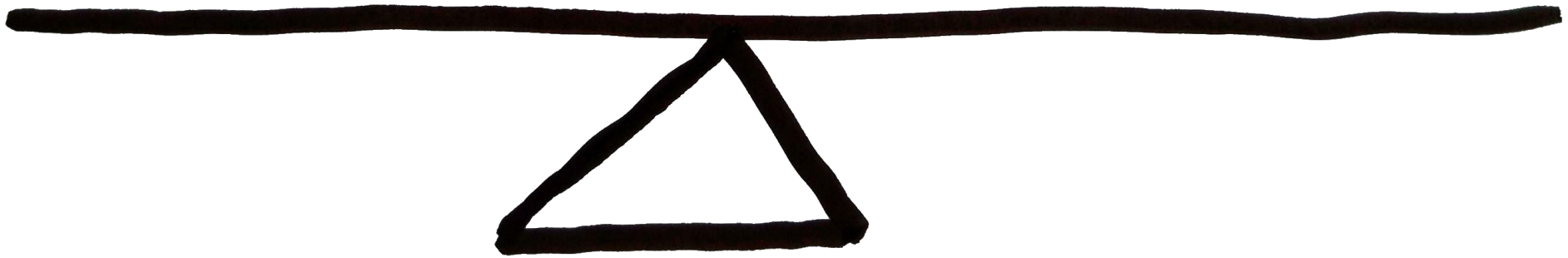


Unsere Lösung sind wohlüberlegt und sauber umgesetzt aber wir liefern extrem langsam

Peter Götz
Agile Coach, Trainer & Software Architect

Technisches Risiko

Risiken werden aufgeschoben, bis sie zu echten Problemen werden

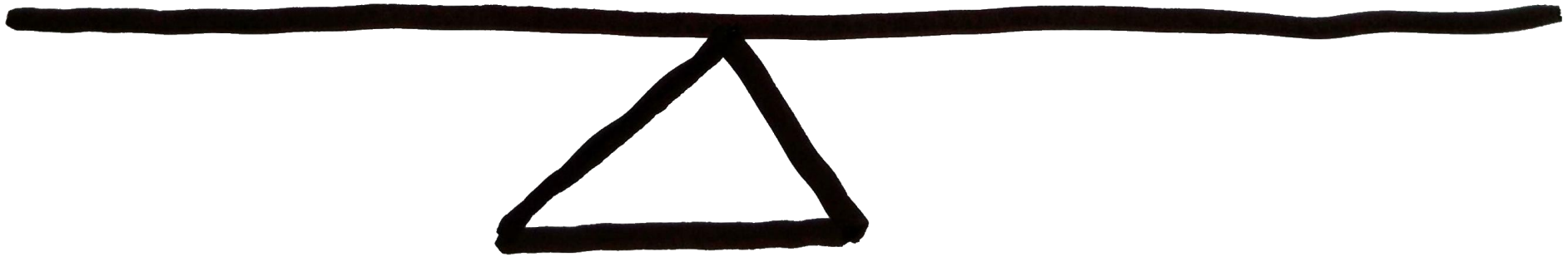


Im Ziel, Risiken frühzeitig zu berücksichtigen sind unsere Entscheidungsprozesse extrem langsam

Peter Götz
Agile Coach, Trainer & Software Architect

Technische Schulden

Wir gehen technische Schulden nur bewusst ein und achten immer auf eine zeitnahe Tilgung



Technische Schulden werden erst dann entdeckt, wenn Probleme entstehen, für eine Tilgung ist zu wenig Zeit