

A large iceberg floating in the ocean. The tip of the iceberg is visible above the water surface, while the much larger, jagged base is submerged below the surface. The sky is a mix of blue and orange, suggesting a sunset or sunrise. A semi-transparent white circle is overlaid on the right side of the image.

JUG SAXONY DAY 2026

Keine bösen Überraschungen:

Was wir vom Untergang der Titanic
für unsere Software lernen können

Uwe Vigenschow
Principal QA Specialist
Head of Consulting Services

NELTA GmbH, Hamburg

DIE NELTA FOKUSTHEMEN

3 Kernfelder der digitalen Welt



Digitalisierung nach Plan

Lassen Sie uns gemeinsam Prozesse optimieren, Geschäftsmodelle neu denken und Time-to-Market beschleunigen.

Qualität als Prinzip

Zuverlässig performante, stabile und sichere Software steigert Kundenzufriedenheit, Markenreputation und Geschäftserfolg.

KI mit Wirkung

Wir identifizieren relevante Anwendungsfälle und entwickeln pragmatische Automatisierungslösungen.

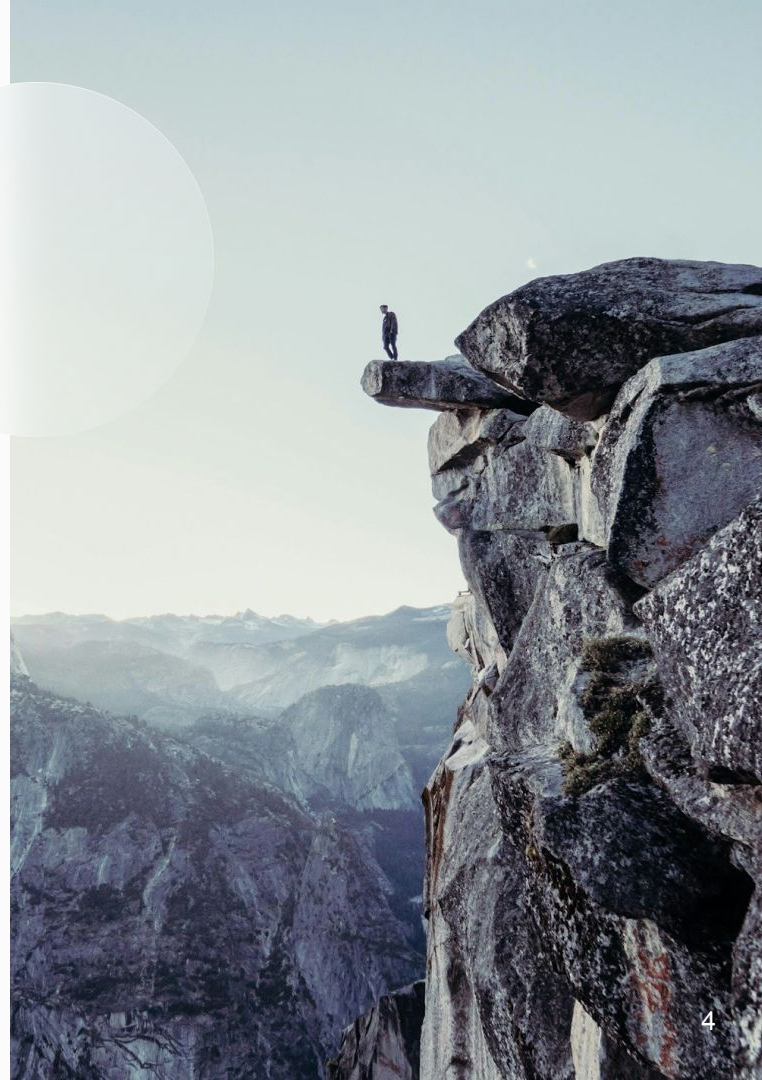
PROJEKTE SIND PER DEFINITIONEM RISKANT

Risikomanagement ist der Kern von Projektmanagement!



DAS TESTEN SELBST UND DIE STEUERUNG
DER TESTS ERFOLGEN RISIKOBASIIERT

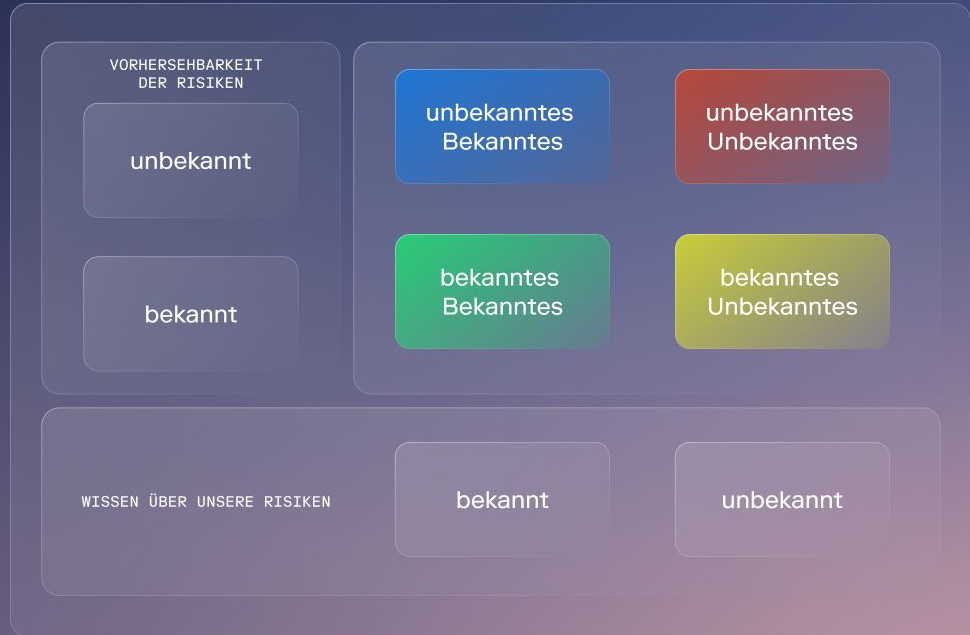
Bei Softwareprojekten ist Testen der Kern des Risikomanagements!



Wir testen nur, was wir kennen!

Rumsfeld-Matrix

→ Wir wissen nicht, was wir nicht wissen!



Emergentes Verhalten komplexer Software

Herausforderung

1

Komplexe, verteilte Multi-User-Systeme verhalten sich z.B. beim Laufzeitverhalten nicht mehr vorhersagbar.

Strategie

2

Das Risikomanagement durch Residuality Theory ergänzen!



Lösung

3

Wir testen, was vom System als Residuum noch übrig bleibt, nachdem das System durch eine Reihe von Stressereignissen massiv unter Druck ist.

Die Faktenlage

Am 14. April 1912 gesunken

Die Titanic rammte am 14. April 1912 um 23:40 Uhr seitlich einen Eisberg und sank nach 2 Stunden und 40 Minuten.

710 Personen überlebten

Nur 710 von ca. 2224 Personen überlebten.

Zu wenige Rettungsboote

Rettungsboote waren nur für 1178 Personen vorhanden, was damals legal war.

Die Ursachen



Spröbruch- Phänomen

Das Metall des Rumpfs wurde bei eisigen Wassertemperaturen extrem spröde.

Minderwertige Nieten

Hoher Schlackeanteil ließ die Eisennieten reißen, wodurch sich die Stahlplatten voneinander lösten.

Nach oben hin offene Schotten

Als der Bug sank, lief das Wasser oben über die Querwände in die nächste Kammer.

WAS WIR VOM UNTERGANG DER TITANIC LERNEN KÖNNEN

Was geholfen hätte:

Nicht ausweichen, sondern
frontal draufhalten!



Was hätte man vorher wissen können?



Wie viele und welche Schäden übersteht das Schiff und ist weiter schwimmfähig? (Residuum)

Welche Schäden beeinträchtigen die Rettungsmaßnahmen? → Stampfen, Rollen, Krängung

Simulationen und Crash-Tests bzw. heutzutage: Computersimulation

Vorgaben ableiten, wie sich im Ernstfall zu verhalten ist

Wichtige Fragestellungen fehlten.

WAS WIR VOM UNTERGANG DER TITANIC LERNEN KÖNNEN

Was bedeutet das für unsere Softwareentwicklung?



Das Ziel sind resiliente Systeme, keine perfekten Systeme, die kontrolliert an Leistung verlieren, anstatt komplett abzustürzen!



EIN SICHTWECHSEL IST NOTWENDIG

Herkömmliche Architektur

- Solution Design
- nach bekannten Anforderungen

Residuality Theory

- komplexe, unvorhersehbare Welt
- die meisten Katastrophen sind vorab unbekannt





RESIDUALITY THEORY (BARRY O'REILLY)

Architektur ist das, was noch steht,
wenn die Server brennen!

Die 4 Säulen der Theorie

Stressoren

Ereignisse, die das System massiv unter Druck setzen

Residuen

Wenn Stressoren auf das System treffen: Was bleibt stehen oder formiert sich neu?

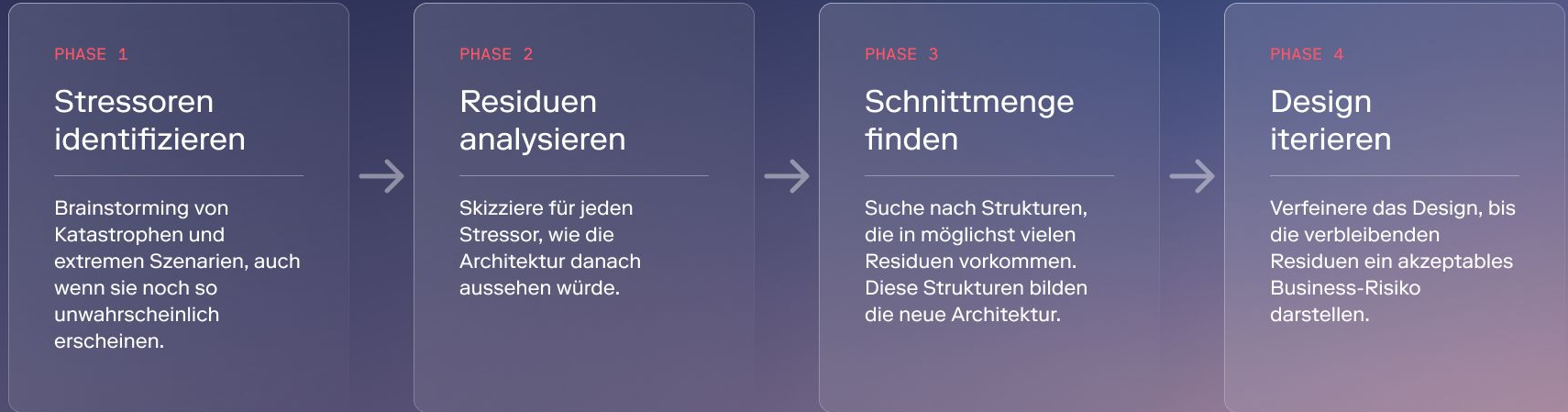
Kontamination

Komponenten beeinflussen sich: Wie können die Ausbreitungspfade durch das Design der Residuen gesteuert werden?

Naivität

Systeme sind naiv, wenn sie nur nach Anforderungen gebaut sind. Ziel: Die Naivität verringern!

Ergänzender Risikomanagement-Prozess



Architektur ist, was stehen bleibt

Simian Army: Chaos Monkey und Co

Simulieren Ausfälle von Servern,
Rechenzentren, AWS-Regionen
und künstliche Verzögerungen.

Naivität vermeiden

Ein System, das nie am Limit
getestet wurde, ist naiv stabil.
Chaos Engineering erzwingt den
Zustand nach dem Kollaps.

Lerneffekt

Was überlebt, ist das belastbare
Residuum. Was kollabiert, ist die
Schwachstelle und wird beseitigt.

Warum sollten wir so arbeiten?

Chaos Engineering als Versicherung für Hochverfügbarkeit.

System-Intuition Entwickler lernen, wie sich das System unter Last wirklich verhält.

Kostenreduktion Ein kontrollierter (Teil-)Ausfall ist billiger als ein unkontrollierbarer Totalausfall.

MTTR Die Teams üben den Ernstfall so oft, dass die Dauer für das Recovery extrem sinkt.

*MTTR = Mean Time To Recovery

Komponente zur Umrechnung von Währungen

PHASE 1

Stressoren identifizieren

Die API des Drittanbieters für Wechselkurse wird dauerhaft abgeschaltet.



PHASE 2

Residuen analysieren

Ein lokaler Cache mit den letzten bekannten Kursen und ein manueller Override- Mechanismus laufen weiter.



PHASE 3

Design iterieren

Dieser Cache ist kein Add-on, sondern wird zum zentralen Design-Element, weil er das Überleben sichert.

e-Commerce Microservice-Architektur

PHASE 1

Stressoren identifizieren

Order, Payment und Inventory Service:
Während einer Transaktion ist die Speicherung von Kundendaten für 48 h verboten. Gleichzeitig fällt der primärer Cloud-Provider für eine Woche aus.



PHASE 2

Residuen analysieren

Das System bricht komplett zusammen. Es gibt keine Residuen, sondern nur toten Code. Die Architektur ist fragil.



PHASE 3

Schnittmenge finden

Idee: zustandslose Architektur (stateless), die Daten im RAM hält, bis die Transaktion abgeschlossen ist

Idee: Multi-Cloud-Fähigkeit oder On-Premise-Fallback-Lösung



PHASE 4

Design iterieren

Die persistente Datenbank ist das größte Risiko.

Event-Driven-Modell einführen, bei dem Daten nur als flüchtige Events existieren.

Payment vom Inventory Service entkoppeln.

Strukturierte Resilienzprüfung

01 e-Commerce und Marktplatzanbieter

Abhängigkeiten zwischen Microservices: Funktioniert der Checkout, wenn das Empfehlungssystem ausfällt?

02 FinTech und Regionalbanken

Compliance und Disaster Recovery: Ausfälle der API von Drittanbietern simulieren. Reagieren die Apps mit Fehlermeldungen oder frieren sie ein?

03 SaaS im B2B-Bereich

Schwachstellen im Staging-System triggern, um die Reaktionszeit der On-Call-Ingenieure zu trainieren.

Was bedeutet das für den Mittelstand?

Fokus auf MTTR



Im Mittelstand fehlen oft die Budgets für redundante High-End-Infrastrukturen.

Die Fähigkeit, nach einem Ausfall blitzschnell und automatisiert wieder online zu sein, ist wertvoller als der Versuch, Fehler zu vermeiden.

Ziel: 10 x schnellere Recovery

Essenzielle Resilienz-Pattern

Circuit Breaker

Verhindert, dass ein fehlerhafter Dienst das gesamte System blockiert.

Bulkheads

Strikte Trennung von Ressourcen-Pools, damit ein lokaler Brand nicht das ganze Schiff versenkt.

Sidecars

Auslagern von Resilienzlogik (Retry, Timeout) in eine dedizierte Infrastrukturschicht

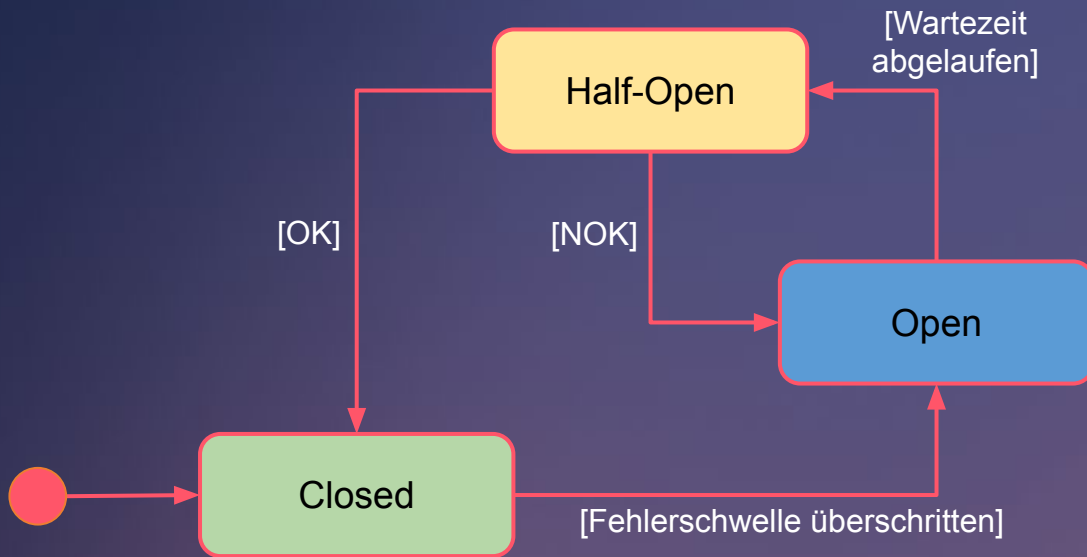
Graceful Degradation

Abschalten von Non-Core-Features bei Lastspitzen, um den Kern zu sichern

Keine Wirkung ohne Nebenwirkung

Die Resilienz-Pattern erhöhen die Komplexität und können dadurch selbst zur Risiko werden!

Circuit Breaker



CLOSED

Alles läuft normal, Anfragen werden durchgelassen.

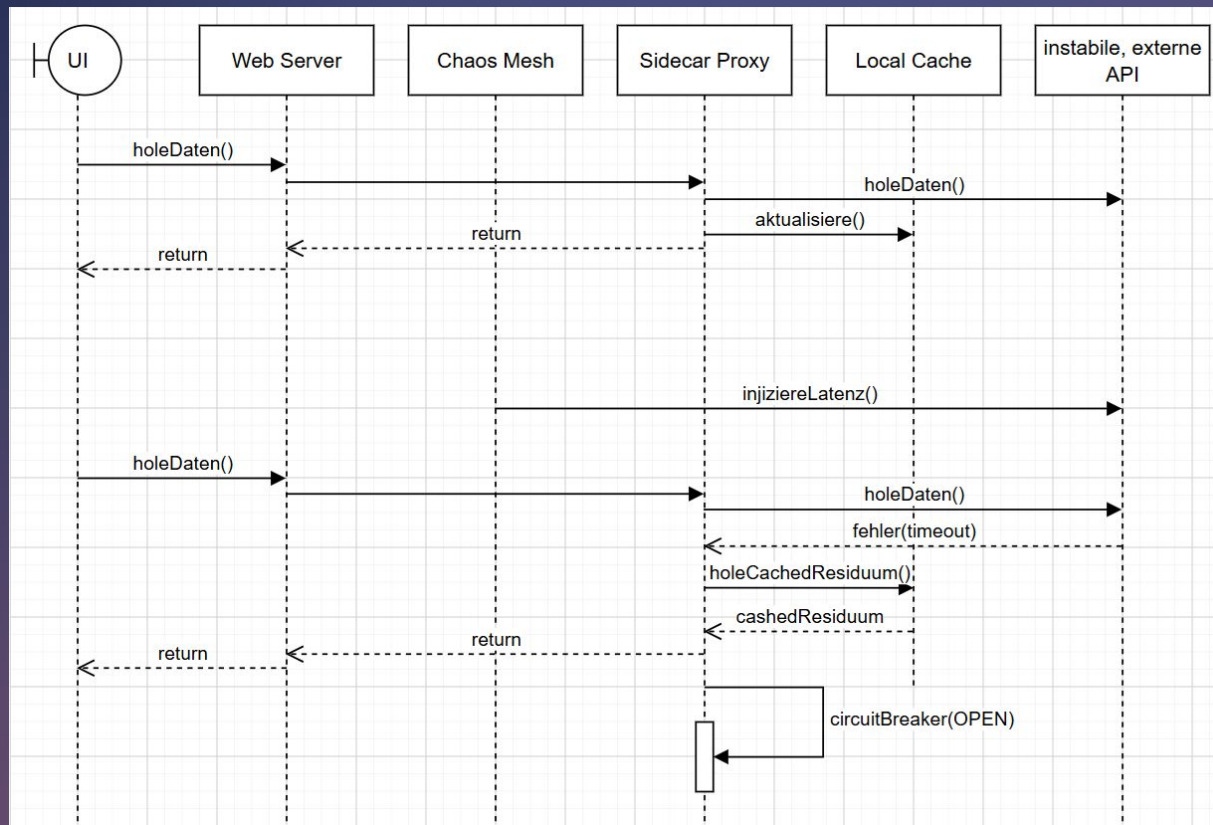
OPEN

Fehlerschwelle überschritten. Anfragen werden sofort mit Fehler beantwortet (Fail Fast), ohne das Backend zu belasten.

HALF-OPEN

Nach einer Wartezeit wird eine Testanfrage durchgelassen. Funktioniert sie ⇒ CLOSED
Funktioniert sie nicht ⇒ OPEN

Circuit Breaker im Sidecar mit lokalem Cache



Resiliente Systeme entwerfen

Warum kann ich ein resilientes System unter Verwendung der Resilienz-Pattern nicht vorab entwerfen?

- Unbeabsichtigte Seiteneffekte: Resilienz-Pattern sind Code.
- Pattern wie Bulkheads oder Timeouts basieren auf Annahmen und geschätzten Zahlenwerten.
- Verteilte Systeme verhalten sich unvorhersehbar (Emergenz).
- Wissen oder Vertrauen?
„Ich glaube, es funktioniert“ ⇨ „Ich weiß, es funktioniert“.

Wie sieht es mit ihren Softwarequalitätsprozessen aus?

Geht es ihnen wie auf der Titanic?

Lassen Sie uns gemeinsam einen Blick darauf werfen!

