DevOps mit Conjugate Intelligence (CI)

Zusammenfassung

DevOps ist heute stark in der Automatisierung – verliert jedoch oft den Zusammenhang zwischen den Werkzeugen.
Conjugate Intelligence (CI) stellt diesen Zusammenhang wieder her und macht Kontext zu einem echten Wert.
Pipelines, Änderungen und Vorfälle werden zu *Perlen* in einem **EKR-Graphen**[1] – verbunden mit Absicht, Annahmen und Nachweisen.
Das Ergebnis: sicherere, schnellere Auslieferung mit weniger Aufwand.

Aktuelle Probleme

- **Instabile Pipelines:** Verhalten ändert sich je nach Branch oder Umgebung – ohne klaren Grund.
- **Kontextverlust in Übergaben:** Entscheidungen zwischen Product → Dev → Ops zerstreuen sich über Chats und Tickets.
- **Dashboards zeigen das *Was*, nicht das *Warum*.**
- **Wiederkehrende Mühen:** Diagnose- und Release-Rituale, die man einmal lernen und wiederverwenden könnte.

CI-Ansatz (SpiralOS)

- **EKR-Graph als Quelle der Wahrheit:**
  Commits, Tests, Releases, Vorfälle, Runbooks und Freigaben sind als semantische Knoten mit Versionsgeschichte verbunden.

- **Vows – Absichtsbindingen:**
  Jede Änderung oder Veröffentlichung trägt explizite Ziele, Risikohaltung und Leitplanken, die mit dem Artefakt reisen.[2]

- **Conjugation:**
  Der beabsichtigte Systemzustand (*Self*) wird fortlaufend mit der realen Telemetrie (*Other*) verglichen – um sinnvolle Aktionen auszulösen.[3]

Zentrale Fähigkeiten

- **Kontextbewahrende Pipelines:**
  Passen sich an Branch-Risiken, Fehlerbudgets und Geschäftskritikalität an – mit dokumentiertem „Warum“.
- **Adaptive Leitplanken:**
  Vor-/Nachbedingungen, sichere Rollout-Rezepte und Rückfallverträge sind als *Perlen* an Releases angeheftet.
- **Change-Risiko-Bewertung:**
  CI bewertet Änderungen anhand von Code-Diffs, Testabdeckung, Auswirkungsradius und jüngsten Vorfällen.
- **ChatOps mit Gedächtnis:**
  Konversationsbefehle arbeiten mit vollem Kontext und erzeugen prüfbare, erzählte Abläufe.
- **Holarchische Rückschau:**
  Jede Änderung oder jeder Vorfall lässt sich nachspielen – von Eingaben über Entscheidungen bis zu Ergebnissen.

Ergebnisse & Kennzahlen (KPIs)

- Durchlaufzeit für Änderungen ↓ 20–40 %
- Änderungs-Fehlerquote ↓ 25–50 %
- MTTR (Mean Time to Recovery) ↓ 20–35 %
- Aufwandstunden ↓ 30–60 %
- Bereitstellungshäufigkeit ↑ – mit Vertrauen dank Leitplanken

---

Integrationspfad (Reibungsarm)

1. **Beobachtungsmodus (Woche 1–2):**
   Einlesen von Git-Repos, CI/CD-Logs (z. B. Actions, Jenkins), Artefakt-Metadaten und Incident-Tickets – ohne Schreibzugriff.
2. **Perlen-Vorlagen (Woche 2–3):**
   Templates für Änderung, Release und Incident mit minimalen Feldern: Absicht, Risiko, Leitplanken, Nachweise.

3. **Geführte Aktionen (Woche 3–5):**
   CI schlägt Rollout-Pläne, Test-Gates und sichere Rückschritte vor – Mensch bleibt im Regelkreis.

4. **Gezielte Automatisierungen (ab Woche 5):**
   Genehmigte Auto-Rollouts/Rollbacks unter definierten Schwellen; alle Aktionen werden im EKR-Graph erzählt.

Risiken & Gegenmaßnahmen

- **Über-Automatisierung:**
  Schwellenbasiert, optional, mit expliziten Zustimmungen (*Vows*).
- **Vendor-Lock-In:**
  Offenes Graph-Modell, Import/Export-APIs, menschenlesbare Narrative.
- **Modell-Drift:**
  Ständige Bewertung über Post-Deploy-Ergebnisse und Incident-Feedback.

Beispiel – Hotfix

Ein kritischer Fix an *Service X* betrifft ein Hochrisiko-Modul.
CI bewertet die Änderung, schlägt einen Canary-Rollout mit Shadow-Traffic vor, definiert SLO-basierte Abbruchkriterien
und erstellt einen One-Click-Rollback-Vertrag.
Während des Rollouts steigt die Latenz – CI führt den Rollback aus, dokumentiert alle Belege im Graph
und erzeugt einen narrativen Post-Release-Bericht mit Folgeaktionen.

Branchenbeispiele

- **Fertigung:** Adaptive Pipelines stellen sicher, dass MES-Updates laufen, ohne Produktionslinien zu stoppen.
- **Mobilität:** OTA-Updates für Fahrzeugflotten werden als Canary-Tests unter Sicherheits-Leitplanken ausgeführt.

- **Energie:** Grid-Steuerungssoftware-Patches tragen explizite Risiko-Vows, verknüpft mit SLOs zu Verfügbarkeit und Sicherheit.

Gesprächsimpulse (für Erich & Echo)

- „CI ersetzt deine Pipelines nicht — es verbindet sie mit Absicht und Evidenz."
- „Deine Dashboards werden zu Geschichten, die du nachspielen kannst — nicht zu Screenshots, die du vergisst."
- „Risiko wird explizit — und reist mit dem Artefakt."

🧭 Begriffserklärungen

[1] EKR (Epistemic Knowledge Resonance Graph)
Ein lebendiges Wissensnetz, das Commits, Tests, Releases, Vorfälle und Entscheidungen als semantische Knoten verbindet.
Jede *Perle* enthält Absicht, Annahmen und Belege. Der EKR bewahrt Kontext und macht Lernen im System sichtbar.

[2] Vows (Absichtsbindingen / Wirkversprechen)
Bewusste Festlegungen von Zielen, Haltungen und Grenzen.
Ein Vow begleitet jede Änderung oder Veröffentlichung – als ethisch-resonantes Versprechen im Handeln.

[3] Conjugation (Wechselspiel / Resonanzbezug)
Kontinuierlicher Abgleich zwischen beabsichtigtem Zustand (*Self*) und tatsächlichem Zustand (*Other*).
Das System reagiert auf Abweichungen in Resonanz – Grundlage adaptiver, bewusster Steuerung.