SituationReport

testdata_generator

Benutzerhandbuch

Synthetische Jira-Issue-Daten für SituationReport erzeugen

---

situation-report -- github.com/Jaegerfeld/situation-report

Fuer Agile Coaches und PI Manager -- Version 0.19.0

1 Was ist der Testdata Generator?

Der Testdata Generator erzeugt synthetische Jira-Issue-Daten im Jira-REST-API-Format. Die generierten Dateien sind direkt mit dem Modul transform_data verarbeitbar — ohne eine echte Jira-Instanz.

Der Generator simuliert realistische Workflow-Historien: Issues durchlaufen die definierten Stages, verbleiben unterschiedlich lange in jeder Stage, werden gelegentlich zurückgestuft und schließen mit konfigurierbarer Rate ab.

> Typische Einsatzgebiete:
> - Neue Installation testen: Prüfen, ob transform_data und build_reports korrekt arbeiten
> - Demos und Präsentationen: Realistische Daten ohne Datenschutzbedenken
> - Schulungen: Lernumgebung mit bekannten, kontrollierten Daten
> - Entwicklung: Reproduzierbare Testdaten über den Seed-Parameter

Der Testdata Generator ergänzt das Modul Get Data (noch in Entwicklung), das echte Daten direkt aus Jira lädt. Für den manuellen Export echter Jira-Daten steht das Benutzerhandbuch Get Data zur Verfügung.

2 Voraussetzungen und Start

2.1 Portables Paket (empfohlen)

Das portable Paket enthält Python bereits eingebettet — keine separate Installation erforderlich.

Betriebssystem	Startdatei
Windows	TestdataGenerator.bat (Doppelklick)
macOS	TestdataGenerator.command (Rechtsklick → Öffnen)
Linux	./TestdataGenerator.sh

Hinweis für Windows: Falls beim Start eine Fehlermeldung erscheint, die Python nicht findet — die ZIP-Datei war möglicherweise durch Windows blockiert. Lösung: Rechtsklick auf die ZIP → Eigenschaften → Sicherheit → Zulassen → OK → erneut entpacken.

2.2 Aus dem Quellcode (Python erforderlich)

```
python -m testdata_generator
```

2.3 Die Benutzeroberfläche

Nach dem Start öffnet sich das Hauptfenster mit Eingabefeldern für alle Parameter. Pflichtfeld ist nur die Workflow-Datei — alle anderen Felder haben sinnvolle Standardwerte.

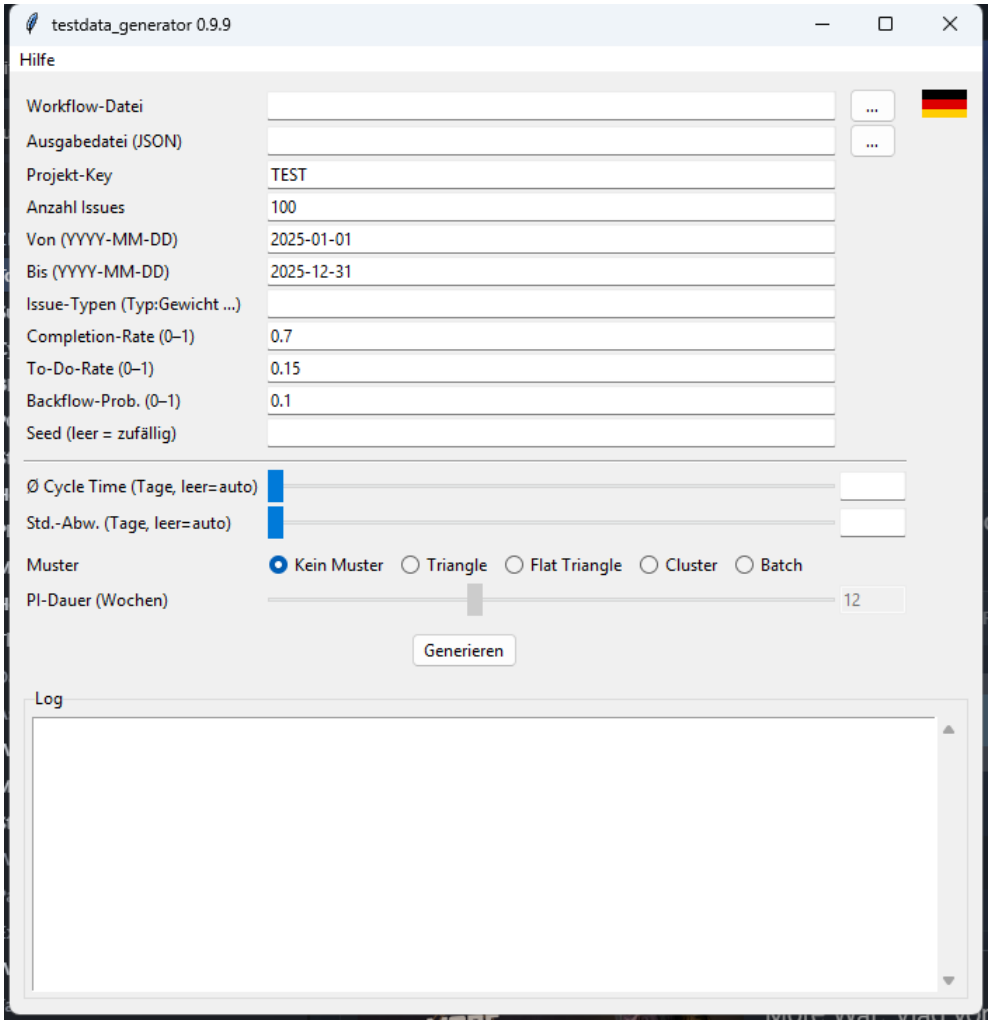

Abb. 1: Benutzeroberfläche des Testdata Generators

3 Die Workflow-Datei

Die Workflow-Datei definiert die Stages Ihres Jira-Boards und ist das einzige Pflichtfeld des Generators. Sie hat dasselbe Format wie in transform_data — eine Datei genügt für beide Module.

3.1 Format

Jede Zeile beschreibt eine Stage. Aliases (alternative Status-Namen im Jira-Export) werden durch Doppelpunkte abgetrennt. Zwei Sonderzeilen markieren Anfang und Ende der aktiven Bearbeitung:

```
KanonischerName:Alias1:Alias2
<First>ErsteAktiveStage
<Closed>AbgeschlosseneStage
```

Zeile	Bedeutung
KanonischerName:Alias1:Alias2	Stage mit einem oder mehreren alternativen Jira-Status-Namen
StageName (ohne :)	Stage ohne Aliases
StageName	Erste aktive Stage — wo Bearbeitung beginnt (Cycle-Time-Start)
StageName	Abgeschlossene Stage — markiert ein Issue als fertig

3.2 ART_A Beispiel-Workflow

Das mitgelieferte Beispiel `workflow_ART_A.txt` bildet einen typischen SAFe-ART-Workflow ab:

```
Funnel:New:Open:To Do
Analysis:In Analysis:Estimated
Program Backlog:Ready for Dev:Backlog
Implementation:In Implementation:In Review:In Progress
Blocker
Validating on Staging:QA
Deploying to Production:Ready for Development
Releasing:Completed
Done:Canceled
<First>Analysis
<Closed>Releasing
```

Anmerkungen: 'Funnel' enthält vier Aliases — alle vier Jira-Status-Namen werden auf 'Funnel' normiert. Analysis ist die erste aktive Stage (Cycle-Time beginnt hier). Releasing gilt als abgeschlossen. Done (nach Releasing) nimmt auch Canceled-Issues auf.

4 Parameter-Referenz

Alle Parameter sind sowohl in der GUI als auch auf der Kommandozeile verfügbar.

Parameter	Standard	Beschreibung
--workflow FILE	(Pflicht)	Workflow-Definitionsdatei (.txt)
--output FILE.json	_generated.json	Pfad der Ausgabedatei
--project KEY	TEST	Jira-Projekt-Key der generierten Issues
--issues N	100	Anzahl zu generierender Issues
--from-date YYYY-MM-DD	2025-01-01	Frühestes Erstellungsdatum
--to-date YYYY-MM-DD	2025-12-31	Spätestes Übergangsdatum
--issue-types TYPE:W ...	Feature:0.6 Bug:0.3 Enabler:0.1	Issue-Typen mit Gewichtung (Summe muss nicht 1 ergeben)
--completion-rate FLOAT	0.7	Anteil Issues, die die Closed-Stage erreichen (0–1)
--todo-rate FLOAT	0.15	Anteil offener Issues, die in frühen Stages verbleiben (0–1)
--backflow-prob FLOAT	0.1	Wahrscheinlichkeit eines Rückschritts bei jedem Übergang (0–1)
--seed INT	(zufällig)	Seed für reproduzierbare Ausgabe (optional)
--mean-cycle-days FLOAT	(keiner)	Mittlere Cycle-Time in Tagen (lognormal); aktiviert CT-Steuerung
--std-cycle-days FLOAT	(30 % des Mittels)	Standardabweichung der Cycle-Time in Tagen
--pattern MUSTER	none	Flow-Antipattern: none / triangle / flat_triangle / cluster / batch
--pi-duration-weeks INT	12	PI-Zyklus-Länge in Wochen (für cluster/batch)
--pattern-strength 0-100	50	Muster-Intensität 0–100 (0=subtil, 50=Standard, 100=stark)

Tipp: Reproduzierbare Ergebnisse
Mit einem festen Seed (z. B. `--seed 42`) erzeugt der Generator bei gleichen Parametern immer dieselbe JSON-Datei — ideal für Demos, bei denen alle Teilnehmer dieselben Ergebnisse sehen sollen.

4b Flow-Antipattern-Muster

Ab Version 0.9.9 kann der Testdata Generator typische Flow-Antipatterns aus der Literatur von Vacanti und Singh simulieren. Dazu wird --mean-cycle-days mit --pattern kombiniert. In der GUI stehen Radiobuttons und Schieberegler zur Verfügung.

Muster	Scatterplot-Erscheinung	Beschreibung
none	Gleichmäßige Punktwolke	Zufällige Cycle-Time ohne Trend — Standard-Verhalten
triangle	Dreieck: rechter Winkel unten rechts	Cycle-Time steigt linear über die Zeit (Multiplikator ~1–4× bei Standard-Stärke, höher mit --pattern-strength). Zeigt einen Prozess, der zunehmend langsamer wird.
flat_triangle	Dreieck, Anstieg flacht am Ende ab	Wie triangle, aber der Anstieg verflacht durch tanh(2.5×t). Zeigt eine Prozessverschlechterung, die sich stabilisiert.
cluster	Senkrechte Punkthäufungen am PI-Ende	Cycle-Time bleibt stabil (lognormal), aber die meisten Issues werden in den letzten 2 Wochen vor PI-Ende abgeschlossen. Deadline-getriebenes Verhalten.
batch	Säulen variabler Höhe am PI-Ende	Wie cluster, aber mit stark unterschiedlicher Cycle-Time (0.1× bis 3× Mittelwert). Zeigt, dass kurze und sehr lange Aufgaben gemeinsam ans PI-Ende geschoben werden.

Cycle-Time-Steuerung

Wenn --mean-cycle-days angegeben ist, wird die Cycle-Time lognormalverteilt mit dem angegebenen Mittelwert erzeugt.

Parameter	Bedeutung
--mean-cycle-days	Mittelwert der Cycle-Time in Tagen (z. B. 20). Pflicht für alle Muster außer none.
--std-cycle-days	Standardabweichung in Tagen (Standard: 30 % des Mittelwerts). Höhere Werte erzeugen breitere Streuung im Scatterplot.
--pi-duration-weeks	Länge eines PI-Zyklus in Wochen (Standard 12). Nur relevant für cluster und batch.

Beispiele

```
# Triangle-Muster: CT steigt von 20d auf 80d
python -m testdata_generator \
    --workflow workflow.txt --issues 300 \
    --pattern triangle \
    --mean-cycle-days 20 --std-cycle-days 6 \
    --completion-rate 0.8 --seed 1 \
    --output triangle.json

# Cluster of Dots: Lieferungen am PI-Ende (12 Wochen)
python -m testdata_generator \
    --workflow workflow.txt --issues 300 \
    --pattern cluster \
    --mean-cycle-days 30 --pi-duration-weeks 12 \
    --completion-rate 0.8 --seed 2 \
    --output cluster.json

# Batch Transfers: PI-Ende + hohe CT-Varianz
python -m testdata_generator \
    --workflow workflow.txt --issues 300 \
    --pattern batch \
    --mean-cycle-days 30 --pi-duration-weeks 12 \
    --completion-rate 0.8 --seed 3 \
    --output batch.json
```

Hinweis: --pattern erfordert --mean-cycle-days. Ohne --mean-cycle-days wird das Muster ignoriert.

5 ART_A – Vollständiges Beispiel

Dieses Beispiel zeigt den vollständigen Ablauf vom Generator bis zum fertigen Report am fiktiven Projekt ART_A.

Schritt 1: Testdaten generieren

```
python -m testdata_generator \
    --workflow testdata_generator/workflow_ART_A.txt \
    --project ART_A \
    --issues 200 \
    --from-date 2025-01-01 \
    --to-date 2025-12-31 \
    --issue-types Feature:0.6 Bug:0.2 Enabler:0.2 \
    --completion-rate 0.75 \
    --backflow-prob 0.08 \
    --seed 42 \
    --output ART_A_generated.json
```

Ergebnis: `ART_A_generated.json` mit 200 Issues, ca. 150 abgeschlossen. Dank Seed 42 jederzeit reproduzierbar.

Schritt 2: Mit transform_data verarbeiten

```
python -m transform_data ART_A_generated.json \
    --workflow testdata_generator/workflow_ART_A.txt
```

Schritt 3: Report mit build_reports erstellen

```
python -m build_reports
```

> Was zu erwarten ist (Seed 42, 200 Issues):
> - Issue-Keys: ART_A-0001 bis ART_A-0200
> - Issue-Typen: ~120 Feature, ~40 Bug, ~40 Enabler
> - Ca. 150 Issues im Status 'Releasing' oder 'Done'
> - Erstellungsdaten verteilt über das Jahr 2025
> - Gelegentliche Rückschritte (backflow-prob 0.08 = ca. 8 %)

5b Das Portfolio-Szenario

Mit `--scenario portfolio` erzeugt der Generator in einem Schritt ein komplettes Demo-Portfolio: zwei Solutions mit je drei ARTs, inklusive aller Artefakte der Verarbeitungskette.

In der GUI: Der Bereich „Demo-Portfolio" unten im Fenster erzeugt das Szenario per Knopfdruck in einen Ordner deiner Wahl (das Seed-Feld wird übernommen, Standard 42); „Portfolio-Report öffnen" rendert den Portfolio-Report und öffnet ihn im Browser — ganz ohne Kommandozeile.

```
python -m testdata_generator --scenario portfolio \
    --output demo/ --seed 42
```

Der Zielordner enthält danach:

Artefakt	Inhalt
workflows/	zwei Workflow-Dateien (Alpha- und Beta-Stages)
raw/	sechs Jira-JSON-Exporte (ein Generatorlauf je ART)
arts/	IssueTimes-, CFD- und Transitions-Dateien je ART
solution_alpha.json	Solution-Config ohne stage_map (Default-Pfad)
solution_beta.json	Solution-Config mit eigener stage_map (Schema 2)
risks_*.json	ROAM-Risiko-Register je Solution
nfr_*.json	NFR-/Runway-Register je Solution
capabilities_*.json	Capability-Map je Solution
dependencies_*.json	Dependency-Register je Solution
decisions_*.json	Decision-/Assumption-Log je Solution
portfolio.json	Portfolio-Config über beide Solutions
pi_config.json	PI-Intervalle über den Datenzeitraum
README.md	Beschreibung der eingebauten Geschichten

Die Daten liegen relativ zum Erzeugungsdatum, damit die Qualitäts-Ampel des Portfolio-Reports sie als aktuell einstuft. Drei Geschichten sind fest eingebaut:

Eingebaute Geschichten:
- ART Alpha-3 ist der Ausreißer (ca. dreifache Cycle Time) — im Comparison-Report der Solution Alpha rot hervorgehoben.
- ART Beta-3 liefert schwache Daten (kein CFD, kaum begonnene Issues, Datenstand 60 Tage alt) — Konfidenz 'low' in der Qualitätstabelle.
- Solution Beta poolt über eine eigene stage_map (Vorlauf/Umsetzung/Fertig); Solution Alpha nutzt den Default-Pfad.
- ROAM-Board: zwei Owned-Risiken sind bewusst alt (45/50 Tage) — die Aging-Hervorhebung springt an.
- NFR & Runway: Betas API-NFR ist verletzt, ein Runway-Element eine überfällige Lücke — das Dashboard zeigt Rot.
- Capability-Map: Betas Data-Insights-Capability ist kritisch, eine Alpha-Capability ohne ART (uncovered).
- Dependency-Heatmap: Alpha-1 → Alpha-3 blockiert und überfällig; Beta-1 → Alpha-1 als Cross-Solution-Integration.
- Decision-Log: Betas offene Annahme hat ihr Prüfdatum überschritten — rot als „review due" markiert.

Der Ordner ist direkt verwendbar: `python -m portfolio demo/portfolio.json` erzeugt den Portfolio-Report; die Solution-Configs funktionieren auch einzeln. Gleicher Seed ergibt am selben Tag identische Daten.

6 Häufige Fragen (FAQ)

Unterschied zwischen Testdaten und echten Daten?

Für Demos, Schulungen und Installationstests sind Testdaten die bessere Wahl. Für echte Flow-Analysen und Entscheidungen spiegeln echte Daten den tatsächlichen Workflow-Zustand wider.

Für den manuellen Export echter Jira-Daten: Siehe Benutzerhandbuch Get Data.

7 Glossar

Begriff	Erklärung
ART	Agile Release Train — ein Team aus mehreren Scrum-Teams in SAFe
API	Application Programming Interface — Programmierschnittstelle
CFD	Cumulative Flow Diagram — zeigt den Issuebestand je Stage kumuliert
Cycle Time	Zeit von der ersten aktiven Stage bis zur Closed-Stage
Helper	SituationReport-Modul zum Zusammenführen mehrerer JSON-Dateien
Issue	Arbeitselement in Jira (Feature, Bug, Enabler, Story, …)
JSON	JavaScript Object Notation — Format der Jira-API-Exporte
Seed	Startwert für den Zufallsgenerator — gleicher Seed = gleiche Ausgabe
Stage	Ein Schritt im Workflow (entspricht einer Jira-Spalte)
Workflow	Geordnete Abfolge von Stages, die ein Issue durchläuft