SituationReport

testdata_generator

Manual do Utilizador

Gerar dados sintéticos de issues Jira para SituationReport

---

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

Para Agile Coaches e Gestores de PI -- Version 0.19.0

1 O que é o Gerador de Dados de Teste?

O Gerador de Dados de Teste (Testdata Generator) cria dados sintéticos de issues Jira no formato Jira REST API. Os ficheiros gerados podem ser processados diretamente pelo módulo transform_data — sem necessidade de uma instância Jira real.

O gerador simula históricos de fluxo de trabalho realistas: as issues percorrem as etapas definidas, permanecem períodos variáveis em cada etapa, ocasionalmente regridem e fecham a uma taxa configurável.

> Casos de uso típicos:
> - Testar uma nova instalação: Verificar se transform_data e build_reports funcionam corretamente
> - Demos e apresentações: Dados realistas sem preocupações de privacidade
> - Formação: Ambiente de aprendizagem controlado com dados conhecidos
> - Desenvolvimento: Dados de teste reprodutíveis via parâmetro seed

O Gerador complementa o módulo Get Data (em desenvolvimento), que carrega dados reais diretamente do Jira. Para exportar dados reais do Jira manualmente, consulte o Manual do Utilizador Get Data.

2 Pré-requisitos e início

2.1 Pacote portátil (recomendado)

O pacote portátil inclui o Python integrado — não é necessária instalação separada.

Sistema operativo	Ficheiro de início
Windows	TestdataGenerator.bat (duplo clique)
macOS	TestdataGenerator.command (clique direito → Abrir)
Linux	./TestdataGenerator.sh

Nota para Windows: Se aparecer um erro sobre Python não encontrado, o ficheiro ZIP foi bloqueado pelo Windows. Solução: clique direito no ZIP → Propriedades → Segurança → Desbloquear → OK → extrair novamente.

2.2 A partir do código fonte (requer Python)

```
python -m testdata_generator
```

2.3 A interface do utilizador

Após o arranque, a janela principal apresenta campos de entrada para todos os parâmetros. Apenas o ficheiro de fluxo de trabalho é obrigatório — todos os outros campos têm valores predefinidos sensatos.

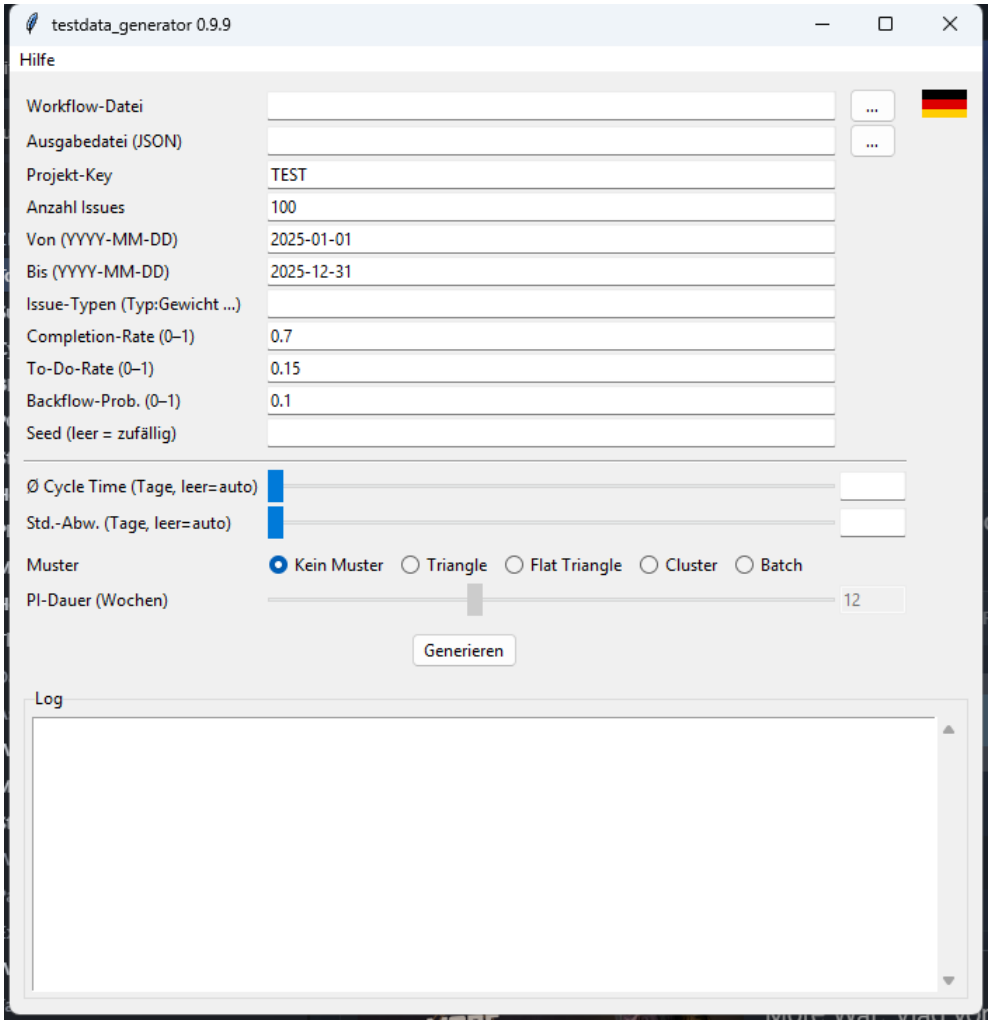

Fig. 1: Interface do utilizador do Gerador de Dados de Teste

3 O ficheiro de fluxo de trabalho

O ficheiro de fluxo de trabalho define as etapas do quadro Jira e é o único parâmetro obrigatório. Usa o mesmo formato que o transform_data — um ficheiro funciona para ambos os módulos.

3.1 Formato

Cada linha descreve uma etapa. Os aliases (nomes de estado alternativos no export Jira) são separados por dois pontos. Duas linhas especiais marcam o início e o fim do trabalho ativo:

```
NomeCanónico:Alias1:Alias2
<First>PrimeiraEtapaAtiva
<Closed>EtapaFechada
```

Linha	Significado
NomeCanónico:Alias1:Alias2	Etapa com um ou mais nomes de estado Jira alternativos
NomeEtapa (sem :)	Etapa sem aliases
NomeEtapa	Primeira etapa ativa — onde o processamento começa (início do cycle time)
NomeEtapa	Etapa fechada — marca uma issue como concluída

3.2 Exemplo de fluxo ART_A

O exemplo incluído `workflow_ART_A.txt` modela um fluxo SAFe ART típico.

```
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
```

4 Referência de parâmetros

Todos os parâmetros estão disponíveis tanto na interface gráfica como na linha de comandos.

Parâmetro	Predefinição	Descrição
--workflow FILE	(obrigatório)	Ficheiro de definição do fluxo de trabalho (.txt)
--output FILE.json	_generated.json	Caminho do ficheiro de saída
--project KEY	TEST	Chave do projeto Jira para as issues geradas
--issues N	100	Número de issues a gerar
--from-date YYYY-MM-DD	2025-01-01	Data de criação mais antiga
--to-date YYYY-MM-DD	2025-12-31	Data de transição mais recente
--issue-types TYPE:W ...	Feature:0.6 Bug:0.3 Enabler:0.1	Tipos de issues com pesos
--completion-rate FLOAT	0.7	Fração de issues que atingem a etapa fechada (0–1)
--todo-rate FLOAT	0.15	Fração de issues abertas que ficam em etapas iniciais (0–1)
--backflow-prob FLOAT	0.1	Probabilidade de transição regressiva em cada passo (0–1)
--seed INT	(aleatório)	Seed para resultado reprodutível (opcional)
--mean-cycle-days FLOAT	(nenhum)	Cycle time médio em dias (lognormal); ativa controlo CT
--std-cycle-days FLOAT	(30 % da média)	Desvio padrão do cycle time em dias
--pattern PADRÃO	none	Anti-padrão: none / triangle / flat_triangle / cluster / batch
--pi-duration-weeks INT	12	Duração do ciclo PI em semanas (para cluster/batch)
--pattern-strength 0-100	50	Intensidade do padrão 0–100 (0=subtil, 50=padrão, 100=forte)

4b Modos de Anti-Padrão de fluxo

Desde a versão 0.9.9, o Gerador de Dados de Teste pode simular anti-padrões de fluxo típicos da literatura de Vacanti e Singh. Combine --mean-cycle-days com --pattern.

Padrão	Aparência scatter plot	Descrição
none	Nuvem de pontos uniforme	Cycle time aleatório sem tendência — comportamento predefinido
triangle	Triângulo: ângulo reto em baixo à direita	Cycle time aumenta linearmente ao longo do tempo (multiplicador ~1–4× na intensidade padrão, maior com --pattern-strength). Indica um processo que está a tornar-se progressivamente mais lento.
flat_triangle	Triângulo, aumento atenua no final	Como triangle mas o aumento é atenuado por tanh(2.5×t). Mostra uma deterioração do processo que está a estabilizar.
cluster	Grupos verticais de pontos no final do PI	Cycle time mantém-se estável, mas a maioria das issues é entregue nas últimas 2 semanas antes do final do PI. Comportamento orientado por prazo.
batch	Colunas de altura variável no final do PI	Como cluster mas com cycle time muito variável (0.1–3× média). Mostra que issues curtas e longas são empurradas juntas para o PI.

Controlo do cycle time

```
# Padrão triângulo: CT aumenta de 20d para 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
```

5 ART_A – Exemplo completo

Este exemplo mostra o pipeline completo do gerador ao relatório final, usando o projeto fictício ART_A.

Passo 1: Gerar dados de teste

```
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 \
    --completion-rate 0.75 --seed 42 \
    --output ART_A_generated.json
```

Passo 2: Processar com transform_data

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

Passo 3: Criar relatório com build_reports

```
python -m build_reports
```

5b O cenário de portfólio

Com `--scenario portfolio` o gerador cria num único passo um portfólio de demonstração completo: duas solutions com três ARTs cada, incluindo todos os artefactos da cadeia de processamento.

Na GUI: a secção 'Demo portfolio' na parte inferior da janela gera o cenário numa pasta à escolha com um clique (o campo seed é respeitado, padrão 42); 'Open Portfolio Report' gera o relatório de portfólio e abre-o no browser — sem linha de comandos.

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

A pasta de destino contém depois:

Artefacto	Conteúdo
workflows/	dois ficheiros de fluxo de trabalho (etapas Alpha e Beta)
raw/	seis exports JSON do Jira (uma execução do gerador por ART)
arts/	ficheiros IssueTimes, CFD e Transitions por ART
solution_alpha.json	config de solution sem stage_map (caminho padrão)
solution_beta.json	config de solution com stage_map próprio (schema 2)
risks_*.json	registo de riscos ROAM por solution
nfr_*.json	registo NFR/runway por solution
capabilities_*.json	capability map por solution
dependencies_*.json	registo de dependências por solution
decisions_*.json	log de decisões/suposições por solution
portfolio.json	config de portfólio sobre ambas as solutions
pi_config.json	intervalos PI sobre a janela de dados
README.md	descrição das histórias incorporadas

Os dados são colocados relativamente à data de geração, para que o semáforo de qualidade do relatório de portfólio os avalie como atuais. Três histórias estão incorporadas:

Histórias incorporadas:
- ART Alpha-3 é o outlier (cycle time cerca de três vezes maior) — destacado a vermelho no relatório de comparação da Solution Alpha.
- ART Beta-3 fornece dados fracos (sem CFD, poucos issues iniciados, dados com 60 dias) — confiança 'low' na tabela de qualidade.
- Solution Beta agrega através de um stage_map próprio (Vorlauf/Umsetzung/Fertig); a Solution Alpha usa o caminho padrão.
- Board ROAM: dois riscos owned são deliberadamente antigos (45/50 dias) — o realce de aging dispara.
- NFR & runway: o NFR de API da Beta está violado e um elemento runway é uma lacuna atrasada — o dashboard mostra vermelho.
- Capability map: a capacidade data-insights da Beta é crítica, uma capacidade Alpha não tem ART (uncovered).
- Heatmap de dependências: Alpha-1 → Alpha-3 bloqueada e atrasada; Beta-1 → Alpha-1 como integração cross-solution.
- Log de decisões: a suposição aberta da Beta passou a data de revisão — marcada a vermelho 'review due'.

A pasta é diretamente utilizável: `python -m portfolio demo/portfolio.json` gera o relatório de portfólio; as configs de solution também funcionam individualmente. A mesma seed produz dados idênticos no mesmo dia.

6 Perguntas frequentes (FAQ)

Diferença entre dados de teste e dados reais?

Para demos, formação e testes de instalação, os dados de teste são a melhor escolha — sem preocupações de privacidade, reprodutíveis e totalmente controlados. Para análises reais de fluxo, os dados reais refletem o verdadeiro estado do fluxo de trabalho.

Para exportar dados reais do Jira manualmente: Consulte o Manual do Utilizador Get Data.

7 Glossário

Termo	Explicação
ART	Agile Release Train — estrutura de múltiplas equipas em SAFe
API	Application Programming Interface — interface de programação
CFD	Cumulative Flow Diagram — contagem cumulativa de issues por etapa
Cycle Time	Tempo desde a primeira etapa ativa até à etapa fechada
Issue	Elemento de trabalho no Jira (Feature, Bug, Enabler, Story, …)
JSON	JavaScript Object Notation — formato dos exports da API Jira
Seed	Valor inicial para o gerador aleatório — mesmo seed = mesmo resultado
Stage	Um passo no fluxo de trabalho (corresponde a uma coluna Jira)
Workflow	Sequência ordenada de etapas que uma issue percorre