SituationReport

testdata_generator

Manuel d'utilisation

Générer des données de tickets Jira synthétiques pour SituationReport

---

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

Pour les Agile Coaches et les PI Managers -- Version 0.19.0

1 Qu'est-ce que le Générateur de données de test ?

Le Générateur de données de test (Testdata Generator) crée des données synthétiques de tickets Jira au format Jira REST API. Les fichiers générés peuvent être traités directement par le module transform_data — sans instance Jira réelle.

Le générateur simule des historiques de flux de travail réalistes : les tickets parcourent les étapes définies, passent des durées variables dans chaque étape, régressent occasionnellement et se ferment à un taux configurable.

> Cas d'utilisation typiques :
> - Tester une nouvelle installation : Vérifier que transform_data et build_reports fonctionnent correctement
> - Démos et présentations : Données réalistes sans problèmes de confidentialité
> - Formation : Environnement d'apprentissage contrôlé avec des données connues
> - Développement : Données de test reproductibles via le paramètre seed

Le Générateur complète le module Get Data (en cours de développement), qui charge des données réelles directement depuis Jira. Pour l'export manuel de données Jira réelles, consultez le Manuel d'utilisation Get Data.

2 Prérequis et démarrage

2.1 Package portable (recommandé)

Le package portable inclut Python déjà intégré — aucune installation séparée requise.

Système d'exploitation	Fichier de démarrage
Windows	TestdataGenerator.bat (double-clic)
macOS	TestdataGenerator.command (clic droit → Ouvrir)
Linux	./TestdataGenerator.sh

Note Windows : Si un message d'erreur concernant Python apparaît, le fichier ZIP a peut-être été bloqué par Windows. Solution : clic droit sur le ZIP → Propriétés → Sécurité → Débloquer → OK → extraire à nouveau.

2.2 Depuis le code source (Python requis)

```
python -m testdata_generator
```

2.3 L'interface utilisateur

Au démarrage, la fenêtre principale affiche des champs de saisie pour tous les paramètres. Seul le fichier de flux de travail est obligatoire — tous les autres champs ont des valeurs par défaut sensées.

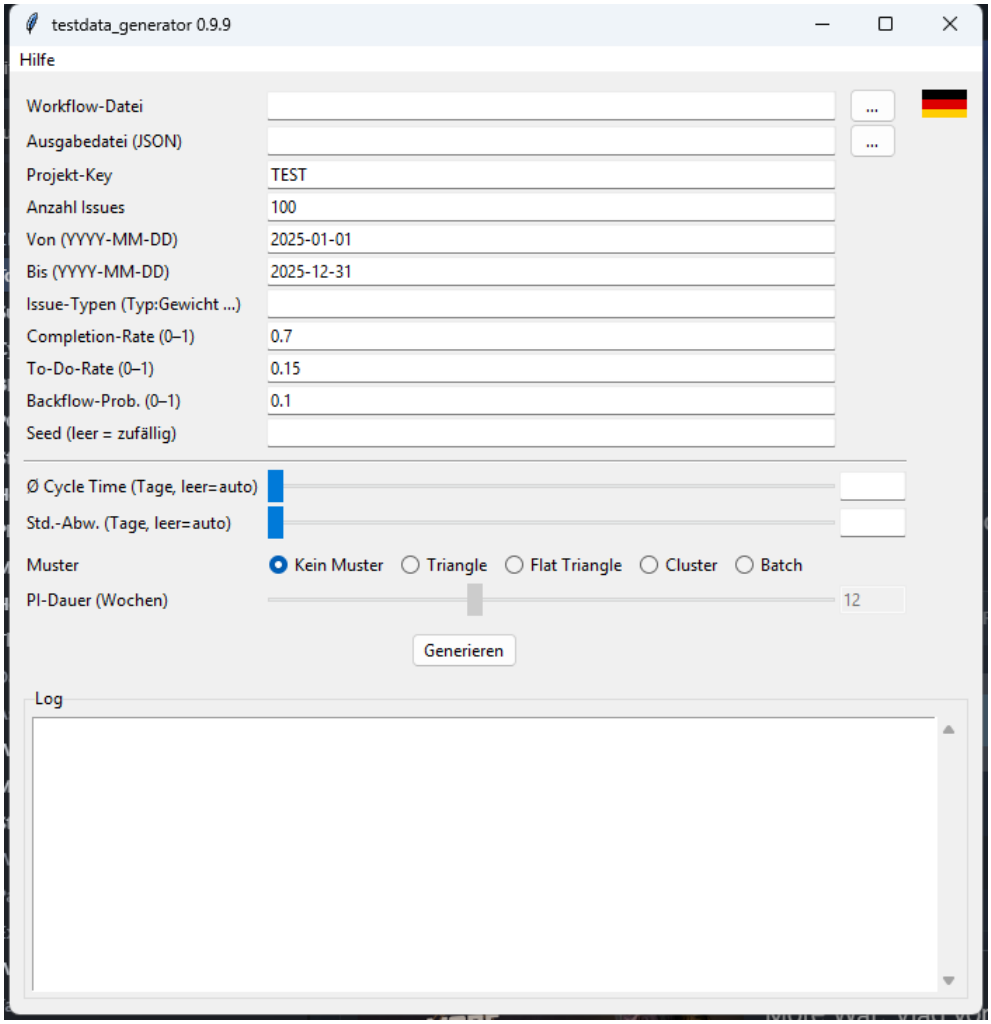

Fig. 1 : Interface utilisateur du Générateur de données de test

3 Le fichier de flux de travail

Le fichier de flux de travail définit les étapes de votre tableau Jira et est le seul paramètre obligatoire. Il utilise le même format que transform_data — un seul fichier fonctionne pour les deux modules.

3.1 Format

Chaque ligne décrit une étape. Les alias (noms de statut alternatifs dans l'export Jira) sont séparés par des deux-points. Deux lignes spéciales marquent le début et la fin du travail actif :

```
NomCanonique:Alias1:Alias2
<First>PremiereEtapeActive
<Closed>EtapeFermee
```

Ligne	Signification
NomCanonique:Alias1:Alias2	Étape avec un ou plusieurs noms de statut Jira alternatifs
NomEtape (sans :)	Étape sans alias
NomEtape	Première étape active — où commence le traitement (début du cycle time)
NomEtape	Étape fermée — marque un ticket comme terminé

3.2 Exemple de flux ART_A

L'exemple fourni `workflow_ART_A.txt` modélise un flux SAFe ART typique.

```
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 Référence des paramètres

Tous les paramètres sont disponibles dans l'interface graphique et en ligne de commande.

Paramètre	Défaut	Description
--workflow FILE	(obligatoire)	Fichier de définition du flux de travail (.txt)
--output FILE.json	_generated.json	Chemin du fichier de sortie
--project KEY	TEST	Clé de projet Jira pour les tickets générés
--issues N	100	Nombre de tickets à générer
--from-date YYYY-MM-DD	2025-01-01	Date de création la plus ancienne
--to-date YYYY-MM-DD	2025-12-31	Date de transition la plus récente
--issue-types TYPE:W ...	Feature:0.6 Bug:0.3 Enabler:0.1	Types de tickets avec pondérations
--completion-rate FLOAT	0.7	Fraction de tickets atteignant l'étape fermée (0–1)
--todo-rate FLOAT	0.15	Fraction de tickets ouverts restant dans les étapes initiales (0–1)
--backflow-prob FLOAT	0.1	Probabilité d'une transition en arrière à chaque pas (0–1)
--seed INT	(aléatoire)	Seed pour résultat reproductible (optionnel)
--mean-cycle-days FLOAT	(aucun)	Cycle time moyen en jours (lognormal) ; active le contrôle CT
--std-cycle-days FLOAT	(30 % de la moyenne)	Écart type du cycle time en jours
--pattern MOTIF	none	Anti-pattern : none / triangle / flat_triangle / cluster / batch
--pi-duration-weeks INT	12	Durée du cycle PI en semaines (pour cluster/batch)
--pattern-strength 0-100	50	Intensité du motif 0–100 (0=subtil, 50=défaut, 100=fort)

4b Modes d'anti-pattern de flux

Depuis la version 0.9.9, le Générateur peut simuler des anti-patterns de flux typiques de la littérature de Vacanti et Singh. Combinez --mean-cycle-days avec --pattern.

Motif	Apparence scatter plot	Description
none	Nuage de points uniforme	Cycle time aléatoire sans tendance — comportement par défaut
triangle	Triangle : angle droit en bas à droite	Le cycle time augmente linéairement dans le temps (multiplicateur ~1–4× à l'intensité par défaut, plus élevé avec --pattern-strength). Indique un processus qui devient progressivement plus lent.
flat_triangle	Triangle, augmentation s'atténue à la fin	Comme triangle mais l'augmentation est atténuée par tanh(2.5×t). Montre une détérioration du processus qui se stabilise.
cluster	Groupes verticaux de points en fin de PI	Le cycle time reste stable, mais la plupart des tickets sont livrés dans les 2 dernières semaines avant la fin du PI. Comportement orienté par délai.
batch	Colonnes de hauteur variable en fin de PI	Comme cluster mais avec cycle time très variable (0.1–3× moyenne). Montre que tickets courts et longs sont tous poussés en fin de PI.

Contrôle du cycle time

```
# Motif triangle : CT augmente de 20j à 80j
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 – Exemple complet

Cet exemple montre le pipeline complet du générateur au rapport final, en utilisant le projet fictif ART_A.

Étape 1 : Générer les données de test

```
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
```

Étape 2 : Traiter avec transform_data

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

Étape 3 : Créer le rapport avec build_reports

```
python -m build_reports
```

5b Le scénario de portefeuille

Avec `--scenario portfolio`, le générateur crée en une seule étape un portefeuille de démonstration complet : deux solutions de trois ARTs chacune, avec tous les artefacts de la chaîne de traitement.

Dans la GUI : la section 'Demo portfolio' en bas de la fenêtre génère le scénario dans un dossier de votre choix en un clic (le champ seed est repris, 42 par défaut) ; 'Open Portfolio Report' génère le rapport de portefeuille et l'ouvre dans le navigateur — sans ligne de commande.

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

Le dossier cible contient ensuite :

Artefact	Contenu
workflows/	deux fichiers de flux de travail (étapes Alpha et Beta)
raw/	six exports JSON Jira (une exécution du générateur par ART)
arts/	fichiers IssueTimes, CFD et Transitions par ART
solution_alpha.json	config de solution sans stage_map (chemin par défaut)
solution_beta.json	config de solution avec stage_map propre (schéma 2)
risks_*.json	registre de risques ROAM par solution
nfr_*.json	registre NFR/runway par solution
capabilities_*.json	capability map par solution
dependencies_*.json	registre de dependances par solution
decisions_*.json	log de decisions/hypotheses par solution
portfolio.json	config de portefeuille couvrant les deux solutions
pi_config.json	intervalles PI sur la fenêtre de données
README.md	description des histoires intégrées

Les données sont placées relativement à la date de génération, afin que le feu de qualité du rapport de portefeuille les évalue comme actuelles. Trois histoires sont intégrées :

Histoires intégrées :
- ART Alpha-3 est la valeur aberrante (cycle time environ triplé) — surligné en rouge dans le rapport de comparaison de la Solution Alpha.
- ART Beta-3 fournit des données faibles (pas de CFD, peu d'issues commencées, données vieilles de 60 jours) — confiance 'low' dans le tableau de qualité.
- Solution Beta agrège via son propre stage_map (Vorlauf/Umsetzung/Fertig) ; la Solution Alpha utilise le chemin par défaut.
- Board ROAM : deux risques owned sont volontairement anciens (45/50 jours) — le surlignage aging se déclenche.
- NFR & runway : le NFR d'API de Beta est viole et un element runway est une lacune en retard — le dashboard montre du rouge.
- Capability map : la capacite data-insights de Beta est critique, une capacite Alpha n'a pas d'ART (uncovered).
- Heatmap des dependances : Alpha-1 → Alpha-3 bloquee et en retard ; Beta-1 → Alpha-1 en integration cross-solution.
- Log de decisions : l'hypothese ouverte de Beta a depasse sa date de revision — marquee en rouge 'review due'.

Le dossier est directement utilisable : `python -m portfolio demo/portfolio.json` génère le rapport de portefeuille ; les configs de solution fonctionnent aussi individuellement. La même graine produit des données identiques le même jour.

6 Foire aux questions (FAQ)

Différence entre données de test et données réelles ?

Pour les démos, la formation et les tests d'installation, les données de test sont le meilleur choix — sans problèmes de confidentialité, reproductibles et entièrement contrôlées. Pour les analyses de flux réelles, les données réelles reflètent l'état réel du flux de travail.

Pour l'export manuel de données Jira réelles : Consultez le Manuel d'utilisation Get Data.

7 Glossaire

Terme	Explication
ART	Agile Release Train — structure multi-équipes dans SAFe
API	Application Programming Interface — interface de programmation
CFD	Cumulative Flow Diagram — comptage cumulé des tickets par étape
Cycle Time	Temps depuis la première étape active jusqu'à l'étape fermée
Issue	Élément de travail dans Jira (Feature, Bug, Enabler, Story, …)
JSON	JavaScript Object Notation — format des exports API Jira
Seed	Valeur initiale pour le générateur aléatoire — même seed = même résultat
Stage	Une étape dans le flux de travail (correspond à une colonne Jira)
Workflow	Séquence ordonnée d'étapes que parcourt un ticket