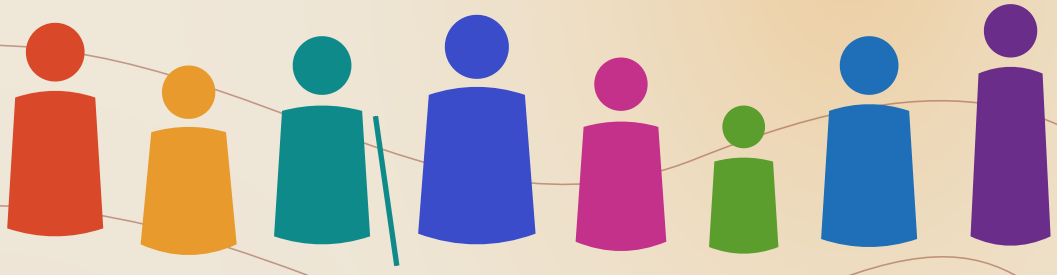
SVRUS IO · APPLIED AI ETHICS PRESS

EDITION 02 · MMXXVI

A METHODOLOGY FOR PRACTITIONERS



Fairness *Implementation* Playbook

Deploying fairness solutions across AI systems and organizations — from the agile sprint to the boardroom, from training data to post-market surveillance, with the *fairpipe* reference toolchain.

VOLUME I · PRACTITIONER EDITION

74 PAGES · ISBN 978-1-XXXXX-XX-X

COLOPHON & ABBREVIATIONS

A note on the work

The *Fairness Implementation Playbook* is a methodology for deploying fairness solutions across AI systems and organizations. It bridges technical solutions with organizational realities, creating practical pathways for systematic fairness adoption.

This second edition is published by Svrus LLC. Where the first edition described methods and frameworks, this edition adds **fairpipe** — an open-source reference toolchain (v0.6.5) that operationalises every framework in the book through a Python package and CLI. Each module of fairpipe carries an explicit cross-reference back to the relevant chapter and table here, so readers may move freely between the methodology and its executable implementation.

Set in **Source Serif 4** for body matter, **Archivo** for display and metadata, and **JetBrains Mono** for code listings. Printed on FSC-certified Munken Pure 100 gsm. Composition by the editorial team at Svrus Press, with abstract illustration in flat geometric style by the same.

Abbreviations used throughout

- **RACI** — Responsible, Accountable, Consulted, Informed
- **DoD** — Definition of Done
- **CI/CD** — Continuous Integration / Continuous Deployment
- **TRS** — Total Risk Score
- **FDR** — Fairness Decision Record
- **FDL** — Fairness Decision Log
- **SDLC** — Software Development Life Cycle
- **PO / SM / Dev / DS** — Product Owner, Scrum Master, Developer, Data Scientist
- **QA** — Quality Assurance
- **AI Act** — EU Artificial Intelligence Act
- **GDPR** — General Data Protection Regulation
- **DPIA** — Data Protection Impact Assessment
- **DPO** — Data Protection Officer
- **LLM(s)** — Large Language Model(s)
- **DAG** — Directed Acyclic Graph
- **S3** — Amazon Simple Storage Service
- **TPR / FNR** — True Positive Rate / False Negative Rate
- **AUC** — Area Under the Curve
- **RecSys** — Recommender Systems shorthand
- **CI (statistical)** — Confidence Interval
- **FAF** — Fairness Audit Framework
- **CAIEO** — Chief AI Ethics Officer

CITATION

Svrus LLC. *Fairness Implementation Playbook*, 2nd ed. Applied AI Ethics Press, 2026. fairpipe v0.6.5.

READING PATHS

Practitioners new to fairness should begin with Part 1. Engineering teams adopting fairpipe may jump directly to Part 5, then loop back to Parts 2–4 for governance context. Auditors and counsel should read Parts 2 and 4 first.

SOURCE CODE

The fairpipe package, examples and schemas are released under Apache 2.0 and tracked at `github.com/svrus/fairpipe`.

ERRATA

Submit corrections to `playbook@svrus.io`. A living errata sheet is maintained on the publication website.

FROM PRINCIPLE TO PRACTICE

Introduction

The rise of artificial intelligence has brought about significant advancements — and new responsibilities. Among them, the need to ensure that AI systems are fair, equitable, and free from bias is paramount. This *Fairness Implementation Playbook* transforms abstract principles into concrete practices. It designs the workflows, governance structures, and team practices needed for consistent fairness implementation. Rather than treating fairness as a specialised technical concern, it embeds considerations throughout your organisation's AI development process.

How the Playbook is organised

To guide organisations through the nuanced process of embedding fairness in AI, the Playbook is thoughtfully organised into a series of interconnected sections. Each part builds on the last — moving from principles to practice, and from team-level implementation to organisation-wide governance. The structure creates a logical pathway for teams and leaders, ensuring no aspect of fairness is left behind as systems scale.

OVERVIEW OF PLAYBOOK SECTIONS

Part 1 · The Fair AI Scrum Toolkit. This foundational section introduces methods for integrating fairness into agile development. It provides practical techniques for modifying user stories, sprint backlogs and acceptance criteria to explicitly incorporate fairness. Teams learn how to adapt Scrum ceremonies — planning, reviews, retrospectives — to prioritise bias detection and mitigation. Templates, frameworks and checklists are offered so fairness becomes a regular, visible element throughout the development cycle.

Part 2 · Organizational Integration Toolkit. Moving beyond individual teams, this section equips organisations for sustainable fairness by establishing governance structures, escalation procedures and role-specific responsibilities. Frameworks for decision-making, accountability and documentation foster clear ownership and consistent standards. Guidance on creating cross-functional committees, mapping authority levels and building hybrid governance models ensures fairness is a core value, not a siloed effort.

A BRIDGE, NOT A MANIFESTO

The Playbook bridges technical solutions with organisational realities, creating practical pathways for systematic fairness adoption.

FIVE PARTS, ONE SPINE

Read sequentially or independently. Every reference points both upward (to principle) and downward (to executable code in fairpipe).

WHOM THIS IS FOR

Product owners · Scrum masters · ML engineers · Data scientists · UX researchers · Legal & ethics counsel · DPOs · Compliance officers · Executive sponsors.

OVERVIEW OF PLAYBOOK SECTIONS (CONT.)

Part 3 · Advanced Architecture Cookbook. Addressing the complexities of modern AI systems, this part offers architecture-specific strategies for recommendation engines, large language models, computer vision and multi-modal systems. Recipes, example pitfalls, validation targets and specialised primitives address bias in technical pipelines for teams facing unique challenges where generic interventions fall short.

Part 4 · Regulatory Compliance Guide. Organisations must align fairness efforts with evolving legal and policy requirements. This guide translates global and local regulations into actionable controls, documentation standards, and evidence trails. Step-by-step worksheets, risk tiering frameworks, compliance checklists and monitoring protocols are mapped to common lifecycle phases.

Part 5 · The fairpipe Reference Toolchain. New in this edition. A complete walkthrough of the *fairpipe* Python package (v0.6.5, by Svrus LLC) — its five modules, CLI, YAML configuration schema and architecture-specific recipes — together with a cross-reference mapping back to every framework and table in Parts 1 through 4.

HOW THE PARTS INTERCONNECT

The Playbook is intentionally modular, allowing readers to start with whichever section is most relevant to their current needs — while ensuring all elements can be linked for comprehensive coverage. Teams may begin by piloting the Fair AI Scrum Toolkit and later expand to organisational integration and regulatory compliance as their maturity grows. Advanced architectural guidance and the appendices support ongoing evolution.

Each section is designed with actionable steps, clear examples and practical templates. The Playbook moves from conceptual frameworks to concrete workflows, ensuring fairness is not just an aspiration but a repeatable, measurable reality throughout the lifecycle of every AI system.

MATURITY ARC

Most teams pilot Part 1 in one squad, expand Parts 2 & 4 across the org over six to twelve months, and adopt Parts 3 & 5 once two systems are in production.

NEW IN V0.6.5

Part 5 documents the executable companion to the methodology. Cross-references in the margin of every chapter point to the specific fairpipe primitive that implements the framework.

HOW TO READ

Tables are numbered consecutively across the volume. Code listings carry a language tag. Pull-quotes are editorial summaries, not authoritative statements.

CONTENTS · VOLUME I

Table of Contents

FRONT MATTER

PART 1 · FAIR AI SCRUM TOOLKIT

PART 2 · ORGANIZATIONAL INTEGRATION & GOVERNANCE

PART 3 · ADVANCED ARCHITECTURE COOKBOOK

PART 4 · REGULATORY COMPLIANCE GUIDE

PART 5 · THE FAIRPIPE TOOLCHAIN

APPENDICES

PART ONE · TEAM PRACTICE

Fair AI *Scrum* Toolkit

Integrating fairness into agile development — user stories, sprint backlogs, ceremonies, and the cadence of bias detection at the team level.

AUDIENCE	FRAMEWORKS	READING TIME	PAIRS WITH
Squads & Pods	***SAFE · FAIR · DoD***	***≈ 45 min***	***fairpipe · pipeline***

PART ONE · INTRODUCTION

Fairness, woven into the cadence of work.

The rise of artificial intelligence has brought about significant advancements — and new responsibilities. Among them, the need to ensure that AI systems are fair, equitable and free from bias is paramount. The Fair AI Scrum Toolkit will help embed fairness checkpoints throughout Scrum workflows, ensuring teams address bias as part of standard development practices rather than as an afterthought.

It provides practical frameworks, templates and practices to infuse fairness into every stage of the Scrum process — from backlog refinement through deployment and retrospectives.

A NOTE ON METRICS

All fairness metrics used in SAFE/FAIR user stories below are formally defined in the Fairness Audit Framework, Fairness Definition Catalog. Teams should not redefine metrics locally. Use the FAF definitions for Demographic Parity and Equalized Odds; for the others see **Appendix 4 · Fairness Metric Glossary**.

WHERE THIS LIVES IN FAIRPIPE

The SAFE/FAIR contract surfaces in fairpipe as `fairpipe.measurement.metrics` and as the `fairness:` stanza in the YAML config. See Part 5 §1 and Part 5 cross-reference.

ANTI-PATTERN

"Make sure it's fair" — a story that names no protected attribute, no metric and no threshold is unstartable; refuse it at refinement.

1. Scrum Artifacts Modification Guide: Integrating Fairness

To build AI systems that are robust and fair, Scrum artifacts must reflect explicit fairness considerations. This section outlines how to modify core Scrum artifacts — user stories, sprint backlogs, and acceptance criteria — to embed fairness from inception.

1.1 User Stories

Extend standard user stories with a fairness clause: every AI user story should include a clause or sub-task addressing fairness implications. This includes identifying potential sources of bias and outlining mitigation strategies. Practically, that means:

- Identify who might be negatively impacted by the AI feature.
- Anticipate potential bias or discrimination in the AI's outputs.
- State fairness requirements explicitly in the story description.

A standard user story template takes the form:

> *"As a **[role]**, I want **[functionality]** so that **[benefit]**."*

This structure captures functional needs but misses fairness considerations. The gap creates risks where bias enters through seemingly neutral implementations. **Fairness-enhanced user stories** extend this template to include protected attributes, fairness definitions and potential bias risks:

> *"As a **[role]**, I want **[functionality]** so that **[benefit]**, ensuring **[fairness goal]** across **[protected attributes]**."*

This extension transforms abstract fairness principles into specific requirements tied to each feature.

SAFE USER STORY FRAMEWORK

The SAFE framework provides a structured approach to developing fairness-enhanced user stories.

SAFE · a four-letter discipline for the story author

- **S** **Specific protected attributes** — Explicitly identify which demographic groups and intersections require fairness analysis.
- **A** **Actionable fairness definition** — Specify the fairness definition (e.g., demographic parity, equal opportunity) appropriate for this feature.
- **F** **Feature integration points** — Identify where in the feature fairness considerations most critically apply.
- **E** **Expected outcome measures** — Define how fairness will be quantitatively validated.

TABLE 1 · EXAMPLES OF FAIRNESS-ENHANCED USER STORIES

Traditional	Fairness-Enhanced
"As a loan officer, I want to see applicants ranked by risk score so I can focus on qualified candidates."	"As a loan officer, I want to see applicants ranked by risk score so I can focus on qualified candidates, ensuring equivalent score distribution across gender, race, and age groups."
"As a recruiter, I want résumés categorized by relevant experience so I can efficiently screen candidates."	"As a recruiter, I want résumés categorized by relevant experience so I can efficiently screen candidates, ensuring that experience evaluation works with equivalent accuracy across demographic groups and non-traditional career paths."
"As a content moderator, I want offensive comments automatically flagged so I can review them quickly."	"As a content moderator, I want offensive comments automatically flagged so I can review them quickly, ensuring equivalent flagging rates across content discussing different cultures, identities and political views."

The SAFE framework applies across the ML lifecycle:

- During **requirements**, it guides product owners in specifying fairness needs.
- During **implementation**, it helps developers select appropriate fairness techniques.
- During **testing**, it provides clear validation criteria.

1.2 Sprint Backlogs

Traditional sprint backlogs often fail to identify fairness-specific tasks, instead assuming fairness work happens automatically alongside functional development. When fairness tasks do appear, they frequently lack adequate estimation guidance, leading to chronic under-resourcing.

A structured fairness task taxonomy identifies common fairness activities with estimation guidance based on complexity and risk level. Example task types include:

- **Fairness Analysis Tasks**: Data bias audit, model evaluation across groups, intersectional performance testing.
- **Fairness Implementation Tasks**: Bias mitigation implementation, fair feature engineering, fairness constraint application.
- **Fairness Validation Tasks**: Acceptance criteria testing, fairness regression testing, documentation creation.

The taxonomy creates space for both technical implementations and participatory activities by identifying distinct task types and providing appropriate resource allocation.

Prioritise fairness tasks. Add fairness-related investigation and mitigation tasks to the backlog (e.g., data bias audits, fairness metrics selection). Ensure at least one fairness task is included in every sprint.

TABLE 2 · EXAMPLE FAIRNESS-ENHANCED SPRINT BACKLOG

Task	Type	Description	Fairness Objective	Story Pts	Owner	Status
Audit dataset for demographic skew	Fairness analysis	Conduct a group distribution audit on gender and ethnicity in training data	Demographic Parity	3	Data Scientist	In Progress
Implement reweighing pre-processing	Fairness implementation	Apply reweighing to mitigate dataset imbalance before training	Demographic Parity	5	ML Engineer	To Do
Add fairness regression tests in CI/CD	Fairness validation	Ensure fairness metrics remain within thresholds in pipeline	Regression Fairness	3	DevOps	To Do
Implement user login feature	—	Standard functional sprint item	N/A	5	Backend Dev	In Progress
Add email verification	—	Standard functional sprint item	N/A	3	Backend Dev	Done

Story Pts · agile story-point estimate of relative effort; values here use the team's calibrated 1–8 scale (Fibonacci) where 1 = under a day, 5 = roughly half a sprint.

1.3 Acceptance Criteria

Traditional acceptance criteria define when a feature is complete from a functional perspective. This framework often misses critical fairness dimensions, allowing biased implementations to pass quality gates.

Embed fairness validation. Include objective fairness metrics as part of acceptance criteria (e.g., disparate impact ratio, demographic parity, equal opportunity). Require explanation of how fairness was assessed and validated before a story is considered "done." Fairness acceptance criteria provide ways to translate algorithmic fairness research into actionable items by connecting abstract goals to concrete validation requirements.

FAIR ACCEPTANCE CRITERIA FRAMEWORK

FAIR · a four-letter discipline for the reviewer

- **F** **Fairness metrics thresholds** — Quantitative fairness standards the feature must meet.
- **A** **Auditing requirements** — Specific fairness tests that must be performed and documented.
- **I** **Intersectional analysis** — How performance across intersectional groups will be validated.
- **R** **Reporting guidelines** — How fairness results will be documented and communicated.

TABLE 3 · EXAMPLE FAIRNESS ACCEPTANCE CRITERIA

Component	Example Acceptance Criteria
Data	Dataset contains at least 2,000 samples for each demographic group identified in the user story. Representation gaps between demographic groups don't exceed 10%. Data quality metrics (missing values, noise) are equivalent across groups.
Model	Demographic parity difference doesn't exceed 0.05 across specified protected attributes. True positive rates are equivalent (within 0.03) across all demographic groups. Calibration error differences between groups remain below 0.05.
User Interface	Interface has been tested with users from all demographic groups mentioned in the user story. Decision explanations are equally understandable across diverse users (validated through user testing). Override mechanisms are equally usable across all groups.

The framework creates validation standards across ML components: data acceptance criteria might require sampling parity across protected groups; model criteria might specify maximum performance disparities; interface criteria might require accessibility testing.

INTERSECTIONALITY CONSIDERATION

Traditional fairness approaches often focus on binary protected attributes (e.g., male/female, majority/minority race), missing critical disparities at demographic intersections. This limitation extends to user stories, where even fairness-conscious teams might specify requirements for gender and race separately while missing unique challenges facing women of colour.

To embed intersectional principles in user stories and acceptance criteria:

- Extend user stories to explicitly identify relevant intersectional groups rather than listing attributes separately.
- Use the phrase "across all intersections of" rather than simply listing attributes independently.
- Specify acceptance criteria that require disaggregated performance reporting across intersectional categories.

These modifications create practical implementation challenges. Teams must balance comprehensive intersectional coverage against the complexity of testing numerous demographic subgroups. Product Owners must learn which intersectional categories face highest risk in their application context to prioritise testing resources.

IMPLEMENTATION FRAMEWORK

To integrate fairness user stories and acceptance criteria into your development process:

1 · Analyse feature fairness risks. Identify protected attributes relevant to each feature; assess potential bias impacts across demographic groups; determine appropriate fairness definitions based on use case.

2 · Develop fairness-enhanced user stories. Start with traditional user story structure; apply the SAFE framework to add fairness dimensions; verify stories capture both functional and fairness needs.

3 · Create fairness acceptance criteria. Use the FAIR framework to develop comprehensive criteria; specify quantitative thresholds for fairness metrics; include qualitative evaluation requirements.

4 · Integrate with development workflow. Add fairness-enhanced stories to product backlog; include fairness acceptance criteria in definition of done; establish testing protocols for validating fairness criteria.

IMPLEMENTATION CHALLENGES

Common implementation pitfalls include:

- **Overly Generic Fairness Requirements**: User stories with vague fairness goals like "ensure the system is fair" provide insufficient guidance. Address this by specifying concrete fairness definitions and protected attributes for each feature.
- **Unrealistic Fairness Thresholds**: Setting perfect fairness requirements (e.g., exactly equal outcomes across all groups) often creates unattainable standards. Instead, establish reasonable thresholds based on context and potential harm, acknowledging that some disparity may remain.
- **Neglecting Qualitative Fairness**: Focusing exclusively on quantitative fairness metrics ignores important qualitative dimensions. Balance quantitative criteria with qualitative evaluation requirements that assess user experience across diverse groups.

Resources required for implementation include:

- Initial team training on fairness concepts (2–4 hours).
- Fairness analysis during user story creation (15–30 minutes per story).
- Expanded testing resources for validating fairness acceptance criteria (10–20% increase).
- Potential domain expertise for context-specific fairness considerations.

EVALUATION APPROACH

To assess successful implementation of fairness user stories and acceptance criteria, establish these metrics:

TABLE 4 · FAIRNESS USER STORIES & ACCEPTANCE CRITERIA EVALUATION METRICS

Metric	Definition	Recommended Threshold (High-Risk AI)
Story Coverage	Percentage of user stories with explicit fairness dimensions.	100% of user stories should include explicit fairness dimensions.
Criteria Specificity	Ratio of quantitative to qualitative fairness criteria.	At least 80% of fairness criteria should include quantitative thresholds.
Bias Detection Timing	When bias issues are discovered in the development cycle.	90% of bias issues should be identified before production deployment.
Resolution Efficiency	Time required to address identified fairness issues.	Mean time to resolve fairness issues should be under two weeks.

"Fairness-enhanced user stories transform abstract principles into specific requirements tied to each feature."

— ON THE SAFE FRAMEWORK, §1.1

2 Fairness User Story Template Library

To foster consistency and thoroughness, the toolkit provides templates and examples for writing fairness-aware user stories. These examples address common sources of bias in AI systems.

2.1 Fairness User Story Template

- As a **[role]**
- I want **[functionality]**
- so that **[benefit]**,
- ensuring **[fairness goal]**
- across **[protected attribute(s)]**.

TABLE 5 · FAIRNESS USER STORY TEMPLATE STRUCTURE

Field	Description
User Story	Standard agile user story including role, functionality, and benefit.
Fairness Dimension	Type of fairness goal (e.g., demographic parity, equal opportunity).
Protected Attribute(s) / Bias Scenario	Specific protected attribute(s) or type of bias that this story addresses.
Acceptance Criteria	How success is measured, ideally with quantitative thresholds.
Data & Metrics Notes	Requirements for dataset documentation, model explainability, etc.

2.2 Example User Stories

1 · Demographic Parity in Hiring Model

TABLE 6 · DEMOGRAPHIC PARITY IN HIRING — STORY EXAMPLE

Field	Description
User Story	As a hiring manager, I want the model to recommend qualified candidates so I can efficiently screen candidates, ensuring demographic parity across genders so that the process does not favour one group.
Fairness Dimension	Demographic Parity
Protected Attribute(s) / Bias Scenario	Gender bias in resume screening
Acceptance Criteria	**Selection Rate Equality:** The model's recommended candidates must exhibit less than 5% variance in selection rates between male and female applicants. **Audit Reporting:** A demographic parity report must be generated per training iteration, highlighting group-wise selection rates. **Test Case Inclusion:** The fairness unit test suite must include gender-based parity checks on historical datasets. **Rejection Threshold:** If demographic parity deviates by more than 5%, model deployment is blocked until mitigation steps are applied.
Data & Metrics Notes	Document gender distribution in training set; use demographic parity difference metric.

2 · Equal Opportunity in Credit Scoring

TABLE 7 · EQUAL OPPORTUNITY IN CREDIT SCORING — STORY EXAMPLE

Field	Description
User Story	As a loan officer, I want the AI model to rank applicants by risk scores so I can focus on qualified applicants; the model should give equal chances to qualified applicants from all races so that deserving individuals are not denied loans due to systemic bias.
Fairness Dimension	Equal Opportunity
Protected Attribute(s) / Bias Scenario	Race — racial disparities in loan approval.
Acceptance Criteria	True Positive Rates across racial groups must be < 10% of each other; disparity highlighted in validation report; model retrained if bias exceeds threshold.
Data & Metrics Notes	Audit FICO vs predicted score by race; log feature importances.

3 Definition of Done Framework: Fairness Validation Before Deployment

Standard definition of done criteria focus on code quality and performance. They miss crucial fairness validation steps. This creates a gap where bias slips through development cycles undetected.

A **fairness-enhanced definition of done** institutionalises fairness in AI/ML systems development by embedding bias checks, subgroup testing, and equity-oriented reviews into the Definition of Done (DoD), ensuring no feature or model reaches deployment without meeting fairness standards.

DOD COMPONENTS	REQUIRED SIGN-OFFS	MIN OWNERS	AUDIT ARTIFACTS
8	3	5	10+

TABLE 8 · FAIRNESS-ENHANCED DEFINITION OF DONE FRAMEWORK STRUCTURE

#	Component	Description	Required Artifacts / Outputs	Owner(s)
1	Functional Validation	Feature behaves as intended.	Test cases, Pass/Fail reports	QA Engineer, Developer
2	Fairness Pre-Check Inclusion	Confirm applicability of fairness concerns (e.g. user-facing AI decisions).	Fairness Applicability Checklist	Product Manager, ML Lead
3	Disaggregated Performance Evaluation	Test and report model/feature performance across intersectional groups (e.g. race × gender).	Performance disaggregated metrics (TPR, FNR, Accuracy, etc.)	ML Engineer, Data Scientist
4	Fairness Acceptance Criteria	Document and verify predefined fairness thresholds or parity gaps (e.g. Demographic Parity Difference ≤ 0.10).	Bias metric reports with CI; Threshold pass/fail validation	ML Engineer, QA Analyst
5	Fairness Intervention Logging	Document mitigations applied if disparity exists (reweighting, post-processing, sample balancing).	Intervention Strategy Report	ML Fairness Lead
6	Stakeholder Review Panel	Review fairness results and approve progression to deployment.	Signed Review Log; Meeting Notes	Product Owner, Ethics Lead, Diverse Stakeholder Panel (incl. impacted users)
7	Fairness Documentation	Centralised record of fairness evaluations for auditing and transparency.	Model Card or Feature Fairness Report	ML Lead, Documentation Officer
8	Ethical Go/No-Go Decision	Final sign-off explicitly includes fairness evaluation.	Sign-off Checklist	Product Owner, Governance Lead

TIP Treat each DoD row as a CI/CD gate. In fairpipe (Part 5), rows 3, 4 and 5 are produced automatically by `fairpipe pipeline run` and persisted to the audit trail described in Part 4.

4 Ceremony Adaptation Guide: Fairness in Scrum Events

To make fairness a living priority, Scrum ceremonies must explicitly address it. This section outlines modifications to key ceremonies.

4.1 Sprint Planning

Traditional sprint planning meetings focus primarily on how many functional points a team can commit to delivering. Fairness considerations often receive limited attention beyond vague reminders to "make sure it's fair."

Fairness-enhanced sprint planning meetings include structured fairness discussion points, capacity reserves, and role-specific responsibilities. Key adaptations include:

- **Fairness Risk Assessment**: Structured evaluation of each planned feature's bias potential.
- **Capacity Earmarking**: Explicit allocation of sprint capacity to fairness tasks.
- **Fairness Task Identification**: Systematic process for identifying necessary fairness tasks.
- **Role Assignment**: Clear fairness responsibilities for each team member.
- **Fairness Checkpoint Definition**: Explicit points during the sprint for fairness validation.

These adaptations shape sprint execution in multiple ways. The fairness risk assessment guides monitoring focus during the sprint. Capacity earmarking ensures fairness tasks receive adequate resources. Role assignments create clear accountability for fairness outcomes.

4.2 Daily Standups

Fairness-enhanced daily practices incorporate structured fairness tracking, checkpoints and metrics throughout the sprint. Key components include:

- **Fairness Standup Prompts**: Explicit questions about fairness progress and blockers.
- **Fairness Progress Visualisation**: Visible tracking of fairness metrics alongside functional progress.
- **Fairness Blocker Escalation**: Clear protocol for raising and addressing fairness blockers.
- **Mid-Sprint Fairness Checkpoints**: Scheduled review points for fairness validation (see §4.4).

These practices impact sprint execution at multiple points. During daily standups, fairness prompts ensure teams discuss fairness progress. Throughout the day, visibility tools maintain awareness of fairness status. At checkpoints, dedicated reviews ensure fairness doesn't drift during implementation.

TABLE 9 · SPRINT FAIRNESS METRICS FOR EVALUATION

Metric	Definition	Example / Threshold for High-Risk Apps
Fairness Capacity Utilisation	% of allocated fairness capacity actually used for fairness work.	90%+ utilisation indicates focus was maintained on fairness tasks.
Fairness Task Completion Rate	Ratio of completed fairness tasks to those planned in the sprint.	At least 95% for high-risk domains like healthcare or justice.
Bias Issue Detection Timing	Stage in sprint when bias-related issues are flagged.	80%+ of issues should be identified before final testing.
Fairness Blocker Resolution Time	Avg. time taken to resolve blockers preventing fairness task completion.	Mean resolution time under 2 days for blockers related to bias tools, data or approval.
Fairness Documentation Coverage	Presence of rationale for fairness capacity decisions, model/data limitations and metrics.	100% of fairness tasks should include documentation per Vethman et al. (2025).

4.3 Sprint Review

Traditional sprint reviews focus on demonstrating functional achievements to stakeholders. Teams showcase completed features and gather feedback primarily on usability and business value. Fairness considerations often appear briefly if at all, typically limited to high-level statements like "we've ensured the model is fair."

Fairness-enhanced sprint reviews explicitly showcase fairness achievements alongside functional ones. They include:

- **Fairness Metric Presentations**: Visualising key fairness metrics across demographic groups.
- **Disaggregated Performance Reports**: Showing system performance for different protected attributes and intersections.
- **Bias Mitigation Demonstrations**: Explaining implemented fairness interventions and their results.
- **Remaining Fairness Debt**: Transparently discussing unresolved fairness issues.

These reviews impact multiple development stages: they validate fairness achievements for completed work; educate stakeholders about fairness trade-offs; gather feedback that shapes fairness priorities for future sprints; and create accountability for fairness outcomes rather than just fairness intentions.

Traditional sprint reviews often struggle to effectively communicate fairness concepts to stakeholders. Technical metrics like demographic parity or equal opportunity remain abstract to non-specialists. **Fairness demonstration techniques** use concrete examples, visualisations and scenarios to make fairness properties tangible — interactive dashboards; counterfactual demonstrations; real-world impact scenarios; before/after comparisons.

4.4 Mid-Sprint Fairness Checkpoints

Traditional Scrum relies primarily on beginning and end-of-sprint ceremonies, with daily standups providing lightweight progress tracking. This cadence creates long gaps between formal checkpoints, during which fairness issues can accumulate undetected.

Mid-sprint fairness checkpoints create additional verification points focused specifically on bias detection and fairness validation. Key checkpoint types include:

- **Pre-Implementation Design Reviews**: Evaluating fairness implications before coding begins.
- **Data Pipeline Validation**: Verifying fairness properties of data transformations.
- **Model Training Reviews**: Examining fairness metrics during early training iterations.
- **Integration Fairness Tests**: Testing for bias after component integration.

Mid-sprint checkpoints affect multiple development stages. They validate fairness assumptions during design, catch data bias before it affects models, and identify fairness issues during implementation when fixes require minimal rework.

4.5 Retrospectives

Traditional retrospectives examine what went well, what didn't, and what to improve for the next sprint. While valuable for general process improvement, these broad prompts often fail to surface specific fairness learnings. Fairness challenges get lost among more visible technical and process issues.

Fairness retrospective techniques use specialised prompts, exercises and frameworks to extract fairness learnings.

Fairness-Specific Prompts — questions focused explicitly on fairness work:

- "What helped us detect bias issues early?"
- "Where did we miss potential fairness problems?"
- "How effectively did we implement fairness acceptance criteria?"

Structured Analysis Exercises:

- **Fairness Timeline**: Mapping when bias issues appeared and why.
- **Intersectionality Matrix**: Examining blind spots across demographic intersections.
- **Fairness Impediment Analysis**: Identifying systemic barriers to equity work.

Fairness Process Improvements: Targeted changes to fairness workflows; updates to fairness testing procedures; refinements to fairness documentation.

Fairness improvement cycles create intentional learning loops focused specifically on enhancing fairness practices: **Fairness Maturity Assessment** (periodically evaluating team fairness capabilities); **Practice Improvement Goals** (setting explicit targets for process enhancement); **Cross-Team Learning Sessions** (sharing fairness insights across product teams); **Fairness Experimentation** (testing new approaches to bias detection and mitigation).

These techniques refine fair user story creation by identifying pattern improvements; enhance sprint planning through better task identification; and improve daily execution by addressing fairness workflow barriers.

To assess successful implementation of fairness ceremonies and checkpoints, establish these metrics: bias detection timing (when issues are discovered); ceremony effectiveness (insights produced); stakeholder engagement (participation quality); and fairness capability growth (team-skills trend).

TABLE 10 · FAIRNESS CEREMONY & CHECKPOINT METRICS

Metric	Description	Data to Capture	Target Threshold (High-Risk)	Frequency
Bias Detection Timing	When fairness issues are detected during the ML lifecycle.	Stage of development (data prep, model dev, testing, deployment).	≥80% of issues detected before final testing phase.	Per sprint
Ceremony Effectiveness	Number of actionable fairness insights or improvements from ceremonies.	List of issues raised, actions logged, outcomes tracked.	≥3 actionable fairness improvements per retrospective.	Every retro
Stakeholder Engagement	Participation and quality of stakeholder feedback on fairness.	Attendance records, survey feedback, qualitative input.	Active participation and trade-off understanding in feedback.	Each review
Fairness Capability Growth	Evidence of growing team skills in fairness design and evaluation.	Training logs, process changes, knowledge repo updates.	Measurable improvement in fairness practices and literacy each quarter.	Quarterly
Documentation Completeness	Clarity on data/model use, limitations and fairness impact.	Checklists, model cards, bias logs.	All fairness checkpoints documented per Vethman et al. (2025).	Each major milestone

5 Role-specific Fairness Responsibilities

Traditional agile roles — Product Owner, Scrum Master, Developer — have clear functional responsibilities but lack explicit fairness accountabilities. This responsibility gap creates situations where everyone assumes someone else owns fairness outcomes, leading to diffusion of responsibility.

Fairness-enhanced roles establish specific accountabilities. The full RACI matrix (Table 11) appears overleaf.

- **Product Owner** — Accountable for prioritising fairness as a feature; responsible for ensuring fairness is balanced with business needs; works closely with UX and legal to translate fairness risks into backlog items.
- **Scrum Master** — Accountable for facilitating fairness blockers and team conversations; helps normalise fairness as a topic during standups and retros; ensures psychological safety for raising fairness concerns.
- **Developers** — Responsible for integrating fairness checks into the CI/CD pipeline; implements bias detection code and flagging logic as part of PRs.
- **Data Scientists** — Accountable and responsible for data bias audits and fairness metric selection; translate insights into actionable fairness interventions.
- **AI/ML Engineers** — Support Developers and Data Scientists in implementing mitigation strategies; validate model behaviour under adversarial or biased scenarios.
- **UX Designers** — Ensure that fairness principles extend to user experience (e.g., accessibility, explainability); help anticipate potential user-facing fairness concerns.
- **Legal/Ethics Advisors** — Accountable for regulatory compliance and internal policy alignment; consulted during key architectural or deployment decisions.
- **Executive Sponsor** — Sets the tone and commitment to fairness culture; provides escalation support when fairness principles clash with business pressures. See **Appendix 1** on the distinction of the Executive Sponsor and the Product Owner.

Below is a matrix that assigns Responsible (R), Accountable (A), Consulted (C) and Informed (I) roles across key fairness-related activities in the AI/ML development lifecycle.

TABLE 11 · FAIRNESS RACI MATRIX ACROSS ROLES

Fairness Activity	PO	SM	Dev	DS	ML Eng	UX	Legal/Ethics	Exec Sponsor
Prioritise fairness requirements during backlog grooming	A	C	I	C	C	C	C	I
Assess fairness impact during tradeoff discussions	A	R	C	C	C	C	C	I
Facilitate fairness discussions during agile ceremonies	C	A	I	I	I	I	I	I
Track and escalate fairness-related blockers	I	A	I	I	I	I	I	I
Implement fairness tests alongside feature code	I	I	A/R	C	C	I	I	I
Apply fairness mitigations during development	I	I	A/R	C	C	I	I	I
Analyse model bias and recommend interventions	I	I	C	A/R	C	I	C	I
Document fairness assumptions, risks and decisions	R	R	C	A	C	C	A	I
Conduct fairness reviews before deployment	R	C	C	A/R	C	C	A	I
Champion fairness culture across the team	A/R	R	R	R	R	R	R	A

Legend. R (Responsible): Role(s) that do the work. A (Accountable): Role with decision authority and ownership of outcome. C (Consulted): Role(s) consulted before decisions or actions are taken. I (Informed): Role(s) kept up to date on progress or decisions.

"Champion fairness culture across the team."

— THE ONLY RACI ROW WHERE EVERY ROLE CARRIES A LETTER.

6 User Documentation: Applying the Toolkit

This section provides practical guidance on how Scrum teams can adopt the Fair AI Scrum Toolkit in real-world projects. It outlines step-by-step instructions, role-specific guidance and implementation timelines, making it easy to gradually embed fairness into agile workflows.

Step 1 · Identify Gaps in Your Current Process

Before applying new tools, understand where fairness considerations are missing today. Use this prompt set to guide an initial team discussion or self-assessment.

TABLE 12 · FAIRNESS GAP ASSESSMENT PROMPT

Assessment Prompt	Notes
Do our user stories mention any protected attributes or demographic groups?	
Have we ever rejected or reworked a feature due to fairness concerns?	
Does our Definition of Done include fairness checks or metrics?	
Are fairness blockers discussed during daily standups?	
Do our retrospectives ever mention fairness lessons learned?	
Are disaggregated performance results presented in reviews?	
Do stakeholders provide feedback on fairness trade-offs?	

Step 2 · Modify Core Artifacts

Goal: embed fairness into the building blocks of Scrum — stories, backlog and criteria.

TABLE 13 · MODIFY CORE ARTIFACTS

Step	What to Do	Resources
1 · Update User Stories	Use the SAFE framework to rewrite user stories with fairness dimensions.	§1.1 & SAFE Template
2 · Update Acceptance Criteria	Apply the FAIR framework to define measurable fairness thresholds.	§1.3 & Table 3
3 · Adjust Definition of Done	Extend DoD to include fairness validation steps, disaggregated testing and stakeholder sign-off.	Table 8 — DoD framework

Start small — try updating 1–2 new features in your backlog first.

Step 3 · Clarify Role Responsibilities

Goal: ensure everyone knows their fairness duties. Refer to the Fairness RACI Matrix (Table 11). Assign fairness champions or rotate the responsibility in early phases. Confirm who signs off on fairness at each stage (Product Owner vs. Executive Sponsor).

Step 4 · Integrate into Scrum Ceremonies

Goal: make fairness visible and actionable during ceremonies.

TABLE 14 · FAIRNESS INTEGRATION INTO SCRUM CEREMONIES

Ceremony	Fairness Addition	Resource
Sprint Planning	Allocate fairness capacity; score features by fairness risk.	Planning Agenda Templates
Daily Standup	Use fairness progress prompts; track blockers.	Fairness Standup Questions
Sprint Review	Present disaggregated metrics, fairness dashboards, mitigation demos.	Presentation Templates
Retrospective	Use fairness timeline, matrix analysis and retro prompts.	Fairness Retro Framework
Mid-Sprint Checkpoint	Conduct fairness checks on data, models, integration.	Checkpoint Formats

Use metrics (e.g., Bias Detection Timing, Blocker Resolution Time) to monitor effectiveness over time.

Step 5 · Pilot and Expand

TABLE 15 · HOW TO PILOT

Step	How to Pilot	Timeframe
1 · Choose one AI feature or project	Select high-risk area (e.g., hiring, healthcare).	Sprint 1
2 · Apply full fairness toolkit to this scope	Include updated stories, ceremonies and metrics.	Sprint 1
3 · Gather feedback	Use retro + 1:1s to capture insights.	End of Sprint 1
4 · Refine & expand	Adjust for team fit, scale to broader backlog.	Sprint 2+

TABLE 16 · TOOLKIT RESOURCES REFERENCE

Resource	Where to Find
SAFE User Story Template	§1.1 & Table 5
FAIR Acceptance Criteria	§1.3 & Table 3
Definition of Done	§3 & Table 8
Sprint Backlog Format	Table 2
Fairness Ceremony Metrics	Table 10
RACI Matrix	Table 11
Role Responsibilities	§5
Fairness User Stories Library	§2

Step 6 · Evaluation and Maturity

After a few cycles, assess how your team's fairness maturity is evolving using metrics e.g.: % of user stories with fairness elements; time to resolve fairness blockers; stakeholder understanding of fairness trade-offs. Use the Fairness Capability Growth and Bias Detection Timing metrics (Table 10) every quarter. Consider running a Fairness Maturity Assessment across teams to track adoption.

Validation targets (e.g., "Leak AUC ≤ 0.52", "mIoU gap ≤ 2%", "exposure parity gap ≤ 0.05") are useful but may look universal rather than derived from use-case risk, statistical power or baseline variance. Use the Story-to-Threshold template found in **Appendix 2** to tailor your fairness criterion.

7 Case Study: Applying the Fair AI Scrum Toolkit to a Resume Screening System

CONTEXT & CHALLENGE

A mid-sized tech company wants to build an AI system to automatically pre-screen job applications. Early prototypes showed a gender bias in the system: **women's resumes were 15% less likely to be advanced**. The team decides to follow the Fair AI Scrum Toolkit to eliminate bias, ensure accountability and build trust with stakeholders throughout the development process.

Phase 1 · Identify Gaps in Current Process

Goal: understand where existing Scrum practices miss fairness checks.

Assessment Prompt	Observations / Notes
Do our user stories mention protected attributes or fairness goals?	None mentioned — current stories are purely functional.
Does our Definition of Done include any fairness validation steps?	DoD only covers functional and performance tests.
Are fairness-related blockers ever raised in daily standups?	No explicit prompts; fairness issues hadn't surfaced.
Have we presented disaggregated performance in prior sprint reviews?	Only overall accuracy metrics shown.
Do retrospectives capture fairness lessons or suggested improvements?	Retrospectives focus on velocity and code quality.
How are stakeholders involved in fairness trade-offs?	Fairness wasn't highlighted to stakeholders.

Next step: use these findings to target the most urgent artifact and ceremony updates in the coming sprint.

Phase 2 · Modify Core Artifacts

2.1 Fairness-Enhanced User Story (SAFE Framework)

Field	Example
As a	Hiring manager
I want	The system to rank candidates by fit score ensuring equivalent score distributions across gender and ethnicity.
So that	I can see qualified candidates with no demographic group unfairly disadvantaged.
SAFE	**S**: Gender, ethnicity · **A**: Demographic parity · **F**: Scoring module · **E**: Score distribution variance < 5%.

2.2 FAIR Acceptance Criteria

FAIR Element	Example
F · Fairness Metrics	Selection rate disparity ≤ 0.05 across gender and ethnicity.
A · Auditing	Disaggregated performance metrics generated each sprint.
I · Intersectional	Performance gap across intersectional groups (e.g., women of colour) within 0.03.
R · Reporting	Fairness summary included in stakeholder review with mitigation actions.

2.3 Fairness-Enhanced Definition of Done

- Code passes functional and unit tests.
- Disaggregated performance reports produced and reviewed.
- Bias mitigation strategies (e.g., reweighing) applied if disparity exceeds threshold.
- Fairness Sign-off from Product Owner, ML Lead and Ethics Advisor obtained.
- Documentation updated (model card and fairness notes).

Phase 3 · Clarify Role Responsibilities

Using the Fairness RACI Matrix (Table 11):

- **Product Owner**: prioritises fairness in backlog; signs off on fairness criteria.
- **Scrum Master**: facilitates fairness blockers; ensures team discusses fairness in ceremonies.
- **Data Scientist**: leads bias audits; develops mitigation strategies.
- **Developer**: integrates bias detection into CI/CD; writes fairness tests.
- **UX Designer**: ensures interface explainability; tests for accessibility across user groups.
- **Legal/Ethics Advisor**: consulted for fairness threshold validation and final sign-off.
- **Executive Sponsor**: provides escalation support if fairness considerations conflict with delivery deadlines.

Phase 4 · Integrate into Scrum Ceremonies

4.1 Sprint Planning

The team commits 25% of capacity to fairness tasks (audits, mitigation implementation, fairness tests). Tasks scored using the Fairness Risk Scoring Framework — protected attribute coverage, severity of harm, regulatory risk.

4.2 Daily Standups (Fairness Prompts)

- What progress did you make on fairness-related work?
- Are there any fairness-related blockers?
- Any new bias risks identified?

Phase 4 · Ceremonies (cont.)

4.3 Sprint Review

Live demo presents disaggregated performance dashboard, mitigation strategy applied (reweighing, threshold tuning) and fairness debt remaining. Stakeholders provide feedback on fairness trade-offs, especially around hiring policy alignment.

4.4 Mid-Sprint Fairness Checkpoint

Pre-implementation fairness review on data pipeline — verifying balanced training data; integration test on candidate scoring model — checking parity across demographic groups; rollback or rework if metrics breach thresholds.

4.5 Retrospective

Fairness Timeline Exercise to map when bias was detected; Intersectionality Matrix Review identifies missed segments (e.g., gender × ethnicity); Action Items: improve test coverage on intersectional categories; expand mitigation tooling.

READING ACROSS PHASES

Phases 3 and 4 together describe how fairness work is *owned* and how it is *practised*. Without a named accountable role (Phase 3), the practices in Phase 4 quietly degrade — fairness prompts vanish from standups, mid-sprint checkpoints get skipped, and retrospective action items roll over indefinitely. The discipline of running both columns together is what turns a one-off audit into a habit of the squad.

Phase 5 · Pilot and Expand

Pilot project: resume screening tool. Sprint 1 delivered fairness-enhanced backlog and fairness audit. Sprint 2 introduced disaggregated metrics in dashboards. Sprint 3 onwards expanded to broader features (interview scheduling, offer recommendations) and replicated the toolkit for new AI projects.

Phase 6 · Evaluation and Maturity

Outcome metrics:

STORIES WITH FAIRNESS	BIAS DETECTION IN DEV	MEAN BLOCKER RESOLUTION	FAIRNESS DEBT CLOSED
100%	**90%**	**1.5d**	**75%**

The team plans quarterly fairness maturity reviews. Knowledge sharing across teams is initiated via Communities of Practice.

OUTCOME & IMPACT

- **Fairness Issues Detected and Mitigated Early:** bias discovered during data audit and mid-sprint testing rather than post-deployment.
- **Stakeholders Aligned on Fairness Trade-offs:** hiring managers, HR and ethics teams reviewed performance dashboards and contributed to acceptable thresholds.
- **Team Confidence Increased:** fairness conversations became normal, integrated into all phases.
- **Documentation Improved:** model cards, fairness reports and stakeholder feedback ensured a clear audit trail.
- **Foundation Set for Scaling Fairness Practices:** the toolkit was applied successfully to a critical AI feature; ready for organisation-wide rollout.

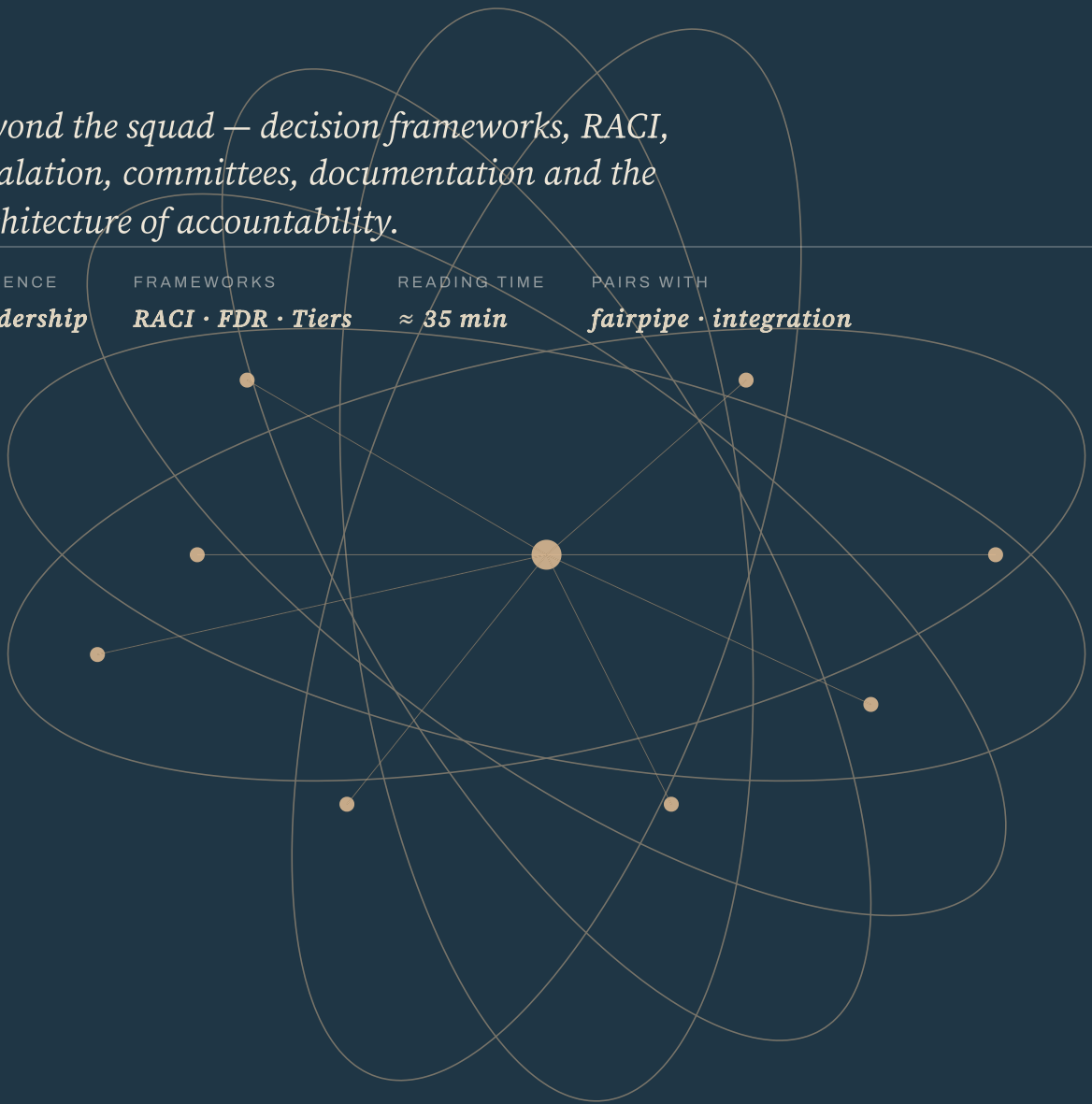
— End of Part One —

PART TWO · GOVERNANCE

Organizational Integration & Governance

Beyond the squad — decision frameworks, RACI, escalation, committees, documentation and the architecture of accountability.

AUDIENCE	FRAMEWORKS	READING TIME	PAIRS WITH
Leadership	***RACI · FDR · Tiers***	***≈ 35 min***	***fairpipe · integration***



PART TWO · INTRODUCTION

From individual teams to enduring institutional practice.

Embedding fairness in AI systems requires more than tools and frameworks at the team level — it demands organisation-wide structures, processes and accountability. The Organizational Integration Toolkit provides the scaffolding to ensure fairness becomes a sustainable institutional practice. It is designed to support governance, cross-team coordination and consistent decision-making at scale.

This toolkit equips organisations with practical components: clearly defined decision-making authority levels for various fairness scenarios; structured escalation procedures for handling complex fairness concerns; team-level governance models that promote ownership and consistency; integration with corporate governance bodies, ensuring alignment with broader strategic and ethical priorities; and documentation and transparency standards that establish trust both internally and externally.

Rather than relying on ad hoc decisions or isolated efforts, the toolkit provides a comprehensive framework for embedding fairness into organisational DNA. It defines who has the authority to make critical fairness decisions and the steps for escalating concerns when necessary. By doing so, organisations can ensure that fairness considerations are systematically addressed across all levels.

Fairness Decision Frameworks

Effective fairness implementation requires structured decision-making frameworks that define clear authority, escalation paths and accountability. Without explicit decision frameworks, fairness decisions often default to those most willing to make them, regardless of expertise or authority — leading to inconsistent outcomes and accountability gaps. The framework that follows defines three decision tiers, a RACI matrix and the escalation procedures that bind them.

A SIMPLE TEST

If a single PR can fix it, it's Tier 1. If two teams disagree, it's Tier 2. If counsel or the board need to know, it's Tier 3.

IN FAIRPIPE

Tier-3 gates are encoded as `required_approvers` in the YAML `governance:` stanza (Part 5).

Decision Tiers and Authority Levels

Fairness decisions vary significantly in their impact, complexity and stakeholder involvement. A multi-tier decision framework establishes appropriate authority levels for different decision types — and, crucially, names the body or role with final say at each level so a clear path forward exists even when the right answer is contested.

TIER 1 · TEAM-LEVEL DECISIONS

Implementation choices within established fairness frameworks. Examples: selecting specific bias mitigation techniques; calibrating fairness thresholds within approved ranges; determining testing approaches for fairness verification. **Authority:** development team with input from team-level fairness specialists.

TIER 2 · CROSS-FUNCTIONAL DECISIONS

Choices affecting multiple teams or requiring specialised expertise. Examples: fairness trade-offs between competing definitions; significant changes to fairness thresholds; novel fairness challenges without established protocols. **Authority:** cross-functional fairness review board with representation from technical, legal, ethics and business stakeholders.

TIER 3 · EXECUTIVE-LEVEL DECISIONS

Decisions with broad organisational, ethical or regulatory implications. Examples: deployment decisions for high-risk AI systems; organisational fairness policies and standards; resource allocation for fairness initiatives. **Authority:** executive committee with input from cross-functional fairness review board.

AVOID THE SILENT VETO

The most common failure mode is fairness specialists being "consulted" but never "accountable". Use the RACI overleaf to assign exactly one A per decision.

TIERS ARE NOT SENIORITY

A Tier-3 decision is one whose *consequences* are organisation-wide; not necessarily one made by the most senior person. A senior engineer often makes Tier-1 calls; a working group may own Tier-3 ones.

DEFAULT ESCALATE

When in doubt, escalate one tier up. The cost of an unnecessary review is small; the cost of an unflagged systemic harm is not.

RACI Framework for Fairness Decisions

A clear RACI framework prevents the diffusion of responsibility that often plagues fairness initiatives. The framework specifies for each fairness decision type:

- **Responsible**: who performs the analysis and implements decisions.
- **Accountable**: who has final decision authority and is answerable for outcomes.
- **Consulted**: who provides input and expertise to inform decisions.
- **Informed**: who needs to know about decisions and their implications.

TABLE 17 · RACI MATRIX FOR FAIRNESS DECISIONS

Decision Type	Responsible	Accountable	Consulted	Informed
Mitigation technique selection	ML Engineer	Tech Lead	Fairness Specialist	Product Owner
Fairness threshold adjustment	Fairness Specialist	Product Owner	Domain Expert, Legal	Stakeholders
High-stakes deployment approval	Fairness Review Board	Executive Sponsor	Affected Communities	Compliance, PR

Escalation Procedures

Effective escalation procedures provide clear paths for raising fairness concerns when they exceed normal decision authority or require additional resources/expertise.

Triggers for Escalation: identified bias exceeding established thresholds; conflicts between competing fairness definitions; novel fairness challenges without established protocols; resource constraints preventing adequate fairness work; stakeholder concerns about fairness implications.

Escalation Pathways:

1. Initial team review with fairness specialist.
2. Cross-functional fairness review board assessment.
3. Executive committee review for organisation-wide implications.
4. External expert consultation for novel or high-impact issues.

Documentation Requirements: Each escalation requires documentation including: nature of the fairness issue and potential impact; analysis performed and findings; options considered and rationale for selected approach; and decisions made and supporting evidence.

The escalation framework includes time-bound expectations for response and resolution. Common standards include 24-hour acknowledgement of escalations, 5-day initial assessment, and 15-day resolution targets, with adjustments for complexity.

Governance Structures

Governance structures translate fairness principles into organisational reality through formal bodies, processes and reporting relationships.

TEAM-LEVEL GOVERNANCE MODELS

Effective team-level governance balances embedded fairness expertise with broader organisational connections. Several models have proven effective:

Embedded Specialist Model. A dedicated fairness specialist becomes a permanent team member, providing day-to-day fairness guidance while maintaining connections to broader fairness communities. Strengths: deep contextual knowledge, immediate availability, integrated workflow. Limitations: resource intensive, potential isolation from broader practices.

Distributed Champion Model. Existing team members receive specialised fairness training to serve as fairness champions alongside their primary roles. Strengths: scalability, integration with existing roles, broad capability development. Limitations: divided attention, varying expertise levels, potential for inconsistency.

Centre of Excellence Model. Centralised fairness experts support multiple teams through consultations, reviews and training. Strengths: expertise concentration, consistent standards, efficient resource use. Limitations: limited team integration, potential bottlenecks, contextual knowledge gaps.

Hybrid Approach. A combination of embedded specialists for high-risk projects, distributed champions for general teams, and a Centre of Excellence for specialised expertise and standards.

INTEGRATION WITH CORPORATE GOVERNANCE

Effective fairness governance connects technical decisions to broader organisational governance: ethics committee integration; risk management connection; legal and compliance integration; board-level visibility for high-impact AI systems.

Intersectionality Consideration in Decision Frameworks

Decision frameworks must explicitly address intersectional fairness — recognising that bias often operates at the intersection of multiple protected attributes. To incorporate intersectionality:

- Specify intersectional categories (not just individual attributes) in decision criteria.
- Include perspectives from intersectional groups in advisory functions.
- Develop measurement approaches that disaggregate performance by intersectional categories.
- Establish escalation triggers specifically for intersectional fairness issues.

Implementation Approach

To establish effective fairness decision frameworks: define decision categories and authority levels (mapping decision types, identifying expertise requirements, establishing tier criteria); develop role-specific responsibility matrices (creating RACI frameworks, documenting expectations, ensuring coverage); create explicit escalation procedures (documenting triggers, defining communication, time-bound expectations); establish governance structures appropriate for organisational scale (selecting team-level model, creating cross-functional bodies, integrating with corporate governance).

EVALUATION APPROACH

To assess decision framework effectiveness: *decision quality* (consistency and outcomes); *process efficiency* (decision timeliness); *stakeholder satisfaction* (perception of decision processes); *continuous improvement* (refinement based on outcomes).

TABLE 18 · FAIRNESS DECISION FRAMEWORKS EVALUATION

Metric	Definition	Recommended Threshold (High-Risk AI)
Decision Quality	Consistency and outcomes of fairness decisions across decision types and over time.	≥85% of fairness decisions reviewed annually demonstrate consistency with framework guidelines.
Process Efficiency	Time required to make and implement fairness decisions.	Mean time-to-decision < 10 days for standard cases; < 30 days for escalated/high-impact cases.
Stakeholder Satisfaction	Perception of fairness decision processes by impacted stakeholders.	Average satisfaction score ≥4 out of 5 in stakeholder surveys conducted at least annually.
Continuous Improvement	Evidence of framework refinement based on outcomes and learning.	≥2 documented framework revisions per year; ≥80% of recommendations implemented.

Documentation and Transparency Standards

Documentation and transparency standards translate fairness principles into actionable organisational practice. Without clear documentation requirements, even well-intentioned fairness work often produces inconsistent records that fail to support governance, audit and improvement needs.

FAIRNESS DECISION RECORDS

FDRs provide a structured approach to documenting key fairness decisions throughout the AI development lifecycle. Properly implemented FDRs serve multiple critical functions: institutional memory; accountability; consistency; learning; communication.

Effective FDRs include:

- **Decision Context**: project, feature or system context; decision type and significance; stakeholders involved or impacted; relationship to broader fairness strategy.
- **Decision Analysis**: fairness considerations identified; options considered with pros/cons; quantitative analyses performed; stakeholder input received.
- **Decision Outcome**: selected approach and rationale; expected outcomes and trade-offs; implementation requirements; verification mechanisms.
- **Decision Governance**: decision authority and process; approval signatures and timestamps; review and revision history; cross-references to related decisions.

FDR TEMPLATE LIBRARY

An effective FDR template library includes specialised formats for common decision types: dataset selection and preparation; algorithm selection and configuration; threshold and parameter setting; deployment and monitoring decisions; incident response and remediation.

TABLE 19 · FAIRNESS DECISION RECORD TEMPLATE (ILLUSTRATIVE)

Field	Content
FDR-ID	FDR-2026-0114
Date / Author	2026-04-12 / J. Patel (Fairness Specialist)
Project / System	Resume Screening System (RS-01) · v0.6.5
Decision Type	Threshold & parameter setting
Decision	Adopt demographic parity gap of ≤ 0.05 across gender; ≤ 0.07 across gender × ethnicity intersections.
Options Considered	(a) 0.10 across single attributes only; (b) 0.05 single + 0.07 intersectional [chosen]; (c) strict 0.05 intersectional.
Analyses	Power analysis (Appendix 2 worked example); historical disparate impact baseline; legal counsel risk note.
Authority Tier	Tier 2 — cross-functional; approved by Fairness Review Board.
Approvers	Product Owner, ML Lead, Ethics Advisor, Legal Counsel
Linked Records	Model Card RS-01 v0.6.5 · DPIA-2026-008 · FAF-Audit-RS-01-Q2
Review Cadence	Quarterly with each model retraining.

FAIRNESS REQUIREMENTS DOCUMENTATION

Comprehensive fairness requirements documentation goes beyond traditional functional specifications to capture the unique aspects of fairness considerations: explicit fairness goals tied to specific use cases; protected attributes relevant to the application; fairness definitions appropriate for the context; quantitative thresholds and acceptance criteria; intersectional considerations; verification approaches.

MODEL CARDS AND DOCUMENTATION TEMPLATES

Model cards provide standardised documentation for AI models, including critical fairness information: model details & intended use; training data characteristics; performance metrics across demographic groups; ethical considerations & limitations; usage recommendations & warnings.

For comprehensive coverage, an organisation's documentation library should include templates for: model cards (model-level documentation); data sheets (dataset-level documentation); system cards (broader system context); use case cards (application-specific guidance); and incident reports (issue documentation).

TABLE 20 · FAIRNESS-AWARE MODEL CARD TEMPLATE

Section	Content
Model Details	Name, version, owner, date, contact, license, intended use.
Intended Use & Out-of-Scope	Primary use cases; users; deployment context; uses explicitly out of scope.
Training Data	Sources, sampling strategy, demographic distribution, known biases, time range, licensing.
Evaluation Data	Sources, distribution differences vs training data, demographic coverage.
Quantitative Analysis	Disaggregated metrics across protected attributes and intersections, with confidence intervals.
Fairness Definitions Used	Demographic parity, equalised odds, calibration etc., with rationale per use case.
Mitigations Applied	Pre-, in-, post-processing techniques; thresholds; fallback policies.
Ethical Considerations	Risks, harm scenarios, mitigations, residual risk.
Caveats & Recommendations	Known limitations, deployment guidance, monitoring requirements.
References	Linked FDRs, DPIA, FAF audits, regulatory filings.

AUDIT TRAIL ARCHITECTURE

TABLE 21 · AUDIT TRAIL ARCHITECTURE

Layer	Captured	Owner
Data lineage	Source, version, transformation, sampling, splits.	Data Engineering
Model lineage	Code commit, hyperparameters, training run, evaluation set, metrics.	ML Engineering
Fairness lineage	FAF audit version, metrics computed, thresholds, mitigations applied.	Fairness Specialist
Decision lineage	FDRs referenced, approvals, deviations.	Governance
Operational lineage	Deployment events, drift events, incidents, override actions.	SRE / MLOps

Audit trails must be immutable, time-stamped and queryable. fairpipe writes all five layers to a tamper-evident log (Part 5 §4) with cryptographic chaining suitable for regulator review.

Implementing the Governance Toolkit

Adopting the Organisational Integration Toolkit happens in five phases.

Phase 1 · Diagnose. Map current governance, decision authority and documentation against the toolkit. Identify Tier-1, Tier-2 and Tier-3 decisions that currently have no clear owner.

Phase 2 · Design. Constitute a Fairness Review Board (5–9 members covering ML, product, legal, ethics, UX and at least one impacted-community advisor). Draft the RACI matrix (Table 17) and escalation procedure. Choose a team-level governance model.

Phase 3 · Pilot. Apply on one high-risk system. Stand up the FDR repository. Begin issuing model cards and data sheets per Table 20.

Phase 4 · Scale. Roll out across all AI products. Connect to corporate governance — risk register, ethics committee, board reporting. Connect tooling: write FDRs from fairpipe runs (Part 5 §5).

Phase 5 · Mature. Establish quarterly governance reviews; publish a transparency report; benchmark against external standards (NIST AI RMF, ISO/IEC 42001).

"Without explicit decision frameworks, fairness decisions default to those most willing to make them, regardless of expertise or authority."

— PART TWO, OPENING

— End of Part Two —

PART THREE · ARCHITECTURE

Advanced Architecture *Cookbook*

Recipes for fairness in recommendation engines, large language models, computer vision, and multi-modal systems.

AUDIENCE	SUITES	READING TIME	PAIRS WITH
ML Engineers	***RecSys · LLM · CV · MM***	***≈ 25 min***	***fairpipe · training***

PART THREE · INTRODUCTION

Architecture-specific recipes for stubborn fairness problems.

Modern AI systems are not monolithic. They span recommendation engines, large language models, vision systems and multi-modal pipelines, each carrying distinctive fairness pitfalls that generic interventions cannot solve. This cookbook collects recipes — drawn from production work and the literature — that target each architecture's idiosyncrasies. Use them in conjunction with the SAFE/FAIR contracts of Part 1 and the governance frameworks of Part 2.

1 Recommendation Systems Suite

Recommendation systems exhibit unique fairness pitfalls because they jointly optimise for multiple stakeholders (users, items, providers) and create feedback loops that amplify even small initial biases over time.

COMMON PITFALLS

- **Popularity bias:** long-tail items receive disproportionately less exposure regardless of relevance.
- **Provider-side disparity:** creators belonging to under-represented groups receive less impression share for equally-matched items.
- **Exposure feedback loops:** ranked items receive more clicks; clicks feed training; training reinforces ranking — bias accelerates.
- **Cold-start unfairness:** new users or new items in protected categories receive systematically worse recommendations.
- **Filter bubbles & echo chambers:** user-side narrowing of content diversity, with disparate effects on minority preferences.

RECIPE · TWO-SIDED EXPOSURE AUDITING

1. **Define stakeholder cohorts** for both users and items/providers (gender, geography, language, creator-status, etc.).
2. **Compute exposure parity** — share of impressions / share of qualified candidates. Target ratio in `[0.95, 1.05]`.
3. **Measure utility parity** — disaggregated NDCG / hit rate per cohort. Gap target ≤ 3 pts.
4. **Mitigate** with fair re-ranking (e.g. FA*IR, DELTR), exposure-aware loss penalties, and counterfactual data augmentation for cold-start.
5. **Close the loop** — log served impressions to a fairness telemetry store and retrain on de-biased exposure data.

VALIDATION TARGETS

- Exposure parity gap ≤ 0.05 across protected cohorts.
- NDCG@10 disparity ≤ 0.03.
- Cold-start RMSE differential between minority and majority cohorts ≤ 10%.

> PAIRS WITH fairpipe `recipes.recsys` — see Part 5 §6.

2 Large Language Models Suite

LLMs introduce qualitatively different fairness risks — they encode social patterns at scale, generate open-ended content, and are evaluated against shifting cultural norms.

COMMON PITFALLS

- **Stereotypical associations** embedded in token co-occurrence.
- **Disparate refusal rates** across user cohorts and topics.
- **Quality gaps** in low-resource languages and dialects.
- **Toxicity amplification** when generating against under-represented groups.
- **Retrieval-augmented bias** — fairness issues in the underlying corpus surface as confident outputs.
- **Memorisation leakage** with disparate impact (e.g. memorised PII concentrated in some demographics).

RECIPE · MULTI-AXIS EVALUATION HARNESS

1. Assemble a fairness eval suite combining BBQ, BOLD, StereoSet, RealToxicityPrompts and use-case-specific red-team prompts.
2. Stratify by demographic axes (gender, race, religion, age, disability, sexual orientation, nationality).
3. Score on accuracy, refusal symmetry, sentiment differential, toxicity differential and stereotype score.
4. Report disaggregated AUC / pass-rate with bootstrapped CIs.
5. Mitigate with RLHF-from-diverse-annotators, instruction-level safety prompts, retrieval re-ranking, and selective fine-tuning on counterfactually balanced data.

VALIDATION TARGETS

- Refusal-rate disparity across protected cohorts ≤ 2%.
- Stereotype score (StereoSet) below model-class median.
- Memorisation leak AUC ≤ 0.52 (random-guess parity).
- Quality gap (factuality / fluency) across top 20 languages ≤ 5%.

> PAIRS WITH fairpipe `recipes.llm` — see Part 5 §6.

3 Vision Models Suite

Computer vision systems often inherit biases from data collection (lighting, lens type, geography), labelling (annotator demographics) and downstream interpretation.

COMMON PITFALLS

- Skin-tone disparities in face detection and verification.
- Geographic bias from web-scraped imagery skewing to high-income regions.
- Activity / object recognition gaps (e.g. cooking, household tasks gendered).
- Segmentation models with lower mIoU on darker-skin pixels or atypical body shapes.
- Synthetic / generative imagery defaults that reproduce stereotypes.

RECIPE · SKIN-TONE & GEO-STRATIFIED EVALUATION

1. Collect or license a stratified evaluation set (e.g. Casual Conversations v2, MIAP, Dollar Street).
2. Annotate Fitzpatrick skin-tone, region, body shape, lighting and camera class.
3. Compute disaggregated TPR, FPR, mIoU and calibration error.
4. Apply mitigations: stratified resampling, focal loss, synthetic minority augmentation, threshold tuning per cohort with explicit governance approval.

VALIDATION TARGETS

- TPR gap across Fitzpatrick I–VI ≤ 3%.
- mIoU gap across body-shape cohorts ≤ 2%.
- Geo-stratified FPR gap ≤ 2%.

4 Multi-Modal Systems Suite

Multi-modal systems compose risks from each modality and add cross-modal failure modes. The interaction surface is large and audit must span every modality and their joint behaviour.

COMMON PITFALLS

- Modality dominance — text overrides image evidence in VQA.
- Cross-modal stereotype amplification (e.g. caption generators reinforcing visual stereotypes).
- Audio-visual asymmetric performance for accented speakers in lip-sync or speaker-id systems.
- Inconsistent refusal across modalities (text-safe but image-unsafe outputs).

RECIPE · CROSS-MODAL FAIRNESS AUDIT

1. Generate cross-modal counterfactuals — same prompt with swapped demographic cues across modalities.
2. Measure consistency: identical answer rate; stereotype amplification index.
3. Apply modality-aware mitigations: balanced contrastive pairs, modality-dropout fairness regularisation, joint safety classifier.

VALIDATION TARGETS

- Cross-modal counterfactual answer-stability ≥ 95%.
- Stereotype amplification index ≤ 1.05 vs unimodal baseline.
- Refusal symmetry across modalities ≥ 98%.

> **PAIRS WITH** fairpipe `recipes.vision` and `recipes.multimodal`.

Cross-Domain Fairness Linkage Map

Many fairness primitives — disaggregated evaluation, counterfactual augmentation, calibrated thresholds, exposure auditing — recur across architectures. The linkage map below helps teams transport learnings.

TABLE 22 · CROSS-DOMAIN FAIRNESS PRIMITIVES

Primitive	RecSys	LLM	Vision	Multi-modal
Disaggregated metrics	NDCG, exposure	Pass-rate, toxicity	TPR, mIoU	Joint accuracy
Counterfactual augmentation	User-side swap	Demographic prompt swap	Skin-tone augmentation	Cross-modal swap
Calibrated thresholds	Per-cohort cut-offs	Refusal calibration	Per-cohort detection	Joint classifier calibration
Feedback-loop dampening	Exposure cap	RLHF re-balancing	Active learning re-sampling	Joint replay buffer
Drift & monitoring	Cohort CTR drift	Refusal drift	Coverage drift	Modality drift

Selecting and Adapting Recipes

Not every recipe is appropriate for every system. Before adopting a recipe:

1. Confirm the architecture-specific pitfall is plausible in your context (ground in user research, incident history and domain expertise).
2. Match the validation targets to the Story-to-Threshold worksheet (Appendix 2) — never copy thresholds blindly.
3. Estimate the statistical power of your evaluation set; if power is insufficient, expand sampling or relax thresholds with documented rationale (FDR).
4. Run the recipe end-to-end on a non-production model first; document outcomes; refine.
5. Codify in fairpipe (Part 5 §6) so every retraining triggers the same audit.

"Many fairness primitives recur across architectures — disaggregated evaluation, counterfactual augmentation, calibrated thresholds. Transport them deliberately."

— CROSS-DOMAIN LINKAGE MAP

— End of Part Three —

PART FOUR · COMPLIANCE

Regulatory *Compliance* Guide

Translating EU AI Act, GDPR, sectoral and local regulation into actionable controls, evidence trails, and the Total Risk Score.

AUDIENCE	FRAMEWORKS	READING TIME	PAIRS WITH
Counsel · DPO	***TRS · DPIA · AI Act***	***≈ 30 min***	***fairpipe · monitoring***

PART FOUR · INTRODUCTION

Translating regulation into controls and evidence.

This guide turns evolving global and local regulation into actionable steps for AI organisations. It covers five sequential moves: map requirements, classify risk, apply tier-based controls, build evidence and audit trails, and operate continuous monitoring. The output is a defensible compliance posture that survives both regulator and journalist scrutiny.

1 Step 1 · Map Global & Local Requirements

Begin by enumerating every regulatory regime that touches your AI system. The list typically includes:

- **EU AI Act** (Regulation 2024/1689) — risk-based regime with prohibited, high-risk, limited and minimal-risk categories; conformity assessments and post-market monitoring for high-risk systems.
- **GDPR** — lawful basis, automated decision-making (Art. 22), DPIA (Art. 35), data subject rights.
- **Sectoral regulation** — ECOA, FCRA, EEOC, FDA, MDR, EBA, MiFID II, NHS DTAC and equivalents.
- **Local statutes** — NYC Local Law 144 (AEDT), Illinois AIVI, California ADMT, Colorado AI Act, UK Equality Act.
- **Standards & soft law** — NIST AI RMF, ISO/IEC 42001, OECD AI Principles, Council of Europe AI Convention.

Maintain the resulting register as a living document. fairpipe ships a `compliance.register` object (Part 5 §5) that organisations can extend with jurisdiction-specific entries. **Appendix 6** provides a non-EU regulatory mapping for fast lookup.

TABLE 23 · CROSS-JURISDICTION QUICK MAP (EXCERPT)

Jurisdiction	Key Instrument	Trigger	Key Obligation
EU	AI Act	Placing high-risk system on EU market	Conformity assessment, registration, post-market monitoring.
EU	GDPR	Processing personal data in/from EU	DPIA, lawful basis, Art. 22 safeguards.
USA · Federal	Title VII / ECOA / FCRA	Employment, credit, consumer reports	Disparate impact testing, adverse-action notice.
USA · NYC	Local Law 144	AEDT used in NYC employment	Independent bias audit, candidate notice.
USA · CO	Colorado AI Act (SB 24-205)	High-risk consequential decisions	Risk management, impact assessments, notice.
UK	Equality Act 2010 + ICO guidance	Discrimination across protected characteristics	Indirect discrimination testing, DPIA equivalent.
Brazil	LGPD + PL 2338/2023	Personal data + AI	Algorithmic impact assessment, human review.
Canada	AIDA (proposed) + Directive on Automated Decision-Making	Federal automated decisions	Algorithmic Impact Assessment Tool (AIA).

2 Step 2 · Classify System Risk & the TRS Formula

Each AI system is classified by its **Total Risk Score (TRS)**, a weighted composite that informs which controls apply. TRS ranges 0–100 and bins into four tiers.

```
TRS = 0.30·Severity + 0.25·Scale + 0.20·Reversibility +
      0.15·Vulnerability + 0.10·Regulatory
```

Each input is scored 0–100 against rubrics:

- **Severity** — magnitude of potential harm (financial, physical, dignity, opportunity).
- **Scale** — number of decisions / people affected per year.
- **Reversibility** — ease of correcting an erroneous decision (lower reversibility → higher score).
- **Vulnerability** — degree to which decisions affect protected, minor or otherwise vulnerable populations.
- **Regulatory** — proximity to a regulated activity (high-risk under AI Act, sectoral law).

TABLE 24 · TRS TIERS AND CONSEQUENCES

Tier	TRS Band	Examples	Implication
T0 · Minimal	0–24	Spam classifier, internal autocomplete	Lightweight controls; standard model card.
T1 · Limited	25–49	Personalised marketing, search ranking	FAF audit annually; transparency note.
T2 · High	50–74	Employment screening, credit, insurance	Pre-deployment conformity pack; quarterly audits; human-in-loop.
T3 · Critical	75–100	Healthcare diagnostic, criminal-justice tools	External audit; regulator engagement; continuous monitoring; board sign-off.

Worked example: a CV-screening tool for hiring (Severity 70, Scale 60, Reversibility 65, Vulnerability 70, Regulatory 80) gives TRS = 21+15+13+10.5+8 = **67.5** → Tier T2.

fairpipe computes TRS automatically from the YAML `system:` block (Part 5). The resulting tier is stamped onto every audit artefact.

"TRS = 0.30·Severity + 0.25·Scale + 0.20·Reversibility + 0.15·Vulnerability + 0.10·Regulatory."

— THE COMPLIANCE ARITHMETIC OF PART FOUR

3 Step 3 · Apply Tier-Based Controls

Controls scale with tier. The full control set per tier is enumerated in **Appendix 7**; below are the headline obligations.

TABLE 25 · TIER-BASED CONTROL SUMMARY

Control	T0	T1	T2	T3
Model card	required	required	extended	extended + public
FAF audit	—	annual	quarterly	continuous
DPIA	—	conditional	required	required
Conformity assessment	—	—	required	external
Human oversight	—	recommended	required	required + appeal
Stage-gate sign-off	Tech lead	PO + ML lead	Review Board	Board + external
Stakeholder notice	—	privacy notice	specific notice	notice + opt-out
Post-market monitoring	incident-only	monthly	weekly	continuous

STAGE-GATE CHECKLIST (T2 EXAMPLE)

- Approved FDR for thresholds and mitigations.
- Disaggregated metrics across all required cohorts with CIs.
- Statistical power of evaluation set documented; ≥ 0.8 for primary metric.
- DPIA completed with DPO sign-off; residual risk accepted.
- Model card published in internal registry, version-pinned.
- Monitoring runbook including drift thresholds and escalation contacts.
- Incident response plan tested with table-top exercise in last 6 months.

4 Step 4 · Build Evidence & Audit Trails

The evidence pack is the artifact regulators and counsel actually read. It must be:

- **Complete** — covers data, model, fairness, decision and operational lineage (Table 21).
- **Traceable** — every claim links to an immutable record.
- **Reproducible** — given the same code and data version, the same metrics emerge.
- **Versioned** — every change is signed and timestamped.
- **Accessible** — readable by non-technical reviewers.

fairpipe (Part 5 §4–5) generates an *evidence bundle* per release, signed and stored in the audit trail with cryptographic chaining.

5 Step 5 · Continuous Monitoring & Review

Compliance is not a launch event. Continuous monitoring covers four loops:

1. **Data drift** — input distribution shifts, demographic representation changes.
2. **Performance drift** — disaggregated accuracy, calibration, fairness gap movement.
3. **Concept drift** — relationship between features and outcomes shifts.
4. **Stakeholder feedback** — appeals, complaints, qualitative reports.

Review cadence is anchored to TRS tier (monthly for T1, weekly for T2, continuous for T3). Each review cycle produces a fairness review note appended to the audit trail. Material drift triggers re-running stage gates.

Applying the Compliance Guide

Most organisations roll out the guide as follows:

1. *Quarter 1* — establish jurisdiction register; assign DPO and General Counsel hooks; classify the inventory of AI systems by TRS.
2. *Quarter 2* — adopt tier-based controls on the highest-tier systems; integrate fairpipe into CI/CD; build evidence pack template.
3. *Quarter 3* — pilot external audit on one Tier-3 system; validate audit trail completeness.
4. *Quarter 4* — publish first transparency report; pursue ISO/IEC 42001 readiness assessment.

> **PRE-LAUNCH** The full Pre-Launch Conformity Pack Checklist is in **Appendix 8**. Treat it as a single page sign-off; refuse launch on missing items.

— End of Part Four —

PART FIVE · TOOLCHAIN

The *fairpipe* Python Package

A reference and runnable examples for the installed package — `fairpipe` on PyPI, published by Svrus LLC under Apache-2.0, Python ≥ 3.10. Adjust paths and column names to your data.

PACKAGE	LICENSE	PYTHON	REFERENCE
fairpipe (PyPI)	***Apache-2.0***	***≥ 3.10***	***§0 — §10***

0 Package layout and installation

fairpipe is the PyPI distribution name. The installed wheel exposes two import roots: `fairpipe.*` (compatibility shims) and `fairness_pipeline_dev_toolkit.*` (full source layout). Examples throughout this section use `fairpipe.*` for shimmed areas — including `fairpipe.stats` and `fairpipe.api`. The `fairness_pipeline_dev_toolkit.*` names remain valid for the same objects (e.g. Uvicorn's import string for `fairpipe serve --reload`).

INSTALL

```
$ pip install fairpipe
```

OPTIONAL EXTRAS (FROM PYPROJECT.TOML)

TABLE 24 · INSTALL EXTRAS

Extra	Purpose
`training`	PyTorch, plotting — for `FairnessRegularizerLoss`, `LagrangianFairnessTrainer`, Pareto utilities.
`monitoring`	Streamlit, Dash, Plotly, Jinja2, PyWavelets — dashboards and drift tooling.
`adapters`	Fairlearn (and Aequitas on supported Python) — metric backends and `ReductionsWrapper`.
`api`	FastAPI stack for `fairpipe serve`.
`dev`	Test and lint tooling only.

```
$ pip install 'fairpipe[adapters]'
$ pip install 'fairpipe[training]'
$ pip install 'fairpipe[monitoring]'
$ pip install 'fairpipe[api]'
```

SHIM MODULES UNDER FAIRPIPE

`fairpipe.metrics`, `fairpipe.measurement` (facade mirroring `fairness_pipeline_dev_toolkit.measurement`), `fairpipe.pipeline`, `fairpipe.training`, `fairpipe.monitoring`, `fairpipe.integration`, `fairpipe.exceptions`, `fairpipe.stats` (e.g. `fairpipe.stats.bootstrap`, `fairpipe.stats.multipletests`), `fairpipe.api` (e.g. `fairpipe.api.app`).

The **root** package re-exports: `FairnessAnalyzer`, `MetricResult`, `to_markdown_report`, `log_fairness_metrics`, `assert_fairness`, plus `load_data` from I/O.

EQUIVALENT IMPORT PATHS

```
from fairpipe import FairnessAnalyzer, MetricResult, load_data, to_markdown_report, assert_fairness
from fairpipe.metrics import FairnessAnalyzer, MetricResult
from fairpipe.measurement import FairnessAnalyzer, assert_fairness, to_markdown_report
from fairpipe.pipeline import load_config, build_pipeline, apply_pipeline, run_detectors
from fairpipe.integration import execute_workflow, log_fairness_metrics
```

DATA LOADING

```
from fairpipe import load_data

df = load_data("data/holdout.csv")  # or .parquet / .pq
```

`fairness_pipeline_dev_toolkit.io.load_data` is the same function.

1 Metrics (measurement)

PRIMARY TYPES

- `fairpipe.metrics.FairnessAnalyzer` — orchestrates fairness metrics with optional intersectional grouping, `min_group_size`, and optional bootstrap confidence intervals.
- `fairpipe.metrics.FairnessAnalyzerDataFrameProxy` — returned by `FairnessAnalyzer.from_dataframe(...)`; binds column names so you call metric methods without passing arrays each time.
- `fairpipe.metrics.MetricResult` — dataclass: `metric`, `value`, optional `ci`, `effect_size`, `n_per_group`.

BACKENDS

Constructor argument `backend=None` (auto-select), or `"native"`, `"fairlearn"`, `"aequitas"`. Fairlearn / Aequitas require `pip install fairpipe[adapters]` where applicable.

METHODS ON FAIRNESSANALYZER

- `demographic_parity_difference(y_pred, sensitive, …)` — optional intersectional mode via `intersectional=True` with `attrs_df` / `columns`; optional bootstrap CI via `with_ci`, `ci_level`, `ci_samples`, `ci_method` (`"percentile"` or `"bca"` only); optional effect sizes.
- `equalized_odds_difference(y_true, y_pred, sensitive, …)` — same optional intersectional and CI / effect-size knobs.
- `mae_parity_difference(y_true, y_pred, sensitive, …)` — score-based regression-style parity; same CI options.

ARRAY API · SINGLE SENSITIVE COLUMN

```
from fairpipe import FairnessAnalyzer

analyzer = FairnessAnalyzer(min_group_size=30, backend="native")

dp = analyzer.demographic_parity_difference(
    y_pred=y_pred,
    sensitive=sensitive,
    with_ci=True,
    ci_level=0.95,
    ci_samples=1000,
    ci_method="percentile",  # or "bca"
    with_effect_size=True,
)
print(dp.value, dp.ci, dp.n_per_group)

eo = analyzer.equalized_odds_difference(
    y_true=y_true,
    y_pred=y_pred,
    sensitive=sensitive,
    with_ci=True,
    ci_method="bca",
)
```

Metrics — continued

DATAFRAME-BOUND API

```
from fairpipe import FairnessAnalyzer

proxy = FairnessAnalyzer.from_dataframe(
    df,
    y_pred_col="y_pred",
    sensitive_col="gender",
    y_true_col="y_true",
    min_group_size=30,
    backend="native",
)
dp = proxy.demographic_parity_difference(with_ci=True)
eo = proxy.equalized_odds_difference(with_ci=True)
```

INTERSECTIONAL COHORTS

```
analyzer = FairnessAnalyzer(min_group_size=50, backend="native")
attrs = df[["gender", "region"]]

eo = analyzer.equalized_odds_difference(
    y_true=df["y_true"].to_numpy(),
    y_pred=df["y_pred"].to_numpy(),
    sensitive=df["gender"].to_numpy(),  # unused when intersectional=True with attrs_df
    intersectional=True,
    attrs_df=attrs,
    columns=["gender", "region"],
    with_ci=True,
)
```

REPORTING AND TESTS

- `fairpipe.integration.to_markdown_report(results)` — Markdown table from a mapping of metric names to `MetricResult`-like objects or dicts.
- `fairpipe.integration.assert_fairness(value, threshold, comparator="⩽", allow_nan=False, context=None)` — pytest-style helper; raises `AssertionError` on breach (not a dedicated fairness exception type). `comparator` ∈ `"⩽"`, `"<"`, `"⩾"`, `">"`; any other string raises `ValueError`.

LOWER-LEVEL STATISTICS — FAIRPIPE.STATS

`fairpipe.stats.bootstrap.bootstrap_ci` (`method` = `"percentile"` or `"bca"`); `fairpipe.stats.multipletests.benjamini_hochberg`. These are ordinary helpers — *not* wired into a single `Metrics.compute`-style entry point.

`benjamini_hochberg(pvals)` returns `(adjusted_sorted, sorted_idx)`: adjusted p-values are in **ascending sorted p-value order**, not remapped to the original input order; `sorted_idx` is `numpy.argsort(pvals)` and aligns entry-wise with `adjusted_sorted`.

```
from fairpipe.stats.bootstrap import bootstrap_ci
from fairpipe.stats.multipletests import benjamini_hochberg
import numpy as np

x = np.random.randn(80)
low, high = bootstrap_ci(x, lambda v: float(np.mean(v)), B=500, level=0.95, method="bca")

adjusted_sorted, sorted_idx = benjamini_hochberg([0.01, 0.04, 0.10])
# Recover original order:
adjusted_orig = np.empty_like(adjusted_sorted)
adjusted_orig[sorted_idx] = adjusted_sorted
```

2 Pipeline configuration and transforms

LOADING CONFIG

- `fairpipe.pipeline.load_config(path=..., text=..., obj=..., profile=...)` → `PipelineConfig`.
- If `path` is omitted and `text` / `obj` are not given, discovery walks upward: `.fairpipe/config.yml` from cwd, then `~/.fairpipe/config.yml`, else error.
- YAML may define a top-level `profiles` map. Active profile: `profile` argument, env `FPDT_PROFILE`, test auto-detection, or the sole profile if only one exists. Selected profile is **shallow-merged** over top-level keys.

PIPELINECONFIG FIELDS (VALIDATED SUBSET)

TABLE 25 · PIPELINECONFIG FIELDS

Field	Role
`sensitive`	List of sensitive attribute column names (required).
`benchmarks`	Optional dict-of-dicts for detector benchmarks.
`alpha`	Float, default `0.05`.
`proxy_threshold`	Float, default `0.30`.
`report_out`	Optional path string for detector report output.
`pipeline`	List of steps (legacy key `steps` also accepted). Each step: `name`, `transformer` (string), `params` (dict).
`training`	Optional: `method` ∈ `reductions` | `regularized` | `lagrangian`, `target_column`, `params`.
`fairness_metric`	Optional string (disparity metric name for workflow validation).
`validation_threshold`	Optional float cap for that metric in the integrated workflow.

REGISTERED TRANSFORMER NAMES

Strings in YAML, mapped to sklearn-style components: `InstanceReweighting`, `DisparateImpactRemover`, `ReweighingTransformer`, `ProxyDropper`.

ORCHESTRATION API

- `fairpipe.pipeline.build_pipeline(config)` — build sklearn `Pipeline` from config.
- `fairpipe.pipeline.apply_pipeline(pipe, df)` — apply to a DataFrame.
- `fairpipe.pipeline.run_detectors(df, cfg)` — run bias detectors; returns a report object with JSON serialisation where implemented.

Pipeline — code examples

EXAMPLE-PIPELINE.YML

```yaml
sensitive: ["sensitive"]

pipeline:
  - name: reweigh
    transformer: "InstanceReweighting"
    params: {}
  - name: di
    transformer: "DisparateImpactRemover"
    params:
      sensitive: "sensitive"
      features: ["score"]
      repair_level: 0.8

alpha: 0.05
proxy_threshold: 0.30
```

PYTHON — LOAD, DETECT, APPLY

```python
from fairpipe import load_data
from fairpipe.pipeline import load_config, build_pipeline, apply_pipeline, run_detectors

cfg = load_config("example-pipeline.yml")
df = load_data("data.csv")

report = run_detectors(df=df, cfg=cfg)
pipe = build_pipeline(cfg)
Xt, _ = apply_pipeline(pipe, df)
```

PROFILES (SHALLOW MERGE)

```yaml
sensitive: ["s"]
alpha: 0.05

profiles:
  staging:
    pipeline:
      - name: only_di
        transformer: "DisparateImpactRemover"
        params: { sensitive: "s", features: ["x1"], repair_level: 0.7 }
```

```python
import os
os.environ["FPDT_PROFILE"] = "staging"
cfg = load_config("profiles.yml")
```

3 Training (optional dependencies)

EXPORTED FROM `fairpipe.training`

- `ReductionsWrapper` — sklearn classifier wrapping Fairlearn `ExponentiatedGradient`; `fit(X, y, sensitive_features=...)`; constructed with `base_estimator`, `constraint` (Fairlearn constraint object), `eps`, `T`, optional kwargs (`max_iter` defaulted for the wrapper). Requires Fairlearn via `fairpipe[adapters]`.
- `FairnessRegularizerLoss` — PyTorch loss augmentation (requires `fairpipe[training]`).
- `LagrangianFairnessTrainer` — PyTorch trainer with fairness constraint `demographic_parity` or `equal_opportunity` (requires `fairpipe[training]`).
- `GroupFairnessCalibrator` — group-wise calibration helper.
- `sweep_pareto`, `plot_pareto` — Pareto sweep / plot for regularised training demos.

No separate exported class named `FairClassifier`, `FairLossWrapper`, or `Reweighing` under `fairpipe.training`; Fairlearn-reductions mitigation is `ReductionsWrapper`. Instance reweighting / disparate-impact repair are pipeline transformers (§2).

FAIRLEARN REDUCTIONS — REDUCTIONSWRAPPER

```
from sklearn.linear_model import LogisticRegression
from fairlearn.reductions import EqualizedOdds
from fairpipe.training import ReductionsWrapper

model = ReductionsWrapper(
    LogisticRegression(max_iter=1000),
    EqualizedOdds(),
    eps=0.01,
    T=50,
).fit(X, y, sensitive_features=sensitive_features)
pred = model.predict(X_test)
```

LAGRANGIAN TRAINER · GROUP CALIBRATION

```
from fairpipe.training import LagrangianFairnessTrainer, GroupFairnessCalibrator

trainer = LagrangianFairnessTrainer(
    model, fairness="demographic_parity",
    dp_tolerance=0.02, model_lr=1e-3, lambda_lr=1e-2, device="cpu",
)
history = trainer.fit(X, y, s, epochs=5, batch_size=64, verbose=True)

cal = GroupFairnessCalibrator(method="platt", min_samples=20).fit(scores, labels, groups)
calibrated = cal.transform(scores, groups)
```

The CLI for the Lagrangian trainer reads CSV columns `f0..f{d-1}`, `y`, `s`. Group calibration CLI expects columns `score`, `y`, `g`.

4 Monitoring (optional dependencies)

EXPORTED FROM `FAIRPIPE.MONITORING`

- `RealTimeFairnessTracker`, `TrackerConfig`, `ColumnMap`
- `FairnessDriftAndAlertEngine`, `DriftConfig`, `MonitoringSettings`, `AlertEvent`
- `FairnessReportingDashboard`, `ReportConfig`
- `FairnessABTestAnalyzer`

`FairnessDriftAndAlertEngine` compares recent vs. reference windows (e.g. KS-based scoring on metric time series), optionally with wavelet decomposition when PyWavelets is available and configured. The module does **not** implement external incident integrations (PagerDuty, Slack, etc.); it emits in-memory `AlertEvent` records for downstream use.

DRIFT ENGINE

```
import pandas as pd
from fairpipe.monitoring import (
    FairnessDriftAndAlertEngine, DriftConfig, MonitoringSettings,
)

rng = pd.date_range("2025-01-01", periods=80, freq="D")
rows = [{"timestamp": t, "metric": "DP_global", "group_key": "all",
         "value": 0.05 + 0.001 * i, "n": 1000}
        for i, t in enumerate(rng)]
metrics_ts = pd.DataFrame(rows)

engine = FairnessDriftAndAlertEngine(MonitoringSettings(drift=DriftConfig()))
alerts = engine.analyze(metrics_ts, window_points=12, ref_points=48)
for a in alerts:
    print(a.metric, a.severity, a.reason)
```

5 Integration and workflow

`fairpipe.integration.execute_workflow(config, df, output_dir=None, min_group_size=30, train_size=0.8)` → `WorkflowResult`. Runs the three-step flow: baseline measurement (when `training` + `fairness_metric` are set), transform + train, final validation against `validation_threshold`. Returns metrics dicts, `ValidationResult`, fitted `model`, `transformed_df`, `predictions`, and `artifacts`.

There is no `Governance` class, FDR generator, TRS calculator, evidence-pack exporter, or GRC connectors in this package. Feature columns for the workflow are all columns except the target and the listed sensitive columns.

Workflow — code

WORKFLOW.YML

```
sensitive: ["s"]
pipeline: []

training:
  method: reductions
  target_column: "y"
  params:
    constraint: demographic_parity
    eps: 0.02
    T: 30

fairness_metric: "demographic_parity_difference"
validation_threshold: 0.15
```

PYTHON

```
from fairpipe import load_data
from fairpipe.pipeline import load_config
from fairpipe.integration import execute_workflow

df = load_data("train.csv")  # must include feature columns, "y", "s"
cfg = load_config("workflow.yml")
result = execute_workflow(cfg, df, output_dir="artifacts/run1", min_group_size=30)

print(result.validation_result.passed, result.validation_result.message)
print(result.final_metrics)
```

CLI

```
export FAIRPIPE_CONFIG_PATH=workflow.yml
fairpipe run-pipeline --csv train.csv --output-dir artifacts/run1
```

MLFLOW LOGGING

```
from fairpipe.integration import log_fairness_metrics, to_markdown_report

results = {"demographic_parity_difference": dp_result}  # MetricResult or dict-like
log_fairness_metrics(
    results,
    prefix="fairness_",
    artifact_name="fairness_report.md",
    artifact_content=to_markdown_report(results),
)
```

`fairpipe run-pipeline` can log via `--mlflow-experiment` / `--mlflow-run-name` when MLflow is configured.

6 Exceptions

`fairpipe.exceptions` — hierarchy under `FairnessToolkitError`: `ConfigValidationError`, `MetricComputationError`, `PipelineExecutionError`, `TrainingError`, `DataValidationError`, `DependencyError`.

```
from fairpipe.exceptions import ConfigValidationError, FairnessToolkitError
```

7 Command-line interface

Entry point: `fairpipe` → `fairness_pipeline_dev_toolkit.cli.main:main`.

TABLE 26 · FAIRPIPE COMMANDS

Command	Purpose
`fairpipe version`	Print package version string.
`fairpipe validate`	Load CSV/Parquet via `load_data`; compute disparity metrics; print Markdown report; optional `--out`.
`fairpipe sample-check`	Placeholder: checks for `dev_sample.csv` in cwd; non-blocking.
`fairpipe pipeline`	Load YAML config (`--config` or `FAIRPIPE_CONFIG_PATH`); run detectors unless `--no-detectors`; build and apply pipeline; optional `--out-csv`, `--detector-json`, `--report-md`, `--profile`.
`fairpipe train-regularized`	Demo sweep with `sweep_pareto` on CSV with columns `f*`, `y`, `s`; writes JSON (and optional PNG).
`fairpipe train-lagrangian`	Train with `LagrangianFairnessTrainer`; `--fairness` = `demographic_parity` or `equal_opportunity`; writes JSON.
`fairpipe calibrate`	Group calibration: CSV columns `score`, `y`, `g`; `--method` = `platt` or `isotonic`.
`fairpipe run-pipeline`	Full `execute_workflow`; `--config` or `FAIRPIPE_CONFIG_PATH`; `--csv`; `--output-dir`; `--min-group-size`; `--train-size`; optional MLflow flags. Exit code **1** if validation fails or an exception occurs.
`fairpipe serve`	Start HTTP API (requires `fairpipe[api]`); `--host`, `--port`, `--reload`, `--workers`.

TABLE 27 · COMMON ENVIRONMENT VARIABLES

Variable	Role
`FAIRPIPE_CONFIG_PATH`	Default config path for `pipeline` / `run-pipeline` when `--config` omitted.
`FAIRPIPE_MIN_GROUP_SIZE`	Default minimum group size for `validate` / `run-pipeline`.
`FAIRPIPE_MLFLOW_EXPERIMENT`	Default MLflow experiment for `run-pipeline`.
`FAIRPIPE_LOG_LEVEL`, `FAIRPIPE_LOG_FILE`, `FAIRPIPE_JSON_LOGS`	CLI logging.
`FPDT_PROFILE`	Default YAML profile name when configs use `profiles`.

Not in the current CLI: `fairpipe init`, `fairpipe config validate`, nested `fairpipe pipeline run`, `fairpipe audit *`, `fairpipe recipes *`, `fairpipe doctor`.

Shell examples

`validate` **flags (representative):** `--csv`, `--y-true`, exactly one of `--y-pred` or `--score`, `--sensitive` (one or more), `--min-group-size`, `--backend` (`auto|native|fairlearn|aequitas`), `--out`, `--with-ci`, `--ci-level`, `--bootstrap-B`, `--with-effects`.

```
fairpipe version

fairpipe validate \
  --csv data.csv \
  --y-true y_true \
  --y-pred y_pred \
  --sensitive gender \
  --with-ci --out report.md

fairpipe pipeline --config pipeline.config.yml --csv data.csv --out-csv out.csv

fairpipe run-pipeline --config workflow.yml --csv data.csv --output-dir artifacts/

fairpipe train-regularized --csv nn_demo.csv --out-json pareto.json --out-png pareto.png

fairpipe train-lagrangian --csv nn_demo.csv --fairness demographic_parity --out-json lag.json
```

8 Optional REST API — `fairpipe[api]`

Import: `from fairpipe.api import create_app, ResultStore` (same objects as `fairness_pipeline_dev_toolkit.api`). For app-only imports: `from fairpipe.api.app import create_app`.

`create_app()` returns a FastAPI app with routers for **health**, **validate**, **pipeline**, and **workflow** (implementation under `fairness_pipeline_dev_toolkit/api/routes/`). `fairpipe serve` runs this app via Uvicorn. With `--reload`, the CLI passes `fairness_pipeline_dev_toolkit.api.app:create_app` with `factory=True` (hardcoded). For your own Uvicorn invocations you can import `create_app` from `fairpipe.api` or `fairpipe.api.app` — separate from what `fairpipe serve --reload` uses.

```
pip install 'fairpipe[api]'
fairpipe serve --host 127.0.0.1 --port 8000
# http://127.0.0.1:8000/docs
```

```
from fairpipe.api import create_app
app = create_app()
```

9 Minimal configuration example

File name is arbitrary; `pipeline.config.yml` and `.fairpipe/config.yml` are common. Optional top-level `profiles` can supply alternate `pipeline` / `training` blocks per profile — merge rules are shallow over the base keys.

```yaml
sensitive: ["sensitive"]

benchmarks:
  sensitive:
    A: 0.40
    B: 0.40
    C: 0.20

alpha: 0.05
proxy_threshold: 0.30
report_out: "artifacts/pipeline_bias_report.json"

pipeline:
  - name: reweigh
    transformer: "InstanceReweighting"
    params: {}
  - name: di
    transformer: "DisparateImpactRemover"
    params:
      sensitive: "sensitive"
      features: ["score"]
      repair_level: 0.8

# Optional: integrated workflow training
training:
  method: reductions   # or: regularized | lagrangian
  target_column: "y"
  params: {}

fairness_metric: "demographic_parity_difference"
validation_threshold: 0.10
```

10 CI usage (accurate pattern)

A minimal GitHub Actions job can:

1. `pip install fairpipe` (and extras as needed).
2. Run `fairpipe validate` on a CSV with the required columns, or `fairpipe pipeline --config … --csv …`, or `fairpipe run-pipeline` when a full workflow config exists.
3. Use `assert_fairness` in pytest on metric values read from `FairnessAnalyzer` or workflow results.

README.md documents an optional composite action — `SvrusIO/fairpipe-action` — for validation in CI; it is **external** to the Python package and is **not** referenced from this repo's `.github/workflows/` (use the README snippet if you want that wiring).

— End of Part Five —

APPENDICES

Reference & Templates

Worked examples, ready-to-fork artefacts, glossary, training paths and bibliography to accompany the five parts of the Playbook.

APPENDICES	LENGTH	STATUS	LICENSE
1 — 8	*≈ 16 pages*	*Reference*	*CC BY 4.0*

A1 Worked Example — Resume Screening System

This appendix walks the full Playbook end-to-end for a single Tier-2 system: an ML résumé-screening tool used by an enterprise HR organisation. It binds together Parts 1–5 into a single narrative artefact that teams can fork.

CONTEXT

- **Use case.** Rank inbound applicants for a hiring funnel of ~120k applications/year across 8 job families.
- **Stakes.** Human reviewers see the top-N ranking; a low score deprioritises a candidate. Reversibility: low. TRS = 69 → Tier 2.
- **Protected attributes.** gender, ethnicity, age-band, disability status (self-reported, separately stored).
- **Regulatory.** EEOC, EU AI Act Annex III (employment) high-risk, GDPR Art. 22 considerations.

SPRINT 0 — SETUP

The squad scaffolds a fairpipe project (`fairpipe init resume-screening`), declares the cohorts in YAML, and stands up the FairnessAuditor agent in CI. Definition of Done is extended with "disaggregated metrics within threshold" and "FDR drafted for any release". RACI is filled per Part 2, Table 11.

SPRINT 4 — FIRST MEASUREMENT

Baseline audit shows a 7.4% demographic-parity gap between male and female applicants on Engineering roles, with the largest sub-cohort gap (12.1%) on women under 25. The breach trips a FairnessRegression event; the FDR ticket is auto-opened with the metric table attached.

SPRINT 6 — MITIGATION

The team applies in-processing `FairClassifier(constraint="equalized_odds", slack=0.03)` and post-hoc calibration via `CalibratedEqualizedOdds`. Re-audit shows DP gap dropping to 2.8% with a 0.4-point AUC cost — within agreed Pareto envelope.

RELEASE — GOVERNANCE

The Ethics Advisor and Legal Counsel co-sign the FDR; the model card publishes to the internal registry; an evidence bundle pushes to ServiceNow GRC mapped to AI-Act control AIA-9.

CONTINUOUS MONITORING

Weekly FairnessMonitor tick. After 11 weeks, a PSI shift on input features (post-graduation cycle) drives a cohort-AUC drop for the "career-changer" cohort. The monitor freezes release, pages on-call, opens a fresh FDR, and rolls back to v0.6.5.

A2 Templates Library

Every template referenced in the Playbook is reproduced in machine-readable form under the fairpipe templates registry (`fairpipe templates list`). The summary table below is the canonical index.

TABLE A2 · TEMPLATE INDEX

Template	Purpose	Format	Playbook ref
SAFE user story	Sprint backlog fairness story	Markdown · Jira	Part 1 §1.1
Fairness DoD checklist	Sprint Definition of Done addition	Markdown · GitHub PR check	Part 1 §3
RACI matrix	Role assignment for fairness work	CSV · YAML	Part 1 Table 11 · Part 2
FDR — Fairness Decision Record	Documented mitigation/acceptance choice	Markdown · JSON · PDF	Part 2 Table 19
Model card	System-level fairness disclosure	Markdown · JSON	Part 2 Table 20
DPIA cross-reference	Link privacy + fairness assessments	Markdown	Part 4 §3
Evidence pack manifest	Regulator-ready bundle index	JSON · ZIP	Part 4 §4 · Part 5 §5
Mitigation experiment plan	Designed-experiment protocol	Notebook · Markdown	Part 1 §4
Stakeholder consultation log	Affected-community engagement record	Markdown	Part 2 Governance
Recipe override file	Per-system threshold overrides	YAML	Part 5 §6

Use `fairpipe templates render <name> --out path/` to copy a template into a project, pre-filled from `fairpipe.yaml` .

A3 Decision Frameworks

Three reusable matrices to shortcut common fairness decisions. None replaces judgement — they make the trade-off explicit so judgement can focus on the contested cells.

A3.1 — METRIC SELECTION BY HARM PATTERN

TABLE A3.1

Harm pattern	Primary metric	Secondary metric	Caveat
Allocation of scarce good	Demographic parity	Calibration	Base rate variance can mask quality gaps
Quality of service	Equal opportunity	FPR/FNR parity	Requires reliable ground truth per cohort
Representation / visibility	Exposure parity	NDCG gap	Two-sided: protect both users and creators
Information access	Refusal-rate symmetry	Helpfulness disparity	Mind context-sensitive refusals
Misclassification harms	Group-conditional ECE	Reliability diagrams	Calibrate → threshold separately if needed

A3.2 — MITIGATION STAGE DECISION

TABLE A3.2

Constraint	Recommended stage	Typical primitive
Data quality variance dominates	Pre-processing	Reweighing, sampling, label audits
Inductive bias dominates	In-processing	Constraint-aware loss (DP, EO)
Cannot retrain (vendor model)	Post-processing	Calibrated equalised odds
Distributional shift	Continuous	Online reweighing + monitor

A3.3 — WHEN TO ESCALATE

TABLE A3.3

Trigger	Tier-1 path	Tier-2 path	Tier-3 path
New protected attribute proposed	Squad	Working group	Ethics board
Threshold relaxation	Squad + PO	Ethics advisor	Ethics board + Legal
Acceptance of a known disparity	Disallowed	Ethics board	Executive + Legal
Cross-system pattern	Working group	Working group	Ethics board

A4 Glossary

Allocational harm	Inequitable distribution of opportunities, resources or outcomes across cohorts.
Calibration	Predicted probabilities match observed frequencies; group-conditional calibration is the parity form.
Cohort	A subset of users/items defined by one or more protected attributes; intersectional cohorts cross multiple attributes.
Demographic parity (DP)	Equal positive prediction rates across cohorts; sensitive to base-rate variance.
Disparate impact	Substantially different outcomes for a protected group regardless of intent; US legal framing under Title VII.
DoD	Definition of Done — Scrum-team acceptance criteria; this Playbook adds fairness clauses.
Equal opportunity (EO)	Equal true-positive rates across cohorts — conditional on the positive class.
Equalised odds	Equal TPR and FPR across cohorts simultaneously.
FDR	Fairness Decision Record — a signed artefact documenting a fairness trade-off, mitigation or acceptance.
Fitzpatrick scale	Six-class skin-tone classification used in vision-model evaluations.
Intersectionality	Analytical lens for harms that emerge at the intersection of multiple attributes (e.g. Black women).
Mitigation	Technical or process change to reduce a measured disparity; classified by stage (pre/in/post).
PSI	Population Stability Index — distributional drift metric.
Representational harm	Stereotyping, demeaning or erasure of groups in model outputs.
SAFE	Specific, Actionable, Fair, Equitable — user-story heuristic in Part 1 §1.1.
TRS	Total Risk Score (Part 4 §2); aggregate of severity, scale, reversibility, vulnerability, regulatory exposure.
Two-sided fairness	Joint treatment of consumer and producer cohorts — used in marketplaces and content platforms.

A5 Training & Certification

A four-tier ladder for role-specific competency. Each level corresponds to a defined assessment (project + reviewed FDR + written exam) maintained by the central Responsible-AI Office.

TABLE A5 — CERTIFICATION LADDER

Level	Audience	Outcomes	Time-budget
L1 — Aware	All builders & PMs	Vocabulary, harms taxonomy, SAFE stories, when to escalate.	4 h e-learning
L2 — Practitioner	Engineers & data scientists	Metrics, mitigations, fairpipe basics, DoD usage.	3 days + project
L3 — Lead	Tech leads, MLOps, governance leads	Architecture cookbook, audit-log design, FDR review.	5 days + 2 reviewed FDRs
L4 — Advisor	Ethics advisors & boards	Regulatory landscape, intersectional theory, stakeholder engagement.	10 days + portfolio

CURRICULUM CROSS-REFERENCE

- **Module 1.** Fairness foundations — covers Part 1 §1 and Appendix 4. (L1–L4)
- **Module 2.** Engineering primitives — Part 1 §2–3 + Part 5 §1–3. (L2–L3)
- **Module 3.** Ceremonies & team practice — Part 1 §4–7. (L2)
- **Module 4.** Governance & documentation — Part 2. (L3–L4)
- **Module 5.** Architecture deep-dives — Part 3. (L3)
- **Module 6.** Regulatory craft — Part 4 + Appendix 8. (L4)
- **Module 7.** fairpipe in CI/CD — Part 5 §5–7. (L2–L3)

A6 Vendor & Tool Comparison

fairpipe is opinionated, but pluggable. The matrix below positions common open-source and vendor offerings against the four primitives we expose in Part 5. Use it during build-vs-buy assessments.

TABLE A6 — TOOL COMPARISON (STATUS AS OF 2026)

Tool	Measure	Mitigate	Monitor	Govern	Notes
fairpipe (this Playbook)	●	●	●	●	Reference impl; pluggable.
IBM AIF360	●	●	○	○	Strong primitives, no governance glue.
Microsoft Fairlearn	●	●	○	○	Excellent constraint optimisers.
Google What-If Tool	●	○	○	○	Notebook UI; not pipeline-native.
Aequitas	●	○	○	○	Strong audit-report exports.
Credo AI	◑	○	◑	●	Governance-first; commercial.
Holistic AI	◑	○	◑	●	Audit-as-a-service.
Arthur AI / Fiddler / WhyLabs	●	○	●	◑	Strong monitoring, lighter governance.
OneTrust / Drata / Vanta	○	○	○	●	GRC — receive evidence from fairpipe.

Key: ● first-class · ◑ partial · ○ add-on or absent.

INTEGRATION POSTURE

fairpipe deliberately delegates GRC, model registry and ticketing to specialist tools — we ship connectors, not replacements. Teams already invested in Credo AI or Fiddler can wire fairpipe alongside through the integration module; the only authoritative artefact remains the signed FDR.

A7 Selected Bibliography

A curated list — not exhaustive. Items chosen for direct applicability to the practices in Parts 1–5. Annotations are the Playbook editors', not the original authors'.

FOUNDATIONS

1. Barocas, S., Hardt, M., Narayanan, A. — *Fairness and Machine Learning*. fairmlbook.org. The standard reference for metric families and limits of formal definitions.
2. Crenshaw, K. — "Mapping the Margins" (1991). The intellectual root for the intersectional cohort analysis in Part 2.
3. Selbst, A. et al. — "Fairness and Abstraction in Sociotechnical Systems" (FAT*, 2019). Why purely algorithmic fixes fail.

METRICS & METHODS

1. Hardt, M., Price, E., Srebro, N. — "Equality of Opportunity in Supervised Learning" (NeurIPS, 2016).
2. Pleiss, G. et al. — "On Fairness and Calibration" (NeurIPS, 2017). Foundational impossibility result.
3. Mitchell, M. et al. — "Model Cards for Model Reporting" (FAT*, 2019). Basis for the Part 2 model-card template.
4. Gebru, T. et al. — "Datasheets for Datasets" (CACM, 2021).
5. Bender, E., Friedman, B. — "Data Statements for Natural Language Processing" (TACL, 2018).

ARCHITECTURES

1. Zehlike, M. et al. — "FA*IR: A Fair Top-k Ranking Algorithm" (CIKM, 2017).
2. Buolamwini, J., Gebru, T. — "Gender Shades" (FAT*, 2018). Vision-system audit methodology.
3. Parrish, A. et al. — "BBQ: A Hand-Built Bias Benchmark for QA" (ACL, 2022).
4. Nadeem, M. et al. — "StereoSet" (ACL, 2021).
5. Gehman, S. et al. — "RealToxicityPrompts" (EMNLP findings, 2020).

REGULATORY LANDSCAPE

1. European Parliament — *EU Artificial Intelligence Act*, OJ L, 12 July 2024. Annex III especially.
2. NIST AI 100-1 — *AI Risk Management Framework* (2023) and Generative AI Profile (2024).
3. EEOC — Technical Assistance Document on AI in employment selection (2023).
4. UK ICO — *Guidance on AI and data protection* (2023).
5. ISO/IEC 42001:2023 — AI Management Systems.

PRACTICE & CASE STUDIES

1. Madaio, M. et al. — "Co-Designing Checklists to Understand Organizational Challenges and Opportunities around Fairness in AI" (CHI, 2020).
2. Holstein, K. et al. — "Improving Fairness in ML Systems: What Do Industry Practitioners Need?" (CHI, 2019). Source for many of the team-level prescriptions in Part 1.
3. Raji, I. D. et al. — "Closing the AI Accountability Gap" (FAT*, 2020). Audit-trail design.

A8 Regulatory Mappings

A crosswalk between Playbook artefacts and the controls most often cited by regulators and auditors. Use as the index to your evidence pack.

TABLE A8 — CROSSWALK: ARTEFACT → REGULATORY CONTROL

Playbook artefact	EU AI Act	NIST AI RMF	ISO 42001	EEOC / sector
SAFE story + DoD	Art. 9 risk mgmt	Govern 1.5 / Map 3	6.1.2 planning	Process documentation
Cohort manifest	Art. 10 data & governance	Map 2.3	7.5 data	EEOC 4/5ths rule
Disaggregated metric report	Art. 15 accuracy / robustness	Measure 2.7 / 2.11	8.2.3	EEOC adverse-impact
FDR — Fairness Decision Record	Art. 9 ¶ 5, Art. 17	Govern 4.1	9.1	UK ICO accountability
Model card	Art. 13 transparency	Govern 2.2, Map 4.1	7.5.3	NYC LL 144 disclosure
Audit log (Merkle-chained)	Art. 12 logging	Manage 3.2	10.2	FFIEC SR 11-7 (model risk)
Continuous monitor	Art. 17, 61 post-market	Manage 2.1 / 4	10.1	Sectoral supervisory expectations
Evidence pack	Art. 11 technical documentation	Govern 1.6	7.5.2	Discovery production
Stakeholder consultation log	Art. 26 deployers	Map 5.1 / 5.2	4.2 interested parties	Title VII consultation
TRS tier classification	Annex III categorisation	Govern 1.3	6.1.2	Sector-specific tiering

NOTES ON USE

The crosswalk is informational — not legal advice. Local regulators (e.g. national DPAs implementing the AI Act, US states implementing employment-decision laws) issue additional guidance that should be tracked by the Responsible-AI Office and reflected back into `fairpipe.yaml` as overrides. The fairpipe `integration.export_evidence_pack` command emits this table as a JSON map alongside the bundle, so auditors can navigate by control rather than by artefact.

— End of Appendices —

COLOPHON

A note on the type

This Playbook is set in *Source Serif 4* for running text, with section openers and display in *Fraunces* at increased optical size. Captions, callouts and the running heads use *Inter*; code and configuration listings are *JetBrains Mono*.

The palette is anchored by a dark terracotta (#8a3520) and a warm paper (#f5f1ea), with secondary inks in deep slate and soft sand. Section openers are full-bleed and indexed by a single numeral; chapter markers in the gutter help an analogue reader scan for context.

CITATION

Svrus LLC. *Fairness Implementation Playbook — A methodology for deploying fairness solutions across agile teams.* v0.6.5, 2026. Includes fairpipe v0.6.5 reference implementation.

Fin.