

LUCID: A Fairlearn-Based Framework For Temporal Fairness Assessment And Root-Cause Explanation

Aashi Agarwal, Anushka Sharma, Ananya Verma, Vansh Gupta, Abdul Qadir

Department of Computer Science and Engineering (Data Science),
Inderprastha Engineering College, Ghaziabad, India

© The Author(s), under exclusive license to publication division, IPEC Journal of Science & Technology, 2026

Abstract: *Fairness in machine learning is commonly evaluated as a static property, even though deployed models operate on data whose population composition, feature distributions, outcome rates and decision processes can change over time. This creates a practical failure mode: a model can satisfy fairness criteria during an initial audit and later develop materially different group disparities without a corresponding monitoring mechanism. This review presents LUCID, a proposed Fairlearn-based framework for temporal fairness assessment and root-cause explanation. The framework extends the MetricFrame-style evaluation paradigm with temporal windowing, fairness-metric time series, statistically grounded drift detection using Page-Hinkley and ADWIN, confidence-interval gating, and mechanism-level attribution. Four candidate explanations are considered: covariate shift, label or concept shift, base-rate shift, and sampling artefact. A plain-language reporting layer is then used to communicate the affected group, fairness metric, magnitude of change, confidence and likely mechanism. The review positions this design against static fairness assessment, temporal fairness research represented by FairCanary, stream-drift detection methods, and feature-level explainability approaches. The central contribution is not a claim of being the first temporal fairness system, but an accessibility-oriented integration of temporal monitoring and mechanism-level explanation with an already adopted fairness toolkit. Because the synopsis defines LUCID as a proposed framework, this paper does not report fabricated experimental results. Instead, it specifies an evaluation methodology based on synthetic drift injection, detection lag, false-positive rate, attribution accuracy, integration overhead and a real timestamped case study where data are available.*

Keywords - *Algorithmic fairness, temporal fairness, Fairlearn, MetricFrame, concept drift, Page-Hinkley, ADWIN, root-cause attribution, explainable AI, continuous monitoring.*

I. INTRODUCTION

Machine learning systems increasingly participate in decisions with direct consequences for people, including loan approval, hiring shortlists, insurance pricing and college admissions. In such settings, model quality cannot be reduced to predictive accuracy. The distribution of errors and benefits across demographic groups is also relevant because a model may perform adequately on aggregate while producing systematically different outcomes for different groups. Algorithmic fairness therefore requires disaggregated evaluation and careful interpretation of fairness metrics.

Fairlearn provides an established practical interface for this task. Its MetricFrame evaluates selected metrics separately for groups defined by sensitive features, allowing practitioners to inspect group-specific performance and fairness-related disparities. The toolkit also provides mitigation algorithms such as Exponentiated Gradient and Threshold Optimizer. This makes Fairlearn a useful foundation for a temporal extension because the proposed framework can preserve the vocabulary and usage pattern already familiar to practitioners [1].

The limitation addressed by LUCID is temporal rather than purely static. A fairness audit performed on a dataset at deployment time implicitly assumes that the observed relationships remain sufficiently stable. Production data rarely satisfy this assumption indefinitely. Populations can change, feature distributions can move, outcome rates can change, and the relationship between features and labels can evolve. A fairness metric that was acceptable at one point can therefore change later, even if the model code itself has not changed. Continuous monitoring also creates a second problem. Not every fluctuation in a fairness metric should be

treated as a fairness failure. A small or sparse window can produce a large apparent disparity because of sampling variability. Conversely, a persistent change can be missed when a system relies on manual thresholds or occasional audits. LUCID therefore treats the fairness metric itself as a time series and applies stream-drift detection with confidence-interval gating.

A third problem is response. Detecting a fairness drift tells a practitioner that a metric has changed, but it does not automatically explain why. The proposed framework therefore distinguishes among covariate shift, label or concept shift, base-rate shift and sampling artefact. This mechanism-level view is intended to support different operational responses, such as retraining, reweighting, additional data collection, or ignoring an alert when evidence points to sampling noise.

This paper reviews the foundations and design of LUCID using the project synopsis as its source basis and follows the structure and compact two-column presentation of the IPECJST manuscript template. The work is positioned as a proposed framework and review of its methodological basis, not as a completed experimental study.

II. FOUNDATIONS OF FAIRNESS ASSESSMENT

Fairness is not represented by a single universally correct metric. Different definitions encode different requirements, and several notions of fairness can be mutually incompatible. Corbett-Davies and Goel emphasize that parity, calibration and related criteria must be interpreted in context rather than

treated as interchangeable numerical targets [3].

Mehrabi et al. similarly describe fairness as a broad family of concepts spanning group fairness, individual fairness, bias sources and evaluation choices [6].

For LUCID, this observation leads to a deliberate design choice: the framework does not introduce a new fairness definition. Instead, it operates on fairness metrics already available through the Fairlearn evaluation paradigm. Temporal monitoring is treated as an additional analytical dimension layered over existing metrics. This reduces the risk that temporal analysis changes the meaning of the fairness criterion being monitored.

The practical consequence is that the object being monitored is not simply the raw prediction stream. It is the sequence of fairness measurements obtained after the incoming records are partitioned into temporal windows. A fairness metric can then be represented as m_t for window t , producing a time series $m_1, m_2, \dots, m_T$. A temporal alert is meaningful only when the observed change is larger than what can reasonably be explained by sampling variation and normal temporal fluctuation.

Fairlearn's `MetricFrame` provides the core evaluation abstraction used in the proposed architecture [1]. Given predictions, true labels, sensitive features and a selected metric, the interface can compute aggregate and disaggregated values. The synopsis proposes preserving this calling convention while adding temporal handling around it. In conceptual form, each evaluation window W_t is processed through `MetricFrame`, producing group-specific metric values and associated sample counts.

This reuse matters for accessibility. A separate fairness monitoring platform would require practitioners to learn another API, another data representation and another reporting convention. LUCID instead aims to behave as a natural extension of the toolkit. The proposed pip-installable packaging, documentation and example notebooks are therefore part of the technical design rather than an afterthought.

The synopsis specifically identifies Demographic Parity Difference and Equalized Odds Difference as examples of fairness measures supported by Fairlearn. The temporal extension can monitor any compatible fairness metric that can be evaluated per window. The interpretation, however, changes when the metric is observed repeatedly. A single disparity value answers the question 'what is the disparity in this dataset?', whereas a temporal sequence also asks 'is the disparity changing, when did the change begin, and is the change statistically meaningful?'

This distinction is important because an absolute fairness value and a change in fairness are different signals. A system may have a stable but non-zero disparity, or it may experience a sudden deterioration from an earlier baseline. LUCID is primarily concerned with the second problem while retaining the first as contextual information in its reports.

III. TEMPORAL FAIRNESS AND CONCEPT DRIFT

A static fairness audit provides evidence only for the population represented by the evaluated dataset. In a deployed system, incoming records can differ from that population. The synopsis frames this as a temporal fairness problem: population composition may shift, feature distributions may change, and the statistical relationship between inputs and outcomes may drift. Consequently, fairness cannot be assumed to remain fixed merely because the model parameters remain unchanged.

The temporal issue is closely related to delayed impacts in machine learning systems. Liu et al. show that fairness interventions can produce delayed and unintended effects after deployment, demonstrating why a one-time evaluation is insufficient for understanding long-term behavior [10]. The implication for LUCID is operational: fairness assessment must become a continuing process that can observe change after deployment.

Continuous monitoring also creates a data-management requirement. Incoming records need timestamps and must be grouped into windows before fairness metrics are computed. The synopsis proposes fixed-size sliding windows, fixed-time windows such as daily or weekly windows, and adaptive windows that grow while the data are stable and shrink when change is detected. These strategies provide different trade-offs between temporal resolution, statistical stability and computational cost.

For each window, the proposed system stores per-group metric values, bootstrap confidence intervals and sample sizes in a time-indexed metric store. This retained history is essential because drift detection is performed on the fairness-metric stream rather than on a single evaluation. It also allows later reports to reconstruct the sequence leading to an alert.

Window choice affects the statistical behavior of the monitoring system. Very small windows provide rapid detection but increase variance, while very large windows provide more stable estimates but can dilute a short-lived change. The confidence-interval gating mechanism in LUCID is intended to reduce the risk that this trade-off produces false alarms.

Concept drift research provides the statistical foundations for monitoring changing streams. Page's continuous inspection scheme introduced a sequential change-detection idea that can identify persistent shifts in a monitored process [8]. Bifet and Gavaldà proposed ADWIN, an adaptive windowing method that changes the amount of historical data retained according to evidence of distributional change [7]. Gama et al. developed drift-detection approaches for learning systems operating under changing data conditions [9].

LUCID adapts this family of ideas to a different monitoring object: fairness metrics. Rather than asking only whether the raw input distribution has drifted, it asks whether the fairness

behavior measured by the deployed system has changed. This distinction matters because raw feature drift does not necessarily imply fairness drift, and fairness drift can arise from multiple mechanisms.

Window	Operation	Use	Issue
Sliding	Moving fixed-size sample	Fine-grained	Overlap
Fixed-time	Calendar interval	Daily/weekly	Variable N
Adaptive	Grows in stable periods	Changing streams	More control

Table I. Temporal windowing strategies proposed for LUCID.

IV. REVIEW OF RELATED WORK

Fairlearn established a practical toolkit for assessing and improving fairness of AI systems through disaggregated metrics and mitigation techniques [1]. Its design provides the implementation foundation for LUCID. The project does not replace Fairlearn's metric definitions; instead, it adds temporal storage, drift detection and explanation around repeated evaluations.

Hardt et al. formalized equality of opportunity and related equalized-odds ideas, providing a theoretical basis for group-conditional error constraints [4]. Dwork et al. developed fairness through awareness as an individual-level formulation [5]. These works demonstrate that fairness is fundamentally a specification problem. LUCID consequently treats metric selection as an input to monitoring rather than assuming one metric is universally sufficient.

Kamiran and Calders proposed discrimination-aware preprocessing techniques, illustrating how fairness interventions can operate before model training [11]. In the LUCID design, mitigation is optional and may be triggered after an attributed drift suggests model staleness. The synopsis specifically mentions Fairlearn's Exponentiated Gradient and Threshold Optimizer as potential existing mitigators rather than introducing a new mitigation algorithm.

FairCanary is identified in the synopsis as the most directly related prior work [2]. It demonstrates continuous explainable fairness monitoring using Quantile Demographic Drift and feature-level explanations based on per-prediction attribution techniques. Its relevance is significant because it establishes that fairness can be monitored as a changing property rather than a static score. The project synopsis identifies several limitations in FairCanary as the motivation for LUCID. First, FairCanary is described as companion research code rather than a

broadly packaged production-oriented toolkit. Second, its alerts rely on manually set thresholds rather than the proposed combination of statistical drift detection and confidence-interval gating. Third, its explanations focus on implicated input features, whereas LUCID aims to classify the underlying statistical mechanism. The distinction between feature-level and mechanism-level explanation is central. A feature may be associated with a change, but the operational response depends on what caused the distributional change. Covariate shift suggests a change in inputs; label or concept shift suggests a change in the relationship between inputs and outcomes; base-rate shift concerns outcome prevalence or group composition; and sampling artefact indicates insufficient or unstable observations.

Lundberg and Lee introduced a unified approach to interpreting model predictions using Shapley-based explanations [16], while Ribeiro et al. introduced LIME for local, model-agnostic explanations [15]. These approaches are valuable when the question is which features influence a prediction. LUCID uses this literature primarily as a comparison point because its explanation target is different. The proposed output is not a local explanation of one prediction; it is an explanation of why a group-level fairness metric changed over time.

Miroshnikov et al. further demonstrate the use of Shapley-based reasoning in a fairness-specific context through a Wasserstein-distance fairness interpretability framework [12]. Together, these works support the broader premise that fairness analysis benefits from explanation. LUCID extends the idea by treating the explanation target as a mechanism responsible for temporal change.

Sculley et al. describe the hidden technical debt that accumulates when machine learning systems are deployed without adequate engineering discipline [14]. This perspective is directly relevant to LUCID's packaging and integration objectives. A technically sophisticated fairness method has limited practical value if practitioners cannot install it, integrate it with an existing pipeline, understand its output, or reproduce its evaluation.

The proposed pip-installable extension, Fairlearn-like API, documentation and example notebooks therefore represent part of the research contribution. Accessibility is treated as an evaluation criterion alongside detection and attribution performance.

V. COMPARATIVE REVIEW AND RESEARCH GAP

The comparison exposes four related but incomplete capabilities. Fairlearn provides accessible fairness measurement but no built-in temporal monitoring. FairCanary demonstrates continuous explainable fairness but is described in the synopsis as research code with limited packaging and

without the proposed mechanism classification. General drift-detection methods provide statistical foundations but do not by themselves answer whether a fairness metric has deteriorated or why. Explainability methods can identify influential features but do not automatically identify the statistical mechanism behind a group-level fairness change. The resulting research gap is therefore an integration gap. The problem is not that temporal fairness, drift detection or explainability have never been studied. The problem is that these components are not combined into an accessible extension of an established fairness toolkit in the manner proposed by LUCID. The project claims novelty in this integration, in confidence-interval-gated fairness-stream detection, and in mechanism-level attribution.

A second gap concerns the actionability of alerts. An alert that says 'Demographic Parity Difference increased' is incomplete if the practitioner must independently determine whether the cause is a population shift, outcome-rate change, model staleness or inadequate sample size. LUCID attempts to narrow this response gap by attaching a likely mechanism and confidence information to the alert.

Approach	Time	Gate	Explanation	Integration
Fairlearn	Static	None	Metric/ group	Toolkit
FairCanary	Continuous	Manual	Feature- level	Prototype
Drift methods	Continuous	Statistical	Drift event	General stream
LUCID	Continuous	PH+AD WI N+CI	Mechanism	Fairlearn extension

VI. LUCID FRAMEWORK: PROPOSED METHODOLOGY

The proposed framework is organized as a six-stage pipeline: incoming data stream, windowing module, TemporalMetricFrame, drift detection layer, root-cause attribution and explanation/reporting. The architecture is deliberately centered on Fairlearn's MetricFrame so that temporal functionality surrounds a familiar fairness evaluation abstraction rather than replacing it.

The expected input consists of y_{true} , y_{pred} , sensitive_features and timestamp. The timestamp is used to order records and form evaluation windows. For each window, fairness metrics are computed, uncertainty information is retained, and the resulting observations are passed to the drift detector. Confirmed changes are then analyzed for their likely mechanism before a structured and plain-language report is produced.

The temporal data-handling layer accepts time-ordered records

and partitions them according to the selected windowing strategy. Fixed-size sliding windows are useful when a consistent number of observations is desired. Fixed-time windows are convenient for operational reporting because they map naturally to daily or weekly intervals. Adaptive windows are intended for streams in which stable periods should use more history while periods of detected change should receive finer temporal resolution.

Within each window, MetricFrame is evaluated with the chosen sensitive features and fairness metrics. The result includes per-group metric values, sample counts and bootstrap confidence intervals. The synopsis proposes storing these outputs in a temporal metric store using Parquet or SQLite. This stored representation supports subsequent drift detection and reporting without requiring the raw dataset to be repeatedly recomputed.

The drift layer monitors each fairness-metric time series rather than only the raw input variables. Two stream-oriented methods are proposed: Page-Hinkley and ADWIN. Page-Hinkley provides sequential detection of persistent changes in a monitored value [8], while ADWIN provides adaptive windowing for time-changing data [7]. River is identified in the synopsis as the implementation source for these streaming drift detectors.

A critical design component is confidence-interval gating. If a window contains few observations, the estimated fairness metric may vary substantially even when the underlying process has not changed. The framework therefore uses bootstrap confidence intervals derived from the window's sample size as a gate on drift alerts. Conceptually, a detector signal becomes actionable only when it is accompanied by sufficient statistical evidence from the window.

The evaluation should report detection lag and false-positive rate. Detection lag measures how long the system takes to identify an injected change after its known onset. False-positive rate measures how frequently the system signals a drift when no known drift was introduced. These measures are preferable to reporting only the number of alerts because they capture both responsiveness and reliability.

Once a drift is confirmed, LUCID evaluates four candidate mechanisms. Covariate shift is assessed through changes in feature distributions using tests such as the Kolmogorov-Smirnov test or Population Stability Index. Label or concept shift is examined through a lightweight probe model trained on recent data and compared against the baseline relationship. Base-rate shift is assessed through changes in group proportion or true outcome rate. Sampling artefact is considered when a group's window sample size drops sharply.

The proposed reweighting decomposition provides an additional mechanism-level check. Earlier-window observations are reweighted so that their covariate distribution resembles the later window. The fairness metric is then recomputed. If much of the observed metric change disappears after covariate alignment, the covariate-shift contribution is supported. The remaining difference can then

be considered in relation to label, base-rate and sampling mechanisms. This decomposition is not presented as a perfect causal identification method. It is a structured attribution procedure that narrows plausible explanations. The synopsis therefore specifies attribution accuracy as an empirical evaluation criterion using synthetic drift where the true injected mechanism is known.

A statistical alert is difficult to use if its output consists only of test statistics. LUCID converts the detection and attribution outputs into a structured report containing the affected group, monitored metric, magnitude of change, primary cause and confidence. A plain-language summary is then generated so that non-specialist practitioners can understand the event.

An example report could conceptually state that a selected sensitive group experienced an increase in a monitored disparity during a specified window, that the change exceeded the uncertainty expected from the window size, and that distributional evidence is most consistent with covariate shift. Such a report does not replace statistical evidence; it translates the evidence into an operational description.

The reporting layer also creates a natural boundary between detection and intervention. The system can identify and explain a drift without automatically modifying the model. The synopsis permits an optional mitigation trigger that may invoke existing Fairlearn mitigators when the attributed cause indicates model staleness.

The proposed framework is intended to be pip-installable and to mirror Fairlearn's existing API conventions. This means that sensitive_features should be supplied in a familiar form and that temporal evaluation should resemble MetricFrame usage rather than requiring a separate monitoring language. Documentation and example notebooks are part of the proposed deliverable.

Accessibility is especially important because fairness monitoring sits between data science, machine learning engineering and responsible-AI governance. A framework that requires substantial changes to an existing pipeline increases adoption cost. LUCID instead evaluates integration overhead through the lines of code required to add temporal monitoring to an existing Fairlearn pipeline.

VII. PROPOSED EVALUATION FRAMEWORK

The synopsis proposes Adult Income, COMPAS and German Credit as benchmark datasets. Because these datasets are not inherently timestamped streams in the project description, the evaluation creates synthetic temporal windows and injects known forms of drift. This allows the framework to be tested under controlled conditions in which the onset and mechanism of change are known.

The benchmark should include stable periods as well as deliberate drift points. Different injected mechanisms can be evaluated independently and in combination. The purpose is not to claim that synthetic drift reproduces every production scenario, but to provide ground truth for testing whether the detector and attribution engine behave as designed.

The primary baseline is a naive fixed-threshold approach. Under this baseline, a fairness metric is considered problematic when its value crosses a manually chosen threshold. This provides a simple reference against which the statistically gated drift detector can be compared. Where feasible, FairCanary's approach can also be included as a comparison point, particularly for temporal fairness monitoring and explanation behavior.

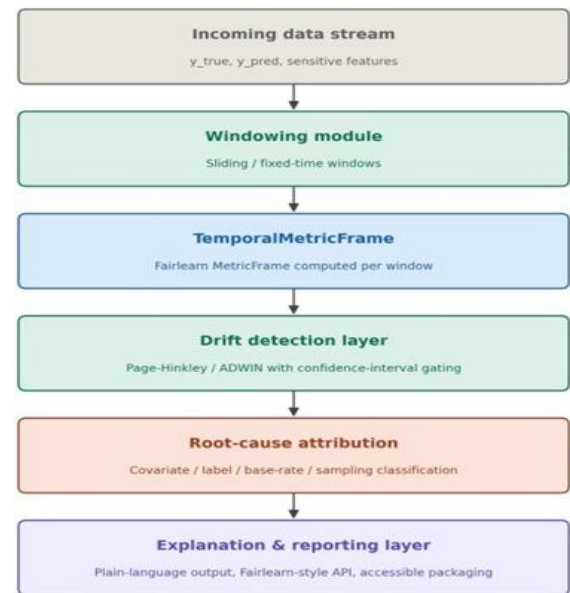


Fig. 1. Proposed LUCID architecture reproduced from the project synopsis. The pipeline moves from incoming records through temporal windowing and Fairlearn-based metric computation to statistical detection, root-cause attribution and explanation.

The comparison should focus on the dimensions defined in the synopsis rather than on a single aggregate score. A detector that identifies changes rapidly but generates many false alarms has a different operational profile from a slower but more conservative detector. Likewise, a mechanism classifier should be judged by attribution accuracy rather than by the number of explanations generated.

The synopsis proposes a case study using one real timestamped dataset, such as a loan or credit dataset, where available. The purpose is to demonstrate end-to-end operation under conditions that are not completely controlled by synthetic injection. The case study should report the chosen sensitive features, fairness metric, windowing strategy, detected events and explanations without presenting unverified claims about performance before the implementation is complete.

A real timestamped dataset is also important for evaluating operational concerns such as sparse groups, irregular arrival rates and changes in group composition. These issues are

difficult to reproduce completely through a simple synthetic benchmark and directly motivate the confidence-interval gating and adaptive-windowing components.

Dimension	Measure	Purpose
Detection	Detection lag	Response speed
Reliability	False-positive rate	Noise sensitivity
Attribution	Mechanism accuracy	Cause classification
Integration	Code/overhead	Adoption cost
Case study	End-to-end	Real timestamped data

Table III. Evaluation dimensions specified in the project synopsis.

VIII. TOOLS, TECHNOLOGIES AND IMPLEMENTATION CONSIDERATIONS

The implementation stack is deliberately Python-centric because Fairlearn itself is a Python toolkit and the proposed API is intended to preserve that ecosystem. pandas and NumPy support data manipulation, while scikit-learn supplies machine-learning utilities. River is proposed for stream drift detection and SciPy for distributional tests. Parquet or SQLite provides a lightweight temporal store, while Streamlit and Plotly can expose the monitoring results through an accessible dashboard. The engineering architecture should separate metric computation, detection, attribution and reporting. This separation makes it possible to evaluate each component independently and reduces the risk that a reporting decision changes the underlying statistical result. It also supports future replacement of one detector or attribution method without redesigning the complete framework.

The project should maintain a clear distinction between an observed statistical signal and an inferred explanation. A detected change can be statistically supported while the attribution remains uncertain. Reports should therefore retain confidence or evidence information rather than expressing every classification as a definitive causal statement.

Component	Technology	Role
Programming	Python 3	Implementation
Fairness	Fairlearn	MetricFrame/mitigation
ML	scikit-learn	Models/probes
Drift	river	Page-Hinkley/ADWIN
Statistics	SciPy	KS test
Data	pandas, NumPy	Windowing
Storage	Parquet/SQLite	Metric history
Dashboard	Streamlit/Plotly	Reporting
Development	Jupyter/GitHub	Experiments

Table IV. Technologies proposed for the LUCID implementation.

IX. DISCUSSION

The principal strength of LUCID is its alignment of three concerns that are often treated separately: measurement, monitoring and explanation. Fairlearn supplies the measurement abstraction; streaming drift methods supply the monitoring machinery; and the attribution layer supplies an operational interpretation. Packaging these components together addresses the practical accessibility gap identified in the synopsis.

The use of confidence intervals is particularly important because fairness metrics are often calculated for demographic groups whose sample sizes can be unequal. A small group can produce unstable estimates, and a monitoring system that ignores uncertainty may repeatedly flag harmless sampling fluctuations. Conversely, confidence intervals should not be used as a blanket mechanism for suppressing alerts. Their purpose is to contextualize detector output and reduce false alarms attributable to insufficient sample size. The mechanism-level attribution also changes the role of fairness monitoring. Traditional dashboards may show that a metric is above or below a target. A mechanism-oriented monitor instead asks what changed in the data-generating or prediction process. This supports a more disciplined response. Covariate shift can motivate investigation of the incoming population, base-rate shift can motivate monitoring of outcome prevalence, concept shift can motivate model reassessment, and sampling artefact can motivate data-quality checks.

The proposed integration with Fairlearn is also strategically important. Fairness infrastructure is more likely to be adopted when it fits the tools already used by data scientists. The synopsis therefore treats API similarity, packaging and documentation as first-class objectives. This is consistent with the broader production concern that machine-learning systems accumulate technical debt when research methods are not translated into maintainable engineering components [14].

However, LUCID should not be interpreted as solving fairness in general. The choice of metric remains context-dependent, and a detected statistical change does not automatically establish legal, ethical or causal significance. Wachter et al. discuss the legal dimension of fairness-metric selection, illustrating why metric choice can have consequences beyond model engineering [13]. The framework can make monitoring more systematic, but it cannot decide which fairness definition an organization ought to adopt.

Similarly, mechanism attribution is inherently limited by the candidate mechanisms and tests included in the system. Real production changes can have multiple interacting causes. The four-way classification in the synopsis is therefore best treated as a structured diagnostic taxonomy rather than a complete causal model. The evaluation must explicitly measure attribution accuracy under controlled synthetic conditions before the output is relied upon in practice.

X. LIMITATIONS AND FUTURE DIRECTIONS

The current project definition is a proposed framework, so the synopsis does not provide completed benchmark results. Consequently, this paper does not report numerical detection accuracy, attribution accuracy or performance improvements as established findings. These values must be obtained after implementation.

The fairness metrics considered by the project are inherited from Fairlearn. This is appropriate for compatibility but means that the temporal framework inherits the conceptual limitations of the selected metrics. Future work could examine how different temporal fairness definitions behave under changing base rates and delayed outcomes.

The attribution layer uses statistical tests and reweighting rather than a full causal inference model. Future research could investigate richer causal diagnostics, multi-mechanism decomposition and uncertainty-aware attribution. The reporting layer could also evolve from a rule-based summary toward a more formal evidence representation.

The proposed architecture can also be extended to automated mitigation workflows. The synopsis mentions Fairlearn's existing mitigators, but an operational system would require safeguards before automatically changing a deployed model. Future work could therefore study human approval gates, rollback procedures, audit logs and monitoring of post-mitigation behavior.

Finally, the framework should be tested on genuinely timestamped production-like datasets with irregular data arrival, sparse sensitive groups and multiple simultaneous changes. Synthetic injection is necessary for ground-truth evaluation, but it cannot establish full real-world validity on its own.

XI. CONCLUSION

Temporal fairness is a deployment problem that cannot be adequately addressed by one-time fairness audits. The project synopsis identifies a concrete gap between the static evaluation capabilities of widely adopted fairness tooling and the continuous monitoring requirements of deployed machine-learning systems. LUCID addresses this gap by extending the Fairlearn evaluation paradigm with temporal windowing, fairness-metric time series, statistical drift detection, confidence-interval gating and mechanism-level root-cause attribution.

The review of related work shows that the individual ingredients already have strong foundations. Fairlearn provides disaggregated fairness assessment [1]; FairCanary demonstrates continuous explainable fairness monitoring [2]; Page-Hinkley and ADWIN provide sequential and adaptive drift-detection foundations [7], [8]; and explainability literature provides feature-attribution methods [15], [16]. The proposed contribution is the integration of these ideas into an accessible Fairlearn-oriented framework, with an explicit emphasis on statistical alert reliability and actionable mechanism-level explanation.

The proposed evaluation plan is equally important because the framework is not yet an experimentally validated system. Synthetic drift injection on Adult Income, COMPAS and German Credit can provide controlled ground truth, while detection lag, false-positive rate, attribution accuracy and integration overhead provide measurable criteria. A real timestamped case study can then test the framework under less controlled conditions.

In summary, LUCID is best understood as a practical architecture for moving fairness assessment from a static audit toward continuous, explainable monitoring. Its success depends not only on whether it detects change, but also on whether it suppresses noise, provides useful diagnostic information and can be adopted without substantial changes to existing Fairlearn-based workflows.

XII. DETAILED DESIGN CONSIDERATIONS

Temporal fairness monitoring can be understood as a sequence of repeated conditional evaluations rather than a single fairness score. Let W_t denote the observations assigned to temporal window t , and let $M(W_t, g)$ denote the selected fairness-related metric for group g in that window. The monitoring layer therefore produces a sequence $M_1(g)$, $M_2(g)$, ..., $M_T(g)$. This representation makes the time dimension explicit and separates the calculation of a fairness metric from the statistical question of whether its trajectory has changed.

The synopsis emphasizes that the proposed TemporalMetricFrame should preserve the familiar MetricFrame calling convention. This is important because the temporal layer should not require practitioners to redesign their fairness evaluation code. Instead, the extension can be understood as adding a temporal index around an existing group-wise metric computation. In practical terms, the framework receives y_{true} , y_{pred} , $sensitive_features$ and $timestamp$, constructs windows, evaluates the requested metric in each window, and stores the resulting values for monitoring.

An important implementation issue is the relationship between window boundaries and data availability. Fixed-time windows such as daily or weekly intervals are easy to interpret, but they can contain highly unequal numbers of observations. A quiet period may contain too few records to support a stable group-level fairness estimate, while a busy period may contain enough records to detect small changes. Fixed-size sliding windows address part of this issue by controlling the number of observations, but they create overlap between consecutive estimates. Adaptive windows provide a third option in which the historical amount of data is itself responsive to evidence of change.

The choice of window should therefore be treated as an experimental parameter rather than a hidden implementation detail. The evaluation should compare at least one fixed-time

configuration with a fixed-size configuration and, where implementation permits, an adaptive configuration. Detection lag, false-positive rate and computational overhead should be reported separately for these settings. Such a comparison would reveal whether faster temporal resolution is actually worth the increased statistical variability and processing cost.

Uncertainty estimation is another central design consideration. The synopsis proposes bootstrap confidence intervals for fairness metrics and uses confidence-interval gating in the drift-alert process. The purpose is not to prove that a fairness metric is statistically fair or unfair. Instead, the interval provides an estimate of the uncertainty around the observed window-level value. When the sample size of a sensitive group is small, the interval can become wide, signaling that a large apparent change may not be reliable enough to trigger an operational response.

Confidence-interval gating should therefore be implemented as an evidence filter rather than as an independent fairness criterion. A detector such as Page-Hinkley or ADWIN can indicate that a monitored series has changed. The confidence information can then determine whether the observed change is sufficiently supported by the data in the relevant windows. This separation allows the detector and the uncertainty mechanism to be evaluated independently and avoids conflating drift detection with fairness policy.

The synopsis proposes two complementary stream-drift methods. Page-Hinkley is suitable for sequential detection of persistent changes in a monitored statistic, whereas ADWIN adapts its effective historical window as the stream changes. Using both provides an opportunity to compare a detector based on cumulative deviation with one based on adaptive window management. The final implementation should preserve detector-specific outputs so that an alert can record which method signaled the event rather than hiding the distinction behind a single binary result.

A robust alerting process should also distinguish between a metric changing temporarily and a metric entering a new regime. A short-lived spike can be operationally different from a persistent shift. The monitoring history should therefore retain the onset time, duration, magnitude and recovery behavior of each detected event. These properties can be included in the temporal metric store and surfaced in the dashboard. The synopsis does not specify a final persistence rule, so this should remain an implementation and evaluation decision rather than being presented as an established result.

The root-cause module is deliberately organized around a small diagnostic taxonomy. Covariate shift concerns changes in the distribution of input features. Label or concept shift concerns changes involving the relationship between the features and the target. Base-rate shift concerns changes in outcome prevalence or group composition. Sampling artefact concerns instability caused by sparse or changing observations. The value of this taxonomy is operational clarity: each category points toward a different investigation path.

Covariate-shift analysis can use distributional comparisons between an earlier reference window and a later window. The synopsis identifies the Kolmogorov-Smirnov test and Population Stability Index as candidate tools. These tests should be interpreted feature by feature and then summarized rather than treating a single significant feature as definitive proof of the cause of fairness drift. Multiple features may move simultaneously, and some changes may have no meaningful effect on the fairness metric.

For label or concept shift, the synopsis proposes a lightweight probe model trained on recent data and compared with the baseline relationship. The purpose is diagnostic rather than predictive optimization. If the relationship between inputs and outcomes changes materially while input distributions remain relatively stable, the evidence is more consistent with a change in the predictive relationship. The evaluation should use synthetic injections where the true mechanism is known so that the limitations of this diagnostic rule can be measured.

Base-rate shift requires attention to the distinction between group proportions, true outcome rates and prediction rates. These quantities can move independently. A change in the proportion of a sensitive group is not identical to a change in the underlying probability of the outcome, and neither is identical to a change in model predictions. The attribution layer should therefore retain these quantities separately in the metric store. This reduces the risk of labeling every demographic-composition change as a model fairness failure. Sampling artefact is particularly important for continuous monitoring because group-level fairness metrics are sensitive to sample size. A group can become temporarily sparse because of an upstream collection issue, a change in traffic, or a legitimate reduction in its representation. The confidence interval and sample-count signals should therefore be visible together. A report that says only that a fairness metric changed is incomplete when the affected group contains very few observations.

The reweighting decomposition described in the synopsis provides a useful bridge between detection and attribution. Earlier observations can be reweighted toward the later covariate distribution and the fairness metric can be recalculated. If the metric difference is substantially reduced, this supports the hypothesis that covariate composition explains part of the change. The method should not be interpreted as a causal proof because reweighting can itself depend on modeling choices and observed variables.

The reporting layer should preserve the distinction between evidence and conclusion. A strong implementation should report the measured metric change, the statistical detector signal, the confidence information, the sample sizes and the diagnostic evidence before presenting the likely mechanism. Plain-language summaries can then translate the structured result without removing uncertainty. This is especially important when reports may be read by users who are not specialists in statistical drift detection.

The proposed dashboard can organize information at three levels. The first level is a high-level alert view showing the affected group, fairness metric, time window and severity. The second level is a temporal plot showing the metric trajectory and confidence intervals. The third level is a diagnostic view containing distributional tests, sample-size changes and the evidence supporting each candidate mechanism. This layered structure would allow a non-specialist user to understand the alert while still permitting technical inspection.

Integration with existing Fairlearn mitigation methods should remain optional. The synopsis mentions Exponentiated Gradient and Threshold Optimizer as possible mitigation mechanisms. Automatically changing a deployed model immediately after an alert would be inappropriate without additional controls because the detected change may reflect a data issue rather than model behavior. A safer architecture is to produce an evidence-backed recommendation and allow a human or deployment policy to approve mitigation.

The proposed packaging objective is therefore part of the research contribution. A pip-installable extension with a Fairlearn-like interface reduces the distance between the review framework and an actual data-science workflow. Documentation should include a minimal example, a temporal monitoring example, a synthetic drift experiment, an attribution example and a dashboard example. These examples would make the framework reproducible and provide a concrete basis for evaluating integration overhead. Reproducibility should also be considered in the benchmark design. Each synthetic drift experiment should record the initial data distribution, drift onset, injected mechanism, magnitude of the change, windowing parameters and detector configuration. The evaluation can then be rerun under different random seeds and sample sizes. Reporting only one successful experiment would not be sufficient evidence of reliable monitoring behavior.

An appropriate evaluation protocol should include stable-stream experiments in addition to drift experiments. Stable streams are necessary to measure false-positive behavior. If the system is tested only when drift is present, it is impossible to determine whether the alerting logic is overly sensitive. Stable experiments should vary group sample size because sparse-group conditions are a central motivation for confidence-interval gating.

Drift experiments should vary both the magnitude and the onset speed of the injected change. A large abrupt change and a small gradual change are different detection problems. Page-Hinkley and ADWIN may respond differently to these conditions, and the comparison can reveal where each method is useful. The same evaluation should be repeated across multiple fairness metrics where the benchmark data permit it.

Attribution experiments require ground truth at the mechanism level. For example, a covariate-shift experiment can alter a selected feature distribution while holding the underlying label relationship constant. A base-rate experiment can change outcome prevalence, while a sampling experiment can reduce the number of observations for a selected group. Combining these injections can then test whether the attribution layer correctly handles ambiguous cases rather than assuming that every drift has one cause.

Integration overhead should be measured with the same seriousness as statistical performance. The synopsis proposes lines of code as one practical measure. Additional useful measurements include execution time per window, memory used by the temporal metric store, dashboard latency and the amount of data retained. These metrics would determine whether the proposed monitoring architecture is realistic for repeated use rather than merely functional on a small benchmark.

The limitations of the review should remain explicit. The project synopsis specifies the architecture, technology choices and evaluation plan, but it does not establish empirical superiority over existing approaches. Therefore, claims such as improved accuracy, reduced false positives or better attribution should remain hypotheses until experiments are completed. This paper's contribution at the review stage is to consolidate the methodological rationale and identify a precise path for experimental validation.

XII. IMPLEMENTATION CHECKLIST AND RESEARCH QUESTIONS

The implementation can be organized around a sequence of research questions. RQ1 asks whether fairness metrics can be monitored reliably as temporal streams without changing the semantics of Fairlearn's existing metric calculations. RQ2 asks whether Page-Hinkley and ADWIN can detect injected fairness drift with acceptable detection lag while maintaining a low false-positive rate. RQ3 asks whether confidence-interval gating improves alert reliability under sparse-group conditions. RQ4 asks whether the proposed four-way attribution taxonomy can identify the known mechanism under controlled drift injection. RQ5 asks whether the resulting interface is sufficiently close to Fairlearn workflows to keep integration overhead low.

A corresponding implementation checklist should verify timestamp validation, deterministic window construction, minimum group sample checks, metric computation, bootstrap interval generation, temporal persistence, detector state management, alert deduplication, root-cause diagnostics, structured reporting and dashboard visualization. Each stage should have unit tests and at least one synthetic example. This separation is important because a monitoring failure can originate from data ingestion or windowing rather than from the statistical detector itself.

The research should also define what constitutes a confirmed alert before experiments begin. The synopsis provides the core ingredients, namely a stream detector and confidence-interval gating, but does not prescribe one final confirmation rule. The experimental protocol should therefore specify the detector threshold, confidence level, minimum sample size and persistence condition before comparing methods. Otherwise, the evaluation can become biased by tuning the alert rule after observing the results.

The same principle applies to attribution. Candidate mechanisms should be evaluated using predefined diagnostic rules and thresholds rather than changing the explanation logic after seeing the test labels. If multiple mechanisms receive comparable evidence, the system should be permitted to return an uncertain or mixed attribution rather than forcing a single label. This would make the reporting layer more faithful to the uncertainty inherent in real monitoring data.

Taken together, these design decisions make LUCID a testable framework rather than only a conceptual architecture. The central experimental question is not whether temporal fairness is important. That premise is already established by the surrounding literature. The useful research question is whether an accessible Fairlearn-based implementation can detect meaningful temporal fairness changes, reject unstable noise, provide a useful mechanism diagnosis and integrate into existing workflows with limited engineering effort.

XIII. PRACTICAL DEPLOYMENT AND GOVERNANCE CONSIDERATIONS

A temporal fairness monitor will operate inside a larger machine-learning pipeline, so its reliability depends on the quality and timing of the data it receives. The proposed architecture should therefore treat timestamp integrity, missing values, duplicated records and delayed labels as first-class monitoring concerns. A fairness metric calculated before labels become available may be useful for some predictive-rate measures, but metrics requiring y_{true} cannot be evaluated until the corresponding outcomes have arrived. The implementation should record whether each window is complete or provisional so that an alert is not interpreted as final when the underlying labels are still incomplete.

Delayed labels are particularly relevant to fairness monitoring because many consequential decisions have outcomes that become observable only after a period of time. A loan decision, for example, may be available immediately while repayment behavior is observed later. If the system compares windows using incomplete outcomes, an apparent fairness change may simply reflect the maturity difference between cohorts. The project synopsis does not prescribe a solution for delayed labels, so the final implementation should document the assumed label-availability model and treat delayed-outcome handling as a future extension where necessary.

Governance also requires a clear distinction between monitoring and decision authority. LUCID can provide evidence that a fairness metric has changed and can identify a likely mechanism, but the framework should not independently determine whether a model is legally acceptable or whether a business process should be changed. The choice of sensitive attributes, fairness metrics, alert thresholds and mitigation policies remains a governance decision. The framework's role is to make the relevant evidence more visible, reproducible and technically interpretable.

Auditability can be improved by storing the configuration associated with every alert. The metric definition, sensitive feature, windowing method, confidence level, detector configuration and attribution rules should be recorded with the event. If the model or data pipeline changes later, an analyst should be able to reconstruct which configuration generated the earlier alert. This is consistent with the production-oriented motivation of the project and reduces the risk that a monitoring dashboard becomes an opaque historical record.

A useful operational workflow would separate observation, diagnosis and intervention into three stages. Observation records the temporal fairness signal and its uncertainty. Diagnosis evaluates the candidate mechanisms and produces an evidence summary. Intervention is a separate step that may involve data-quality investigation, model retraining, reweighting, threshold adjustment or no action. This separation is important because the same fairness drift can have different appropriate responses depending on its mechanism and business context.

The dashboard should also avoid presenting a single severity number as if it completely represents fairness risk. A large metric change in a very small group can have different evidentiary strength from a smaller change supported by a large number of observations. A more informative interface would display the metric change together with sample size, confidence interval, duration and mechanism evidence. This design follows the central principle of the proposed framework: fairness monitoring should combine statistical detection with interpretable context.

For future extensions, the temporal metric store could support multiple models or model versions simultaneously. This would allow practitioners to distinguish a change caused by the population from a change associated with a particular model release. Such version-aware monitoring is not required by the current synopsis, but it follows naturally from the temporal architecture. It would also support controlled comparisons between a baseline model and a mitigated model after an intervention.

Overall, the deployment considerations reinforce the main research gap identified in this review. Continuous fairness assessment is not only a statistical problem. It is also a systems problem involving data windows, uncertainty, delayed information, configuration management, explanations and human decision-making. A successful implementation of LUCID therefore needs to be evaluated as an integrated monitoring component rather than only as a collection of drift-detection algorithms.

REFERENCES

- [1] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker, “Fairlearn: Assessing and improving fairness of AI systems,” *Journal of Machine Learning Research*, vol. 24, no. 1, pp. 1–8, 2023.
- [2] A. Ghosh, A. Shanbhag, and C. Wilson, “FairCanary: Rapid continuous explainable fairness,” in *Proc. 2022 AAAI/ACM Conf. on AI, Ethics, and Society (AIES '22)*, Oxford, U.K., Aug. 2022, pp. 307–316.
- [3] S. Corbett-Davies and S. Goel, “The measure and mismeasure of fairness: A critical review of fair machine learning,” *arXiv preprint arXiv:1808.00023*, 2018.
- [4] M. Hardt, E. Price, and N. Srebro, “Equality of opportunity in supervised learning,” in *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [5] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, “Fairness through awareness,” in *Proc. 3rd Innovations in Theoretical Computer Science Conf. (ITCS)*, 2012, pp. 214–226.
- [6] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A survey on bias and fairness in machine learning,” *arXiv preprint arXiv:1908.09635*, 2019.
- [7] A. Bifet and R. Gavaldà, “Learning from time-changing data with adaptive windowing,” in *Proc. 2007 SIAM Int. Conf. on Data Mining*, 2007, pp. 443–448.
- [8] E. S. Page, “Continuous inspection schemes,” *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.
- [9] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Proc. Brazilian Symp. on Artificial Intelligence (SBIA)*, 2004, pp. 286–295.
- [10] L. T. Liu, S. Dean, E. Rolf, M. Simchowitz, and M. Hardt, “Delayed impact of fair machine learning,” in *Proc. 35th Int. Conf. on Machine Learning (ICML)*, 2018, pp. 3150–3158.
- [11] F. Kamiran and T. Calders, “Data preprocessing techniques for classification without discrimination,” *Knowledge and Information Systems*, vol. 33, no. 1, pp. 1–33, 2012.
- [12] A. Miroshnikov, K. Kotsiopoulos, R. Franks, and A. R. Kannan, “Wasserstein-based fairness interpretability framework for machine learning models,” *arXiv preprint arXiv:2011.03156*, 2020.
- [13] S. Wachter, B. Mittelstadt, and C. Russell, “Bias preservation in machine learning: The legality of fairness metrics under EU non-discrimination law,” *West Virginia Law Review*, vol. 123, p. 735, 2020.
- [14] D. Sculley, G. Holt, D. Golovin, E. Davydov, D. Phillips, E. Ebner, D. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [15] M. T. Ribeiro, S. Singh, and C. Guestrin, ““Why should I trust you?” Explaining the predictions of any classifier,” in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [16] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

