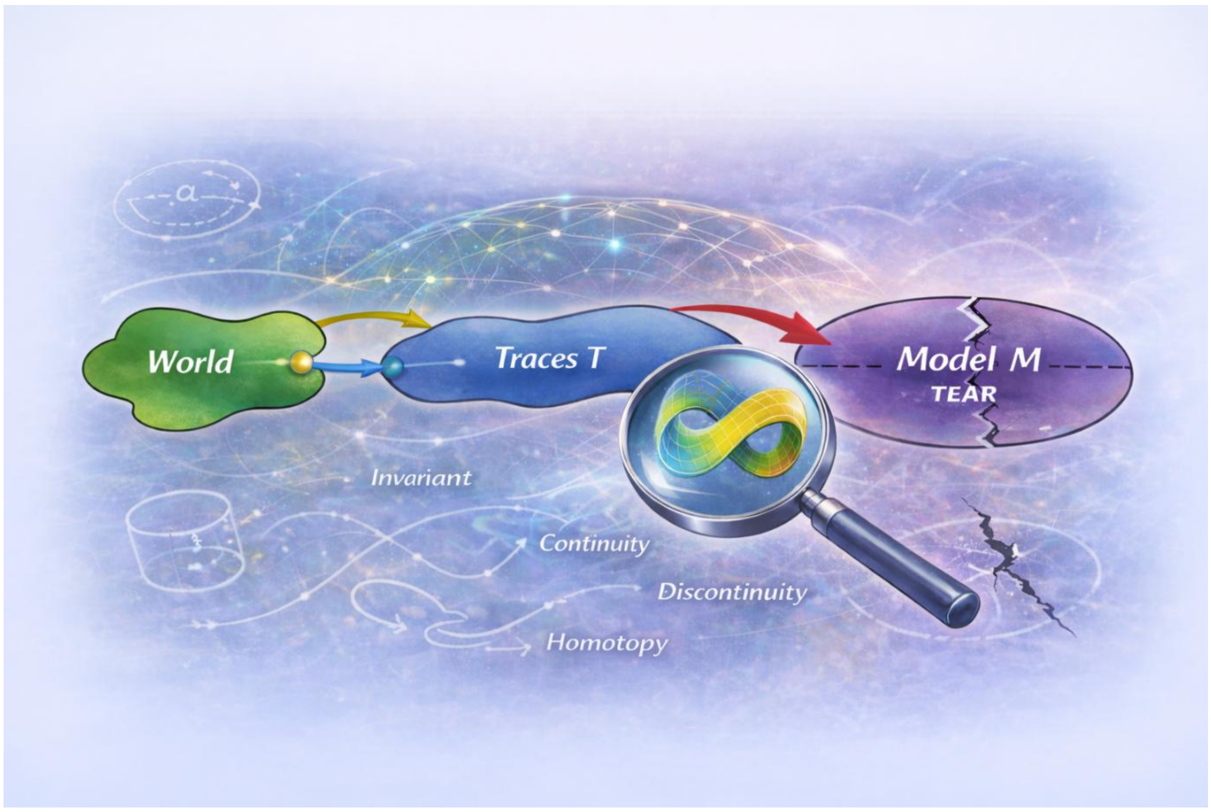

# Pattern-Oriented Diagnostics and Observability of World Models: A Topological Perspective

## Introduction

Models of the world are traditionally evaluated by their ability to predict observable outcomes. Accuracy, likelihood, loss functions, and benchmark scores dominate both scientific and engineering practice. Yet such criteria say little about how a model organizes reality, how stable its explanations are under perturbation, or how it changes as systems evolve. In long-lived software systems, diagnostic pipelines, simulators, and modern AI models, these questions become central. What is required is a framework for comparing models not merely by performance, but by structure. We propose a framework that treats model comparison as a topological diagnostic problem, building on the previous ideas and earlier work on software defects[1], causal models[2], geometrical debugging[3], models of software behavior[4], higher-order pattern narratives[5],

[1] What is a Software Defect? Theoretical Software Diagnostics, Fourth Edition, page 33 (https://www.dumpanalysis.org/theoretical-software-diagnostics-book)
[2] Causal Models, Ibid., page 52
[3] Geometrical Debugging, Ibid., page 60
[4] Models of Software Behavior, Ibid., page 108
[5] Higher-Order Pattern Narratives (Analyzing Diagnostic Analysis), Ibid., page 202

topological software trace and log analysis[6], traces and logs as models of computation[7], and pattern-oriented observability[8].

## World Models as Topological Mappings

At the foundation of this approach lies a simple abstraction. Any model of the world may be understood as a mapping $f_i: W \to M_i$, where $W$ denotes a space of world states, observations, executions, traces, or narratives, and $M_i$ denotes the internal representational space of the model, including latent variables, symbolic explanations, diagnoses, predictions, or narratives produced by the model. The crucial assumption is that both $W$ and $M_i$ are not mere sets, but spaces equipped with topology, that is, with a notion of neighborhood and nearness. This assumption is deliberately weak. It does not require metrics, probabilities, or differentiability, yet it is strong enough to speak meaningfully about continuity, deformation, invariants, and discontinuities.

## Software as a World

In this framework, software systems themselves may be treated as worlds rather than merely as artifacts within a world. A running software system defines its own space of states, executions, interactions, and narratives, shaped by control flow, data flow, concurrency, configuration, and environmental inputs. From the perspective of diagnostics and observability, this space is not secondary or derivative; it is the primary reality with which engineers interact. Crashes, performance anomalies, security violations, and semantic failures occur within this software world, even when they have no immediate counterpart in the physical environment. Treating software as a world enables the direct application of topological reasoning to execution behavior, traces, and models, without requiring an appeal to external "ground truth." The world $W$ is thus not restricted to physical systems or user-visible outcomes, but includes the internally generated realities of computation itself. This perspective underlies treating traces, logs, and models as world models: they are successive representations of a world that is already artificial, structured, and designed.

## Traces as First-Class World Models

Execution traces and logs are often treated as neutral records of system behavior, positioned between the world and higher-level analytical models. In practice, however, traces and logs already function as world models in their own right. They are selective, structured, and interpretive representations of execution, shaped by instrumentation

[6] Topological Software Trace and Log Analysis, Ibid., page 257
[7] Trace Bias, Memory Dump Analysis Anthology, Volume 16, page 55 (https://www.dumpanalysis.org/Memory+Dump+Analysis+Anthology+Volume+16), and Trace Parameters and Hyperparameters, Ibid., Volume 17, page 34 (https://www.dumpanalysis.org/Memory+Dump+Analysis+Anthology+Volume+17)
[8] Pattern-Oriented Observability, Ibid., page 65

choices, logging schemas, sampling strategies, and storage constraints. As such, they impose structure on reality before any further analysis takes place.

Formally, tracing introduces a mapping $f_{\text{trace}}: W \to T$, from the space of executions $W$ to a trace or log space $T$. This mapping is neither complete nor transparent. It encodes decisions about which events matter, how time is represented, how causality is inferred, and which invariants are preserved or discarded. Consequently, traces do not merely expose the world; they actively model it.

Recognizing traces as first-class world models has important diagnostic consequences. Continuity, homotopy, invariants, and discontinuities apply at the trace level just as they do at the level of analytical models. Discontinuities may arise not from changes in execution, but from changes in instrumentation, log formatting, sampling, or correlation mechanisms. Similarly, two tracing systems may produce structurally different traces that are nevertheless homotopic, preserving diagnostic meaning despite representational variation.

This perspective also clarifies the composition of diagnostic models. Analytical world models operate not directly on execution, but on trace-based representations. The effective mapping from the world to a model is therefore a composition of mappings, $W \overset{f_{\text{trace}}}{\rightarrow} T \overset{f_{\text{model}}}{\rightarrow} M$, and failures of observability may originate at any stage in this chain. Treating traces as first-class world models allows diagnostic reasoning to localize such failures precisely, distinguishing system faults from tracing artefacts and modeling defects.

By elevating traces and logs to the status of world models, observability itself becomes a modeling problem rather than a purely instrumental one. The quality of diagnosis depends not only on the sophistication of analytical models, but on the structural faithfulness of the trace models on which they are built. In this sense, trace design is model design, and tracing infrastructure becomes an integral part of the diagnostic and epistemic foundation of complex systems.

## Continuity as a Diagnostic Criterion

Under this abstraction, a model is no longer merely a function producing outputs, but a structural interpreter of the world. Comparing models, therefore, means comparing how they preserve or distort the neighborhood structure of $W$. Continuity becomes the first diagnostic criterion. A model is continuous if small perturbations in the world induce only local changes in the model's internal representation. When two models are compared, the model that preserves neighborhoods more faithfully is structurally superior, even if both achieve comparable accuracy on test data. Continuity violations manifest operationally as brittle behavior, unstable explanations, excessive sensitivity to noise, adversarial fragility, or abrupt diagnostic jumps.

## Homotopy and Model Equivalence

Continuity alone is insufficient to capture deeper notions of model equivalence. Two models may differ substantially in implementation, architecture, or parameterization, yet encode the same understanding of the world. This motivates treating homotopy as an equivalence relation up to deformation. Two models $f_1, f_2 : W \to M$ are homotopic if there exists a continuous deformation $H : W \times [0,1] \to M$ such that $H(w, 0) = f_1(w)$ and $H(w, 1) = f_2(w)$. Homotopy captures the idea that one model can be transformed into another without tearing, collapsing, or introducing new discontinuities. When such a homotopy exists, the models differ representationally but not conceptually. When no such homotopy exists, the difference is not cosmetic: the models carve the world differently. This notion generalizes the *Trace Homotopy*[9] analysis pattern from execution traces to world models, preserving endpoints and invariants while allowing internal variation.

## Invariants and Semantic Anchors

A further dimension of comparison arises from invariants. Certain structures are expected to remain unchanged under valid modelling transformations. These may include causal ordering, temporal directionality, conservation-like quantities, semantic identities, protocol structure, or the persistence of specific message components across executions. In Pattern-Oriented Diagnostics, this idea is formalized in the *Message Invariant*[10] trace and log analysis pattern. Empirically, trace messages emitted from the same source code fragment typically contain invariant parts, such as function names and symbolic descriptions, alongside mutable parts, such as pointer values, object identifiers, version numbers, or error codes. By separating invariant structure from variable content, diagnosticians can isolate semantically meaningful differences while suppressing incidental variation.

When lifted to the level of world models, invariants act as semantic anchors. A model may preserve topological continuity while violating message invariants, thereby maintaining neighborhood structure but destroying meaning. Conversely, a model that rigidly enforces invariants while distorting continuity risks suppressing legitimate variation. Invariant preservation, therefore, defines what must remain fixed for two representations to be models of the same world.

## Discontinuities as Diagnostic Signals

[9] Trace, Log, Text, Narrative, Data: An Analysis Pattern Reference for Information Mining, Diagnostics, Anomaly Detection, Fifth Edition, page 320 (https://www.dumpanalysis.org/trace-log-analysis-pattern-reference)

[10] Ibid., page 198 (Also, Fundamentals of Trace and Log Analysis: A Pattern-Oriented Approach to Monitoring, Diagnostics, and Debugging, https://link.springer.com/book/10.1007/978-1-4842-9896-1)

Discontinuities occupy a central place in this diagnostic perspective and are directly linked to the *Discontinuity*[11] trace and log analysis pattern. In trace and log analysis, discontinuities appear as abrupt changes in execution flow, unexpected gaps, reordered sequences, missing segments, or sudden shifts in narrative vocabulary. Such events often mark fault boundaries, regime changes, configuration transitions, version mismatches, or failures of observation itself.

The same insight applies at the level of world models. When a model introduces a discontinuity in the mapping $f: W \to M$, it implicitly claims that the world itself undergoes a meaningful transition at that point. The diagnostic question, therefore, is whether this transition corresponds to a genuine break in execution narratives or is merely a model-induced tear. Unaligned discontinuities signal structural mismatch between the model and the world, whereas aligned discontinuities often correspond to real events.

**Structural Translation Between Models**

The strongest form of model comparison arises when one asks whether models can be translated into one another in a structurally faithful way. Suppose there exists a mapping $T: M_1 \to M_2$ such that $f_2 \approx T \circ f_1$. If this diagram commutes up to isomorphism and the translation preserves continuity and invariants, then the two models are structurally equivalent and differ only in representation. If no such translation exists, the disagreement is fundamental: the models impose incompatible organizations on the world.

**Models as Diagnostic Artefacts**

Viewed through the lens of Pattern-Oriented Diagnostics, models themselves become diagnostic artefacts. They can be inspected, compared, versioned, and classified using the same structural reasoning traditionally applied to traces, logs, and memory dumps. Model comparison thus becomes an exercise in identifying continuity violations, invariant breaks, homotopy obstructions, and unexplained discontinuities, rather than merely ranking predictive performance.

**Engineering World Models for Observability**

Observability has traditionally been treated as a property of systems rather than of models. Logs, traces, metrics, and signals are collected around a system, and diagnostic reasoning is applied externally to infer what the system is doing and why. In this view, models are consumers of observability data or predictors built on top of it, but not themselves objects of observability. As models increasingly mediate understanding, explanation, and control of complex systems, this separation between models and

[11] Ibid., page 115 (Also, Ibid.)

observability becomes untenable. Engineering the models for observability requires a fundamental inversion: observability must become a property of the model itself.

When a model is engineered for observability, it is no longer sufficient for it to produce accurate predictions. The model must preserve the diagnostic structure of the world and make it inspectable. If the model is understood as a mapping $f: W \to M,$ from a space of world states, executions, or observations to an internal representational space, then engineering for observability means designing this mapping so that distinctions, transitions, and invariants in the world remain visible within the model. Understanding is no longer reconstructed after the fact; the model's structure carries it.

At the foundation of observable models lies continuity. An observable model is continuous by default: small changes in the world induce small, traceable changes in the model's internal state. This property is essential for diagnosis because, in the absence of continuity, there is no reliable basis for comparison. Noise becomes indistinguishable from events, and explanations jump unpredictably across regimes. Continuity provides a baseline against which deviations can be detected and interpreted. In this sense, continuity is not a mathematical refinement but a prerequisite for trustworthy observability.

Continuity alone, however, is insufficient. Real systems undergo genuine transitions, failures, and regime changes, and an observable model must make such events explicit. This requires engineering for controlled discontinuity. Rather than allowing arbitrary or emergent breaks in behavior, an observable model introduces discontinuities deliberately, at semantically meaningful boundaries. These discontinuities preserve identifiable pre- and post-states and can be aligned with discontinuities observed in traces, logs, or execution narratives. When discontinuities are engineered in this way, they become diagnostic events rather than opaque failures. When they are not, they manifest as model-induced tears that obscure rather than reveal system behavior.

Observability also extends across time, versions, and evolution. Models are retrained, refactored, or replaced, yet diagnostic understanding must persist across these changes. Engineering the models for observability, therefore, requires homotopy: the ability to relate different versions of a model through continuous deformation. When successive models are homotopic, representational changes do not destroy meaning. Diagnostic baselines remain comparable, explanations evolve smoothly, and understanding accumulates rather than resets. Without homotopy, every model update becomes an epistemic rupture, forcing diagnostics to start from scratch.

Another essential aspect of observability is the preservation of invariants. Observable models must retain stable semantic anchors that survive across executions and contexts. These include conceptual identifiers, protocol meanings, narrative roles, and other invariant structures that define what remains the same when everything else varies. This requirement generalizes message-level invariants from trace and log analysis to the level of models themselves. A model may preserve continuity yet violate invariants, producing internally smooth behavior that silently destroys meaning. Conversely, a model that enforces invariants rigidly while distorting continuity suppresses legitimate variation. Engineering for observability requires balancing continuity and invariance so that both structure and meaning remain accessible.

A deeply observable model also supports partial reversibility. While full inversion of the world model mapping is neither possible nor necessary, an observable model allows diagnostic backtracking. From an explanation, one can infer a class of world states; from an anomaly, one can identify a structural cause; from an output, one can locate an internal regime. This partial reversibility enables diagnostic reasoning to flow not only forward, from world to model, but also backward, from model behavior to plausible world structure.

Taken together, these properties define a clear design principle. A model is engineered for observability if its internal structure preserves the diagnostic patterns present in the world and makes violations of those patterns explicit, localized, and interpretable. This principle is stronger than interpretability and orthogonal to accuracy. A model may be accurate yet unobservable, just as it may be interpretable yet diagnostically opaque.

The consequences of engineering models for observability are substantial. Failures become explainable rather than mysterious; diagnostics remain stable across versions; anomalies acquire structural meaning; and long-lived systems accumulate understanding rather than lose it. Most importantly, the model itself becomes part of the observability stack rather than merely a consumer of signals. Observability is no longer something added around a model; it is something the model embodies.

From this perspective, engineering the models for observability is not an optional refinement but a necessary response to the increasing role models play in shaping how complex systems are understood, diagnosed, and trusted.

## Related Work

The ideas developed here intersect with several established bodies of work, including topological methods in data analysis, semantic models in software engineering, research on robustness and interpretability in machine learning, and observability in complex systems. At the same time, they differ from these traditions in both emphasis and scope, treating world models themselves as diagnostic artefacts and grounding their comparison and observability explicitly in Pattern-Oriented Diagnostics.

Topological concepts have long been used in the analysis of physical systems, dynamical systems, and, more recently, in data analysis through techniques such as topological data analysis and manifold learning. In these contexts, topology is typically applied either to the underlying system under study or to the geometry of the observed data. Continuity, connectedness, and invariants are used to reason about stability, phase transitions, or qualitative structure. By contrast, we apply topological reasoning not directly to data or physical state spaces, but to the mapping induced by a model. The object of analysis is not the world alone, but the relationship between the world and its representation inside a model, and, crucially, how this relationship affects what can be observed, interpreted, and diagnosed through the model.

In machine learning, related concerns appear under the headings of robustness, generalization, adversarial stability, and representation learning. Many approaches implicitly assume continuity by enforcing smoothness, imposing Lipschitz constraints, or regularizing latent spaces. Other work studies model sensitivity and brittleness through perturbation analysis. However, such approaches typically remain metric- or loss-driven and focus on performance degradation rather than structural interpretation. Observability, when discussed, is often reduced to logging activations, inspecting attention maps, or probing internal representations post hoc. The framework presented here reframes robustness and sensitivity as topological properties of the world–model mapping, making discontinuities, folds, and violations of invariance first-class diagnostic and observability signals rather than secondary effects of optimization.

Interpretability and explainability research also overlap conceptually with this work. Post hoc explanation methods, concept-based explanations, and symbolic abstractions aim to render model behavior understandable to humans. While valuable, these techniques often operate locally or retrospectively. The present approach instead emphasizes structural faithfulness and structural observability: whether a model preserves neighborhood relations, semantic invariants, and execution narratives globally and across versions. In this sense, the work aligns more closely with epistemic notions of model validity and observability than with visualization or explanation alone.

In software engineering and systems diagnostics, traces, logs, metrics, and execution narratives have long been treated as primary artefacts for observability and diagnostics. Pattern-Oriented Diagnostics, including patterns such as Discontinuity, Trace Homotopy, and Message Invariant, provide a vocabulary for reasoning about stability, change, and failure in observed system behavior. Prior work in this area has focused on concrete execution artefacts and observability signals emitted by running systems. The present contribution extends this diagnostic and observability reasoning to models themselves, treating them as higher-order artefacts that can be inspected, compared, and classified using the same structural principles. In this view, models are not merely consumers of observability data but participants in the observability stack.

The notion of homotopy used here is inspired by both classical topology and its diagnostic analogue in trace analysis. In execution traces, homotopy captures equivalence between different internal paths that preserve endpoints and invariants, enabling observability across execution variability. By lifting this idea to world models, the paper introduces a principled way to distinguish representational change from conceptual drift, and to maintain observability across model evolution, retraining, and refactoring. While homotopy has occasionally appeared as a metaphor in machine learning discourse, we use it systematically as a model comparison and observability criterion grounded in diagnostic practice.

Finally, the idea of invariant preservation connects this work to formal methods, protocol verification, and contract-based design, where invariants play a central role. However, those approaches typically treat invariants as hard constraints or correctness conditions. In contrast, the diagnostic perspective adopted here treats invariant violations as observability signals to be interpreted, localized, and explained, rather than simply as failures. This positions invariant reasoning within observability and diagnosis rather than within verification alone.

In summary, we synthesize mathematical and conceptual components drawn from well-established traditions into a Pattern-Oriented Diagnostics framework for world models that explicitly incorporates observability as a structural property. The unification of topological reasoning, diagnostic patterns, and engineered observability lies at the intersection of systems diagnostics, AI model evaluation, and epistemic analysis of complex systems.

## Conclusion

World models should be treated as diagnostic artefacts whose structure, stability, and evolution can be examined using the same principles applied to execution traces, logs, and memory dumps. By adopting a topological perspective, models are understood as mappings between spaces of observations and internal representations, allowing continuity, homotopy, invariants, and discontinuities to serve as principled criteria for comparison and diagnosis.

Within this framework, continuity characterizes epistemic robustness, homotopy distinguishes representational change from conceptual drift, message invariants anchor semantic commitments, and discontinuities become interpretable signals rather than mere errors. Established trace and log analysis patterns, such as Discontinuity, Trace Homotopy, and Message Invariant, generalize naturally to the level of models, providing a direct bridge between concrete diagnostic practice and abstract model evaluation. These same properties also define model-level observability, ensuring that internal model states, transitions, and explanations remain traceable, comparable, and diagnostically meaningful.

The resulting shift is conceptual as much as technical. Model evaluation moves away from narrow performance-centric criteria toward questions of structural faithfulness:

where a model preserves neighborhood structure, where it introduces tears, which invariants it respects, and which changes correspond to real transitions in the world rather than artefacts of modelling. In an era of long-lived software systems, complex observability pipelines, and increasingly opaque AI models, such questions are no longer optional.

From the perspective of Pattern-Oriented Diagnostics, models themselves become first-class subjects of diagnosis. A good model is one whose topology changes only when the world itself changes, whose invariants break only when semantic commitments change, and whose discontinuities correspond to genuine events rather than modelling artefacts. Topological comparison, therefore, complements traditional evaluation by restoring structure, meaning, and interpretability to the practice of modelling complex systems.

## Appendix

## Case Study: Diagnosing Model Evolution in a Distributed Order Processing System

This case study illustrates how continuity, homotopy, message invariants, and discontinuities jointly support diagnostic reasoning across system behavior and model evolution. The example is drawn from a distributed order-processing system, whose execution traces are used both for operational diagnosis and for training and validating a world model that predicts and explains system behavior.

### System Context

The system consists of an order service, a pricing component, a payment gateway, and a persistence layer. Execution traces are collected across multiple deployments and model versions. A world model $f: W \to M$ is constructed to map observed executions to internal representations that capture causal structure, service interactions, and outcome states. Over time, the system undergoes minor configuration changes, internal refactoring, and a major incident, providing a natural setting in which all four diagnostic concepts arise.

### Continuity: Stable Behavior Under Small Perturbations

Under normal operation, the system processes orders with slight variations in input parameters, such as order amount or user identifier. Traces collected during this phase exhibit stable structure: the same services are invoked in the same order, and the same outcome is produced. Only numeric parameters and identifiers differ.

From the perspective of the world model, these executions correspond to nearby points in the world space $W$ that are mapped to nearby representations in $M$. The mapping is continuous: small changes in input lead to small, localized changes in representation.

This continuity establishes a baseline of epistemic stability. It allows the diagnostician to interpret deviations meaningfully, knowing that the model does not amplify noise into structural change.

**Homotopy: Representational Change Without Conceptual Drift**

The authentication subsystem is later refactored to improve performance. The refactoring alters the internal order of validation and profile-loading steps but preserves all externally observable semantics. Traces before and after the refactoring differ in internal sequencing but share the same start conditions, invariants, and outcomes.

These trace families are homotopic. There exists a continuous deformation from the pre-refactoring mapping $f_0$ to the post-refactoring mapping $f_1$ such that, for each execution, intermediate representations preserve identity, authorization outcome, and security constraints. The world model reflects this by allowing a smooth deformation of its internal structure while maintaining semantic anchors. Diagnostic understanding is preserved across versions: the model evolves without conceptual drift.

**Message Invariants: Anchoring Semantic Comparison**

During routine comparative analysis across users, a discrepancy in object-creation behavior was detected. Trace messages emitted from the same code location contain invariant components, such as function names and symbolic descriptors, alongside variable components, such as object addresses and version numbers. The invariant message structure enables direct comparison across executions.

By isolating invariant parts, diagnosticians identify that a particular user's trace consistently references a different object version than all others. This difference is semantically meaningful rather than incidental. In the world model, these invariants correspond to fixed semantic anchors that must be preserved across mappings. The violation indicates a divergence in interpretation for a specific subset of world states, prompting further investigation.

**Discontinuity: Structural Rupture and Diagnostic Event**

A production incident occurs following a configuration update. Traces that previously exhibited stable cache behavior suddenly show cache eviction followed by a null reference failure. The transition is abrupt: new execution paths appear without a gradual lead-up, and previously invariant assumptions are violated.

This event constitutes a discontinuity. In trace space, it appears as a sudden change in structure. In the world-model mapping, it manifests as a jump that cannot be explained by continuous deformation. The model implicitly asserts that the world has undergone a significant transition. The diagnostic analysis confirms that the configuration update introduced an incompatible cache policy. The discontinuity is aligned with a real-world event rather than being a modelling artefact.

### Integrated Diagnostic Interpretation

Viewed in isolation, each of these phenomena is familiar to engineers. Viewed together, they form a coherent diagnostic narrative. Continuity establishes a stable interpretive baseline. Homotopy allows representational change without loss of meaning across evolution. Message invariants anchor semantic comparison and expose subtle divergences. Discontinuities signal genuine regime changes requiring explanation.

Crucially, the same concepts apply at both the trace and model levels. The world model is not merely trained on traces; it is itself subject to diagnosis using the same structural principles. Its continuity, homotopy, invariance under transformations, and discontinuities determine whether it remains a faithful, observable representation of the system.

### Implications

This case study demonstrates that the proposed topological framework is operationally meaningful. It unifies routine trace analysis and higher-level model evaluation within a single diagnostic vocabulary. By engineering models that preserve continuity, admit homotopy, respect invariants, and expose discontinuities, diagnostic understanding can accumulate over the course of system evolution rather than being repeatedly reset. In this sense, the model becomes a first-class diagnostic artefact and an integral component of the observability stack.

## Case Study: Diagnosing a Faulty World–Model Mapping

This case study examines a scenario in which observed execution behavior remains stable, yet diagnostic failures arise due to defects in the world-model mapping itself. Unlike the previous case, in which discontinuities reflected genuine system events, the present example demonstrates that the model can violate continuity, homotopy, and invariants even when the underlying system remains well-behaved.

### System Context

The same distributed order processing system is operating under stable production conditions. Execution traces collected over time exhibit consistent structure, ordering, and outcomes. No configuration changes, refactoring, or incidents occur during the observation window. A world model $f: W \to M$ is trained to map execution traces into an internal representation used for anomaly detection and root-cause analysis.

A new version of the model is introduced after retraining with additional data and a modified representation layer. Shortly after deployment, the model begins reporting anomalous regime changes despite the absence of corresponding changes in the execution traces.

### Apparent Discontinuity Without Trace Evidence

The model flags abrupt transitions between internal states corresponding to “normal operation” and “degraded performance.” These transitions appear as sharp jumps in the model’s internal space, triggering alerts and diagnostic workflows. However, a review of the underlying traces reveals no corresponding discontinuities. Message ordering, component interactions, and outcomes remain consistent with historical baselines.

From a topological perspective, this discrepancy is immediately significant. The mapping from the world to traces is continuous, whereas the mapping from traces to the model is not. The latter mapping introduces a discontinuity that is not present in the world. This indicates a model-induced tear rather than a world event.

**Failure of Homotopy Across Model Versions**

Further investigation compares the previous model version $f_0$ with the newly deployed version $f_1$. Although both models are trained on overlapping data and intended to represent the same system semantics, no continuous deformation exists that relates $f_0$ to $f_1$ while preserving invariants. Small variations in traces that previously produced small representational changes now result in qualitatively different internal states.

The absence of a homotopy between $f_0$ and $f_1$ indicates conceptual drift introduced by the model rather than by the system. The models are not merely different implementations; they impose incompatible organizations on the same world.

**Invariant Violation in the Representation**

Inspection of the internal representations reveals that certain semantic anchors previously preserved by the model have been lost. Messages that were formerly mapped to stable conceptual identifiers are now distributed across multiple internal categories based on incidental parameters, such as timing or identifier ranges. Message-level invariants present in the traces are no longer preserved in the model representation.

This violation is subtle. The traces themselves still contain invariant structure, but the mapping fails to respect it. As a result, semantically identical executions are interpreted as distinct, fragmenting the diagnostic narrative. The failure is not one of observability instrumentation, but of representational fidelity.

**Continuity Breakdown as a Mapping Defect**

The combined effect of these changes is a breakdown of continuity in the world-model mapping. Nearby points in the world space, nearly identical executions, are mapped to distant regions in the model space. The model amplifies insignificant variation into structural differences. This violates the baseline assumption required for meaningful diagnosis: that small causes produce small effects in representation.

Importantly, this breakdown occurs without any corresponding trace-level instability. Continuity holds in the world and in the observability artefacts, but fails in the model.

### Diagnostic Resolution

Recognizing the issue as a mapping fault rather than a system fault reframes the diagnostic task. Instead of searching for hidden incidents or race conditions, attention shifts to the model's design and training process. Analysis reveals that a normalization change in the representation layer introduced a sharp boundary condition, effectively partitioning the model space in a way that has no counterpart in the world.

The corrective action is therefore model-centric: restoring invariant preservation, smoothing the representation, and re-establishing continuity and homotopy with the prior model version. Once these properties are restored, spurious anomalies disappear without altering the underlying system.

### Implications

This case study demonstrates the diagnostic power of treating models themselves as artefacts subject to topological analysis. By comparing continuity, homotopy, and invariant preservation across traces and models, it becomes possible to distinguish genuine world events from representational artefacts. Discontinuities that arise in the absence of trace-level evidence can be correctly attributed to the mapping rather than the system.

## Case Study: Diagnosing Failure Across a Hierarchy of World Models

This case study examines a production incident in which execution traces and logs serve as an intermediate world model linking system behavior to higher-level analytical models. The example demonstrates how continuity, homotopy, message invariants, and discontinuities must be evaluated not only between the world and a model, but across a hierarchy of models. The failure ultimately happens neither in the system nor in the analytical model, but in the trace model itself.

### System and Observability Context

Consider a distributed payment processing system comprising an API gateway, an authorization service, a fraud-detection component, and a payment provider. Execution traces and logs are collected using a centralized tracing infrastructure that emits structured logs with correlation identifiers. These traces serve as the primary input to a diagnostic world model $f_{\text{model}}: T \to M$, where $T$ denotes the trace space and $M$ an internal representation used for anomaly detection and causal analysis.

Implicitly, tracing itself defines a prior mapping $f_{\text{trace}}: W \to T$, from actual executions in the world $W$ to observable traces.

**Continuity at the Trace Level**

Under normal operation, small variations in user behavior, such as transaction amounts or geographic locations, produce traces with a stable structure. The same services are invoked, messages appear in consistent order, and outcomes are unchanged. Minor parameter differences are reflected only in message payloads.

At the trace level, the continuity holds: nearby executions in $W$ map to nearby traces in $T$. The analytical model $f_{\text{model}}$ further preserves this continuity, mapping these traces to nearby internal states in $M$. Diagnostic baselines remain stable, and no anomalies are reported.

**Homotopy Across Trace Schemas**

To reduce log volume, the tracing team deploys a new version of the logging schema. Some messages are merged, others reordered, and verbosity is reduced. Despite these changes, semantic anchors, such a transaction identifiers, authorization outcomes, and service boundaries, are preserved.

Although the resulting traces differ structurally from earlier ones, they are homotopic with respect to diagnostic meaning. There exists a deformation between the old trace mapping $f_{\text{trace}}^{(0)}$ and the new mapping $f_{\text{trace}}^{(1)}$ that preserves endpoints and invariants. Experienced diagnosticians can still interpret the traces correctly, and the analytical model continues to function as expected. Observability across trace evolution is preserved.

**Message Invariants as Trace-Model Commitments**

Within the trace model, certain message components function as invariants: correlation IDs, service names, and outcome labels. These invariants anchor meaning across executions and across schema versions. The analytical model relies on them to group events, infer causality, and detect deviations.

At this stage, invariant preservation holds at both levels: the system produces consistent behavior, and the trace model faithfully preserves its semantic structure.

**Discontinuity Introduced by the Trace Model**

A subsequent optimization introduces aggressive sampling in the tracing infrastructure. Under high load, certain internal service calls are no longer logged, and correlation identifiers are omitted for sampled-out spans. No changes are made to the system itself.

From the system's perspective, behavior remains continuous. However, the trace mapping $f_{\text{trace}}$ now introduces a discontinuity. Traces that were previously structurally similar suddenly lack key messages and identifiers. From the viewpoint of the analytical model, executions appear to jump abruptly between unrelated internal states.

The model flags severe anomalies: apparent partial executions, missing authorization steps, and unexplained failures. Yet inspection of raw system metrics and business outcomes reveals no corresponding incident. The discontinuity exists only in the trace space.

**Diagnostic Localization Across the Model Hierarchy**

Applying the framework proposed in this paper, diagnosticians analyze continuity and invariants across the composed mapping $W \overset{f_{\text{trace}}}{\rightarrow} T \overset{f_{\text{model}}}{\rightarrow} M$. They observe that continuity holds in $W$, fails in $T$, and is amplified in $M$. Homotopy between trace versions is broken: no deformation preserves diagnostic meaning after sampling removes invariant components. The analytical model is behaving correctly with respect to its input; the failure lies in the trace model.

Corrective action is therefore taken at the level of observability, not at the level of system logic or model architecture. Sampling policies are revised to preserve invariant messages and correlation structure. Once this is done, continuity is restored in $T$, homotopy with prior traces re-emerges, and spurious anomalies disappear from $M$.

**Integrated Interpretation**

This case study demonstrates that traces and logs are not passive artefacts, but active world models whose structure directly determines what higher-level models can observe and infer. Continuity, homotopy, invariants, and discontinuities must be evaluated across all layers of representation. A failure at any layer can manifest as a diagnostic anomaly, even when the underlying system is stable.

By treating traces as first-class world models, the diagnostic process can correctly attribute failures to their source. In this case, the problem was neither a system fault nor a modeling defect, but a breakdown in the trace-world mapping. Recognizing this distinction prevents misdirected remediation efforts and reinforces the central claim of this work: effective observability and diagnosis depend on the structural faithfulness of every model in the representational hierarchy.

These examples complete the diagnostic picture. Failures may originate either in the world or in the models. The proposed framework provides tools for distinction. In doing so, it reinforces the central claim: that observability and diagnosis depend as much on the structure of the mapping as on the behavior of the system being observed.