

METaverse 2.0

1. The Digital Butterfly Effect

Domain: Software Engineering | Suggested category: Software

Problem Statement

Develop an intelligent system that analyzes dependencies among interconnected software applications, APIs, databases, microservices, and external platforms to predict potential failures caused by a proposed software change before deployment.

Background

Modern software systems consist of multiple interconnected components that exchange data and depend on one another to perform business operations. A seemingly minor modification, such as changing an API response field, updating a database schema, modifying an authentication rule, or upgrading a shared library, can unintentionally affect several dependent services.

Traditional testing methods primarily validate the component being modified and may fail to identify indirect dependencies across large distributed systems. As a result, unexpected failures may emerge only after deployment, leading to service interruptions, data inconsistencies, financial losses, and increased recovery costs.

Detailed Problem Description

There is a need for a proactive software change-impact analysis platform that can identify direct and indirect dependencies across an organization's technology infrastructure. The system should analyze source code, API specifications, configuration files, service logs, database schemas, and deployment information to construct a continuously updated dependency graph.

When a developer proposes a code change or configuration update, the platform should estimate which components could be affected, identify critical dependency paths, assess the potential severity of failures, and recommend appropriate tests before deployment.

Expected Solution

Develop a software platform that:

- Automatically discovers relationships among applications, services, APIs, databases, and shared libraries.
- Builds a visual dependency graph showing direct and indirect relationships.
- Compares proposed software changes against existing dependencies.
- Predicts potential failure points and assigns risk scores based on dependency criticality and historical incidents.
- Recommends targeted regression tests and identifies components requiring additional validation.
- Integrates with CI/CD pipelines to generate warnings or recommend blocking high-risk deployments.

Expected Outcome

A working prototype that accepts a simulated or real software change, identifies affected components, displays possible failure paths, and produces a risk assessment with recommended preventive tests.

Innovation Focus: Predicting cascading software failures before they occur, rather than merely reporting errors after deployment.

2. The Trust Paradox

Domain: Cybersecurity | Suggested category: Software

Problem Statement

Develop a behavior-based security monitoring system that detects suspicious changes in the runtime activities of trusted applications, even when their digital signatures remain valid and no conventional indicators of compromise are detected.

Background

Organizations depend on digitally signed applications, verified software packages, and trusted vendors to maintain secure computing environments. However, a valid digital signature establishes information about software authenticity and integrity; it does not guarantee that every subsequent action performed by an application is safe.

An application update, altered configuration, compromised dependency, excessive permission, or unexpected change in operating behavior may introduce security risks without invalidating the application's signature.

Conventional security mechanisms that rely primarily on known malware signatures or static trust classifications may not identify such behavior quickly enough.

Detailed Problem Description

There is a need for a security solution that establishes a baseline of normal application behavior and continuously compares actual activities against that baseline. The system should monitor relevant indicators such as network connections, file access, process creation, privilege usage, configuration changes, and interactions with sensitive resources.

When an application begins exhibiting unusual behavior, the platform should determine whether the deviation is benign, suspicious, or potentially dangerous. It should correlate events over time, explain the reasons for its assessment, and generate actionable alerts while minimizing false positives.

Expected Solution

Develop a runtime application-trust monitoring platform that:

- Establishes behavioral baselines for approved applications.
- Monitors changes in permissions, network activity, file operations, process execution, and access to sensitive resources.
- Detects deviations from expected behavior, including suspicious activity following an update.
- Correlates multiple events to distinguish isolated anomalies from potentially harmful activity.
- Generates explainable risk scores and alerts for security administrators.
- Supports configurable responses, such as requesting human review, restricting selected operations, or isolating an application where technically supported.

Expected Outcome

A demonstrable prototype that monitors sample applications, detects deliberately introduced behavioral anomalies, explains the triggering evidence, and generates security alerts.

Innovation Focus: Evaluating whether software continues to behave safely after it has been trusted, rather than relying on its initial trust status alone.

3. The Data Time Machine

Domain: Data Engineering | Suggested category: Software

Problem Statement

Develop a time-aware data reconstruction and decision-audit platform that enables organizations to recreate the information available at a particular historical moment and explain decisions using the records and rules that existed at that time.

Background

Organizations routinely modify databases as new information becomes available. Records may be corrected, transactions may be backdated, customer details may change, and historical entries may be overwritten or deleted. Although the current database may contain the latest and most accurate information, it may not accurately represent what the organization knew when an earlier decision was made.

This creates difficulties in financial audits, compliance reviews, insurance claims, customer disputes, inventory investigations, and operational decision-making. Conventional backups can restore data from selected points in time, but they may not provide an efficient, queryable explanation of how individual records and decisions evolved.

Detailed Problem Description

There is a need for a system that preserves the history of data changes and reconstructs historical data states without losing the distinction between when a fact was valid in the real world and when it was recorded in the system.

For example, a loan application may have been rejected based on an income record available on a particular date. If that record is corrected months later, investigators should be able to reconstruct the original information, identify when the correction occurred, and understand which rules and data contributed to the original decision.

The system should support historical queries, data lineage, version comparisons, and traceable decision explanations while preserving appropriate access controls and audit records.

Expected Solution

Develop a historical data reconstruction platform that:

- Records data changes with timestamps, version identifiers, and change sources.
- Supports both valid-time and transaction-time queries where appropriate.
- Reconstructs the data state as it existed at a specified date and time.
- Compares historical and current records and identifies corrections, deletions, and backdated updates.
- Links historical decisions to the data inputs, business rules, and model versions used at the time, where available.
- Generates audit reports with traceable evidence and access-controlled historical records.

Expected Outcome

A prototype that ingests versioned sample datasets, reconstructs a selected historical state, compares it with current data, and explains a past decision using the recorded evidence.

Innovation Focus: Reconstructing not only what the data says now, but what an organization could actually have known when a decision was made.

4. The Internet of Unreliable Devices

Domain: IoT and Edge Computing | Suggested category: Software / Hardware

Problem Statement

Develop a resilient edge-computing system that detects unreliable or conflicting sensor readings, makes safe decisions using multiple data sources, and continues essential operations during internet or central-server outages.

Background

Internet of Things (IoT) systems are increasingly used in smart buildings, agriculture, industrial automation, transportation, and environmental monitoring. These systems rely on sensors and connected devices to collect information and support automated decisions.

However, sensors may produce inaccurate readings because of hardware faults, calibration errors, environmental interference, battery depletion, or communication failures. Multiple sensors monitoring the same environment may report contradictory values. Furthermore, many IoT solutions depend on cloud platforms for processing and decision-making, making them vulnerable to network interruptions.

In safety-sensitive environments, blindly trusting a single sensor or waiting for cloud connectivity may result in incorrect actions, operational delays, or unsafe conditions.

Detailed Problem Description

There is a need for a distributed decision-making system that can evaluate the reliability of sensor readings, resolve conflicting observations, and operate locally when central connectivity is unavailable.

The system should distinguish between genuine changes in environmental conditions and faulty measurements. It should use factors such as sensor history, confidence levels, agreement among independent devices, timestamps, and known physical constraints to determine the most reliable interpretation.

When uncertainty becomes too high, the system should adopt a predefined safe operating mode rather than making potentially dangerous decisions.

Expected Solution

Develop an intelligent edge-monitoring platform that:

- Collects data from multiple sensors and connected devices.
- Detects abnormal, stale, inconsistent, or potentially faulty readings.
- Assigns confidence scores to sensor observations.

Combines information from multiple sensors using suitable sensor-fusion techniques.

- Executes essential decision logic locally during internet or cloud-server outages.
- Stores events locally and synchronizes records when connectivity is restored.
- Activates safe fallback actions and alerts operators when reliable decisions cannot be made.

Expected Outcome

A working prototype using simulated or physical sensors that introduces conflicting readings and network failures, demonstrates reliable local decision-making, and logs the system's responses.

Innovation Focus: Building smart systems that remain dependable when their sensors and communication networks cannot be fully trusted.

5. The Self-Breaking AI

Domain: Artificial Intelligence and MLOps | Suggested category: Software

Problem Statement

Develop an AI reliability monitoring platform that identifies hidden model failures, detects changes in data and prediction quality, and recommends corrective actions even when aggregate accuracy or other overall performance metrics remain satisfactory.

Background

Artificial intelligence models are often evaluated using overall accuracy, precision, recall, or other aggregate metrics. However, these metrics may conceal serious weaknesses affecting specific user groups, rare events, particular environments, or unfamiliar input patterns.

After deployment, real-world data can differ from training data because of changing user behavior, environmental conditions, market trends, sensor characteristics, or changes in the underlying process. This phenomenon can cause model drift and unreliable predictions.

A model may continue reporting high overall accuracy while failing on a small but important category of cases. In addition, ground-truth labels may become available only after a delay, making immediate detection of performance deterioration difficult.

Detailed Problem Description

There is a need for a continuous monitoring system that evaluates AI models beyond their aggregate scores. The platform should analyze incoming data distributions, prediction confidence, subgroup performance, delayed outcome labels, and the consequences of incorrect predictions.

It should distinguish between changes in input data, changes in the relationship between inputs and outcomes, and genuine deterioration in model performance. When a significant reliability issue is detected, the system should explain the evidence, identify affected cases, and recommend appropriate interventions.

The system should not automatically retrain or replace a model without evaluating whether the new model is safer and more effective.

Expected Solution

Develop an AI reliability and MLOps platform that:

- Monitors deployed model predictions, confidence levels, and input data distributions.
- Detects data drift, prediction drift, and performance deterioration when ground-truth labels become available.
- Evaluates performance across relevant subgroups, rare cases, and operational conditions.
- Identifies situations in which high aggregate accuracy hides serious localized errors.
- Generates explainable alerts with examples of affected predictions.
- Recommends human review, fallback models, threshold adjustments, or retraining when justified.
- Compares candidate models against existing models before recommending deployment.

Expected Outcome

A prototype that monitors a sample AI model under changing data conditions, identifies hidden performance gaps, and produces a reliability report with recommended corrective actions.

Innovation Focus: Measuring whether an AI model is dependable where it matters most, rather than assuming that a good average score guarantees reliable decisions.

6. The Factory With a Memory

Domain: Manufacturing Industry | Suggested category: Software / Hardware

Problem Statement

Develop a predictive manufacturing analytics system that identifies hidden combinations of production conditions responsible for recurring product defects, including defects that reappear after corrective actions have been implemented.

Background

Manufacturing defects can result from complex interactions among machine parameters, raw material batches, environmental conditions, tool wear, operator practices, production speed, and maintenance schedules.

Traditional quality-control systems often examine individual variables, apply fixed thresholds, or investigate defects only after they occur. However, each variable may appear normal when examined separately, while a specific combination of otherwise acceptable conditions may repeatedly produce defective products.

Production teams may also lose valuable historical knowledge when equipment is replaced, personnel change, or earlier investigations are not adequately documented.

Detailed Problem Description

There is a need for a system that integrates historical production records, machine sensor data, material batch information, maintenance logs, inspection results, and environmental measurements to identify recurring defect patterns.

The platform should compare defective and defect-free production periods, investigate combinations of contributing factors, and determine whether previously corrected problems are returning under similar circumstances.

Recommendations should be supported by historical evidence and should distinguish between statistical associations and confirmed causal relationships.

Expected Solution

Develop a manufacturing defect intelligence platform that:

- Collects and integrates machine telemetry, production parameters, quality inspections, material batches, and maintenance records.
- Maintains a searchable historical record of defects and corrective actions.
- Identifies recurring defects and hidden interactions among production variables.
- Detects combinations of conditions associated with elevated defect rates.
- Predicts potential quality problems before affected products complete production.
- Recommends preventive inspections, parameter adjustments, maintenance activities, or material checks.
- Tracks the effectiveness of corrective actions over time.

Expected Outcome

A prototype using historical or simulated production data that identifies recurring defect patterns, highlights the associated operating conditions, and recommends preventive measures.

Innovation Focus: Giving manufacturing systems a persistent operational memory so that previously encountered defects can inform future production decisions.

7. The Invisible Cargo

Domain: Logistics and Transportation | Suggested category: Software

Problem Statement

Develop an intelligent shipment reconciliation system that reconstructs the physical journey of packages and identifies discrepancies when digital handover records appear complete but actual inventory, location evidence, or delivery outcomes do not match the recorded shipment history.

Background

Logistics networks involve multiple warehouses, transportation providers, sorting facilities, delivery agents, and customer handovers. Each participant may record a shipment as received, dispatched, transferred, or delivered.

However, complete digital tracking records do not always guarantee that a package has physically followed the expected route. Scanning errors, incorrect package identification, duplicate events, mislabeled shipments, unauthorized transfers, and inventory reconciliation failures can create situations in which the tracking system reports success while the package is missing or misplaced.

Conventional tracking systems frequently display recorded events without comprehensively reconciling them against the physical movement of goods.

Detailed Problem Description

There is a need for a shipment investigation platform that correlates package identifiers, timestamps, inventory movements, route plans, scanning events, vehicle manifests, location data, and proof-of-delivery evidence.

The system should reconstruct the expected and observed shipment journeys, identify impossible or inconsistent transitions, and determine the most likely stage at which a discrepancy occurred.

It should account for clock differences, delayed scans, duplicate records, and missing data to avoid incorrectly classifying legitimate shipments as lost.

Expected Solution

Develop a cargo journey reconciliation platform that:

- Creates a unified timeline of shipment events across logistics partners.
- Compares expected package movements with recorded handovers and inventory balances.
- Detects missing scans, duplicate events, incorrect routes, unexplained inventory changes, and inconsistent delivery evidence.
- Identifies the earliest point where the shipment journey becomes inconsistent.
- Assigns discrepancy confidence scores and ranks potential investigation locations.
- Provides a visual route timeline and supporting evidence for each alert.
- Generates investigation reports for logistics personnel.

Expected Outcome

A prototype that processes simulated shipment records, reconstructs package journeys, identifies inconsistencies, and pinpoints the most likely stage requiring investigation.

Innovation Focus: Reconciling the digital history of a shipment with evidence of its physical movement, instead of treating every successful handover record as proof of delivery.

8. The Energy Budget Puzzle

Domain: Energy Management | Suggested category: Software / Hardware

Problem Statement

Develop an intelligent building energy optimization system that coordinates electricity consumption, renewable generation, battery storage, and time-varying electricity tariffs to minimize operating costs while satisfying essential energy requirements and equipment constraints.

Background

Buildings consume electricity for lighting, heating, cooling, computing, machinery, and other operations. Electricity prices and renewable energy availability may change throughout the day, while the building's demand follows its own occupancy and operational schedule.

Using electricity during the cheapest period may not always be feasible because certain processes must operate at specific times. Similarly, charging a battery at the lowest available price may be ineffective if the battery is already full, has limited capacity, or must retain energy for a later peak-demand period.

A practical energy management system must balance cost, energy availability, operational continuity, battery degradation, and sustainability objectives.

Detailed Problem Description

There is a need for an optimization platform that forecasts energy demand, renewable generation, and electricity prices and then recommends an appropriate operating schedule.

The platform should identify flexible loads that can be shifted without affecting essential activities, determine when batteries should be charged or discharged, and account for constraints such as maximum power demand, battery capacity, equipment operating windows, and reserve energy requirements.

The system should also evaluate the consequences of uncertain forecasts and provide alternative schedules when actual demand or renewable generation differs from predictions.

Expected Solution

Develop an intelligent energy scheduling platform that:

- Forecasts electricity demand, renewable generation, and tariff variations.
- Identifies flexible and non-flexible electrical loads.
- Optimizes equipment schedules and battery charging or discharging.
- Balances electricity costs, peak demand, renewable energy utilization, and emissions where suitable data is available.
- Enforces equipment limits, battery constraints, and essential energy requirements.
- Simulates alternative operating scenarios and compares their projected costs.
- Monitors actual consumption against the optimized schedule and adjusts recommendations as conditions change.

Expected Outcome

A working dashboard or simulation that compares conventional energy usage against an optimized schedule and demonstrates projected savings while maintaining essential building operations.

Innovation Focus: Optimizing when and how energy is used, rather than simply minimizing electricity consumption without considering operational needs.

9. The Product With Two Lives

Domain: Retail and Sustainability | Suggested category: Software

Problem Statement

Develop an intelligent product recovery decision-support system that evaluates returned goods and recommends the most economically and environmentally appropriate recovery pathway, including resale, repair, refurbishment, component harvesting, or recycling.

Background

Retailers and manufacturers handle products returned because of damage, defects, incorrect orders, customer dissatisfaction, warranty claims, or changes in customer requirements. These products may retain significant value, but their condition and recovery requirements vary considerably.

- Automatically reselling returned goods may result in customer dissatisfaction when repairs are necessary. Conversely, discarding or recycling products that could be economically refurbished can lead to unnecessary material waste and lost revenue.

Selecting an appropriate recovery option requires considering inspection results, repair costs, component availability, resale demand, remaining product life, environmental impact, and the time required to recover value.

Detailed Problem Description

There is a need for a product disposition platform that evaluates each returned item and compares the costs, benefits, risks, and environmental implications of possible recovery pathways.

For example, a returned electronic device may be suitable for direct resale after testing, refurbishment after replacing a component, dismantling for spare parts, or recycling when repair costs exceed the recoverable value.

The system should account for uncertainty in product condition and market demand, distinguish between estimated and confirmed recovery costs, and explain why a particular option is recommended.

Expected Solution

Develop an intelligent circular-economy decision platform that:

- Records product identity, return reason, condition, inspection results, and warranty status.
- Estimates repair, refurbishment, handling, storage, and transportation costs.
- Evaluates expected resale value, parts demand, and potential recovery revenue.
- Compares the financial and environmental implications of alternative recovery pathways.
- Recommends the most suitable disposition option based on configurable business objectives.
- Prioritizes human inspection when condition data is insufficient.
- Tracks actual recovery outcomes to improve future recommendations.

Expected Outcome

A prototype that evaluates sample returned products, compares alternative recovery pathways, and provides an explainable recommendation with estimated financial and environmental outcomes.

Innovation Focus: Treating returned products as assets with multiple possible future uses, rather than making a binary decision between resale and disposal.

10. The Supplier Illusion

Domain: Supply Chain Management | Suggested category: Software

Problem Statement

Develop a supply chain dependency intelligence platform that discovers hidden common dependencies among apparently independent suppliers and assesses how disruptions affecting shared upstream facilities, raw materials, transportation routes, or sub-suppliers could impact business continuity.

Background

Organizations frequently diversify their procurement by purchasing materials from several suppliers. However, these suppliers may depend on the same upstream manufacturer, raw-material producer, shipping port, transport corridor, energy source, or geographic production region.

As a result, a company may believe that it has diversified its supply chain while remaining exposed to a single point of failure. A disruption affecting one shared upstream dependency can simultaneously affect multiple suppliers, resulting in shortages, production delays, increased costs, and contractual failures.

Traditional supplier management systems often focus on direct vendor relationships and may not provide sufficient visibility into multi-tier supply chain dependencies.

Detailed Problem Description

There is a need for a supply chain mapping and risk analysis platform that combines procurement records, supplier disclosures, shipment information, geographic data, and other available evidence to identify shared dependencies across multiple supplier tiers.

The system should represent supplier relationships as a network, identify concentrated dependencies, and simulate the impact of potential disruptions on material availability, production schedules, and delivery commitments.

Because complete supplier information may not be available, the platform should distinguish between verified relationships, reported dependencies, and uncertain or inferred connections.

Expected Solution

Develop a supplier dependency intelligence platform that:

- Builds a multi-tier network of suppliers, sub-suppliers, facilities, materials, and transportation dependencies.
- Identifies shared upstream suppliers and other potential single points of failure.
- Calculates supplier concentration and disruption exposure for critical materials.
- Simulates scenarios such as port closures, factory shutdowns, raw-material shortages, and regional disruptions.
- Estimates the potential impact on inventory, production schedules, and procurement costs.
- Recommends alternative sourcing options, additional inventory buffers, or supplier diversification.
- Visualizes dependency paths and explains the evidence behind each risk assessment.

Expected Outcome

A prototype that maps a sample supply chain, discovers hidden common dependencies, simulates a disruption, and recommends measures to reduce the resulting operational risk.

Innovation Focus: Measuring genuine supply chain independence by examining upstream dependencies, rather than counting the number of direct suppliers.