International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group | Volume 3, Issue 3, 40-49, 2022
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V3I3P105

Original Article

Postmortem Culture in Practice: What Production Incidents Taught Us about Reliability in Insurance Tech

Lalith Sriram Datla
Software Developer at Chubb Limited, USA.

Abstract - *In the rapidly developing insurance technology sector, the reliability of systems is not only a technical necessity but a basic business requirement as well. This work presents how a strong, transparent, and responsive insurance technology system is developed by means of an efficient post-mortem culture. Examining real-world system events helps us to highlight basic and often overlooked flaws in our systems, procedures, and assumptions, therefore enabling areas of vulnerability to be found. Instead of a blaming culture, postmortems are leveraged to educate the team and lay bare various organization-wide deficiencies, such as the fragility of integrations, undefined ownership, or late alerting. We discuss recurrent issues found in our analyses, namely, overlooked edge cases and scaling failures, and how these observations, in turn, allowed us to make effective adjustments like adding automated regression checks, enriched runbooks, and smoother cross-functional communication. After every incident, not only did our teams improve their infrastructure but they also were able to build the culture of sharing and continuous improvement. The piece underscores the pivotal role of creating a psychologically safe space for open post-mortems, ensuring that failures are not only the system's driving force but also play a significant part in the growth of the people representing the next wave of leadership. By being transparent, sensitizing the people, and giving action points, and from our experience, we found out that reliability is not only about the time that the code runs properly; it is about trust, quickness, and the ability to learn from one's mistakes in order to get ahead together.*

Keywords - *Postmortem Culture, Production Incidents, Reliability, Insurance TechnologyDevops, Root Cause Analysis, Resilience Engineering, Incident Response, System Thinking, Continuous Improvement, Operational Excellence, Failure Analysis, Infrastructure Reliability, Cross-Functional Learning, Psychological Safety.*

1. Introduction

In the high-stakes area of insurance technology, dependability is not an add-on; it's a must-have feature. One particular case that can be used to amplify this point further is the fact that in the industries where downtime might cause only mild nuisances, in the case of insurance, these failures can become the reason for the failure of vital claims, the wrong calculation of policy premiums and even the violation of the regulations set up by the government. If a customer is attempting to file a claim after a car accident or a broker is in the process of finalizing the underwriting for a multimillion-dollar policy, the realization of each of these deals depends on the correct, quick, and accurate performance of the system. That means that technological solutions that support the management of risk and that are also up to date must not be susceptible to any instances of unavailability, inconsistency, or vulnerability regardless of whether the complexity of the system increases or if the number of the system's users grows.

However, achieving this kind of availability and precision is very tricky. Insurance systems have to be connected to the old databases, to the services provided by third parties and to the modern APIs but at the same time to be able to deal with the ongoing demands for compliance, the requirements for data security and the massive volumes of data. Such incidents could be the result of conditions on the periphery that are not well known, race conditions, neglected dependencies or human mistakes. Traditional compliance remediation, albeit being pivotal, mainly struggles to address the constraints of deferred system reconfiguration. This is exactly where postmortem culture makes its point.

Throughout the technology industry, postmortem analysis has emerged as a leading element of modern DevOps and reliability engineering practices. Instead of assigning blame and pointing out the guilty, postmortems in high-performance teams are organized, empathetic retrospectives that scrutinize not only what went wrong and the reasons behind the situation but also how to minimize the chances of its recurrence. Insurtech sectors have been quite vigorous in adopting this model and have mentioned the fact that post-incident reviews are not a weakness but simply signs of a more mature and resilient culture.

This article will demonstrate the evolution of the postmortem analysis from being a useful instrument in increasing the reliability of insurance technology systems. We will uncover the actual incidents that were behind the veil, exposing the weak

points in the code, architecture or processes. We will go into detail about how oversight of postmortems led us to enhance monitoring, strengthen interdepartmental communication, and realize a culture where learning is more than blaming.

Fig 1: Postmortem Culture in Practice

The article is sectioned into four main parts. The first one is about the particular reliability problems in the insurance tech environment. Then, we dive into the making of a great postmortem and how it is different from the traditional incident report. After that, we deliver the main insights and themes of real postmortems, with our teams talking about their takeaways and consequent changes. Finally, we talk about how the practice of postmortems shapes the operational state of an organization making it steady and solid and how trust is built in technical and business teams for the future.

2. Understanding Postmortem Culture

Postmortem culture, in simple terms, is the systematic examination of and learning from a failure that occurred in production. Basically, the principle that this practice follows is uncovering the weaknesses of the system and giving the system a chance to improve without any negative attitude and assigning no blame within the team. One of the key concepts of this retrospective without blame is the usage of SRE (Site Reliability Engineering) principles, which is all about a safe psychological environment, accountability that comes without shaming someone and also continuous self-refinement. It suggests that the source of most faults lies in imperfect system design rather than individuals' mistakes and thus it fosters the idea of openly discussing the events, the reasons behind them, and the ways in which the system can be made more viable.

The origins of the postmortem practices can be found in ITIL (Information Technology Infrastructure Library), which established a formalized framework for incident and problem management in IT operations of that time. However, ITIL often turned out to be unnecessary processes, too much documentation and hierarchical decision-making, sometimes even working against the company's agility. At the same time, traditional IT operations were being replaced by the modern step of software development; in particular through Agile and DevOps revolutions that significantly enriched the focus on fast delivery, constant iteration and learning. DevOps introduced the idea of tearing down walls between departments and promoting a culture of collaboration between developers and operations teams. Then the SRE team at Google came along and not only took advantage of but also standardized the qualitative reliability engineering practice of maintaining service supportability and also making the postmortems an essential part of the solution.

Despite finding postmortem culture's benefits for tech-driven companies, postmortem culture has not been very successful in regulated industries, such as insurance. The environments subject to strict governance are more likely to avoid risky situations and do not want any kind of openness around their failures, yet the situation changes hastely certainly when audits, compliance reviews, and reputational concerns are in play. Even just saying "I'm sorry" in front of a system that failed points to a threat. As a result, the reviews are half-hearted, the conversations are hidden, and the root cause is simply bypassed. As a result, this resistance often interferes with development. The same mistakes will not disappear only because nobody has had the courage to talk about them. They will be present for a long time, hidden by hasty and temporary decisions.

The most effective way to solve this problem is the leader, who will solve the problem in a short time. When top managers support postmortem practices, reframe failure as a learning experience, and demonstrate the quality of openness in the face of

adversity, the moods of their teams start to change and emulate the same kind of safety practices. The power to allow engineers to present paradoxical truths, ask probing questions, and suggest much-needed improvements is a level of safety-psychological safety which nobody can say is irrelevant in the conduct of productive postmortems. The presumed engineers get the courage to reveal extremely hard facts, they freely ask questions and they put forward proposals for various improvements. Needless to say, when this cultural change is under strict discipline for a long time, it results in the inevitable thing, such as resilient teams and stronger systems.

To ensure that there is uniformity and concrete implementation, many organizations choose standardized instruments and models for postmortems. The sections that are usually found in the templates are incident timeline, impact summary, root cause analysis, contributing factors, immediate remediation, long-term fixes, and action items with owners and deadlines. A lot of teams are using such platforms as Jira or Confluence for documentation, while others get help from the specialized systems with integrations with the incident management systems such as Jeli, Rootly, or Blameless. These instruments bring not only order to the review process but also facilitate the acquisition of trend analysis information over a period of time allowing teams to establish the repetition of such failure modes or organizational bottlenecks.

Fundamentally, the prevailing organizational culture of postmortem is about moving from reactive firefighting to proactive learning. This demands a skill set that has not only technical rigor but also emotional intelligence, trust, and leadership buy-in. For insurance tech teams exploring the territory of high stakes and heavy regulation, adoption of such a culture can be welcoming, and indeed, it can serve as a harbinger of dependability, quicker recovery, and profound organizational insights.

3. Production Incidents in Insurance Tech: Common Patterns

Production incidents in the insurance technology industry sector frequently occur in ways that not only disrupt services but at the same time impact the trust, compliance, and financial outcomes. It's different from customer applications where these encounters might appear just for a short time. The insurance systems manage the most crucial, high-stakes operations, i.e., the premium calculations, the policy renewals, the claim processing, the fulfillment of stringent regulatory requirements, and more. There has been a noticeable acceleration in digital transformation all over the industry that has been followed by the appearance of typical scenarios of failure, emphasizing the systemic gaps and also the areas that need to be seized upon; in other words, they must be thought of as opportunities for investing in the system.

The event that seems to be more common is the one that is connected with the error of the pricing engine. The most frequent reasons for this mistake are the new rating algorithms or the underwriting rules that don't behave as expected in the production. Another frequent pattern is the process of policy renewal batch errors, which, if they occur, can affect large cohorts of customers overnight. These batch jobs mostly depend on numerous interdependent services such as payment gateways, document generation systems, and policy administration platforms.

If any of the above components does not work properly, for example, a timeout from a third-party provider or a missed schema change, a failure can either cause a silent and subtle change in the data or make a policy not be renewed, which would result in delay, non-payment, and regulatory misconduct. As these processes run during non-business hours, the detection is usually delayed, and abnormalities will be reported either by customers or downstream systems.

Claim system failures are characterized as the most severe incidents, especially in the period of natural disasters or surge events when call volumes skyrocket. Even a brief period of time when the system is not working can lead to a queue of claims issues that were not processed, an increase in the manual workload, customer frustration, and a withdrawal of Service Level Agreements (SLAs). The cause of the outages is often found in back-end services that are not properly configured, missing circuit breakers, or database locking problems when the server is under stress. Sometimes the APIs of the vendors downstream introduce what we call cascading failures, which later manifest as the claims ecosystem crumbling.

There is a less noticeable, more insidious category of incidents having to do with regulatory compliance misalignments. Such incidents do not always disturb the user experience but may cause failed audits or other sanctions. For instance, a situation where the customer data in the policy systems doesn’t match the logs retained for audit trails points to data sync or transformation logic issues. These mistakes are usually due to data pipelines with weak links, schema mismatches, or lack of validation checkpoints within the ETL (extract, transform, and load) processes.

Several basic reasons are characteristic of these incidents. One significant problem that occurs repeatedly is incomplete regression testing, specifically in situations of edge cases and batch runs. In many cases, the agile delivery cycles put the emphasis on speed, and while the unit and integration tests might return positive results, the interactions between services or legacy systems

and reality are often overlooked. Flaky third-party APIs are the other major cause, in particular, in the case of insurance platforms that rely on their partners for real-time data since the issuance of motor vehicle records, credit scores, or address validation can be obtained in this way. When an API gets under stress or provides unstable answers, the workflow is likely to be disrupted.

Scarce observability only adds to the issue. The fact that several systems do not have sufficient telemetry and alerting devices is at the root of the problem, which in turn leads to slow discovery and diagnosis. Often, the logs are scattered apart between services, making it quite difficult to establish the timing of incidents. Among the deployment pipeline issues are misconfigured CI/CD tools, manual changes that bypass controls, and so on, eventually causing a lot of risk falling into the trap of releasing faulty code or configurations to production without being detected.

Incident detection and escalation gaps are other reasons for the delay in response to issues that occur. This may occur if the runbooks are not detailed, there are people who are not on call, or the people who are on call are too many and hence the situation is not taken seriously. In several instances, insurance tech teams encountered situations where it took hours before incidents were detected owing to various reasons, for example, misrouting of alerts or the fact that the symptom didn't apparently seem to be critical at the time of the occurrence. Recognizing these frequent patterns should be the initial phase toward building more dependable systems. Of course, in the next section, we will cover in detail that well-structured postmortems are necessary to give these insights life and to change the nature of every collapse into a season of learning that will fortify the system in the future.

4. Embedding Postmortems in the Workflow

In order to have postmortems that continue to be effective, they must be seen not like an option but as a part of the bigger incident response lifecycle that is ingrained and structured. In insurance tech, the interpenetration of production failures with financial operations, customer trust and compliance posture makes this pulling together vital. Through setting the rules of when, how and by whom an incident will be turned into learning and systemic improvement instead of just a fire to extinguish, the teams can be sure that every incident is a treasure to mine in the future.

4.1. Integrating Postmortems into Incident Response Protocols

The very first thing you need to do to operationalize the postmortems is to add them as the default part of the incident response workflow. All top-notch insurance tech teams make clear when a postmortem is necessary such as SEV-1 or SEV-2 incidents, non-compliance errors, or outages of certain duration. The thresholds are set down in the incident management policies so that there is no question about when the retrospective should be carried out.

As soon as the incident is resolved, the incident commander or responder records the event into a centralized tracking system (e.g., Jira, Pager Duty, or Service Now) and marks it as a postmortem review. Then a postmortem template is auto-populated that captures the timeline, services involved, and the responders' key. This automation process not only reduces the effort required but also ensures the continuity of the work once the urgency of the incident is over.

4.2. Use of Runbooks and Automated Alerts

The documentation that is written by the team of specialists in the area of emergency incidents is important to the postmortem process as well as the whole handling of the incidents. They guide the teams to perform the same steps, for instance, collecting logs, alerting interested parties, or uninstalling the release. A descriptor of the well-kept runbooks, they can also be instrumental in the postmortem process, enabling the teams to investigate whether the responders had any protocol non-adherence or had some valid concerns, and also point out the areas requiring improvement.

Automated alerts and telemetry, through monitoring tools like Data dog, New Relic, or Prometheus/Grafana, ensure that the incident diagnosis becomes a reality. This process of data collection from various systems becomes the first-hand material for an incident post-event analysis, and on top of the mentioned, it scrubs and replaces the subjectivity of human memory or hearsay narratives. In case, however, alerts are supported by the links to something specific, such as dashboards or certain traces, they make the discussion on the postmortem review more enlightened and focused on the problem actually without just the assumptions being enough.

4.3. Slack/Teams as Real-Time Incident Coordination Hubs

Online incident management has been increasingly shifting to chat communication. Slack, Microsoft Teams, and other platforms are not only used for real-time team alerts but also to create the central command center of the crisis. Among the operations, a large number of companies build temporary "war rooms" or channels that are dedicated to a particular incident, and they are naturally archived as soon as the issue is fixed. The conversations in these channels document all the decisions, escalation

paths, and workarounds that the team has attempted and all this information is later on used as input data for the postmortem analysis.

Chat is the preference for some of the teams as they match it with the implementation of chat-centric tools like Incident.io or Fire Hydrant that manage the whole incident lifecycle, from notification through communication to resolution. This merger of two components also allows responders to remain in their workflow while being acquainted with the required context of the retrospective.

4.4. Timing and Cadence: Postmortem Meetings, Documentation, and Follow-up

It is necessary to act quickly. It is important that the postmortem process starts between 24 and 48 hours after closing the trouble, while the elements are still fresh. The postmortem draft is usually authored by the incident lead or another trained responder, who also outlines the incident time, impact, detection, response, and root cause(s). Then, a structured meeting usually 30–60 minutes is arranged where all the involved parties, including engineering, product, QA, and business stakeholders if relevant, are invited.

During this meeting the team does not only read the content the aim is to enable communication, find out unclear points, and do the collaborative identification of the contributing factors. It is crucial to distinguish between **root cause** and **contributing causes**, especially in complex systems. The objective is to outline the technical defects as well as the process issues, communication lapses, and decision-making failures.

4.5. Tracking Action Items: Ownership, Visibility, and SLA for Remediation

Ultimately, the actual document of the postmortem is not the most influential result; rather, it is the **action items**. In this way, the actions should be clarified and assigned to specific members and should, therefore, be controlled for the ambiguity of the implementation and the delivery time. Teams who use a mature approach will put these action items into sprint planning or incident tracking tools, thus turning them into visible and accountable things.

Take a deployment event that caused a regression related to the absence of a test as an example, in this case, action items would be such as adding the test, enhancing the CI pipeline, and updating the runbooks. Issues of different severity levels can be easily identified by teams using such tags as a possibility. For SEV-1, the expected response should be within two weeks, and in the case of lower severity issues, the timelines can go to 30 or 60 days.

Managers and SREs could be able to keep track of the status of the action items by using the open vs. resolved action items dashboard and observe the patterns of postmortem follow-up participation trends. Such a method of evaluating things creates a sense of responsibility and eliminates the 'carcinogenic forgetfulness' disease that lets the knowledge in the postmortems be documented but never gets used.

4.6. Lessons Learned vs. Blame Assigned

One of the most delicate yet highly impactful elements of the postmortem culture lies in establishing the border between learning and blame. Blame, whether in a direct or indirect form, leads to a culture of fear, withholds openness and just as often veils the systematic flaws that led to the failure. On the contrary, since "without blame" can be read as a negative, misfortune post-mortems can establish a sense of safety, resulting in engineers being open about their mistakes, identifying a void in their knowledge, or shortcomings in their judgment. A significant point is to be objective and use non-biased language in the documentation. To illustrate, the sentence "Engineer X deployed faulty code" could be rewritten as "The deployment included code that had not been tested against legacy batch scenarios."

This change of emphasis is the process, not the person. The given reflection section at the close of a postmortem is found to be transformative. What was good? What could have been improved? What reminded us of inadequacies? Through this qualitative approach, not only are curiosity and systemic thinking nurtured, but paths to growth are also explored. Postmortems, once fully lodged and automated in workflows and made accountable and empathetic, can not only help insurance tech organizations to foster trust and resilience but can also mitigate the likelihood of human error. In an era where trust, compliance, and customer satisfaction are key, this switch from being reactive to reflective can bring about a competitive edge.

5. Case Study: A Policy Quoting System Outage

5.1. Background

During the peak of the yearly open enrollment event that sees a tremendous surge in the traffic of the insurance platform, a major outage occurred in the policy quoting system of a top insurance tech provider. This particular system was the main source of

information for those who wanted to purchase an insurance policy. In one and a half hours, the quoting function was either very slow or not available; therefore, thousands of users experienced a lack of service making brokers, direct consumers and the internal sales team feels abandoned. Definitely, the occurrence has implicated the company's earnings; furthermore, the security of their clients' trust was questioned in their most contested time of the year.

5.2. Timeline of Events

- **What Failed and How It Was Detected:** The incident kicked off at 8:05 a.m. on a Monday morning, which was the same time when the traffic flow was the most significant. They sought insurance quotes on the website, but the users reported that it took a long time for the pages to load. In just a few minutes, the error rate in the micro service that deals with the quoting module soared from the initial 0.3% to 40%, and this, in turn, caused most of the sessions to show HTTP 503s (service unavailable). The first alert was received from the latency monitors of Data dog and was further verified by the failed transaction logs feature.
- **Initial Diagnosis and Team Mobilization:** The SRE on-call was paged at 8:07 a.m., who then made quick arrangements and shared the news with the application engineering team, the customer support leads, and the incident response coordinator through Slack. At 8:12 a.m., a war room channel was set up, and the quoting service dashboard experienced a spike in both CPU and memory usage. At 8:20 a.m., when several squads went live to examine the service logs, the latest deployments caused the CPU increase. However, the addition of a deployment carried out at the end of Sunday became the key suspect.
- **Incident Resolution Steps:** At 8:30 a.m., a temporary rollback had been successfully made; the quoting service was restored to the latest stable build. But because of the wrongdoing of the marked rollback script in the deployment tool, the process was carried out manually for several Kubernetes pods, and, thus, the period was increased in length. Full-service operation had come back at 8:48 a.m., and by 9:00 a.m. the public-facing errors had been resolved. A public-facing status page update was issued, and the affected business teams were informed.

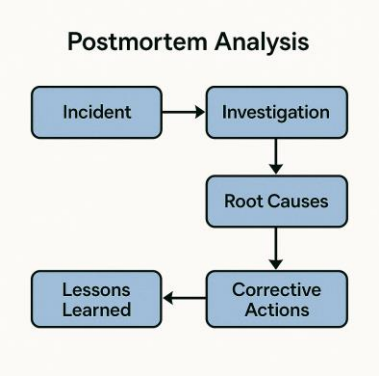

Fig 2: Postmortem Analysis

5.3. Postmortem Analysis

5.3.1. Primary and Contributing Causes

A newly released caching layer, which was originally purposed to boost quote calculation speeds, was discovered to be the root cause. A logic bug in the implemented code was leading to an infinite loop when the input from a multi-vehicle quote was not accepted, especially in situations that are called open enrollment. The code had undergone unit testing and was working properly at that point but there were no tests on this edge case for regression. Also, the deployment of the code was conducted without canary rollout checks thus; all users were made vulnerable at the same time.

Contributing causes included:

- Insufficient test coverage for multi-vehicle edge cases
- Absence of a canary deployment or phased rollout strategy
- Inadequate rollback automation
- Overly high alert thresholds that delayed incident detection

5.3.2. Systems Impacted and User Pain Points

The quoting micro service was directly affected, but the cascading effect impacted several systems, including:

- CRM platforms where brokers tracked customer quotes
- Internal dashboards used for sales forecasting
- Customer support systems handling complaints in real-time

Users reported frustration at the inability to retrieve quotes, abandoned shopping sessions, and duplicated quote attempts that later created reconciliation issues in the database.

5.3.3. Technical Debt Uncovered

It should be noted that the event touched upon different aspects of technical debt the company has. Uncovering exposed layers of technical debt, under which legacy CI/CD scripts had not been audited in months, relying on manual rollback steps without standard documentation, and using observability metrics that are not shared across squads were the new cases. On top of that, quotes did not have a high level of detail that could help in identifying the exact cause of the failure immediately; thus, it was very difficult to debug the problem during the initial stages of the investigation.

5.4. Remediation Actions

- **Alert Threshold Tuning**: SRE teams have made necessary amendments after evaluating the situation by lowering the latency and error rate alert thresholds for high-traffic services. Simultaneously, they attached extra synthetic checks to the system that were designed to simulate the most typical quote scenarios and generate preemptive throwaway lines.
- **Rollback Strategy Enhancements:** The CI/CD tool has been equipped with a fresh automated rollback workflow, which consists of the blue-green deployment scripts for the reversion of the process and the pre-configured rollback templates. From the present, all further deployments are subject to a rollback validation before being pushed into the QA environment.
- **CI/CD Pipeline Fix**: The CI approach now included regression tests stressing common quotation forms especially those involving many vehicles or large input quantities. Linting and policy reviews improved to stop releases below advised test coverage levels.
- **Knowledge Sharing Across Squads**: Not only was the problem looked at with the main technical team but also with related groups. Emphasizing lessons learned and demonstrating the new rollback technique, a knowledge-sharing session occurred during the weekly engineering town hall. An extensive internal "failure wiki" page was developed to document these and like conditions for forthcoming onboarding and training requirements.

5.5. Organizational Impact

- **Leadership Communication**: Engineers' management openly spoke the truth to their own teams and the most important top-level staff in engineering. The tone was that of taking responsibility and the wish to learn, thus emphasizing the absence of systems rather than the clumsiness of individuals. In the end, the CTO's follow-up letter was a list of actions to be done and a reaffirmation of the top management's readiness for retrospectives without the blame.
- **Customer Trust:** The outage may have been short, but the fact that it happened during a crucial period lent it an even greater significance. The teams assisting the clients informed those affected users about it in the most transparent way and with several goodwill gestures, including credits on their accounts and redeemer windows. Proactive communication was a positive step to keep the company’s reputation and save it from losing out due to reliability
- **Engineering Morale:** At first, the team was under a lot of stress due to the issue. However, the postmortem culture, which is not just about the technical side but also about the emotional health of the team, led to the entire group seeing the event as a shared affair and thus as a growth opportunity. A number of engineers mentioned that the well-ordered brief and the actual visible signs of leadership support that were present made them feel more self-confident in their roles and showed them that this was the right way to increase the resilience of the platform.

6. Key Takeaways and Cultural Shifts

Strong postmortem culture development inside the business motivated not only technical developments but also significant structural and cultural transformations among several departments. Seeing events as teaching opportunities instead of punishing ones helped the organization to enhance its infrastructure, promote team cohesion, and create more cross-functional trust. Although the transformation was slow, the general outcomes in terms of dependability and organizational style were impressive.

6.1. Systemic Changes Driven by Postmortem Culture

One big and obvious benefit was the great increase in test coverage regular postmortems produced. Tests have focused heavily on fundamental happy-path scenarios and unit-level verification while mostly ignoring system-wide interactions and edge conditions. Some flaws such as unverified multi-vehicle quotation logic or erased policy renewal edge cases that truly caused failures were found via incident investigations. The engineering company defined minimum coverage limits for critical services and developed more strict regression testing techniques. Regularly carried out based on post-mortem situations, new tests helped to promote a good cycle of learning and avoidance.

One important development was the application of chaotic engineering experiments. Driven by ongoing issues with unstable dependencies and poor failover control, SRE teams started the intentional simulation of failure occurrences in production environments. Under stress, network split, or dependence timeout, controlled tests assessed the dependability of key micro services.
This improved system durability and provided engineers with more production-ready confidence.

Moreover, the group made great advancements in dependability mapping and documentation accuracy. Postmortems often showed issues resulting from unknown or unreported upstream and downstream interactions. Teams answered by starting to maintain dynamic service dependent maps and demanding modifications in required documentation during the deployment process. Included among the engineering knowledge base were operational runbooks, API contracts, and diagrams, improving the efficiency of onboarding and incident response.

6.2. Cultural Wins

Apart from the technical outcomes, the postmortem approach created significant cultural changes. The biggest shift was from individual guilt to group responsibility. Postmortem investigations today focus more on "what systematic factors permitted this occurrence" than on "who deployed the code." The company created an environment whereby workers felt free to communicate honestly, exposes knowledge gaps, and asks difficult questions without fearing consequences by normalizing the idea that people make mistakes and that good systems allow for such blunders.

This change made honest conversation possible and fostered a growing psychological safety. Early on, spotters of problems realized that their fixes would assist rather than create confusion. Junior developers were urged to express their ideas and challenge presumptions during incident evaluations. This transparency gradually crept into planning meetings, retrospectives, and sprint reviews, therefore improving team interactions to be more honest and useful.

In the end, retrospectives earned respect from people outside of engineering unexpected but helpful. Attending postmortems, product managers, compliance authorities, and customer success executives started noticing through these interactions a unique insight into the operation and defects of systems, products, and regulations. They became more open to engineering constraints and actively assisted with the fixes. This cross-functional alignment guaranteed that follow-up tasks were not deprioritized in the next sprints and that system reliability was seen as a shared responsibility instead of only an engineering one.

7. Conclusion and Lesson Learned

7.1 Conclusion:

In insurance technology, dependability is equally a question of systems and philosophy. By means of continuous postmortem analysis, this company has transformed production errors from disruptive events into useful teaching and development tools. Every failed project exposed flaws in test coverage, deployment plans, monitoring, and communication; each postmortem provided a thorough, psychologically safe way to address these problems. Apart from reducing the frequency and severity of accidents, the shift from reactive solutions to proactive, methodical improvements promoted more coherent, capable teams. This encounter leads one to recognize, essentially, the requirement of institutionalizing information. Teams built an increasing store of operational intelligence by accurately and simply documenting the backdrop, underlying causes, and corrective actions of every event. New hires had an onboarding tool as well as a reference during design reviews and a basis for improving architecture from this growing knowledge base.

The approach made sure that harsh lessons were neither skipped over or repeated, therefore turning transient events into long-lasting organizational memory.Post-mortem analyses have improved the dependability of the infrastructure supporting insurance technologies. CI/CD techniques, dependability awareness, and resilience testing all show notably superior results from retroactive insights. Crucially, they changed how teams saw failure not as something to totally avoid but rather as a chance for brave and creative learning. Developing a postmortem-driven learning culture goes beyond basic avoidance of the next disturbance. It affects how resilient and flexible one may be under uncertainty. Resilient companies stand out in an industry defined by ongoing progress, open communication, fast change and great risks for their possibility of swift recovery. Sensibly done postmortems help to create such a culture.

7.2. Lesson Learned

7.2.1. Silent Failures Are the Most Dangerous

Many negative occurrences began not from total outages but from undiscovered partial failures that is, where a service appeared operational but generated erroneous or inadequate data (e.g., absent policy endorsements or unacknowledged quote

rejection). These showed teams that excellent observability beyond simple uptime measurements is absolutely essential. Early degradation detection depends now mostly on loggers, distributed tracing, and canary analysis.

7.2.2. Recovery Time Matters More Than Uptime

Clients exhibit better tolerance for sporadic outages than for delayed or disorganized recovery methods, especially in the spheres of healthcare and insurance for education. Events underscored the requirement of chaotic simulations, automatic rollback systems, and operational runbooks. Mean time to recovery (MTTR) became the most crucial statistic because it led to real-time alarm correlation and increased infrastructure spending on self-healing.

7.2.3. Over-Customization Leads to Fragility

Custom client rules for claims, billing, or enrollment usually bring untested code routes under load or edge-case scenarios. Postmortem studies exposed that even with excellent intentions, too particular client logic compromised system dependability. The lesson is to completely test every client option in the pre-production environment first priority over hard-coded branching logic and provide configuration-driven flexibility top priority.

7.2.4. Cultural Transparency Drives Systemic Improvements

Postmortems were most effective when conducted publicly, without credit, and for many purposes, all of which were shared. Teams who considered events as teaching opportunities rather than points of responsibility identified design, communication, and ownership errors as their fundamental causes. Establishing a real "postmortem culture" made it possible for all-encompassing platform-wide fixes to be rapidly applied rather than simply localized improvements.

7.2.5. SLIs and SLOs Must Reflect Real User Pain

Originally focused on availability and latency at the edge, first service-level indicators (SLIs) showed that downstream difficulties not apparent in those measurements were causing failure of essential business actions such as "submit claim" or "update dependent." Based on comprehensive user experiences, teams acquired the ability to restructure Service Level Objectives (SLOs) and coordinate them with legally mandated procedures inside insurance systems.

References

[1] Sheaff, Michael T., and Deborah J. Hopster. *Post mortem technique handbook*. Springer Science & Business Media, 2005.

[2] Lundberg, George D. "Low-tech autopsies in the era of high-tech medicine: continued value for quality assurance and patient safety." *Jama* 280.14 (1998): 1273-1274.

[3] Yardley, Iain E., Andrew Carson-Stevens, and Liam J. Donaldson. "Serious incidents after death: content analysis of incidents reported to a national database." *Journal of the Royal Society of Medicine* 111.2 (2018): 57-64.

[4] Paté-Cornell, M. Elisabeth. "Learning from the piper alpha accident: A postmortem analysis of technical and organizational factors." *Risk analysis* 13.2 (1993): 215-232.

[5] Kupunarapu, Sujith Kumar. "AI-Enhanced Rail Network Optimization: Dynamic Route Planning and Traffic Flow Management." *International Journal of Science And Engineering* 7.3 (2021): 87-95.

[6] Kim, Gene, et al. *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations*. It Revolution, 2021.

[7] Talakola, Swetha, and Sai Prasad Veluru. "How Microsoft Power BI Elevates Financial Reporting Accuracy and Efficiency". *Newark Journal of Human-Centric AI and Robotics Interaction*, vol. 2, Feb. 2022, pp. 301-23

[8] Atluri, Anusha. "Insights from Large-Scale Oracle HCM Implementations: Key Learnings and Success Strategies ". *Los Angeles Journal of Intelligent Systems and Pattern Recognition*, vol. 1, Dec. 2021, pp. 171-89

[9] O'Mara Sage, Elizabeth, et al. "Investigating the feasibility of child mortality surveillance with postmortem tissue sampling: generating constructs and variables to strengthen validity and reliability in qualitative research." *Clinical Infectious Diseases* 69.Supplement_4 (2019): S291-S301.

[10] Beyer, Betsy, et al. *The site reliability workbook: practical ways to implement SRE*. " O'Reilly Media, Inc.," 2018.

[11] Yasodhara Varma, and Manivannan Kothandaraman. "Leveraging Graph ML for Real-Time Recommendation Systems in Financial Services". *Essex Journal of AI Ethics and Responsible Innovation*, vol. 1, Oct. 2021, pp. 105-28

[12] Atluri, Anusha. "Redefining HR Automation: Oracle HCM's Impact on Workforce Efficiency and Productivity". *American Journal of Data Science and Artificial Intelligence Innovations*, vol. 1, June 2021, pp. 443-6

[13] Jasanoff, Sheila. *Science at the bar: Law, science, and technology in America*. Vol. 9. Harvard University Press, 1997.

[14] Sangaraju, Varun Varma. "AI-Augmented Test Automation: Leveraging Selenium, Cucumber, and Cypress for Scalable Testing." *International Journal of Science And Engineering* 7 (2021): 59-68.

[15] Hill, Rolla B., and Robert E. Anderson. *The autopsy—medical practice and public policy*. Elsevier, 2016.

[16] Paidy, Pavan. "Post-SolarWinds Breach: Securing the Software Supply Chain". *Newark Journal of Human-Centric AI and Robotics Interaction*, vol. 1, June 2021, pp. 153-74
[17] Sangeeta Anand, and Sumeet Sharma. "Leveraging AI-Driven Data Engineering to Detect Anomalies in CHIP Claims". *Los Angeles Journal of Intelligent Systems and Pattern Recognition*, vol. 1, Apr. 2021, pp. 35-55
[18] Murphy, Niall Richard, et al. *Site Reliability Engineering: How Google Runs Production Systems*. " O'Reilly Media, Inc.," 2016.
[19] Talakola, Swetha. "The Importance of Mobile Apps in Scan and Go Point of Sale (POS) Solutions". *American Journal of Data Science and Artificial Intelligence Innovations*, vol. 1, Sept. 2021, pp. 464-8
[20] Fuller, Joanna Kotcher, and Joanna Ruth Fuller. *Surgical technology: Principles and practice*. Elsevier Health Sciences, 2012.
[21] Veluru, Sai Prasad. "Real-Time Model Feedback Loops: Closing the MLOps Gap With Flink-Based Pipelines". American Journal of Data Science and Artificial Intelligence Innovations, vol. 1, Feb. 2021, pp. 485-11
[22] Ali Asghar Mehdi Syed. "Cost Optimization in AWS Infrastructure: Analyzing Best Practices for Enterprise Cost Reduction". *JOURNAL OF RECENT TRENDS IN COMPUTER SCIENCE AND ENGINEERING (JRTCSE)*, vol. 9, no. 2, July 2021, pp. 31-46
[23] Bennett, W. Lance, and Martha S. Feldman. *Reconstructing reality in the courtroom: Justice and judgment in American culture*. Quid Pro Books, 2014.
[24] Veluru, Sai Prasad, and Swetha Talakola. "Edge-Optimized Data Pipelines: Engineering for Low-Latency AI Processing". *Newark Journal of Human-Centric AI and Robotics Interaction*, vol. 1, Apr. 2021, pp. 132-5
[25] . Carlson, Matt, Sue Robinson, and Seth C. Lewis. *News after Trump: Journalism's crisis of relevance in a changed media culture*. Oxford University Press, 2021.
[26] Ali Asghar Mehdi Syed, and Shujat Ali. "Evolution of Backup and Disaster Recovery Solutions in Cloud Computing: Trends, Challenges, and Future Directions". *JOURNAL OF RECENT TRENDS IN COMPUTER SCIENCE AND ENGINEERING (JRTCSE)*, vol. 9, no. 2, Sept. 2021, pp. 56-71
[27] Vasanta Kumar Tarra. "Policyholder Retention and Churn Prediction". *JOURNAL OF RECENT TRENDS IN COMPUTER SCIENCE AND ENGINEERING (JRTCSE)*, vol. 10, no. 1, May 2022, pp. 89-103
[28] Paidy, Pavan. "Log4Shell Threat Response: Detection, Exploitation, and Mitigation". *American Journal of Data Science and Artificial Intelligence Innovations*, vol. 1, Dec. 2021, pp. 534-55
[29] Postman, Neil. *Technopoly: The surrender of culture to technology*. Vintage, 2011.
[30] Pamies, David, et al. "Advanced good cell culture practice for human primary, stem cell-derived and organoid models as well as microphysiological systems." (2018).