DynaCov

Safety Automation

Product Specification

Document Title	DynaCov Safety Automation — Product Specification
Version	1.0
Status	Released
Date	April 2026
Product	DynaCov Safety Automation
Platform	Red Hat OpenShift (dev / staging namespaces)
Owner	DynaCov, Inc.

> DynaCov Safety Automation is a resilience validation platform built specifically for OpenShift development and staging environments. It requires no elevated privileges, no cluster-wide operator installation, and no SRE or platform team involvement. Developers and testers install and run it themselves — and every test produces a PDF/A-1b audit report that is legally archivable and regulator-ready from day one.

Table of Contents

1. Executive Summary
2. Product Overview
3. Architecture & Deployment Model
4. Competitive Differentiators
5. Core Capabilities
6. Test Scenarios
7. Safety Automation Framework
8. Observability & Reporting
9. Competitive Comparison
10. Target Markets & Use Cases
11. Constraints & Scope
12. Glossary

1. Executive Summary

Modern enterprise software environments rely heavily on automated remediation — self-healing scripts, restart policies, failover logic, and scaling rules — to maintain service continuity. These automation systems operate largely without oversight. When they work correctly, they accelerate recovery. When they do not, they can amplify failures, extend outages, and obscure root causes — often while appearing to function normally in monitoring dashboards.

DynaCov Safety Automation addresses this gap by validating whether automated responses actually improve system outcomes or make them worse. It introduces a Control Plane Validation layer — a structured methodology for proving, not assuming, that automation is safe to trust.

The platform is designed for the reality of regulated enterprise OpenShift environments. It installs in a development or staging namespace without elevated privileges, works immediately with three pre-built test scenarios, surfaces results through a preconfigured Grafana dashboard, and generates PDF/A-1b audit reports accepted by regulators including the FCA, OCC, and SOC 2 auditors.

> Core value proposition: Stop deploying untested authority. DynaCov proves — does not hope — that your automation helps rather than harms.

2. Product Overview

2.1 Product Identity

Product Name: DynaCov Safety Automation

Product Category: Resilience Validation / Automation Safety / Quality Engineering

Deployment Target: Red Hat OpenShift — development and staging namespaces

Primary Users: Software developers, QA engineers, test automation engineers

Installation Model: Namespace-scoped, self-service, no operator required

2.2 Problem Statement

Automated remediation systems in Kubernetes and OpenShift environments are trusted by default. LivenessProbes trigger restarts. HorizontalPodAutoscalers add capacity. Circuit breakers reroute traffic. Failover rules promote secondary instances. Each of these actions executes autonomously, often within milliseconds, based on logic that was configured at a point in time and has never been independently validated under realistic failure conditions.

The consequences of unvalidated automation are well-documented. A restart policy designed to clear a deadlocked thread can trigger a crash-loop when the underlying database connection pool is saturated. A scaling policy designed to handle traffic spikes can exhaust IP address pools and prevent healthy pods from joining the cluster. A circuit breaker designed to isolate a failing service can inadvertently block healthy downstream dependencies.

In each case, the automation executes as designed. The failure occurs because the automation was never proven safe for the context in which it fired. DynaCov exists to close this gap.

2.3 Product Vision

DynaCov Safety Automation reframes resilience testing from infrastructure risk management into a developer quality concern. Just as unit tests prove application logic, Safety Automation proves control plane logic — the scripts, policies, and rules that govern how systems respond to failure.

The platform is positioned alongside test suites and integration pipelines, not alongside chaos engineering platforms that require cluster administrators and SRE teams. It is installed by developers, run by developers, and produces output that serves both developers (immediate feedback) and compliance functions (audit-ready documentation).

3. Architecture & Deployment Model

3.1 Deployment Topology

DynaCov Safety Automation is deployed as a standard application within an OpenShift namespace. It does not require a cluster-wide operator, ClusterRole or ClusterRoleBinding, access to namespaces outside its own, or any form of elevated RBAC privileges. Installation and operation are scoped entirely to the namespace in which the product is deployed.

> Architectural principle: DynaCov is namespace-scoped by design, not by configuration. There is no privileged mode. There is no cluster-admin path. This is a structural constraint, not a policy setting.

3.2 Supported Environments

- Development namespaces on Red Hat OpenShift
- Staging namespaces on Red Hat OpenShift

3.3 Excluded Environments

> NOTE: DynaCov Safety Automation is explicitly not designed for and must not be deployed in production namespaces. This constraint is architectural. The product operates exclusively in pre-production environments where failure injection and remediation testing can be conducted safely.

3.4 Installation Requirements

- Red Hat OpenShift access to a development or staging namespace
- Namespace-level permissions (standard developer access — no cluster-admin required)
- No cluster-wide operator installation
- No SRE, platform engineer, or operations team involvement required
- Installable by software developers, QA engineers, and test automation engineers

3.5 Integration Points

- **Grafana** — Preconfigured dashboard included — no additional setup required
- **OpenShift service mesh / HTTP routing** — Used for latency and error injection scenarios
- **Pod lifecycle management** — Used for Pod Kill and Restart scenario

4. Competitive Differentiators

The following six differentiators represent DynaCov Safety Automation's primary competitive advantages over existing resilience testing and chaos engineering platforms. They are ranked by the width of the competitive gap — the degree to which no existing alternative delivers an equivalent capability.

4.1 Zero Elevated Privileges [Widest Gap]

Every major competing platform — Gremlin, Chaos Mesh, LitmusChaos, AWS FIS, Harness Chaos Engineering — requires cluster-admin or elevated RBAC permissions and, in most cases, a cluster-wide operator installation. In regulated OpenShift environments, this creates a mandatory escalation path through platform and security teams, introducing procurement delays of days to weeks before any experiment can run.

DynaCov eliminates this requirement entirely. The product installs and operates within a single OpenShift namespace using standard developer-level permissions. No RBAC escalation of any kind is required at any stage of installation or operation.

4.2 Developer Self-Service [Wide Gap]

Competing platforms are operationally positioned as infrastructure tools: they are provisioned by platform teams, maintained by SRE teams, and governed through change management workflows. For development teams in regulated enterprises, this means that running a chaos experiment requires a ticket, an approval, a scheduled maintenance window, and specialist involvement.

DynaCov is positioned as a developer quality tool. A developer or tester installs it directly in their own namespace, in their own development environment, on their own schedule — without involving any other team. This shifts the activation cost of safety validation from weeks to minutes and embeds quality assurance of control plane logic into the development workflow.

4.3 Production-Safe by Architecture [Strategic Differentiator]

The most common procurement objection to chaos engineering and resilience testing tools is the risk of accidental production impact. Competing platforms typically address this through policy-based guardrails — configuration settings, access controls, and runbook procedures that prevent production execution. These controls can be misconfigured, bypassed, or overridden under operational pressure.

DynaCov's production safety is architectural. The product is designed to operate in development and staging environments and has no production execution path. This eliminates the procurement objection entirely — not by promising a policy will hold, but by making production execution structurally impossible.

4.4 PDF/A-1b Audit-Ready Reports [Widest Gap]

No competing chaos engineering or resilience testing platform produces output in PDF/A-1b format. Competing tools generate dashboards, log files, and pipeline artifacts — none of which satisfy the ISO 19005-1 archival standard required for evidentiary records in regulatory contexts.

PDF/A-1b is the format required by regulators including the FCA, OCC, and internal audit functions conducting SOC 2 and ISO 27001 assessments. DynaCov generates a Decision Validation Log in this format after every test run — a timestamped, immutable, regulator-ready record stating definitively whether each automated response helped or harmed during the test scenario.

This capability directly addresses the post-incident documentation burden that follows automation failures in regulated industries. Rather than reconstructing what happened from logs and dashboards after the fact, DynaCov produces the record in advance — during development and staging — as proof that automation has been validated.

4.5 Preconfigured Grafana Dashboard [Moderate Gap]

Most resilience testing tools require significant configuration work before operational metrics are visible. DynaCov ships with a preconfigured Grafana dashboard that displays results immediately upon first test execution. No dashboard design, data source configuration, or metric mapping is required. The first experiment produces immediately readable output.

This substantially reduces the activation cost for initial adoption and eliminates a common point of abandonment — where teams install a tool but cannot quickly see meaningful output and disengage before completing a full test cycle.

4.6 Three Scenarios Out of the Box [Moderate Gap]

Competing platforms typically ship with either no pre-built scenarios (requiring teams to design experiments from scratch) or with an overwhelming catalogue of experiment types that creates cognitive overhead and requires expert guidance to navigate.

DynaCov ships with three curated test scenarios that cover the most common and highest-impact failure patterns in microservices environments: pod kill and restart, HTTP dependency latency injection, and HTTP dependency error injection. Each scenario is pre-configured, immediately executable, and directly relevant to the control plane validation use case. Teams do not need a chaos engineering specialist to get value on day one.

5. Core Capabilities

5.1 Control Plane Validation

DynaCov Safety Automation validates the efficacy of automated responses — not merely whether infrastructure survives a fault injection, but whether the automated remediation that fires in response actually improves system state. This is the foundational capability that differentiates DynaCov from conventional chaos engineering.

Validation is performed through counterfactual testing: the same failure scenario is run twice — once with automation active and once with automation suppressed. The delta between these two outcomes constitutes the proof of automation efficacy. If the automated response improves system state relative to the baseline, it is validated. If it does not, it is flagged as requiring remediation before the system proceeds to production.

5.2 Stability Quotient Measurement

After each automated response fires during a test scenario, DynaCov measures the Stability Quotient — a composite metric that determines whether the system moved toward equilibrium or toward greater volatility following the automated action. A declining Stability Quotient is the signature of thrashing, crash-loops, and positive feedback failure patterns.

5.3 Blast Radius Mapping

DynaCov maps the side-effects of automated responses across the microservices mesh — identifying whether automated actions intended to isolate a failing component inadvertently affect healthy downstream dependencies. This addresses the common failure pattern where circuit breakers, scaling policies, or failover logic export stress to other layers of the infrastructure.

5.4 Decision Validation Logging

Every automated action taken during a test scenario — and the system's measured response to that action — is recorded in a Decision Validation Log. This log is generated in PDF/A-1b format and constitutes the primary audit artifact produced by DynaCov. Each log entry records the action taken, the time at which it was taken, the system metrics observed before and after, and the validation verdict: whether the action helped, was neutral, or harmed.

6. Test Scenarios

DynaCov Safety Automation ships with three pre-configured test scenarios. Each scenario is ready to execute immediately upon installation with no additional configuration required. Scenarios are designed to cover the three most common and highest-impact automated remediation failure patterns in OpenShift microservices environments.

Test Scenario	What It Measures	Automation Safety Question Answered
Pod Kill & Restart	Whether automated restart logic resolves the fault or initiates a crash-loop under saturation conditions	Does restarting the pod help or amplify the failure?
HTTP Latency Injection	Whether the system's automation responds correctly to degraded upstream HTTP dependencies, including retry storms and timeout escalation	Does the automation stabilise or worsen latency under dependency degradation?
HTTP Error Injection	Whether circuit-breaker logic, failover routing, and error-handling automation behaves as expected under hard dependency failure	Does the failover automation contain the blast radius or expand it?

6.1 Scenario 1: Pod Kill and Restart Test

This scenario terminates a targeted pod and observes whether the automated restart policy resolves the simulated fault or initiates a crash-loop under resource contention conditions. It directly tests the efficacy of LivenessProbe-driven restart logic and the threshold at which restart automation transitions from beneficial to harmful.

Specific failure patterns detected: crash-loop creation under DB connection saturation, thrashing patterns where CPU consumption from restart cycles exceeds the original fault, and the Zombie State condition where a restarted service appears healthy but cannot process requests due to upstream resource exhaustion.

6.2 Scenario 2: HTTP Dependency Latency Injection

This scenario injects controlled latency into HTTP calls between services and observes whether the automation responsible for managing degraded dependency conditions — including retry logic, timeout escalation, and circuit breaker activation — responds in a manner that stabilises or worsens system state.

Specific failure patterns detected: retry storms where aggressive retry logic amplifies latency across the dependency graph, timeout escalation cascades where upstream timeouts propagate downstream, and circuit breaker premature activation that isolates services before recovery has been attempted.

6.3 Scenario 3: HTTP Dependency Error Injection

This scenario injects HTTP error responses (5xx class) from upstream dependencies and observes whether circuit breaker logic, failover routing, and error-handling automation correctly contains the blast radius or inadvertently expands it to healthy downstream services.

Specific failure patterns detected: split-brain conditions arising from inconsistent failover decisions across service instances, blast radius expansion where error-handling logic blocks healthy downstream dependency paths, and false-positive circuit breaker activation that treats transient errors as systemic failures.

7. Safety Automation Framework

DynaCov Safety Automation is built around a three-pillar validation framework. Each pillar addresses a distinct dimension of automation safety. Together they provide a comprehensive assessment of whether automated remediation is fit for production deployment.

Pillar	Definition	Barclays Example
1. Efficacy	Was the automated response the right tool? Did it produce a net-positive resilience gain?	Restart loop detected within first recovery cycle — flag raised before crash-loop fully developed
2. Stability	Did the system return to steady state after automation fired, or enter a more volatile state?	Stability Quotient declining after each restart — oscillation pattern identified and halted
3. Blast Radius	Did the automated response remain contained, or did it export stress to other layers?	Scaling policy contention with recovery traffic flagged — HMRC payment flows protected

7.1 Pillar 1: Efficacy of Automated Response

Efficacy validation asks whether the automated response was the right tool for the failure condition it encountered. It measures the net-positive or net-negative resilience gain produced by each automated action. Positive validation confirms that the automation produced measurable improvement in system state. Negative validation — the Zombie State or crash-loop pattern — confirms that the automation consumed resources without improving conditions and may actively worsen the failure trajectory.

7.2 Pillar 2: Stability and Damping Analysis

Stability validation assesses whether the system returns to a steady state following automated intervention, or whether the automated action introduces oscillation, thrashing, or secondary failure cascades. The Stability Quotient measurement tracks system behaviour across multiple recovery cycles, identifying positive feedback loops — where each automated response triggers conditions that make the next response more likely to fail.

7.3 Pillar 3: Blast Radius and Side-Effect Mapping

Blast radius validation confirms that automated responses are contained within their intended scope and do not export stress to adjacent services or infrastructure layers. It tests whether circuit breakers correctly isolate failing components, whether scaling policies for one service starve resources required by another, and whether failover logic maintains data consistency rather than creating split-brain conditions.

8. Observability & Reporting

8.1 Preconfigured Grafana Dashboard

DynaCov ships with a fully preconfigured Grafana dashboard that requires no setup, no data source configuration, and no metric mapping. The dashboard is operational immediately upon the completion of the first test scenario and displays:

- Stability Quotient over time for each test run
- Automated action timeline with before/after metric comparison
- Blast radius visualisation across the service dependency graph
- Pass/fail verdict for each of the three validation pillars
- Comparative view: automation active versus automation suppressed

8.2 PDF/A-1b Decision Validation Log

After every test scenario execution, DynaCov generates a Decision Validation Log in PDF/A-1b format. This is an ISO 19005-1 compliant archival document suitable for long-term preservation and regulatory submission.

Each Decision Validation Log includes:

- Test scenario identifier, execution timestamp, and environment details
- Full timeline of automated actions fired during the scenario
- Pre- and post-action system metrics for each automated response
- Stability Quotient measurement and trend
- Blast radius assessment with affected service enumeration
- Pillar-by-pillar validation verdict with supporting evidence
- Overall safety verdict: automation validated, requires tuning, or requires remediation

> Compliance note: PDF/A-1b is the format required by the FCA, OCC, and most internal audit functions conducting SOC 2, ISO 27001, and DORA compliance assessments. DynaCov is the only resilience validation platform that generates this format natively.

9. Competitive Comparison

The following table provides a direct comparison of DynaCov Safety Automation against the primary alternatives evaluated in enterprise OpenShift environments. Assessments reflect the default out-of-the-box capabilities of each platform without additional custom development or specialist configuration.

Capability	Gremlin	LitmusChaos	Harness Chaos	DynaCov
No elevated privileges required	No	No	No	**Yes**
Developer self-install	No	No	No	**Yes**
Namespace-scoped (no cluster-wide)	No	Partial	Partial	**Yes**
Production-safe by architecture	No	No	No	**Yes**
PDF/A-1b audit-ready reports	No	No	No	**Yes**
Preconfigured Grafana dashboard	Partial	No	Partial	**Yes**
Works out of the box	Partial	Partial	Partial	**Yes**
OpenShift native install	No	Partial	Partial	**Yes**

Partial: capability exists but requires significant additional configuration, elevated permissions, or specialist involvement to achieve.

10. Target Markets & Use Cases

10.1 Financial Services & Regulated Enterprises

The primary target market. Financial services organisations running OpenShift — banks, insurers, wealth management firms, and fintechs — face the combination of strict regulatory requirements, complex microservices architectures, and strong disincentives to deploy unvalidated automation into production. DynaCov's PDF/A-1b output and production-safe architecture are specifically relevant to FCA, OCC, Basel III, GDPR, and DORA compliance contexts.

10.2 Healthcare & Life Sciences

Safety-critical software in clinical, laboratory, and life sciences environments carries zero tolerance for automation-amplified failures. The counterfactual validation methodology and audit-ready reporting provide the evidence trail required by FDA, HIPAA, and clinical software validation frameworks.

10.3 Government & Public Sector

Government agencies operating OpenShift environments benefit from DynaCov's namespace-scoped architecture, which fits naturally within security-conscious environments that restrict cluster-admin access and require audit documentation for all system changes.

10.4 Cloud-Native Development Teams

Any development team running microservices on OpenShift with automated remediation in place — particularly teams managing distributed systems with Kubernetes-native scaling, circuit breakers, and health probe logic — represents a natural fit regardless of industry vertical.

10.5 Legacy Modernisation Programs

Organisations migrating microservices or APIs from legacy mainframe or monolithic environments to OpenShift need to validate that the automated management logic in the target environment behaves correctly before each migration milestone reaches production. DynaCov provides this validation as a native part of the OpenShift development workflow.

11. Constraints & Scope

11.1 In Scope

- OpenShift development and staging namespace deployments
- Pod-level fault injection and remediation validation
- HTTP service dependency latency and error injection
- Automated response efficacy measurement
- Stability Quotient and blast radius assessment
- PDF/A-1b audit report generation
- Preconfigured Grafana dashboard output

11.2 Out of Scope

- Production namespace deployment or execution
- Cluster-wide operator installation or cluster-level permissions
- Database-level fault injection
- Network partition simulation at the infrastructure layer
- Physical infrastructure failure simulation
- Load testing or performance benchmarking (distinct use case from resilience validation)

> NOTE: DynaCov Safety Automation is a pre-production validation tool. Its value is in proving automation safety before production deployment — not in testing production systems. This scope constraint is intentional and is the source of the production-safety competitive advantage described in Section 4.3.

12. Glossary

- **Blast Radius** — The scope of system components affected by an automated response, including unintended side-effects on adjacent services
- **Control Plane Validation** — The process of verifying that automated remediation logic produces correct outcomes under realistic failure conditions
- **Counterfactual Testing** — A validation methodology that compares system behaviour with automation active versus automation suppressed, to isolate the causal effect of the automated response
- **Crash-Loop** — A failure pattern in which an automated restart policy continuously restarts a service that cannot recover due to an unresolved underlying condition, consuming increasing resources with each cycle
- **Decision Validation Log** — The primary audit artifact produced by DynaCov Safety Automation — a timestamped, PDF/A-1b formatted record of every automated action taken during a test scenario and the measured system response to each action
- **PDF/A-1b** — ISO 19005-1 compliant archival PDF format designed for long-term document preservation. Required by regulators including the FCA, OCC, and internal audit functions conducting SOC 2 and ISO 27001 assessments
- **Stability Quotient** — A composite metric measuring whether a system moves toward equilibrium or toward greater volatility following an automated response. A declining Stability Quotient indicates thrashing or oscillation
- **Zombie State** — A failure pattern in which an automated restart policy returns a service to an apparently healthy state while an unresolved underlying resource condition prevents the service from processing requests
- **Safety Automation** — DynaCov's product category term for the discipline of validating automated remediation systems before they are trusted in production