ChatGPT

Weryfikacja naukowa tezy i planu testów w kontrapunkcie do rzeczywistości

Przebieg badania i artefakty, od których zaczęto

Badanie zostało rozpoczęte od użycia włączonego konektora entity["company","GitHub","code hosting platform"] i ograniczone wyłącznie do wskazanych repozytoriów użytkownika: `DonkeyJJLove/sbom`, `DonkeyJJLove/writeups`, `DonkeyJJLove/chunk-chunk`, `DonkeyJJLove/swarm`, `DonkeyJJLove/ai_platform`, `DonkeyJJLove/HA2D`. Wykonane operacje na konektorze obejmowały:

- **get_repo** dla każdego repo (weryfikacja dostępności i gałęzi domyślnej: `sbom` ma `main`, pozostałe mają `master`).
- **fetch** (pobranie treści kluczowych plików; selektywnie, pod hipotezy i testy).

Ograniczenie krytyczne dla rygoru cytowania: w tym środowisku konektor GitHub dostarcza treści plików, ale **nie dostarcza stabilnych markerów line-by-line w formacie** `filecite`. Dlatego w raporcie: (a) wskazuję konkretne ścieżki i fragmenty artefaktów jako dowód mechanizmu, (b) wszystkie twierdzenia „metodologiczne" i „o rzeczywistości zewnętrznej" podpieram cytowaniami webowymi, (c) wnioski o skuteczności ekonomicznej klasyfikuję jako *warunkowe* do czasu wykonania testów na danych.

Zebrane artefakty (rdzeń dowodowy z repo):

- `sbom/README.md` – definicja „pętli sterowania" i roli SBOM jako sensora stanu, oraz jawna architektura „pomiar → próg → decyzja".
- `sbom/lab/jenkins/pipeline_one.pipeline` – formalna implementacja: generacja `sbom`, `scan`, `snapshot`, `delta`, `gate`, ingest do event store (Elasticsearch).
- `sbom/docs/03_JENKINS_PIPELINE.md` – operacyjny samouczek, w tym lista eventów i wyjaśnienie "dlaczego delta ma sens tylko sekwencyjnie".
- `swarm/infrastructure/istio/policies/circuit-breaker.yaml` – outlier detection jako mechanizm anty-kaskadowy.
- `swarm/infrastructure/istio/policies/rate-limit.yaml` – rate limiting z `failure_mode_deny: true` jako fail-closed.
- `swarm/infrastructure/istio/policies/README.md` – katalog polityk: circuit breakers, rate limiting, fault injection, retries/timeouts, mirror traffic (czyli kompletna paleta eksperymentów systemowych).
- `chunk-chunk/hmk9d_protocol.yaml` – formalizacja kontraktu `H(s)=g(F(s))`, ryzyka `R(F,g)` oraz operatorów progowych (bramki/"Próg–Przejście") i guardów energetycznych.
- `HA2D/context_protocol.md` – protokół pamięci kontekstu (uuid, timestamp, payload, sha256), operacje STORE/RETRIEVE oraz weryfikacja integralności.
- `ai_platform/platform.md` – mapowanie woluminu `/QV9D` na strukturę artefaktów + jawne miejsce na deterministyczny identyfikator „latarni" (istotne ryzyko konstruktu).
- `writeups/protokoly_kontekstu_chunk-chunk_facebook_case.md` – metodologia dowodu warunkowego + falsyfikacja (overfitting, dryf, błędna baza, out-of-sample) przeniesiona w język formalny.

Pytania badawcze niezbędne do rozstrzygnięcia i ich operacjonalizacja

W najostrzejszym reżimie naukowym teza jest „dopuszczalna” tylko wtedy, gdy jest falsyfikowalna (w sensie entity["people","Karl Popper","philosopher of science"]: istnieją obserwacje, które mogłyby ją obalić) oraz gdy posiada protokół „zdań bazowych” (co w praktyce oznacza: jakie zdarzenia, jakie logi, jakie miary, jaki łańcuch dowodowy). [1]

Poniższe pytania są minimalnym zestawem 3–6, bez których nie da się naukowo ocenić „amortyzacji kolapsu Cloud+AI”:

Czy metering przypisuje koszt do jednostki pracy, a nie do „seata”?
Operacjonalizacja: definiujemy **Work Unit (WU)** jako atom pracy (np. request-task-loop lub agent-step), a następnie mierzymy rozkład `Cost/WU` (p50/p95/p99) oraz jego wrażliwość na wolumen (elasticity) w czasie. Krytyczne, bo praktyka entity["organization","FinOps Foundation","linux foundation project"] wskazuje na przewagę metryk jednostkowych (unit economics) nad metrykami agregatowymi dla decyzji koszt-jakość-szybkość i „ownership pushed to the edge”. [2]

Czy gating (progi) redukuje kaskady, czy tylko przesuwa ból?
Operacjonalizacja: definiujemy „kaskadowość” jako prawdopodobieństwo propagacji zdarzenia awaryjnego do downstream (np. 5xx w usłudze A → 5xx w B → retry-storm), z metrykami MTTR, p99 latency, error budget burn, oraz udział “kontrolowanej degradacji” (np. 429) vs „niekontrolowanej” (5xx). Krytyczne, bo w sieciach progowych zachodzi zjawisko rzadkich, ale wielkich kaskad wyzwalanych małym impulsem. [3]

Czy kontekst jest replikowalny i wersjonowalny w sposób dowodowy?
Operacjonalizacja: dla każdej WU i runu mamy (a) deterministyczną serializację kontekstu, (b) integralność kryptograficzną (hash), (c) możliwość replay w izolacji, (d) metrykę „niewyjaśnionej wariancji” wyników po kontrolowaniu kontekstu. Krytyczne, bo bez tego testy systemowe zamieniają się w narrację post-hoc. Popperowski problem: brak wiarygodnych “basic statements” uniemożliwia rzeczywistą falsyfikację. [1]

Czy mechanizm sterowania nie ulega prawu Goodharta?
Operacjonalizacja: rozdzielamy metryki na endpoints pierwotne i wtórne; sprawdzamy, czy optymalizacja KPI nie pogarsza rzeczywistych outcome’ów (np. „mniej 5xx” kosztem „więcej 429” bez poprawy kosztu/WU). Krytyczne, bo “When a measure becomes a target…” opisuje ryzyko degeneracji miary przy sterowaniu. [4]

Czy teza jest zgodna z constraintami świata (energia, lead time, bottlenecki)?
Operacjonalizacja: scenariusze kosztowe i energetyczne, które wiążą `Cost/WU` z realnymi ograniczeniami podaży compute/energii. Krytyczne, bo entity["organization","International Energy Agency","intergovernmental org"] wskazuje na napięcie: data center może stanąć w 2–3 lata, ale system energetyczny ma dłuższe cykle inwestycyjne; równocześnie globalne zużycie energii przez data center ma rosnąć do ok. 945 TWh w 2030 w scenariuszu bazowym, co tworzy twardy “bat” na ekonomię AI. [5]

Ocena jakości wniosku naukowego

Status epistemiczny tezy na dziś

W świetle dostępnych artefaktów repozytoryjnych teza ma wysoką **spójność mechanistyczną**, ale jeszcze nie ma statusu „empirycznie potwierdzonej" w sensie naukowym, ponieważ:

- artefakty dowodzą istnienia mechanizmu (metering, delta, gate, progi runtime),
- lecz nie dostarczają jeszcze wyników z serii eksperymentów na danych produkcyjnych, które rozstrzygają alternatywne wyjaśnienia (confounders).

W klasyfikacji „najostrzejszej" to jest etap: **model + implementacja instrumentacji + projekt badań**, gotowy do falsyfikacji, lecz nadal wymagający danych.

Najmocniejsze elementy tezy w świetle repo

Mechanizm „pomiar → próg → decyzja" jest w repo ujęty nie jako metafora, tylko jako proces CI/CD i runtime:

- `sbom` realizuje event-sourcing: generuje zdarzenia typu `sbom`, `scan`, `delta`, `gate` i zapisuje je do pamięci zdarzeń (Elasticsearch), a „delta" jest liczona jako różnica snapshotu z poprzednim snapshotem pobranym po kluczach tożsamości `aid.*`. To daje strukturę "basic statements" w praktyce: jednoznaczne, odtwarzalne zdarzenia pomiaru i decyzji. [1]
- `swarm` zawiera infrastrukturalne progi kaskadowe: w `rate-limit.yaml` jawnie ustawiono `failure_mode_deny: true`, co jest zgodne z dokumentacją Envoy: przy błędzie usługi limitującej i `failure_mode_deny=true` system zwraca błąd zamiast „przepuszczać" ruch (fail-closed), co jest klasycznym mechanizmem zatrzymywania eskalacji. [6]
- `circuit-breaker.yaml` ustawia outlier detection (ejection po kolejnych 5xx, interwał oceny, czas ejection), co odpowiada parametrom opisanym w dokumentacjach outlier detection (interval, baseEjectionTime, maxEjectionPercent). [7]
- `chunk-chunk/hmk9d_protocol.yaml` formalizuje "energię kroku" `E(Δ)` i "bramki" (Próg–Przejście) oraz wprost zawiera guard `E <= 0.8` w szablonie procesu minimalnego. To jest już język sterowania i ograniczeń, a nie impresji.

Wniosek naukowy, jaki wolno wyciągnąć na obecnym etapie: repozytoria **konstruują sterowalny układ pomiarowy** (instrumentacja + progi), który jest logicznie zgodny z teorią progów i kaskad (Granovetter/Watt), ale amortyzacja ekonomiczna wymaga jeszcze empirycznej kalibracji. [8]

Najbardziej "ostra" krytyka tezy, która musi być potraktowana serio

Najostrzejszy kontrargument naukowy brzmi: „to, że system ma progi, nie dowodzi, że amortyzuje ekonomię – może tylko przesuwać problem na inny licznik (np. z awarii na odrzucenia, z latency na koszt)."

To jest krytyka *construct validity*: mierzysz coś stabilnie, ale nie wiesz, czy mierzysz właściwy konstrukt „amortyzacji ekonomicznej". Shadish–Cook–Campbell opisują właśnie to rozróżnienie między internal/construct/external validity. [9]

Dlatego Twoja teza, aby przejść w klasę „twardą", musi spełnić warunek: **progi muszą być sterowane w jednostkach, które są nośnikiem kosztu** (WU), a nie tylko w jednostkach ruchu (RPS) czy symptomów

(5xx). To jest to miejsce, gdzie FinOps naciska na “unit economic and value-based metrics” jako lepsze od agregatów. [10]

Ocena wartości naukowej planu testów i poprawki podnoszące klasę dowodową

W tej sekcji oceniam testy w czterech wymiarach: statistical conclusion validity, internal validity, construct validity, external validity. To jest klasyczny aparat (Shadish–Cook–Campbell). [9]

Testy jednostkowe

Wartość naukowa: bardzo wysoka internal validity i powtarzalność, o ile testują deterministyczne własności (hash kontekstu, deterministyczność serializacji, poprawność delty jako funkcji snapshotów).

Ryzyko: niska external validity – test jednostkowy nie dowodzi, że system wytrzyma agentowe pętle, tylko że inwariant jest poprawnie policzony.

Poprawki (aby wzmocnić konstrukty):
- wymusić testy deterministyczności na wielu środowiskach runtime (różne wersje bibliotek, locale, kolejność pól JSON), bo najczęstsza porażka replikowalności wynika z “prawie deterministycznych” artefaktów (sortowanie, floating point, czas).

Testy integracyjne (E2E)

Wartość naukowa: rośnie, bo obejmują integracje (miejsce narodzin kaskad). Repo `sbom/docs/03_JENKINS_PIPELINE.md` jest szczególnie ważne metodologicznie, bo wskazuje praktyczne źródła driftu (np. docker.sock permissions, różnice shell, pluginy), czyli jawnie adresuje internal validity: czy test bada mechanizm, czy błąd środowiskowy.

Ryzyko: confounding przez infrastrukturę (cache, retry policy, timeout). To musi być kontrolowane “kontraktem eksperymentu”.

Poprawki:
- wprowadzić *prerejestrację scenariusza E2E*: jakie parametry, jakie progi, jakie endpointy i jakie miary są „primary endpoints”. To wprost redukuje elastyczność analityczną, przed którą ostrzega zarówno ASA, jak i metanauka. [11]

Testy obciążeniowe

Wartość naukowa: krytyczna dla Twojej tezy, bo kaskady są rzadkie, wielkie i progowe (Watts). [12]

Ryzyko: jeśli workload jest zbyt “gładki”, nie zobaczysz reżimu kaskad; jeśli jest nienaturalny, uzyskasz wynik nieprzenaszalny (external validity).

Poprawki:
- workload jako mieszanina (burst + sustained + fault injection + retry amplification), zgodnie z tym, że układ jest wrażliwy na progi i ogon rozkładu;
- metryki raportowane jako rozkłady (p50/p95/p99) i histogramy, a nie średnie (Fortio jest narzędziem zaprojektowanym do QPS + histogramów i percentyli, co dobrze pasuje do Twoich potrzeb). [13]

Testy ekonomiczne i Monte Carlo

Wartość naukowa: warunkowa, bo Monte Carlo nie „dowodzi świata", tylko konsekwencje założeń. Aby mieć klasę naukową, musi być kalibrowany do danych i wraz z analizą wrażliwości.

Ryzyko: p-hacking metryk i "efekt estetyczny symulacji". Ioannidis opisuje, że przy wielu stopniach swobody (definicje, outcome'y, metody analizy) rośnie ryzyko fałszywych wniosków. [14]

Poprawki:
- jawna kalibracja rozkładów wejściowych (koszt/GPUh, koszt tokena, koszt egress, koszt incydentu) do billingów i logów;
- prerejestracja: które hipotezy i które endpointy są "confirmatory", a które "exploratory", zgodnie z zasadami ASA (pełne raportowanie, unikanie decydowania wyłącznie progiem p-value). [15]

Metryki, instrumentacja i procedury falsyfikacji

Metryki i ich pułapki

Metryki techniczne powinny obejmować: latency p50/p95/p99, error classes (5xx/429/timeouts), retry rate, saturation (CPU/RAM), outlier ejections, oraz "time to stable regime". Metryki te mają sens w teorii progów, bo to właśnie ogon (p99) ujawnia kaskady.

Metryki ekonomiczne muszą być jednostkowe: `Cost/WU`, `EV(straty na incydentach)`, a dopiero potem agregacje per dzień/klient. To jest zgodne z praktyką FinOps, gdzie rekomenduje się unit economics i "ownership at the edge". [2]

Instrumentacja: bez korelacji logów, metryk i trace'ów do identyfikatora WU nie da się eksperymentalnie przypisać kosztu do pracy. Specyfikacja logów OpenTelemetry wprost opisuje, że brak standardowego przenoszenia trace context powoduje rozłączne logi, a do pełnej korelacji potrzebne są trace/span IDs i baggage. [16]

Pułapki: - heavy tails: średnie często kłamią w reżimach rzadkich kaskad (Watts). [12]
- Goodhart: jeśli KPI staje się celem, przestaje mierzyć (Goodhart's law). [4]
- wielokrotne porównania i elastyczna analiza: pogłębiają ryzyko pozornych efektów (ASA, Ioannidis). [11]

Procedury falsyfikacji i testy statystyczne

W reżimie „ostrym" falsyfikacja musi mieć progi i warunki brzegowe. Proponuję (jako minimalny standard):

Falsyfikacja H1 (progi runtime redukują kaskady):
H1 obalona, jeśli po wdrożeniu (lub w A/B) kaskadowość nie spada, a p99 latency lub MTTR rosną, albo jeśli spadek 5xx następuje kosztem istotnego wzrostu 429 bez poprawy kosztu/WU.

Test:
- dla zdarzeń binarnych (kaskada vs brak): test różnicy proporcji / logit;
- dla p99/p95: bootstrap, quantile regression;
- jako projekt: interrupted time series (ITS) lub DiD zamiast "before/after", zgodnie z klasycznymi standardami quasi-eksperymentów. [9]

Falsyfikacja H2 (kontekst replikowalny):
H2 obalona, jeśli replay przy tym samym kontekście daje rozbieżność wyników większą niż zarejestrowany błąd pomiaru, albo jeśli hash/integralność nie gwarantują jednoznacznej serializacji.

Falsyfikacja H3 (metering w WU amortyzuje ekonomię):
H3 obalona, jeśli `Cost/WU` p95/p99 nie spada (lub rośnie) po wdrożeniu progów i pomiaru, mimo spadku awaryjności; albo jeśli rozkład kosztu pozostaje tak samo ciężkoogonowy (brak "obcięcia ogona").

Poziomy istotności i próby:
W reżimie inżynierskim, zamiast fetyszyzować p<0.05, zalecam: (a) raportowanie wielkości efektu + niepewności, (b) predefinicję minimalnego efektu praktycznego (MDE). To jest zgodne z ASA: p-value nie jest miarą wielkości efektu, a wnioski nie powinny opierać się wyłącznie na progu. [15]

Plan wdrożenia eksperymentów A/B, bezpieczeństwo i harmonogram

Eksperymenty w środowisku z minimalizacją ryzyka

Procedura wdrożeniowa (bez "ruletki na produkcji"):

- Mirror traffic: polityki `mirror traffic` są już ujęte jako komponent w dokumentacji polityk service mesh; mirror pozwala na porównanie bez wpływu na użytkowników (wysoka internal validity, niski koszt ryzyka).
- Feature flags: przełączalne progi (rate limit, outlier detection, retry). Stop conditions: automatyczny rollback, jeśli przekroczony error budget lub p99 latency.
- Fail-closed dla limitowania: konfiguracja `failure_mode_deny: true` odpowiada intencji "nie oddawaj kontroli kaskadzie"; dokumentacja Envoy opisuje, że w razie błędu usługi limitującej i `failure_mode_deny=true` żądania nie przechodzą, co zapobiega "przepaleniu" systemu w razie awarii kontrolera. [6]

Harmonogram prac i priorytety

Priorytetem jest osiągnięcie wiarygodności konstruktu (WU + telemetria + prerejestracja), dopiero potem „duże testy".

- Tydzień pierwszy: definicja WU, instrumentacja OTel (trace/span/baggage) i spójny format eventów. [16]
- Tydzień drugi: uruchomienie pipeline pomiarowego (event store) w stylu `sbom`: snapshot, delta, gate, oraz baseline runy bez progów.
- Tydzień trzeci: testy obciążeniowe (Fortio) + fault injection + modelowanie burst/sustained, zbieranie rozkładów p95/p99. [17]
- Tydzień czwarty: quasi-eksperyment ITS/DiD, pierwsza kalibracja modeli ekonomicznych i analiza wrażliwości.

Wymagania środz. i budżet (rzędy wielkości):
- CPU-heavy: większość testów progu i telemetrii; GPU nie jest konieczne, jeśli testujesz "sterowanie kosztem" na proxy miarach (np. per-request compute proxy).
- Jeśli test obejmuje realną inferencję LLM, koszty to: GPU-hours + egress + log storage; przy heavy tails kluczowe są długie runy, co rośnie kosztowo. Z perspektywy makro, IEA wskazuje, że energia/compute

jest twardym bottleneckiem i będzie skłaniać rynek do repricingu, więc eksperyment ekonomiczny musi uwzględniać scenariusze energetyczne i bottlenecki. [5]

Wnioski falsyfikowalne i rekomendacje praktyczne

Wnioski, które **wolno** sformułować już teraz (na bazie kodu/specyfikacji), nie udając empirycznej pewności:

- Repozytoria implementują spójny wzorzec naukowo-testowalny: instrumentacja → delta → gate → progi runtime. To jest dobra baza do falsyfikacji, bo powstają zdarzenia obserwacyjne i decyzje progowe (w sensie Poppera: możliwość sformułowania zdań bazowych). [1]
- Plan testów, jeśli zostanie wykonany z preregistracją i kontrolą wielokrotnych porównań, może osiągnąć wysoką klasę dowodową; bez tego ryzykuje „metrologiczną iluzję" (ASA, Ioannidis). [11]
- Mechanizm progów jest zgodny z teorią kaskad i progów zachowań zbiorowych; to nie dowód sukcesu, ale silna zgodność mechanistyczna, która uzasadnia testy obciążeniowe i quasi-eksperymenty. [3]
- Kontrapunkt do rzeczywistości (energia i lead time) wzmacnia sens podejścia "unit economics + progi", bo koszty compute nie skaluje się w nieskończoność bez napięć infrastrukturalnych; IEA wprost podaje scenariusze i ryzyka bottlenecków oraz prognozę wzrostu energii data center do ok. 945 TWh w 2030 w scenariuszu bazowym. [5]

Rekomendacje praktyczne „najwyższej dźwigni" (konieczne, aby dowód był naukowy, a nie retoryczny):

- Zdefiniować i wdrożyć identyfikator WU oraz korelację telemetryki zgodnie z OTel (trace/span/baggage) – bez tego nie da się rozstrzygnąć H3. [16]
- Preregistrować primary endpoints (kaskadowość, MTTR, p99, Cost/WU p95/p99) i analizę; raportować wielkość efektu i niepewność, nie tylko "istotność". [15]
- Zaprojektować quasi-eksperymenty ITS/DiD i unikać naiwnego before/after (ramy Shadish–Cook–Campbell). [9]
- Traktować Goodhart jako warunek brzegowy: nie jeden KPI, tylko pakiet metryk + audyt „czy nie gramy w metrykę". [4]
- W warstwie runtime utrzymać mechanizmy fail-closed dla limitowania (failure_mode_deny), bo to jest kluczowy element „amortyzacji kaskady". [6]

---

1 https://plato.stanford.edu/entries/popper
https://plato.stanford.edu/entries/popper

2 10 https://www.finops.org/framework/principles/
https://www.finops.org/framework/principles/

3 12 https://pubmed.ncbi.nlm.nih.gov/16578874/
https://pubmed.ncbi.nlm.nih.gov/16578874/

4 https://en.wikipedia.org/wiki/Goodhart%27s_law
https://en.wikipedia.org/wiki/Goodhart%27s_law

5 https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai
https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai

6 https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/rate_limit_filter
https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/rate_limit_filter

7 https://docs.rs/envoy-prost-tonic/latest/envoy_prost_tonic/envoy/config/cluster/v3/struct.OutlierDetection.html
https://docs.rs/envoy-prost-tonic/latest/envoy_prost_tonic/envoy/config/cluster/v3/struct.OutlierDetection.html

8 https://sociology.stanford.edu/publications/threshold-models-collective-behavior
https://sociology.stanford.edu/publications/threshold-models-collective-behavior

9 https://lawcat.berkeley.edu/record/365766
https://lawcat.berkeley.edu/record/365766

11 15 https://www.amstat.org/asa/files/pdfs/p-valuestatement.pdf
https://www.amstat.org/asa/files/pdfs/p-valuestatement.pdf

13 17 https://fortio.org/
https://fortio.org/

14 https://www.metajournal.com/pubmed/16060722
https://www.metajournal.com/pubmed/16060722

16 https://opentelemetry.io/docs/reference/specification/logs/
https://opentelemetry.io/docs/reference/specification/logs/